CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

PROTEIN FOLDING: PLANAR CONFIGURATION SPACES OF DISC ARRANGEMENTS AND HINGED POLYGONS

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Applied Mathematics

by

Clinton Bowen

August 2014

The thesis of Clinton Bowen is approved:

Dr. Silvia Fernandez	Date
Dr. John Dye	Date
Dr. Csaba Tóth, Chair	Date

California State University, Northridge

Table of Contents

ABSTRACT

PROTEIN FOLDING: PLANAR CONFIGURATION SPACES OF DISC ARRANGEMENTS AND HINGED POLYGONS

By

Clinton Bowen

Master of Science in Applied Mathematics

Chapter 1

Background

Decidability problems study whether there exists a way to determine whether an element is a member of a set. In this paper we focus on four such decidability problems surrounding graph theory and geometry. The first set of problems involve a special type of graph called a tree and the second set of problems involve something called a polygonal linkage. In each problem, set membership is determined if the tree or polygonal linkage has a particular property when visualized in the plane.

This thesis first presents the preliminary information needed to pose our four problems, then we formally pose each problem and then provide solutions on decidability for each problem.

1.1 Disk Arrangements

A *disk arrangement* is a set of interior disjoint disks, D. If for any pair of disks in D intersect at a boundary point, they are said to be in contact (kissing). A *contact graph* $G = (V,E)$ corresponding to a given

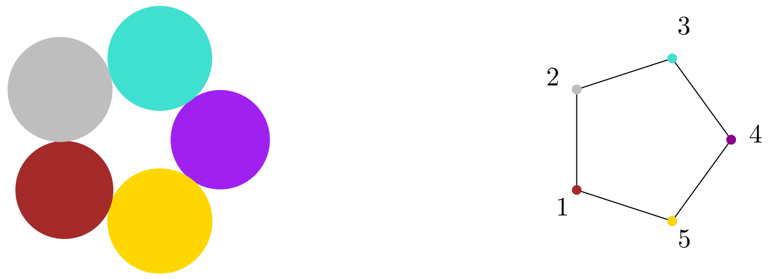


Figure 1.1: This example represents a disk arrangement transformed to and from its corresponding graph G_2

disk arrangement where there is a bijection $b_V : V \mapsto D$ and a bijection that maps an edge $e_{i,j} \in E$ to an interior disjoint pair of disks $d_i, d_j \in D$. Let $\omega : V \mapsto \mathbb{R}^+$ be the *weight function*. ω assigns a weight to each vertex in V. *D respects radii assignments* if for every $v \in V$ such that $b_V(v) = D_v$, $\omega(v)$ is the radius of D_v. Let $\Pi : V \mapsto \mathbb{R}^2$ be that planar mapping of vertices. *D respects center assignments* if for every $v \in V$, $\Pi(v)$ is the center of D_v. In this thesis, unless otherwise stated we assume that the contact graph associates to an ω and Π mapping that respect radii and center assignments. Koebe's theorem states that for every connected simple planar graph G, there exists a planar disk arrangement whose contact graph is G.

Given a disk arrangement, the contact graph can be thought of as a linkage because the distance between two kissing disk equal the sum of radii. However if the two disks don't kiss, the distance between their centers is strictly greater than the sum of their radii.

For a tree with positive weighted vertices, we posed two realizability problems:

Problem 1 (Unordered Realizibility Problem for the Tree). Given a tree with positive weighted vertices, is it a contact graph of some disk arrangement where the radii equal the vertex weights?

Problem 2 (Ordered Realizibility Problem for the Tree). Given an ordered tree with positive weighted vertices, is it the ordered contact graph of some disk arrangement where the radii equal the vertex weights?

