Faster algorithms for circuits in the Cayley-Menger algebraic matroid

Goran Malić

Computer Science Department, Smith College, Northampton, MA, USA

Ileana Streinu*

Computer Science Department, Smith College, Northampton, MA, USA

Abstract

A classical problem in Distance Geometry, with multiple practical applications (in molecular structure determination, sensor network localization etc.) is to find the possible placements of the vertices of a graph with given edge lengths. For minimally rigid graphs, the double-exponential Gröbner Bases algorithm with an elimination order can be applied, in theory, but it is impractical even for small instances. By relating the problem to the computation of circuit polynomials in the Cayley-Menger ideal, [23] recently proposed an algebraic-combinatorial approach and an elimination algorithm for circuit polynomials. It is guided by a tree structure whose leaves correspond to complete \( K_4 \) graphs and whose nodes perform algebraic resultant operations.

In this paper we uncover further combinatorial structure in the Cayley-Menger algebraic matroid that leads to an extension of their algorithm. In particular, we generalize the combinatorial resultant operation of [23] to take advantage of the non-circuit generators and irreducible polynomials in the Cayley-Menger ideal and use them as leaves of the tree guiding the elimination. Our new method has been implemented in Mathematica and allows previously unobtainable calculations to be carried out. In particular, the \( K_{3,3} \)-plus-one circuit polynomial, with over one million terms in 10 variables and whose calculation crashed after several days with the previous method of [23], succeeded now in approx. 30 minutes.

Keywords: Cayley-Menger ideal, rigidity matroid, circuit polynomial, combinatorial resultant, inductive construction, Gröbner basis elimination

*Corresponding author

Email addresses: gmalic@smith.edu (Goran Malić), istreinu@smith.edu (Ileana Streinu)
1. Introduction.

Given a graph \( G = (V, E) \) and a collection of edge weights \( \{w_e | e \in E\} \), the main problem in Distance Geometry \([10, 20, 1]\) asks for placements of the vertices \( V \) in some Euclidean space (in this paper, 2D) in such a way that the resulting edge lengths match the given weights. The problem has many applications, in particular in molecular structure determination \([12, 24]\) and sensor network localization \([30]\). A polynomial-time related problem is to find the possible values of a single unknown distance corresponding to a non-edge (a pair of vertices that are not connected by an edge in \( G \)). A system of quadratic equations can be easily set up so that the possible values of the unknown distance are the (real) solutions of this system. In theory, the general but impractical (double-exponential time) Gröbner basis algorithm with an elimination order can be used to solve the system. The set of solutions can be finite (if the given weighted graph is rigid) or continuous (if it is flexible).

By formulating the Single Unknown Distance Problem in Cayley coordinates (with variables \( x_{ij} \) corresponding to edges \( ij \in E \)), the problem can be reduced to finding a specific irreducible polynomial in the Cayley-Menger ideal, called the circuit polynomial. The unknown distance is then computed by solving the uni-variate polynomial obtained by substituting the given edge lengths in the circuit polynomial.

The topic of this paper is the effective computation of circuit polynomials in the 2D Cayley-Menger ideal. These are multi-variate polynomials in the ideal generated by the rank conditions of the Cayley matrix associated to a generic set of \( n \) points. The supports of such polynomials correspond to a family of sparse graphs called rigidity circuits, i.e. circuits in the (combinatorial) rigidity matroid. They are particularly difficult to compute, as the number of terms in a circuit polynomial grows very rapidly as a function of the number of nodes in the underlying graph.

Background and related work. The literature on distance geometry is vast, with classical results \([4, 10]\) (some going back to Cayley in the 19th century), recent applications \([6, 24, 20]\) and new techniques emerging from various areas: theory of semidefinite programming, distance matrices, rigidity theory, combinatorics and computational algebra \([11, 31, 29, 15, 8, 2]\). Our paper relies on insights from the theory of matroids \([25, 31]\), as well as from the distinguished properties of circuit polynomials in algebraic matroids \([11]\). A computational perspective to circuit polynomials, recently popularized by \([28]\), emerged from \([26]\). There, circuits in algebraic matroids of arbitrary polynomial ideals were studied and small cases were explored using Macaulay2 code \([27]\). They also pointed out the smallest circuit polynomial example in the context of 2D distance geometry. This polynomial was the only known example until last year: it is supported by the \( K_4 \) graph and, being a generator of the Cayley-Menger ideal, needs no calculation.

