Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2022-05. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres . ISSN 1587–4451.

Fast algorithms for sparsity matroids and the global rigidity augmentation problem

Csaba Király and András Mihálykó

January 21, 2023

Fast algorithms for sparsity matroids and the global rigidity augmentation problem

Csaba Király[⋆] and András Mihálykó[⋆⋆]

Abstract

A (k, ℓ)-sparsity matroid is a matroid defined on the edge set of a graph $G = (V, E)$ with sparsity conditions on the edge set involving the values k and ℓ corresponding to the matroid independence. We show that for a given pair of k and ℓ the components of the (k, ℓ)-sparsity matroid can be calculated in $O(|V|^2)$ time.

Components of a (k, ℓ)-sparsity matroid appear in several applications connected to rigidity theory. For example, for generic frameworks in the plane (with Euclidean or non-Euclidean norm), or on a cylinder (with Euclidean norm), the global rigidity of a framework is characterized by using the connectivity of a (k, ℓ)-sparsity matroid of its graph (for a given pair of k and ℓ) along with some vertex-connectivity properties. We use the above mentioned algorithm to check global rigidity of such frameworks in $O(|V|^2)$ time. We also show how the algorithm can be used to solve the global ridigity augmentation problem in $O(|V|^2)$ time. In the latter problem the input is a graph which is generically rigid (in the considered space) and our goal is to find a minimum edge set which makes this graph generically globally rigid.

1 Introduction

In this paper we consider problems on (k, ℓ)-sparsity matroids from an algorithmic point of view. Many of these matroids are related to rigidity theory. Moreover, some of the questions considered in this paper has their motivation rooted in rigidity theory.

In rigidity theory a d-dimensional **framework** is a pair (G, p) where $G = (V, E)$ is a graph and $p : V \to S \subseteq \mathbb{F}^d$ is a map of the vertices of G to a given subset of a d-dimensional normed vector space $(\mathbb{F}^d, || \cdot ||)$. (G, p) and (G, q) are *equivalent* if $||p(u) - p(v)|| = ||q(u) - q(v)||$ for every $uv \in E$, and they are *congruent*, if $||p(u) - p(v)|| = ||q(u) - q(v)||$ holds for every vertex pair $u, v \in V$. A framework (G, p) is called **globally rigid** in S, if each equivalent framework (G, q) (in S) is also congruent with

[⋆]Department of Operations Research, ELTE Eötvös Loránd University and the ELKH-ELTE Egerváry Research Group on Combinatorial Optimization, Eötvös Loránd Research Network (ELKH), Pázmány Péter sétány 1/C, Budapest, 1117, Hungary. E-mail: `cskiraly@cs.elte.hu`

[⋆⋆]Department of Operations Research, ELTE Eötvös Loránd University, Pázmány Péter sétány 1/C, Budapest, 1117, Hungary. E-mail: `mihalyko@cs.elte.hu` it. (G,p) is **rigid**, when the above condition only holds for realizations $q : V \to S$ for which $||p(v) - q(v)|| < \varepsilon$ for some $\varepsilon > 0$, that is, when the framework has no continuous deformation. Due to their connection to the (k,ℓ)-sparsity matroids, in this paper the following types of frameworks will arise: *frameworks in the Euclidean plane* (when $\mathbb{F} = \mathbb{R}$, $d = 2$, the norm is the Euclidean norm, and $S = \mathbb{R}^2$), *frameworks in a non-Euclidean plane* (when $\mathbb{F} = \mathbb{R}$, $d = 2$, the norm is a non-Euclidean analytic norm, and $S = \mathbb{R}^2$), *frameworks on a cylinder* (when $\mathbb{F} = \mathbb{R}$, $d = 3$, the norm is the Euclidean norm, and $S = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 = 1\}$). In these cases, the rigidity and global rigidity of a framework only depends on the underlying graph if the realization is *generic*, that is, when the only algebraic dependency of the coordinates of the points over the rationals arise from the definition of S [1, 4, 14, 21, 27, 33]. Thus in this cases rigidity and global rigidity is in fact a graph property and we can talk about the rigidity and global rigidity of a graph.

In the previously mentioned cases, the rigidity and global rigidity of a graph can be characterized by using some sparsity properties of the graph. For the rest of this paper let k always denote a positive integer and ℓ an integer such that $0 < \ell < 2k$. A (multi)graph $G = (V,E)$ is called $(\boldsymbol{k},\boldsymbol{\ell})$**-sparse** if $i_G(X) \leq k|X| - \ell$ for all $X \subseteq V$ with $k|X| - \ell \geq 0$, where $\boldsymbol{i_G(X)}$ denotes the number of edges of G induced by X. A (k,ℓ)-sparse graph is called $(\boldsymbol{k},\boldsymbol{\ell})$**-tight** if $|E| = k|V| - \ell$. Due to its connections and extensive usage in rigidity theory, a graph G is called $(\boldsymbol{k},\boldsymbol{\ell})$**-rigid** if G has a (k,ℓ)-tight spanning subgraph. It is known that the edge sets of (k,ℓ)-sparse subgraphs of a given graph G form the independent sets of a matroid on the edge set, callet the $(\boldsymbol{k},\boldsymbol{\ell})$**-sparsity matroid** or **count matroid** of G (see [31], [38, Appendix A]). We call an edge e of $G = (V,E)$ a $(\boldsymbol{k},\boldsymbol{\ell})$**-redundant edge** if a (k,ℓ)-rigid subgraph of $G - e$ spans both endpoints of e, or equivalently if E and $E - e$ have the same rank in the (k,ℓ)-sparsity matroid of G. A (k,ℓ)-rigid graph G is a $(\boldsymbol{k},\boldsymbol{\ell})$**-redundant graph** if each edge of G is (k,ℓ)-redundant.

Sparsity conditions occur in connection with several graph properties ranging from edge-connectivity (e.g. $(1,1)$-rigid graphs are exactly the connected ones), through tree-connectivity [32] to the maximum size of a matching [9, Section 13.5]. However, we are most interested in the applications regarding rigidity theory:

- A graph is rigid in the Euclidean plane if and only if it is $(2,3)$-rigid. [28, 34]
- A simple graph on at least 4 vertices is globally rigid in the Euclidean plane if and only if it is $(2,3)$-redundant and 3-connected. [20]
- A simple graph is rigid in a non-Euclidean plane if and only if it is $(2,2)$-rigid. [27]
- A simple graph on at least 5 vertices is globally rigid in a non-Euclidean plane if and only if it is $(2,2)$-redundant and 2-connected. [4]
- A simple graph is rigid on the cylinder if and only if it is $(2,2)$-rigid. [33]
- A simple graph on at least 5 vertices is globally rigid on the cylinder if and only if it is $(2,2)$-redundant and 2-connected. [21]

It is known that the (k,ℓ)-rigidity of a graph $G = (V,E)$ can be tested in $O(|V|^2)$ time (see [3, 22, 23] for the case where $(k,\ell) = (2,3)$, and [29] for general k and ℓ). Hence the above metioned rigidity properties can be tested efficiently. We note that in the above characterizations of global rigidity the condition that the graph G is (k,ℓ)-redundant (for $(k,\ell) = (2,3)$ or $(2,2)$) can be substituted by the requirement that the (k, ℓ)-sparsity matroid of G is connected (see definition later). In Section 3 we show how the components of the (k, ℓ)-sparsity matroid of $G = (V, E)$ can be calculated in $O(|V|^2)$ time. Since the 2- or 3-connectivity of G can be tested in $O(|V| + |E|) \leq O(|V|^2)$ time [5, 15, 17, 18], this implies that the above mentioned global rigidity properties can be checked in $O(|V|^2)$ time. As another application of the algorithm given in Section 3, we consider the global rigidity augmentation problem. This problem takes a rigid graph as its input and looks for a minimum edge set which makes the graph globally rigid. The optimum of this problem was characterized in [25]. In Section 4, we summarize the results of [26, 25] which we use for the algorithm. In Section 5, we show how the global rigidity augmentation problem can be solved in $O(|V|^2)$ time.

2 Preliminaries

While graphs are the main focus of this paper, (k, ℓ)-sparsity and (k, ℓ)-tightness also appear several times in context of hypergraphs. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called **(k, ℓ)-sparse** if $i_{\mathcal{H}}(X) \leq k|X| - \ell$ holds for all $X \subseteq V$ with $k|X| - \ell \geq 0$, where $\boldsymbol{i_{\mathcal{H}}(X)}$ denotes the number of hyperedges of $\mathcal{H}$ induced by X. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called **(k, ℓ)-tight** if it is sparse and $|\mathcal{E}| = k|V| - \ell$. We call a hypergraph **(k, ℓ)-rigid** if it contains a spanning (k, ℓ)-tight subhypergraph. Similarly to the graphic case the hyperedge sets of the (k, ℓ)-sparse subhypergraphs of a given hypergraph correspond to the independent sets of the so-called **(k, ℓ)-sparsity matroids** or **count matroids** (see [9, Section 13.5], [31], and [38, Appendix A]). We can also define redundancy of hyperedges and **(k, ℓ)-redundant** hypergraphs as a direct generalization of (k, ℓ)-redundant edges and graphs.

It is easy to see from the definition that a (k, ℓ)-tight subhypergraph of a (k, ℓ)-sparse hypergraph is always an induced subhypergraph. The following well-known statements (which can be found in e.g. [26]) show other important properties of the (k, ℓ)-sparisity.

Lemma 2.1. *Let $\mathcal{H} = (V, \mathcal{E})$ be a (k, ℓ)-sparse hypergraph with $0 < \ell < 2k$, and let $\mathcal{T}_1 = (V_1, \mathcal{E}_1)$ and $\mathcal{T}_2 = (V_2, \mathcal{E}_2)$ be (k, ℓ)-tight subhypergraphs of $\mathcal{H}$. If $k|V_1 \cap V_2| \geq \ell$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ and $\mathcal{T}_1 \cap \mathcal{T}_2$ are (k, ℓ)-tight hypergraphs, and $d_{\mathcal{H}}(V_1, V_2) = 0$.* □

Lemma 2.2. *Let $G = (V, E)$ be a (k, ℓ)-tight graph on at least 3 vertices. Then $d(v) \geq k$ for any $v \in V$. Moreover, there exists an $i \in V$ such that $d(i) \leq 2k - 1$.* □

Let us list now some basic definitions and properties concerning the sparsity matroid without proofs, as they play important roles in this paper. We refer to [9, 24, 38] for more details.

First, a subhypergraph, the edges set of which forms a circuit in the (k, ℓ)-sparsity matroid is called a **(k, ℓ)-M-circuit**. In other words, a subhypergraph C is a (k, ℓ)-M-circuit if it is not (k, ℓ)-sparse and $C - e$ is (k, ℓ)-sparse for every single hyperedge e of C. In particular, if $\mathcal{H}$ is (k, ℓ)-sparse, $e = ij$ is a new (graph) edge, and there exists a (k, ℓ)-tight subhypergraph of $\mathcal{H}$ with vertex set V' so that $i, j \in V'$, then $\mathcal{H} + e$ has a unique (k,ℓ)-M-circuit, denoted by $\boldsymbol{\mathcal{C}_{\mathcal{H}}(ij)}$ or $\boldsymbol{\mathcal{C}_{\mathcal{H}}(e)}$. This circuit contains e. In this case, $(V(\mathcal{C}_{\mathcal{H}}(e)), \mathcal{E}(\mathcal{C}_{\mathcal{H}}(e)) - e)$ forms a (k,ℓ)-tight subhypergraph of $\mathcal{H}$, that we call $\boldsymbol{\mathcal{T}_{\mathcal{H}}(e)}$ or $\boldsymbol{\mathcal{T}_{\mathcal{H}}(ij)}$. For the sake of convenience, we do not distinguish a hypergraph from its edge set, that is, $\mathcal{T}_{\mathcal{H}}(e) = \mathcal{E}(\mathcal{C}_{\mathcal{H}}(e)) - e$, while we denote the vertex set of $\mathcal{T}_{\mathcal{H}}(ij)$ by $\boldsymbol{V_{\mathcal{H}}(ij)} = V(\mathcal{C}_{\mathcal{H}}(e))$. For every hyperedge e' of $\mathcal{C}_{\mathcal{H}}(ij)$, $\mathcal{H}' = \mathcal{H} + ij - e'$ is also (k,ℓ)-sparse and the unique (k,ℓ)-circuit of $\mathcal{H}' + e'$ is again $\mathcal{C}_{\mathcal{H}}(ij)$. Moreover, if $e'' \notin \mathcal{C}_{\mathcal{H}}(ij)$, then $\mathcal{H}' + ij - e''$ is not (k,ℓ)-sparse. For the sake of simplicity, when the hypergraph $\mathcal{H}$ is clear from the context, we will omit the subscript $\mathcal{H}$ from all the notation throughout this entire paper.

The following well-known statement (which can be deduced from for example [26, Lemma 2.1]) highlights the most important property of $\mathcal{T}(ij)$.

Lemma 2.3. *[26] Let $\mathcal{H} = (V, \mathcal{E})$ be a (k,ℓ)-sparse hypergraph with $0 < \ell < 2k$ and let $i, j \in V$. Assume that $\mathcal{H} + ij$ is not (k,ℓ)-sparse and there exists a (k,ℓ)-tight subhypergraph of $\mathcal{H}$, denoted by $\mathcal{H}' = (V', \mathcal{E}')$, so that $i, j \in V'$. Then $\mathcal{T}(ij) \subseteq \mathcal{H}'$ and $\mathcal{T}(ij) = \bigcap\{\mathcal{T}_h : \mathcal{T}_h$ is a (k,ℓ)-tight subhypergraph of $\mathcal{H}$ including i and $j\}$.* □

It is also well-known (and is presented for the (k,ℓ)-sparsity matroids of (k,ℓ)-rigid graphs in [25]) that an equivalence relation can be defined on the ground set S of an arbitrary matroid $\mathcal{M}$ (by using the circuit axioms of a matroid), as follows. Two elements $x, y \in S$ are equivalent if there exists a circuit C of $\mathcal{M}$, such that $x, y \in C$. The equivalence classes of this matroid are called *components* of $\mathcal{M}$. A component of the (k,ℓ)-sparsity matroid is called a $\boldsymbol{(k,\ell)}$**-M-component**. Note that if an edge e of G is not (k,ℓ)-redundant, then the singleton $\{e\}$ is a (k,ℓ)-M-component of G and it is called a **trivial** (k,ℓ)-M-component of G. If G consists of only one (k,ℓ)-M-component, then G is called $\boldsymbol{(k,\ell)}$**-M-connected**. The following lemma from [11] shows that non-trivial (k,ℓ)-M-components are (k,ℓ)-rigid induced subgraphs. (We note here that the (k,ℓ)-M-components do *not* coincide with the (k,ℓ)-rigid components of the graph used in [30].)

