Title:
Graph Constructive Geometric Constraint Solving: Challenges and Machine Learning

Authors:
Ioannis Fudos, fudos@uoi.gr, University of Ioannina
Vasiliki Stamati, vstamati@uoi.gr, University of Ioannina

Keywords:
Geometric constraint solving, CAD, root navigation, animation, robotics, rigidity, machine learning

DOI: 10.14733/cadconfP.2024.288-294

Introduction:
Computer-aided design (CAD) has played a crucial role in various engineering applications, such as modeling, analysis, optimization, manufacturing, and maintenance. The concept of CAD dates back to the 1960s, but it was the introduction of parametric CAD modelers in the late 1980s that led to widespread recognition within the user and engineering community.

Parametric CAD allows for geometry reuse, enabling users to create a family of geometry variants by adjusting embedded parameters in the model. Parametric modeling software is used in various industries and helps in, i.e., creating detailed 3D models and generating manufacturing-ready designs. The associativity of parametric CAD techniques allows for automatic change propagation when local parametric changes are made, achieved by a system of geometric constraints and constraint solving.

Feature-based CAD is a design methodology that uses predefined shapes or features to create complex 3D models. These features can be simple shapes (e.g., lines, arcs, and circles) or more complex ones (e.g., extrusions, chamfers, and fillets). Feature-based CAD allows designers to create models more quickly and easily than with traditional CAD methods. Features can be easily modified and updated, making it easier to make changes to a design. Feature-based CAD is also more parametric than traditional CAD; the model is easily updated if the underlying parameters are changed. However, since it involves design parts arbitrarily interconnected through geometric constraints, it needs to be supported by a powerful, robust, and efficient geometric constraint-solving engine.

Geometric constraint solving (GCS) in CAD ensures that the elements of a design maintain their intended relationships and properties, thereby facilitating the creation of accurate, precise, and consistent parts. Also, GCS enables the automation of design modifications, leading to a more efficient use of time and resources. Geometric constraints help enforce standardization in design practices, ensuring that designs adhere to specific rules and guidelines. GCS involves the use of a geometric constraint system consisting of geometric entities and constraints. The process of solving systems of geometric constraints involves translating constraints into algebraic equations and finding valid instantiations of geometric entities that satisfy all the constraints. Challenges in GCS include characterizing constraint states, detecting ill-constrained parts, and decomposing geometric constraint systems into smaller subsystems for efficient solving. Research efforts have focused on developing effective criteria for detection and decomposition algorithms for general constraint systems. Various techniques have been employed for GCS, including rule-based constraint solving, nonlinear optimization using homotopy, analysis of the graph of geometric relationships, and constructive constraint solving.

In this paper, we explore graph constructive constraint solving by identifying the major challenges and suggesting plausible research directions that leverage machine learning (ML) and theoretical breakthroughs. More specifically, we investigate the use of graph neural networks (GNNs) to encode a GCS problem as a matter of relationships among elementary rigid bodies. Each node in the graph

represents a rigid body, and methods are developed to detect whether a node belongs in a minimal constructible rigid body cluster. This approach provides a sequence of rigid body placements that will derive a solution for the GCS problem, making it explainable, user-intuitive, and aligned with design intent. Additionally, we address the root selection or root navigation problem by employing a neural network to select the appropriate root (solution) for an elementary construction, trained on examples drawn from standard design libraries. We will also delve into the theoretical foundations for GCS and attempt to provide novel, sufficient, and necessary conditions for the well-constrainedness of 2D and 3D GCS problems. The deep graph constructive GCS approach suggested may have applications in various fields besides CAD, including kinematics analysis for robotics, determining the degrees of freedom (dof) in animation sequences, molecular biology for determining feasible placements of molecules in 3D, and other applications that involve geometric constraints in both 2D and 3D spaces.

State of the Art:

Numerical methods offer powerful tools for iteratively solving large systems of equations. Typically, ensuring convergence requires providing a reasonably accurate initial approximation of the desired solution. Hence, if the initial point is derived from the user-defined sketch, it should closely align with the intended solution. The widely adopted Newton-Raphson stands out as a local method used in solvers described by various sources (e.g. [7]). Homotopy or continuation (i.e. [10]) represents a family of global methods that can determine all solutions, but exhibit lower efficiency compared to local ones.

