A Semidefinite Programming Approach to Tensegrity Theory and Realizability of Graphs

Anthony So

Department of Computer Science

Yinyu Ye

Department of Management Science and Engineering and

by courtesy, Electrical Engineering

Stanford University

Stanford, CA 94305, U.S.A

In SODA'06

http://www.stanford.edu/~yyye

Outlines

- Graph Realization Problem

Outlines

- Graph Realization Problem
- d-Realizable Graphs

Outlines

- Graph Realization Problem
- d-Realizable Graphs
- SDP Formulation

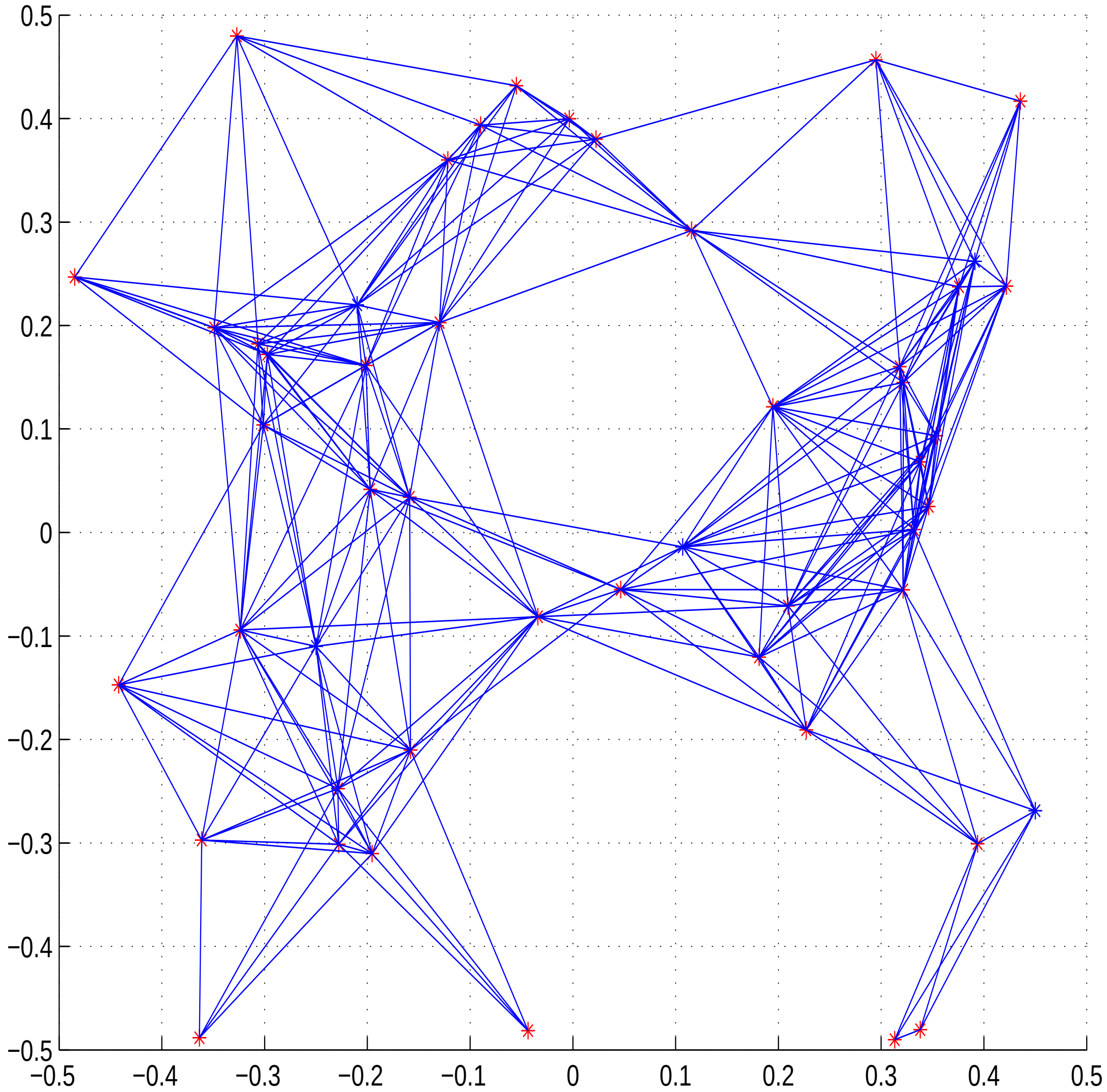
Outlines

- Graph Realization Problem
- d-Realizable Graphs
- SDP Formulation
- Realization Algorithm

The Graph Realization Problem

Given a graph $G = (V, E)$ and a set of non–negative edge weights $\{d_{ij} : (i, j) \in E\}$, and the goal is to compute a realization of G in the Euclidean space $\mathbf{R}^d$ for a given dimension d, i.e. to place the vertices of G in $\mathbf{R}^d$ such that the Euclidean distance between every pair of adjacent vertices v_i, v_j equals to the prescribed weight d_{ij}.

Figure 1: 50-node Graph Realization in 2D



Applications

- Global Position System (GPS)

Applications

- Global Position System (GPS)
- Sensor network localization

Applications

- Global Position System (GPS)
- Sensor network localization
- Molecular conformation

Applications

- Global Position System (GPS)
- Sensor network localization
- Molecular conformation
- Data dimension reduction

Applications

- Global Position System (GPS)
- Sensor network localization
- Molecular conformation
- Data dimension reduction
- Euclidean ball parking

Related Work

- Schoenberg and Young/Householder studied the case where all pairwise distances are given, which formed the basis of various multidimensional scaling algorithms.