Computing other circuit polynomials would require the double-exponential time Gröbner Bases algorithm with an elimination order. The longest successful
calculation carried out in this fashion (5 days and 6 hrs, using Mathematica’s
GroebnerBases function and some combinatorial considerations) was reported
in [23] for the Desargues-plus-one circuit graph from Fig. 1(left).

The starting point of our paper is a recent result of [23], who proposed a
mixed combinatorial-algebraic approach that led to faster calculations of non-
trivial examples. This algorithm is guided by a tree structure whose leaves cor-
respond to complete $K_4$ graphs and whose nodes correspond to rigidity circuits
obtained by a combinatorial operation abstracted from the classical algebraic
(Sylvester) resultant. This method, best described in the language of alge-
braic matroids, uncovers a rich algebraic-combinatorial structure in the Cayley-
Menger ideal. The resulting algorithm significantly outperforms the general
method: the previous example that took over 5 days to compute with Groeb-
nerBases was completed in less than 15 seconds.

![Figure 1: The Desargues-plus-one and the $K_{3,3}$-plus-one rigidity circuits.](image)

**Our Results.** In this paper we generalize the combinatorial resultant tree
introduced in [23] from circuits to dependent sets in the rigidity matroid. This
allows us to extend the algorithm of [23] to take advantage of the other non-
circuit generators of the Cayley-Menger ideal besides those supported on $K_4$
graphs. We show that these extended trees can lead to more tractable algebraic
calculations than those based exclusively on circuits. Implemented in Matha-
matica, executions that were prohibitively expensive using the previous method
now ran to completion. In particular, the polynomial for the $K_{3,3}$-plus-one
circuit (Fig. 1 right), with over one million terms in 10 variables and whose
calculation crashed after several days with the method of [23], succeeded now
in approx. 30 minutes.

**Overview of the paper.** In Section 2 we briefly introduce the necessary
discrete structures from 2D rigidity theory, matroids and combinatorial resul-
tants. The extended combinatorial resultant tree is defined and characterized
in Section 3. A brief introduction to algebraic matroids, elimination ideals and
resultants, the Cayley-Menger ideal and its circuit polynomials appears in Sec-
tion 4. We discuss the dependent, non-circuit generators of the Cayley-Menger
ideal in Section 5. To help build the intuition on how our algorithm works,
we demonstrate in Section 6 how we succeeded in computing the $K_{3,3}$-plus-one
circuit polynomial. The relevant step of the extended algorithm for computing
circuit polynomials is described in Section 7. We conclude with a few remaining
open questions in Section 8.
2. Preliminaries: rigid graphs and matroids.

In this section we introduce the essential, previously known concepts and results from combinatorial rigidity theory in 2D and from [23] that are relevant for setting up the foundation of our paper and carrying out the proofs. We work with (sub)graphs \( G \) given by subsets \( E \) of edges of the complete graph \( K_n \) on vertices \( [n] := \{1, \cdots, n\} \). We denote with \( V(G) \), resp. \( E(G) \) the vertex, resp. edge set (support) of \( G \). Let \( V(E) \) be the vertex span of edges \( E \), i.e. the set of all vertices appearing in \( E \). A subgraph \( G \) is spanning if its edge set \( E(G) \) spans \( [n] \). The neighbours \( N(v) \) of a vertex \( v \) are the vertices connected to \( v \) by an edge in \( G \).

**Frameworks.** A 2D bar-and-joint framework is a pair \( (G, p) \) of a graph \( G = (V, E) \) and a planar point set \( p \). Its vertices \( V = [n] \) are mapped to points \( p = \{p_1, \ldots, p_n\} \) in \( \mathbb{R}^2 \) via the placement map \( p: V \to \mathbb{R}^2 \) given by \( i \mapsto p_i \). We view the edges as rigid bars and the vertices as rotatable 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 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 in exactly one configuration, and flexible otherwise.

Combinatorial rigidity theory of bar-and-joint frameworks seeks to understand the rigidity and flexibility of frameworks in terms of their underlying graphs. The following theorem [19] relates the rigidity of 2D bar-and-joint frameworks to a specific type of graph sparsity: the hereditary property (b) below is also referred to as the \((2, 3)\)-sparsity condition.

**Theorem 1.** A bar-and-joint framework \( (G, p) \) is generically minimally rigid in 2D iff its underlying graph \( G = (V, E) \) satisfies two conditions: (a) it has exactly \( |E| = 2|V| - 3 \) edges, and (b) any proper subset \( V' \subset V \) of vertices spans at most \( 2|V'|-3 \) edges.