Lemma 2.4. *[11, Lemma 2.4] Let G' be a non-trivial (k,ℓ)-M-component of the graph G. Then G' is an induced subgraph and is (k,ℓ)-rigid.* □

Given a graph $G = (V, E)$, the $\boldsymbol{(k,\ell)}$**-M-component hypergraph** of G is a hypergraph $\boldsymbol{\mathcal{H}_G = (V, \mathcal{E})}$, where $\mathcal{E}$ consists of the union of the non-redundant edges of G and $k|V(C)| - \ell$ parallel copies of the hyperedge formed on $V(C)$ for each non-trivial (k,ℓ)-M-component C of G. The above definition appeared in a paper by Fekete and Jordán [8] for $(k,\ell) = (2,3)$ and has been generalized for general (k,ℓ) in [25]. Note that in [25] the above definition appeared only for (k,ℓ)-rigid graphs, however, as the rigidity cannot be assured throughout the whole run of the algorithm, we need the definition for general graphs. By using Lemma 2.4, most of the results from [25] on the (k,ℓ)-M-component hypergraph can be generalized to non-rigid graphs, as well. For example [25, Lemma 3.1] can be generalized as follows.

Lemma 2.5. *Let $G = (V, E)$ be a graph and let $G^* = (V, E^*)$ be an arbitrary maximal spanning (k,ℓ)-sparse subgraph of G. Then every trivial (k,ℓ)-M-component of G is contained in E^*, and, for any non-trivial (k,ℓ)-M-component C of G, $i_{G^*}(V(C)) = k|V(C)| - \ell$.*

Proof. If C is a trivial (k,ℓ)-M-component of G, then C consists of a single non-redundant edge e of G. Remember, in this case non-redundant means that the number of edges in the maximal (k,ℓ)-sparse subgraph of G is *not* equal to the number of edges in the maximal (k,ℓ)-sparse subgraph of $G-e$. Thus e must be an edge of G^*.

Suppose now that C is non-trivial. Let $B = E^* \cap C$, that is, $i_{G^*}(V(C)) = |B|$. Now B must be a maximal (k,ℓ)-sparse subgraph of C, since otherwise we may add edges from C to G^* while still maintaining its sparsity (as the edges in C are only contained in (k,ℓ)-circuits of G consisting of the edges of C by the definition of a (k,ℓ)-M-component). This, together with Lemma 2.4 shows that $|B| = k|V(C)| - \ell$. □

Moreover, the following lemma (which implies the (k,ℓ)-sparsity of $\mathcal{H}_G$) can be proved by a straightforward generalization of the proof of [25, Lemma 3.3], using Lemma 2.5.

Lemma 2.6. *Let $G = (V,E)$ be a graph, let G^* be an arbitrary maximal spanning (k,ℓ)-sparse subgraph of G, and let $\mathcal{H}_G$ be the (k,ℓ)-M-component hypergraph of G. Then $i_{\mathcal{H}_G}(X) \le i_{G^*}(X)$ holds for each $X \subseteq V$. Furthermore, equality holds exactly when X induces either all or none of the edges of each (k,ℓ)-M-component of G.* □

Lemma 2.6 has the following important corollary.

Observation 2.7. *If G is a (k,ℓ)-rigid graph, then the (k,ℓ)-M-component hypergraph $\mathcal{H}_G$ of G is a (k,ℓ)-tight hypergraph. Furthermore, if X induces a (k,ℓ)-tight subhypergraph of $\mathcal{H}_G$, then $G[X]$ is a (k,ℓ)-rigid subgraph of G.*

2.1 Efficient algorithm for testing (k,ℓ)-rigidity

Algorithms for testing the (k,ℓ)-rigidity of a graph already exist for each pair of k and ℓ (see the works of Hendrickson, Jacobs and Thorpe [22, 23] and Berg and Jordán [3] for the case of $(k,\ell) = (2,3)$, the paper of Lee and Streinu [29] for general k and ℓ, and the extension of this latter paper for hypergraphs by Streinu and Theran [36]). The algorithm is based on the Orientation Lemma of Hakimi [16] and uses in-degree constrained orientations. For later purposes, we state the hypergraphic version of this lemma here from [10] (see also [9, Theorem 9.4.2]). We say that $\vec{\mathcal{H}}$ is an **orientation of a hypergraph** $\mathcal{H}$ if a unique head is designated for each hyperedge from its endvertices. The in-degree of a vertex v in $\vec{\mathcal{H}}$ is the number of hyperedges which has v as its head.

Lemma 2.8 (Hypergraphic orientation lemma, [10])**.** *Let $\mathcal{H} = (V,\mathcal{E})$ be a hypergraph and let $m : V \to \mathbb{Z}_+$. Then $\mathcal{H}$ has an orientation such that the in-degree of each vertex v is at most $m(v)$ if and only if $i(X) \le \sum_{x\in X} m(x)$ holds for each $X \subseteq V$.* □

All the above mentioned algorithms construct a maximal (k,ℓ)-sparse subgraph of G by considering its edges one by one and adding them to the spare subgraph greedily by using Lemma 2.8. (Note that we have the (k,ℓ)-sparsity matroid in the background and hence such a greedy method suffices.) The running time of the algorithm is $O(|V||E|)$ on graphs for fixed k and ℓ. It was observed in [3, 29] that this running time can be reduced to $O(|V|^2)$ (by using a data structure presented in [30] which also appeared independently in [2]). The main idea is to maintain the family of $\boldsymbol{(k,\ell)}$**-rigid components** (which are the maximal (k,ℓ)-tight subgraphs) of our sparse subgraph. By using this extra data, edges which are induced by a (k,ℓ)-rigid component can be skipped in constant time. Finally we note that the running time is worse in the case of hypergraphs by a factor of the maximum cardinality of a hyperedge, however, we will see in Section 3 that we can get rid of this extra factor for (k,ℓ)-M-component hypergraphs.

2.2 Connectivity

As we have mentioned in the Introduction, 2- and 3-connectivity is used in the characterization of global rigidity for the types of frameworks we are interested in. In this section we present some structural properties and efficient algorithms related to the vertex-connectivity of graphs.

A graph G on at least $k+1$ vertices is called $\boldsymbol{k}$**-(vertex-)connected** (or respectively $\boldsymbol{k}$**-edge-connected**) if $G-X$ is connected for any set X of $k-1$ vertices (edges, respectively). If a k-connected graph G is not $(k+1)$-connected, then there exists a set $X \subset V$ of cardinality k such that $G-X$ is not connected. For the sake of simplicity, we call such a set X a *mincut* of G. For a mincut X of cardinality k, let $\boldsymbol{b^k_X(G)}$ denote the number of connected components of $G-X$. Let $\boldsymbol{b^k(G)}$ denote the maximum value of $b^k_X(G)$ over all k element subsets of V. (When G is $(k+1)$-connected, let $b^k(G)=1$.) For a cut-set X of cardinality k, the connected components of $G-X$ are called $\boldsymbol{(k+1)}$**-fragments** of G. An inclusion-wise minimal $(k+1)$-fragment is called a $\boldsymbol{(k+1)}$**-end**.

Let $\boldsymbol{c_{k,\ell}}$ be an integer which is 1 if $0<\ell\le k$ and is 2 if $k<\ell<2k$. The following folklore statement is about the vertex-connectivity of a (k,ℓ)-rigid graph.

Proposition 2.9. *If $G=(V,E)$ is a (k,ℓ)-rigid graph for which $|V|\ge 3$, then G is $c_{k,\ell}$-connected.* □

This statement implies that in the applications corresponding to global rigidity we need to test the 2- or 3-connectivity of an already 1- or 2-connected graph or augment its connectivity by one. When G is connected but not 2-connected, a mincut X is a singleton, and its single element is called a **cut-vertex**. When G is 2-connected but not 3-connected, we call a mincut a **cut-pair**. We shall use the structure of the graph determined by its cut-vertices or cut-pairs, respectively.

The structure of the 2-connected components of a connected graph. A **2-connected component** of G is a maximal 2-connected subgraph of G (which could be a single edge). Any connected graph decomposes into a tree of 2-connected components called the **block-cut tree** (**BC-tree** or **2-block tree**) that we denote by $\boldsymbol{BC(G)}$. The vertices of this tree are called **nodes** (and denoted by N), and they correspond to the 2-connected components and to the the cut-vertices of the graph, thus there is a map $\boldsymbol{\varphi}: N \to 2^V$ for which $\varphi(n)$ is either a singleton formed by a cut-vertex of G or the vertex set of a 2-connected component of G for each $n \in N$. There is an edge between a node representing a 2-connected component and a node representing a cut-vertex in G, if the cut-vertex is in the 2-connected component. The 2-ends of G correspond to the leaves of $BC(G)$. The following lemma, which is based on the results of Hopcroft and Tarjan [17] and Rosenthal and Goldner [35], summarizes the properties of $BC(G)$.

Lemma 2.10. *[17, 35] Let $G = (V, E)$ be a connected graph. Then $BC(G)$ can be calculated in $O(|V| + |E|)$ time. If c is a node of $BC(G)$ corresponding to a cut-vertex $v \in V$, and C is the node set of a connected component of $BC(G) - c$ then $\varphi(C) - \varphi(v)$ is the vertex set of a 2-fragment that arise after deleting v. If $u, v \in V$ are not cut-vertices of G and u' and v' are the nodes of $BC(G)$ which correspond to the 2-connected components containing u and v, respectively, then $BC(G + uv) = BC(BC(G) + u'v')$.* □

The structure of the 3-connected components of a 2-connected graph. Observe that a cut-pair $\{u_1, v_1\}$ may separate an other cut-pair $\{u_2, v_2\}$ (that is, u_2 and v_2 are in different components of $G - \{u_1, v_1\}$). Such a pair $\{u_1, v_1\}$ is called a **weak cut-pair**. It can be proved that in this case $\{u_2, v_2\}$ is also a weak cut-pair and separates $\{u_1, v_1\}$. A cut-pair which is not a weak cut-pair, is called a **strong cut-pair**. To store the 2-cuts of a 2-connected graph G, we shall use a structure defined by Tutte [37], and Di Battista and Tamassia [5, 6]. This structure has similar properties to the BC-trees, however, its definition is more complex due to the existence of weak pairs. It is based on the 3-connected components defined by Tutte [37]. We sketch its construction and its main properties below based on [6, 15]. We assume that G is a simple 2-connected graph. (When G is not simple, we just take its maximal simple subgraph.) First we split G into smaller graphs, called *split components*, along its cut-pairs, as follows. Assume that $\{u, v\}$ decomposes the edge set of (a split component G_0 of) G into more than one equivalence classes $E_1, \ldots, E_t$, in which two edges are equivalent if they are on the same path which is internally disjoint from $\{u, v\}$, and there exists a subset $I \subset \{1, \ldots, t\}$ of indices such that the sets $E' = \bigcup_{i \in I} E_i$ and $E'' = \bigcup_{i \in (\{1,\ldots,t\} - I)} E_i$ both have cardinality at least two. We replace (the split component G_0 of) G with the two split components $G' = (V(E'), E' \cup \{uv\})$ and $G'' = (V(E''), E'' \cup \{uv\})$ where we call uv a *virtual edge*. When all possible splits have been done, we have three types of split components: 3-connected graphs, triangles, and *3-bonds* (which are two vertices connected by three edges). We say that two split components are adjacent if they share a virtual edge which is added to G at some split. Finally we merge (along with deleting the shared virtual edges) the adjacent 3-bonds to *bonds* (with at least three parallel edges) and the adjacent triangles to circles which are called *polygons*. The resulting graphs are called the **3-blocks** of G. By Tutte [37], this structure is unique for a 2-connected graph. (Note that the splitting procedure may result in different structures depending on the order of splits. These become the same during the merging steps.)

We can define a tree on the 3-blocks, called the **3-block-tree** of G, by considering 3-blocks as (tree-)nodes which we connect if they share a virtual edge which is added at some split. For the sake of simplicity, contrary to other papers, we split each edge connecting two nodes of the tree none of which is a bond node by an extra bond node of degree two. (This way each edge of the connects a bond node to a non-bond node.) By using the 3-blocks and 3-block-tree, we can store the main 3-connectivity properties of G in a data structure, called the **SPQR-tree** of G that we denote by $\boldsymbol{SPQR(G)}$. In $SPQR(G)$ we store the 3-block-tree as a tree, along with the vertex sets and types of 3-blocks corresponding to their nodes (that is, like for the BC-tree, we again have a map $\boldsymbol{\varphi'}$ from the node set of $SPQR(G)$ to 2^V), and all the edges of each polygon (which is a simple circle that may contain virtual edges and edges of G). For each vertex v of G, we also store the nodes which span v in their 3-block. The resulting structure has $O(|V|)$ nodes and it can be stored in $O(|V|)$ space.

The following lemma, which is based on the results of [5, 6, 15, 18, 19], summarizes the properties of $SPQR(G)$.

Lemma 2.11. *[5, 6, 15, 18, 19] Let $G = (V,E)$ be a 2-connected graph. Then $SPQR(G)$ can be calculated in $O(|V|+|E|)$ time and this structure can be updated in $O(|V|)$ time after the addition of a new edge. A vertex pair $\{u,v\}$ is a strong cut-pair of G if and only if $\{u,v\} = \varphi'(c)$ for a bond node c of $SPQR(G)$. In this case, for the node set of a component C of $SPQR(G)$, $\varphi'(C) - \{u,v\}$ is the vertex set of a 3-fragment that arises after deleting $\{u,v\}$. On the other hand, weak cut-pairs are exactly the non-adjacent vertex pairs in polygon-nodes. In this case the corresponding 3-fragments can be calculated by splitting the polygon (and this way the whole tree) along this cut-pair and calculating the φ' image of the resulting two components.* □

Observe that Lemma 2.11 implies that a 2-connected graph may have two types of 3-ends: the leaves of the SPQR-tree which are not polygons are all 3-ends, and the vertices of degree 2 are also all 3-ends (such vertices may be contained in polygons which are not leaves of the SPQR-tree).

Note that the main difference between the properties of BC-trees and SPQR-trees is the usage of polygon nodes. The usage of these special nodes is mandatory when we want to store the structure of all cut-pairs (and the corresponding 3-fragments) of $G = (V,E)$ in a structure of size $O(|V|)$, as there could be $O(|V|^2)$ cut-pairs in a 2-connected graph on V (for example, in a cycle). On the other hand, if G has no weak cut-pairs the structure gets much simpler. In this case, each polygon in $SPQR(G)$ is a triangle and all cut-pairs of G are represented by bond-nodes of the tree. Thus the whole structure has almost the same properties as the BC-tree. The following result of Jackson and Jordán [20] shows why the case where G has no weak cut-pairs is significant in this paper.

Lemma 2.12. *[20] If G is a $(2,3)$-rigid graph, then G contains no weak cut-pairs.* □

3 Computing the (k,ℓ)-M-components

In this section we present an efficient algorithm that computes the (k,ℓ)-M-components of a graph G by constructing its (k,ℓ)-M-component hypergraph in $O(|V|^2)$ running time. We note here that Berg and Jordán [3] gave an algorithm which calculates the $(2, 3)$-M-components of a graph $G = (V, E)$ in $O(|V|^2)$ time (see also [2] for more details). In some sense, our algorithm is a direct generalization of the algorithm of [3]. However, contrary to their approach, we shall use the (k, ℓ)-M-component hypergraph of some subgraph of G and maintain its k-in-degree constrained orientation during the algorithm.