Related Work

- Schoenberg and Young/Householder studied the case where all pairwise distances are given, which formed the basis of various multidimensional scaling algorithms.
- Barvinok and Alfakih/Wolkowicz used SDP models to show that the problem is solvable in polynomial time if the dimension of the realization is not restricted. Moreover, they have given bounds on the dimension needed to realize the given distances.

Related Work

- Schoenberg and Young/Householder studied the case where all pairwise distances are given, which formed the basis of various multidimensional scaling algorithms.
- Barvinok and Alfakih/Wolkowicz used SDP models to show that the problem is solvable in polynomial time if the dimension of the realization is not restricted. Moreover, they have given bounds on the dimension needed to realize the given distances.
- However, if we require the realization to be in $\mathbf{R}^d$ for some fixed d, then the problem becomes NP–complete (Aspnes, Goldenberg, and Yang 2004).

Related Work

- Schoenberg and Young/Householder studied the case where all pairwise distances are given, which formed the basis of various multidimensional scaling algorithms.
- Barvinok and Alfakih/Wolkowicz used SDP models to show that the problem is solvable in polynomial time if the dimension of the realization is not restricted. Moreover, they have given bounds on the dimension needed to realize the given distances.
- However, if we require the realization to be in $\mathbf{R}^d$ for some fixed d, then the problem becomes NP–complete (Aspnes, Goldenberg, and Yang 2004).
- Identify families of graph instances that admit polynomial time algorithms for computing a realization in the required dimension (Biswas, So, Toh, and Ye 2004-2005; SODA'05, ACM, IEEE).

d–Realizable Graphs

A graph is d–realizable if it can always be realized in $\mathbf{R}^d$ whenever it is realizable (the edge weights are Euclidean metric).

d–Realizable Graphs

A graph is d–realizable if it can always be realized in $\mathbf{R}^d$ whenever it is realizable (the edge weights are Euclidean metric).

- Connelly and Sloughter have recently given a complete characterization of the class of d–realizable graphs, where $d = 1, 2, 3$

d–Realizable Graphs

A graph is d–realizable if it can always be realized in $\mathbf{R}^d$ whenever it is realizable (the edge weights are Euclidean metric).

- Connelly and Sloughter have recently given a complete characterization of the class of d–realizable graphs, where $d = 1, 2, 3$
- It is trivial to find a realization of an 1–realizable graph, since a graph is 1–realizable iff it is a forest.

d–Realizable Graphs

A graph is d–realizable if it can always be realized in $\mathbf{R}^d$ whenever it is realizable (the edge weights are Euclidean metric).

- Connelly and Sloughter have recently given a complete characterization of the class of d–realizable graphs, where $d = 1, 2, 3$
- It is trivial to find a realization of an 1–realizable graph, since a graph is 1–realizable iff it is a forest.
- A polynomial time algorithm for realizing 2–realizable graphs exists: trilateralization.

d–Realizable Graphs

A graph is d–realizable if it can always be realized in $\mathbf{R}^d$ whenever it is realizable (the edge weights are Euclidean metric).

- Connelly and Sloughter have recently given a complete characterization of the class of d–realizable graphs, where $d = 1, 2, 3$
- It is trivial to find a realization of an 1–realizable graph, since a graph is 1–realizable iff it is a forest.
- A polynomial time algorithm for realizing 2–realizable graphs exists: trilateralization.
- Finding a corresponding algorithm for 3–realizable graphs is posed as an open question.

3–realizable graph I

A graph is 3–realizable iff it does not contain K_5 or $K_{2,2,2}$ as a minor (Connelly and Sloughter 2004).

