Algorithm xxx: Cayley Analysis of Mechanism Configuration Spaces using CayMos: Software Functionalities and Architecture

MENGHAN WANG and MEERA SITHARAM, University of Florida

For a common class of 2D mechanisms called *1-dof tree decomposable linkages*, we present a software package, `CayMos` which uses new theoretical results from Sitharam and Wang [2014] and Sitharam et al. [2011a,b] to implement efficient algorithmic solutions for: (a) meaningfully representing and visualizing the connected components in the Euclidean realization space; (b) finding a path of continuous motion between two realizations in the same connected component, with or without restricting the *realization type* (sometimes called orientation type); (c) finding two "closest" realizations in different connected components.

Categories and Subject Descriptors: I.3.5 [**Computational Geometry and Object Modeling**]: Geometric algorithms, languages, and systems

General Terms: Algorithms

Additional Key Words and Phrases: 1-degree-of-freedom linkage, computer-aided design (CAD), motion space, realization space

ACM Reference Format:
Menghan Wang and Meera Sitharam. 2014. Algorithm xxx: Cayley analysis of mechanism configuration spaces using CayMos: software functionalities and architecture. *ACM Trans. Math. Softw.* V, N, Article A (January YYYY), 8 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

A key underlying barrier in understanding underconstrained geometric constraint systems is the classical problem of representing and efficiently finding the *Euclidean realization spaces* of *1-degree-of-freedom linkages*, or *mechanisms* in 2D. A *linkage* $(G, \bar{l})$, is a graph $G = (V, E)$ with fixed length bars as edges, i.e. $\bar{l} : E \to \mathbb{R}$. A 2D *Euclidean realization* or *configuration* $G(p)$ of $(G, \bar{l})$ is an assignment of points $p : V \to \mathbb{R}^2$ to the vertices of G satisfying the bar lengths in $\bar{l}$, modulo Euclidean motions. If the linkage is *flexible* (i.e., the space of realizations of a linkage is infinite), but the addition of a single bar causes the linkage to become *minimally rigid* (i.e. only finitely many realizations), then the linkage is called a *mechanism* with *1-degree-of-freedom (1-dof)*.

Describing and analyzing realization spaces of 1-dof linkages in 2D is a difficult problem with a long history. In fact, even for rigid linkages, the number of realizations can be exponential in the number of vertices and not easy to estimate [Borcea and Streinu 2004]. For flexible linkages, a well-known early result [Kempe 1875] shows that an arbitrary algebraic curve can be traced by the motion of a linkage joint. One outstanding example is the Peaucellier-Lipkin linkage, which transforms planar rotary motion into straight-line motion [Kempe 1877]. Versions of the problem play an important role in Computer-Aided-Design (CAD), robotics and molecular geometry [Sacks and Joskow-

This research was supported in part by the research grant NSF CCF-1117695 and a research gift from SolidWorks.

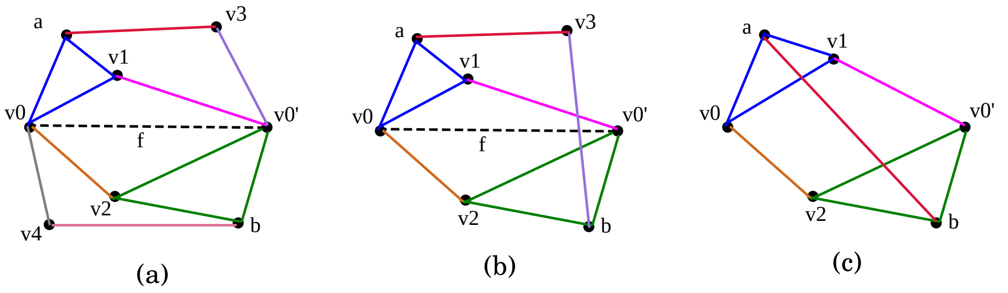

Fig. 1. (a) A 1-dof tree-decomposable linkage with low Cayley complexity (satisfies the Four-cycle Theorem). (b) A 1-dof tree-decomposable linkage without low Cayley complexity (does not satisfy the Four-cycle Theorem). (c) A linkage that is not 1-dof tree-decomposable.

icz 2010; Sitharam 2005; Ying and Iyengar 1995], but few results are known beyond individual or specific families of linkages.
Note: In the remainder of this article the term “linkage” refers specifically to “2D linkage”.