For building the (k, ℓ)-M-component hypergraph, we shall use some similar techniques to the ones mentioned in Section 2.1. During the procedure, we collect the already considered edges to a graph $G' = (V, E')$. We maintain the (k, ℓ)-M-component hypergraph $\mathcal{H}_{G'} = (V, \mathcal{E}')$ of G', and its orientation $\vec{\mathcal{H}}_{G'}$ in which the in-degree of each vertex is at most k. For technical reasons, and to obtain some extra data in our output, we will also maintain a maximal (k, ℓ)-sparse subgraph $G'' = (V, E'')$ of G'. At the beginning of our procedure G', G'', $\mathcal{H}_{G'}$ and $\vec{\mathcal{H}}_{G'}$ are empty.

To achieve the $O(|V|^2)$ running time, we need to deploy a rather complex data structure that we describe below and illustrate in Figure 1. Besides storing each edge of $\mathcal{H}_{G'}$ corresponding to a trivial M-component of G', we store just one instance of every (undirected) hyperedge of $\mathcal{H}_{G'}$ which corresponds to a non-trivial M-component of G'. For each directed hyperedge, we maintain a pointer to the underlying undirected hyperedge and a pointer to their head. For every vertex v, we keep a doubly linked list of pointers to the directed hyperedges that v is head of. We call it the **head-list** of v.

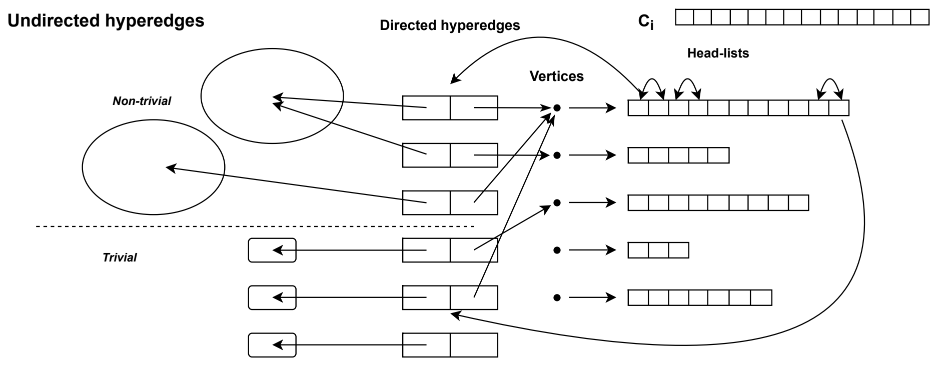

Figure 1: Illustration of the data structure used for Algorithm 3.2.

The following lemma implies that $\mathcal{H}_{G'}$ can always be stored in $O(|V|)$ space with our data structure since G' has at most $k|V| - \ell$ trivial M-components.

Lemma 3.1. *Given a graph $G = (V, E)$, let $\mathcal{C}$ denote the family of the vertex sets of its non-trivial (k, ℓ)-M-components. Then $\sum_{C \in \mathcal{C}} |C| \le 2(k|V| - \ell)$.*

Proof. By Lemma 2.6 $\mathcal{H}_G$ is a (k, ℓ)-sparse hypergraph. Let m_i denote the number of non-trivial (k, ℓ)-M-components of G with cardinality i, where $i \in \{1, \dots, |V|\}$. (Note that $m_1 = 0$ when $k \le \ell$.) Then counting the hyperedges of $\mathcal{H}_G = (V, \mathcal{E})$ corresponding to the non-trivial M-components of G we get that its number equals $\sum_{i=1}^{|V|} (ki - \ell) m_i$. Hence $\sum_{i=1}^{|V|} (ki - \ell) m_i \le |\mathcal{E}| \le k|V| - \ell$. Observe that $i \le 2(ki - \ell)$

holds for each $i \in \{2,\dots,|V|\}$ as from $2k > \ell$ we get $i \le 2ki - \ell i$. $1 \le 2(k-\ell)$ also holds when $k > \ell$ (and hence m_1 can be nonzero). Thus we get $\sum_{C\in\mathcal{C}} |C| = \sum_{i=1}^{|V|} im_i \le \sum_{i=1}^{|V|} 2(ki-\ell)m_i \le 2(k|V|-\ell)$, as claimed. □

We shall consider the edges of G one by one in a breath first manner. When we reach a new vertex i, we create a 0-1 vector c_i of length $|V|$ that indicates for every vertex v, if v is contained in a joint non-trivial (k,ℓ)-M-component with i (that is, c_i is 1 exactly at the $(\mathcal{E}' - E')$-neighbors of i). Note that c_i can be created for any particular vertex i in $O(|V|)$ time, as we can traverse all the hyperedges of $\mathcal{H}_{G'}$ in $O(|V|)$ time by Lemma 3.1.

Now, once we consider a new edge ij adjacent to the vertex i, we use the following subroutine.

Algorithm 3.2. INPUT: *A maximal (k,ℓ)-sparse subgraph G'' of a graph G', the M-component hypergraph $\mathcal{H}_{G'}$ of G', its orientation $\vec{\mathcal{H}}_{G'}$ where the in-degree $d^{in}_{\vec{\mathcal{H}}_{G'}} \le k$ for each vertex v, a vertex i (all of these stored in the data-structure mentioned above along with the vector c_i), and an edge ij.*
OUTPUT: *A maximal (k,ℓ)-sparse subgraph G'' of $G'+ij$, the M-component hypergraph $\mathcal{H}_{G'+ij}$ of $G'+ij$, and its orientation $\vec{\mathcal{H}}_{G'+ij}$ where the in-degree of each vertex is at most k.*

```
1 FoundSmallIndegree := True, $\mathcal{H} := \mathcal{H}_{G'}$, $\vec{\mathcal{H}} := \vec{\mathcal{H}}_{G'}$.
2 If $c_i(j) = 0$, then
3     While $(d^{in}_{\vec{\mathcal{H}}}(i) + d^{in}_{\vec{\mathcal{H}}}(j)) \ge 2k-\ell$ and FoundSmallIndegree = True do
          FoundSmallIndegree := False
4         Conduct a backwards DFS in $\vec{\mathcal{H}}$ from the set $\{i,j\}$, let $X \subseteq V$ be the
          set of vertices reached during the backwards DFS.
5         If $\exists v \in X - \{i,j\}$ with $d^{in}_{\vec{\mathcal{H}}}(v) < k$, then
6             Reorient a path leading from $v$ to $\{i,j\}$ to the other direction in $\vec{\mathcal{H}}$
              FoundSmallIndegree := True
7     If FoundSmallIndegree = True then
          Add $ij$ to $G''$, to $\mathcal{H}$, and also to $\vec{\mathcal{H}}$ with an orientation such that $d^{in}_{\vec{\mathcal{H}}}(i) \le$
          $k$ and also $d_{\vec{H}}(j)^{in} \le k$.
8     else
          Update $\mathcal{H}$ by deleting all hyperedges induced by $X$ and adding $k|X|-\ell$
          copies of $X$ as a hyperedge to it. The heads of these hyperedges in $\vec{\mathcal{H}}$ will
          be exactly the heads of the deleted hyperedges. $G''$ remains the unchanged.
9 Return $G''$, $\mathcal{H}_{G'+ij} := \mathcal{H}$, and $\vec{\mathcal{H}}_{G'+ij} := \vec{\mathcal{H}}$
```

Next we show that the above procedure maintains our data structure correctly.

Lemma 3.3. *Algorithm 3.2 outputs a maximal (k,ℓ)-sparse subgraph G'' of $G'+ij$, the M-component hypergraph $\mathcal{H}_{G'+ij}$ of $G'+ij$, and its orientation $\vec{\mathcal{H}}_{G'+ij}$ with in-degree of at most k for each vertex.*

Proof. If the condition at STEP **2** is not satisfied, then there exists a (k,ℓ)-M-component, which contains i and j and hence the edge ij cannot be used to construct larger (k,ℓ)-

M-components. Hence $G' + ij$ has the same (k,ℓ)-M-components and G'' is still its maximal (k,ℓ)-sparse subgraph.

We might assume now that the condition at Step **2** is satisfied. If the condition FoundSmallIndegree = True holds in Step **7**, then $\mathcal{H}_{G'} + ij$ is (k,ℓ)-sparse by Lemma 2.8, since it implies that each set Y containing i and j can induce at most $2k - \ell + k(|Y| - 2) = k|Y| - \ell$ hyperedges in $\mathcal{H}_{G'} + ij$. To show that $G'' + ij$ is also (k,ℓ)-sparse, let us take a set $Z \subseteq V$ containing both i and j. (When Z does not contain both i and j, $i_{G''+ij}(Z) = i_{G''}(Z) \leq k|Z| - \ell$ is obvious.) Then $i_{G''+ij}(Z) = i_{G''}(Z) + 1 \leq k|Z| - \ell + 1$ and hence the (k,ℓ)-sparsity condition follows whenever $i_{G''}(Z) < k|Z| - \ell$. Hence we may assume that $i_{G''}(Z) = k|Z| - \ell$ and hence $i_{\mathcal{H}_{G'}}(Z) < i_{G''}(Z)$ by the (k,ℓ)-sparsity of $\mathcal{H}_{G'} + ij$. Now, by the second statement in Lemma 2.6, it follows that there is a (k,ℓ)-M-component of G' with vertex set C, so that Z induces some, but not all of the edges of $G'[C]$. Lemma 2.6 also implies that $k|C| - \ell \leq i_{\mathcal{H}_{G'}}(C) \leq i_{G''}(C) \leq k|C| - \ell$, that is, equality holds throughout. As X and C share an edge, we can use Lemma 2.1 on G'' to show that $i_{G''}(Z \cup C) = k|Z \cup C| - \ell$. However, again by the second statement of Lemma 2.6 and the (k,ℓ)-sparsity of $\mathcal{H}_{G'} + ij$, there exists a (k,ℓ)-M-component of G' so that $Z \cup C$ induces some but not all of its edges. Subsequently adding these vertex sets of (k,ℓ)-M-components of G' to our set with which it shares at least one edge, we get the set V, contradicting the (k,ℓ)-sparsity of $\mathcal{H}_{G'} + ij$. Therefore, $G'' + ij$ is (k,ℓ)-sparse. This also implies that ij is non-redundant in $G' + ij$, since it increases the size of the maximal (k,ℓ)-sparse subgraph G'' of G' and hence $\mathcal{H}_{G'} + ij$ is the M-component hypergraph of $G' + ij$. Clearly, the orientation of $\mathcal{H}_{G'} + ij$ provided by the algorithm in Step **7** satisfies our conditions.

If, in contrast, FoundSmallIndegree = False in Step **7** (and hence we get to Step **8**), the algorithm takes the set X, which has in-degree 0 as it is the set of vertices from which $\{i,j\}$ is reachable. As no reorientation can be made, every vertex has in-degree k in $X - \{i,j\}$. Hence X induces at least $k(|X| - 2) + 2k - \ell = k|X| - \ell$ hyperedges in $\mathcal{H}_{G'}$. By the (k,ℓ)-sparsity of $\mathcal{H}_{G'}$, X induces exactly $k|X| - \ell$ hyperedges in $\mathcal{H}_{G'}$. By its construction, it is easy to see that X is the unique minimal set with the property of containing i and j and inducing $k|X| - \ell$ hyperedges simultaneously.

By Lemma 2.6 and the (k,ℓ)-sparsity of G'', X induces exactly $k|X| - \ell$ edges in G''. Moreover, this is true for any maximal (k,ℓ)-sparse subgraph of G'. Hence $G' + ij$ cannot contain any (k,ℓ)-M-circuit which contains both of i and j and is not a subgraph of $G'[X] + ij$, that is, the vertex set X_0 of the (k,ℓ)-M-component of $G' + ij$ containing both i and j is a subset of X.

As X_0 forms a (k,ℓ)-M-component in $G' + ij$ and ij is a (k,ℓ)-redundant edge in $G' + ij$, G'' is still a maximal (k,ℓ)-sparse subgraph of $G' + ij$ and the (k,ℓ)-M-component hypergraph $\mathcal{H}_{G'+ij}$ of $G' + ij$ induces at least $k|X_0| - \ell$ parallel copies of the hyperedge on X_0. Hence, by Lemma 2.6, X_0 induces exactly $k|X_0| - \ell$ hyperedges in $\mathcal{H}_{G'+ij}$ and $G''[X_0]$ is (k,ℓ)-tight. By the definition of (k,ℓ)-M-components, we can conclude that X_0 induces no edge of any other (k,ℓ)-M-component of $G' + ij$. Note that the definition of (k,ℓ)-M-components also implies that the (k,ℓ)-M-components of $G' + ij$ other than X_0 are (k,ℓ)-M-components of G', as well. Thus X_0 induces either all or none of the edges of each (k,ℓ)-M-component of G'. By applying the second statement of Lemma 2.6 to $\mathcal{H}_{G'}$, G'' and X_0, we can see that $\mathcal{H}_{G'+ij}[X_0]$ is a (k,ℓ)-tight hypergraph containing i and j. This, together with the definition of X, proves that $X \subseteq X_0$ and hence $X = X_0$.

Finally, it is obvious that the in-degree of each vertex is at most k in the orientation provided by the algorithm. □

Next we show that Algorithm 3.2 needs $O(|V|)$ running time and the total running time of our algorithm computing the (k,ℓ)-M-component hypergraph is $O(|V|^2)$.

Theorem 3.4. *Let k and ℓ be positive integers such that $\ell < 2k$. Then the (k,ℓ)-M-component hypergraph $\mathcal{H}_G$ of a graph $G = (V,E)$ (with at most $O(|V^2|)$ edges) can be calculated in $O(|V|^2)$ time. Also, a maximal (k,ℓ)-sparse subgraph G^* of G can also be found in the same running time. Moreover, the algorithm provides an orientation $\vec{\mathcal{H}}_G$ of $\mathcal{H}_G$, with which we can decide whether $\mathcal{H}_G + ij$ is (k,ℓ)-sparse and, if not, we can determine $\mathcal{T}(ij)$ for arbitrary $i, j \in V$ in $O(|V|)$ running time.*

Proof. We shall use Algorithm 3.2 as presented above. We have seen that the vector c_i used in our data structure can be maintained in $O(|V|^2)$ total time if we consider the edges of G in a breath-first manner. Throughout the whole method we run Algorithm 3.2 for each edge.

Let us show that each step of Algorithm 3.2 requires at most $O(|V|)$ running time. Again, we might assume that the condition at Step **2** is satisfied.