Figure 2: K-5

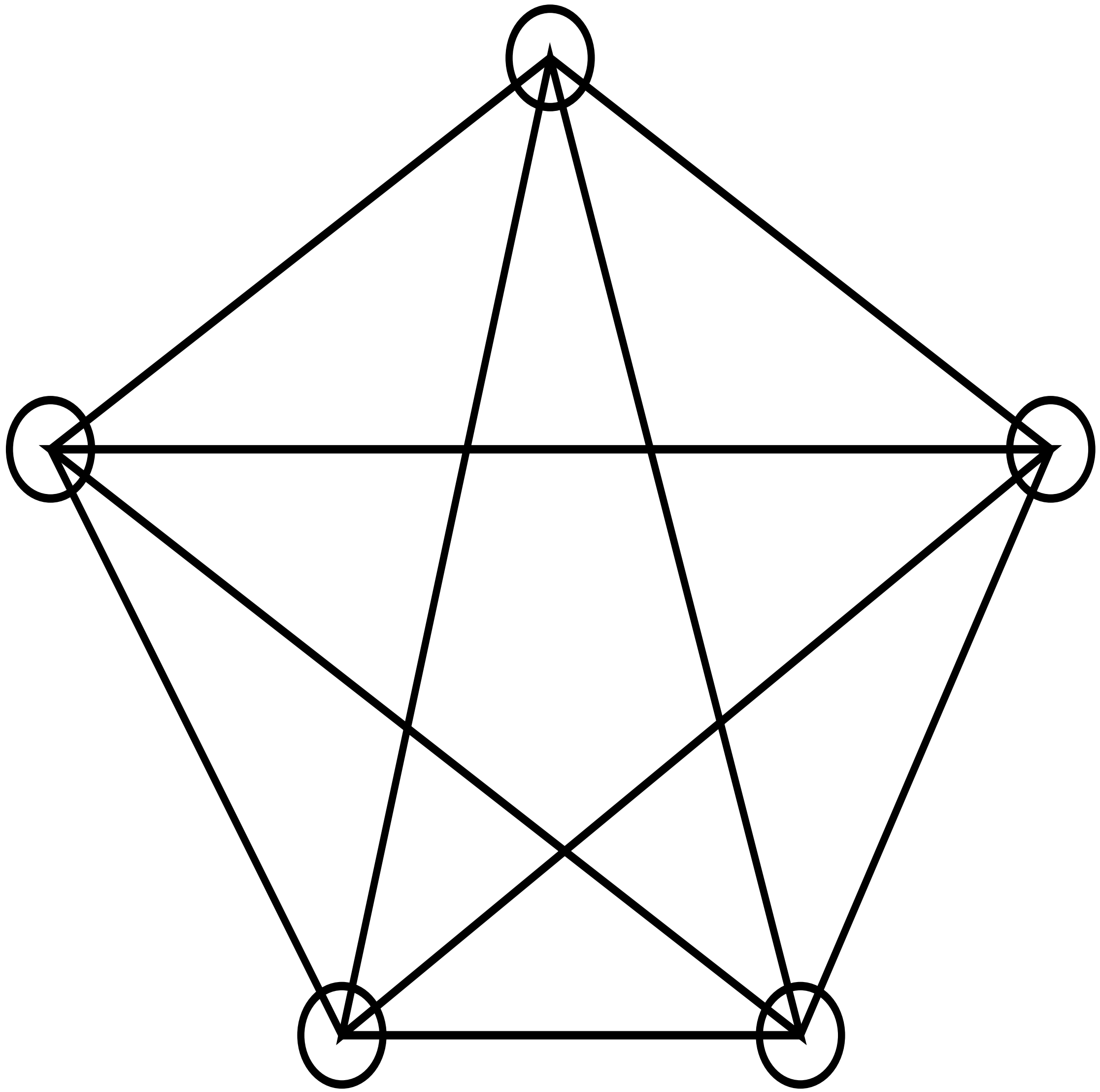
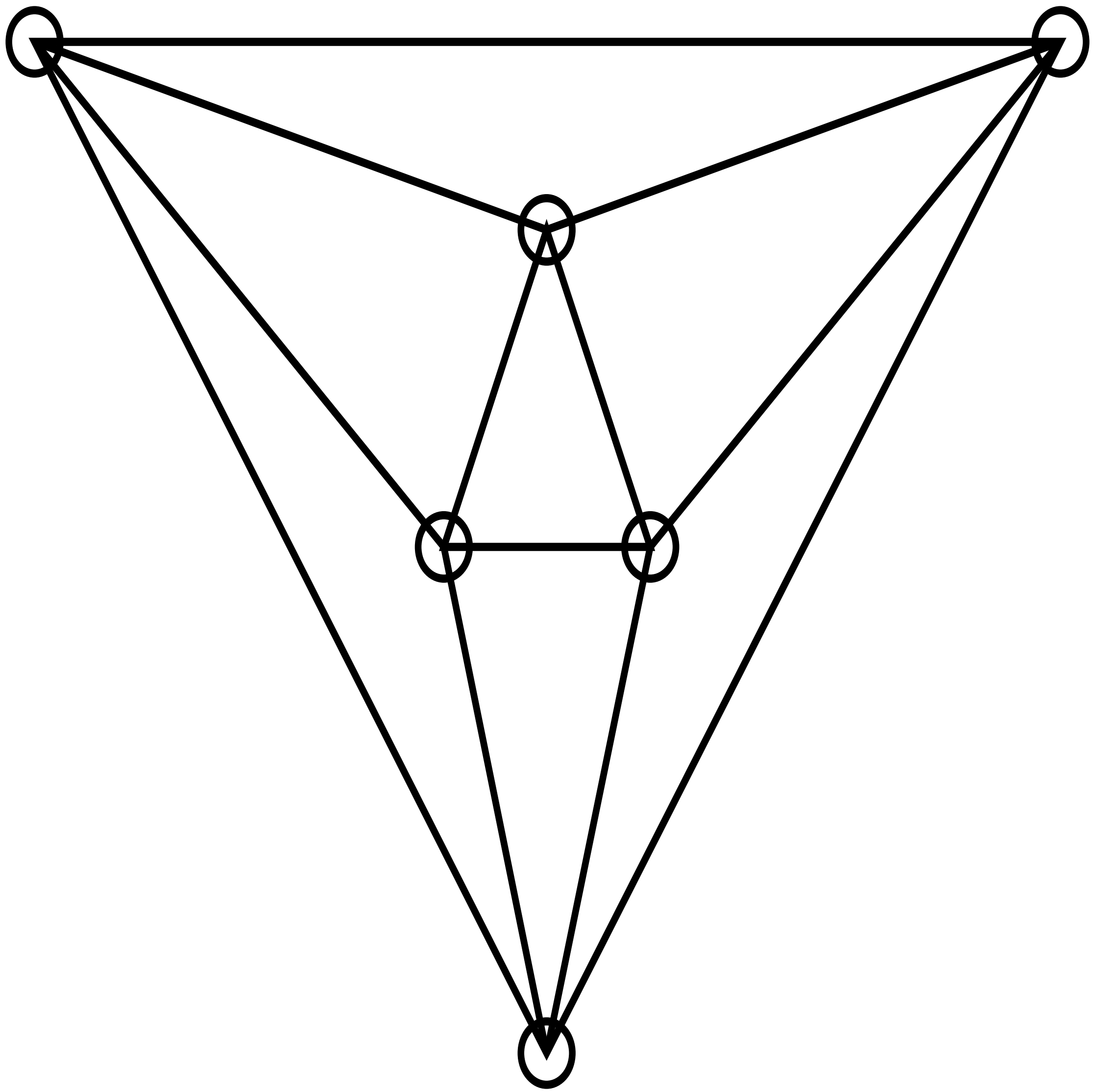