In this paper, we restrict ourselves to *1-dof tree-decomposable linkages*. The underlying graphs G of such linkages are obtained by dropping an edge from so-called *tree-decomposable graphs*. Tree-decomposable linkages are minimally rigid and well-studied, for example, in geometric constraint solving and CAD. For tree-decomposable linkages, if the bar lengths $\bar{l}$ are in $\mathbb{Q}$, the coordinate values of a realization are solutions to a triangularized quadratic system with coefficients in $\mathbb{Q}$ (i.e. the coordinates of the realization belong to an extension field over $\mathbb{Q}$ obtained by nested square-roots). Such values are called ruler-and-compass solvable or *quadratically-radically solvable (QRS)* values.

A graph G is *1-dof tree-decomposable* if there exists a non-edge f, called a *base non-edge* (there could be more than one), such that G has the following graph theoretical construction from f: at *Construction Step* k, the graph constructed so far, G_{k-1}, is extended by adding two new maximal tree-decomposable subgraphs, or *clusters* C_{k1} and C_{k2} sharing a vertex v_k. In addition, C_{k1} and C_{k2} each has exactly one shared vertex, u_k and w_k respectively, with G_{k-1}. Such a construction step is denoted $v_k \triangleleft (u_k, w_k)$. For example, the underlying graph of the linkage in Figure 1(b) has three construction steps from the non-edge f: $v_1 \triangleleft (v_0, v_0')$ which appends $\triangle v_0 v_1 a$ and the edge (v_0', v_1), $v_2 \triangleleft (v_0, v_0')$ which appends the edge (v_0, v_2) and $\triangle v_0' v_2 b$, and $v_3 \triangleleft (a, b)$ which appends two edges (v_3, a) and (v_3, b). Similarly, the underlying graph of the linkage in Figure 1(a) has four construction steps from the non-edge f. On the other hand, the underlying graph of the linkage in Figure 1(c) is not 1-dof tree-decomposable, as there does not exist a construction sequence from any non-edge.

A *realization type* for a 1-dof tree-decomposable linkage specifies a *local orientation* for the triple (v_k, u_k, w_k), for each construction step $v_k \triangleleft (u_k, w_k)$. For example, for the linkage in Figure 1(b), the construction step $v_3 \triangleleft (a, b)$ has a local orientation such that a, b, v_3 lie in counterclockwise order. Another possible orientation, with a, b, v_3 in clockwise order, can be obtained by flipping v_3 to the other side of (a, b). For a given realization type, a 1-dof tree-decomposable linkage $(G, \bar{l})$ has a simple ruler and compass realization $G(p)$ which parallels the graph theoretic construction. On the other hand, when a realization type is not specified, determining the existence of a realization is however NP-hard [Sacks and Joskowicz 2010].

We note here that each cluster has finitely many realizations according to its inner realization types. As no continuous motion is possible between those realizations, in this paper, we assume that the clusters are *globally rigid*, i.e. we fix the inner realization type of each cluster so that each cluster has a single realization. We also reduce clusters sharing only two vertices with the rest of the graph into edges.

1.1. Problems with the state of the art

There are numerous existing softwares suites dealing with 1-dof tree-decomposable linkages, such as Geometry Expressions [Todd 2007], SAM [Artas Engineering 2010], Phun [Algoryx 2013], Sketchpad [Key Curriculum 1995], Geogebra [GeoGebra Inc 2001], D-cubed [Siemens 1999], GIM [Petuya et al. 2014], CUIK [Porta et al. 2014], etc. They have the following major functionalities: (i) designing 1-dof tree-decomposable linkages for tracing out specific curves, especially by building new mechanisms based on a library of existing ones; (ii) accepting user-specified parameters, ranges and realization types to generate continuous motion of the linkages.