Symbolic algebraic techniques calculate a Grobner basis for a given system of equations, with notable algorithms (e.g. [12]). Analytical methods focusing on system structures assess whether a system is under-constrained, well-constrained, or over-constrained. These approaches can be extended to decompose systems into minimal graphs, independently solvable, serving as a pre-processing step to reduce the variables and equations to be solved simultaneously. These methods can be applied to almost any system of equations but are slow, cannot capture over and under-constrained systems, and do not provide an intuitive placement of geometric elements according to geometric constraints since they do not exploit the geometric features of the problem.

In logic-based approaches (aka rule-based), the problem undergoes translation into a set of assertions and axioms that characterize both the constraints and the geometric objects. Researchers (e.g. [2]) employ first-order logic to extract geometric information using a set of axioms derived from Hilbert's geometry. These methods essentially generate geometric loci representing the locations where the elements must be positioned. These methods are nice for prototyping and testing new repertoires of constraints, geometries and rules but cannot be used in real world systems since they use exhaustive matching techniques.

The constructive approach entails deriving solutions to geometric constraint problems through a sequence of elementary construction steps. Each step follows rules from a predetermined set of operations, positioning specific geometric elements. This method effectively preserves the inherent geometric meaning of each operation in the solution. Depending on the analytical technique applied, constructive approaches can be categorized as top-down or bottom-up[9]. [5],[6] use clusters as rigid body sets of geometries and constraints, employing a rule that merges 3 clusters that, pairwise, share one element. This provides a unified approach to treat geometric constraint systems as rigid bodies where constraints are reduced to incidence constraints (rigid bodies that share a geometry). While this provides a fast and intuitive approach to the GCS problem, there are 3 major issues that should be addressed: (i) extend the repertoire of rigid body patterns that can be merged/placed, (ii) extend the repertoire of the elementary geometric object to include higher dimension geometry (e.g. circle with variable radius) and (iii) extend the algorithm to 3D.

Degrees of Freedom (dof) Analysis involves assigning dof to geometric elements by labeling the vertices of the constraint graph. Each graph edge is tagged with the number of dof eliminated by the associated constraint. The method then resolves the problem by examining the resulting labeled graph. [5] uses reduction systems to detect triangle decomposable well-constrained systems. [10] breaks down the labeled graph into minimal connected components called balanced sets. [8] presents a flow-based method for decomposing graphs of geometric constraint problems by iterating to decompose the underlying algebraic system into minimal dense subgraphs. While this method fully generalizes dof calculations and prior flow-based approaches, the resulting decomposition may lack geometric sense, as

minimal dense subgraphs can exhibit arbitrary complexity beyond classical geometric construction problems. In 2D, the Laman theorem provides a sufficient and necessary condition for characterizing the well-constrainedness of a GCS if the geometries have 2 dof and the geometric constraints reserve 1 dof. If we include geometries with 3 dof the Laman condition is not sufficient anymore. A necessary and sufficient condition for rigidity in 3D has been presented in [11] for an extended set of geometries and constraints. Note that this characterization does not capture the cases explained in Fig. 2 and 3. The condition holds for both cases, but rigidity is not guaranteed since they involve point coincidences between rigid bodies directly or indirectly. Leveraging a similar analysis on the system of equations that describe a GCS [4] have presented the witness configuration method which also has many limitations and cannot be applied to solve large GCS problems.

Challenges and Research Directions:

Some major challenges in CAD, graphics and GCS are the following:

(C1) Making CAD and animation tools easier to use and more productive involves improving the user interface, streamlining workflows, and enhancing the functionality of the software. This could include features such as intuitive controls, real-time feedback, and simplified processes for creating and editing designs or animations. The goal is to make the tools more accessible and efficient for users, ultimately increasing their productivity.

(C2) Advancing theoretical foundations for GCS entails further developing the mathematical principles and algorithms that underpin the process of solving geometric constraints. This could involve research into new mathematical models, optimization techniques, and computational methods to improve the accuracy, efficiency, and robustness of GCS.

(C3) Building powerful and explainable GCS engines involves creating software systems that are capable of accurately and reliably solving complex geometric constraints while providing clear explanations for their solutions. This could include developing algorithms that can handle a wide range of geometric scenarios, providing detailed reasoning for each step of the solving process, and ensuring transparency in the decision-making of the solving engine.

We propose the following directions of research on graph constructive GCS and explain how they affect the above challenges:

(O1) Enhance the theoretical foundations by understanding the structural properties that ensure well-constrainedness (addressing challenges C1, C2) and proposing methods for effectively solving consistently over and under-constrained systems (addressing C3).

