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

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

We consider four decision problems surrounding graph theory and geometry. The graph theory based problems involve polygonal linkages and the geometry based problems involve something called a contact graph of disks. In each problem, we decide whether a polygonal linkage or contact graph has a certain realization in the plane.

This thesis first presents preliminary information needed to pose our four problems, then we formally pose each problem and then provide the hardness results in all four cases. We show that all four problems are intractable, or NP hard (see definition below).

1.1 Graphs

A *graph* is an ordered pair $G = (V, E)$ comprising of a set of vertices V and a set of edges E. An edge is a two element subset of V. Note that with this definition of an edge it is not possible to have one element subset of V as an edge (sometimes referred to as a self-adjacent edge or loop). The *degree* of a vertex v is the number of edges that v is an element of. Vertices are said to be *adjacent* if they form an edge in E. *Neighbors* of a vertex v are the vertices adjecent to v. Edges are said to be adjacent if they share a vertex.

A *simple graph* has no self-adjacent vertices. In this thesis every graph is a simple graph. 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. If $G' = (V', E')$ is a graph such that $V' \subset V$ and $E' \subset E$, then G' is a *subgraph* of G.

To formally show when two graphs are the same, we use the concept of graph isomorphism. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijective function $f : V_1 \mapsto V_2$ such that for any two vertices $u, v \in V_1$, we have $\{u, v\} \in E_1$ if and only if $(f(u), f(v)) \in E_2$. See an example in Table 1.1 and Figure 1.1.

Graph	Vertices	Edges
G_1	$\{a, b, c, d, e\}$	$\{(a, b), (b, c), (c, d), (d, e), (e, a)\}$
G_2	$\{1, 2, 3, 4, 5\}$	$\{(1, 2), (2, 3), (3, 4), (4, 5), (5, 1)\}$

Table 1.1: Two graphs that are isomorphic with the alphabetical isomorphism $f(a) = 1$, $f(b) = 2$, $f(c) = 3$, $f(d) = 4$, $f(e) = 5$.

Figure 1.1: This figure depicts the graph isomorphism shown in Table 1.1 between V_1 and V_2.

To visualize a graph, G, we create a drawing Γ, of G. The *drawing* of a graph $G = (V, E)$ is 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 continuous, injective mapping $c_{u,v} : [0,1] \mapsto \mathbb{R}^2$ such that $c_{u,v}(0) = \Pi(u)$, $c_{u,v}(1) = \Pi(v)$, and the curve $c_{u,v}$ does not pass through any other vertex in V. In this thesis, we will work with straight line drawings and orthogonal drawings. Straight line drawings have mappings $c_{u,v}$ that are straight line segments. Orthogonal drawings have mappings $c_{u,v}$ which are a sequence of alternating horizontal line segments and vertical line segments. For orthogonal drawings, the endpoint of one line segment is the starting point of the next line segment, i.e. every $c_{u,v}$ is piecewise continuous. The endpoints of the line segments of $c_{u,v}$ that are not $\Pi(u)$ or $\Pi(v)$ are called *bends*.

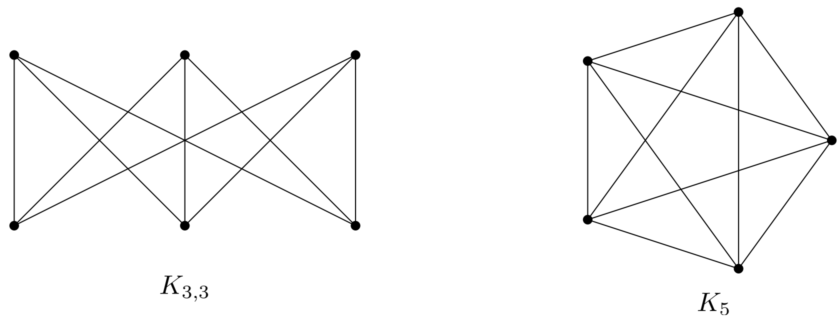


Figure 1.2: The K_5 and $K_{3,3}$ drawn in the plane.