However, while some theoretical answers have been provided recently for the following issues (which will be described in the next subsection), until now there has been no theoretically guaranteed software implementation that addresses these issues.
(a) How do we canonically represent and visualize the connected components? In the above softwares, the realization space is typically represented as separate curves in 2D that are traced by each vertex of the linkage. In fact, a realization actually corresponds to a tuple of points, one on each of these curves. I.e., the realization space is bijectively represented by a curve in the full *ambient dimension* of $2|V| - 3$ after factoring out rigid transformations, where $|V|$ is the number of vertices in the linkage.
(b) How do we generate all connected components efficiently and how do we find a path of continuous motion between two arbitrary input realizations in the same connected component? In order to generate continuous motion in most of the existing software suites except for CUIK, the user must specify a range of a parameter containing the parameter value at the given realization. Then either a single connected component is generated for a subset of the specified range, or multiple segments of the realization space, under only the given realization type, are generated within the specified range. We discuss this issue in more detail in the following subsection.
(c) How do we find the “distance” between connected components? Until now, for two realizations in different connected components, there is no software that finds how “close” they can get towards each other by continuous motion, using a meaningful definition of “distance” between connected components.

In this paper, we provide a theoretically guaranteed software implementation to solve the above problems for a natural and commonly occurring subclass of 1-dof tree-decomposable linkages. We also show that for any class of linkages larger than this particular subclass, heuristic approaches adopted by existing software suites are necessarily intractable or incomplete.

1.2. Previous work on Cayley configuration spaces of 1-dof 2D linkages

The recent papers Sitharam and Wang [2014]; Sitharam et al. [2011a,b] introduced the use of *Cayley configuration space* to describe the realization space of a 1-dof, 2D linkage $(G, \bar{l})$. A Cayley configuration space is obtained by taking an *independent non-edge* f with $G \cup f$ being minimally rigid, and asking for all possible lengths that f can attain (i) over all the realizations of $(G, \bar{l})$; (ii) over all realizations of $(G, \bar{l})$ of a particular realization type. For (i) (resp. (ii)), each realizable length of f is called a (resp. *oriented*) *Cayley configuration*, and the set of all such configurations is called the (resp. *oriented*) *Cayley configuration space* of the linkage $(G, \bar{l})$ on f, parametrized by the length of f. The Cayley configuration space (non-oriented) is a set of disjoint closed intervals on the real line, called *non-oriented Cayley intervals*. As the Cayley configuration space is the union of all the different oriented Cayley configuration spaces,

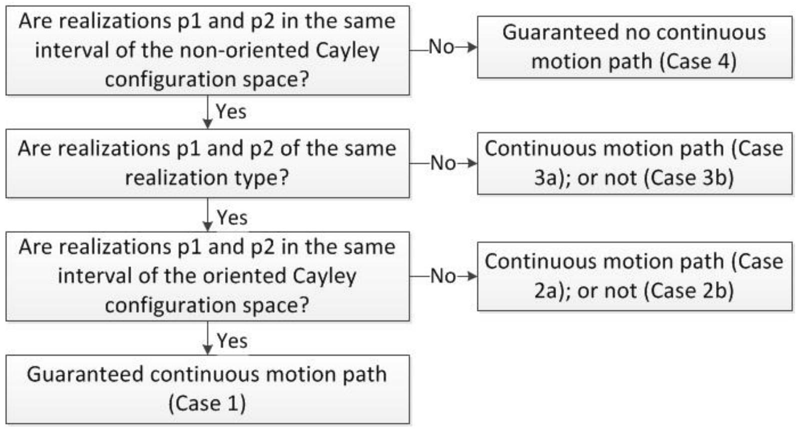
Are realizations p1 and p2 in the same interval of the non-oriented Cayley configuration space?
No
Guaranteed no continuous motion path (Case 4)
Yes
Are realizations p1 and p2 of the same realization type?
No
Continuous motion path (Case 3a); or not (Case 3b)
Yes
Are realizations p1 and p2 in the same interval of the oriented Cayley configuration space?
No
Continuous motion path (Case 2a); or not (Case 2b)
Yes
Guaranteed continuous motion path (Case 1)