The genericity condition appearing in the statement of this theorem refers to the vertex placement \( p \). Without going into details, we retain its most important consequence, namely that small perturbations of \( p \) do not change the rigidity or flexibility properties of a generic framework. This theorem allows for the rigidity and flexibility of generic frameworks to be studied in terms of only their underlying graphs.

A graph satisfying the conditions of this theorem is called a Laman graph. It is minimally rigid in the sense that it has just enough edges to be rigid: if one edge is removed, it becomes flexible. Adding extra edges to a Laman graph keeps it rigid, but the minimality is lost: these graphs are said to be rigid and overconstrained. In short, for a graph to be rigid, its vertex set must span a
Laman graph; otherwise the graph is flexible. A flexible graph decomposes into edge-disjoint rigid components, each component spanning on its vertex set a Laman graph with (possibly) additional edges. Fig. 2 and Fig. 3 illustrate the concepts. These properties have matroidal interpretations.

Figure 3: (Left) A dependent graph containing a circuit (highlighted). (Middle) Its redundant edges are in black and its critical edges are colored green. (Right) The removal of a critical edge creates a flexible graph, whose two edge-disjoint rigid components are highlighted: orange (dependent) and green (independent).

**Matroids.** A matroid \cite{25} 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 by 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 \cite{25} (skipped here, since they are not relevant for our presentation). 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 two types of matroids: a graphic rigidity matroid, defined on a ground set given by all the edges \( E_n := \{ij : 1 \leq i < j \leq n\} \) of the complete graph \( K_n \); this is the \((2,3)\)-sparsity matroid or the generic rigidity matroid described below; and an algebraic matroid, defined on an isomorphic ground set of variables \( X_n := \{x_{ij} : 1 \leq i < j \leq n\} \); this is the algebraic rigidity matroid associated to the Cayley-Menger ideal that will be defined in ...
Section 4 Two examples of (spanning) circuits in the (graphic) rigidity matroid on 6 vertices were given in Fig. [1]

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 $S_n$ on the ground set $E$ of edges of the complete graph $K_n$, called the (generic) 2D rigidity matroid, or the (2,3)-sparsity matroid [15]. The bases of the matroid $S_n$ are the maximal independent sets, hence the Laman graphs. A set of edges which is not sparse is a dependent set. A minimal dependent set is a (sparsity) circuit. The edges of a circuit span a subset of the vertices of $V$. For instance, adding one edge to a Laman graph creates a dependent set of $2n - 2$ edges, called a Laman-plus-one graph (Fig. 2), which contains a unique circuit. A circuit spanning $V$ is said to be a maximal or spanning circuit in the sparsity matroid $S_n$; it has $2n - 2$ edges on $n$ vertices and satisfies the Laman (2,3)-sparsity property on all strict subsets of $n' < n$ vertices. Examples are given in Fig. 2.

A dependent rigid graph is a dependent graph which contains a spanning Laman graph; for example, the three rightmost graphs in Fig. 2. An edge $e \in G$ is said to be redundant if the graph $G \{e\}$ is still rigid, otherwise the edge is said to be critical: its removal makes the graph flexible. It is well known that (a) all the edges of a circuit are redundant and that (b) a redundant edge is contained in a subgraph that spans a circuit (see e.g. [17]). Examples of dependent graphs with critical edges are those Laman-plus-one graphs which are not themselves spanning circuits: each one contains a unique, strict subgraph which is a circuit, and all the edges that are not in this circuit are critical (Fig. 2). We’ll make use of these concepts in Section 3.

Combining graphs and circuits: 2-sums and combinatorial resultants. We define now operations that combine two graphs (with some common vertices and edges) into one. 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 $V_\cup = V(G_1) \cup V(G_2)$, $V_\cap = V(G_1) \cap V(G_2)$, $m_\cup = |E_\cup|$ and similarly for edges, with $n_\cap = |E_\cap|$. 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 having 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. It is known [1] that the 2-sum of two circuits is a circuit, and the 2-split of a circuit is a pair of circuits. This operation, used for the Tutte decomposition of a 2-connected graph into 3-connected components, allows us to focus (when computing circuits and circuit polynomials) on the more challenging case of 3-connected graphs.

Let $G_1$ and $G_2$ be two distinct graphs with non-empty intersection $E_\cap \neq \emptyset$ and let $e \in E_\cap$ be a common edge. The combinatorial resultant [23] of $G_1$...
and $G_2$ on the elimination edge $e$ is the graph $\text{CRes}(G_1, G_2, e)$ with vertex set $V_\cup$ and edge set $E_\cup \setminus \{e\}$. The 2-sum is the simplest kind of combinatorial resultant.