First, we try to find a reorientation by running backwards DFSs from i and j on $\vec{\mathcal{H}}$, shown in Step **4**. With the head-lists each DFS needs $O(|V|)$ running time, since we need to traverse only on one copy of each undirected hyperedge of $\mathcal{H}$, which is fast by Lemma 3.1. Now, if we found a vertex v with in-degree at most k, we need to reorient the path leading from $\{i,j\}$ to v. With the help of the data achieved by the DFS, the directed hyperedges of such a path can be found in $O(|V|)$ running time. Note that we also need to refresh the head-lists in every reorientation. Clearly, we can add a directed hyperedge to the head-list of v in $O(1)$ time. As we store the head list in doubly linked lists, we can also delete and add directed hyperedges from and to the head-list of each vertex on the reoriented path in $O(1)$ time. With these steps, one possible reorientation can be found and executed in $O(|V|)$ running time. As we need to run at most $2k - \ell = O(1)$ reorientations, in total the loop of Step **3** requires $O(|V|)$ time.

If FoundSmallIndegree = True in Step **7**, than we can obviously finish the run in $O(1)$ time. On the other hand, we need some more work for Step **8**. Here, to meet our running time constraints, we first create one instance of the undirected hyperedge induced by X. Then we loop through all the undirected hyperedges of $\mathcal{T}(ij)$, and delete them. Now we refresh the corresponding directed edges: we redirect the pointers of the underlying undirected hyperedge, while we do not change the heads or the head-lists. (After this step we also need to update c_i as we described after its definition.) This whole procedure requires $O(|V|)$ time.

Note that we add at most $k|V| - \ell = O(|V|)$ edges to G'' during the whole algorithm since it is (k,ℓ)-sparse. Observe also that, when a new (k,ℓ)-M-component arises, we merge at least two (k, ℓ)-M-components. Only the edges of G'' arise as a new (trivial) M-component during the algorithm, hence we can have at most $O(|V|)$ merges. Therefore, we have $O(|V|)$ edges of G for which we run Steps **3**–**8** from Algorithm 3.2 and for all other edges, the structure is left unchanged following Step **2** in $O(1)$ time. Thus the total running time is indeed $O(|V|^2)$.

Regarding the last statement of the Theorem, given an extra edge ij, we can use the orientation provided, and run Steps **3**–**8** of Algorithm 3.2 in $O(|V|)$ time again to decide whether it maintains the sparsity of $\mathcal{H}_G$. If it does not, we calculate $\mathcal{T}(ij)$, which is the subhypergraph induced by the new hyperedge provided by Step **8**. $\square$

We note here that we used only $O(|V|)$ extra memory for the data structure by Lemma 3.1 and the fact that $\mathcal{H}_{G'}$ has $O(|V|)$ directed hyperedges. Hence, Algorithm 3.2 uses only $O(|V|)$ extra memory.

Note that a graph $G = (V, E)$ is (k, ℓ)-M-connected if and only if its (k, ℓ)-M-component hypergraph consists of $k|V| - \ell$ copies of the hyperedge, which contains all of the vertices of G. Hence Theorem 3.4 implies that the (k, ℓ)-M-connectivity of G can be checked in $O(|V|^2)$ time. Moreover, G is (k, ℓ)-redundant if and only if it is (k, ℓ)-rigid and each of its edges is contained by a (k, ℓ)-circuit, that is, no edge of G is contained by a trivial (k, ℓ)-M-component. This implies the following.

Corollary 3.5. *Let k and ℓ be positive integers such that $\ell < 2k$, and let $G = (V, E)$ be a graph. Then it can be decided in $O(|V|^2)$ time whether G is (k, ℓ)-redundant.* $\square$

Recall the characterizations of global rigidity of several types of frameworks from the Introduction. Recall also from Section 2.2 that the 2- or 3-connectivity of a graph $G = (V, E)$ can be checked in $O(|V|^2)$ time. Hence Corollary 3.5 has the following important consequence.

Corollary 3.6. *Let $G = (V, E)$ be a graph. Then it can be decided in $O(|V|^2)$ time whether G is generically globally rigid in the Euclidean plane, in a non-Euclidean plane, or on the cylinder.* $\square$

4 Augmentation problems on (k, ℓ)-rigid graphs

Algorithms for augmenting (k, ℓ)-rigidity have a long history. There exists an $O(|V| + |E|) \leq O(|V|^2)$ time algorithm that searches for the smallest set of edges making a graph $G = (V, E)$ 2-edge-connected (that is, (1,1)-redundant) [7]. García and Tejel [12] showed an $O(|V|^2)$ method to add the minimum number of edges so that a $(2, 3)$-tight graph becomes $(2, 3)$-redundant. In [26], a min-max theorem was given to this problem and also to its extension, called the **(k, ℓ)-redundant rigidity augmentation problem**, in which the goal is to find the smallest edge set that makes a given (k, ℓ)-tight graph (k, ℓ)-redundant. (It was also shown that the extension of this problem for (k, ℓ)-rigid inputs is NP-hard whenever $\ell > k$.)

Let $\boldsymbol{\mathcal{R}_{\mathcal{H}}(i_1j_1, \ldots, i_rj_r)}$ denote the subhypergraph of $\mathcal{H} = (V, \mathcal{E})$ induced by the hyperedges in $\mathcal{E}$ which are (k, ℓ)-redundant in $\mathcal{H} \cup \{i_1j_1, \ldots, i_rj_r\}$ where $i_1, \ldots, i_r$, $j_1, \ldots, j_r \in V$. Note that $\mathcal{R}_{\mathcal{H}}(ij) = \mathcal{T}_{\mathcal{H}}(ij)$ for any (k, ℓ)-tight hypergraph $\mathcal{H}$ and any $i, j \in V$. The following lemma extends this simple fact.

Lemma 4.1. *[26] If $\mathcal{H}$ is a (k,ℓ)-tight hypergraph with $0 < \ell < 2k$, then*

$$\mathcal{R}_{\mathcal{H}}(i_1j_1,\dots,i_rj_r) = \mathcal{T}_{\mathcal{H}}(i_1j_1) \cup \dots \cup \mathcal{T}_{\mathcal{H}}(i_rj_r).$$

□

A vertex set C of a (k,ℓ)-tight hypergraph $\mathcal{H} = (V,\mathcal{E})$ is called **(k,ℓ)-co-tight**, if $V - C$ induces a (k,ℓ)-tight subhypergraph. We say that an edge set F **touches** a vertex set X if at least one edge in F has an endvertex in X. The following statement shows why co-tight sets are important in this topic.

Observation 4.2. *[26] Let $\mathcal{H} = (V,\mathcal{E})$ be a (k,ℓ)-tight hypergraph on at least $k^2 + 2$ vertices and let C be a (k,ℓ)-co-tight set of $\mathcal{H}$. Then, for any edge set F, for which $\mathcal{H} + F$ is (k,ℓ)-redundant, F touches C.* □

The main result of [26] is the following.

Theorem 4.3. *[26] Let $\mathcal{H} = (V,\mathcal{E})$ be a (k,ℓ)-tight hypergraph on at least $k^2 + 2$ vertices. If there exists any (k,ℓ)-co-tight set in $\mathcal{H}$, then* $\min\{|F| : H = (V,F)$ *is a graph for which $\mathcal{H} \cup H$ is (k,ℓ)-redundant*$\} = \max\left\{ \left\lceil \frac{|\mathcal{C}|}{2} \right\rceil : \mathcal{C}\right.$ *is a family of pairwise disjoint (k,ℓ)-co-tight sets of $\mathcal{H}$* $\left.\right\}$. *Otherwise, $\mathcal{H} + uv$ is (k,ℓ)-redundant for every pair $u,v \in V$.* □

The minimal (k,ℓ)-co-tight sets of $\mathcal{H}$ will be called the **(k,ℓ)-MCT** sets of $\mathcal{H}$. The key lemmas in the proof of Theorem 4.3 are the following.

Lemma 4.4. *[26] Let $\mathcal{H}$ be a (k,ℓ)-tight hypergraph on at least $k^2 + 2$ vertices, with $\mathcal{C}^*$ as the family of its (k,ℓ)-MCT sets. Then the members of $\mathcal{C}^*$ are either pairwise disjoint and $|\mathcal{C}^*| \geq 3$ or there exists a pair $u,v \in V$ such that $\mathcal{T}(uv) = \mathcal{H}$, that is, $\mathcal{H}+uv$ is (k,ℓ)-redundant. Furthermore, in the first case no hyperedge of $\mathcal{H}$ intersects more than one (k,ℓ)-MCT sets.* □

A set $P \subseteq V$ is called a **transversal** of a set system $\mathcal{S}$ if $P \subseteq \bigcup \mathcal{S}$ and $|P \cap S| = 1$ for each $S \in \mathcal{S}$.

Lemma 4.5. *[26] Let $\mathcal{H}$ be a (k,ℓ)-tight hypergraph on at least k^2+2 vertices. Suppose that there is no edge uv for which $\mathcal{H} + uv$ is (k,ℓ)-redundant. Let P be a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}$. Then for any connected graph H on P (for example a star $H = K_{1,|P|-1}$), $\mathcal{H} + H$ is (k,ℓ)-redundant.* □

Lemma 4.6. *[26] Let $\mathcal{H}$ be a (k,ℓ)-tight hypergraph on at least k^2+2 vertices. Suppose that there is no edge uv for which $\mathcal{H} + uv$ is (k,ℓ)-redundant. Let y, x_1, x_2, x_3 be elements of distinct (k,ℓ)-MCT sets of $\mathcal{H}$. Let $\mathcal{T}^* = \mathcal{T}(yx_1) \cup \mathcal{T}(yx_2) \cup \mathcal{T}(yx_3)$. Then $\mathcal{T}^* = \mathcal{T}(x_1y) \cup \mathcal{T}(x_2x_3)$ or $\mathcal{T}^* = T(x_2y) \cup \mathcal{T}(x_1x_3)$ holds.* □

Let us call a (k,ℓ)-tight subhypergraph $\mathcal{H}_0$ of $\mathcal{H}$ **generated** if there are $u,v \in V$ such that $\mathcal{T}_{\mathcal{H}}(uv) = \mathcal{H}_0$. The following result of [26] provides a connection between the (k,ℓ)-MCT sets and the 'classes of extreme vertices' defined by García and Tejel [13].

Lemma 4.7. *[26] Let $\mathcal{H}$ be a (k,ℓ)-tight hypergraph on at least k^2+2 vertices. Suppose that there is no edge uv so that $\mathcal{H}+uv$ is (k,ℓ)-redundant. Then $\mathcal{T}(uv)$ is inclusion-wise maximal amongst all the generated subgraphs of $\mathcal{H}$ if and only if $u,v \in V$ are elements of two distinct (k,ℓ)-MCT sets. Moreover, two inclusion-wise maximal generated subgraphs $\mathcal{T}(uv_1)$ and $\mathcal{T}(uv_2)$ are equal if and only if v_1, v_2 are in the same (k,ℓ)-MCT set.* □

Besides the min-max theorem, an $O(|V|^2)$ time algorithm was given for the (k,ℓ)-redundant rigidity augmentation problem in [26] for graph inputs. However, for hypergraphs, its running time has an extra $|V|$ factor which comes from the extra factor in the running time of the algorithm of Section 2.1 corresponding to the potential size of the hyperedges. Since $\mathcal{H}_G$ can be calculated in $O(|V|^2)$ time and after this $\mathcal{T}_{\mathcal{H}_G}(ij)$ can be calculated in $O(|V|)$ time by Theorem 3.4, the (k,ℓ)-redundant rigidity augmentation problem on the (k,ℓ)-M-component hypergraph of a (k,ℓ)-rigid graph can also be solved in $O(|V|^2)$ time. The combination of Theorem 3.4 and the algorithmic results of [26] can be summarized as follows.

Theorem 4.8. *Let $G=(V,E)$ be a (k,ℓ)-rigid graph. Let $\mathcal{H}_G$ be the (k,ℓ)-M-component hypergraph of G. Then a minimum cardinality edge set F for which $\mathcal{H}_G+F$ is (k,ℓ)-redundant can be calculated in $O(|V|^2)$ time. If there is no edge uv for which $\mathcal{H}_G+uv$ is (k,ℓ)-redundant, then the algorithm also provides a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$.* □

For completeness, we shall give an alternative proof to Theorem 4.8 in the Appendix by extending the ideas of the paper by Gacía and Tejel [13].

As we mentioned in the Introduction, one of our goals is to give an efficient algorithm that can be applied in several global ridity augmentation problems. Hence we consider the following generalization: Given (k,ℓ)-rigid graph G, we want to find a minimum cardinality edge set F for which $G+F$ is (k,ℓ)-redundant and $(c_{k,\ell}+1)$-connected. (In fact, we also need the simplicity of the output which will be handled later.) To make this problem more approachable we may use the fact that a (k,ℓ)-redundant and $(c_{k,\ell}+1)$-connected graph is (k,ℓ)-M-connected (if it has not too many parallel edges when $k>\ell$). With this, it was shown in [25] that the (generalized) global ridity augmentation problem is equivalent to the following problem, if we assume that $0<\ell\le\frac{3}{2}k$, and G is simple if $\ell>k$.

Problem 1. *Let k and ℓ be positive integers such that $0<\ell\le\frac{3}{2}k$, and let $G=(V,E)$ be a (k,ℓ)-rigid graph, which is simple if $k<\ell$. Let $\mathcal{H}_G=(V,\mathcal{E})$ denote the (k,ℓ)-M-component hypergraph of G. Find a graph $H=(V,F)$ with a minimum cardinality edge set F such that $\mathcal{H}_G\cup H=(V,\mathcal{E}\cup F)$ is (k,ℓ)-redundant and $G\cup H=(V,E\cup F)$ is $(c_{k,\ell}+1)$-connected, where $c_{k,\ell}=\lceil\frac{\ell}{k}\rceil$.*

The main result of [25] is the following.