There exists a tree with no realizable disk arrangement. To show that this can be done with even the simplest of trees and linkages, consider the binary trees $\{T_i\}_{i=1}^{n}$ with unit distance between adjacent vertices. To show that as $i \to \infty$, the corresponding disk arrangement will not have a drawing. We construct the disk arrangement as follows: (1) place a disk of unit radius centered at origin; (2) for each disk continue to add two non-intersecting kissing disks of unit radius to it. To illustrate, see Figure 1.3: The first round has one

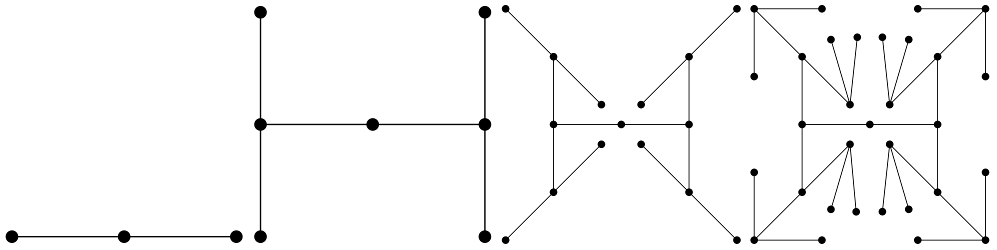
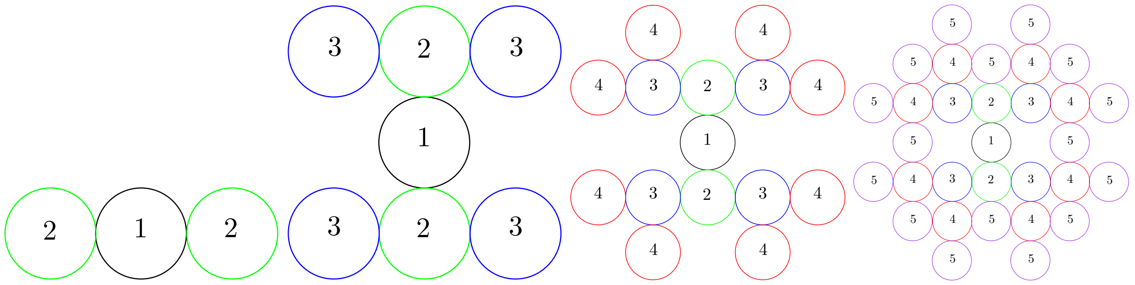

(a) Binary tree T_1. (b) Binary tree T_2. (c) Binary tree T_3. (d) Binary tree T_4.

Figure 1.2: We show the linkages T_1 through T_4 with unit distance between adjacent vertices.



(a) A disk arrangement with two layers of disks (b) A disk arrangement with three layers of disks (c) A disk arrangement with four layers of disks (d) A disk arrangement with five layers of disks

Figure 1.3: The gradual growth of disk arrangements by adding two kissing disks to each of the previously generated disks. By continuing this arrangement growth, the space needed to contain the kissing disks will exceed the area containing the disk arrangements.

unit disk. The next round, two unit radii disk are in contact with the first, i.e. Figure 1.3a. The subsequent rounds are shown in Figures 1.3b, 1.3c, and 1.3d. For each round i we are adding $2^{(i-1)}$ disks, each with an area of π. The area that the disk arrangement is bounded at round i is a box of length $2\cdot(2\cdot(i-1)+1)$ totalling to an area of $(4\cdot i^2-4\cdot i+1)$. Meanwhile the total area of the disk arrangement at i is $\pi\cdot(2^i-1)$. The exponential growth rate of the disk packing will exceed its bounded area for sufficiently large i, i.e. pick $i\geq 6$.

Let the total area of the disk arrangement be the sum of the areas of the disks. The upper bound of the of the total area of the disk arrangement is the boxed region about the disk arrangement. The upper bound of the total area is $(2\cdot(i-1))^2$. Suppose the root of the disk arrangement is centered at origin. The centers of the disks at level i are at a distance at most $2\cdot(i-1)$ away from the origin. Every disk at level up to i is contained in a disk of radius $2\cdot i-1$ centered at the origin. The total area of the disk arrangement is $(2\cdot i-1)^2\cdot\pi$. When $i>$??? the total area of the disk arrangement exceeds the total bounded area and thus we have a contradiction.

The embedding problems for trees and corresponding disk arrangements are as follows:

1.2 Configuration Spaces

Just as one can compose colors or forms, so one can compose motions.

Alexander Calder, 1933

Recall Figure **??** illustrating the hinged dissection that formed a square and triangle and several drawings of the hinged dissections that simulate the motion of moving the polygons around the hinge points to form each shape. The set of all drawings in that motion represents the *configuration space* for that polygonal linkage. In this section we will formally describe the configuration space for each object we've drawn thus far.

