Computing Circuit Polynomials in the Algebraic Rigidity Matroid

Goran Malić
Smith College

Ileana Streinu
Smith College, istreinu@smith.edu

SIAM J. APPL. ALGEBRA GEOMETRY
Vol. 7, No. 2, pp. 345–385

Computing Circuit Polynomials in the Algebraic Rigidity Matroid[*]

Goran Malić[†] and Ileana Streinu[†]

Abstract. We present an algorithm for computing *circuit polynomials* in the algebraic rigidity matroid $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$ associated to the Cayley–Menger ideal CM_n for n points in 2D. It relies on *combinatorial resultants*, a new operation on graphs that captures properties of the Sylvester resultant of two polynomials in this ideal. We show that every rigidity circuit has a *construction tree* from K_4 graphs based on this operation. Our algorithm performs an *algebraic elimination* guided by such a construction tree and uses classical resultants, factorization, and ideal membership. To highlight its effectiveness, we implemented the algorithm in *Mathematica*: it took less than 15 seconds on an example where a Gröbner basis calculation took 5 days and 6 hours. Additional speed-ups are obtained using non-K_4 generators of the Cayley–Menger ideal and simple variations on our main algorithm.

Key words. Cayley–Menger ideal, rigidity matroid, circuit polynomial, combinatorial resultant, inductive construction, Gröbner basis elimination

MSC codes. 05B35, 13P15, 52C25, 14Q20, 51K05, 51K99, 68W30, 13P10

DOI. 10.1137/21M1437986

1. Introduction.

The focus of this paper is the following problem straddling combinatorial rigidity and algebraic matroids.

Main Problem. *Given a rigidity circuit, compute its corresponding circuit polynomial.*

Its motivation comes from the following ubiquitous problem in *distance geometry*.

Localization. A graph together with *weights* associated to its edges is given. The goal is to find *placements* for its vertices in some Euclidean space (2D, in our case), so that the resulting edge lengths match the given weights. To this purpose we set up a system of quadratic equations with unknowns corresponding to the Cartesian coordinates of the vertices. The possible *placements* (or *realizations*) are among its (real) solutions and can be found with numerical methods (see, e.g., [36, 49, 3]). A related problem is to look for the possible values of a *single unknown distance* corresponding to a *nonedge* (a pair of vertices that are not connected by an edge). If we could solve this second problem for a collection of nonedge pairs that, together with the original edges, contain a trilateration, then one placement for the graph could be obtained afterwards in linearly many steps of quadratic equation solving.

[*]Received by the editors August 2, 2021; accepted for publication (in revised form) February 13, 2023; published electronically May 23, 2023. This paper extends the conference abstract 37th *International Symposium on Computational Geometry* (SoCG 2021), Leibniz Internat. Proc. Inform. (LIPIcs) 189, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 52:1–52:16, where the main result was announced, and includes results from the preprint *Faster Algorithms for Circuits in the Cayley-Menger Algebraic Matroid*, 2021.
https://doi.org/10.1137/21M1437986
Funding: The work of the second author was supported by the NSF through grants CCF:1703765 and CCF:2212309.

[†]Computer Science Department, Smith College, Northampton, MA 01063 USA (gmalic@smith.edu, http://www.goranmalic.com; istreinu@smith.edu, streinu@cs.umass.edu, http://cs.smith.edu/~istreinu).

Rigidity circuits. The *generic* version of the *single unknown distance* problem, where the weights are symbols rather than concrete numbers, is amenable to techniques from rigidity theory. In 2D, one can predict whether, generically, the set of solutions for the unique unknown distance will be discrete (if the given graph is *rigid*) or continuous (if the graph is *flexible*). We formulate the problem algebraically by using Cayley coordinates $X_n = \{x_{ij} : 1 \leq i < j \leq n\}$, with x_{ij} denoting the squared distance between vertices i and j, and n being the number of vertices. There are certain dependencies between these variables, captured by the polynomials $f \in \mathbb{Q}[X_n]$ generating the Cayley–Menger ideal. When G is a minimally rigid graph, the addition of a new edge e induces a unique subgraph $C \subseteq G \cup \{e\}$, which is a *circuit* in the 2D rigidity matroid whose bases are the minimally rigid graphs. There also exists a unique (up to multiplication by a scalar) polynomial dependency p_C between the distances corresponding to the edges of C. This is a *circuit polynomial* in the Cayley–Menger ideal and is the main object of study in this paper. The unique unknown distance problem is solved by substituting in this circuit polynomial concrete values for the edge weights of G and then computing the roots of the resulting univariate polynomial.

How tractable is the problem? Circuit polynomial computations can be done, in principle, by using the Gröbner basis algorithm with an elimination order.[1] In the worst case, this is a doubly exponential method, but in practice the complexity and performance of Gröbner basis algorithms depends heavily on the choice of a *monomial order*. There exist known cases, e.g., zero-dimensional polynomial ideals [15, 32], which have single-exponential complexity with respect to any monomial order. However, *elimination orders* have been reported to behave badly. In general, the main problems of elimination theory, such as the Ideal Triviality Problem, the Ideal Membership Problem for Complete Intersections, the Radical Membership Problem, the General Elimination Problem, and the Noether Normalization, are in the PSPACE complexity class [40].

In our experimentation, the GroebnerBasis function of *Mathematica* 12 (running on a 2019 iMac computer with 6 cores at 3.6 Ghz) took 5 days and 6 hours to compute the Desargues-plus-one circuit (a graph on 6 vertices) reported in Table 1 of section 13, but in most cases it timed out or crashed.

Overview of results. Our goal is to make such calculations *more tractable* by taking advantage of *structural information* inherent in the problem. We describe a new *algorithm to compute a circuit polynomial with known support*. It relies on resultant-based elimination steps guided by a novel *inductive construction for rigidity circuits*. Inductive constructions have been often used in rigidity theory, most notably the Henneberg sequences for Laman graphs [27] and Henneberg II sequences for 3-connected rigidity circuits [5]. We argue that our combinatorial construction is more *natural* due to its direct algebraic interpretation, a property not shared with any of the other previously known constructions. We have implemented our method in *Mathematica* and applied it successfully to compute all but one of the circuit polynomials on up to 6 vertices, as well as a few on 7 and 8 vertices, the largest of which having over nine million terms. The previously mentioned example of the Desargues-plus-one circuit that took over 5 days to complete with GroebnerBasis, was solved by our algorithm in less than 15 seconds.

[1]See Exercises 5 and 6 in [13, Chapter 3, section 1].

The only example on 6 vertices that remained elusive was the circuit polynomial for the $K_{3,3}$-plus-one circuit (see Table 1 in section 13): the computational resources for its computation far exceeded the capabilities of both our machines and of an HPC system we experimented with. We succeeded by extending the basic algorithm to work with additional generators of the Cayley–Menger ideal, besides those corresponding to K_4's. These are irreducible polynomials supported on dependent rigid graphs that are not necessarily circuits.

Related work. Our approach builds upon ideas from *distance geometry* and *rigidity theory* and combines them with the theory of algebraic matroids. The former enjoy a long and distinguished history—too long to survey here, but see [6, 14]. Combinatorial and linear (but not algebraic) matroids occupy a central place in rigidity theory [24, 55]. To the best of our knowledge, the study of circuit polynomials in *arbitrary* polynomial ideals was initiated in the Ph.D. thesis of Rosen [45]. His Macaulay2 code [46] is useful for exploring small cases, but the Cayley–Menger ideal is beyond its reach. A recent article [47] popularizes algebraic matroids and uses for illustration the smallest circuit polynomial K_4 in the Cayley–Menger ideal. *We could not find nontrivial examples anywhere*. Indirectly related to our problem are results such as [54], where an explicit univariate polynomial of degree 8 is computed (for an unknown angle in a $K_{3,3}$ configuration given by edge lengths, from which the placement of the vertices is determined), and [48], for its usage of Cayley coordinates in the study of configuration spaces of some families of distance graphs. A closely related problem is that of computing the *number of embeddings of a minimally rigid graph* [9], which has received a lot of attention in recent years (see, e.g., [11, 1, 19, 18], to name a few). References to specific results in the literature that are relevant to the theory developed here and to our proofs are given throughout the paper.

Overview of the paper. Our main theoretical result is split into a combinatorial theorem, Theorem 1, and an algebraic theorem, Theorem 2, each with an algorithmic counterpart and each preceded by a section introducing the concepts necessary for a self-contained presentation. Section 2 reviews 2D combinatorial rigidity matroids. Then in section 3 we define the *combinatorial resultant* of two graphs as an abstraction of the classical resultant, prove Theorem 1, and describe the algorithm for computing a *combinatorial circuit-resultant (CCR) tree*.

Theorem 1. *Each rigidity circuit can be obtained, inductively, by applying combinatorial resultant operations starting from K_4 circuits. The construction is captured by a binary resultant tree whose nodes are intermediate rigidity circuits and whose leaves are K_4 graphs.*

This leads to a *graph algorithm* for finding a *CCR tree* of a circuit. Each step of the construction can be carried out in polynomial time using variations on the *pebble game* matroidal sparsity algorithms [35] combined with Hopcroft and Tarjan's linear time 3-connectivity algorithm [28]. However, it is conceivable that the tree could be exponentially large, and thus the entire construction could take an exponential number of steps: understanding in detail the algorithmic complexity of our method *remains a problem for further investigation.*

In sections 4, 5, 6, and 7 we include a brief, self-contained overview of the algebraic concepts relevant to this paper: ideals and their algebraic matroids, the Cayley–Menger ideal, resultants, and the circuit polynomials in the Cayley–Menger ideal. In section 8 we prove the following theorem.

Theorem 2. *Each circuit polynomial can be obtained, inductively, by applying resultant operations. The procedure is guided by the combinatorial circuit-resultant (CCR) tree from*

Theorem 1 *and builds up from K_4 circuit polynomials. At each step, the resultant produces a polynomial that may not be irreducible. A polynomial factorization and a test of membership in the ideal are then applied to identify the factor which is the actual circuit polynomial.*

The algorithmic counterpart of Theorem 2 appears in section 9. Overall, the resulting *algebraic elimination algorithm* runs in exponential time, in part because of the growth in size of the polynomials that are being produced. Several theoretical *open questions* remain, whose answers may affect the precise time complexity analysis.

In section 10 we define and characterize a more general *combinatorial resultant tree* which generalizes the CCR tree by allowing more freedom in the choice of graphs used at the leaves of the tree: besides K_4 circuits, we now can use dependent rigid graphs. This extension allows the use of polynomials supported on dependent sets in the Cayley–Menger ideal that are not necessarily circuits. The dependent, noncircuit generators of the Cayley–Menger ideal are discussed in section 11, and the full generalization of our main algorithm is given in section 12.

The preliminary experimental results we carried out with the implementation of our method in *Mathematica* are discussed in section 13. We used *Mathematica* v13 on an 2019 iMac with the following specifications: Intel i5-9600K 3.7GHz, 16GB RAM, macOS Monterey 12.3.1. We also explored Macaulay2, but it was much slower than *Mathematica* (hours vs. seconds) in computing one of our examples. The resulting polynomials are made available on a GitHub repository [39].

Open questions are introduced throughout the paper and in the concluding remarks, section 14.

Further connections: Circuit polynomials in matroid theory. The matroid theory literature is rich in realizability questions of various sorts [43] and has seen in recent years a surge of interest in algebraic matroids. Ingleton [29] proved that algebraic matroids over fields of characteristic 0 are linearly realizable, but this is not the case in positive characteristic [43]. Recently, [7] identified an infinite class of algebraic matroids over fields of positive characteristic that have a linear representation in the same characteristic, namely, those for which the so-called Lindström valuation is trivial. The problem of computing the Lindström valuation was addressed in [12], where the fundamental step is to compute all circuit polynomials of a given algebraic matroid in positive characteristic. We remark that for the algebraic matroids whose combinatorial structure allows descriptions of their circuits in terms of an operation similar to our combinatorial resultants, the methods presented in this paper are applicable and likely to be more efficient than Gröbner basis methods.

Remark. The main results of this paper have been announced in the conference abstract [37] and in [38].

2. Preliminaries: Rigidity circuits.

We start with the combinatorial aspects of our problem and review the relevant notions and results from combinatorial rigidity theory of bar-and-joint frameworks in dimension 2.

Notation. We work with (sub)graphs given by subsets E of edges of the complete graph K_n on vertices $[n] := \{1, \ldots, n\}$. If G is a (sub)graph, then $V(G)$, respectively, $E(G)$, denotes its vertex set, respectively, edge set. The support of G is $E(G)$. The *vertex span* $V(E)$ of edges E is the set of all edge-endpoint vertices. A subgraph G is *spanning* if its edge set $E(G)$ spans $[n]$. The *neighbors* $N(v)$ of vertex v are the vertices adjacent to v in G.

Frameworks. A $2D$ *bar-and-joint framework* is a pair (G,p) of a graph $G=(V,E)$ and a *placement map* $p:V\to\mathbb{R}^2$. We view the edges as *rigid bars* and the vertices as *rotational joints* which allow the framework to deform continuously as long as the bars retain their original lengths. The *realization space* of the framework is the set of all of its possible placements in the plane with the same bar lengths. Two realizations are congruent if they are related by a planar isometry. The *configuration space* of the framework is made of congruence classes of realizations. The *deformation space of a given framework* (G,p) is the connected component of the configuration space that contains this particular placement (given by p). A framework is *rigid* if its deformation space consists of exactly one configuration, and is *flexible* otherwise. We say that a framework is *minimally rigid* if it is rigid and, when any of its edges is removed, it becomes *flexible*.

Laman graphs. The concept of a *generic framework* is introduced rigorously in section 5. All but a measure-zero set of possible placements of a graph are generic. The following theorem allows us to refer to the rigidity and flexibility of a generic framework solely in terms of its underlying graph. The proof goes through the intermediate concept of *infinitesimal rigidity*, which implies rigidity; this is also introduced in section 5.

Theorem 3 (see [44, 33]). *A generic bar-and-joint framework is minimally rigid in* $2D$ *if and only if its underlying graph* $G=(V,E)$ *satisfies two conditions:* (a) *it has exactly* $|E|=2|V|-3$ *edges, and* (b) *any proper subset* $V'\subset V$ *with* $|V'|\geq 2$ *of vertices spans at most* $2|V'|-3$ *edges.*

A graph satisfying the conditions of Theorem 3 is said to be a *Laman graph*, or just *Laman*. The hereditary property (b) is also referred to as the $(2,3)$-*sparsity condition*. Together, properties (a) and (b) define a graph said to be $(2,3)$-*tight* (in addition to being $(2,3)$-sparse).

Theorem 3 allows us to talk now about *(minimal) rigidity* of graphs rather than frameworks. A Laman graph is *minimally rigid* and becomes *flexible* when any of its edges is removed. Adding extra edges to a Laman graph keeps it rigid, but the minimality is lost: these graphs are said to be rigid and *overconstrained* or *dependent*. In short, for a graph to be rigid, its vertex set must span a Laman graph; otherwise the graph is flexible. Other graphs may be simultaneously flexible and overconstrained. In this paper, we work primarily with graphs which are rigid and dependent. The minimally dependent ones, called *rigidity circuits*, are introduced next.

Matroids. A matroid is an abstraction capturing (in)dependence relations among collections of elements from a *ground set* and is inspired by both *linear* dependencies (among, say, rows of a matrix) and *algebraic* constraints imposed by algebraic equations on a collection of otherwise free variables. The standard way to specify a matroid is via its *independent sets*, which have to satisfy certain axioms (which we omit, and refer the interested reader to [43]). A *base* is a maximal independent set, and a set which is not independent is said to be *dependent*. A minimal dependent set is called a *circuit*. Relevant for our purposes are the following general aspects: (a) (hereditary property) a subset of an independent set is also independent; (b) all bases have the same cardinality, called the *rank* of the matroid. Further properties will be introduced in context, as needed.

In this paper we encounter three *types of rigidity-related matroids*: a *graphic*[2] *matroid*, defined on a ground set given by all the edges $E_n := \{ij : 1 \le i < j \le n\}$ of the complete graph K_n; this is the $(2,3)$-*sparsity matroid* or the *generic 2D rigidity matroid* described below; a *linear matroid*, defined on an isomorphic set of *row vectors* of the *rigidity matrix* associated to a bar-and-joint framework; and an *algebraic matroid*, defined on an isomorphic ground set of variables $X_n := \{x_{ij} : 1 \le i < j \le n\}$; this is the *algebraic matroid associated to the Cayley–Menger ideal*. The linear and algebraic matroids will be defined in section 5.

The $(2,3)$-sparsity matroid: Independent sets, bases, circuits. The $(2,3)$-sparse graphs on n vertices form the collection of independent sets for a matroid $\boldsymbol{\mathcal{S}}_n$ on the ground set E of edges of the complete graph K_n [55], called the (generic) $2D$ *rigidity matroid*, or the $(2,3)$-*sparsity matroid*. The bases of the matroid $\boldsymbol{\mathcal{S}}_n$ are the maximal independent sets, and hence are Laman graphs. A set of edges which is not sparse is a *dependent* set. For instance, adding one edge to a Laman graph creates a dependent set of $2n-2$ edges, called a Laman-plus-one graph; examples are given in Figure 1.

A *minimal* dependent set is a (sparsity) *circuit*. The edges of a circuit span a subset of the vertices of V. A circuit spanning V is said to be a *spanning* or *maximal* circuit in the sparsity matroid $\boldsymbol{\mathcal{S}}_n$. See Figure 1 (right) and Figure 2 for examples.

A *Laman-plus-one* graph contains a unique subgraph which is *minimally dependent*, in other words, a unique circuit. A spanning rigidity circuit $C = (V,E)$ is a special case of a Laman-plus-one graph: it has a total of $2n-2$ edges but it satisfies the $(2,3)$-sparsity condition on all proper subsets of at most $n' \le n-1$ vertices. Simple sparsity considerations can be used to show that the removal of *any* edge from a spanning circuit results in a Laman graph.