![Diagram of a 2-sum](image)

**Figure 4:** (Bottom) A 4-wheel $W_4$ and a complete $K_4$ graph, with their common Laman graph (a triangle) shown in red, the elimination edge (in a circle) and (Top) their combinatorial resultant, the 5-wheel $W_5$ circuit.

**Circuit Resultant Trees.** In [23], the focus was on combinatorial resultants that produce circuits from circuits. It was proven there that the combinatorial resultant of two circuits has $m = 2n - 2$ edges iff the common subgraph $G_\cap$ of the two circuits is Laman, in which case the two circuits are said to be properly intersecting. If the combinatorial resultant operation applied to two properly intersecting circuits results in a spanning circuit, we say that it is circuit-valid (as in Fig. 4). It was further shown in [23] that each circuit can be obtained via circuit-valid combinatorial resultant operations from a collection of $K_4$ circuits on subsets of 4 vertices whose union is the span of the desired circuit. This construction (illustrated in Fig. 5 for the $K_3,3$-plus-one circuit) is captured by a tree (technically, an algebraic expression tree, where the operator at each internal node is a combinatorial resultant). It has the following properties: (a) all its nodes are labeled by circuits, with $K_4$ graphs at the leaves; (b) each internal node corresponds to a combinatorial resultant operation that combines the circuits of the node’s two children and eliminates the edge marked under the node. An additional property of the tree produced by the method of [23] is that, when combining two circuits with the resultant operation, it always adds at least one new vertex. Hence the depth of the tree is at most $n - 4$; it is exactly 2 in Fig. 5 for a circuit with $n = 6$ vertices.

This type of tree was called in [23] a combinatorial resultant tree. For a more descriptive terminology, in this paper we refer to this object with an additional qualifier as a circuit (combinatorial) resultant tree, reserving the combinatorial resultant tree for the more general structure defined in the next section.
Figure 5: A combinatorial circuit resultant tree for $K_{3,3}$-plus-one: all of its nodes are circuits.
3. Combinatorial Resultant Trees.

We relax now some of the constraints imposed on the resultant tree by the construction from [23]. 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 1. A finite collection $Gen$ of dependent graphs such that $K_4 \in Gen$ will be called a set of generators.

The generators will be the graphs allowed to label the leaves. For the purpose of generating (combinatorial) circuits and computing (algebraic) circuit polynomials, meaningful sets of generators will be discussed in Section 5. We restrict the generators to graphs which are dependent in the rigidity matroid because these correspond precisely to the supports of polynomials in the Cayley-Menger ideal, as we will see in Sections 4 and 5.

Definition 2. A combinatorial resultant 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 $i$ (marked with a graph $G_i$ and an edge $e_i \not\in G_i$) corresponds to a combinatorial resultant operation applied on the two graphs $G_1, G_2$ labeling its children. Specifically, $G_i = CRes(G_1, G_2, e_i)$, where the edge $e_i \in G_1 \cap G_2$.

An example is illustrated in Fig. 6.

Lemma 1. 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 root. For the inductive step, assume that $G_1$ and $G_2$ are the dependent graphs labeling the two children of a node labeled with $G = CRes(G_1, G_2, e)$, where $e \in E_n$ is an edge in the common intersection $G_1 \cap G_2$. We consider two cases, depending on whether $e$ is critical in none or 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, 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 $C_1 \subseteq G_1$ and $C_2 \subseteq G_2$, 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 $2n_{ij} - 2$ edges (cf. the proof of Lemma 3.1. in [23]), hence it is dependent.
Case 2: $e$ is critical either in $G_1$ or 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 $R$, $R$ remains dependent in the resultant graph $G = CRes(G_1, G_2, e)$. □

Definition 3. 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 Fig. [6] 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 Section [6] to demonstrate our extended algebraic elimination algorithm, which will be then described in Section [7].
4. Preliminaries: The Cayley-Menger Ideal and Algebraic Matroid.

We turn now to the algebraic aspects of our problem. For full details, the reader may consult the comprehensive sections 2-6 of [21].

Polynomial ideals. Let $X$ be an arbitrary set of variables. 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 $\mathbb{Q}[X]$. A generating set for an ideal is a set $S \subset \mathbb{Q}[X]$ of polynomials such that every polynomial in the ideal is an algebraic combination of elements in $S$ with coefficients in $\mathbb{Q}[X]$. Ideals generated by a single polynomial are called principal. An ideal $I$ is said to be prime 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 non-constant polynomials in $\mathbb{Q}[X]$. A principal ideal is prime iff it is generated by an irreducible polynomial. However, an ideal generated by two or more irreducible polynomials is not necessarily prime.