1.2.1 Configuration Spaces of Graph Drawings

Recall that for a graph drawing we have an injective mapping $\Pi : V \mapsto \mathbb{R}^2$ which maps vertices to distinct points in the plane and for each edge $\{u,v\} \in E$, a straight line segment, $c_{u,v} : [0,1] \mapsto \mathbb{R}^2$ such that $c_{u,v}(0) = \Pi(u)$ and $c_{u,v}(1) = \Pi(v)$, and does not pass through other vertices. For each vertex of G, the embedding of the vertex lies in the plane, i.e. $\Pi(v) \in \mathbb{R}^2$. By enumerating each vertex of G, e.g. $v_1, v_2, \ldots, v_k, \ldots, v_n$, we can create a projection mapping from $\mu : \Pi \mapsto \mathbb{R}^{2|V|}$ where the corresponding coordinates of $\Pi(v_k)$ are in the $(2k)^{\text{th}}$ and $(2k+1)^{th}$ coordinates in $\mathbb{R}^{2|V|}$. $\mu(\Pi)$ is a configuration. The configuration space is the set of $\mu(\Pi)$ for all drawings Π.

1.2.2 Configuration Spaces of Linkages

Consider drawings of a graph that respects the length assignment. A *realization* of a linkage, (G, ℓ), is a drawing of a graph, Π, such that for every edge $\{u,v\} \in E$, $\ell(\{u,v\}) = |\Pi(u) - \Pi(v)| = |\Pi(v) - \Pi(u)|$. A *plane realization* is a plane drawing with the property, $\ell(\{u,v\}) = |\Pi(u) - \Pi(v)|$. First let's define the space of realizations for a corresponding linkage, i.e.:

$$P_{(G,\ell)} = \left\{ \Pi_{(G,\ell)} \,|\, \forall \{u,v\} \in E,\ \ell(\{u,v\}) = |\Pi(u) - \Pi(v)| \right\}$$

With respect to P, we can establish a *configuration space* that allows one to study problems of motion. For each vertex of G, the drawing of the vertex lies in the plane, i.e. $\Pi(v) \in \mathbb{R}^2$. By enumerating each vertex of G, e.g. $v_1, v_2, \ldots, v_k, \ldots, v_n$, we can create a projection mapping from $\mu : P \mapsto \mathbb{R}^{2|V|}$ where the corresponding coordinates of $\Pi(v_k)$ are in the $(2k)^{\text{th}}$ and $(2k+1)^{th}$ coordinates in $\mathbb{R}^{2|V|}$. The configuration space is $\mu(P)$.

Using standard definitions from real analysis, we can begin to pose problems about linkages with respect to a corresponding configuration space. We define a path $\gamma : [0,1] \mapsto \mu(P)$ where $\gamma(0)$ corresponds the the projection of a realization of a linkage Π_0 and $\gamma(1)$ corresponds to another realization of a linkage Π_1. If for any two elements $a, b \in \mu(P)$ that there exists a continuous path γ such that $\gamma(0) = a$ and $\gamma(1) = b$, $\mu(P)$ is said to be path connected. For γ to be continuous we would have that for every $\varepsilon > 0$, there exists a $\delta > 0$ such that if $x, y \in [0,1]$ and $|x - y| < \delta$ then $||\gamma(x) - \gamma(y)|| < \varepsilon$. γ can be thought of as an animation of drawings that starts at $\gamma(0)$ and ends at $\gamma(1)$. To ask if $\mu(P)$ is a connected space, is to ask if $\mu(P)$ is connected in $\mathbb{R}^{2|V|}$. The Carpenter's Rule states that every realization of a path linkage can be continuously moved (without self-intersection) to any other realization [3, 7]. In other words, the realization space of such a linkage is always connected.

1.2.3 Configuration Spaces of Polygonal Linkages

Recall a realization of a polygonal linkage is an interior-disjoint placement of congruent copies of the polygons in $\mathscr{P}$ such that the copies of a hinge are mapped to the same point (e.g., Figure **??**). First consider the set of all realizations for the polygonal linkage. $(\mathscr{P}, \mathscr{H})$ and call it P. For any realization $R \in P$, the parameterization $\mu : R \mapsto \mathbb{R}^{2m}$ where m is the number of distinct vertices in $\mathscr{P}$. The configuration space is the set $\mu(P)$.