Combining graphs and circuits. We define now operations that combine two graphs (with some common vertices and edges) into one.

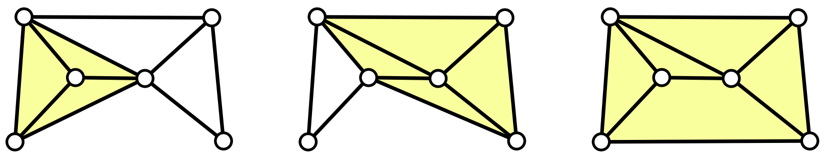

Figure 1. *A Laman-plus-one graph contains a unique circuit (highlighted): (Left and center) The circuit is not spanning the entire vertex set. (Right) A spanning circuit.*

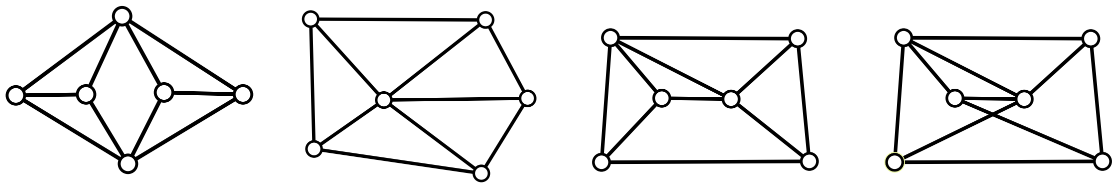

Figure 2. *The four types of spanning circuits on $n = 6$ vertices: $2D$ double-banana, 5-wheel W_5, Desargues-plus-one, and $K_{3,3}$-plus-one.*

[2]Not to be confused with the matroid of spanning trees of the complete graph.

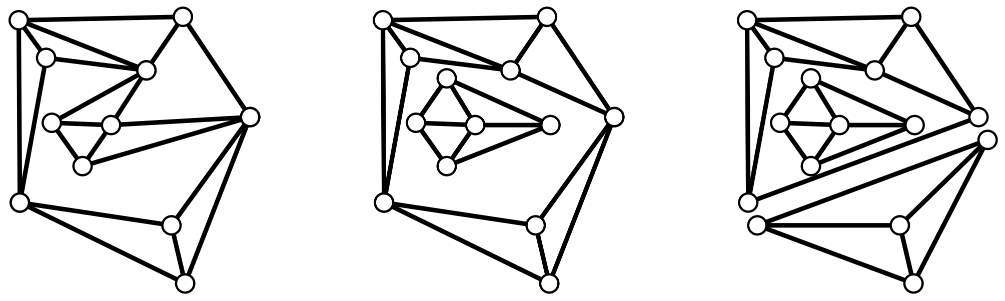

Figure 3. *(Left to right) Separating a 2-connected circuit into three 3-connected circuits via 2-split operations. (Right to left) Combining three 3-connected circuits into a larger (not-3-connected) one, via 2-sum operations.*

If G_1 and G_2 are two graphs, we use a consistent *notation* for their number of vertices and edges $n_i = |V(G_i)|$, $m_i = |E(G_i)|$, $i = 1, 2$, and for their union and intersection of vertices and edges, as in $V_\cup = V(G_1) \cup V(G_2)$, $V_\cap = V(G_1) \cap V(G_2)$, $n_\cup = |V_\cup|$, $n_\cap = |V_\cap|$, and similarly for edges, with $m_\cup = |E_\cup|$ and $m_\cap = |E_\cap|$. The *common subgraph* of two graphs G_1 and G_2 is $G_\cap = (V_\cap, E_\cap)$.

Let G_1 and G_2 be two graphs with exactly two vertices $u, v \in V_\cap$ and one edge $uv \in E_\cap$ in common. Their 2-*sum* is the graph $G = (V, E)$ with $V = V_\cup$ and $E = E_\cup \setminus \{uv\}$. The inverse operation of splitting G into G_1 and G_2 is called a 2-split or 2-separation (Figure 3).

Lemma 4 (see [5, Lemmas 4.1 and 4.2]). *The* 2*-sum of two circuits is a circuit. The* 2*-split of a circuit is a pair of circuits.*

Connectivity. It is well known and easy to show that a circuit is always a 2-connected graph. *If a circuit is not* 3*-connected,* we refer to it simply as a 2*-connected circuit*. The Tutte decomposition [51] of a 2-connected graph into 3-connected components amounts to identifying separating pairs of vertices. For a circuit, the separating pairs induce 2-split (inverse of 2-sum) operations and produce smaller circuits (see also Lemma 2.4(c) in [5]). Thus a 2-connected circuit can be constructed from 3-connected circuits via 2-sums, as illustrated in the right-to-left sequence from Figure 3.

Inductive constructions for 3-connected circuits. A *Henneberg* II extension (also called an *edge splitting* operation) is defined for an edge uv and a nonincident vertex w, as follows: the edge uv is removed, and a new vertex a and three new edges au, av, aw are added. Berg and Jordán [5] have shown that if G is a 3-connected circuit, then a Henneberg II extension on G is also a 3-connected circuit. The *inverse Henneberg* II operation on a circuit removes one vertex of degree 3 and adds a new edge among its three neighbors in such a way that the result is also a circuit; see Figure 4. Berg and Jordan have shown that every 3-connected circuit admits an inverse Henneberg II operation which also maintains 3-connectivity. As a consequence, a 3-connected circuit has an *inductive construction*, i.e., it can be obtained from K_4 by Henneberg II extensions that maintain 3-connectivity. Their proof is based on the existence of two nonadjacent vertices with 3-connected inverse Henneberg II circuits. We will make use in section 3 of the following weaker result, which does not require maintaining of 3-connectivity in the inverse Henneberg II operation.

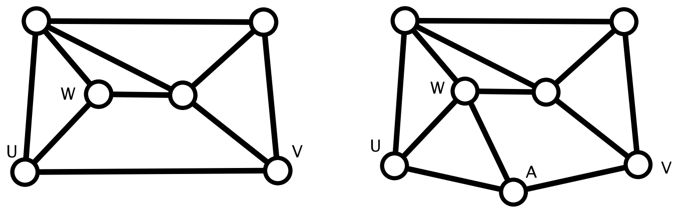

Figure 4. *A Henneberg* II *extension of the Desargues-plus-one circuit.*

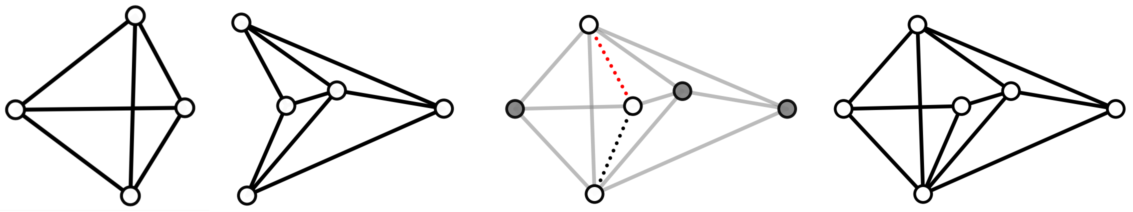

Figure 5. *A complete K_4 graph, a 4-wheel W_4, their common edges (dotted, with elimination edge in red), and their combinatorial resultant, which has more than $2n-2$ edges and thus is not a circuit.*

Lemma 5 (Theorem 3.8 in [5]). *Let $G=(V,E)$ be a 3-connected circuit with $|V|\geq 5$. Then either G has four vertices that admit an inverse Henneberg II that is a circuit, or G has three pairwise nonadjacent vertices that admit an inverse Henneberg II that is a circuit (not necessarily 3-connected).*

3. Combinatorial resultant constructions.

We define now a new operation, the *combinatorial resultant* of two graphs, prove Theorem 1, and describe its algorithmic implications.

3.1. Definition: Combinatorial resultant.

Let G_1 and G_2 be two distinct graphs with nonempty intersection $E_\cap \neq \emptyset$ and let $e\in E_\cap$ be a common edge. The *combinatorial resultant* of G_1 and G_2 on the *elimination edge* e is the graph $\mathrm{CRes}(G_1,G_2,e)$ with vertex set $V_\cup$ and edge set $E_\cup\setminus\{e\}$.

The 2-sum appears as a special case of a combinatorial resultant when the two graphs have exactly one edge in common, which is eliminated by the operation. Circuits are closed under the 2-sum operation, but they are not closed under this general combinatorial resultant operation; two examples are shown in Figures 5 and 6.

Circuit-valid combinatorial resultants. We are interested in combinatorial resultants that produce circuits from circuits. Towards this goal, we say that two circuits are *properly intersecting* if their common subgraph (of common vertices and common edges) is Laman. The example in Figure 5 is not properly intersecting, but those in Figures 6 and 7 are.

Lemma 6. *The combinatorial resultant of two circuits has $m=2n-2$ edges if and only if the common subgraph $G_\cap$ of the two circuits is Laman.*

Proof. Let C_1 and C_2 be two circuits with n_i vertices and m_i edges, $i=1,2$, and let C be their combinatorial resultant with n vertices and m edges. By inclusion-exclusion $n=n_1+n_2-n_\cap$ and $m=m_1+m_2-m_\cap-1$. Substituting here the values for $m_1=2n_1-2$ and $m_2=2n_2-2$, we get $m=2n_1-2+2n_2-2-m_\cap-1=2(n_1+n_2-n_\cap)-2+2n_\cap-3-m_\cap=$

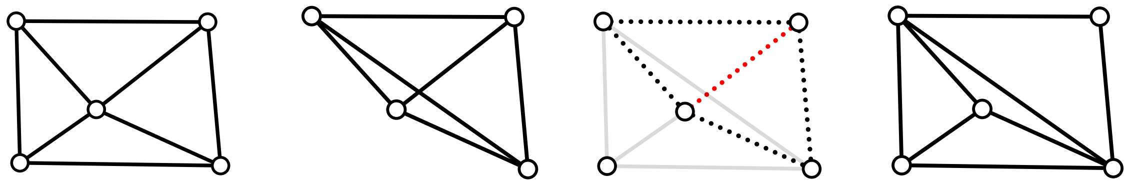

Figure 6. *A 4-wheel W_4, a complete K_4 graph, their common Laman graph (dotted, with red elimination edge), and their combinatorial resultant, which is a Laman-plus-one graph but not a circuit.*

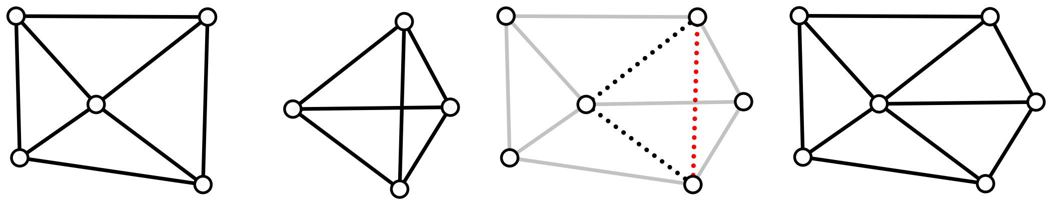

Figure 7. *A 4-wheel W_4 and a complete K_4 graph, their common Laman graph (dotted, with red elimination edge), and their combinatorial resultant, the 5-wheel W_5 circuit.*

$(2n-2)+(2n_\cap-3)-m_\cap$. We have $m=2n-2$ if and only if $m_\cap=2n_\cap-3$. Since both C_1 and C_2 are circuits, it is not possible that one edge set is included in the other: circuits are minimally dependent sets of edges and thus cannot contain other circuits. As a proper subset of both $E_1=E(C_1)$ and $E_2=E(C_2)$, $E_\cap$ satisfies the hereditary $(2,3)$-sparsity property. If furthermore $G_\cap$ has exactly $2n_\cap-3$ edges, then it is Laman. ■

It is important to retain that the *common subgraph* is defined on both the common vertex and the common edge set. The following lemma allows us to sometimes consider just the graph *induced on the common vertex set* in the union of G_1 and G_2, when checking if two circuits are properly intersecting. This observation is applicable to the type of combinatorial resultants used from now on in this paper.

Lemma 7. *Let $C_1=(V_1,E_1)$ and $C_2=(V_2,E_2)$ be two circuits whose common vertex set $V_\cap$ is a strict subset of both V_1 and V_2. If the common subgraph $G_\cap=(V_\cap,E_\cap)$ is Laman, then neither C_1 nor C_2 contains additional edges (besides $E_\cap$) spanned by their common vertices.*

Proof. Assume that C_1 contains an additional edge spanned by $V_\cap$. Since $(V_\cap,E_\cap)$ is Laman, this edge induces a circuit, entirely contained in C_1 and spanned by a proper subset of the vertices of V_1; this contradicts the fact that C_1 is a circuit: by the definition of a circuit, as a minimal dependent set of edges, a circuit cannot contain a subgraph that is smaller, yet dependent. ■

A combinatorial resultant operation applied to two properly intersecting circuits is said to be *circuit-valid* if it results in a spanning circuit. An example is shown in Figure 7. Being properly intersecting is a necessary condition for the combinatorial resultant of two circuits to produce a circuit, but the example in Figure 6 shows that this is not sufficient.

Open Problem 8. Find necessary and sufficient conditions for the combinatorial resultant of two circuits to be a circuit.

Our first goal is to show that each circuit can be obtained from K_4 circuits via a sequence of circuit-valid combinatorial resultant operations, in a manner that adds at least one new vertex at each step.

3.2. Proof of Theorem 1. We prove now that each rigidity circuit can be obtained, inductively, by applying combinatorial resultant operations starting from K_4 circuits. The proof handles separately the 2- and 3-connected cases. In section 2 we have seen that a 2-connected circuit can be obtained from 3-connected circuits via 2-sums. The bulk of the proof is in the following proposition, which handles the 3-connected circuits.

Proposition 9. *Let $C=(V,E)$ be a 3-connected circuit spanning $n+1\geq 5$ vertices. Then we can find two circuits, A and B, such that A has n vertices, B has at most n vertices, and C can be represented as the combinatorial resultant of A and B.*

Proof. We apply Lemma 5 to find two nonadjacent vertices a and b of degree 3 such that a circuit A can be produced via an inverse Henneberg II operation on vertex a in C (see Figure 8). Let the neighbors of vertex a be $N(a)=\{u,v,w\}$ such that $e=uv$ was not an edge of C and is the one added to obtain the new circuit $A=(V\setminus\{a\},(E\setminus\{au,av,aw\})\cup\{uv\})$.

To define circuit B, we first let L be the subgraph of C induced by $V\setminus\{b\}$. Simple sparsity consideration shows that L is a Laman graph. The graph D obtained from L by adding the edge $e=uv$, as in Figure 9 (left), is a Laman-plus-one graph containing the three edges incident to a (which are not in A) and the edge e (which is in A). D contains a unique circuit B (Figure 9, left) with edge $e\in B$ (see, e.g., [43, Proposition 1.1.6]). It remains to prove that B contains a and its three incident edges. If B does not contain a, then it is a proper subgraph of A. But this contradicts the minimality of A as a circuit. Therefore a is a vertex

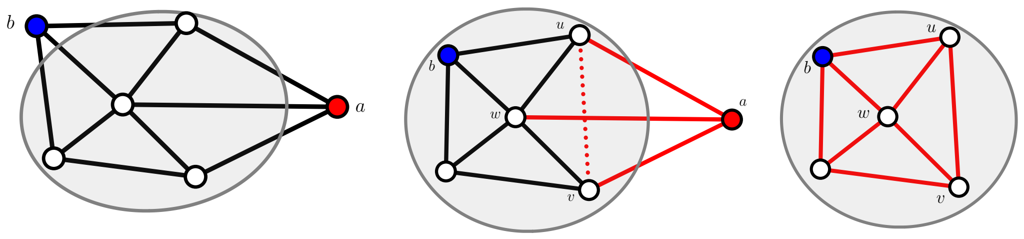

Figure 8. *The 3-connected circuit C spanning $n+1$ vertices with two nonadjacent vertices a (red) and b (blue) of degree 3. Note that $N(a)$ and $N(b)$ may not be disjoint. An inverse Henneberg II at a removes the red edges at a and adds dotted red edge $e=uv$. Circuit A (red).*

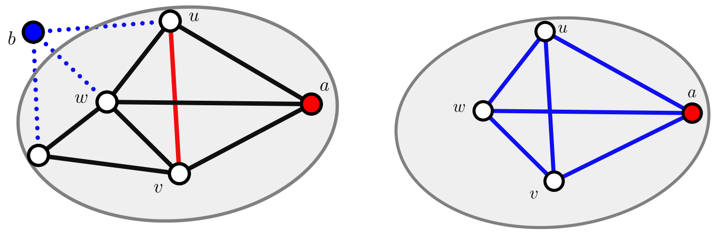

Figure 9. *Remove from C the edges from b (blue dotted) and add red edge e. Circuit B (blue).*

in B, and because a vertex in a circuit cannot have degree less than 3, B contains all three of its incident edges.

The combinatorial resultant $\mathrm{CRes}(A,B,e)$ of the circuits A and B with e the eliminated edge satisfies the desired property that $C=\mathrm{CRes}(A,B,e)$. ■

3.3. Algorithmic aspects. Algorithm 3.1 captures the procedure described in Proposition 9. It can be applied recursively until the base case K_4 is attained. Its main steps, the inverse Henneberg II step on a circuit at line 4 and finding the unique circuit in a Laman-plus-one graph at line 6, can be carried out in polynomial time using slight variations of the $(2,3)$- and $(2,2)$-sparsity pebble games from [35].

The algorithm faces many choices for the two degree-3 vertices a and b. These choices may lead to different representations of a circuit as the combinatorial resultant of two other circuits.