Figure 3: K-2-2-2



3–realizable graph II

Then, it either

- contains an V_8 or an $C_5 \times C_2$ as a minor

Figure 4: V-8

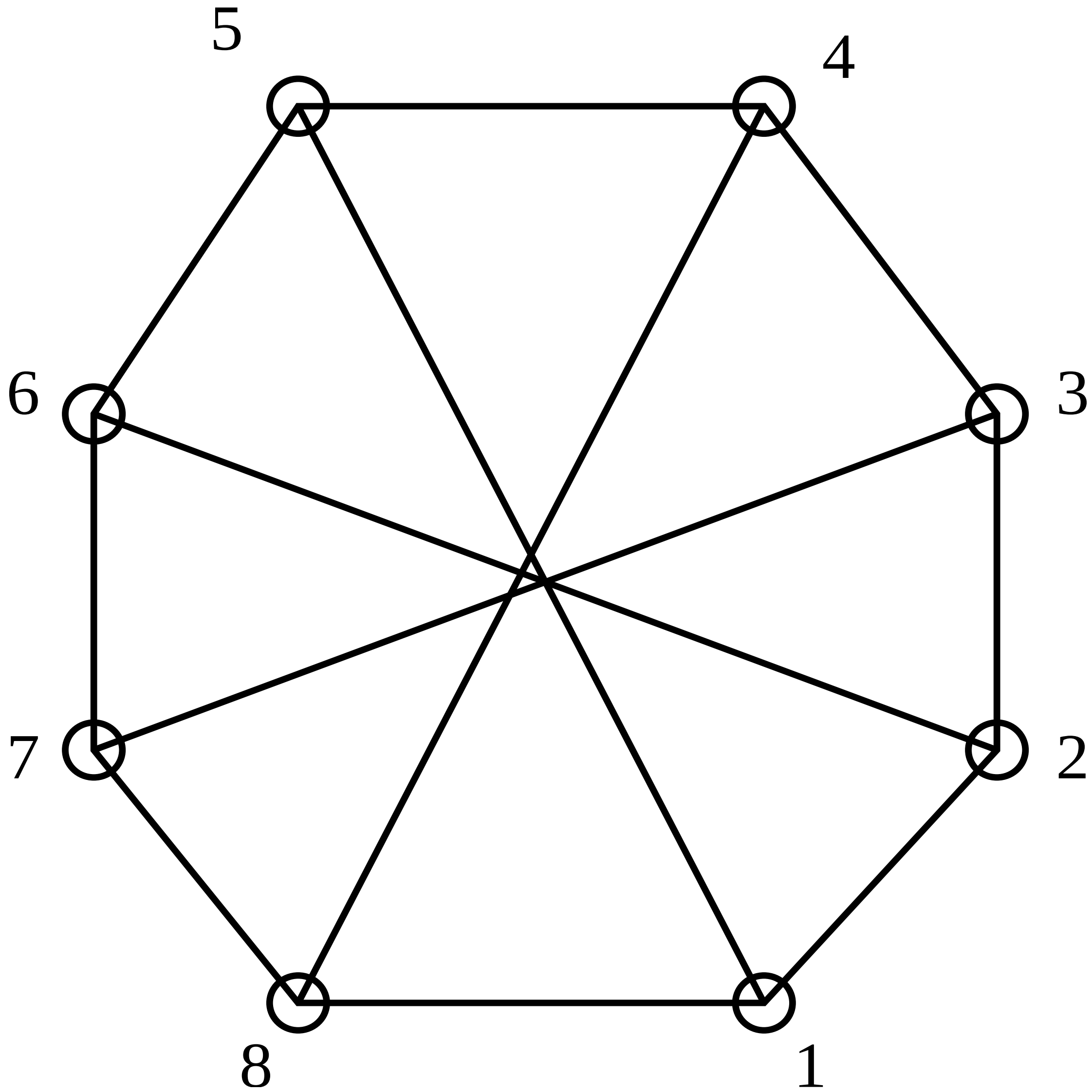
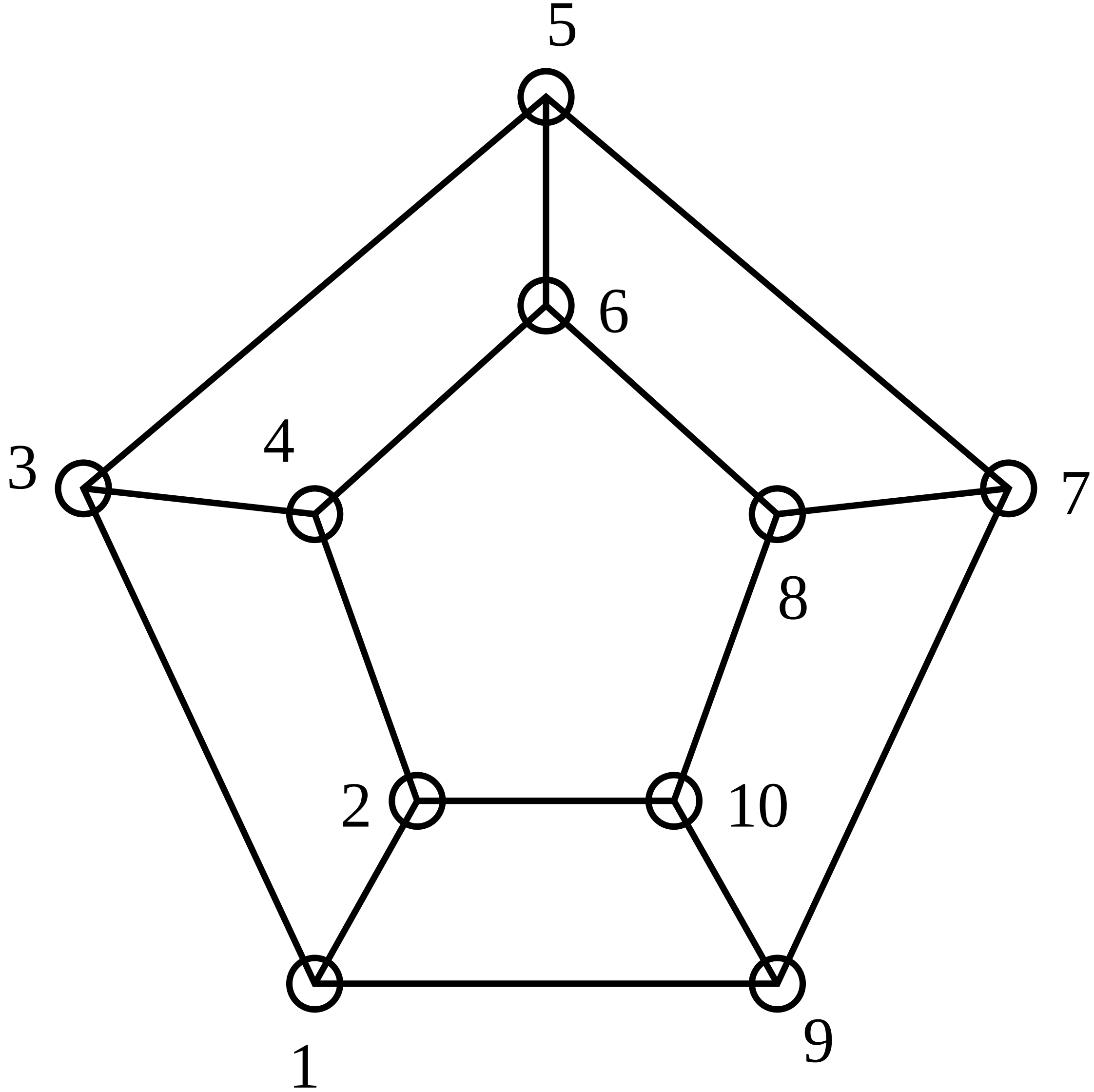