Kuratowski's theorem characterizes finite planar graphs. A finite graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$ [**?**]. Figure 1.2 shows a drawing of K_5 and $K_{3,3}$. Two edges in a drawing *cross* if they have a common interior point. The *crossing number* of a graph is the smallest number of edge crossings for a graph over all drawings. A drawing is said to be *planar* if no two distinct edges cross [**?**]. A planar drawing is also called an *embedding*. Two embeddings of a graph G are *equivalent* if for every vertex the counter-clockwise order of neighbors are the same.A combinatorial *embedding* is a planar drawing with a corresponding counter-clockwise order of the neighbors of each vertex. An orientation preserving rigid transformation (i.e., rotation and translation) map an embedding to

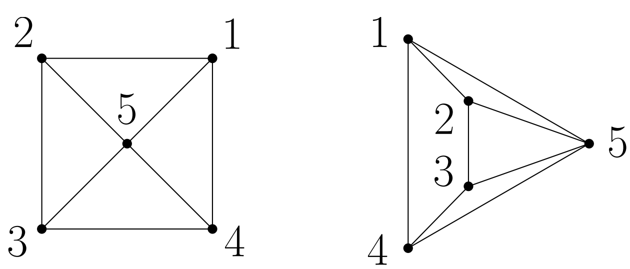


Figure 1.3: Here is a wheel graph, W_5, in two separate drawings with the same counterclockwise ordering of neighbors for each vertex.

an equivalent embedding. Reflections reverse the counter-clockwise order around each vertex.

Figure 1.3 depicts two different drawings of the wheel graph W_5.The drawings have the followings counterclockwise order of neighbors for each vertex: Referencing Table 1.2 and Figure 1.3, we realize that the two drawings of W_5 are equivalent.

Vertex	Left & Middle Drawing	Right Drawing
1	$(2,5,4)$	$(4,5,2)$
2	$(3,5,1)$	$(1,5,3)$
3	$(2,4,5)$	$(5,4,2)$
4	$(1,5,3)$	$(3,5,1)$
5	$(2,3,4,1)$	$(4,3,2,1)$

Table 1.2: A table showing the counter-clockwise circular ordering of neighbors for the left and right drawing in Figure 1.3. Note that the permutation cycles are equivalent for the right and left drawings.

1.1.1 Trees

A *path* is a sequence of vertices in which every two consecutive vertices are connected by an edge.

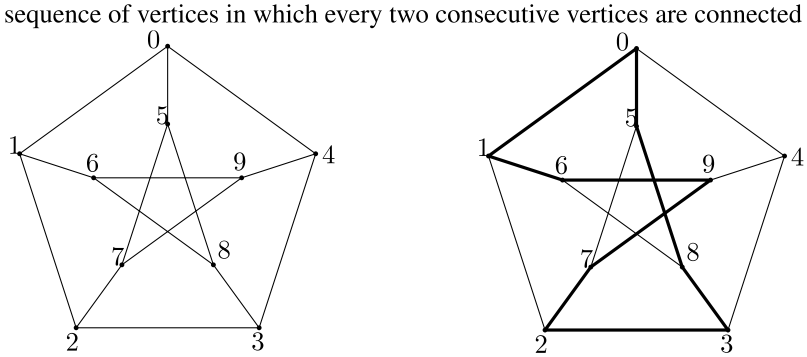


Figure 1.4: An embedding of the Peterson graph with a simple cycle of (2,7,9,6,1,0,5,8,3).

A *simple cycle* of a graph is a sequence, $(v_1, v_2, \ldots, v_{t-1}, v_t)$, of distinct vertices such that every two consecutive vertices are connected by an edge, and the last vertex, v_t, connects to v_1 (see Figure 1.4). A graph is *connected* if for any two vertices, there exists a path between the two points. A *tree* is a graph that has no simple cycles and is connected (see Figure 1.5). Every tree is planar. A *forest* is a disjoint union of trees.

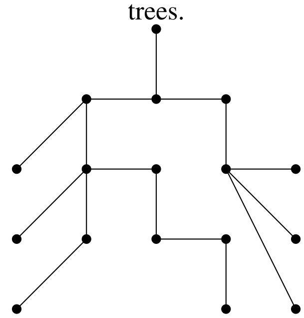


Figure 1.5: An example of a tree.