Theorem 4.9. *[25] Let k and ℓ be two positive integers such that $\ell\le\frac{3}{2}k$. Let $G=(V,E)$ be a (k,ℓ)-rigid graph on at least k^2+2 vertices. Suppose also that G is simple if $k<\ell$. Let $\mathcal{H}_G=(V,\mathcal{E})$ be the M-component hypergraph of G. If*

G is $(c_{k,\ell}+1)$-connected, (k,ℓ)-tight, and there is no (k,ℓ)-co-tight set in $\mathcal{H}_G$, then any new edge makes $\mathcal{H}_G$ (k,ℓ)-redundant. Otherwise, $\min\{|F| :$ *F is a set of graph edges for which $\mathcal{H}_G + F = (V, \mathcal{E} \cup F)$ is (k,ℓ)-redundant and $(c_{k,\ell}+1)$-connected*$\} = \max\left\{b^{c_{k,\ell}}(G) - 1, \max\left\{\left\lceil \frac{|\mathcal{A}|}{2} \right\rceil :\right.\right.$ *$\mathcal{A}$ is a family of pairwise disjoint (k,ℓ)-co-tight sets of $\mathcal{H}_G$ and $(c_{k,\ell}+1)$-fragments of G*$\left.\left.\right\}\right\}$. □

The minimal members of the family of the (k,ℓ)-co-tight sets of $\mathcal{H}_G$ and the $(c_{k,\ell}+1)$-fragments of G will be called the **atoms** of G. Note that if G is (k,ℓ)-rigid and $(c_{k,\ell}+1)$-connected graph then the statement of Theorem 4.9 coincide with the statement of Theorem 4.3 for $\mathcal{H}_G$. The key lemmas in the proof of Theorem 4.9, for the case where G is not $(c_{k,\ell}+1)$-connected, are the following. In the following lemmas we always assume that $G = (V, E)$ *is a (k,ℓ)-rigid graph on at least k^2+2 vertices which is not $(c_{k,\ell}+1)$-connected. Moreover, we suppose that $0 < \ell \le \frac{3}{2}k$, and G is simple if $k < \ell$.*

Lemma 4.10. *[25] Let A be an atom of G and let $a \in A$. Then $G - a$ is $c_{k,\ell}$-connected.* □

Lemma 4.11. *[25] The atoms of G are pairwise disjoint and no edge of G connects two atoms.* □

Lemma 4.12. *[25] Assume that there is no edge uv for which $\mathcal{H}_G + uv$ is (k,ℓ)-redundant and $G + uv$ is $(c_{k,\ell}+1)$-connected. Let P be a transversal of the atoms of $\mathcal{H}_G$. Then for any connected graph H on P (for example a star $H = K_{1,|P|-1}$), $\mathcal{H}_G + H$ is (k,ℓ)-redundant and $G + H$ is $(c_{k,\ell}+1)$-connected. Furthermore, for any two connected graphs H' and H'' on a subset $P' \subseteq P$, the sets of (k,ℓ)-redundant hyperedges of $\mathcal{H}$ in $\mathcal{H} + H'$ and $\mathcal{H} + H''$ coincide.* □

Lemma 4.13. *[25] Assume that there is no edge uv for which $\mathcal{H}_G + uv$ is (k,ℓ)-redundant and $G+uv$ is $(c_{k,\ell}+1)$-connected. Let P be a transversal of the atoms of $\mathcal{H}_G$ and let $x_1, x_2, x_3, y \in P$ be distinct vertices. Let $\mathcal{T}^* = \mathcal{T}_{\mathcal{H}_G}(x_1y) \cup \mathcal{T}_{\mathcal{H}_G}(x_2y) \cup \mathcal{T}_{\mathcal{H}_G}(x_3y)$. Then $\mathcal{T}^* = \mathcal{T}_{\mathcal{H}_G}(x_1y) \cup \mathcal{T}_{\mathcal{H}_G}(x_2x_3)$ or $\mathcal{T}^* = \mathcal{T}_{\mathcal{H}_G}(x_2y) \cup \mathcal{T}_{\mathcal{H}_G}(x_1x_3)$ holds.* □

5 Global rigidity augmentations

In this section we give an algorithm for Problem 1 with running time $O(|V|^2)$. Our input is a (k,ℓ)-rigid graph on at least k^2+2 vertices (where $\ell \le \frac{3}{2}k$) which is assumed to be simple, when $k < \ell$. The algorithms differ in some details depending on $c_{k,\ell}$ hence, after a common introduction, we show the algorithm first for $c_{k,\ell} = 1$. Next we present how it can be extended to $(k,\ell) = (2,3)$ and to the cases where $3 \le k < \ell \le \frac{3}{2}k$.

We start our algorithm by constructing $BC(G)$ or $SPQR(G)$ (depending on $c_{k,\ell}$). This can be done in $O(|V|^2)$ time by Lemmas 2.10 and 2.11. We also calculate $\mathcal{H}_G$ in $O(|V|^2)$ time by using the algorithm of Theorem 3.4. If G is $(c_{k,\ell}+1)$-connected, then we only need to augment $\mathcal{H}_G$ to a (k,ℓ)-redundant hypergraph which can be done in $O(|V|^2)$ time by Theorem 4.8. Hence from now on, we assume that G is not $(c_{k,\ell}+1)$-connected. In this case the atoms of G are pairwise disjoint by Lemma 4.11. Next, we need to construct a transversal P of the atoms.

Constructing the transversal of the atoms. Our algorithm will rely on Theorem 4.8 and the following greedy subroutine which we got as an extension of [26, Algorithm 6.7].

Algorithm 5.1. INPUT: *The (k,ℓ)-M-component hypergraph $\mathcal{H}_G = (V, \mathcal{E})$ of a (k,ℓ)-rigid graph G on at least k^2+2 vertices, a set $L \subseteq V$ and a vertex $i \in V$.*
OUTPUT: *A (minimum cardinality) vertex set $V'(i,L)$ such that $V = \bigcup_{j \in V'(i,L)} V(ij)$.*

```
1 Initialize $V'(i,L) := \emptyset$. All vertices are unmarked.
2 Mark $i$.
3 Explore all vertices $j \in L$. Subsequently, explore all other vertices $j \in V - L$:
4       If $j$ is unmarked, then
5             Calculate $\mathcal{T}_{\mathcal{H}_G}(ij)$
6             Mark all unmarked vertices in $V_{\mathcal{H}_G}(ij)$
7             $V'(i,L) := [V'(i,L) - V_{\mathcal{H}_G}(ij)] + j$
8 Return $V'(i,L)$
```

Lemma 5.2. *Algorithm 5.1 runs in $O(|V|^2)$ time.*

Proof. Recall that $\mathcal{T}_{\mathcal{H}_G}(ij)$ can be calculated in $O(|V|)$ running time for any vertex j by Theorem 3.4. Hence, as we compute $\mathcal{T}_{\mathcal{H}_G}(ij)$ only $O(|V|)$ times, Algorithm 5.1 concludes in $O(|V|^2)$ running time. Note that the marking and set operations can be executed without additional complexity in $O(|V|)$ running time each. □

Lemma 5.3. *Let k and ℓ be two positive integers such that $\ell < 2k$. Let G be a (k,ℓ)-rigid graph on at least k^2+2 vertices and let $\mathcal{H}_G$ be its (k,ℓ)-M-component hypergraph. Suppose that there is no edge uv such that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant. Let i' be a vertex from a (k,ℓ)-MCT set of $\mathcal{H}_G$ and let L be an arbitrary subset of V. Let $V'(i',L)$ be the result of Algorithm 5.1 with the input $\mathcal{H}_G$, i' and L. Then $V'(i',L) \cup \{i'\}$ is a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$, and hence every vertex from $V'(i',L)$ is a vertex from a (k,ℓ)-MCT set of $\mathcal{H}_G$. Moreover, if $L \cap C \neq \emptyset$ for a (k,ℓ)-MCT set C for which $i' \notin C$, then $V'(i',L) \cap C \cap L \neq \emptyset$.*

Proof. If i' is from a (k,ℓ)-MCT set of $\mathcal{H}_G$, then by Lemma 4.7 we know that the inclusion-wise maximal generated (k,ℓ)-tight subhypergraphs of $\mathcal{H}_G$ are exactly the ones generated by two vertices from different (k,ℓ)-MCT sets. Hence it follows from Observation 4.2 and Lemma 4.7 that $V'(i',L)$ consists of exactly one vertex from every other (k,ℓ)-MCT set. As we check first the vertices in L, it is clear that we choose a vertex from $L \cap C$ for each (k,ℓ)-MCT set C for which $C \cap L \neq \emptyset$ and $i \notin C$ by Lemma 4.7. □

Lemma 5.4. *Let k and ℓ be positive integers such that $\ell \leq \frac{3}{2}k$. Let $G = (V,E)$ be a (k,ℓ)-rigid graph on at least k^2+2 vertices, which is not $(c_{k,\ell}+1)$-connected. Suppose also that G is simple if $k < \ell$. Then a transversal P of the atoms of G can be found in $O(|V|^2)$ running time.*

Proof. First we find all the $(c_{k,\ell}+1)$-ends of G. We can obtain this by first computing $BC(G)$ or $SPQR(G)$ in $O(|V|^2)$ running time. The set of the leaves of $BC(G)$ and of

$SPQR(G)$ correspond to the 2-ends and 3-ends (along the maps φ and φ' and deleting the corresponding cut-vertex or cut-pairs from them) by Lemmas 2.10 and 2.11, respectively. (Note that a vertex of degree two, which is contained in the vertex set of a polygon of length at least four, possibly could also be a 3-end of G as a singleton. However, this cannot happen in a simple (k,ℓ)-tight graph where $k<\ell$. To see this, observe that a (k,ℓ)-tight graph has no vertex of degree 2 when $k \geq 3$, by Lemma 2.2. On the other hand, when $k=2$, $\ell=3$ and $d(v)=2$, then $G-v$ is also $(2,3)$-rigid and hence 2-connected by Proposition 2.9. Hence v is not contained by a polygon of length at least four in this case.) This also implies that the $(c_{k,\ell}+1)$-ends of G are pairwise disjoint. Let L_0 denote the set of vertices which are contained in any $(c_{k,\ell}+1)$-end of G. As P must intersect every (k,ℓ)-MCT set of $\mathcal{H}_G$, we can use the algorithm of Theorem 4.8 (with $O(|V|^2)$ running time) to construct a transversal P_0 of the (k,ℓ)-MCT sets of $\mathcal{H}_G$ or conclude that $\mathcal{H}_G + u'v'$ is (k,ℓ)-redundant for a pair $u',v' \in V$.

If there is no edge uv, so that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant, then the (k,ℓ)-MCT sets of $\mathcal{H}_G$ are pairwise disjoint by Lemma 4.4, and the output P_0 of the algorithm of Theorem 4.8 contains exactly one element of each of them. Let $i' \in P_0$ and let us run Algorithm 5.1 for $i=i'$ and $L=L_0$ (in $O(|V|^2$ time by Lemma 5.2). By Lemma 5.3, $V'(i',L_0)\cup\{i'\}$ is also a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$, and if $L_0 \cap C \neq \emptyset$ for a (k,ℓ)-MCT set C for which $i' \notin C$, then $V'(i',L_0)\cap C \cap L_0 \neq \emptyset$. Next we choose a vertex $i'' \in V'(i',L_0)$ and run Algorithm 5.1 for $i=i''$ and $L=L_0$ (again in $O(|V|^2$ time by Lemma 5.2) to calculate $V'(i'',L_0)$. Now, by Lemma 5.3 (and by the choice of i''), $X := V'(i'',L_0)\cup\{i''\}$ is also a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$, and if $L_0\cap C \neq \emptyset$ for a (k,ℓ)-MCT set C, then $X \cap C \cap L_0 \neq \emptyset$.

Note that Lemmas 4.4 and 4.11 imply that a (k,ℓ)-MCT set of $\mathcal{H}_G$ and a $(c_{k,\ell}+1)$-end of G are either disjoint or one of them contains the other. Thus the above set X is a transversal of the (k,ℓ)-MCT sets which intersects the most $(c_{k,\ell}+1)$-ends possible. After getting X we can choose one vertex from each $(c_{k,\ell}+1)$-end which does not intersect X. Adding these vertices to X we get a set P. Note that we got P in $O(|V|^2)$ running time.

Claim 5.5. *P is a transversal of the atoms of G.*

Proof. Recall first that the $(c_{k,\ell}+1)$-ends of G are pairwise disjoint. On the other hand, in this case the (k,ℓ)-MCT sets of $\mathcal{H}_G$ are also pairwise disjoint. Since the atoms are also pairwise disjoint and they are the minimal elements of the family of all $(c_{k,\ell}+1)$-ends of G and all (k,ℓ)-MCT sets of $\mathcal{H}_G$, if a (k,ℓ)-MCT set C of $\mathcal{H}_G$ intersects a $(c_{k,\ell}+1)$-ends K of G, then $C\subseteq K$ or $K\subseteq C$ must hold. Hence the vertices in X are all chosen from pairwise disjoint atoms. On the other hand, the rest of the vertices of P are choosen from the $(c_{k,\ell}+1)$-ends of G which do not intersect X, hence these $(c_{k,\ell}+1)$-ends are also all atoms and hence P is indeed a transversal of the atoms of G. □

Let us turn to the case where there is an edge $u'v'$ so that $\mathcal{H}_G + u'v'$ is (k,ℓ)-redundant. In this case, the algorithm of Theorem 4.8 finds such an edge uv. If there are any (k,ℓ)-MCT sets that are also atoms, they must contain either u or v

by Observation 4.2. In fact, if there are two of them, then one of them must contain u while the other contains v. Given a transversal of the $(c_{k,\ell}+1)$-ends P_0 such that $|P_0 \cap \{u,v\}|$ is as large as possible (that is, we chose u or v from a $(c_{k,\ell}+1)$-end if possible), either P_0, $P_0+\{u\}$, $P_0+\{v\}$ or $P_0+\{u,v\}$ is a transversal of the atoms. We shall test each of them to determine the actual transversal. The algorithm for this is as follows.

Let X denote the actual vertex set of interest. If X is a transversal on the atoms of G, then, for the edge set F of any connected graph on X, $\mathcal{H}_G+F$ is (k,ℓ)-redundant by Lemma 4.5. On the other hand, if X does not contain a transversal of the atoms, then $\mathcal{H}_G+F$ clearly cannot be (k,ℓ)-redundant, as there is a (k,ℓ)-MCT set which is not intersected by $V(F)$ contradicting Observation 4.2. By Lemma 4.1 and Theorem 3.4, it can be checked in $O(|V|^2)$ time whether $\mathcal{H}_G+F$ is (k,ℓ)-redundant and hence we can find out in $O(|V|^2)$ time whether P_0, $P_0+\{u\}$, $P_0+\{v\}$ or $P_0+\{u,v\}$ is a transversal of the atoms of G in this case. □

Now, we know that the addition of any star S_P on a transversal P of the atoms of G makes G $(c_{k,\ell}+1)$-connected and $\mathcal{H}_G$ (k,ℓ)-redundant by Lemma 4.12. By Lemma 4.13, for any three edges x_1y, x_2y and x_3y of this star, $\mathcal{H}_G+S_P-\{x_2y,x_3y\}+\{x_2x_3\}$ or $\mathcal{H}_G+S_P-\{x_1y,x_3y\}+\{x_1x_3\}$ is (k,ℓ)-redundant. Observe that, for $i=1,2$, $S_P-\{x_iy,x_3y\}$ is a star on $P-\{x_i,x_3\}$. By Lemma 4.1 and the last statement of Lemma 4.12, it follows that, if $\mathcal{H}_G+S_P-\{x_iy,x_3y\}+\{x_ix_3\}$ is (k,ℓ)-redundant, then $\mathcal{H}_G+S_{P-\{x_i,x_3\}}+\{x_ix_3\}$ is also (k,ℓ)-redundant for any star $S_{P-\{x_i,x_3\}}$ on $P-\{x_i,x_3\}$. Hence from this point the algorithm may choose three vertices x_1, x_2 and x_3 from a subset $N\subseteq P$ of 'non-fixed' vertices and delete either x_2 and x_3, or x_1 and x_3 along with the addition of the edge x_2x_3 or x_1x_3 to the final augmenting edge set, respectively. The only issue is that we also want to get a $(c_{k,\ell}+1)$-connected graph at the end, hence the three vertices must be chosen in such a way that, after fixing the edge, the $(c_{k,\ell}+1)$-connectivity augmentation can still be solved by the same number of edges. This issue was handled by an algorithmic proof in [25]. In what follows, we show how the algorithm which arise from the proof of [25, Lemma 4.15] can be run in $O(|V|^2)$ time by using the structures presented in Section 2.2. As the structures differ for the 2- and 3-connectivity augmentations, we first give the algorithm for the case where $0<\ell\le k$. We shall denote a star on the vertex set X by $\boldsymbol{S_X}$.