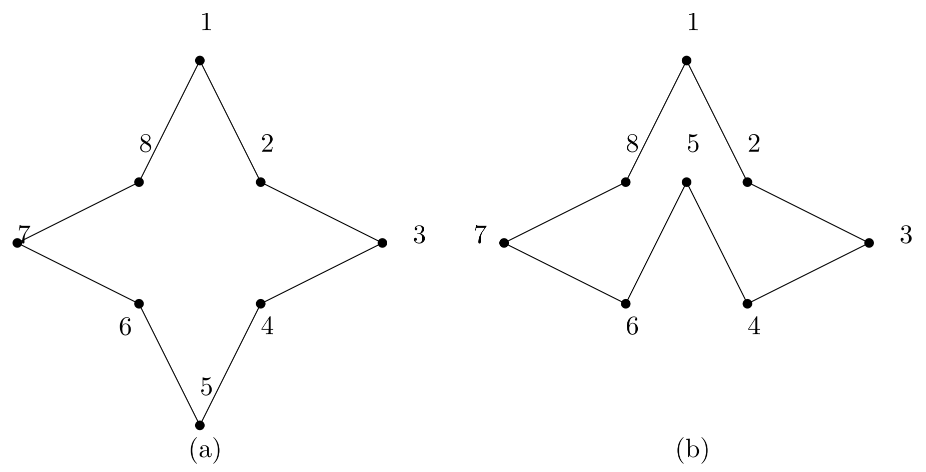


Figure 1.4: (a) and (b) show a linkage in two embeddings. Any realization of a path can be continuously moved without self-intersection to any other realizations.

1.2.4 Configuration Spaces of Disk Arrangements

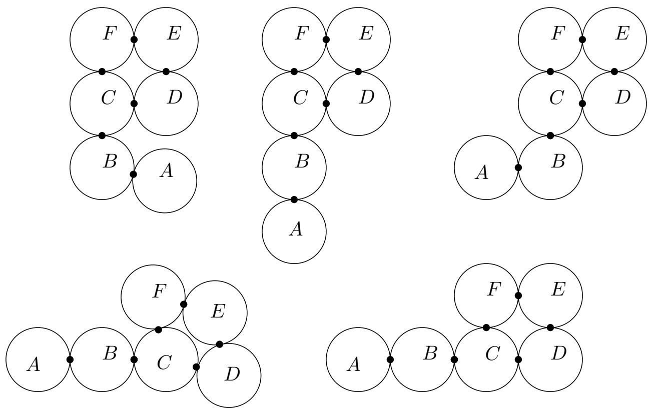


Figure 1.5: An example of a disk arrangement where A and B have a large range of freedom to move around. C, D, E, and F are limited in their range of motion to due to their hinge points.

Consider the set of realizations P for a given disk arrangement $\mathscr{D} = \{D_i\}_{i=1}^n$. For any realization $R \in P$, there exists a corresponding contact graph, C. The configuration spaces of $\mathscr{D}$ are sets of $R \in P$ that are classified by the equivalent contact graphs, i.e. if R_1, $R_2 \in P$ and their corresponding contact graphs C_1 and C_2 have a graph isomorphism, ϕ, then R_1 and R_2 belong to the same configuration space.

1.3 Algorithm Complexity

Algorithms are a list of instructions. When an algorithm executes its procedure it can be measured in terms of units of consumed resources (in computers, that is memory) and the time it takes to complete the procedure of calculations. Ideally, a desirable algorithm would primarily have a small run time and secondarily utilize a small amount of resources.

Determining the time and space that algorithms use determine their efficiency. The *worst-case* running time is the largest possible running time that an algorithm could have over all inputs of a given size N. *Brute force* is when an algorithm tries all possibilities to see if any formulates a solution. An algorithm is said to be *efficient* if the time needed to perform the list of instructions can be determined from a polynomial.

For combinatorial problems, as the number of inputs of the problem grows, the solution space tends to grow exponentially. In general, as problems grow, it is desirable to minimize the *running time*, time take to run an algorithm that solves a problem. Formally, we quantify running time with Big O notation.