Corollary 10. *The representation of C as the combinatorial resultant of two smaller circuits is in general not unique. An example is the "double-banana" 2-connected circuit shown in Figure 10.*

3.4. Combinatorial circuit-resultant tree. Each one of the possible constructions of a circuit using combinatorial resultant operations can be represented in a *tree* structure. Let C be a rigidity circuit with n vertices. A *combinatorial circuit-resultant (CCR) tree* T_C for the circuit C is a rooted binary tree with C as its root and such that (a) the nodes of T_C are circuits; (b) circuits on level l have at most $n-l$ vertices; (c) the two children $\{C_j,C_k\}$ of

Algorithm 3.1 Inverse Combinatorial Resultant

```
Input: 3-connected circuit $C$
Output: circuits $A$, $B$ and edge $e$ such that $C = \mathrm{CRes}(A,B,e)$
1: for each vertex $a$ of degree 3 do
2:    if inverse Henneberg II is possible on $a$
3:    and there is a nonadjacent degree 3 vertex $b$ then
4:       Get the circuit $A$ and the edge $e$ by inverse Henneberg II in $C$ on $a$
5:       Let $D = C$ without $b$ (and its edges) and with new edge $e$
6:       Compute the unique circuit $B$ in $D$
7:       return circuits $A, B$ and edge $e$
```

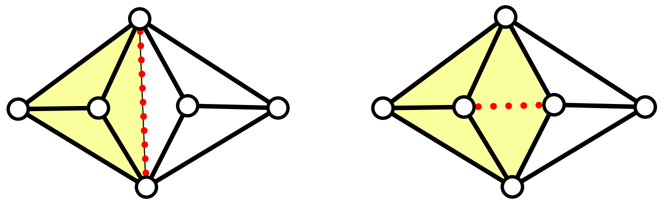

Figure 10. *The 2-connected double-banana circuit can be obtained as a combinatorial resultant from two K_4 graphs (left, 2-sum), and from two wheels on 4 vertices sharing two triangles (right). Dashed lines indicate the eliminated edges, and in each case one of the two circuits is highlighted to distinguish K_4 from W_4.*

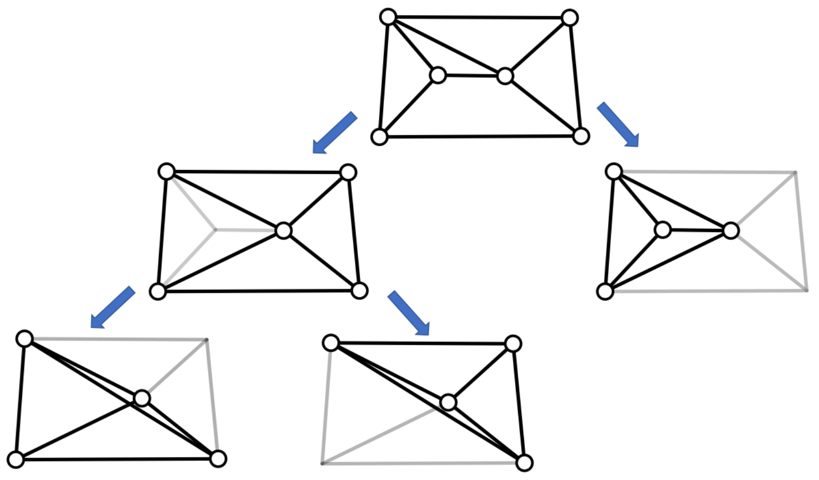

Figure 11. *A CCR tree for the Desargues-plus-one circuit. To help the reader visualize the common Laman subgraphs and the eliminated edge at each node of the tree, the lower circuits are shown, in black and with large vertices, in the context of the combinatorial resultant circuit above them (light gray).*

a parent circuit C_i are such that $C_i = \mathrm{CRes}(C_j, C_k, e)$ for some common edge e; and (d) the leaves are complete graphs on 4 vertices. An example is illustrated in Figure 11.

Complexity of CCR trees. If the intermediate circuits are all 3-connected, the depth of a tree obtained by our method is $n-4$, and this is the worst possible case. The best case for depth is $\log n$ and occurs when all the intermediate circuits are 2-connected and are split into two circuits of the same size.

In terms of size (number of nodes), the CCR tree may be, in principle, anywhere between linear to exponential in size. Best cases occur when the resultant tree is path-like, with each internal node having a K_4 leaf, or when the tree is balanced of depth $\log n$ and each resultant operation is a 2-sum. Conceivably, the worst case (exponential size) could be a complete (balanced) binary tree of linear depth: each internal node at level k would combine two circuits with the same number of vertices $n-k-1$ into a circuit with $n-k$ vertices. Sporadic examples of small, full height, and balanced CCR trees exist (e.g., for K_{33}-plus-one), but we do not know how far they generalize.

Open Problem 11. Are there infinite families of circuits with linear-depth, balanced CCR trees?

It would be interesting to understand the worst-case size of these trees, even if families as above do not exist.

Open Problem 12. Characterize the circuits produced by the worst-case size of the CCR tree.

Understanding the worst cases may help our Algorithm 3.1 avoid the corresponding choices of vertices a and b in steps 1–3. The goal would then be to produce the *best CCR tree,* or at least a good one, according to some well-defined measure of *CCR tree complexity*. We will return to this question in section 9.

In order to answer Problems 11 and 12 one may have to do experimentation with CCR trees. However, the number of trees can be very large, which leads to the following.

Open Problem 13. Develop an efficient algorithm for enumerating CCR trees of a circuit.

Open Problem 14. Compute or estimate the number of distinct CCR trees of a circuit.

4. Preliminaries: Ideals and algebraic matroids.

We turn now to the algebraic aspects of our problem in order to introduce algebraic matroids and circuit polynomials. We work over the field of rational numbers $\mathbb{Q}$. In this section, the set of variables X_n denotes $X_n = \{x_i : 1 \le i \le n\}$; when we turn to the Cayley–Menger ideal, it will be $X_n = \{x_{ij} : 1 \le i < j \le n\}$. Polynomial rings R are always of the form $R = \mathbb{Q}[X]$, over sets of variables $X \subset X_n$. The *support* $\operatorname{supp} f$ of a polynomial $f \in \mathbb{Q}[X_n]$ is the set of indeterminates appearing in it. The degree of a variable x in a polynomial f is denoted by $\deg_x f$.

4.1. Polynomial ideals.

A set of polynomials $I \subset \mathbb{Q}[X]$ is an *ideal of* $\mathbb{Q}[X]$ if it is closed under addition and multiplication by elements of $\mathbb{Q}[X]$. Every ideal contains the zero ideal $\{0\}$. A *generating set* for an ideal is a set $S \subset \mathbb{Q}[X]$ of polynomials such that every polynomial in the ideal is a finite algebraic combination of elements in S with coefficients in $\mathbb{Q}[X]$. *Hilbert's Basis Theorem* (see, e.g., [13]) guarantees that every ideal in a polynomial ring has a finite generating set. Ideals generated by a single polynomial are called *principal*. An ideal I is a *prime* ideal if, whenever $fg \in I$, then either $f \in I$ or $g \in I$. A polynomial is *irreducible* (over $\mathbb{Q}$) if it cannot be decomposed into a product of nonconstant polynomials in $\mathbb{Q}[X]$. A principal ideal is prime if and only if it is generated by an irreducible polynomial. An ideal generated by two or more irreducible polynomials is not necessarily prime. The *dimension* $\dim I$ of an ideal I of $\mathbb{Q}[X]$ is the cardinality of the maximal subset $S \subseteq X$ with the property $I \cap \mathbb{Q}[S] = \{0\}$.

Let I be an ideal of $\mathbb{Q}[X_n]$ and $X' \subset X_n$ nonempty. The *elimination ideal* of I with respect to X' is the ideal $I \cap \mathbb{Q}[X']$ of the ring $\mathbb{Q}[X']$. Elimination ideals frequently appear in the context of Gröbner bases [10, 13] which give a general approach for computing elimination ideals: if $\mathcal{G}$ is a Gröbner basis for I with respect to an *elimination order* (see Exercises 5 and 6 in section 1 of Chapter 3 in [13]), e.g., the lexicographic order with $x_{i_1} > x_{i_2} > \ldots > x_{i_n}$, then the elimination ideal $I \cap \mathbb{Q}[x_{i_{k+1}}, \ldots, x_{i_n}]$ which eliminates the first k indeterminates from I in the specified order has $\mathcal{G} \cap \mathbb{Q}[x_{i_{k+1}}, \ldots, x_{i_n}]$ as its Gröbner basis.

4.2. Algebraic independence and algebraic matroids.

Recall that a set of vectors in a vector space is linearly dependent if there is a nontrivial linear relationship between them. Similarly, given a finite collection A of complex numbers, we say that A is *algebraically dependent* if there is a nontrivial polynomial relationship between the numbers in A.

Definition 15. *Let k be a field (e.g., $k = \mathbb{Q}$) and $k \subset F$ a field extension of k. A finite subset $A = \{\alpha_1, \ldots, \alpha_n\}$ of F is said to be algebraically dependent over k if there is a nonzero (multivariate) polynomial with coefficients in k vanishing on A. Otherwise, we say that A is algebraically independent over k.*

It was noticed by van der Waerden that the algebraically independent subsets A of a finite subset E of F satisfy matroid axioms [52, 53] and therefore define a matroid.

Definition 16. *Let k be a field and $k \subset F$ a field extension of k. Let $E = \{\alpha_1, \ldots, \alpha_n\}$ be a finite subset of F. The algebraic matroid on E over k is the matroid whose independent sets are the algebraically independent (over k) subsets of E.*

4.3. Algebraic matroid of a prime ideal. An equivalent definition of algebraic matroids, in terms of polynomial ideals, is more useful for the purposes of this paper. Intuitively, a collection of variables is *independent* with respect to an ideal I if it is not constrained by any polynomial in I, and is *dependent* otherwise. The *algebraic matroid* induced by the ideal is, informally, a matroid on the ground set of variables X_n whose independent sets are subsets of variables that are *not* supported by any polynomial in the ideal. Its *dependent sets* are supports of polynomials in the ideal.

Definition 17. *Let I be a prime ideal in the polynomial ring $\mathbb{Q}[X_n]$. The algebraic matroid of I, denoted $\boldsymbol{\mathcal{A}}(I)$, is the matroid $(X_n, \mathcal{I})$ whose independent sets are*

$$\mathcal{I} = \{X \subseteq X_n \mid I \cap \mathbb{Q}[X] = \{0\}\}.$$

4.4. Equivalence of the definitions. It is well known that every algebraic matroid of a prime ideal I arises as an algebraic matroid of a field extension in the sense of Definition 16, and vice versa. For completeness, we include a proof.

From a field extension to a prime ideal. Let $E = \{\alpha_1, \ldots, \alpha_n\}$ be a set of elements in a field extension of $\mathbb{Q}$, and let $\mathcal{M}$ be the algebraic matroid on E over $\mathbb{Q}$ whose dependent sets are algebraically dependent subsets $A \subset E$. To realize $\mathcal{M}$ as an algebraic matroid of a prime ideal I of $\mathbb{Q}[X_n]$, we define $I := \ker \varphi$ as the kernel of the homomorphism $\varphi : \mathbb{Q}[X_n] \to \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ mapping $x_i \mapsto \alpha_i$ for $i \in \{1, \ldots, n\}$ and $a \mapsto a$ for $a \in \mathbb{Q}$. Kernels of homomorphisms are known to be prime ideals [34]. The kernel $\ker \varphi$ is nonzero, since any polynomial in $\ker \varphi$ defines a dependency in $\mathcal{M}$, and any dependent set $A \subset \{\alpha_1, \ldots, \alpha_n\}$ in $\mathcal{M}$ vanishes on a polynomial in $\mathbb{Q}[X_n]$. Let $\mathbb{Q}[X_A]$ be the ring of polynomials supported on subsets of $X_A := \varphi^{-1}(A)$. We have $\ker \varphi \cap \mathbb{Q}[X_A] \neq \{0\}$ if and only if A is a dependent set of $\mathcal{M}$. Hence φ induces an isomorphism between dependent sets in the matroid induced by $\ker \varphi$ and $\mathcal{M}$.

From a prime ideal to a field extension. Let I be a prime ideal in $\mathbb{Q}[X_n]$. We construct a finite field extension F and a subset $\{\overline{x_1}, \ldots, \overline{x_n}\} \in F$ via an isomorphism that takes sets $X \subset X_n$ that are in/dependent in the ideal I to algebraically in/dependent sets $\overline{X} \subset \{\overline{x_1}, \ldots, \overline{x_n}\}$. The quotient ring $\mathbb{Q}[X_n]/I$ is an integral domain with a well-defined fraction field $F = \operatorname{Frac}(\mathbb{Q}[X_n]/I)$ which contains $\mathbb{Q}$ as a subfield. The image of X_n under the canonical injections $\mathbb{Q}[X_n] \hookrightarrow \mathbb{Q}[X_n]/I \hookrightarrow \operatorname{Frac}(\mathbb{Q}[X_n]/I) = F$ is the subset $\{\overline{x_1}, \ldots, \overline{x_n}\}$ of F, where $\overline{x_j}$ denotes the equivalence class of x_j in both $\mathbb{Q}[X_n]/I$ and F.

Let X be a nonempty subset of X_n (taken w.l.o.g. to be $X = \{x_1, \ldots, x_i\}$), and let $\overline{X} = \{\overline{x_1}, \ldots, \overline{x_i}\}$ in F be its image under the canonical injections. The set $\overline{X}$ is by definition algebraically dependent over $\mathbb{Q}$ if and only if there exists a nonzero polynomial $f \in \mathbb{Q}[x_1, \ldots, x_i]$ vanishing on $\overline{X}$, i.e., $f(\overline{x_1}, \ldots, \overline{x_i}) = \overline{0}$. This happens if and only if $f(x_1, \ldots, x_i) \in I$, that is, if and only if $I \cap \mathbb{Q}[X] \neq \{0\}$. Similarly, $\overline{X}$ is algebraically independent over $\mathbb{Q}$ if and only if $I \cap \mathbb{Q}[X] = \{0\}$.

We are now ready to define the core algebraic concept underlying this paper.

4.5. Circuits and circuit polynomials. A *circuit* in a matroid is a minimal dependent set. In an algebraic matroid, a circuit $C \subset X_n$ is a minimal set of variables supported by a polynomial in the prime ideal I defining the matroid. An irreducible polynomial whose support is a circuit C is called a *circuit polynomial* and is denoted by p_C. A theorem of Dress and Lovász [16] states that, up to multiplication by a constant, *a circuit polynomial p_C is the unique irreducible polynomial in the ideal with the given support $C \subset X_n$*. We'll just say, in short, that it is *unique*.

We retain the following property, stating that *circuit polynomials generate elimination ideals supported on circuits.*

Theorem 18 (see [47, Theorem 11]). *Let I be a prime ideal in $\mathbb{Q}[X_n]$ and $C \subset X_n$ a circuit of the algebraic matroid $\boldsymbol{\mathcal{A}}(I)$. The ideal $I \cap \mathbb{Q}[C]$ is principal, prime, and generated by the circuit polynomial p_C.*

5. The Cayley–Menger ideal.

In this section we introduce the 2D Cayley–Menger ideal CM_n. We will show[3] that its algebraic matroid is isomorphic to the $(2,3)$-sparsity matroid $\boldsymbol{\mathcal{S}}_n$. As a consequence, we get a full combinatorial characterization of the supports of circuit polynomials in the Cayley–Menger ideal: they are in one-to-one correspondence with the rigidity circuits introduced in section 2.

Throughout this section and later, when working with the Cayley–Menger ideal, we use variables $X_n = \{x_{i,j} : 1 \le i < j \le n\}$ for unknown squared distances between pairs of points.

5.1. The Cayley–Menger ideal and its algebraic matroid.

The *distance matrix* of n labeled points is the matrix of squared distances between pairs of points. The *Cayley matrix* is the distance matrix bordered by a new row and column of 1's, with zeros on the diagonal:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ 1 & x_{1,2} & 0 & x_{2,3} & \cdots & x_{2,n} \\ 1 & x_{1,3} & x_{2,3} & 0 & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1,n} & x_{2,n} & x_{3,n} & \cdots & 0 \end{pmatrix}.$$

Cayley's Theorem says that if the distances come from a point set in the Euclidean space $\mathbb{R}^d$, then the rank of this matrix must be at most $d+2$. Thus all the $(d+3)\times(d+3)$ minors of the Cayley matrix should be zero. An additional condition, due to Menger [42] (see also [6, 14]), guarantees that the entries in a Cayley matrix correspond to actual squared distances between n points in $\mathbb{R}^d$. Menger's condition states that all $m \times m$ minors containing $m-1$ points have the sign $(-1)^{m-1}$ or are zero, for $m \le d+2$. For our purposes, we will make use only of Cayley's but not Menger's condition.

The set of all $(d+3)\times(d+3)$ minors of the Cayley matrix, each minor inducing a polynomial in $\mathbb{Q}[X_n]$, constitutes a generating set for the (n,d)-Cayley–Menger ideal CM_n^d. These generators are *homogeneous polynomials* with integer coefficients *irreducible* over $\mathbb{Q}$ and

[3]This equivalence is well known; however, we were not able to track down an original reference, and include a proof for completeness.

will be discussed in more detail in section 11. The (n,d)-Cayley–Menger ideal is a *prime ideal* of dimension $dn - \binom{d+1}{2}$ [8, 23, 26, 30] and codimension $\binom{n}{2} - dn + \binom{d+1}{2}$.

As defined in section 4, the algebraic matroid $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n^d)$ of the Cayley–Menger ideal is the matroid on the ground set $X_n = \{x_{i,j} \mid 1 \le i < j \le n\}$ where a subset of distance variables $X \subseteq X_n$ is independent if $\mathrm{CM}_n^d \cap \mathbb{Q}[X] = \{0\}$, i.e., X supports no polynomial in the ideal.

As an immediate consequence of the definition of dimension of an ideal in a ring of polynomials (subsection 4.1), we obtain the following proposition.

Proposition 19. *The rank of* $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n^d)$ *is equal to* $\dim \mathrm{CM}_n^d = dn - \binom{d+1}{2}$.