Figure 5: C-5×C-2



3–realizable graph II

Then, it either

- contains an V_8 or an $C_5 \times C_2$ as a minor
- or does not contain either graphs as a minor.

3–realizable graph II

Then, it either

- contains an V_8 or an $C_5 \times C_2$ as a minor
- or does not contain either graphs as a minor.

If it is the latter, G is a partial 3–tree.

An k-tree is defined recursively as follows. The complete graph on k vertices is an k–tree. An k–tree with $n+1$ vertices (where $n \geq k$) can be constructed from an k–tree with n vertices by adding a vertex adjacent to all vertices of one of its k–vertex complete subgraphs, and only to those vertices.

A partial k–tree is a subgraph of an k–tree.

Our Contributions

We resolve the above open question by giving a polynomial time algorithm for (approximately) realizing 3–realizable graphs.

Our Contributions

We resolve the above open question by giving a polynomial time algorithm for (approximately) realizing 3–realizable graphs.

The main bottleneck in the proof is to show that two graphs, V_8 and $C_5 \times C_2$, are 3–realizable.

Our Contributions

We resolve the above open question by giving a polynomial time algorithm for (approximately) realizing 3–realizable graphs.

The main bottleneck in the proof is to show that two graphs, V_8 and $C_5 \times C_2$, are 3–realizable.

There exists a realization $\mathbf{p}$ of $H \in \{V_8, C_5 \times C_2\}$ such that the distance between a certain pair of non–adjacent vertices (i, j) is maximized. Such a realization induces a non–zero equilibrium stress on the graph H' obtained from H by adding the edge (i, j). Then use this equilibrium force to prove that H' must be in $\mathbf{R}^3$.

Our Contributions

We resolve the above open question by giving a polynomial time algorithm for (approximately) realizing 3–realizable graphs.

The main bottleneck in the proof is to show that two graphs, V_8 and $C_5 \times C_2$, are 3–realizable.

There exists a realization $\mathbf{p}$ of $H \in \{V_8, C_5 \times C_2\}$ such that the distance between a certain pair of non–adjacent vertices (i, j) is maximized. Such a realization induces a non–zero equilibrium stress on the graph H' obtained from H by adding the edge (i, j). Then use this equilibrium force to prove that H' must be in $\mathbf{R}^3$.

Our main result is to show that the problem of computing the desired $\mathbf{p}$ can be formulated as an SDP. More interesting is that the optimal dual multipliers of our SDP give rise to a non–zero equilibrium stress.

Preliminaries

Let $G = (V, E; \mathbf{d})$ be a weighted connected graph that contains neither loops nor multiple edges, such that $d_{ij} \geq 0$ for all $(i, j) \in E$.

Preliminaries

Let $G = (V, E; \mathbf{d})$ be a weighted connected graph that contains neither loops nor multiple edges, such that $d_{ij} \geq 0$ for all $(i, j) \in E$.

A tensegrity $G(\mathbf{p})$ is a graph $G = (V, E)$ together with a configuration $\mathbf{p} = (p_i) \in \mathbf{R}^D \times \cdots \times \mathbf{R}^D = \mathbf{R}^{|V|D}$ such that each edge is labelled as a cable, strut, or bar, and each vertex is labelled as pinned or unpinned. $G(\mathbf{p})$ is the realization of G in $\mathbf{R}^D$ obtained by locating vertex i at point $p_i \in \mathbf{R}^D$.