Elimination ideals. If $I$ is an ideal of $\mathbb{Q}[X]$ and $X' \subset X$ non-empty, then 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 [7, 9] which give a general approach for computing elimination ideals: if $G$ is a Gröbner basis for $I$ with respect to an elimination order, e.g. $x_{i_1} > x_{i_2} > \cdots > x_{i_n}$, 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 $G \cap \mathbb{Q}[x_{i_{k+1}}, \ldots, x_{i_n}]$ as its Gröbner basis. If $I$ is a prime ideal of $\mathbb{Q}[X]$ and $X' \subset X$ is non-empty, then the elimination ideal $I \cap \mathbb{Q}[X']$ is prime.

Algebraic matroid of a prime ideal. Intuitively, a collection of variables $X'$ is independent with respect to an ideal $I \subset \mathbb{Q}[X]$ if it is not constrained by any polynomial in $I$, and dependent otherwise. The algebraic matroid induced by the ideal is, informally, a matroid on the ground set of variables $X$ 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.

Circuits and circuit polynomials. A circuit in a matroid is a minimal dependent set. In an algebraic matroid on ground set $X$, a circuit $C \subset X$ is a minimal set of variables supported by a polynomial in the prime ideal $I$ defining the matroid. A polynomial whose support is a circuit $C$ is called a circuit polynomial and is denoted by $p_C$. A theorem of Dress and Lovasz [11] states that, up to multiplication by a constant, a circuit polynomial $p_C$ is the unique polynomial in the ideal with the given support $C \subset X$. We’ll just say, shortly, that it is unique. Furthermore, the circuit polynomial is irreducible. In summary, circuit polynomials generate elimination ideals supported on circuits.
The Cayley-Menger ideal. When working with the Cayley-Menger ideal we use variables $X_n = \{x_{i,j} : 1 \leq i < j \leq n\}$ for unknown squared distances between pairs of points. 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.

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

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-Menger matrix should be zero. These minors induce polynomials in $\mathbb{Q}[X_n]$ and are the standard generators of the $(n,d)$-Cayley-Menger ideal $\text{CM}^d_n$. They are homogeneous polynomials with integer coefficients and are irreducible over $\mathbb{Q}$. The $(n,d)$-Cayley-Menger ideal is a prime ideal of dimension $dn - (\binom{d+1}{2})$ and codimension \((n^2) - dn + (d+1)\). In this paper we work with $d = 2$ and denote by $\text{CM}_n$ the $(n,2)$-Cayley-Menger ideal. Its generators are tabulated in Section 5. The algebraic matroid $\mathcal{A}(\text{CM}^d_n)$ of the Cayley-Menger ideal is the matroid on the ground set $X_n$, where a subset of distance variables $X \subseteq X_n$ is independent if $\text{CM}^d_n \cap \mathbb{Q}[X] = \{0\}$, i.e. $X$ supports no polynomial in the ideal. The rank of $\mathcal{A}(\text{CM}^d_n)$ is equal to $\dim \text{CM}^d_n = dn - (\binom{d+1}{2})$. In particular, the rank of $\text{CM}_n$ is the Laman number $2n - 3$. In fact, the algebraic matroid $\mathcal{A}(\text{CM}^d_n)$ is isomorphic to the (combinatorial) rigidity matroid introduced in Section 2. The isomorphism maps Cayley variables $x_{i,j}$ to edges $ij$, thus the support of a polynomial in $\text{CM}_n$ is in one-to-one correspondence with a subgraph of the complete graph $K_n$. All polynomials in $\text{CM}_n$ are, by definition, dependent and their supports define dependent graphs in the rigidity matroid. The supports of circuit polynomials are rigidity circuits.

Resultants and elimination ideals. The resultant is the determinant of the Sylvester matrix associated to the coefficients of two uni-variate polynomials, see e.g. [9]. We skip here the technical details, retaining only the key property that the resultant of two multi-variate polynomials relative to a common variable $x$ is a new polynomial in the union of the variables of the two polynomials, minus $x$. If $R$ is a polynomial ring and $I$ is an ideal in $R[x]$ with $f, g \in I$, then $\text{Res}(f, g, x)$ is in the elimination ideal $I \cap R$ of the ring $R$. In the context of the Cayley-Menger ideal, the effect (on the supports of the involved

\[1\text{Historically, it has also been called the eliminant.}\]
polynomials) of taking a resultant is captured by the combinatorial resultant defined in Section 2.
5. **Standard generators of the 2D Cayley-Menger ideal.**

The standard generators of the 2D Cayley-Menger ideal are the $5 \times 5$ minors of the Cayley matrix. Each standard generator $g$ is identified with its support graph $G_g$. To motivate the possible choices for the family of graphs $Gen$ for the combinatorial resultant trees defined in Section 3, we now tabulate the support graphs of all standard generators, up to multiplication by an integer constant, relabeling and graph isomorphism.