5.2. Equivalence of the $(2,3)$-sparsity matroid and the algebraic matroid of CM_n. From now on,[4] we work only with the $2D$ *Cayley–Menger ideal* $\mathrm{CM}_n := \mathrm{CM}_n^2$, generated by the 5×5 minors of the Cayley matrix, and its algebraic matroid, denoted by $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$. In this case, the rank of the algebraic matroid is precisely the rank of the $(2,3)$-sparsity matroid $\boldsymbol{\mathcal{S}}_n$ on n vertices, introduced in section 2. We establish the equivalence of the two matroids by proving that both are isomorphic to the 2-dimensional generic linear rigidity matroid that we now introduce.

2D linear rigidity matroids. Let $G = (V,E)$ be a graph and (G,p) a $2D$ bar-and-joint framework on points $\{p_1, \dots, p_n\} \subset \mathbb{R}^2$.

The *rigidity matrix* $R_{(G,p)}$ (or just R_G when there is no possibility of confusion) of the bar-and-joint framework (G,p) is the $|E| \times 2n$ matrix with *pairs of columns* indexed by the vertices $\{1,2,\dots,n\}$ and rows indexed by the edges $ij \in E$ with $i<j$. The ith entry in the row ij is $p_i - p_j$ (2 coordinates), the jth entry is $p_j - p_i$, and all other entries are 0.

The rigidity matrix is defined up to an order of the vertices and the edges; to eliminate this ambiguity we fix the order on the vertices as $1 < 2 < \cdots < n$, and we order the edges ij with $i<j$ lexicographically. For example, let $G = K_4$. Then the rows are ordered as 12, 13, 14, 23, 24, and 34 and the corresponding rigidity matrix R_{K_4} is given by

$$R_{K_4} = \begin{pmatrix} p_1 - p_2 & p_2 - p_1 & 0 & 0 \\ p_1 - p_3 & 0 & p_3 - p_1 & 0 \\ p_1 - p_4 & 0 & 0 & p_4 - p_1 \\ 0 & p_2 - p_3 & p_3 - p_2 & 0 \\ 0 & p_2 - p_4 & 0 & p_4 - p_2 \\ 0 & 0 & p_3 - p_4 & p_4 - p_3 \end{pmatrix}.$$

The *linear matroid* associated to a matrix is defined on the ground set given by its rows. An *independent set* is a linearly independent collection of rows.

The 2D *linear rigidity matroid* $\boldsymbol{\mathcal{L}}_{(G,p)}$ *induced by a framework* (G,p) is the linear matroid associated to the rigidity matrix of the framework. Note that it depends not just on G but also on the plane configuration p. For example, if $G = K_4$, p is a configuration in which at most two vertices of K_4 are on a line, and q is a configuration in which the vertices $\{2,3,4\}$ are on the same line, then $\operatorname{rank} \boldsymbol{\mathcal{L}}_{(K_4,p)} > \operatorname{rank} \boldsymbol{\mathcal{L}}_{(K_4,q)}$.

The $2D$ *linear rigidity matroid* $\boldsymbol{\mathcal{L}}_p$ is the linear matroid associated to the rigidity matrix of a complete graph framework (K_n,p).

[4]This section is included for completeness and can be skipped.

Genericity. Let G be a graph and consider the set of all possible plane configurations p for G. We say that a 2D bar-and-joint framework (G, p) is *generic* if the rank of the row space of $R_{(G,p)}$ is maximal among all these configurations. If p and p' are distinct generic plane configurations for a graph G, the 2D linear matroids $\boldsymbol{\mathcal{L}}_{(G,p)}$ and $\boldsymbol{\mathcal{L}}_{(G,p')}$ are isomorphic [24, Theorem 2.2.1]. Hence we can define the *2D generic linear matroid* $\boldsymbol{\mathcal{L}}_G$ as the 2D linear matroid $\boldsymbol{\mathcal{L}}_{(G,p)}$ for a generic plane configuration p.

An alternative viewpoint [50] is to work with coordinate indeterminates $p_i = \{x_i, y_i\}, i \in [n]$, over the set of variables $X_n \cup Y_n$. We define the *generic rigidity matrix* as having entries in these variables. The generic rigidity matrix has rank at least r if there exists an $r \times r$ minor which, as a polynomial in $\mathbb{Q}[X_n \cup Y_n]$, is not *identically zero*. An alternative proof of Theorem 3 given in [50] shows that maximal independent sets of rows in the generic rigidity matrix of K_n correspond to Laman graphs on n vertices. The maximal minors of the generic rigidity matrix of a Laman graph vanish on a measure-zero set of points, and all points in the complement of the vanishing locus are said to be *generic* for the given Laman graph.

The equivalence between the algebraic Cayley–Menger and the sparsity matroids. We are now ready to prove the following.

Theorem 20. *The algebraic matroid $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$ of the 2D Cayley–Menger ideal and the $(2,3)$-sparsity matroid $\boldsymbol{\mathcal{S}}_n$ are isomorphic.*

Proof. It follows from Theorem 3 that, for a given graph G on n vertices, the generic linear matroid $\boldsymbol{\mathcal{L}}_{(G,p)}$ and the $(2,3)$-sparsity matroid $\boldsymbol{\mathcal{S}}_n$ are isomorphic. It remains to show that the algebraic matroid $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$ is equivalent to the generic linear rigidity matroid $\boldsymbol{\mathcal{L}}_{K_n}$.

This equivalence is a consequence of a classical result of Ingleton [29, section 6] (see also [17, section 2]) stating that algebraic matroids over a field of characteristic zero are linearly representable over an extension of the field, with the linear representation given by the Jacobian. We now note that the Cayley–Menger variety is realized as the Zariski closure of the image of the map $f = (f_{ij})_{\{i,j\} \in \binom{n}{2}} : (\mathbb{C}^2)^n \to \mathbb{C}^{\binom{n}{2}}$ given by the edge function:

$$(p_1, \ldots, p_n) \mapsto (||p_i - p_j||^2)_{\{i,j\} \in \binom{n}{2}}.$$

The Jacobian of the edge function at a generic point in $(\mathbb{C}^2)^n$ is precisely the matrix $2R_{(K_n,p)}$ for a generic configuration p of the complete graph. ■

From now on, we will use the isomorphism to move freely between the formulation of algebraic circuits as subsets of variables $X \subset X_n$ and their graph-theoretic interpretation as graphs that are rigidity circuits.

Comment: Beyond dimension 2? Note that the d-dimensional linear rigidity matroid $\boldsymbol{\mathcal{L}}_n$ and the algebraic matroid $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n^d)$ of the (n,d)-Cayley–Menger matroid are isomorphic by the same Jacobian argument as above. However, the equivalence between the 2D sparsity matroid $\boldsymbol{\mathcal{S}}_n$ and $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$ does not extend, in higher dimensions, to some known graphical matroid. The generalization $dn - \binom{d+1}{2}$ of the $(2n-3)$-sparsity condition from dimension 2 to dimension d, called Maxwell's sparsity [41], does not satisfy matroid axioms and is known to be only a necessary but not sufficient condition for minimal rigidity in dimensions $d \geq 3$.

6. Preliminaries: Resultants.

In this section we review known concepts and facts about resultants; in the next section we specialize this setup to the Cayley–Menger ideal. In section 8, in order to prove Theorem 2, we will use the resultant of two circuit polynomials in the Cayley–Menger ideal as the algebraic counterpart of the combinatorial resultant operation which deletes a common edge e of two circuits.

Resultants. The resultant can be introduced in several equivalent ways [22]. Here we use its definition as the determinant of the Sylvester matrix.

Let $f, g \in R[x]$ be two polynomials in x with coefficients in some ring of polynomials R, with $\deg_x f = r$ and $\deg_x g = s$, such that at least one of r or s is nonzero, and let

$$\begin{aligned} f(x) &= a_r x^r + \cdots + a_1 x + a_0, \\ g(x) &= b_s x^s + \cdots + b_1 x + b_0. \end{aligned}$$

The *resultant* of f and g with respect to the indeterminate x, denoted $\mathrm{Res}(f, g, x)$, is the determinant of the $(r+s) \times (r+s)$ Sylvester matrix made from the coefficients of f and g arranged in staggered rows according to the following pattern

$$\mathrm{Syl}(f,g,x) = \begin{pmatrix} a_r & a_{r-1} & a_{r-2} & \cdots & a_0 & 0 & 0 & \cdots & 0 \\ 0 & a_r & a_{r-1} & \cdots & a_1 & a_0 & 0 & \cdots & 0 \\ 0 & 0 & a_r & \cdots & a_2 & a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & a_r & a_{r-1} & a_{r-2} & \ddots & a_0 \\ b_s & b_{s-1} & b_{s-2} & \cdots & b_0 & 0 & 0 & \cdots & 0 \\ 0 & b_s & b_{s-1} & \cdots & b_1 & b_0 & 0 & \cdots & 0 \\ 0 & 0 & b_s & \cdots & b_2 & b_1 & b_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & b_s & b_{s-1} & b_{s-2} & \ddots & b_0 \end{pmatrix},$$

where the submatrix S_f containing only the coefficients of f is of dimension $s \times (r+s)$, and the submatrix S_g containing only the coefficients of g is of dimension $r \times (r+s)$. Unless $r = s$, the columns $(a_0\, a_1 \,\cdots\, a_r)$ and $(b_0\, b_1 \,\cdots\, b_s)$ of S_f and S_g, respectively, are not aligned in the same column of $\mathrm{Syl}(f, g, x)$, as displayed above, but rather the first is shifted to the left or right of the second, depending on the relationship between r and s. We will make implicit use of the following well-known symmetric and multiplicative properties of the resultant.

Proposition 21. *Let $f, g, h \in R[x]$. The resultant of f and g satisfies*
(i) $\mathrm{Res}(f, g, x) = (-1)^{rs}\, \mathrm{Res}(g, f, x)$,
(ii) $\mathrm{Res}(fg, h, x) = \mathrm{Res}(f, h, x)\, \mathrm{Res}(g, h, x)$,
(iii) *f and g have a common factor in $R[x]$ if and only if* $\mathrm{Res}(f, g, x) = 0$.

The first two properties can be found in [22, p. 398]. The third one is stated, without proof, in [25, p. 9] for unique factorization domains. When R is a field, a proof of this property can be found in [13, Chapter 3, Proposition 3 of section 6], and it directly generalizes to polynomial rings via Hilbert's Nullstellensatz.

Resultants and elimination ideals. We will work with multivariate homogeneous polynomials f and g in $\mathbb{Q}[X_n]$, where a particular variable $x \in X_n$ is singled out. Since the resultant is a polynomial in the coefficients of f and g, its net effect is that the specific variable x is being *eliminated*. Formally, let $X' \subset X$ be nonempty and let $R = \mathbb{Q}[X']$. Let $f, g \in R[x]$, where $x \in X \setminus X'$. It is clear from the definition of the resultant that $\operatorname{Res}(f,g,x) \in R$. We will make frequent use of the following proposition, summarizing this observation; its proof can be found in [13, p. 167].

Proposition 22. *Let I be an ideal of $R[x]$ and $f, g \in I$. Then $\operatorname{Res}(f,g,x)$ is in the elimination ideal $I \cap R$.*

Homogeneous properties. From the next section on we will be working in the Cayley–Menger ideal, where the generators and the circuit polynomials are homogeneous. In sections 8 and 13 we will make use of the following proposition.

Proposition 23. *Let $f = a_{m-r}x^r + \cdots + a_{m-1}x + a_m$ and $g = b_{n-s}x^s + \cdots + b_{n-1}x + b_n$ be homogeneous polynomials in $\mathbb{Q}[y_1, \ldots, y_t, x]$ of homogeneous degree m, respectively, n, so that the coefficients $a_i, b_j \in \mathbb{Q}[y_1, \ldots, y_t]$ are polynomials of homogeneous degree i, respectively, j, for all $i \in \{m-r, \ldots, m\}$ and all $j \in \{n-s, \ldots, n\}$. If $\operatorname{Res}(f,g,x) \neq 0$, then it is a homogeneous polynomial in $\mathbb{Q}[y_1, \ldots, y_t]$ of homogeneous degree*

$$m \deg_x g + n \deg_x f - \deg_x f \cdot \deg_x g = ms + nr - rs.$$

We were not able to find a reference for this proposition in the literature. In [13, p. 454] (Lemma 5 of section 7 of Chapter 8) we found the following special case: let f and g be homogeneous polynomials of degree r, respectively, s, with $\deg_x f = r$ and $\deg_x g = s$, so that $f = a_0 x^r + \cdots + a_1 x + a_r$ and $g = b_0 x^s + \cdots + b_1 x + b_s$. In this case $\operatorname{Res}(f,g,x)$ is of homogeneous degree rs. The proof below is a direct adaptation of the proof of this special case, which itself follows directly from Proposition 23 by substituting $m \to r$ and $n \to s$ so to obtain $rs + sr - rs = rs$.

Proof. Let $\operatorname{Syl}(f,g,x) = (S_{i,j})$ be the Sylvester matrix of f and g with respect to x, and let, up to sign, $\prod_{i=1}^{r+s} S_{i,\sigma(i)}$ be a nonzero term in the Leibniz expansion of its determinant for some permutation σ of $[r+s]$.

A nonzero entry $S_{i,\sigma(i)}$ has degree $m - (r + i - \sigma(i))$ if $1 \le i \le s$ and degree $n - (i - \sigma(i))$ if $s + 1 \le i \le r + s$. Therefore, the total degree of $\prod_{i=1}^{r+s} S_{i,\sigma(i)}$ is

$$\sum_{i=1}^{s}[m - (r + i - \sigma(i))] + \sum_{i=s+1}^{r+s}[n - (i - \sigma(i))] = \sum_{i=1}^{s}(m - r) + \sum_{i=s+1}^{s+r} n - \sum_{i=1}^{r+s}(i - \sigma(i))$$
$$= s(m - r) + rn - 0 = m \deg_x g + n \deg_x f - \deg_x f \cdot \deg_x g. \qquad \blacksquare$$

7. Circuit polynomials in the Cayley–Menger ideal. In this section we define *circuit polynomials* in the Cayley–Menger ideal and make the connection with combinatorial rigidity circuits via their supports.

Circuits of $\mathcal{A}(\mathrm{CM}_n)$ and circuit polynomials in CM_n. The isomorphism between the algebraic matroid $\mathcal{A}(\mathrm{CM}_n)$ and the sparsity matroid $\mathcal{S}_n$ (Theorem 20) immediately implies that the sets of circuits of these two matroids are in a one-to-one correspondence. We will identify a sparsity circuit $C = (V_C, E_C) \in \mathcal{S}_n$, with the algebraic circuit $\{x_{i,j} \mid ij \in E_C\} \in \mathcal{A}(\mathrm{CM}_n)$, and similarly for dependent sets. Conversely, we will identify the support of a polynomial $f \in \mathbb{Q}[\{x_{i,j} \mid 1 \leq i < j \leq n\}]$ with the graph $G_f = (V_f, E_f)$, where

$$V_f = \{i \mid x_{i,j} \text{ or } x_{j,i} \in \operatorname{supp} f\} \text{ and } E_f = \{ij \mid x_{i,j} \in \operatorname{supp} f\}.$$

Given a (rigidity) circuit C, we denote by p_C the corresponding *circuit polynomial* in the Cayley–Menger ideal CM_n. Recall that by Theorem 18 the circuit polynomial of a circuit C in CM_n is the unique (up to multiplication with a unit) polynomial p_C irreducible over $\mathbb{Q}$ such that $\operatorname{supp} p_C = C$. Hence *we will identify from now on a circuit C with the support* $\operatorname{supp} p_C$ *of its circuit polynomial p_C*. Furthermore, p_C generates the elimination ideal $\mathrm{CM}_n \cap \mathbb{Q}[C]$.

Proposition 24. *Circuit polynomials in* CM_n *are homogeneous polynomials.*

Proof. Since CM_n is generated by homogeneous polynomials, any reduced Gröbner basis of CM_n consists only of homogeneous polynomials (see, e.g., Theorem 2 in section 3 of Chapter 8 of [13]). If C is a circuit in CM_n, we can choose an elimination order in which all the indeterminates in the complement of C are greater than those in C. The Gröbner basis $\mathcal{G}_C$ with respect to that elimination order will necessarily contain p_C because $\mathcal{G}_C \cap \mathbb{Q}[C]$ must generate the elimination ideal $\mathrm{CM}_n \cap \mathbb{Q}[C]$. ■

Example: The K_4 circuit. The smallest circuit polynomials are found among the generators of CM_n. Their supports are in correspondence with the edges of complete graphs K_4 on all subsets of 4 vertices in $[n]$. The circuit polynomial $p_{K_4^{1234}}$ given below corresponds to a K_4 on vertices 1234. It is homogeneous of degree 3, has 22 terms, and has degree 2 in each of its variables:

$$\begin{aligned} p_{K_4^{1234}} = {} & x_{3,4}x_{1,2}^2 + x_{3,4}^2 x_{1,2} + x_{1,3}x_{2,3}x_{1,2} - x_{1,4}x_{2,3}x_{1,2} - x_{1,3}x_{2,4}x_{1,2} \\ & + x_{1,4}^2 x_{2,3} + x_{1,3}x_{2,4}^2 + x_{1,4}x_{2,4}x_{1,2} - x_{1,3}x_{3,4}x_{1,2} - x_{1,4}x_{3,4}x_{1,2} \\ & + x_{1,3}^2 x_{2,4} + x_{1,4}x_{2,3}^2 - x_{2,3}x_{3,4}x_{1,2} - x_{2,4}x_{3,4}x_{1,2} + x_{2,3}x_{2,4}x_{3,4} \\ & - x_{1,3}x_{2,4}x_{3,4} - x_{1,3}x_{1,4}x_{2,3} - x_{1,3}x_{1,4}x_{2,4} - x_{1,3}x_{2,3}x_{2,4} \\ & - x_{1,4}x_{2,3}x_{2,4} + x_{1,3}x_{1,4}x_{3,4} - x_{1,4}x_{2,3}x_{3,4}. \end{aligned}$$

Resultants of circuit polynomials. Let f, g be two polynomials in the Cayley–Menger ideal with x_{ij} one of their common variables. We treat them as polynomials in x_{ij}, and therefore the coefficients are themselves polynomials in the remaining variables. Our *main observation*, which motivated the definition of the combinatorial resultant, is that the entries in the Sylvester matrix are polynomials supported exactly on the variables corresponding to the *combinatorial resultant* of the supports of f and g on elimination variable (edge) ij.