Fig. 2. Complete case analysis of continuous motion paths between two realizations p_1 and p_2.

each non-oriented Cayley interval is a union of several *oriented Cayley intervals*. For a 1-dof tree-decomposable linkage $(G, \bar{l})$, any base non-edge f yielding a construction sequence can be taken as the Cayley non-edge parameter to give a reasonable Cayley configuration space.

An important complexity measure of the Cayley configuration space is its *Cayley complexity*, i.e. the algebraic descriptive complexity of the interval endpoints in the Cayley configuration space [Sitharam et al. 2011a, Section 1.2]. Specifically, if the endpoints are QRS values, the corresponding linkage is said to have *low Cayley complexity*. Low Cayley complexity is the minimal requirement on 2D linkages for tractable algebraic descriptive complexity of the configuration space. In other words, for general linkages without low Cayley complexity, even the algebraic description complexity of a single endpoint of the configuration space is NP-hard [Sitharam et al. 2011a]. We observe that many commonly studied mechanisms, including the Cardioid linkage, Limacon linkage and the Strandbeest [Jansen 2009], have low Cayley complexity.

In Sitharam et al. [2011a], algorithms are given for obtaining an (oriented) Cayley configuration space for generic 1-dof tree-decomposable linkages. Here a *generic linkage* means that no bar length is zero, all bars have distinct lengths and at most one pair of adjacent bars can be collinear in any realization. A realization of a generic linkage automatically satisfies the usual notion of genericity in the rigidity literature. It is also shown that for generic 1-dof tree-decomposable linkages with low Cayley complexity, the number of continuous motion paths between two realizations is *at most two*, and can be directly obtained from the oriented Cayley configuration spaces with complexity linear in a natural, discrete measure of the length of the path [Sitharam et al. 2011a, Theorem 3].

Sitharam et al. [2011b] gives the following theorem for recognizing low Cayley complexity linkages:

THEOREM 1.1 (FOUR-CYCLE THEOREM, [SITHARAM ET AL. 2011B, THEOREM 2]). *A 1-dof tree-decomposable linkage has low Cayley complexity if and only if every construction step of the underlying graph is based an adjacent pair of clusters, from a* four-cycle *of clusters in the previously constructed graph. This yields an algorithm to recognize 1-dof tree-decomposable linkages with low Cayley complexity, with time complexity quadratic in the number of construction steps of the underlying graph.*

For example, the linkage in Figure 1(b) does not have low Cayley complexity, since the construction step $v_3 \lhd (a, b)$ violates the four-cycle theorem: although $v_3 \lhd (a, b)$ is based on the four-cycle of the four clusters $\triangle v_0v_1a$, (v_1, v_0'), $\triangle v_0'v_2b$, (v_0, v_2), the base clusters $\triangle v_0v_1a$ and $\triangle v_0'v_2b$ containing a and b are not adjacent in the four-cycle. On the other hand, the linkage in Figure 1(b) satisfies the four-cycle theorem and has low Cayley complexity.