Preliminaries

Let $G = (V, E; \mathbf{d})$ be a weighted connected graph that contains neither loops nor multiple edges, such that $d_{ij} \geq 0$ for all $(i, j) \in E$.

A tensegrity $G(\mathbf{p})$ is a graph $G = (V, E)$ together with a configuration $\mathbf{p} = (p_i) \in \mathbf{R}^D \times \cdots \times \mathbf{R}^D = \mathbf{R}^{|V|D}$ such that each edge is labelled as a cable, strut, or bar, and each vertex is labelled as pinned or unpinned. $G(\mathbf{p})$ is the realization of G in $\mathbf{R}^D$ obtained by locating vertex i at point $p_i \in \mathbf{R}^D$.

The label on each edge is intended to indicate its functionality: cables (resp. struts) are allowed to decrease (resp. increase) in length (or stay the same length), but not to increase (resp. decrease) in length; bars are forced to remain the same length.

A pinned vertex is forced to remain where it is.

Equilibrium Stress

An equilibrium stress for $G(\mathbf{p})$ is an assignment of real numbers $\omega_{ij} = \omega_{ji}$ to each edge $(i, j) \in E$ such that for each unpinned vertex i of G, we have $\sum_{j:(i,j)\in E} \omega_{ij}(p_i - p_j) = \mathbf{0}$. Furthermore, we say that the equilibrium stress $\omega = \{\omega_{ij}\}$ is proper if $\omega_{ij} = \omega_{ji} \geq 0$ (resp. ≤ 0) if (i, j) is a cable (resp. strut).

A Semidefinite Programming (SDP) Formulation

Consider a simple model with C (or S) is a set of cables (or strut):

$$
\begin{aligned}
\text{max} \quad & \textstyle\sum_{(i,j)\in S} \|x_i - x_j\|^2 - \sum_{(i,j)\in C} \|x_i - x_j\|^2 \\
\text{s.t.} \quad & \|x_i - x_j\|^2 = d_{ij}^2, \ \forall\, (i,j) \in N_x, \ i < j, \\
& \|a_k - x_j\|^2 = d_{kj}^2, \ \forall\, (k,j) \in N_a.
\end{aligned}
$$

Matrix Representation

Let $X = [x_1 \; x_2 \; ... \; x_n]$ be the $d \times n$ matrix that needs to be determined. Then

$\|x_i - x_j\|^2 = e_{ij}^T X^T X e_{ij}$ and $\|a_k - x_j\|^2 = (a_k; e_j)^T [I \;\; X]^T [I \;\; X] (a_k; e_j)$,

where e_{ij} is the vector with 1 at the ith position, -1 at the jth position and zero everywhere else; and e_j is the vector of all zero except -1 at the jth position.

Matrix Representation

Let $X = [x_1\ x_2\ ...\ x_n]$ be the $d \times n$ matrix that needs to be determined. Then

$$\|x_i - x_j\|^2 = e_{ij}^T X^T X e_{ij} \text{ and } \|a_k - x_j\|^2 = (a_k; e_j)^T [I\ \ X]^T [I\ \ X](a_k; e_j),$$

where e_{ij} is the vector with 1 at the ith position, -1 at the jth position and zero everywhere else; and e_j is the vector of all zero except -1 at the jth position.

$$\begin{array}{ll}
\text{max} & \sum_{(i,j)\in S} e_{ij}^T Y e_{ij} - \sum_{(i,j)\in C} e_{ij}^T Y e_{ij} \\
\text{s.t.} & e_{ij}^T Y e_{ij} = d_{ij}^2,\ \forall\, i,j \in N_x,\ \forall\,(i,j) \in N_x,\ i<j, \\
& (a_k; e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; e_j) = d_{kj}^2,\ \forall\, k,j \in N_a, \\
& Y = X^T X.
\end{array}$$

where Y denotes the Gram matrix $X^T X$.

SDP Relaxation

Change

$$Y = X^T X$$

to

$$Y \succeq X^T X.$$

This matrix inequality is equivalent to (e.g., Boyd et al. 1994)

$$\begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succeq 0.$$

SDP standard form

$$Z = \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix}.$$

Find a symmetric matrix $Z \in \mathbf{R}^{(2+n)\times(2+n)}$ such that

$$\begin{array}{ll} \text{sup} & \left(\sum_{(i,j)\in S}(\mathbf{0};e_{ij})(\mathbf{0};e_{ij})^T - \sum_{(i,j)\in C}(\mathbf{0};e_{ij})(\mathbf{0};e_{ij})^T\right) \bullet Z \\ \text{s.t.} & Z_{1:d,1:d} = I \\ & (\mathbf{0};e_{ij})(\mathbf{0};e_{ij})^T \bullet Z = d_{ij}^2, \ \forall\, i,j \in N_x,\ i<j, \\ & (a_k;e_j)(a_k;e_j)^T \bullet Z = d_{kj}^2, \ \forall\, k,j \in N_a, \\ & Z \succeq 0. \end{array}$$

The Dual of the SDP Relaxation

$$\begin{array}{ll} \text{inf} & I \bullet V + \sum_{i<j \in N_x} w_{ij} d_{ij}^2 + \sum_{k,j \in N_a} w_{kj} d_{kj}^2 \\ \text{s.t.} & \begin{pmatrix} V & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \sum_{i<j \in N_x} w_{ij} (\mathbf{0}; e_{ij})(\mathbf{0}; e_{ij})^T \\ & + \sum_{k,j \in N_a} w_{kj} (a_k; e_j)(a_k; e_j)^T \succeq \\ & \sum_{(i,j) \in S} (\mathbf{0}; e_{ij})(\mathbf{0}; e_{ij})^T - \sum_{(i,j) \in C} (\mathbf{0}; e_{ij})(\mathbf{0}; e_{ij})^T, \end{array}$$

where variable matrix $V \in \mathcal{M}^d$, varaible w_{ij} is the weight on edge from x_i to x_j, and w_{kj} is the weight on edge from a_k to x_j. As we shall see, the optimal w_{ij} are closely related to an equilibrium stress for a certain realization of G.