The following lemma, whose proof follows immediately from Proposition 22, will be used frequently in the rest of the paper.

Lemma 25. *Let I in $\mathbb{Q}[X_n]$ be an ideal, and let $f, g \in I$ be polynomials with support graphs $G_f = \operatorname{supp} f$ and $G_g = \operatorname{supp} g$ and with x_{ij} a common variable, i.e., with edge $ij \in G_f \cap G_g$. Let the combinatorial resultant of the support graphs be $S = \mathrm{CRes}(G_f, G_g, ij)$, viewed as a set of variables $S \subset X_n$. Then* $\mathrm{Res}(f, g, x_{ij}) \in I \cap \mathbb{Q}[S]$.

Algorithm 8.1 CircuitPolynomialResultant($\{A, B, e\}$, $\{p_A, p_B, x_e\}$)
Compute a circuit polynomial based on a given combinatorial resultant decomposition

```
Input:
Circuits $A$, $B$ and edge $e$ such that $C = \mathrm{CRes}(A, B, e)$.
Circuit polynomials $p_A$ and $p_B$ and elimination variable $x_e$.
Output: Circuit polynomial $p_C$ for $C$.
1: Compute the resultant $p = \mathrm{Res}(p_A, p_B, x_e)$.
2: if $p$ is irreducible then
3:     $p_C = p$
4: else
5:     $p_C$ = CleanUpResultant($p$)
6: return $p_C$
```

8. Computing a circuit polynomial as a resultant of two smaller ones.

We are now ready to complete the proof of our second result, Theorem 2. We show that combinatorial resultants are the combinatorial analogue of classical polynomial resultants in the following sense: if a (rigidity) circuit C is obtained as the combinatorial resultant $\mathrm{CRes}(A, B, e)$ of two circuits A and B with the edge e eliminated, then the resultant $\mathrm{Res}(p_A, p_B, x_e)$ of circuit polynomials p_A and p_B with respect to the indeterminate x_e is supported on C and contained in the elimination ideal $\langle p_C \rangle$ generated by the circuit polynomial p_C. When $\mathrm{Res}(p_A, p_B, x_e)$ is irreducible then it will be equal to p_C. However, in general p_C will only be one of its irreducible factors over $\mathbb{Q}$. In fact *exactly one factor* (counted with multiplicity) of $\mathrm{Res}(p_A, p_B, x_e)$ may correspond to p_C, and that factor can be deduced by examining the supports of the factors and performing an ideal membership test on those factors that have the support of p_C.

These facts are summarized by Algorithm 8.1, where the work to *clean up* the resultant in order to extract the circuit polynomial is presented as the separate Algorithm 8.2. The rest of this section is devoted to the proof of correctness of Algorithms 8.1 and 8.2, along with several remaining open problems.

8.1. Correctness of Algorithm 8.1.

We proceed by analyzing the steps.

Steps 1–4. Their correctness is established by Theorem 26 and Corollary 27 below.

Theorem 26. *Let C be a sparsity circuit on $n+1$ vertices and p_C its corresponding circuit polynomial. There exist sparsity circuits A and B on at most n vertices with circuit polynomials p_A and p_B such that p_C is an irreducible factor over $\mathbb{Q}$ of* $\mathrm{Res}(p_A, p_B, x_e)$*, where $e \in A \cap B$.*

Proof. Given a sparsity circuit C on $n+1$ vertices we can find two sparsity circuits A and B on at most n vertices such that $C = \mathrm{CRes}(A, B, e)$ for some $e \in A \cap B$ by the proof of Proposition 9. Let p_A and p_B be the corresponding circuit polynomials.

The polynomials p_A and p_B are contained in CM_m for some $m \geq n+1$ and the resultant $\mathrm{Res}(p_A, p_B, x_e)$ is a nonconstant polynomial in $R = \mathbb{Q}[(A \cup B) \setminus \{x_e\}]$ supported on C. Since $\langle p_A, p_B \rangle \subset \mathrm{CM}_m$, we have that $\mathrm{Res}(p_A, p_B, x_e)$ is contained in the elimination ideal $\mathrm{CM}_m \cap \mathbb{Q}[C] = \langle p_C \rangle$ (by Lemma 25). ∎

Corollary 27. *Under the assumptions of Theorem 26, the resultant* $\mathrm{Res}(p_A, p_B, x_e)$ *is a circuit polynomial if and only if it is irreducible (over* $\mathbb{Q}$*).*

The clean-up part would not be necessary if the resultant would always be irreducible. But in general p_C will only be one of the irreducible factors over $\mathbb{Q}$ of $\mathrm{Res}(p_A, p_B, x_e)$.

Lemma 28. *The resultant of two circuit polynomials is not always a circuit polynomial.*

Proof. We prove the lemma with an example, which can be easily generalized. Recall from Corollary 10 that in general a sparsity circuit C can be represented as the combinatorial resultant of two circuits in more than one way. If $C = \mathrm{CRes}(C_1, C_2, e) = \mathrm{CRes}(C_3, C_4, f)$ and p_{C_i} for $i \in \{1, \ldots, 4\}$, are the corresponding circuit polynomials, then $\mathrm{Res}(p_{C_1}, p_{C_2}, x_e)$ and $\mathrm{Res}(p_{C_3}, p_{C_4}, x_f)$ will in general be distinct elements of $\langle p_C \rangle$. The 2-connected circuit in Figure 10 has two distinct CCR trees, one in which the root is obtained as the combinatorial resultant of two K_4's, and the other in which the root is obtained as the combinatorial resultant of two wheels on 4 vertices. The corresponding circuit polynomials in the former case are of homogeneous degree 3 and quadratic in any indeterminate, and in the latter case they are of homogeneous degree 8 and quartic in any indeterminate (see section 13). Using Proposition 23 to compute the homogeneous degrees of the resultants, we obtain homogeneous degrees 8 and 48, respectively. Both resultants have the same circuit as its supporting set, and hence they are both in the elimination ideal $\langle p_C \rangle$, but only the one of homogeneous degree 8 is the circuit polynomial (which was verified by checking for irreducibility). ■

We can generalize the example in the proof of Lemma 28 in the following way. Let C be a sparsity circuit on $n \geq 5$ vertices. Consider the set of all possible decompositions of C as a combinatorial resultant of two sparsity circuits A and B on at most n vertices,

$$\mathit{Decompositions}(C) = \{(A, B, e) \mid C = \mathrm{CRes}(A, B, e), |V(A)|, |V(B)| \leq |V(C)|\},$$

and the set of all resultants of corresponding circuit polynomials,

$$\mathit{Resultants}(C) = \{\mathrm{Res}(p_A, p_B, x_e) \mid (A, B, e) \in \mathit{Decompositions}(C)\}.$$

The circuit polynomial p_C of the circuit C in the proof of Lemma 28 had the property of being the polynomial in $\mathit{Resultants}(C)$ of minimal homogeneous degree. One might therefore conjecture that for any sparsity circuit C, the polynomial in $\mathit{Resultants}(C)$ of minimal homogeneous degree is the circuit polynomial for C; in that case no irreducibility check would be required as we can compute the homogeneous degree of $\mathrm{Res}(p_A, p_B, x_e)$ from the homogeneous degrees and the degrees in x_e of p_A and p_B (Proposition 23). However, we will show in Proposition 48 that in general the circuit polynomial of a circuit C is not necessarily *by itself* in $\mathit{Resultants}(C)$; only a multiple of it (by a nontrivial polynomial) is. This fact leads to the following natural question.

Open Problem 29. Identify sufficient conditions under which $\mathrm{Res}(p_A, p_B, x_e)$ is p_C.

If $\mathrm{Res}(p_A, p_B, x_e)$ is not irreducible, Algorithm 8.1 invokes **CleanUpResultant** (Algorithm 8.2), whose correctness we now analyze.

Step 1. In Step 1 we first factorize p over $\mathbb{Q}$, which can be achieved in polynomial time (see [31] for a historical overview). Up to multiplicity, exactly one of the irreducible factors of

Algorithm 8.2 CleanUpResultant(C, p)
Extract the circuit polynomial from a reducible polynomial.
Preconditions:
p is a resultant of two other circuit polynomials.
p is supported on a circuit C.

Input: A circuit $C = \text{CRes}(A, B, e)$ and the polynomial p obtained as $\text{Res}(p_A, p_B, x_e)$. Assume that p is reducible.
Output: Circuit polynomial p_C for C.

```
1: factors = factorize $p$ over $\mathbb{Q}$
2: factors = discard factors with support not equal to $C$
3: **if** exactly one remaining factor (possibly with multiplicity) **then**
4:     $p_C$ = the unique factor supported on $C$
5:     **return** $p_C$
6: **else**
7:     apply a test of membership in the CM ideal on the remaining factors
8:     $p_C$ = unique factor for which ideal membership test succeeded
9:     **return** $p_C$
```

p is in CM_n, and that factor is precisely the circuit polynomial p_C (because p_C generates the elimination ideal $\text{CM}_n \cap \mathbb{Q}[C]$). The desired factor can be deduced in two steps: an analysis of the supports of all the factors and an ideal membership test.

Steps 2–5: Analyzing the supports of the irreducible factors. Recall that we identify a circuit C with the variables $\text{supp}\, p_C$ in the support of the corresponding circuit polynomial p_C and that the elimination ideal $\langle p_C \rangle$ is an ideal of $\mathbb{Q}[C]$. Let $C = \text{CRes}(A, B, e)$. Since $\text{Res}(p_A, p_B, x_e) \in \langle p_C \rangle$, any irreducible factor (over $\mathbb{Q}$) of this resultant is supported on a subset of $\text{supp}\, p_C$ that is not necessarily proper. At least one of these factors must be supported on exactly $\text{supp}\, p_C$, and if there is only one such factor, then that factor must be p_C.

Open Problem 30. Identify sufficient conditions for which $\text{Res}(p_A, p_B, x_e)$ has exactly one factor (up to multiplicity) supported on C.

Lacking a definitive answer at this time, we proceed to step 6.

Steps 6–9: Ideal membership test. We take into consideration only those irreducible factors of $\text{Res}(p_A, p_B, x_e)$ that are supported on $\text{supp}\, p_C$ (the others are automatically discarded as not belonging to the ideal). We then have to test each factor for membership in CM_n. This test can be done via a Gröbner basis algorithm with respect to any monomial order, not necessarily an elimination order. The first factor determined to be in CM_n is p_C.

It is not yet clear that this test is necessary: in practical experiments with our method, we have not yet encountered the need.

Open Problem 31. Produce an example where the resultant of two circuit polynomials in the Cayley–Menger ideal, whose combinatorial resultant is a circuit C, has a factor different from p_C but supported on $\text{supp}\, p_C$, or prove that this never happens.

8.2. The impact of the ideal membership test. The main complexity-theoretic bottleneck in our approach for computing circuit polynomials is that we *may* still have to compute a Gröbner basis in order to apply an ideal membership test. If it turns out that this step cannot be avoided, there are results suggesting that this test will not reduce our method back to a costly version of a Gröbner basis calculation.

An ideal membership test is indeed done by computing a Gröbner basis, but it does not require an elimination order, which is by all accounts impractical. Elimination orders are only necessary for computing elimination ideals (and this is what we are avoiding with our resultant-based algorithm): it is well documented that they behave badly (see [4, section 4] and the "Complexity Issues" section in [13, section 10 of Chapter 2]). On the other hand, graded orders show better performance but cannot be used to compute elimination ideals.

In summary, our approach avoids the use of an elimination order, requires only one elimination step that is obtained with resultants, and is followed by a factorization with a potential ideal membership test that can be performed by a Gröbner basis with respect to *any* monomial order. Hence we are free to choose a monomial order for CM_n that we expect to have the best performance. Of course, it is difficult to know a priori what that *good* order will be. A further investigation of this part of the algorithm remains to be pursued, in connection with the open problems described previously.

9. Computing a circuit polynomial from a combinatorial circuit-resultant tree.

We now have all the ingredients to describe an algorithmic solution to the main problem stated in the introduction: given a rigidity circuit C, compute its circuit polynomial p_C.

One way of doing this is captured by Algorithm 9.1. It uses a combinatorial circuit-resultant (CCR) tree T_C that was precomputed with Algorithm 3.1. It inductively computes polynomials supported by circuits at levels of the tree closer to the root from polynomials supported on circuits on a higher level. This algorithm stores all circuit polynomials on one level prior to going to the next level. The method becomes impractical when the CCR tree has a large number of vertices on some level, as would be the case, say, when the binary CCR tree is balanced. The correctness of Algorithm 9.1 follows directly from Algorithms 3.1 and 8.1.

Algorithm 9.2 takes an alternative approach and traverses the CCR tree in postfix order. This is naturally described as a recursive procedure. The recursion stack retains left child

Algorithm 9.1 CircuitPolynomial(T_C)
Compute a circuit polynomial from a CCR tree, inductively.

```
Input: A CCR tree $T_C$ with root a circuit $C$.
Output: Circuit polynomial $p_C$ for $C$.
Method: Traverse the tree $T_C$ bottom-up, level by level.
1: $h =$ height of $T_C$
2: level $= h - 1$
3: while level $\geq 0$ do
4:    At all the nodes $C_i$ of the current level, compute the circuit polynomial $p_{C_i}$ from the
      polynomials at its two children nodes $\{C_j, C_k\}$ using CircuitPolynomialResultant
      (Algorithm 8.1)
5:    level $=$ level $- 1$
6: return $p_C$
```

Algorithm 9.2 CircuitPolynomialRecursive
Circuit polynomial from CCR tree, postfix traversal processing

```
Input: A CCR tree $T_C$ with root a circuit $C$.
Output: Circuit polynomial $p_C$ for $C$.
Method: Traverse the tree $T_C$ in postfix order.
1: if $C$ is isomorphic to $K_4$ then
2:     $p_C = p_{K_4}$ with the appropriate relabeling of vertices
3:     return $p_C$
4: else
5:     Let $T_A, T_B$ be the left and right subtrees of $T_C$, with $C = \mathrm{CRes}(A, B, e)$ and $x_e$ the
   elimination variable.
6:     $p_A$ = CircuitPolynomialRecursive($T_A$)
7:     $p_B$ = CircuitPolynomialRecursive($T_B$)
8:     $p_C$ = CircuitPolynomialResultant($\{A, B, e\}$, $\{p_A, p_B, x_e\}$) (Algorithm 8.1)
9: return $p_C$
```

circuit polynomials along a path to a node from the root in the CCR tree, and thus its space complexity depends on the depth of the tree.

Finding a performance-optimal CCR tree for the computation of a specific circuit polynomial is a problem that remains to be investigated. It is expected that a tree that balances depth, breadth, and various algebraic parameters of the polynomials involved in the resultant steps would yield the best performance.

9.1. The "delayed clean up" heuristic. Algorithms 9.1 and 9.2 described above invoke a **CleanUpResultant** within the **CircuitPolynomialResultant** call associated to each node of the CCR tree. This is not necessary: we could just compute the resultant instead of invoking the whole **CircuitPolynomialResultant** (Algorithm 8.1) and delay the cleaning up of the resultant polynomials until we reach the root or when absolutely necessary. *Absolutely necessary* means that either (a) a resultant vanishes, or (b) the Gröbner Basis calculation for the ideal membership test in the clean up of the resultant is too expensive in terms of resources (time and memory), e.g., it takes too long, exhausts the available memory resources, or crashes. This simple "delayed clean up" heuristic may be useful in practice, in the sense that it may speed up the calculations in specific cases. We prove now that it is correct if we handle the vanishing resultant as follows.

Let $r_C = \mathrm{Res}(r_A, r_B, x_e)$ be the resultant of two previously computed polynomials, r_A and r_B, that have not been cleaned up. They contain the circuit polynomials p_A, respectively, p_B, among their (not common) factors. If r_C vanishes, then r_A and r_B have some common factors. We proceed with a **SimplifiedCleanUp** and factorize r_A and r_B, remove their common factors to obtain q_A and q_B, and recompute the new (nonvanishing) resultant $q_C = \mathrm{Res}(q_A, q_B, x_e)$. This simplified cleaning up procedure does not require an ideal membership test. The resultant q_C is well defined, because q_A (resp., q_B) contains the circuit polynomial p_A (resp., p_B) among its factors, and hence x_e is in the support of both. The multiplicativity of the resultant (Proposition 21 (ii)) implies that the resultant q_C of the simplified polynomials

q_A and q_B will be nonzero and contain a unique factor (up to multiplicity) equal to the circuit polynomial p_C for $C = \mathrm{CRes}(A, B, e)$. Therefore, the algorithm can proceed in a "delayed clean up" fashion until it encounters another vanishing resultant, performs another factorization, and so on, until it reaches the root, at which point a full clean up must be performed.

We do not know whether vanishing resultants will ever occur because in our experiments we have encountered only irreducible polynomials. High-performance computing may help answer the following remaining questions.

Open Problem 32. Find an example where a reducible polynomial appears in an intermediate step of a delayed clean up circuit polynomial calculation.

Open Problem 33. Find an example where a delayed clean up circuit polynomial calculation has an intermediate resultant equal to zero.

Open Problem 34. Provide experimental evidence on whether the "delayed clean up" heuristic can speed up a circuit polynomial calculation.

9.2. Complexity measures for CCR trees. Recall from Corollary 10 that a circuit C can have more than one CCR tree. The circuit polynomial itself is independent of this choice, but in its calculation it is useful to keep the size of the intermediate polynomials, with respect to the number of monomial terms and homogeneous degree, as small as possible. In other words, for a rigidity circuit C we would like to be able to identify an *optimal* CCR tree. The complexity of the algebraic Algorithms 9.1 and 9.2 is influenced by several factors encoded in the CCR tree: its size (total number of resultant operations), its breadth (number of nodes on the largest level), depth (longest path from root to a leaf), as well as the specificity of the elimination edge at each internal node. This motivates the following open problem.