To find all such support graphs, it is sufficient to consider the set of all $5 \times 5$ minors of $CM_{10}$. Using a computer algebra package we can verify that this set has 109 619 distinct 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 being shown in Section 5. The only two representatives with less than 6 vertices are $K_4$ and $K_5$. There are 3 isomorphism classes on 6, 7, 8 vertices (one is $K_6$), 2 on 9 and one on 10 vertices. The corresponding 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 (non-isomorphic) polynomials.

![Figure 7: 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.](image-url)
6. Example: the $K_{3,3}$-plus-one circuit polynomial.

As a preview of the algorithm for computing circuit polynomials that will be given in Section 7, we demonstrate now how the algorithm will use the (extended) combinatorial resultant tree from Fig. 6, to guide the computation of the circuit polynomial for the $K_{3,3}$-plus-one circuit shown at the root.

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. 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:

\[ x_{15}x^{2}_{34} - x_{16}x^{2}_{34} - x_{56}x^{2}_{34} - x_{14}x_{35}x_{34} + x_{16}x_{35}x_{34} + x_{14}x_{36}x_{34} - 2x_{15}x_{36}x_{34} \\
+ x_{16}x_{36}x_{34} - x_{13}x_{45}x_{34} + x_{16}x_{45}x_{34} + x_{36}x_{45}x_{34} + x_{13}x_{46}x_{34} - 2x_{15}x_{46}x_{34} \\
+ x_{16}x_{46}x_{34} + x_{35}x_{46}x_{34} - 2x_{36}x_{46}x_{34} + x_{13}x_{56}x_{34} + x_{14}x_{56}x_{34} - 2x_{16}x_{56}x_{34} \\
+ x_{36}x_{56}x_{34} + x_{46}x_{56}x_{34} - x_{14}x^{2}_{36} + x_{15}x^{2}_{36} - x_{13}x^{2}_{46} + x_{15}x^{2}_{46} - x_{35}x^{2}_{46} + x_{14}x_{35}x_{36} \\
- x_{16}x_{35}x_{36} - x^{2}_{36} + x_{13}x_{36}x_{45} - 2x_{14}x_{36}x_{45} + x_{16}x_{36}x_{45} - 2x_{13}x_{35}x_{46} + x_{14}x_{35}x_{46} \\
+ x_{16}x_{35}x_{46} + x_{13}x_{36}x_{46} + x_{14}x_{36}x_{46} - 2x_{15}x_{36}x_{46} + x_{35}x_{36}x_{46} + x_{13}x_{45}x_{46} \\
- x_{16}x_{45}x_{46} + x_{36}x_{45}x_{46} - x_{13}x_{36}x_{56} + x_{14}x_{36}x_{56} + x_{13}x_{46}x_{56} - x_{14}x_{46}x_{56} \]

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 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 $p_{D_{2}} = \text{Res}(g_{D_{1}}, p_{C_{1}}, x_{35})$ is an irreducible polynomial supported on the graph $D_2$ in Fig. 6. 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. \textit{Thus the resultant tree is not strictly increasing with respect to the set of vertices along a path, as was the case in [22].} However, when the set of vertices remains constant (as demonstrated with this example), the dependent graphs on the path towards the root are \textit{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 222108 terms and two non-constant 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 CM 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 AMD Ryzen 9 5950x CPU with 64GB of DDR4-
3600 RAM in Mathematica v12.3, including reducibility checks and factoriza-
tions was 1788.65 seconds. The computation and factorization of the final re-
sultant step took up most of the computational time (1023.7, resp. 748.056
seconds). The circuit polynomial for the \( K_{3,3} \)-plus-one circuit is stored in the
compressed WDX Mathematica format (file size approx. 7MB) on the GitHub
repository [22].
7. **Algorithm: circuit polynomial from combinatorial resultant tree.**