Algorithm 5.6 (Based on [25, Lemma 4.15]). Input: *A (k,ℓ)-redundant graph $G=(V,E)$ (where $0<\ell\le k$) on at least k^2+2 vertices which is not 2-connected, the (k,ℓ)-M-component hypergraph $\mathcal{H}_G$ of G, along with its orientation $\vec{\mathcal{H}}_G$ where the in-degree of each vertex is at most k, the block-cut tree $BC(G)$ of G, and a transversal P of the atoms of G.*

Output: *A minimum size edge set F for which $\mathcal{H}_G+F$ is (k,ℓ)-redundant and $G+F$ 2-connected.*

```
1 $N := P$, $F := \emptyset$
2 **While** $|N| \ge \max\{4, b^1(G+F)+1\}$ **do**
3   **If** $b^1(G+F)-1 \ge \left\lceil \frac{|N|}{2} \right\rceil$, **then**
4       **If** there is only one cut-vertex $v$ such that $b^1_v(G+F) = b^1(G+F)$, **then**
```

```
            Choose $x_1, x_2$ from a component of $G + F - \{v\}$ that contains at least two
            vertices from $N$. Let $x_3 \in N$ be a vertex from a component of $G + F - \{v\}$
            that does not contain $x_1$ and $x_2$.
5        else
            Let $v_1$ and $v_2$ be two cut-vertices for which $b^1_{v_1}(G+F) = b^1(G+F) = b^1_{v_2}(G+F)$.
            Choose $x_1, x_2 \in N$ from two different components of $G + F - \{v_1\}$ that
            do not contain $v_2$. Choose $x_3 \in N$ from a component of $G + F - \{v_2\}$ that
            does not contain $v_1$.
6    else
7        If there is a 2-fragment $K$ of $G$ such that $|N \cap K| \geq 2$ and $|N - K| \geq 2$, then
            Choose $x_1, x_2$ from $N \cap K$ and choose $x_3$ from $N - K$.
8        else         (Notice that if $b^1(G + F) = 1$, then this is the only possible case.)
            Choose $x_1, x_2, x_3 \in N$ arbitrarily.
9     Take $y \in N - \{x_1, x_2, x_3\}$.
10    If $\mathcal{T}_{\mathcal{H}_G}(x_1y) \cup \mathcal{T}_{\mathcal{H}_G}(x_2y) \cup \mathcal{T}_{\mathcal{H}_G}(x_3y) = \mathcal{T}_{\mathcal{H}_G}(x_2y) \cup \mathcal{T}_{\mathcal{H}_G}(x_1x_3)$, then
         $x := x_1, x' := x_3$.
      else
         $x := x_2, x' := x_3$.
11    $F := F + \{xx'\}$, $N = N - \{x, x'\}$. Refresh $BC(G + F)$.
12 $F := F \cup S_N$. Return: $F$.
```

It was shown in [25] that Algorithm 5.6 returns an edge set F for which $\mathcal{H}_G + F$ is (k, ℓ)-redundant, $G + F$ is 2-connected, and its size is optimal. Hence we just need to show that the running time of Algorithm 5.6 is $O(|V|^2)$.

Theorem 5.7. *Let $G = (V, E)$ be a (k, ℓ)-rigid graph where $\ell \leq k$. Then Algorithm 5.6 can be executed in $O(|V|^2)$ time on G and hence Problem 1 can be solved in $O(|V|^2)$ time on G.*

Proof. Recall the properties of BC-trees from Section 2.2. As Algorithm 5.6 starts with $F = \emptyset$, we have $BC(G + F) = BC(G)$ initially. Now suppose that we know $BC(G + F)$. Remember that the degree of a node c_v of the BC-tree corresponding to a cut-vertex v of G equals to $b^1_v(G + F)$. Hence the computation of $b^1(G + F)$ takes $O(|V|)$ running time (if we use $BC(G + F)$), as well, as finding the single vertex v for which $b^1_v(G + F) = b^1(G + F)$ or the pair of vertices u, v for which $b^1_u(G + F) = b^1_v(G + F) = b^1(G + F)$. Furthermore, since the components of $BC(G + F) - c_v$ correspond to the components of $G + F - v$, we can execute Steps **4** and **5** in $O(|V|)$ time. Similarly, $BC(G + F)$ can help us finding the appropriate K in Step **7** also in $O(|V|)$ time.

By Theorem 3.4, we can compute $\mathcal{T}_{\mathcal{H}_G}(ij)$ in $O(|V|)$ running time. Thus executing Step **10** needs $O(|V|)$ time. For Step **11**, we also need to refresh the BC-tree of $G+F$. Recall that $BC(G+e) = BC(BC(G)+e)$ holds for any edge e by Lemma 2.10. Thus, as $BC(G)$ has $O(|V|)$ edges, we can refresh it in $O(|V|)$ running time. Clearly, the rest of the steps need only $O(1)$ time. Hence each execution Steps **3**–**11** takes only $O(|V|)$ time. By the condition of Step **2** and the fact that $|N| \leq |V|$ and $|N|$ decreases by two in each loop, we can conclude that the loop can be executed at most $O(|V|)$ times. This proves that the whole algorithm runs in $O(|V|^2)$ time. □

Let us consider next the case of $(k, \ell) = (2, 3)$. Here, G has no weak cut-pairs by Lemma 2.12. Hence we can describe its algorithm with simpler terms, separated from the general case of $k < \ell \leq \frac{3}{2}k$.

As now G is 2-connected but contains no weak cut-pairs, the properties of $SPQR(G)$ are quite similar to the properties of a BC-tree by Section 2.2. Now, by changing cut-vertices to cut-pairs, b^1 to b^2, $BC(G + F)$ to $SPQR(G + F)$, and 2-fragment to 3-fragment in Algorithm 5.6, we can run it to solve Problem 1 for $(k, \ell) = (2, 3)$. By [25, Lemma 4.15] Algorithm 5.6 with these changes indeed results an optimal solution. Now, by copying the proof of Theorem 5.7 and using the fact that the SPQR-tree can be updated in $O(|V|)$ time by Lemma 2.11, we can get the following.

Theorem 5.8. *Let $G = (V, E)$ be a simple $(2, 3)$-rigid graph. Then Problem 1 can be solved in $O(|V|^2)$ time.* □

The solution of Problem 1 for $k < \ell \leq \frac{3}{2}k$ in $O(|V|^2)$ time requires some more changes in Algorithm 5.6. The reason for this is the possible existence of weak cut-pairs, with which there might be $O(|V|^2)$ cut-pairs in total. In what follows, we sketch how to handle these difficulties.

As before, Lemma 5.4 provides us a transversal P of the atoms. We shall follow and change the steps of Algorithm 5.6 with the use of the SPQR-tree, in a similar manner to the case of $(k, \ell) = (2, 3)$. Let us take a look at the weak cut-pairs. For a weak cut-pair $\{u, v\}$ of $G + F$, $b^2_{\{u,v\}}(G + F) = 2$. Thus no weak cut-pairs could be considered until Step **6** and hence no change is needed to be made in the algorithm until this point compared to the case of $(k, \ell) = (2, 3)$. However, to handle the weak cut-pairs, we need a slightly modified version of Step **7**, introduced in [25, Section 4.2].

```
7'  If there is a 3-fragment $K$ of $G$ such that $|N \cap K| \geq 2$ and $|N - K| \geq 2$, then
        Choose $x'_1, x'_2$ from $N \cap K$ and choose $x_3$ from $N - K$.
        If every 3-end of $G + F + x'_1x_3$ contains a vertex from $N - \{x'_1, x_3\}$, then
            Let $x_1 = x_2 := x'_1$,
        else
            Let $x_1 = x_2 := x'_2$.
```

As it was shown in [25], after this modification the algorithm indeed provides an optimal solution for Problem 1 when $3 \leq k < \ell \leq \frac{3}{2}k$. Since all the other steps work like in the case of $(k, \ell) = (2, 3)$, we only need to show that Step **7'** can be executed in $O(|V|)$ time and that the SPQR-tree can be updated in the same time, as well. To check whether $G + F$ has any 3-fragment K for which both $N \cap K$ and $N - K$ have cardinality at least two, we first check this condition for the 3-fragments which arise as a connected component after the deletion of a strong cut-pair. We do this by using the properties of the SPQR-tree described in Lemma 2.11 (like we did for $(k, \ell) = (2, 3)$). If we cannot find a proper strong cut-pair, then the deletion of each strong cut-pair leads to a graph with exactly two connected components, one of which contains exactly one vertex from N. (Note that each 3-fragment of $G + F$ must contain at least one element of N as the algorithm results a 3-connected graph in the end.) In this case, we check whether $SPQR(G + F)$ has any polygon node with degree at least three. If not, then it is easy to see that the condition of Step **7'** fails and we can go to the next step. Otherwise, let the cyclic order of the vertices on the polygon be $x_1, \ldots x_k, x_{k+1} = x_1$. Recall from Section 2.2 that the neighbors of the node of the SPQR-tree corresponding to this polygon belong to different strong cut-pairs (which are some neighboring vertices on the polygon), and all the non-neighboring vertices of the polygon form weak cut-pairs in $G + F$. If $\{x_i, x_{i+1}\}$ is a cut-pair, then, by our previous observation, $G + F - \{x_i, x_{i+1}\}$ has two connected components, one of which contains exactly one vertex from N. Note that the other component contains all other vertices from N as N is disjoint from the cut-pairs by Lemma 4.10. Hence, it is easy to see that our polygon node must have degree at least four in $SPQR(G + F)$, that is, there are indices $1 \le i_1 < i_2 < i_3 < i_4 \le k$ such that $\{x_{i_j}, x_{i_j+1}\}$ is a cut-pair of $G + F$ for $j = 1, 2, 3, 4$. Then, by the properties of SPQR-trees, one can see that each of the two components of $G + F - \{x_{i_1}, x_{i_3}\}$ contains at least two elements of N. (Illustrated by Figure 2.)

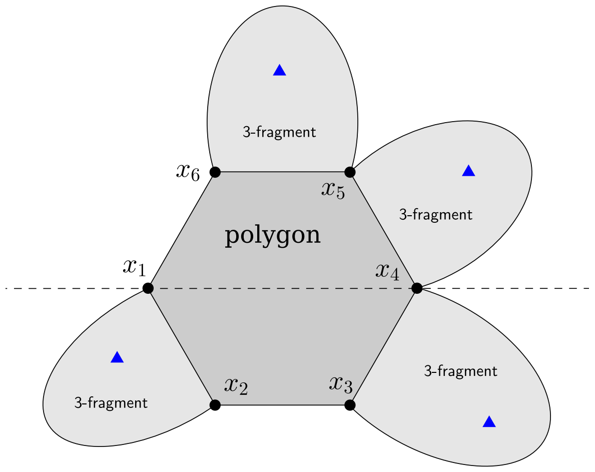

Figure 2: A schematic figure of a polygon node (dark gray area) of the SPQR-tree and the 3-fragments (light gray areas) corresponding to its strong cut-pairs. If no 3-fragment, arising from a deletion of a strong cut-pair, fulfills the condition of Step **7'** and we have at least 3 strong cut-pairs on the polygon, then each light gray 3-fragment contains exactly one element of N (the triangular vertices). As $|N| \ge 4$, this implies that we have at least 4 strong cut-pairs in the polygon. The areas below and above the dashed line correspond to the connected components after the deletion of the weak cut-pair $\{x_1, x_4\}$. These fulfill the condition of Step **7'**.

Next, based on this two-separation, the choice of x_1', x_2', and x_3 can be made in $O(|V|)$ time. For the second condition (that is, whether every 3-end of $G + F + x_1'x_3$ contains a vertex from $N - \{x_1', x_3\}$), we need to calculate $SPQR(G+F+x_1'x_3)$, which can be done again in $O(|V|)$ running time by Lemma 2.11. We can also update the SPQR-tree in $O(|V|)$ time by the same consideration. Therefore, Step **7'** and hence every loop of Step **2** can be executed in $O(|V|)$ running time. Thus, the running time of the whole algorithm is indeed $O(|V|^2)$.

Recall that we assumed G being a graph with at least k^2+2 vertices in this whole section due to technical reasons. Since there are constantly many graphs with at most k^2+1 vertices, the solution of the problem can be provided in constant (although potentially considerable) time. Hence finally we can conclude the following theorem.

Theorem 5.9. *Let $G=(V,E)$ be a simple (k,ℓ)-rigid graph where $k<\ell\leq\frac{3}{2}k$. Then Problem 1 can be solved in $O(|V|^2)$ time.* $\square$

Simple outputs. Recall that in the original problems motivated by rigidity theory, mentioned in the Introduction, we also need to ensure that the augmenting edges are not parallel to any edge of the input graph G. Note that Lemmas 4.4 and 4.11 ensure that the output of our algorithm fulfills this requirement when the augmenting edge set has cardinality at least two. Moreover, it is also clear that a singleton augmenting edge set for Problem 1 must be formed by an edge which is not parallel to any edge of G, when the input G is not $(c_{k,\ell}+1)$-connected. When G is $(c_{k,\ell}+1)$ connected though, as it is explained in [25], Problem 1 is equivalent to finding a minimum cardinality edge set F, for which $G+F$ is (k,ℓ)-redundant. Hence both problems may be formulated as follows.

Let $G=(V,E)$ be a (k,ℓ)-rigid graph which is not (k,ℓ)-redundant, and let e' be an edge parallel to an edge e in E, such that $G+e'$ is (k,ℓ)-redundant. Find an edge f, which is not parallel to any edge in E, such that $G+f$ is also (k,ℓ)-redundant.

This problem was solved in [26] with running time $O(|V|^3)$, when the input is (k,ℓ)-tight. Here with a new idea we reduce this running time to $O(|V|^2)$ and we also sketch how the same idea can be used for (k,ℓ)-rigid inputs. The main idea of [26] was that the complete graph on sufficiently many (that is, at least $2k+1$) vertices is (k,ℓ)-redundant. Thus if we add the set of non-edges $\bar{E}$ to G, then it becomes (k,ℓ)-redundant (regardless whether G was (k,ℓ)-tight or (k,ℓ)-rigid).