Open Problem 35. Define a meaningful measure of CCR-tree complexity that would lead to effective computations of larger[5] circuit polynomials.

One can aim for a CCR tree in which the homogeneous degrees at each level are minimized according to the formula given in Proposition 23; however, it is not clear if this is the best approach. Indeed, in the first algorithm the degree of the circuit polynomial at a node may be smaller than predicted by Proposition 23, since the circuit polynomial may be just a factor and not the whole resultant.

Identifying optimal trees would impact the practical calculations of circuit polynomials. The concrete results reported later on in section 13 of this paper were possible because we could easily select, when $n < 7$, an optimal resultant tree from a small set of possibilities, but this set grows fast with n. It is desirable to be able to directly compute an optimal CCR tree, rather than having to iterate through all the possibilities when searching for an optimal one.

Open Problem 36. Refine Algorithm 3.1 (and its analysis) to produce an optimal CCR tree, according to a measure of CCR-tree complexity leading to efficient resultant-based calculations of circuit polynomials.

With the methods developed so far we were able to compute all the circuit polynomials in CM_6 except for the $K_{3,3}$-plus-one circuit. The computation of the circuit polynomial for

[5]For example, larger than those reported in section 13.

the $K_{3,3}$-plus-one circuit exhausted all memory at the resultant step, i.e., step 1 of Algorithm 8.1. However, by modifying the algorithm so that it also allows polynomials supported on *dependent sets* in CM_n that are not necessarily circuits, we were able to compute the circuit polynomial for the $K_{3,3}$-plus-one circuit. We now present this extended algorithm.

10. Combinatorial resultant trees.

We generalize the algorithms in section 9 by allowing all dependent sets in the rigidity matroid at the nodes, with the aim of improving computational performance.

First we relax some of the constraints imposed on the resultant tree by the construction from subsection 3.4. The internal nodes correspond, as before, to combinatorial resultant operations, but (a) they are no longer restricted to be applied only on circuits or to produce only circuits, (b) the leaves can be labeled by graphs other than K_4's, and (c) the sequence of graphs on the nodes along a path from a leaf to the root is no longer restricted to be strictly monotonically increasing in terms of the graphs' vertex sets.

Definition 37. *A finite collection* Gen *of dependent graphs such that* $K_4 \in \mathrm{Gen}$ *will be called a set of generators.*

The generators in Gen will be the graphs allowed to label the leaves. For the purpose of generating (combinatorial) circuits and computing (algebraic) circuit polynomials, we choose a set of generators, discussed in section 11, that are *dependent* in the rigidity matroid.

Definition 38. *A combinatorial resultant (CR) tree with generators in* Gen *is a finite binary tree such that* (a) *its leaves are labeled with graphs from* Gen, *and* (b) *each internal node marked with a graph G and an edge $e \notin G$ corresponds to a combinatorial resultant operation applied on the two graphs G_1, G_2 labeling its children. Specifically,* $G = \mathrm{CRes}(G_1, G_2, e)$, *where the edge $e \in G_1 \cap G_2$.*

Hence, CCR trees are special cases of CR trees. An example of a CR tree which is not a CCR tree is illustrated in Figure 12.

Lemma 39. *If the generators Gen are dependent graphs (in the rigidity matroid), then all the graphs labeling the nodes (internal, not just the leaves) of a combinatorial resultant tree are also dependent.*

Proof. The proof is an induction on the tree nodes, with the base cases at the leaves. We define an edge of G to be redundant if after its deletion the graph remains rigid; otherwise the edge is said to be critical: its removal makes the graph flexible. For the inductive step, assume that G_1 and G_2 are the dependent graphs labeling the two children of a node labeled with $G = \mathrm{CRes}(G_1, G_2, e)$, where $e \in E_\cap$ is an edge in the common intersection $G_\cap$. We consider two cases, depending on whether e is *redundant* in both or *critical* in at least one of G_1 and G_2. In each case, we identify a subset of the combinatorial resultant graph G which violates Laman's property, and hence we'll conclude that the entire graph G is dependent.

Case 1: *e is redundant in both G_1 and G_2.* This means that there exist subsets of edges $C_1 \subset G_1$ and $C_2 \subset G_2$, both containing the edge e, which are circuits (their individual spanned-vertex sets may possibly contain additional edges, but this only makes it easier to reach our desired conclusion). Their intersection $C_1 \cap C_2$ cannot be dependent (by the minimality of circuits). Hence their union, with edge e eliminated, has at least $2n_\cup - 2$ edges (cf. the proof of Lemma 6), and hence it is dependent.

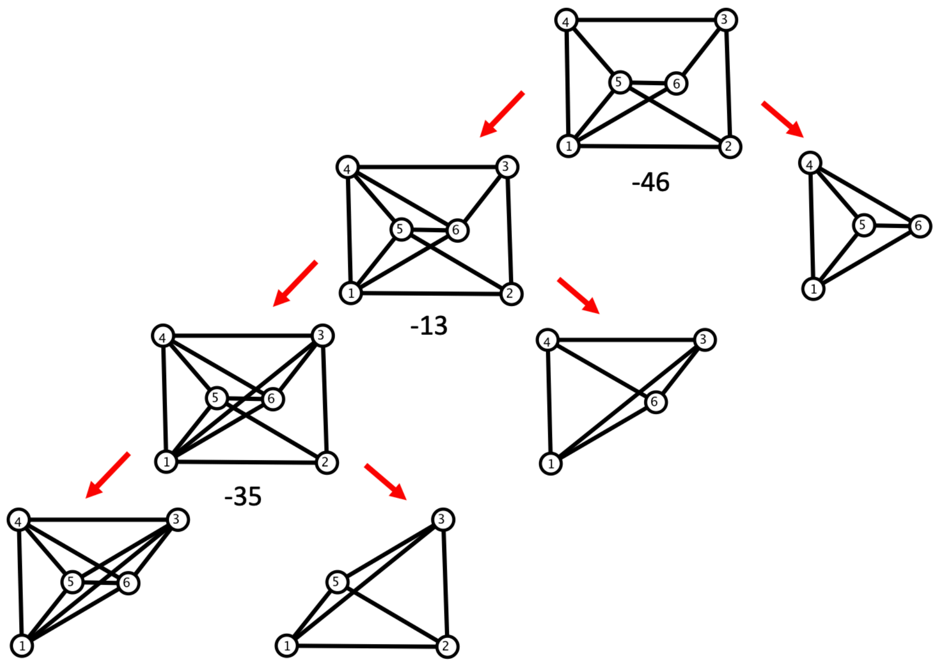

Figure 12. *A combinatorial resultant tree for the $K_{3,3}$-plus-one circuit: its leftmost leaf and the two internal nodes along the leftmost path to the root are labeled with rigid dependent graphs which are not circuits.*

Case 2: *e is critical in G_1 or critical in G_2.* Let's assume it is critical in G_1. Since G_1 is dependent and $e \in G_1$ is critical, it means that the removal of e from G_1 creates a flexible graph which is still dependent. As a flexible graph, it splits into edge-disjoint rigid components; in this case, at least one of these components R is dependent. Then, since the removal of e does not affect R, it follows that R and thus the resultant graph $G = \text{CRes}(G_1, G_2, e)$ remain dependent. ■

Definition 40. *Given a circuit C, a valid combinatorial resultant tree for C is a combinatorial resultant tree with root C and whose leaves (and hence nodes) are dependent graphs.*

The example in Figure 12 is a valid combinatorial resultant tree for the $K_{3,3}$-plus-one circuit. After reviewing the necessary algebraic notions in the next section, we will use it in subsection 13.4 to demonstrate our generalized algebraic elimination algorithm described in section 12.

11. Generators of the 2D Cayley–Menger ideal.

We work with the set GenCM_n of generators for the 2D Cayley–Menger ideal CM_n as given by the set of all 5×5 minors of the $(n+1) \times (n+1)$ Cayley matrix. Each generator $g \in \text{GenCM}_n$ is identified with its support graph G_g, as defined in section 7. To motivate the possible choices for the family of graphs Gen for the generalized combinatorial resultant trees defined in section 10, we now tabulate the support graphs of all generators, up to multiplication by a nonzero constant, relabeling, and graph isomorphism.

To find all these graphs, it is sufficient to consider the set GenCM_{10} of all 5×5 minors of CM_{10}. Using a computer algebra package, we can verify that this set has 109 619 distinct

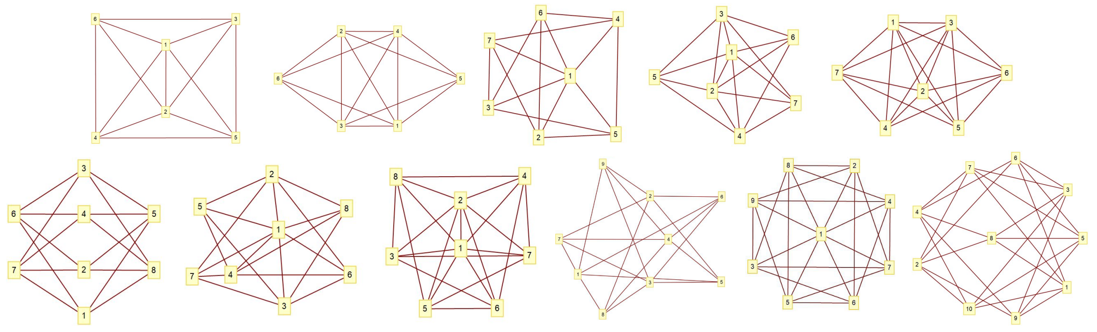
Figure 13. *The* 14 *graph isomorphism classes of Cayley–Menger generators consist in the three complete graphs* K_4, K_5*,* K_6 *and the* 11 *graphs on* 6 *to* 10 *vertices shown here.*

minors, of which 106 637 have distinct support graphs. The IsomorphicGraphQ function of *Mathematica* was used to reduce them to the 14 graph isomorphism classes, 11 of which are shown in Figure 13. The only two representatives with fewer than 6 vertices are K_4 and K_5. There are three isomorphism classes on 6, 7, 8 vertices (one is K_6), two on 9, and one on 10 vertices. The corresponding generator polynomials are, up to isomorphism (relabeling of variables induced by relabeling of the vertices), unique for the given support, with a few exceptions: for K_5, we found 3 distinct (nonisomorphic) polynomials.

Note that there may be polynomials in CM_n supported on the same set as a generator from GenCM_n, but which themselves do not arise from a single 5×5 minor of a Cayley matrix. For example, if $p \in \mathrm{GenCM}_n$ is supported on a K_5 and $q \in \mathrm{GenCM}_n$ is supported on a K_6 such that $\operatorname{supp} p \subset \operatorname{supp} q$, then $p+q$ has the support of a generator on K_6 but itself is not in GenCM_n.

12. Algorithm: Circuit polynomial from combinatorial resultant tree.

We now have all the ingredients for describing Algorithm 12.1, which computes the circuit polynomial p_C for a circuit C from a given combinatorial resultant tree T_C, or returns a message that p_C cannot be computed using T_C. Just like the algorithms of section 9, it computes resultants at each node of the tree, starting with the resultants of generators of CM_n supported on leaf nodes. At the root node the circuit polynomial for C is extracted from the irreducible factors of the resultant at the root. The main difference lies at the intermediate (nonroot) nodes, as described in Algorithm 12.1 below. This is because the polynomials sought at nonleaf nodes, not being supported on circuits, are not necessarily irreducible polynomials *supported on the desired dependent graph*, as was the case in section 9. Hence, conceivably, they may have factors that are not in the Cayley–Menger ideal, and it might be the case that none of their factors that are in the Cayley–Menger ideal are supported on the desired graph, but their product with other factors is. Moreover, it might be the case that an intermediate resultant $\mathrm{Res}(f,g,x)$ is zero, with x being present only in the supports of common factors of f and g, in which case the algorithm cannot resume along the chosen tree T_C. It remains, however, as an open question (which may entail experimentation with gigantic polynomials) to explicitly find such examples (we did not find any so far) and to prove what may or may not happen.

Algorithm 12.1 Computing a polynomial in the Cayley–Menger ideal supported on a node of a combinatorial resultant tree—simple version.

Input: Nonleaf node G of a combinatorial resultant tree T_C. Polynomials $v, w \in \mathrm{CM}_n$ supported on the child nodes of G and x_e, the indeterminate to be eliminated.
Output: Polynomial $p \in \mathrm{CM}_n$ supported on G or a string stating that p could not be computed.

```
 1: Compute the resultant $r = \mathrm{Res}(v, w, x_e)$.
 2: If $r = 0$ return "Not possible to compute $p$".
 3: Factorize $r$ over $\mathbb{Q}$ and store all factors supported on dependent sets in the list
    $candidates$.
 4: if $candidates = \{p\}$ then
 5:    if $\mathrm{supp}\, p = G$ then return $p$
 6:    else return $p \cdot \Pi_{x \in G \setminus \mathrm{supp}\, p} x$
 7: else
 8:    for all $p \in candidates$ do
 9:       Test $p$ for membership in $\mathrm{CM}_n$ with an ideal membership test
10:       if $p \in \mathrm{CM}_n$ then
11:          if $\mathrm{supp}\, p = G$ then return $p$
12:          else return $p \cdot \Pi_{x \in G \setminus \mathrm{supp}\, p} x$
```

Proof of correctness of Algorithm 12.1. Recall that $\mathbb{Q}[G]$ denotes the ring of polynomials with indeterminates x_{ij} with $i < j$ given by the edges ij of G.

Steps 1–2. Compute the resultant. If the resultant is zero, the algorithm terminates with the message that it is not possible to continue along T_C. We can attempt to replace one or both of v and w with other polynomials in CM_n with appropriate support that would lead to a nonzero resultant; however, in our presentation we assume that all the choices made in previous calls of Algorithm 12.1 (e.g., the choice of a candidate in line 9) remain fixed.

Step 3. The elimination ideal $\mathrm{CM}_n \cap \mathbb{Q}[G]$ is prime, and hence at least one irreducible factor p of r is in CM_n.

Step 4. If there is exactly one factor p supported on a dependent set, then that factor must necessarily be in CM_n. This follows from the primality of $\mathrm{CM}_n \cap \mathbb{Q}[G]$: assume for simplicity that r factors as $q_1 \cdot q_2 \cdot p$, with only p being supported on a dependent set. If $q = q_1 \cdot q_2$ is supported on an independent set, then it is not in CM_n, and hence p must be in $\mathrm{CM}_n \cap \mathbb{Q}[G]$. If q is supported on a dependent set, then $q \in \mathrm{CM}_n$ would imply that one of q_1 or q_2 is in CM_n, but none of the two are. Therefore $p \in \mathrm{CM}_n \cap \mathbb{Q}[G]$ in any case.

Steps 5–6. There are now two possibilities for p: either it is supported on G, in which case we return it, or it is supported on a proper subset of G. If its support is a proper subset of G, we can in principle return any polynomial qp such that $\mathrm{supp}\, qp = G$. Recall that the resultant is multiplicative (Proposition 21); hence in a subsequent invocation of the algorithm, in the computation of $\mathrm{Res}(qp, f, y) = \mathrm{Res}(q, f, y)\,\mathrm{Res}(p, f, y)$ for some f and y we can keep the factor $\mathrm{Res}(q, f, y)$ unevaluated. An alternative would be to modify the resultant tree T_C by replacing G with the graph G_p given by the support of p (as defined in section 7). However, in our presentation we keep the resultant tree fixed throughout and choose q to simply be the product $\Pi_{x\in G\setminus \operatorname{supp} p} x$ of all indeterminates in $G\setminus \operatorname{supp} p$.

In our experiments we are yet to encounter an example in which an irreducible factor supported on a dependent set that is a proper subset of G appears. We leave as an open problem to find such an example, or prove that it cannot occur.

Open Problem 41. Consider an intermediate node G in a combinatorial resultant tree and let $r=\operatorname{Res}(f,g,x_e)$ be the resultant supported on G with respect to the polynomials supported on the child nodes of G, as in Algorithm 12.1. Find examples where r has exactly one irreducible factor supported on a dependent set, and such that it is properly contained in G, or prove that this never happens.

Steps 7–12. If there is more than one irreducible factor supported on a dependent set, we store them in the list *candidates* in some order. Factors are then tested for membership in CM_n with an ideal membership test, in the order in which they are stored in the list *candidates*. The first irreducible factor that passes the test is returned if its support is G, or it is completed to a polynomial supported on G in the same way as described above and then returned.

We have not encountered examples in which more than one irreducible factor supported on a dependent set appeared; however, this is most likely because we were only able to perform computations on graphs with up to 8 vertices.

Open Problem 42. Consider an intermediate node G in a combinatorial resultant tree and let $r=\operatorname{Res}(f,g,x_e)$ be the resultant supported on G with respect to the polynomials supported on the child nodes of G, as in Algorithm 12.1. Find examples where r has more than one irreducible factor supported on a dependent set, or prove that this never happens.

Since G is not necessarily a circuit, the elimination ideal $\mathrm{CM}_n\cap\mathbb{Q}[G]$ is no longer necessarily principal, and we can no longer guarantee the existence of a unique irreducible factor p of r that is supported both on G and in CM_n. We have not encountered this possibility in our experiments, and we leave it as an open question.

Open Problem 43. If Open Problem 42 has a positive answer, find examples with two or more irreducible factors supported on G, or prove that this never happens.

Refinements of Algorithm 12.1. If at a node of T_C we have $\operatorname{Res}(v,w,e)=0$, we can attempt to replace v or w with other appropriate polynomials in CM_n. In particular we can attempt to recompute v or w by choosing a different polynomial from the list of candidates in line 9. This approach, however, might require recomputing v and w many times, and we can still not guarantee that $\operatorname{Res}(v,w,x_e)$ would be nonzero. We leave as an open problem to find the conditions on v and w so that $\operatorname{Res}(v,w,x_e)$ is not zero.