(O2) Develop an efficient, versatile technique for deriving a construction sequence for placing geometric elements that satisfy the imposed geometric constraints (addressing C1). The construction sequence should be easy to explain and align with the design intent (addressing C3).

(O3) Introduce a user-friendly root selection process, enabling users, designers, and artists to navigate through the solution space. The process will be powered by a deep learning method that automatically recommends the most plausible root based on supervised learning from examples drawn from CAD libraries (addressing C1, C3).

(O4) Utilize advanced deep graph constructive GCS techniques for (i) editing parametric and feature-based CAD models, (ii) generating novel feature-based CAD models quickly, with minimal user intervention, (iii) constraining and animating articulated characters, and (iv) animating machine parts and robotic setups by deriving the dof of the geometric elements (all addressing C1).

Machine Learning for Graph Constructive GCS:

Definition 1: Let *P* be a geometric constraint system *P = (U, F)* where *U* are the geometric elements and *F* the constraints imposed on them. The constraint graph *G = (V, E) of P* is a labeled undirected graph whose vertices are the geometric objects in *U*, each labeled with its dof. There is an edge *(u, v) in E* if there is a constraint between the geometric objects *ui* and *vi,* corresponding to *u* and *v* respectively. The edge is labeled by the number of independent equations corresponding to the constraint between *ui* and *vi.*

Below we will omit the edge labels when they represent 1 dof and the labels of the vertices when they are 2 in 2D and 3 in 3D. Every geometric element has a certain number of dof, i.e., a line in 3D has 4 dof. Each constraint reserves 1 or more dof. For example, in 3D a coincidence constraint “point on line”

reserves 1 dof, whereas a coincidence constraint "point on point" reserves 3 dof. The deficit of a constraint graph is the sum of dof minus the reserved by constraints dof.

Fig. 1 shows a geometric constraint problem in 2D which cannot be solved by merely placing objects sequentially. If we create the 3 rigid bodies as shown in the middle (from A, I we place H, then from I,H we place G etc.) we end up with the situation shown on the right. There, we must place points A,D and G and then rotate and translate the 3 rigid bodies to fit on the 3 points. This implies that the geometric constraint system can be solved by successive reductions of well constrained rigid body configurations. Rigid bodies share geometric elements. Shared geometric elements can be represented as coincidence constraints that eliminate the same number of dof as those of the shared geometry. For example, each shared point in 2D eliminates 2 dof. This is a powerful and intuitive GCS method that can be applied in 2D or 3D. Currently there are certain limitations that we will attempt to overcome in this project.

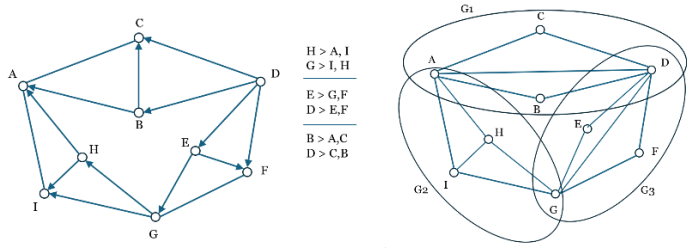

Fig. 1: (Left) a GCS problem (A through I are points in 2D, edges represent distances); the constraint graph is the same undirected graph. (Middle) three plausible construction sequences for placing the 3 groups of points; every group of points represents a rigid body. (Right) 3 rigid bodies sharing a point pairwise.

Theoretical foundations: extending Laman's theorem

Laman's theorem determines whether a geometric constraint problem is well constrained, meaning that the problem has a finite number of solution instances for non-degenerate configurations.

Laman's theorem: Let $G = (V, E)$ be a connected, undirected graph whose vertices represent points in *2D* and edges represent distances between points. G is a well-constrained constraint graph of a geometric constraint problem iff the deficit of G is 3 and, for every subset $U \subset V$, the induced subgraph (U, F) has a deficit of no less than 3.

Laman's theorem holds even if we extend the repertoire of geometries to any geometry having 2 dof and the constraints to virtually any constraint. But, if we extend the set of geometries to include e.g., variable radius circles (which acquires 3 dof), then the Laman condition is no longer sufficient. Fig. 2 depicts a Laman graph in 2D that is not rigid (V_1: variable radius circle, V_2, V_3, V_2', V_3': points, V_4, V_4': lines, e_1, e_2, e_1', e_2': points, e_3, e_4, e_3', e_4' : on constraints, e_5, e_5': distances, e_6, e_6' :distances from circle (circumference).