Analysis of the SDP Formulation

Theorem 1. *Let $\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_n]$ be the positions of the unpinned vertices obtained from the optimal primal matrix $\bar{Z}$, and let $\{\bar{\theta}_{ij}, \bar{w}_{kj}\}$ be a set of optimal dual multipliers. Suppose that we assign the stress $\bar{\theta}_{ij}$ (resp. $\bar{w}_{kj}$) to the bar $(i,j) \in N_x$ (resp. $(k,j) \in N_a$), a stress of 1 to all the cables, and a stress of -1 to all the struts. Then, the resulting assignment yields a non–zero proper equilibrium stress for the realization $\{(a_1; \mathbf{0}), \ldots, (a_m; \mathbf{0}), \tilde{x}_1, \ldots, \tilde{x}_n\}$.*

Proof: The primal is feasible and the dual is strictly feasible. Let $\bar{Z}$ (resp. $\bar{U}$) be the optimal primal (resp. dual) solution matrix. Then, the absence of a duality gap implies complementarity:

$$\bar{Z}\bar{U} = \mathbf{0}.$$

Algorithm Tasks

1. Realizing a partial 3–tree;

Algorithm Tasks

1. Realizing a partial 3–tree;

2. finding a subdivision of V_8 or $C_5 \times C_2$ in an 3–realizable graph;

Algorithm Tasks

1. Realizing a partial 3–tree;
2. finding a subdivision of V_8 or $C_5 \times C_2$ in an 3–realizable graph;
3. realizing an V_8 and its subdivisions;

Algorithm Tasks

1. Realizing a partial 3–tree;
2. finding a subdivision of V_8 or $C_5 \times C_2$ in an 3–realizable graph;
3. realizing an V_8 and its subdivisions;
4. realizing an $C_5 \times C_2$ and its subdivisions.

1: Realizing Partial 3–Trees

Suppose that we are given a 3–tree G with feasible edge lengths, and that G is constructed by adding the vertices $v_1, v_2, \ldots, v_n$, in that order. Then, to find a realization of G in $\mathbf{R}^3$ can be done in linear time. A partial 3–tree can be completed into a 3–tree by solving an SDP.

2: Finding a Subdivision of V_8 or $C_5 \times C_2$

Let G be an 3–realizable graph. We now show how the algorithm of Matou**v**sek and Thomas can be used to obtain a subgraph of G that is a subdivision of V_8 or $C_5 \times C_2$. We shall also use the term "homeomorphic" for subdivision – a graph H_1 is homeomorphic to H_2 if H_1 is a subdivision of H_2.

1. (Asano) For an 3–connected graph H, a graph H' has a subgraph homeomorphic to H iff there is an 3–connected component of H' that has a subgraph homeomorphic to H.

2. (Connelly and Sloughter) If an edge is added between a non–adjacent pair of vertices of V_8 (resp. $C_5 \times C_2$), then the resulting graph has K_5 (resp. K_5 or $K_{2,2,2}$) as a minor.

3. (Connelly and Sloughter) Let G be an 3–realizable graph. Suppose that G contains a subdivision of H, where $H \in \{V_8, C_5 \times C_2\}$. Remove the subdivision of H from G and consider the components of the resulting graph.

Then, each component is connected in G to exactly one of the subdivided edges of H.

Theorem 2. *Let G be an 3–realizable graph containing a subgraph homeomorphic to $H \in \{V_8, C_5 \times C_2\}$. Then, one of the triconnected components of G is isomorphic to H.*

Algorithm: First, decompose G into triconnected components. Then, we check each of the triconnected components for the presence or absence of V_8 or $C_5 \times C_2$. For this we can run the algorithm on each of those components and see if the component reduces to a null graph or not. If the component does not reduce to a null graph, then it is isomorphic to either V_8 or $C_5 \times C_2$, and the number of vertices in the component will determine which one it is. The desired subdivision can then be extracted from G.

Proposition 1. *Let G be an 3–realizable graph with n vertices. Then, a subdivision of V_8 or $C_5 \times C_2$ in G can be found in $O(n)$ time.*