Let $G'=(V,E')$ be a (k,ℓ)-tight spanning subgraph of the (k,ℓ)-rigid input graph $G=(V,E)$ so that $e\in E'$ holds for the edge e specified in the problem. Let us assume that $|V|\geq 2k+1$ and hence the complete graph K_V on V is (k,ℓ)-redundant. Note that all edges in $E-E'$ are (k,ℓ)-redundant in G, however, $E'\neq\mathcal{R}_{G'}(E-E')=\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')$ by Lemma 4.1, as G is not (k,ℓ)-redundant. Let $\bar{E}$ denote the set of the non-edges of G and $\bar{E}'$ the set of non-edges of G'. By Lemma 4.1 and our assumption that the complete graph on V is (k,ℓ)-redundant, $E'=\mathcal{R}_{G'}(\bar{E}')=\bigcup_{\bar{e}\in\bar{E}'}\mathcal{T}_{G'}(\bar{e})$. Thus there exists an $f\in\bar{E}'$ such that $e\in\mathcal{T}_{G'}(f)$. Now remember that e and e' are parallel. As $G+e'$ is (k,ℓ)-redundant, $E'=\mathcal{T}_{G'}(e')\cup\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')=\mathcal{T}_{G'}(e)\cup\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')\subseteq T_{G'}(f)\cup\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')$ by Lemma 2.1. Therefore, $G'+(E-E')+f$ and hence $G+f$ is (k,ℓ)-redundant. Here $f\in\bar{E}$ since otherwise $f\in E-E'$ (as $f\in\bar{E}'$), and $T_{G'}(f)\cup\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')=\bigcup_{e''\in E-E'}\mathcal{T}_{G'}(e'')\neq E'$ by our observation above.

To find f, one needs to check $\mathcal{T}_{G'}(\bar{e})$ for all $\bar{e}\in\bar{E}$ which needs $O(|\bar{E}||V|)=O(|V|^3)$ time. This time can be reduced by giving a subset $F\subseteq\bar{E}$ of cardinality $O(|V|)$ for which $G+F$ is also (k,ℓ)-redundant, as in this case the running time can be reduced to the time that we need to calculate F plus $O(|F||V|)=O(|V|^2)$ (which time is needed to calculate $\mathcal{T}_G(f)$ for all $f\in F$). The following lemma shows that a maximal

(k,ℓ)-sparse subset of $\bar{E}$ fulfills this aim, moreover, it can be calculated in $O(|V|^2)$ time by Section 2.1.

Lemma 5.10. *Let $G=(V,E)$ be a (k,ℓ)-rigid graph on at least $2k+1$ vertices and let $\bar{E}$ denote the complement set of E on V. Suppose that $H=(V,F)$ is a maximal (k,ℓ)-sparse subgraph of $\bar{G}=(V,\bar{E})$. Then $G+H$ is (k,ℓ)-redundant.*

Proof. Since G is (k,ℓ)-rigid, we only need to show that each edge $e\in E$ is (k,ℓ)-redundant in $G+H$. Let us take an edge $e\in E$ and let $r_{(k,\ell)}$ denote the rank function of the (k,ℓ)-sparsity matroid. As H is a maximal (k,ℓ)-sparse subgraph of $\bar{G}$, $r_{(k,\ell)}(F)=r_{(k,\ell)}(\bar{E})$. Hence $r_{(k,\ell)}(E-e\cup F)=r_{(k,\ell)}(E-e\cup\bar{E})\geq r_{(k,\ell)}(K_V-e)$ where K_V is the edge set of the complete graph on V. Since K_V is clearly redundantly rigid if $|V|\geq 2k+1$, $r_{(k,\ell)}(K_V-e)=k|V|-\ell$ and hence $G+H-e$ is (k,ℓ)-rigid which completes our proof. □

Recall the characterizations of rigidity and global rigidity of several types of frameworks from the Introduction. We have seen above that the algorithms of Theorems 5.7 and 5.8 can be modified in such a way that the output edge set has no edge parallel to an edge of the input, which has the following consequence.

Corollary 5.11. *If $G=(V,E)$ is a graph which is generically rigid in the Euclidean plane, in a non-Euclidean plane, or on the cylinder, then there exists an algorithm which calculates in $O(|V|^2)$ time a minimum edge set F for which $G+F$ is generically globally rigid in the Euclidean plane, in a non-Euclidean plane, or on the cylinder, respectively.* □

6 Concluding remarks

Non-rigid inputs. When the input is not (k,ℓ)-rigid, we can still get a 2-approximation for Problem 1 by the following method of [25]. Take first an optimal (k,ℓ)-rigid augmentation of the input graph $G=(V,E)$ first and then solve Problem 1 on this (k,ℓ)-rigid graph. The optimal (k,ℓ)-rigid augmentation can be found by trying to add extra edges to a maximal (k,ℓ)-sparse subgraph of G with the algorithm of Section 2.1. Since the complete graph on sufficiently many vertices is (k,ℓ)-rigid, this way we can find an optimal (k,ℓ)-rigid augmentation in $O(|V|^2)$ time. Hence the whole 2-approximation algorithm for this extended problem also needs $O(|V|^2)$ time.

The pinning problem in $\mathbb{R}^2$. Problems, which are closely related to the augmentation problems presented in this paper are the pinning problems from rigidity theory. In these problems, our goal is to achieve the rigidity or global rigidity of our framework by anchoring (that is, fixing the location) of a minimum set of vertices. The anchoring procedure can be modeled as the addition of a complete graph to the set of anchored vertices. This type of problems can be particularly useful in case of localization, as anchoring/localizing any point also provides valuable information on the location of the whole framework. It was shown in [26] that the transversal of the (k,ℓ)-MCT sets of a (k,ℓ)-tight graph $G=(V,E)$ is an optimal pinning set for (k,ℓ)-redundancy.

Furthermore, by the results in [25], the transversal of the atoms of a (k, ℓ)-rigid graph $G = (V, E)$ is an optimal pinning set for (k, ℓ)-redundancy and $(c_{k,\ell} + 1)$-connectivity. Hence by Lemma 5.4 this optimum can also be found in $O(|V|^2)$ time, and hence the global rigidity pinning problem can be solved in $O(|V|^2)$ time for rigid generic frameworks in the plane with Euclidean or non-Euclidean norm, and on the cylinder.

Acknowledgments

Projects no. NKFI-128673, K-135421, and PD-138102 have been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Hungarian Scientific Research Fund FK_18, K_20, and PD_21 funding schemes. The first author was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the ÚNKP-20-5 and ÚNKP-21-5 New National Excellence Program of the Ministry for Innovation and Technology. The second author was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002). The authors are grateful to Tibor Jordán for the inspiring discussions and his comments.

References

[1] L. Asimow and B. Roth. The rigidity of graphs. *Transactions of the American Mathematical Society*, 245:279–289, 1978.

[2] A.R. Berg. *Rigidity of Frameworks and Connectivity of Graphs.* PhD thesis, Department of Computer Science, University of Aarhus, 2004.

[3] A.R. Berg and T. Jordán. Algorithms for graph rigidity and scene analysis. In G. Di Battista and U. Zwick, editors, *Algorithms - ESA 2003*, volume 2832 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2003.

[4] S. Dewar, J. Hewetson, and A. Nixon. Uniquely realisable graphs in analytic normed planes, 2022.

[5] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In M.S. Paterson, editor, *Automata, Languages and Programming*, pages 598–611, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[6] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996.

[7] K.P. Eswaran and R.E. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976.

[8] Zs. Fekete and T. Jordán. Uniquely localizable networks with few anchors. In S.E. Nikoletseas and J.D.P. Rolim, editors, *Algorithmic Aspects of Wireless Sensor Networks*, pages 176–183, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[9] A. Frank. *Connections in Combinatorial Optimization.* Oxford University Press, 2011.

[10] A. Frank, T. Király, and Z. Király. On the orientation of graphs and hypergraphs. *Discrete Applied Mathematics*, 131(2):385–400, 2003.

[11] D. Garamvölgyi, T. Jordán, and Cs. Király. Count and cofactor matroids of highly connected graphs. Technical Report TR-2022-12, Egerváry Research Group, Budapest, 2022. `egres.elte.hu`.

[12] A. García and J. Tejel. Augmenting the rigidity of a graph in $\mathbb{R}^2$. *Algorithmica*, 59(2):145–168, 2011.

[13] A.V. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. In *STOC*, pages 136–146. ACM, 1986.

[14] S.J. Gortler, A.D. Healy, and D.P. Thurston. Characterizing generic global rigidity. *American Journal of Mathematics*, 132(4):897–939, 2010.

[15] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *Graph Drawing 2000*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2001.

[16] S.L. Hakimi. On the degrees of the vertices of a directed graph. *J. Franklin Inst.*, 279(4):290–308, 1969.

[17] J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372—378, jun 1973.

[18] J.E. Hopcroft and R.E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[19] T.S. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. *Annual Symposium on Foundations of Computer Science (Proceedings)*, pages 548–559, 1991.

[20] B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. *J. Comb. Theory, Ser. B*, 94:1–29, 2005.

[21] B. Jackson and A. Nixon. Global rigidity of generic frameworks on the cylinder. *J. Comb. Theory, Ser. B*, 139:193–229, 2019.

[22] D.J. Jacobs and B. Hendrickson. An algorithm for two dimensional rigidity percolation: The pebble game. *Journal of Computational Physics*, 137:346–365, 1997.

[23] D.J. Jacobs and M.F. Thorpe. Generic rigidity percolation: The pebble game. *Phys. Rev. Lett.*, 75:4051–4054, Nov 1995.

[24] T. Jordán. Combinatorial rigidity: Graphs and matroids in the theory of rigid frameworks. In *Discrete Geometric Analysis*, volume 34 of *MSJ Memoirs*, pages 33–112. Mathematical Society of Japan, Japan, 2016.

[25] Cs. Király and A. Mihálykó. Globally rigid augmentation of rigid graphs. *SIAM Journal on Discrete Mathematics*, 36(4):2473–2496, 2022.

[26] Cs. Király and A. Mihálykó. Sparse graphs and an augmentation problem. *Math. Program.*, 192(1):119–148, 2022.

[27] D. Kitson and S.C. Power. Infinitesimal rigidity for non-Euclidean bar-joint frameworks. *Bulletin of the London Mathematical Society*, 46(4):685–697, 04 2014.

[28] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engineering Mathematics*, 4:331–340, 1970.

[29] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–37, 2008.

[30] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proceedings of the 17th Canadian Conference on Computational Geometry, CCCG'05, University of Windsor, Ontario, Canada, August 10-12, 2005*, pages 219–222, 2005.

[31] M. Lorea. On matroidal families. *Discrete Mathematics*, 28(1):103 – 106, 1979.

[32] C.St.J.A. Nash-Williams. Decomposition of finite graphs into forests. *J. London Math. Soc.*, 39:12, 1961.

[33] A. Nixon, J.C. Owen, and S.C. Power. Rigidity of frameworks supported on surfaces. *SIAM Journal on Discrete Mathematics*, 26(4):1733–1757, 2012.

[34] H. Pollaczek-Geiringer. Über die Gliederung ebener Fachwerke. *ZAMM - Journal of Applied Mathematics and Mechanics*, 7(1):58–72, 1927.

[35] A. Rosenthal and A. Goldner. Smallest augmentations to biconnect a graph. *SIAM Journal on Computing*, 6(1):55–66, 1977.

[36] I. Streinu and L. Theran. Sparse hypergraphs and pebble game algorithms. *European Journal of Combinatorics*, 30(8):1944–1964, 2009.

[37] W. T. Tutte. *Connectivity in graphs*, volume 15 of *Mathematical Expositions*. University of Toronto Press, 1966.

[38] W. Whiteley. Some matroids from discrete applied geometry. In J.E. Bonin, J.G. Oxley, and B. Servatius, editors, *Matroid Theory*, volume 197 of *Contemporary Mathematics*, pages 171–311. AMS, 1996.

7 Appendix: Making (k,ℓ)-M-component hypergraphs (k,ℓ)-redundant

Let G be a (k,ℓ)-rigid graph and $\mathcal{H}_G = (V,\mathcal{E})$ its (k,ℓ)-M-component hypergraph. In this section we present an algorithm with $O(|V|^2)$ running time that finds a minimum cardinality edge set F such that $\mathcal{H}_G + F$ is (k,ℓ)-redundant. This gives an alternative proof to Theorem 4.8. If G is a (k,ℓ)-tight graph, then $\mathcal{H}_G$ is a (k,ℓ)-tight graph that coincides with G. Hence the algorithm presented below can also be used to find an optimal solution for the (k,ℓ)-redundant rigidity augmentation problem.

Throughout this section, *G is a (k,ℓ)-rigid graph on at least k^2+2 vertices, where k and ℓ are positive integers such that $\ell < 2k$. $\mathcal{H}_G$ denotes the (k,ℓ)-M-component hypergraph of G, G^* is a (k,ℓ)-tight spanning subgraph of G and, if not stated otherwise, $\mathcal{T}(ij)$ and $V(ij)$ denote $\mathcal{T}_{\mathcal{H}_G}(ij)$ and $V_{\mathcal{H}_G}(ij)$ for $i,j \in V$, respectively.*

We start with running the algorithm of Section 3 on G. As we showed it in Theorem 3.4, this method generates an auxiliary orientation $\vec{\mathcal{H}}_G$ of the (k,ℓ)-M-component hypergraph that can be used to determine $\mathcal{T}(ij)$ for arbitrary $i,j \in V$ in $O(|V|)$ running time. Our algorithm will rely on Algorithm 5.1. Recall that the output of Algorithm 5.1 is denoted by $V'(i,L)$.

Observation 7.1. *If $V'(i,L) = \{j\}$ for a vertex $j \in V$, then $\mathcal{T}(ij) = \mathcal{H}_G$, and hence $\mathcal{H}_G + ij$ is (k,ℓ)-redundant.* □

The following lemma will be used several times.

Lemma 7.2. *Let $i \in V$, such that $d_{G^*}(i) \leq 2k-1$. Suppose that $\mathcal{T}_1$ and $\mathcal{T}_2$ are two (k,ℓ)-tight subhypergraphs of $\mathcal{H}_G$ such that $i \in V(\mathcal{T}_1) \cap V(\mathcal{T}_2)$ and $|V(\mathcal{T}_1)| \geq 3$, $|V(\mathcal{T}_2)| \geq 3$. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a (k,ℓ)-tight subhypergraph of $\mathcal{H}_G$ and and $d_{\mathcal{H}_G}(V(\mathcal{T}_1) - V(\mathcal{T}_2), V(\mathcal{T}_2) - V(\mathcal{T}_1)) = 0$.*

Proof. By Lemma 2.6, $G^*[V(\mathcal{T}_1)]$ and $G^*[V(\mathcal{T}_2)]$ are (k,ℓ)-tight subgraphs of G^*. By Lemma 2.2 i has degree at least k in both of them implying that $\mathcal{T}_1$ and $\mathcal{T}_2$ have a common edge incident with i and hence $1 \leq i(V(\mathcal{T}_1) \cap V(\mathcal{T}_2)) \leq k|V(\mathcal{T}_1) \cap V(\mathcal{T}_2)| - \ell$. Now our statement follows by Lemma 2.1. □

An output $V'(i,L)$ is called **simple**, if $|V(i,j)| \geq 3$ for every $j \in V'(i,L)$. Let us show now how Algorithm 5.1 can be used with a simple output to determine, if there is an edge uv for which $\mathcal{H}_G + uv$ is (k,ℓ)-redundant.