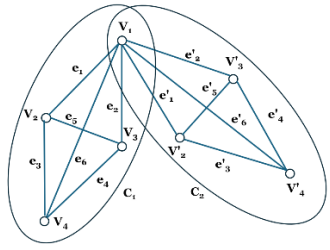

Fig. 2: A constraint graph in 2D. It consists of 2 rigid bodies that share a 3 dof geometry. Although the Laman condition is valid, the graph is not well-constrained.

Graph analysis for spatial constraint problems is not nearly as mature as the planar case. In 3D, the Laman condition is not sufficient. Fig. 3 illustrates 2 hexahedra sharing 2 vertices. If the length of the edges is given, the GCS that arises is also known as the double banana problem. The graph is a Laman graph, but the problem corresponds to 2 rigid bodies (each hexahedron is a rigid body) sharing 2 vertices and is thus nonrigid in the sense that the 2 rigid bodies are free to rotate around the axis defined by the 2 shared points. The problem is also clearly over-constrained since the distance of the 2 shared geometries can be derived independently by each of the 2 rigid bodies.

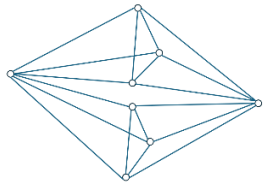

Fig. 3: A constraint graph in 3D. Also known as the double banana problem, since it reduces to 2 rigid bodies that share 2 points. Although the Laman condition is valid, the graph is not well-constrained.

Therefore, we propose the following conjecture that extends Laman's theorem:
Conjecture 1: A geometric constraint problem is well constrained iff the reduction sequence does not derive 2 rigid bodies that share geometries with a total dof equal to the dof of a rigid body in the corresponding space (6 for 3D, 3 for 2D).

If such rigid bodies (that share pairwise 3 in 2D or 6 in 3D dof) appear, we can safely conclude that the problem is not well constrained. It remains to determine whether the reverse holds. We shall study the correctness of Conjecture 1 and provide an efficient algorithm using deep graph reduction to determine the well-constrainedness of a geometric constraint scheme.

Extending the repertoire of reductions: Deep graph reduction
The simplest reduction is the triangle reduction (see Fig. 1 (right)), which has been studied extensively. Fig. 4 (left) illustrates 2 more minimal cases of well-constrained configurations of graph constraint problems.

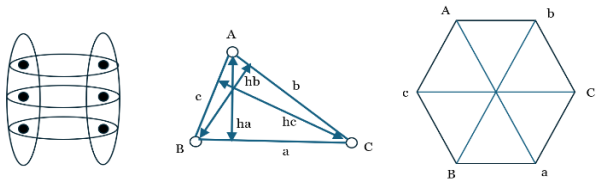

Fig. 4: (Left) A minimal well-constrained configuration that consists of 5 rigid bodies in 2D and contains no triangles. (Middle) A constraint problem in 2D: Derive the geometry of a triangle given its 3 altitudes and (right) the corresponding constraint graph that contains no triangle (each edge is a rigid body).

We propose to systematically generate all such cases of minimal well-constrained rigid body graphs along with placement of geometries. Then, one can introduce ML approaches for detecting minimal well-constrained rigid body graphs based on graph density with minCutPool layers with spectral clustering in GNNs [1] (unsupervised learning) or for detecting certain graph patterns with neighborhood kernels in GNNs [3] (supervised learning). To increase the scope of our approach, we may (i) include composite geometric objects by transforming the composite object to a set of direct and indirect constraints and elementary geometries and (ii) extend the repertoire of elementary geometries (e.g., circle of variable radii).

The root selection problem: AI-assisted root navigation

A well-constrained geometric constraint system for which we have derived a construction sequence based on graph analysis with m construction steps and an average of k discrete per construction steps will have $O(k^m)$ discrete solutions. Fig. 5 illustrates a geometric constraint problem with 3 discrete real solutions (up to rotation and translation). Even the more generic problem of computing a real solution for a solvable well-constrained problem was shown to be NP-hard in a strong combinatorial sense[6].

We propose developing a user-friendly and powerful tool for the root navigation problem employing deep learning by using a hybrid GNN-LSTM with attention architecture for choosing the appropriate solution based on the construction sequence. There will be a training phase with hundreds of construction sequences from design libraries.

Applications in CAD, animation and robotics

GCS can be employed for placing geometric objects in CAD models. We investigate cases with simple geometries(points, lines, planes, conics, spheres), composite parts, and surface/curve patches. We also suggest investigating encodings of graph construction sequences for generating novel, meaningful construction sequences (for CAD). Also, the proposed tool suite can be used for Dof analysis for skeletons used in the animation of articulated characters and for determining forward kinematics in robotics.