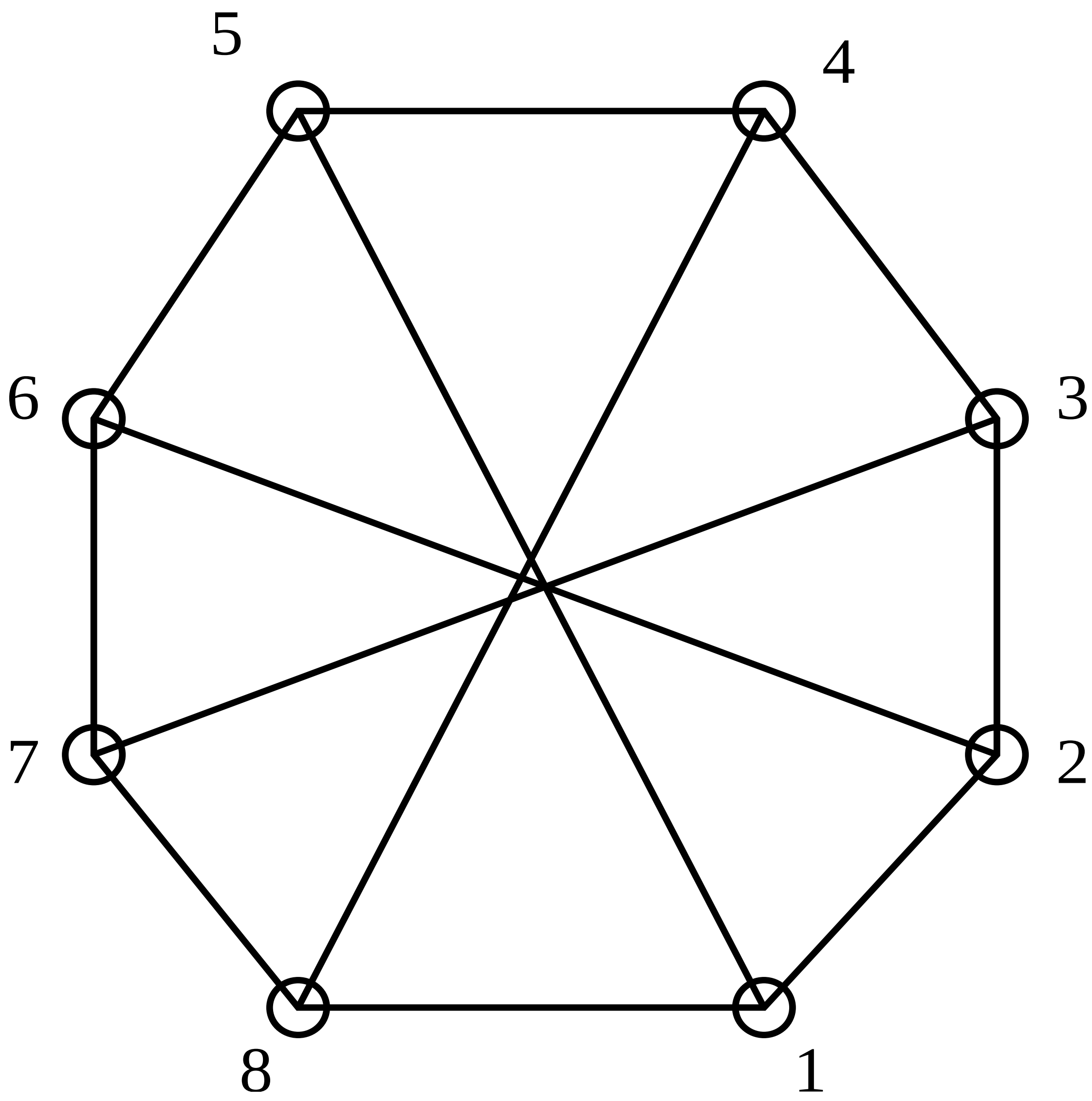
3: Realizing V_8 and its Subdivisions

The graph V_8 is 3–realizable. We first augment V_8 to V_8' by adding a strut between vertices 1 and 4 Then, we pin vertex 1 at the origin.

In other words, we would like to find a realization that maximizes the length of the strut.

$$
\begin{array}{lll}
\sup & (\mathbf{0}; e_4)(\mathbf{0}; e_4)^T \bullet Z & \\
\text{s.t.} & Z_{1:3,1:3} = I_3 & \\
 & (0; e_{ij})(0; e_{ij})^T \bullet Z = d_{ij}^2 & (i,j) \in E(V_8) \\
 & & 1 \neq i < j \\
 & (\mathbf{0}; e_j)(\mathbf{0}; e_j)^T \bullet Z = \bar{d}_{1j}^2 & (1,j) \in E(V_8) \\
 & Z \succeq 0 &
\end{array}
\qquad (1)
$$

Figure 6: V-8



4: Realizing $C_5 \times C_2$ and its Subdivisions

The graph $C_5 \times C_2$ is 3–realizable. We first augment $C_5 \times C_2$ to G by adding a strut between vertices 1 and 6, and we pin vertex 1 at the origin.

Putting Everystep Together

Theorem 3. *Therer is a polynomial time algorithm for (approximately) realizing 3–realizable graphs*.

Conclusion

- We have studied a connection between SDP and tensegrity theories, as well as the notion of d–realizability of graphs.

Conclusion

- We have studied a connection between SDP and tensegrity theories, as well as the notion of d–realizability of graphs.
- We have shown that the problem of finding an equilibrium stress can be formulated as an SDP. This gives a constructive proof of (a variant of) a result in tensegrity theory that is previously established by non–constructive means.

Conclusion

- We have studied a connection between SDP and tensegrity theories, as well as the notion of d–realizability of graphs.
- We have shown that the problem of finding an equilibrium stress can be formulated as an SDP. This gives a constructive proof of (a variant of) a result in tensegrity theory that is previously established by non–constructive means.
- We then combine this result with other techniques to design an algorithm for realizing 3–realizable graphs, thus answering an open question posed before.

Conclusion

- We have studied a connection between SDP and tensegrity theories, as well as the notion of d–realizability of graphs.
- We have shown that the problem of finding an equilibrium stress can be formulated as an SDP. This gives a constructive proof of (a variant of) a result in tensegrity theory that is previously established by non–constructive means.
- We then combine this result with other techniques to design an algorithm for realizing 3–realizable graphs, thus answering an open question posed before.
- We believe that our techniques can be applied to derive some other interesting properties of tensegrity frameworks.