Definition 1 (Big O Notation)**.** Let f and g be defined on some subset of $\mathbb{R}$. $f(x) = O(g(x))$ if and only if there exists a positive real number M and x_0 such that

$$|g(x)| \leq M\,|f(x)|$$

for all $x \geq x_0$

An algorithm has a *polynomial running time* if there is a polynomial function p such that for every input string s, the algorithm terminantes on s in at most $O(p(|s|))$ steps.

To categorize problems [5], we ask the following:

Problem 3. Can arbitrary instances of problem Y by solved using a polynomial number of standard computational steps, plus a polynomial number of calls to an algorithm that solves X?

The class of problems that can be solved in polynomial running time is called the *polynomial time* class, P. A second property of problems is whether if its solution can be verified efficiently. This property is independent of whether it can be solved efficiently. B is said to be an efficient certifier for a problem X if the following properties hold:

(i) B is a polynomial-time algorithm that takes two inputs s and t.

(ii) There exists a polynomial function p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s,t) =$ 'yes'.

The class of problems which have an efficient certifier is said to be the *nondeterministic polynomial time* class, NP. Before we continue with the definitions for NP and NP complete, we will look into a type of problem, a reduction of a problem, and what an efficient certification is. This facilitates the reader for the definitions and illustrate complexity better.

A *polynomial time reduction* is when arbitrary instances of problem Y be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to a black box that solves problem X.

1.3.1 Independent Sets and Vertex Covers

To illustrate what a reduction is, we cover an example of independent sets and vertex covers. Given a graph $G = (V,E)$, a set of vertices $S \subset V$ is *independent* if no two vertices in S are joined by an edge. A *vertex cover* of a graph $G = (V,E)$ is a set of vertices $S \subset V$ if every edge $e \in E$, has at least one end corresponding in S.

Theorem 1. *Let $G = (V,E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.*

Proof 1. *If S is an independent set. Then for any pair of vertices in S, the pair are not joined by an edge if and only if for any $v_1, v_2 \in S$, $e = (v_1, v_2) \notin E$. We have two cases. The first case is if $v \in S$, then any vertex $u \in V$ that forms an edge $e = (v,u) \in E$ must reside in $V - S$. The second case is if there is an edge which no pair of vertices is in S, then both vertices are in $V - S$. Both cases together imply that every edge has at least one end corresponding in $V - S$.*

If $V - S$ is a vertex cover. Every edge $e \in E$ has at least one vertex in $V - S$. The two possible cases, the first case is that the second vertex is in $V - S$, and the second case is that the second vertex is in S. The first *case would yield $S = \emptyset$. The second case implies that the edge $e \in E$ has exactly one vertex in $V - S$ and exactly one vertex in S. $V - S$ is a vertex cover would disallow S to have a pair of vertices to form an edge in the graph.*

Theorem 1 allows for problem reductions for independent set and vertex cover problems.

There are two problems for the independent set: an optimization problem and a decision problem.

Problem 4 (Optimization of an Independent Set in G). Given a graph G, what is the largest independent set in G?

Problem 5 (Decision of an Independent Set of Size k). Given a graph G and a number k, does G contain an independent set of size at least k?

Consider an algorithm A that determines whether G contains an independent set of size k. By querying A with $k = 1, 2, ..., |V|$, one can find a maximal independent set size in G. Conversely, if we have an algorithm B that can the largest independent set in $S \subset V$, then we know the order of S, $|S|$. Any subset of S is also an independent set; any subset $W \subset V$ such that $|S| < |W| \leq |V|$ is not independent. While B is an algorithm that solves Problem 4, B can be leveraged to solve Problem 5. Likewise, A solve Problem 5 but can be leveraged to solve Problem 4.

1.4 Satisfiability

Let $x_1, \ldots, x_n$ be boolean variables. A boolean formula is a combination of conjunction, disjunctions, and negations of the boolean variables $x_1, \ldots, x_n$. A *clause* is a disjunction of distinct literals. A *literal* is a variable or a negated variable, x_i or $\bar{x}_i$, for $i = 1, \ldots, n$. A boolean formula is *satisfiable* if one can assign true or false value to each variable so that the formula is true. It is known that every boolean formula can be rewritten in *conjunctive normal form* (CNF), a conjunction of clauses, via DeMorgan's law and distributive law. Furthermore, it is also known that every boolean formula can be written in CNF such that each clause has exactly three literals. This form is called 3-CNF. For example, consider the clause $A \vee B \vee C \vee D \vee E$. This clause can be rewritten in 3-CNF form as $(A \vee B \vee x_1) \wedge (\neg x_1 \vee C \vee x_2) \wedge (\neg x_2 \vee D \vee E)$ where x_1 and x_2 are literals that allow us to form 3-CNF clauses.