While judiciously chosen Cayley parameters shed light on many aspects of 1-dof tree-decomposable linkages, one persistent problem has been that a non-oriented Cayley interval, being a union of multiple oriented Cayley intervals, could correspond to multiple connected components of the realization space. On the other hand, although an oriented Cayley interval corresponds to a unique connected component, the mapping is not bijective, since that same connected component could contain more than one oriented interval [Sitharam and Wang 2014, Section 1.2]. Figure 2 summarizes the different cases when determining existence of a continuous motion path between two realizations. There are two cases (2 and 3) where there may or may not exist a continuous motion path (a and b). Previous software mentioned in Section 1.1, except for CUIK, generate continuous motion within a specified Cayley interval, or multiple segments of continuous motion, each corresponding to a different oriented Cayley interval with the same realization type. Thus they cannot consistently distinguish Case 2a from Case 2b, or Case 3a from Case 3b. The CUIK software deals with general mechanisms and provides comprehensive functionalities in continuous motion and connected components generation. However, the derivative tracing numerical methods used by CUIK must necessarily have non-polynomial complexity (unless P $=$ NP), as opposed to the linear complexity algorithm [Sitharam et al. 2011a, Theorem 3] for linkages of low Cayley complexity.

Sitharam and Wang [2014]; Sitharam et al. [2011a] give a bijective representation to represent the realization space, by characterizing a *canonical* method of picking a minimal set of non-edges, called a *complete Cayley vector*, such that adding those non-edges as bars results in global rigidity. The distance vectors on these non-edges (called *complete Cayley distance vectors*) for each realization of the realization space form a *canonical Cayley curve*, which bijectively represents the realization space. In addition, the *Cayley distance* between two Cartesian realizations is defined to be the Euclidean distance between their complete Cayley distance vectors, and the *Cayley distance* between two connected components is defined to be the minimum Cayley distance taken over all pair of realizations, one from each component.

In this paper, we present CayMos, a new system implementing these existing theoretical results for configuration spaces analysis of 1-dof tree-decomposable linkages with low Cayley complexity. Practical uses of CayMos include the following. (1) A substantial class of under-constrained systems occurring in mechanical CAD can be completed appropriately into well-constrained systems, by providing the user with a comprehensive set of values (the Cayley configuration space) for additional constraints, that guarantee the existence of a solution for the augmented system. (2) A large class of 3D kinematic mechanisms are “parallel” 2D mechanisms based on common linkages such as the Strandbeest [Jansen 2009]. These mechanisms are used in robotics, self-deploying structures etc. CayMos provides efficient and comprehensive analysis of their motions: connected components and continuous motion paths.

In Section 2, we list in detail the contributions of this paper in the form of CayMos functionalities. We briefly describe the algorithms implemented in CayMos in Section 3.

2. CONTRIBUTIONS: CAYMOS FUNCTIONALITY

Based on the theoretical contributions of Sitharam and Wang [2014]; Sitharam et al. [2011a,b], we present software package, CayMos [Wang and Sitharam 2014a] (source

code and user manual available at http://calgo.acm.org/), which implements efficient algorithmic solutions for the following:
(1) Determining low Cayley complexity and generating Cayley configuration spaces for a given linkage, implementing [Sitharam et al. 2011a, Section B] and [Sitharam et al. 2011b, Theorem 2].
(2) Meaningfully visualizing the connected components of the realization space as projection of the canonical Cayley curves, implementing [Sitharam and Wang 2014, Theorem 3], addressing Issue (a) from Section 1.1.
(3) Generating the connected components of the realization space, and finding a continuous motion path between two given realizations, implementing [Sitharam et al. 2011a, Theorem 3] and [Sitharam and Wang 2014, Theorem 5(i)], addressing Issue (b) from Section 1.1.
(4) Finding the realizations representing the shortest Cayley distance between two different connected components of the realization space, implementing [Sitharam and Wang 2014, Theorem 5(ii)], addressing Issue (c) from Section 1.1.
In the following we will briefly introduce the functionalities provided by `CayMos`.

Screen-shots and movies generated by `CayMos` may be found in Sitharam and Wang [2014]; Sitharam et al. [2011a].