Lemma 7.3. *Let $i \in V$, such that $d_{G^*}(i) \leq 2k-1$ and let $V'(i,L)$ be the output of Algorithm 5.1 with inputs G and i (and an arbitrary subset L of V). Suppose that there is an edge uv so that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant. Suppose moreover that $V'(i,L)$ is simple and $|V'(i,L)| \geq 2$. Then there exists $j_1, j_2 \in V - i$ such that $V'(i,L) = \{j_1, j_2\}$, and there exists a vertex $y \in V$ such that $\mathcal{T}(j_1 y) = \mathcal{H}_G$ or $\mathcal{T}(j_2 y) = \mathcal{H}_G$.*

Proof. Let $j_1, j_2 \in V'(i,L)$ be two vertices for which $u \in V(ij_1)$ and $v \in V(ij_2)$ for the two endvertices of the edge uv. Note that $v \notin V(ij_1)$ since otherwise $\mathcal{H}_G = \mathcal{T}(uv) \subseteq \mathcal{T}(ij_1)$ would hold by Lemma 2.3, contradicting $|V'(i,L)| \geq 2$. By the run of Algorithm 5.1 $j_2 \notin V(ij_1)$. Similarly one can prove that $u, j_1 \notin V(ij_2)$. As $V'(i,L)$ is simple, both $V(ij_1)$ and $V(ij_2)$ have cardinality at least three. Hence Lemma 7.2 implies that $\mathcal{T}(ij_1)\cup\mathcal{T}(ij_2)$ is (k,ℓ)-tight and $d_{\mathcal{H}_G}(V(ij_1)-V(ij_2), V(ij_2)-V(ij_1)) = 0$. Since $\mathcal{T}(ij_1) \cup \mathcal{T}(ij_2)$ induces both u and v, $\mathcal{H}_G = \mathcal{T}(uv) \subseteq \mathcal{T}(ij_1) \cup \mathcal{T}(ij_2)$ by Lemma 2.3. Hence $V'(i,L) = \{j_1, j_2\}$ as the first statement of the lemma claimed.

In case of $|V(ij_1) - V(ij_2)| = 1$ (or $|V(ij_2) - V(ij_1)| = 1$, respectively), $u = j_1$ (or $v = j_2$, respectively) hence the last statement is obviously true by having $y = v$ (or $y = u$, respectively). Thus we may suppose that $|V(ij_1) - V(ij_2)| \geq 2$ and $|V(ij_2) - V(ij_1)| \geq 2$. As $d_{\mathcal{H}_G}(V(ij_1) - V(ij_2), V(ij_2) - V(ij_1)) = 0$, no (graph) edge connects j_1 to j_2 in $\mathcal{H}_G$ and hence $\mathcal{H}_G[\{j_1, j_2\}]$ is not (k,ℓ)-tight. Thus $|V(j_1j_2)| \geq 3$. As $V(ij_1) \cup V(ij_2) = V$, $V(j_1j_2)$ intersects at least one of $V(ij_1)$ and $V(ij_2)$ (say, $V(ij_2)$) in at least 2 vertices. Hence, by Lemma 2.1, $\mathcal{T}(ij_1) \cup \mathcal{T}(j_1j_2)$ is (k,ℓ)-tight, containing i and j_2. Thus $\mathcal{T}(ij_2) \subseteq \mathcal{T}(ij_1) \cup \mathcal{T}(j_1j_2)$ by Lemma 2.3. Consequently, $V(j_1j_2) \supseteq V(ij_2) - V(ij_1)$ and similarly $V(j_1j_2) \supseteq V(ij_1) - V(ij_2)$. This implies that $u, v \in V(j_2j_3)$, resulting $\mathcal{H}_G = \mathcal{T}(uv) \subseteq \mathcal{T}(j_1j_2)$ by Lemma 2.3, as we claimed. □

Lemma 7.4. *Let $i \in V$, such that $d_{G^*}(i) \leq 2k-1$ and let $V'(i,L)$ be the output of Algorithm 5.1 with inputs G and i (and an arbitrary subset L of V). Suppose that there is no edge uv such that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant and that $V'(i,L)$ is simple. Then every vertex of $V'(i,L)$ is contained in a (k,ℓ)-MCT set.*

Proof. For any $v \in V'(i,L)$, the sequential applications of Lemma 7.2 imply that if $\bigcup_{j\in V'(i,L)-v} \mathcal{T}(ij)$ is (k,ℓ)-tight (as $|V'(i,L)| \geq 2$ by Observation 7.1). Hence $V - \bigcup_{j\in V'(i,L)-v} V(ij)$ is a (k,ℓ)-co-tight set which contains v and has a (k,ℓ)-MCT subset C which has no vertex from $V'(i,L) \cup \{i\}$. Now as $\bigcup_{j\in V'(i,L)} \mathcal{T}(ij) = \mathcal{H}_G$, Observation 4.2 and Lemma 4.1 imply that $v \in C$ is from a (k,ℓ)-MCT set. □

Let us consider now the case, when $V'(i,L)$ is not simple.

Lemma 7.5. *Let $i \in V$, such that $d_{G^*}(i) \leq 2k-1$ and let $V'(i,L)$ be the output of Algorithm 5.1 with inputs G and i (and an arbitrary subset L of V). Suppose that $V'(i,L)$ is not simple and $|V'(i,L)| \geq 2$. Let $N := \{j : j \in V'(i,L) \text{ and } |V(ij)| = 2\}$. If there exists an edge uv such that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant, then there exists a vertex $n \in N$ and a vertex $y \in V$ for which $\mathcal{T}(ny) = \mathcal{H}_G$. Otherwise, there exists a vertex in N which is contained in a (k,ℓ)-MCT set of $\mathcal{H}_G$.*

Proof. As $V'(i,L)$ is not simple, N is non-empty. Suppose first that there exists an edge uv so that $\mathcal{H}_G + uv$ is (k,ℓ)-redundant. If $u \in N$ or $v \in N$, the statement obviously holds. Otherwise, there are the vertices $j_1, j_2 \in V'(i,L)$ for which $u \in V(ij_2)$, $v \in V(ij_3)$, $|V(ij_2)| \geq 3$, and $|V(ij_3)| \geq 3$. In this case, similarly to the proof of Lemma 7.3, $T(ij_1) \cup T(ij_2) = \mathcal{H}_G$ by Lemma 7.2. This contradicts $N \neq \emptyset$.

If there is no edge uv for which $\mathcal{H}_G + uv$ is (k,ℓ)-redundant, then let $\mathcal{T} = \bigcup\{\mathcal{T}(ij) : j \in V'(i,L) - N\}$. Now $\mathcal{T}$ is non-empty by our assumption that $|V| \geq k^2 + 2$ and the fact that $|N| \leq 2k-1$ since $d_{G^*}(i) \leq 2k-1$. Hence $\mathcal{T}$ is a (k,ℓ)-tight subhypergraph of $\mathcal{H}_G$ by the sequential application of Lemma 7.2, and thus N is a (k,ℓ)-co-tight set of $\mathcal{H}_G$. Hence N has a (k,ℓ)-MCT subset and thus a vertex of N is included in a (k,ℓ)-MCT set of $\mathcal{H}_G$. □

Now, by these results, we can construct the core algorithm to solve the (k,ℓ)-redundant rigidity augmentation problem on (k,ℓ)-M-component hypergraphs.

Algorithm 7.6. INPUT: *(k,ℓ)-rigid graph G on at least k^2+2 vertices and a vertex set $L \subseteq V$.*
OUTPUT: *If there exists an edge e, for which $\mathcal{H}_G + e$ is (k,ℓ)-redundant, then an edge ij for which $\mathcal{H}_G + ij$ is (k,ℓ)-redundant. Otherwise, a transversal vertex set P on the (k,ℓ)-MCT sets of $\mathcal{H}_G$, so that $|L \cap P|$ is maximal amongst all the transversals of the (k,ℓ)-MCT sets.*

```
 1 Run the algorithm of Theorem 3.4 on $G$. Get the $(k,\ell)$-M-component hypergraph
   $\mathcal{H}_G$ and a $(k,\ell)$-tight spanning subgraph $G^*$ of $G$.
 2 Choose a vertex $i$ with minimum degree in $G^*$.
 3 Run Algorithm 5.1 with $i$ and $L$, resulting $V'(i,L)$.
 4 If $V'(i,L) = \{j\}$, then
       Return the edge $ij$
 5 If $V'(i,L) = \{j_1, j_2\}$, then
       Check if $\mathcal{T}(j_1 v) = \mathcal{H}_G$ or $\mathcal{T}(j_2 v) = \mathcal{H}_G$ for every $v \in V$.
       If there exists such an edge $jv$, then
           Return $jv$.
 6 If $V'(i,L)$ is not simple, then
           $N := \{v | v \in V'(i,L)$ and $|V(iv)| = 2\}$
   else
           $N := \{v\}$ for any $v \in V'(i,L)$.
 7 For every $i' \in N$,
           Run Algorithm 5.1 with $i'$ and $L$ resulting $V'(i',L)$.
 8 Choose $i_0 = \text{argmin}\{|V'(i',L)| : i' \in N\}$.
 9 Run Algorithm 5.1 with $i_0$ and $L$ resulting $V'(i_0,L)$. Take $i_1 \in V'(i_0,L)$.
10 If $V'(i_0,L) = \{i_1\}$, then
       Return the edge $i_0 i_1$.
11     else
           Run Algorithm 5.1 with $i_1$ and $L$ resulting $V'(i_1,L)$.
           Return $P := V'(i_1,L) \cup \{i_1\}$.
```

Lemma 7.7. *Let $G = (V,E)$ be a (k,ℓ)-rigid graph on at least k^2+2 vertices, where $0 < \ell < 2k$, and let $\mathcal{H}_G$ be its (k,ℓ)-M-component hypergraph. Then Algorithm 7.6 returns either one edge ij, for which $\mathcal{H}_G + ij$ is (k,ℓ)-redundant, or a transversal vertex set P on the (k,ℓ)-MCT sets of $\mathcal{H}_G$ such that $|P \cap L|$ is maximum. The running time of Algorithm 7.6 is $O(|V|^2)$.*

Proof. Let G^* be the (k,ℓ)-tight subgraph of G that we get in STEP **1** of Algorithm 7.6. By Lemma 2.2, G^* has a vertex i so that $d_{G^*}(i) \leq 2k-1$. Hence $d_{G^*}(i) \leq 2k-1$ holds for the vertex i chosen in STEP **2**.

By Observation 7.1 and Lemmas 7.3 and 7.5, the algorithm returns with one edge in STEP **4**, **5** or **10** if and only if exists an edge for which $\mathcal{H}_G + e$ is (k,ℓ)-redundant. Hence we may assume, that no single edge addition can make $\mathcal{H}_G$ (k,ℓ)-redundant.

Thus Lemmas 7.4 and 7.5 imply that at least one vertex $i^* \in N$ is from a (k,ℓ)-MCT set of $\mathcal{H}_G$. Now, $V'(i^*,L) \cup \{i^*\}$ is a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$ by Lemma 5.3. Note that $V'(v,L) \cup \{v\}$ must intersect each (k,ℓ)-MCT set C for arbitrary choice of $v \in V$ since otherwise $V(vj) \subseteq V - C$ holds for each $j \in V'(i,L)$ by Lemma 2.3, contradicting its construction. Hence the choice of i_0 in Step **8** indeed results a vertex from a (k,ℓ)-MCT set.

Now Lemma 5.3 implies that the vertex $i_1 \in V'(i_0,L)$ choosen in Step **9** is contained by a (k,ℓ)-MCT set C of $\mathcal{H}_G$, moreover, if $C \cap L \neq \emptyset$, then $i_1 \in L$. This implies by using Lemma 5.3 again that P is a transversal of the the (k,ℓ)-MCT sets of $\mathcal{H}_G$, moreover, $|L \cap P|$ is the maximum amongst all the transversals of the (k,ℓ)-MCT sets.

Step **1** has $O(|V|^2)$ running time by Theorem 3.4. We can easily find i in $O(|V|^2)$ running time (in fact, $O(|V|)$ time is enough). Step **3** runs in $O(|V|^2)$ time by Lemma 5.2. As we compute $\mathcal{T}(ij)$ only $O(|V|)$ times in Step **5**, this can be executed in $O(|V|^2)$ total time by Theorem 3.4. As $d_{G^*}(i) \leq 2k-1$, $|N| \leq (2k-1)/(2k-\ell)$, hence $|N| = O(1)$ and Steps **7**–**8** run in $O(|V|^2)$ time by Lemma 5.2. Steps **9** and **11** need $O(|V|^2)$ running time again by Lemma 5.2. Therefore, the total running time of Algorithm 7.6 is $O(|V|^2)$. □

After running Algorithm 7.6 (with $L = \emptyset$), Lemma 7.7 implies that we get either an edge e, for which $\mathcal{H}_G + e$ is (k,ℓ)-redundant and hence is an optimal augmentation, or a set $P = V'(i_1,\emptyset) \cup \{i_1\}$ which is a transversal of the (k,ℓ)-MCT sets of $\mathcal{H}_G$. By Lemma 4.5, $\mathcal{H}_G + \{i_1 j : j \in V'(i_1,\emptyset)\}$ is (k,ℓ)-redundant. Next we can use Lemma 4.6, like it is used in [26], to algorithmically reduce this augmenting edge set to the size of $\left\lceil \frac{|P|}{2} \right\rceil$ which is optimal by Theorem 4.3. Since $\mathcal{T}(ij)$ can be calculated in $O(|V|)$ time by Theorem 3.4 and P has size $O(|V|)$, the running time of this algorithm is $O(|V|^2)$. This finishes the proof of Theorem 4.8.

Finally, we note that in [26] the redundant rigidity augmentation problem was solved for so-called (m,ℓ)-tight inputs (where we have a map $m : V \to \mathbb{Z}_+$ and the sparsity condition $i(X) \leq k|X| - \ell$ is substituted by $i(X \leq \sum_{x \in X} m(x) - \ell)$, as this extension was needed to solve the problem for (k,ℓ)-rigid inputs when $k \geq \ell$. We note that the algorithm presented above can be extended for this concept. The main difference is that, instead of a minimum degree vertex (which had degree less than $2k$ for (k,ℓ)-tight inputs), we need to take an initial vertex v which has degree less than $2m(v)$. Since $|E| = \sum_{v \in V} m(v) - \ell$ in an (m,ℓ)-tight graph, such a vertex can be found when $\ell > 0$. Finally, we also note that the algorithm presented in Section 3 can be extended to calculate the (m,ℓ)-M-component hypergraph.