Problem 6 (Satisfiability Problem (SAT)). Given a boolean formula, is it satisfiable? [6]

Problem 7 (3-SAT Problem). Given a boolean formula in 3-CNF, is it satisfiable?

The problems we focus on in this thesis have a geometry. A special geometric 3-SAT problem is that Planar 3-SAT Problem. Given a 3-CNF boolean formula, B, we define the associated graph as follows: the vertices correspond to the variables and clauses in B, when a variable or its negation appears in a clause there is an edge between the corresponding vertices.

Problem 8 (Not All Equal 3 SAT Problem (NAE3SAT)). Given a boolean formula in 3-CNF, is it satisfiable so that each clause contains a true and a false literal?

Problem 9 (Planar 3-SAT). Given a boolean formula B in 3-CNF such that its associated graph is planar, decide whether it is satisfiable is a *3-SAT problem.*

1.5 Problem

The *realizability* problem for a polygonal linkage asks whether a given polygonal linkage has a realization (resp., orientated realization). For a weighted planar (resp., plane) graph,, it asks whether the graph is the contact graph (resp., ordered contact graph) of some disk arrangement with specified radii. These problems, in general, are known to be NP-hard. Specifically, it is NP-hard to decide whether a given planar (or plane) graph can be embedded in $\mathbb{R}^2$ with given edge lengths [2, 4]. Since an edge of given length can be modeled by a suitably long and skinny rhombus, the realizability of polygonal linkages is also NP-hard. The recognition of the contact graphs of unit disks in the plane (a.k.a. coin graphs) is NP-hard [1], and so the realizability of weighted graphs as contact graphs of disks is also NP-hard. However, previous reductions crucially rely on configurations with high genus: the planar graphs in [2, 4] and the coin graphs in [1] have many cycles.

In this thesis, we consider the above four realizability problems when the union of the polygons (resp., disks) in the desired configuration is simply connected (i.e., contractible). That is, the contact graph of the disks is a tree, or the "hinge graph" of the polygonal linkage is a tree (the vertices in the *hinge graph* are the polygons in $\mathscr{P}$, and edges represent a hinge between two polygons). Our main result is that realizability remains NP-hard when restricted to simply connected structures.

Theorem 2. *It is NP-Hard to decide whether a polygonal linkage whose hinge graph is a tree can be realized (both with and without orientation).*

Theorem 3. *It is NP-Hard to decide whether a given tree (resp., plane tree) with positive vertex weights is the contact graph (resp., ordered contact graph) of a disk arrangements with specified radii.*

The unoriented versions, where the underlying graph (hinge graph or contact graph) is a tree can easily be handled with the logic engine method (Section **??**). We prove Theorem 2 for *oriented* realizations with a reduction from PLANAR-3SAT (Section **??**), and then reduce the realizability of ordered contact trees to the oriented realization of polygonal linkages by simulating polygons with arrangements of disks (Section **??**).

Bibliography

[1] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Comput. Geom*, 9:3–24, 1998.

[2] R. Connelly, E. D. Demaine, M. L. Demaine, S. P. Fekete, S. Langerman, J. S. B. Mitchell, A. Ribó, and G. Rote. Locked and unlocked chains of planar shapes. *Discrete Comput. Geom*, 44:439–462, 2010.

[3] R. Connelly, E. D. Demaine, and G. Rote. Straightening polygonal arcs and convexifying polygonal cycles. *Discrete Comput. Geom*, 30:205–239, 2003.

[4] P. Eades and N. C. Wormald. Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28:111–134, 1990.

[5] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education, 2006.

[6] S.S. Skiena. *The Algorithm Design Manual*. Springer, 2009.

[7] I. Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete Comput. Geom*, 34:587–635, 2005.