2.1. Determining low Cayley complexity and generating Cayley configuration spaces

As Contribution (1), the user can generate the Cayley configuration spaces for 1-dof tree-decomposable linkages with low Cayley complexity. The following functionalities are provided:
(i) Determining whether the given linkage is 1-dof tree-decomposable with low Cayley complexity.
(ii) Generating both the oriented and non-oriented Cayley configuration space(s) for a linkage with low Cayley complexity. The user can change the length of the chosen base non-edge to see corresponding realizations, as well as specify the realization type.
See Figure 1 in the user manual [Wang and Sitharam 2014b] for an annotated illustration of the user interface.

2.2. Visualizing the connected components and finding continuous motion paths of the realization space

As Contribution (2) and (3), the user can generate and visualize the connected components of the realization space, as well as find a continuous motion path between two realizations. The following functionalities are provided:
(i) Showing the non-edges in the complete Cayley vector of the linkage, as well as displaying the complete Cayley distance vector for the current realization.
(ii) Generating all the connected components of the realization space, as well as finding a continuous motion path (if one exists) between two realizations specified by the user.
(iii) Visualizing the connected component by showing the corresponding canonical Cayley curve, projected on three non-edges picked by the user from the complete Cayley vector.
(iv) Showing the curves traced out by vertices of the linkage in continuous motion.
See Figure 3 in the user manual [Wang and Sitharam 2014b] for an annotated screen-shot highlighting the functionalities described in (i), (ii) and (iii) above.

2.3. Cayley distance between connected components

As Contribution (4), when the user tries to find a continuous motion path between two realizations in different connected components, `CayMos` will find the two nearest realizations of these two components.

3. ALGORITHMS IMPLEMENTED IN CAYMOS

Several algorithms from our previous papers Sitharam and Wang [2014]; Sitharam et al. [2011a,b] are implemented in `CayMos`. Here we give a brief description of these algorithms.

Four-cycle algorithm: to determine whether a given 1-dof tree-decomposable linkage has low Cayley complexity, we implement the *Four-cycle algorithm* [Sitharam et al. 2011b, Theorem 2], which follows the construction of the linkage, and tests whether each construction step satisfies the condition given by the Four-cycle Theorem (see Section 1.2). For each of the $O(|V|)$ construction steps, we need to check $O(|V|)$ candidate base pairs of clusters, so the overall time complexity is $O(|V|^2)$,

ELR algorithm: to find the Cayley configuration space of a 1-dof tree-decomposable linkage with low Cayley complexity, we implement the *ELR (extreme linkage realization) algorithm* [Sitharam et al. 2011a, Section B]. The algorithm works by realizing all the *extreme linkages* consistent with each realization type, and finding the intersection of all the candidate intervals of the base non-edge. The overall time complexity is exponential in $|V|$. As the problem of computing the Cayley configuration space is NP-hard [Sitharam et al. 2011a, Observation 2], an improvement over exponential time complexity is unexpected unless P = NP. Sitharam et al. [2011a] also contains another algorithm called *QIM (quadrilateral interval mapping)*, which also computes the Cayley configuration space, but only applies to a special subclass of linkages called *1-path* 1-dof tree-decomposable linkages. The implementation of QIM algorithm is not included in `CayMos`.

Continuous motion algorithm: to find continuous motion paths and connected component(s) of the realization space, we implement the *Continuous motion algorithm* [Sitharam et al. 2011a, Theorem 3], [Sitharam and Wang 2014, Theorem 5(i)]. The algorithm works by traversing the intervals in the oriented Cayley configuration space via the common interval endpoints. As mentioned in Section 1.2, the time complexity is linear in the number of oriented Cayley configuration space endpoints along the continuous motion path or connected component.

Closest pair algorithm: to find the pair of "closest" realizations between two different connected components, we implement the *Closest pair algorithm* [Sitharam and Wang 2014, Theorem 5(ii)], which samples both components and returns the pair of realizations with the smallest Cayley distance, which is computed using the complete Cayley distance vector (see Section 1.2).

The backend of CayMos consistes of two parts: a) 1-dof tree decomposable linkages and Cayley configuration spaces, and b) continuous motion generation and representation. The major classes and methods within each of these areas are described in further detail in the user manual that accompanies the software component [Wang and Sitharam 2014b].