We now have all the ingredients for describing the specificities of the new algorithm for computing a circuit polynomial from a given combinatorial resultant tree for a circuit $C$. Overall, just like the algorithm of [23], it computes resultants at each node of the tree, starting with the resultants of generators of $\text{CM}_n$ 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 (non-root) nodes, as described in Algorithm 1 below. This is because the polynomials sought at non-leaf nodes, not being supported on circuits, are not necessarily irreducible polynomials supported on the desired dependent graph as was the case in [23]. Hence, conceivably, they may have factors that are not in the Cayley-Menger ideal, or there may be several choices of factors supported on the dependent graph, or a combination of these cases. 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 1** Computing a polynomial in the Cayley-Menger ideal supported on a node of a combinatorial resultant tree

**Input:** Non-leaf node $G$ of a combinatorial resultant tree $T_C$. Polynomials $v, w \in \text{CM}_n$ supported on the child nodes of $G$.

**Output:** Polynomial $p \in \text{CM}_n$ supported on $G$.

1. Compute the resultant $r = \text{Res}(v, w, x_e)$, where $x_e$ is the indeterminate to be eliminated according to $T_C$.
2. Factorize $r$ over $\mathbb{Q}$.
3. if $r$ has a unique irreducible factor $p$ in $\text{CM}_n$ supported on $G$ then
   4. return $p$
4. else if $r$ factors as $p \cdot q$ such that $p$ is possibly a reducible polynomial in the Cayley-Menger ideal supported on $G$ then
   5. return $p$
5. end if

**Summary of differences from [23].** Since we do not require to have a circuit polynomial at each node, there may be a non-unique choice among the factors of the resultant $r$ that are in the Cayley-Menger ideal and supported on the rigid graph $G$. When $r$ factors as in step 5, the following can occur:

1. $r$ is irreducible or it has a unique irreducible factor in the CM ideal supported on $G$.

2. $r$ has 2 or more irreducible factors in the CM ideal supported on $G$.

Hence we can choose $p$ to be any of the factors supported on $G$; in principle we choose the factor which will minimize the computational cost in the subsequent step. Heuristically this will be the factor with
minimal degree in the indeterminate that is to be eliminated in the subsequent step since this will keep the dimension of the determinant to be computed as small as possible. If there is more than one such factor, we choose one with the least homogeneous degree.

3. $r$ does not have an irreducible factor in the CM ideal supported on $G$. In this case, for $p$ we have to choose a reducible factor of $r$ that is in the CM ideal and supported on $G$. Such a reducible factor is not necessarily unique. If it is not unique, we choose one according to the same heuristics as in the previous case.
8. Concluding Remarks.

We have extended the algorithmic approach of [23] for computing circuit polynomials. The additional structure in the Cayley-Menger ideal that we identified allows for a systematic calculation of polynomials that were previously unattainable with the only generally available method, the double-exponential Gröbner Basis algorithm, or with the previously, recently proposed method of [23]. The new perspective on distance geometry problems offered by combinatorial resultants raises further directions of research and open questions, besides those stated by [23], whose answers may clarify the theoretical complexity of the algorithm.

We conclude the paper with a few such open problems concerning the combinatorial and algebraic structure of the (combinatorial and algebraic) resultant in the context of the Cayley-Menger ideal.

**Open Problem 1.** We work now with an extended collection of generators, not all of them circuits (such as those from Section 5). Given a circuit, decide if it has a combinatorial resultant tree with at least one non-$K_4$ leaf from the given generators.

**Open Problem 2.** Consider an intermediate node $G$ in a combinatorial resultant tree and let $r = \text{Res}(v, w, x_e)$ be the resultant supported on $G$ with respect to the polynomials supported on the child nodes of $G$, as in Algorithm 1. Find examples where $r$ factors as $r = f \cdot g$, such that neither $f$ nor $g$ are supported on $G$ and are not necessarily irreducible, or prove that this never happens.
References

[1] Abdo Y. Alfakih. *Euclidean Distance Matrices and Their Applications in Rigidity Theory*. Springer International Publishing, 2018.

[2] Evangelos Bartzos, Ioannis Z. Emiris, Jan Legerský, and Elias Tsigaridas. On the maximal number of real embeddings of minimally rigid graphs in $\mathbb{R}^2$, $\mathbb{R}^3$, and $S^2$. *Journal of Symbolic Computation*, 102:189–208, 2021.

[3] Alex R Berg and Tibor Jordán. A proof of Connelly’s conjecture on 3-connected circuits of the rigidity matroid. *Journal of Combinatorial Theory, Series B*, 88(1):77 – 97, 2003.