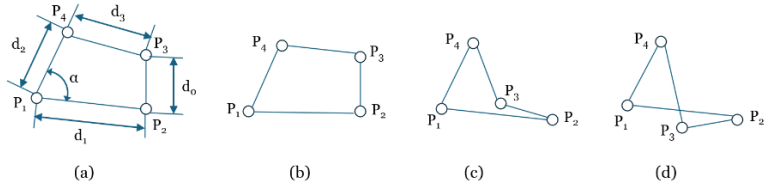

Fig. 5: (a) A geometric constraint problem consisting of 4 points, 4 straight segments, 4 point-point distances, and an angle. (b),(d) 2 different solution instances to constraint problem (a). By changing the value of d_3 one may also get solution (c) which yields a non-intersecting non-convex closed polyline. In most cases, (b) captures design intent since it is the only non-intersecting convex closed polyline.

Conclusions

It is a fact that Geometric Constraint Solving is a vital element in CAD, given that it ensures the creation of accurate, precise, and consistent parts. By combining traditional methods of GCS with graph theory and ML approaches, we can rise to the research challenges presented in this paper. The problem of well-constraindness of 2D and 3D GCS problems is addressed, based on the theoretical foundations for GCS, along with the matter of AI-assisted root selection for graph constructive constraint solving.

Ioannis Fudos, https://orcid.org/0000-0002-4137-0986
Vasiliki Stamati, https://orcid.org/0000-0002-4225-3685

References:

[1] Bianchi, F.; Grattarola, D.; Alippi, C.: Spectral Clustering with Graph Neural Networks for Graph Pooling, Proc. of the 37th international conference on Machine learning, 2729-2738, ACM 2020.
[2] Brüderlin, B.: Using geometric rewrite rules for solving geometric problems symbolically, In Theoretical Computer Science 116, 1993, 291–303. Elsevier Science Publishers B.V.
[3] Feng, A.; You, C.; Wang, S.; Tassiulas, L.: KerGNNs: Interpretable Graph Neural Networks with Graph Kernels, Proc of the AAAI Conference on Artificial Intelligence 36(6), 2022, 6614–6622, https://doi.org/10.1609/aaai.v36i6.20615.
[4] Foufou, S.; Michelucci, D.: Interrogating witnesses for geometric constraint solving, Information and Computation, 216, 2012, 24–38, https://doi.org/10.1016/j.ic.2011.09.006.
[5] Fudos, I.; Hoffmann, C.M.: Correctness proof of a geometric constraint solver, Intl J Computational Geometry and Applications, 6(4), 1996, 405-420, https://doi.org/10.1142/S0218195996000253.

[6] Fudos, I.; Hoffmann, C.M.: A graph-constructive approach to solving systems of geometric constraints, ACM Trans. On Graphics, 16, 1997, 179-216, https://doi.org/10.1145/248210.248223.
[7] Heydon, A.; Nelson, G.: The Juno-2 Constraint-Based Drawing Editor, Research Report 131a, Digital Systems Research Center, December 1994.
[8] Hoffman, C.M.; Lomonosov, A.; Sitharam, M.: Decomposition Plans for Geometric Constraint Systems, Part I: Performance Measures for CAD, J Symb Comp 31, 2001, 367–408, https://doi.org/10.1006/jsco.2000.0402.
[9] Fudos, I.; Hoffmann, C.M.; Juan-Arinyo, R.: Tree-decomposable and Underconstrained Geometric Constraint Problems, M. Sitharam, A. St. John, J. Sidman (Eds.), Handbook of Geometric Constraint Systems Principles, Chapter 6,2017, https://doi.org/10.1201/9781315121116.
[10] Lamure, H.; Michelucci, D.: Solving geometric constraints by homotopy, C. Hoffmann, J. Rossignac (Eds.),3rd Symp. Solid Modeling and Applications, ACM Press, Salt Lake City, 1995, 263–269, https://doi.org/10.1145/218013.218071.
[11] Lee-St.john, A.; Sidman, J.: Combinatorics and the rigidity of CAD systems, Computer-Aided Design, 45(2), 2013, 473–482, https://doi.org/10.1016/j.cad.2012.10.030.
[12] Wu, W.:Mechanical Theorem Proving in Geometries, Texts and Monographs in Symbolic Computations, Springer,Vienna,1994,https://doi.org/10.1007/978-3-7091-6639-0.