Note: The source code and user manual of `CayMos` are available at http://calgo.acm.org/.

REFERENCES

Algoryx. 2013. Algodoo (Phun): 2D Physics sandbox. http://www.algodoo.com. (2013).

Artas Engineering. 2010. SAM: The ultimate mechanism designer. http://www.artas.nl/. (2010).

Ciprian Borcea and Ileana Streinu. 2004. The Number of Embeddings of Minimally Rigid Graphs. *Discrete and Computational Geometry* 31 (2004), 287–303. Issue 2. 10.1007/s00454-003-2902-0.

GeoGebra Inc. 2001. Geogebra. http://www.geogebra.org/. (2001).

Theo Jansen. 2009. Strandbeest. http://www.strandbeest.com/. (2009).

Alfred Bray Kempe. 1875. On a general method of describing plane curves of the nth degree by linkwork. *Proceedings of the London Mathematical Society* s1–7, 1 (1875), 213–216.

Alfred Bray Kempe. 1877. *How to draw a straight line: a lecture on linkages*. Macmillan and Co.

Key Curriculum. 1995. The geometers sketchpad. http://www.keycurriculum.com/. (1995).

Víctor Petuya, Erik Macho, Oscar Altuzarra, Charles Pinto, and Alfonso Hernandez. 2014. Educational software tools for the kinematic analysis of mechanisms. *Computer Applications in Engineering Education* 22, 1 (2014), 72–86. GIM.

Josep M Porta, Lluís Ros, Oriol Bohigas, Montserrat Manubens, Carlos Rosales, and Léonard Jaillet. 2014. An Open-Source Toolbox for Motion Analysis of Closed-Chain Mechanisms. In *Computational Kinematics*. Springer, 147–154. CUIK.

E. Sacks and L. Joskowicz. 2010. *The configuration space method for kinematic design of mechanisms*. The MIT Press.

Siemens. 1999. D-Cubed. http://www.plm.automation.siemens.com/en_us/products/open/d-cubed/. (1999).

Meera Sitharam. 2005. Combinatorial Approaches to Geometric Constraint Solving: Problems, Progress, and Directions. *In: DIMACS: Series in Discrete Mathematics and Theoretical Computer Science* 67 (2005), 117.

Meera Sitharam and Menghan Wang. 2014. How the beast really moves: Cayley analysis of mechanism realization spaces using CayMos. *Computer-Aided Design* 46, 0 (2014), 205 – 210. 2013 SIAM Conference on Geometric and Physical Modeling.

Meera Sitharam, Menghan Wang, and Heping Gao. 2011a. Cayley Configuration Spaces of 1-dof Tree-decomposable Linkages, Part I: Extreme points, continuous motion paths and minimal representations. (2011). *Under review. arXiv:1112.6008*.

Meera Sitharam, Menghan Wang, and Heping Gao. 2011b. Cayley Configuration Spaces of 1-dof Tree-decomposable Linkages, Part II: Combinatorial Characterization of Complexity. (2011). *Under review. arXiv:1112.6009*.

Philip Todd. 2007. Geometry expressions: A constraint based interactive symbolic geometry system. In *Automated Deduction in Geometry*. Springer, 189–202.

Menghan Wang and Meera Sitharam. 2014a. CayMos software. (2014). Web accessible at: http://www.cise.ufl.edu/~menghan/caymos/, software available at: http://calgo.acm.org/ and http://code.google.com/p/caymos/.

Menghan Wang and Meera Sitharam. 2014b. CayMos user manual: user interface, functionalities and high level sourcecode. (2014).

Zhiyuan Ying and S. Sitharama Iyengar. 1995. Robot reachability problem: A nonlinear optimization approach. *Journal of Intelligent and Robotic Systems* 12 (1995), 87–100. Issue 1. http://dx.doi.org/10.1007/BF01258308

Received Month Year; revised Month Year; accepted Month Year