Open Problem 44. Consider the case in which at an intermediate node of T_C we have $\operatorname{Res}(v,w,e)=0$. Is it always possible to recompute v and w with Algorithm 12.1 by choosing a different polynomial from the list of candidates (line 9 of the algorithm) so that $\operatorname{Res}(v,w,e)\neq 0$?

Alternatively we can replace one or both branches of the resultant tree for G (taken as the subtree of T_C rooted at G) with a tree that would lead to a nonzero resultant at G. For that purpose it would be useful to have an algorithm that enumerates the resultant trees of a dependent graph. Such enumeration appears to be much more challenging than for CCR trees (Open Problem 13), and it is unclear that an efficient solution to the following problem can be obtained.

Open Problem 45. Develop an algorithm for enumerating resultant trees of a dependent graph.

If the answer to Open Problem 42 is positive, we have to decide which polynomial to output. In Algorithm 12.1 the first irreducible factor with dependent support that passes the ideal membership test is chosen and returned (possibly padded by the indeterminates in $G\setminus \operatorname{supp} p$). However, it may be the case that the first irreducible factor that passes the ideal membership test is not the best choice if what we have in mind is the goal of simplifying the resultant computation when this algorithm is invoked on the parent of G. For example, relative to the remaining factors that pass the ideal membership test, the first factor that passed the test could have a very large degree in the indeterminate that is to be eliminated in the subsequent invocation of the algorithm, which, as a consequence, would lead to a very large dimension of the Sylvester determinant.

We propose the following decision criteria in the case when r has multiple irreducible factors $\{p_1,\dots,p_k\}$ in CM_n. From the set $\{p_1,\dots,p_k\}$ choose the polynomial:

(i) Choose the one with the least degree in the indeterminate to be eliminated when Algorithm 12.1 is invoked on the parent of G.
(ii) If there is more than one such choice, we choose the one with the least homogeneous degree.
(iii) If there still is more than one choice, we choose the first one with the least number of monomials.

Criterion (i) ensures that when the algorithm is invoked on the parent of G, the dimension of the Sylvester determinant will be the least possible; criterion (ii) ensures that the resultant will be of least possible homogeneous degree (Proposition 23), while criterion (iii) minimizes the total number of monomials that appear as entries in the Sylvester determinant.

This choice of decision criteria may not be the best possible, and we leave as an open problem to formulate other decision criteria.

Open Problem 46. If Open Problem 42 has a positive answer, establish criteria for deciding which polynomial to return as output.

13. Experiments.

In this section we discuss our experimental work, carried out with the algorithms presented in this paper, that led to effective computations of all circuit polynomials in CM_6. Table 1 summarizes the results. To the best of our knowledge, except for the circuit polynomial of K_4, these polynomials have not been computed before. Each example of a circuit polynomial is presented up to relabeling of vertices. All the circuit polynomials computed in this section are available at the GitHub repository [39]. For comparison purposes, we also include some preliminary calculations done or attempted with Gröbner basis methods.

The K_4 circuit. The only circuit polynomial that is directly obtainable as a generator of CM_n for any $n \geq 4$, and does not require Gröbner basis methods or resultant computations,

Table 1

Results: all circuit polynomials on $n \leq 6$ vertices, two circuit polynomials on $n = 7$ vertices, and two circuit polynomials on $n = 8$ vertices. The method Gröbner is the computation of a Gröbner basis of ideals generated by two circuit polynomials, as explained in subsection 13.1. The method Resultant A9.1 *is Algorithm 9.1, and the method Resultant* A12.1 *is Algorithm 12.1.*

n	Circuit	Method	Comp. time (seconds)	No. terms	Hom. degree
4	K_4	Determinant	0.0008	22	3
5	Wheel on 4 vertices	Gröbner	0.02	843	8
		Resultant A9.1	0.013		
6	2D double banana	Gröbner	0.164	1 752	8
		Resultant A9.1	0.029		
6	Wheel on 5 vertices	Gröbner	10 857	273 123	20
		Resultant A9.1	7.07		
6	Desargues-plus-one	Gröbner	454 753	658 175	20
		Resultant A9.1	14.62		
6	$K_{3,3}$-plus-one	Resultant A12.1	979.42	1 018 050	18
7	2D double banana $\oplus_{16}$ K_4^{1567}	Resultant A9.1	38.14	1 053 933	20
7	2D double banana $\oplus_{56}$ K_4^{4567}	Resultant A9.1	89.86	2 579 050	20
8	2D double banana $\oplus_{45}$ K_4^{4578}	Resultant A9.1	109.8	3 413 204	20
8	2D double banana $\oplus_{56}$ K_4^{5678}	Resultant A9.1	302.47	9 223 437	20

is the circuit polynomial of a K_4 graph (possibly relabeled). This polynomial has 22 terms, homogeneous degree 3, and is of degree 2 in any of its variables.

13.1. Computation of circuit polynomials via Gröbner bases. In principle a circuit polynomial $p \in \mathrm{CM}_n$ can be computed by computing a Gröbner basis $\mathcal{G}_{\mathrm{CM}_n}$ for CM_n with respect to an *elimination order* on the set $\{x_{i,j} \mid 1 \leq i < j \leq n\}$ in which all the indeterminates in the complement of $\operatorname{supp} p$ are greater than all the indeterminates in $\operatorname{supp} p$.

Given $\mathcal{G}_{\mathrm{CM}_n}$ it is straightforward to determine a Gröbner basis $\mathcal{G}_{\langle p \rangle}$ for the ideal $\langle p \rangle = \mathrm{CM}_n \cap \mathbb{Q}[\operatorname{supp} p]$: it is the intersection $\mathcal{G}_{\langle p \rangle} = \mathcal{G}_{\mathrm{CM}_n} \cap \mathbb{Q}[\operatorname{supp} p]$. Therefore, the only element in $\mathcal{G}_{\mathrm{CM}_n}$ supported on $\operatorname{supp} p$ is precisely p, possibly multiplied by a nonzero scalar.

Gröbner basis for CM_n with respect to an elimination order. We were able to compute a Gröbner basis with respect to an elimination order only for $n = 5$. Already for $n = 6$ we did not succeed in carrying out such a computation, within a reasonable amount of time, neither in *Mathematica* nor in Macaulay2.

Gröbner basis of ideals generated by two circuit polynomials. For comparison purposes, we describe a second method that we experimented with. This one takes into account the combinatorial structure presented in section 3 but works with Gröbner bases rather than resultants. Let A, B, and C be circuits such that $C = \mathrm{CRes}(A, B, e)$, where e is a common edge of A and B. To compute the circuit polynomial p_C of the circuit C, it is sufficient to calculate only a Gröbner basis β of the ideal $\langle p_A, p_B \rangle$ generated by the circuit polynomials of A and B, with respect to an elimination order in which the indeterminates in $(A \cup B) \setminus C$ are eliminated. This follows from $\langle p_A, p_B \rangle \cap \mathbb{Q}[C] \subseteq \mathrm{CM}_n \cap \mathbb{Q}[C] = \langle p_C \rangle$, where if $\langle p_A, p_B \rangle$ is prime, then the Gröbner basis β will be exactly equal to $\beta = \{p_C\}$. Otherwise, a factorization and a subsequent ideal membership test for the factors supported on C of each polynomial in β will be required.

With this method we were able to compute all the circuit polynomials of circuits on 6 vertices except the $K_{3,3}$-plus-one circuit. It took us 0.164 seconds to compute the 2D double banana, a bit over 3 hours to compute the wheel on 5 vertices, and 126 hours to compute the Desargues-plus-one circuit polynomial (see Table 1).

13.2. Computation of circuit polynomials with resultants.

We demonstrate now the effectiveness of our algorithm by computing all the circuit polynomials on up to 6 vertices. They are supported on five types of graphs: a 4-wheel W_4 (on 4 cycle vertices with a 5th vertex at the center), a 5-wheel, a 2D "double banana" obtained as a 2-sum of two K_4 graphs, the Desargues-plus-one graph, and the $K_{3,3}$-plus-one graph. They are shown in Figures 6 and 2. We are recording only the computation of the root of a particular resultant tree. We chose resultant trees that were most efficient for each computation. The relevant parameters of each circuit (size, homogeneous degree) and comparative timings for its computation are shown in Table 1. Two more circuits on 7 vertices, as well as two on 8 vertices, were also computed using 2-sum resultants, which give the best resultant trees.

Wheel on 4 vertices. This circuit was very fast to compute. It has (up to relabeling) exactly one resultant tree with two K_4 leaves and a single application of a resultant, which produces an irreducible polynomial. Irreducibility was verified with *Mathematica*. This polynomial has 843 terms, its homogeneous degree is 8, and it is of degree 4 in each of its variables.

The "2D double banana". Recall from Figure 10 that the 2D double banana can be obtained as the combinatorial resultant of two K_4's or of two 4-wheels. The first tree led to a very fast calculation, and the resultant produced an irreducible polynomial. This polynomial has 1752 terms, its homogeneous degree is 8, and it is of degree 4 in each of its variables.

However, on our computers we did not succeed in calculating the circuit polynomial using the second resultant tree, or as a Gröbner basis of an ideal generated by the circuit polynomials of the two 4-wheels, with respect to an elimination order. Here is a possible explanation. Recall that Proposition 23 allows us to predict the homogeneous degree of the resultant of two homogeneous polynomials. In particular, the homogeneous degree of the resultant for two 4-wheels has homogeneous degree 48, whereas the resultant of the circuit polynomials of two K_4 graphs has homogeneous degree 8. Hence, we could see immediately that we should discard the former, as in the latter case we obtain a much simpler polynomial. This example inspires the following conjecture.

Open Problem 47. Prove that a 2-sum is more efficient than any other type of combinatorial resultant, in computing a circuit polynomial as a resultant of two circuits.

Wheel on 5 vertices. We computed this circuit from a 4-wheel and a K_4 and obtained directly an irreducible polynomial. Irreducibility was verified in *Mathematica*. This polynomial has 273 123 terms, its homogeneous degree is 20, and it is of degree 8 in each of its variables.

The Desargues-plus-one circuit. The rigidity theory literature refers to the graph D with edges $\{12, 14, 15, 23, 26, 34, 36, 45, 56\}$ as the Desargues graph, due to its similarity to the incidence structure arising from the classical Desargues configuration of lines. The graph D can be completed to a circuit (what we call Desargues-plus-one) by adjoining to it exactly one of the missing edges, with all choices of missing edge resulting in isomorphic graphs. The circuit can be obtained as a combinatorial resultant of a 4-wheel (with cycle $1,2,3,4$, and 5 at the center) and a K_4 on vertices $2,3,5,6$ by eliminating the edge 35. Using the previously computed 4-wheel circuit polynomial, the resultant calculation took under 15 seconds, which is impressive when compared to the 5 days and 6 hours taken by the Gröbner basis method. The resultant polynomial is irreducible, has homogeneous degree 20, and is of degree 12 in the variable $x_{2,5}$ and of degree 8 in the remaining variables.

13.3. The $K_{3,3}$-plus-one circuit.

The complete bipartite graph $K_{3,3}$ on the vertex partition $\{1,4,5\} \cup \{2,3,6\}$ is minimally rigid. It can be completed to a circuit by adding to it exactly one of the missing edges. All these choices result in isomorphic graphs.

We were not able to compute its circuit polynomial with Algorithm 9.1 or Algorithm 9.2. All attempts completely exhausted all computational resources at the resultant step. However, we succeeded with the approach described in section 12. This method allowed us to carry out the full computation, described step by step in subsection 13.4. The irreducible circuit polynomial has 1 018 050 terms, has homogeneous degree 18, and is of degree 8 in each variable.

The properties of this polynomial imply an interesting fact, which is relevant for a better understanding of Algorithm 8.1: it provides, indirectly, the first example of a circuit polynomial on which the last resultant step *in any of the possible combinatorial resultant trees* would have to produce a polynomial which is *never* irreducible. Hence a factorization and an inspection of factors for membership in the Cayley–Menger ideal will be necessary at the root, either by inspecting the supports or by performing a test of membership in the Cayley–Menger ideal. The proof is instructive and we include it here.

Proposition 48. *Let A and B be rigidity circuits on 6 or fewer vertices such that neither is the $K_{3,3}$-plus-one circuit and such that $\mathrm{CRes}(A,B,e)$ is the $K_{3,3}$-plus-one circuit for some common edge e. If p_A and p_B are the circuit polynomials for A and B, then $\mathrm{Res}(p_A,p_B,x_e)$ is reducible.*

Proof. Let h_A and h_B be the homogeneous degrees, and let d_A and d_B be the degrees in x_e of p_A and p_B, respectively. By Proposition 23, the homogeneous degree of $\mathrm{Res}(p_A,p_B,x_e)$ is $h_Ad_B + h_Bd_A - d_Ad_B$, so if $\mathrm{Res}(p_A,p_B,x_e) = c \cdot p_{K_{3,3}\text{-plus-one}}$ for some $c \in \mathbb{Q}$, then $h_Ad_B + h_Bd_A - d_Ad_B = 18$. However, by subsection 13.2 the values of (h_A,d_A) and (h_B,d_B) can only be in the set $\{(3,2),(8,4),(20,8),(20,12)\}$ and no choice corresponds to $h_Ad_B + h_Bd_A - d_Ad_B = 18$. ■

As a final observation, we note that the $K_{3,3}$-plus-one graph can be obtained as the combinatorial resultant of two 4-wheels: one wheel on $1,2,3,4$ with 5 in the center, and the other on $1,3,4,6$ with 5 in the center, on the elimination edge 15. Since the circuit polynomial for a 4-wheel has homogeneous degree 8 and both have degree 4 in $x_{1,5}$, it follows from Proposition 23 that their resultant has homogeneous degree 48. Hence the circuit polynomial for $K_{3,3}$-plus-one appears as a factor in this resultant, with multiplicity not greater than 2. Unfortunately, we were not able to compute the resultant of these two 4-wheels before our machines ran out of memory. We have attempted to brute-force the computation by first computing the resultant of two general degree-4 polynomials in the variable x, which has 219 monomials. We then substituted the coefficients (with respect to x) of the circuit polynomials for the two wheels into the 219 monomials. We then proceeded to expand them, and save each of the 219 expansions to disk. This took approximately 5 days of computing on an HPC and in total occupies approximately 1.7TB of data (stored in *Mathematica*'s uncompressed .mx format). However, adding together the 219 expanded monomials failed, and we did not pursue this direction further. We estimate that a powerful enough machine with at least 2TB of RAM could be forced to compute the resultant of two wheels on 4 vertices.

13.4. Example: The $K_{3,3}$-plus-one circuit polynomial. At the leaves of the tree we are using irreducible polynomials from among the generators of the Cayley–Menger ideal. The polynomials corresponding to the nodes on the leftmost path from a leaf to the root are referred to, below, as D_1 (leftmost leaf), D_2 and D_3 (for the next two internal nodes with dependent graphs on them), and C for the circuit polynomial at the root; see Figure 12. The leaves on the right are three K_4 circuit polynomials: C_1 supported on vertices $\{1,2,3,5\}$, C_2 supported on $\{1,3,4,6\}$, and C_3 supported on $\{1,4,5,6\}$. For the polynomial D_1 at the bottom leftmost leaf, supported by a dependent K_5 graph, we have used the generator:

$$\begin{aligned}
&x_{15}x_{34}^2 - x_{16}x_{34}^2 - x_{56}x_{34}^2 - x_{14}x_{35}x_{34} + x_{16}x_{35}x_{34} + x_{14}x_{36}x_{34} - 2x_{15}x_{36}x_{34}\\
&+ x_{16}x_{36}x_{34} - x_{13}x_{45}x_{34} + x_{16}x_{45}x_{34} + x_{36}x_{45}x_{34} + x_{13}x_{46}x_{34} - 2x_{15}x_{46}x_{34}\\
&+ x_{16}x_{46}x_{34} + x_{35}x_{46}x_{34} - 2x_{36}x_{46}x_{34} + x_{13}x_{56}x_{34} + x_{14}x_{56}x_{34} - 2x_{16}x_{56}x_{34}\\
&+ x_{36}x_{56}x_{34} + x_{46}x_{56}x_{34} - x_{14}x_{36}^2 + x_{15}x_{36}^2 - x_{13}x_{46}^2 + x_{15}x_{46}^2 - x_{35}x_{46}^2 + x_{14}x_{35}x_{36}\\
&- x_{16}x_{35}x_{36} - x_{36}^2x_{45} + x_{13}x_{36}x_{45} - 2x_{14}x_{36}x_{45} + x_{16}x_{36}x_{45} - 2x_{13}x_{35}x_{46} + x_{14}x_{35}x_{46}\\
&+ x_{16}x_{35}x_{46} + x_{13}x_{36}x_{46} + x_{14}x_{36}x_{46} - 2x_{15}x_{36}x_{46} + x_{35}x_{36}x_{46} + x_{13}x_{45}x_{46}\\
&- x_{16}x_{45}x_{46} + x_{36}x_{45}x_{46} - x_{13}x_{36}x_{56} + x_{14}x_{36}x_{56} + x_{13}x_{46}x_{56} - x_{14}x_{46}x_{56}.
\end{aligned}$$

The set of generators supported on K_5 contains more than this polynomial. There are two other available choices, of homogeneous degrees 4 or 5, which, in addition, can have quadratic degree in the elimination indeterminate x_{35}. The choice of this particular generator was done so as to minimize the complexity of (the computation of) the resultant: its homogeneous degree 3 and degree 1 in the elimination variable x_{35} are both minimal among the three available options.

At the internal nodes of the tree we compute, using resultants and factorization, irreducible polynomials in the ideal whose support matches the dependent graphs of the combinatorial tree, as follows.