[4] Leonard Blumenthal. *Theory and Applications of Distance Geometry*. Oxford at the Clarendon Press, 1953.

[5] Ciprian S. Borcea. Point Configurations and Cayley-Menger Varieties, 2002.

[6] Ciprian S. Borcea and Ileana Streinu. The number of embeddings of minimally rigid graphs. *Discrete and Computational Geometry*, 31:287–303, February 2004.

[7] B. Buchberger. Ein algorithmisches Kriterium für die Lösbarekeit eines algebraischen Gleichungssystems. *Aequationes Math.*, 4:374–383, 1970.

[8] Jose Capco, Matteo Gallet, Georg Grasegger, Christoph Koutschan, Niels Lubbes, and Josef Schicho. Computing the number of realizations of a Laman graph. *Electronic notes in Discrete Mathematics*, 61:207–213, 2017.

[9] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015.

[10] Gordon M. Crippen and Timothy F. Havel. *Distance Geometry and Molecular Conformation*. John Wiley and Research Studies Press, Somerset, England, 1988.

[11] A. Dress and L. Lovász. On some combinatorial properties of algebraic matroids. *Combinatorica*, 7(1):39–48, 1987.

[12] Ioannis Z. Emiris and Bernard Mourrain. Computer algebra methods for studying and computing molecular conformations. *Algorithmica*, 25:372–402, 1999.

[13] Ioannis Z. Emiris, Elias P. Tsigaridas, and Antonios Varvitsiotis. Mixed Volume and Distance Geometry Techniques for Counting Euclidean Embeddings of Rigid Graphs. In Mucherino, Antonio and Lavor, Carlile and Liberti, Leo and Maculan, Nelson, editor, *Distance Geometry. Theory, Methods, and Applications*, chapter 2, pages 23–46. Springer, New York, Heidelberg, Dordrecht, London, 2013.
[14] G.Z. Giambelli. Sulle varietà rappresentate coll’annullare determinanti minori contenuti in un determinante simmetrico od emisimmetrico generico di forme. *Atti della R. Acc. Sci, di Torino*, 44:102–125, 1905/06. Original source for the dimension result.

[15] Jack Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1993.

[16] J. Harris and L.W. Tu. On symmetric and skew-symmetric determinantal varieties. *Topology*, 23:71–84, 1984.

[17] Bill Jackson and Tibor Jordán. Connected rigidity matroids and unique realizations of graphs. *Journal of Combinatorial Theory Series B*, 94(1):1–29, May 2005.

[18] T. Józefiak, A. Lascoux, and P. Pragacz. Classes of determinantal varieties associated with symmetric and skew-symmetric matrices. *Math. USSR Izvestija*, 18:575–586, 1982.

[19] Gerard Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.

[20] Carla Lavor and Leo Liberti. *Euclidean Distance Geometry: An Introduction*. Springer Undergraduate Texts in Mathematics and Technology. Springer, 2017.

[21] Goran Malic and Ileana Streinu. Combinatorial resultants in the algebraic rigidity matroid. arxiv 2103.08432.

[22] Goran Malić and Ileana Streinu. CayleyMenger - Circuit Polynomials in the Cayley Menger ideal, a GitHub repository. https://github.com/circuitPolys/CayleyMenger, 2020-21.

[23] Goran Malic and Ileana Streinu. Combinatorial resultants in the algebraic rigidity matroid. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:16, Dagstuhl, Germany, June 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[24] Antonio Mucherino, Carla Lavor, Leo Liberti, and Nelson Maculan, editors. *Distance Geometry: Theory, Methods and Applications*. Springer, 2013.

[25] James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, second edition, 2011.

[26] Zvi Rosen. *Algebraic Matroids in Applications*. PhD thesis, University of California, Berkeley, 2015.
[27] Zvi Rosen. algebraic-matroids, a GitHub repository. https://github.com/zvihr/algebraic-matroids, 2017.

[28] Zvi Rosen, Jessica Sidman, and Louis Theran. Algebraic matroids in action. The American Mathematical Monthly, 127(3):199–216, February 2020.

[29] Meera Sitharam and Heping Gao. Characterizing graphs with convex and connected Cayley configuration spaces. Discrete and Computational Geometry, 43(3):594–625, 2010.

[30] Anthony Man-Cho So and Yinyu Ye. Theory of semidefinite programming for sensor network localization. In SODA ’05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, pages 405–414, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[31] B. L. van der Waerden. Moderne Algebra, 2nd edition. Translated by Fred Blum and John R. Schulenberger. Springer, Berlin, Heidelberg, New York, 1967.