The resultant $p_{D_2} = \text{Res}(p_{D_1}, p_{C_1}, x_{35})$ is an irreducible polynomial supported on the graph D_2 in Figure 12. This graph contains the final result $K_{3,3}$-plus-one as a subgraph, as well as two additional edges, which will have to be eliminated to obtain the final result. *Thus the resultant tree is not strictly increasing with respect to the set of vertices along a path, as was the case in subsection* 3.4. However, when the set of vertices remains constant (as demonstrated with this example), the dependent graphs on the path towards the root are *strictly decreasing with respect to the edge set.*

The resultant $p_{D_3} = \text{Res}(p_{D_2}, p_{C_2}, x_{13})$ is a reducible polynomial with 222 108 terms and two nonconstant irreducible factors. Only one of the factors is supported on D_3, with the other factor being supported on a minimally rigid (hence independent) graph. Thus this factor, the only one which can be in the Cayley–Menger ideal (and it must be, by primality considerations), is chosen as the new polynomial p_{D_3} with which we continue the computation.

The final step to obtain C is to eliminate the edge 46 from D_3 by a combinatorial resultant with C_3. The corresponding resultant polynomial p_C is a reducible polynomial with 15 197 960 terms and three irreducible factors. As in the previous step, the analysis of the supports of the irreducible factors shows that only one factor is supported on the $K_{3,3}$-plus-one circuit, while the other two factors are supported on minimally rigid graphs. This unique irreducible factor is the desired circuit polynomial for the $K_{3,3}$-plus-one circuit.

The computational time on a 2019 iMac with 6 CPU cores at 3.7GHz in *Mathematica* v13, including factorizations to irreducible components was 979.42 seconds. The computation and factorization of the final resultant step took up most of the computational time (562.5 and 394.9 seconds, resp.).

14. Concluding remarks.

In this paper we introduced the combinatorial resultant operation, analogous to the classical resultant of polynomials. We offer here some final comments and suggestions for further research.

Irreducibility test. Our methods still have several computational drawbacks, in that they require irreducibility checks, with a possible further factorization and an ideal membership test for those factors that have the support of a circuit.

Ideally we would like to detect combinatorially when a resultant of two circuit polynomials that has the support of a circuit will be irreducible. The absolute irreducibility test of Gao [20], which states that a polynomial is absolutely irreducible if and only if its Newton polytope is integrally indecomposable, in conjunction with the description of the Newton polytope of the resultant of two polynomials by Gelfand, Kapranov, and Zelevinsky [21, 22], gives a combinatorial criterion for absolute irreducibility, but not for irreducibility over $\mathbb{Q}$. However, not every circuit polynomial is absolutely irreducible; for example, the circuit polynomial of a wheel on 4 vertices is irreducible over $\mathbb{Q}$ but not absolutely irreducible.

What we observed in practice. It is worth noticing that whenever in our computations we had to decide which factor of a resultant belonged to CM_n, we never had to perform an ideal membership test. It was always sufficient to inspect only the supports of the irreducible factors of the resultant. In all cases where the calculation succeeded, all but one irreducible factor were supported on Laman graphs, and one factor was supported on a dependent set. It seems unlikely that this is the general case, and it would be of interest to determine under which conditions the resultant has exactly one factor (up to multiplicity) supported on a dependent set in $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$.

Open problems. We conclude the paper with a few more open problems concerning the algebraic and geometric structure of the resultant of two circuit polynomials.

Open Problem 49. Let A, B, and C be circuits such that $C = \mathrm{CRes}(A, B, e)$. Let p_C, p_A, and p_B be the corresponding circuit polynomials. Under which conditions is it the case that $\mathrm{Res}(p_A, p_B, x_e)$ is of the form $\alpha \cdot p_C^m$ for $m \geq 1$ with $\alpha \in \mathbb{Q}$?

Open Problem 50. More generally, for two polynomials $p, q \in \mathrm{CM}_n$ with $x_e \in \operatorname{supp} p \cap \operatorname{supp} q$, under which conditions has the resultant exactly one irreducible factor supported on a dependent set in $\boldsymbol{\mathcal{A}}(\mathrm{CM}_n)$?

Open Problem 51. *Generalize Proposition* 48 *to the question of whether reducibility of* $\mathrm{Res}(p_A, p_B, x_e)$ *can be inferred from graph-theoretic data (circuits* C, A, B *and edge* e *such that* $C = \mathrm{CRes}(A, B, e)$*).*

This question appears to be very challenging. The answer depends heavily on the specific polynomials p_A, p_B and the variable x_e and pertains to the relationship between (affine) varieties related to $r = \mathrm{Res}(p_A, p_B, x_e)$, p_A, and p_B. Let $R = \mathbb{C}[C]$ be a polynomial ring, let $p_A, p_B \in R[x_e]$, and let I_{x_e} denote the elimination ideal $\langle p_A, p_B\rangle \cap R$. Let l_A (resp., l_B) be the leading coefficient of p_A (resp., p_B) with respect to x_e. Then by the Extension Theorem [13, Theorem 8 in section 6 of Chapter 3] and the Closure Theorem [13, Theorem 4 in section 4 of Chapter 4] we have the following equality of (affine) varieties: $V(r) = V(l_A, l_B) \cup V(I_{x_e})$. Furthermore, if r factors as $q \cdot p_C^k$ for some positive integer k, then $V(r) = V(q) \cup V(p_C)$. Ideally we would want $V(r) = V(p_C) = V(I_{x_e})$ but in general $V(p_C)$ is only contained in $V(I_{x_e})$ and $V(r)$. Hence the structure of $V(r)$, in particular its irreducibility, depends on *algebraic* data $V(l_A, l_B)$ and $V(I_{x_e})$, whose relationship to the combinatorial, graph-theoretical data is yet to be found.

Further interesting questions pertain to parameters of circuit polynomials such as the degree in a single variable or the number of monomials. The first one, the degree with respect to a single variable x_e in the support of a circuit polynomial, is related to the literature on the number of embeddings of Laman graphs, where the best known upper bound is 3.77^n [2] for n vertices. Bounds on the degree of an individual indeterminate of a 3-connected circuit polynomial can be inferred from here, while for the 2-connected ones their decomposition into 3-connected components is needed. On the other hand, we are not aware of any such bounds on the number of monomial terms of circuit polynomials, but have observed that their number quickly becomes large, as shown in the Table 1.

Open Problem 52. *How big do circuit polynomials get, i.e., what are upper and lower bounds on the number of monomial terms relative to the number of vertices* n*?*

Open Problem 53. *When working with an extended collection of generators, not all of them circuits (such as those from section* 11*), decide if a given circuit has a combinatorial resultant tree with at least one non-*K_4 *leaf from the given generators.*

Acknowledgments. We would like to thank the anonymous reviewers for their comments, which have helped improve the presentation, and for suggesting references that have increased the scope of the paper.

REFERENCES

[1] E. Bartzos, I. Z. Emiris, J. Legerský, and E. Tsigaridas, *On the maximal number of real embeddings of minimally rigid graphs in* R^2, R^3 *and* S^2, J. Symb. Comput., 102 (2021), pp. 189–208, https://doi.org/10.1016/j.jsc.2019.10.015.

[2] E. Bartzos, I. Z. Emiris, and C. Tzamos, *The m-Bézout bound and distance geometry*, in Computer Algebra in Scientific Computing, F. Boulier, M. England, T. M. Sadykov, and E. V. Vorozhtsov, eds., Springer, Cham, 2021, pp. 6–20, https://doi.org/10.1007%2F978-3-030-85165-1_2.

[3] D. J. Bates, A. J. Sommese, J. D. Hauenstein, and C. W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, SIAM, 2013, https://doi.org/10.1137/1.9781611972702.

[4] D. Bayer and D. Mumford, *What can be computed in algebraic geometry?*, in Computational Algebraic Geometry and Commutative Algebra, D. Eisenbud and L. Robbiano, eds., Cambridge University Press, 1993, pp. 1–48.

[5] A. R. Berg and T. Jordán, *A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid*, J. Comb. Theory Ser. B, 88 (2003), pp. 77–97, https://doi.org/10.1016/S0095-8956(02)00037-0.

[6] L. Blumenthal, *Theory and Applications of Distance Geometry*, AMS Chelsea Publishing Series, Chelsea Publishing, 1970.

[7] G. P. Bollen, J. Draisma, and R. Pendavingh, *Algebraic matroids and Frobenius flocks*, Adv. Math., 323 (2018), pp. 688–719, https://doi.org/10.1016/j.aim.2017.11.006.

[8] C. S. Borcea, *Point Configurations and Cayley-Menger Varieties*, https://arxiv.org/abs/math/0207110, 2002.

[9] C. S. Borcea and I. Streinu, *The number of embeddings of minimally rigid graphs*, Discrete Comput. Geom., 31 (2004), pp. 287–303, https://doi.org/10.1007/s00454-003-2902-0.

[10] B. Buchberger, *Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems*, Aequationes Math., 4 (1970), pp. 374–383, https://doi.org/10.1007/BF01844169.

[11] J. Capco, M. Gallet, G. Grasegger, C. Koutschan, N. Lubbes, and J. Schicho, *The number of realizations of a Laman graph*, SIAM J. Appl. Algebra Geom., 2 (2018), pp. 94–125, https://doi.org/10.1137/17M1118312.

[12] D. Cartwright, *Construction of the Lindström valuation of an algebraic extension*, J. Comb. Theory Ser. A, 157 (2018), pp. 389–401, https://doi.org/10.1016/j.jcta.2018.03.003.

[13] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Undergrad. Texts Math., 4th ed., Springer, Cham, 2015.

[14] G. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*, Chemometrics Research Studies Press Series, Research Studies Press, 1988.

[15] A. Dickenstein, N. Fitchas, M. Giusti, and C. Sessa, *The membership problem for unmixed polynomial ideals is solvable in single exponential time*, Discrete Appl. Math., 33 (1991), pp. 73–94, https://doi.org/10.1016/0166-218X(91)90109-A.

[16] A. Dress and L. Lovász, *On some combinatorial properties of algebraic matroids*, Combinatorica, 7 (1987), pp. 39–48, https://doi.org/10.1007/BF02579199.

[17] R. Ehrenborg and G.-C. Rota, *Apolarity and canonical forms in homogeneous polynomials*, European J. Combin., 14 (1993), pp. 157–181, https://doi.org/10.1006/eujc.1993.1022.

[18] I. Emiris and B. Mourrain, *Computer algebra methods for studying and computing molecular conformations*, Algorithmica, 25 (1999), pp. 372–402, https://doi.org/10.1007/PL00008283.

[19] I. Z. Emiris, E. P. Tsigaridas, and A. Varvitsiotis, *Mixed volume and distance geometry techniques for counting Euclidean embeddings of rigid graphs*, in Distance Geometry. Theory, Methods, and Applications, A. Mucherino, C. Lavor, L. Liberti, and N. Maculan, eds., Springer, New York, Heidelberg, Dordrecht, London, 2013, pp. 23–46, https://doi.org/10.1007/978-1-4614-5128-0.

[20] S. Gao, *Absolute irreducibility of polynomials via Newton polytopes*, J. Algebra, 237 (2001), pp. 501–520, https://doi.org/10.1006/jabr.2000.8586.

[21] I. Gelfand, M. Kapranov, and A. Zelevinsky, *Newton Polytopes of the Classical Resultant and Discriminant*, Adv. Math., 84 (1990), pp. 237–254, https://doi.org/10.1016/0001-8708(90)90047-Q.

[22] I. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, Resultants, and Multidimensional Determinants*, Modern Birkhäuser Classics, Birkhäuser Boston, 2009, https://doi.org/10.1007/978-0-8176-4771-1.

[23] G. Giambelli, *Sulle varietá rappresentate coll'annullare determinanti minori contenuti in un determinante simmetrico od emisimmetrico generico di forme*, Atti R. Acc. Sci. Torino, 44 (1905/06), pp. 102–125.

[24] J. Graver, B. Servatius, and H. Servatius, *Combinatorial Rigidity*, Grad. Stud. in Math. 2, American Mathematical Society, Providence, RI, 1993.

[25] P. A. Griffiths and J. Harris, *Principles of Algebraic Geometry*, Wiley, New York, 1994.

[26] J. Harris and L. Tu, *On symmetric and skew-symmetric determinantal varieties*, Topology, 23 (1984), pp. 71–84, https://doi.org/10.1016/0040-9383(84)90026-0.

[27] L. Henneberg, *Die graphische Statik der starren Systeme*, B. G. Teubner, 1911.

[28] J. E. Hopcroft and R. E. Tarjan, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158, https://doi.org/10.1137/0202012.

[29] A. Ingleton, *Representation of matroids*, in Combinatorial Mathematics and Its Applications (Proceedings of a Conference held at the Mathematical Institute, Oxford, 1969), D. Welsh, ed., Academic Press, 1971, pp. 149–167.

[30] T. Józefiak, A. Lascoux, and P. Pragacz, *Classes of determinantal varieties associated with symmetric and skew-symmetric matrices*, Math. USSR Izvestija, 18 (1982), pp. 575–586, https://doi.org/10.1070/im1982v018n03abeh001400.

[31] E. Kaltofen, *Polynomial factorization* 1987–1991, in LATIN 1992, Lecture Notes in Comput. Sci., I. Simon, ed., Springer, Berlin, Heidelberg, 1992, pp. 294–313, https://doi.org/10.1007/BFb0023837.

[32] Y. N. Lakshman, *A single exponential bound on the complexity of computing Gröbner bases of zero dimensional ideals*, in Effective Methods in Algebraic Geometry, T. Mora and C. Traverso, eds., Birkhäuser Boston, Boston, MA, 1991, pp. 227–234, https://doi.org/10.1007/978-1-4612-0441-1_15.

[33] G. Laman, *On graphs and rigidity of plane skeletal structures*, J. Eng. Math., 4 (1970), pp. 331–340, https://doi.org/10.1007/BF01534980.

[34] S. Lang, *Algebra*, 3rd ed., Grad. Texts in Math. 211, Springer-Verlag, New York, 2002.

[35] A. Lee-St John and I. Streinu, *Pebble game algorithms and sparse graphs*, Discrete Math., 308 (2008), pp. 1425–1437, https://doi.org/10.1016/j.disc.2007.07.104.

[36] T. Y. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Acta Numer., 6 (1997), pp. 399–436, https://doi.org/10.1017/S0962492900002749.

[37] G. Malić and I. Streinu, *Combinatorial resultants in the algebraic rigidity matroid*, in 37th International Symposium on Computational Geometry (SoCG 2021), Leibniz Internat. Proc. Inform. (LIPIcs) 189, K. Buchin and E. Colin de Verdière, eds., Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 52:1–52:16, https://doi.org/10.4230/LIPIcs.SoCG.2021.52.

[38] G. Malić and I. Streinu, *Faster Algorithms for Circuits in the Cayley-Menger Algebraic Matroid*, Technical report, https://arxiv.org/abs/2111.14307, 2021

[39] G. Malić and I. Streinu, *CayleyMenger - Circuit Polynomials in the Cayley Menger ideal*, GitHub repository, 2023, https://github.com/circuitPolys/CayleyMenger.

[40] G. Matera and J. M. Turull Torres, *The space complexity of elimination theory: Upper bounds*, in Foundations of Computational Mathematics (Rio de Janeiro, 1997), Springer, Berlin, 1997, pp. 267–276, https://doi.org/10.1007/978-3-642-60539-0_20.

[41] J. C. Maxwell, *On the calculation of the equilibrium and stiffness of frames*, Philos. Mag., 27 (1864), pp. 294–299, https://doi.org/10.1080/14786446408643668.

[42] K. Menger, *New Foundation of Euclidean Geometry*, Amer. J. Math., 53 (1931), pp. 721–745, https://doi.org/10.2307/2371222.

[43] J. Oxley, *Matroid Theory*, 2nd ed., Oxford Grad. Texts in Math. 21, Oxford University Press, Oxford, 2011.

[44] H. Pollaczek-Geiringer, *Über die Gliederung ebener Fachwerke*, ZAMM, 7 (1927), pp. 58–72, https://doi.org/10.1002/zamm.19270070107.

[45] Z. Rosen, *Algebraic Matroids in Applications*, Ph.D. thesis, University of California, Berkeley, 2015, https://digitalassets.lib.berkeley.edu/etd/ucb/text/Rosen_berkeley_0028E_15261.pdf.

[46] Z. Rosen, *algebraic-matroids*, GitHub repository, 2017, https://github.com/zvihr/algebraic-matroids.

[47] Z. Rosen, J. Sidman, and L. Theran, *Algebraic matroids in action*, Amer. Math. Monthly, 127 (2020), pp. 199–216, https://doi.org/10.1080/00029890.2020.1689781.

[48] M. Sitharam and H. Gao, *Characterizing graphs with convex and connected Cayley configuration spaces*, Discrete Comput. Geom., 43 (2010), pp. 594–625, https://doi.org/10.1007/s00454-009-9160-8.

[49] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, 2005, https://doi.org/10.1142/5763.

[50] I. Streinu and L. Theran, *Slider-pinning rigidity: A Maxwell-Laman-type theorem*, Discrete Comput. Geom., 44 (2010), pp. 812–834, https://doi.org/10.1007/s00454-010-9283-y.

[51] W. T. Tutte, *Connectivity in Graphs*, Toronto University Press, Toronto, 1966.

[52] B. L. van der Waerden, *Modern Algebra*, 2nd ed., Springer, Berlin, Heidelberg, New York, 1967; translated by F. Blum and J. R. Schulenberger.

[53] B. L. van der Waerden, *Algebra*, Vol. 2, Frederick Ungar Publishing, New York, 1970; translated by J. R. Schulenberger.

[54] D. Walter and M. Husty, *On a nine-bar linkage, its possible configurations and conditions for flexibility*, in Proceedings of IFToMM 2007, J.-P. Merlet and M. Dahan, eds., Besançon, France, 2007, http://geometrie.uibk.ac.at/cms/datastore/husty/A681.pdf.

[55] W. Whiteley, *Some matroids from discrete applied geometry*, in Matroid Theory, Contemp. Math. 197, J. Bonin, J. G. Oxley, and B. Servatius, eds., American Mathematical Society, 1996, pp. 171–311, https://doi.org/10.1090/conm/197/02540.