An *ordered tree* is a tree T together with a cyclic order of the neighbors for each vertex (see Figure 1.6).

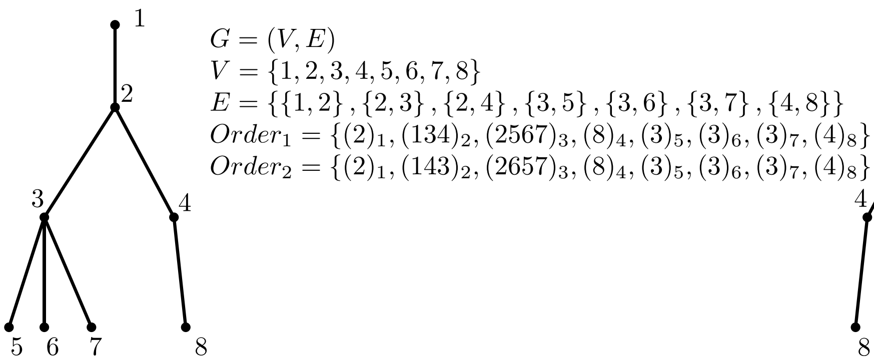


Figure 1.6: A tree with two embeddings with different cyclic orderings around vertices.

Embeddings of ordered trees are combinatorially equivalent if for each node the counter-clockwise ordering of adjacent nodes are the same.

1.2 Linkages

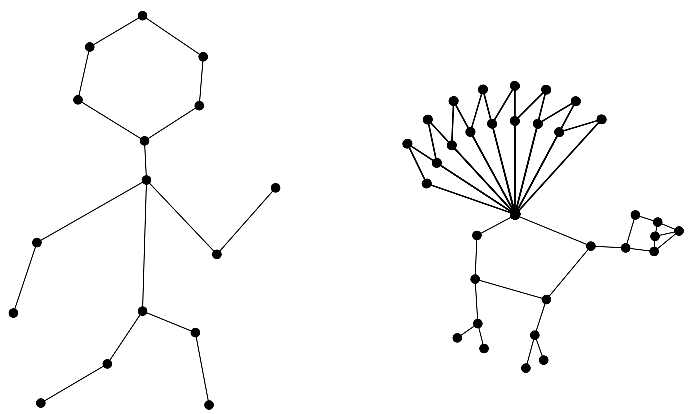


Figure 1.7: Here are skeleton drawings of a human and a turkey. When animating skeletons, one tends to make sure that the lengths of the skeleton segments are kept the same length throught the animation. Otherwise, the animation may depart from what is ideally understood of skeletal motions.

]

When graph drawings model physical objects, other qualities about the graph can be contextualized in a geometric sense. Distance, angular relationships and other geometric qualities may be relevant. In any drawing, edges have length, angles formed by adjacent edges, and so on. In this thesis we are interested in the inverse problem where we would like to embed a graph with specific geometric properties, for example, an embedding with specified edge lengths. This motivates the following definition. A *length assignment* of a graph $G = (V, E)$ is a function $\ell : E \mapsto \mathbb{R}^+$. If $\ell(e)$ is the length of an edge e, $\ell(e)$ must be strictly positive in a drawing, otherwise it may result in two distinct vertices with the same coordinates. Similar to combinatorial embeddings which is an equivalence class of embeddings of the same counter-clockwise order of vertices, we can also define an equivalence class of drawings with the same length assignment. A *linkage* is a graph $G = (V, E)$ with a length assignment $\ell : E \mapsto \mathbb{R}^+$ (e.g., see Figure 1.7).

1.3 Polygonal Linkages

A generalization of linkages is a polygonal linkage where edges of given lengths are replaced with rigid polygons. Formally, a *polygonal linkage* is an ordered pair $(\mathcal{P}, \mathcal{H})$ where $\mathcal{P}$ is a finite set of polygons and $\mathcal{H}$ is a finite set of hinges; a *hinge* $h \in \mathcal{H}$ corresponds to two or more points on the boundary of distinct polygons in $\mathcal{P}$. A *realization of a polygonal linkage* is an interior-disjoint placement of congruent copies of the polygons in $\mathcal{P}$ such that the copies of a hinge are mapped to the same point (e.g., Figure 1.8). A

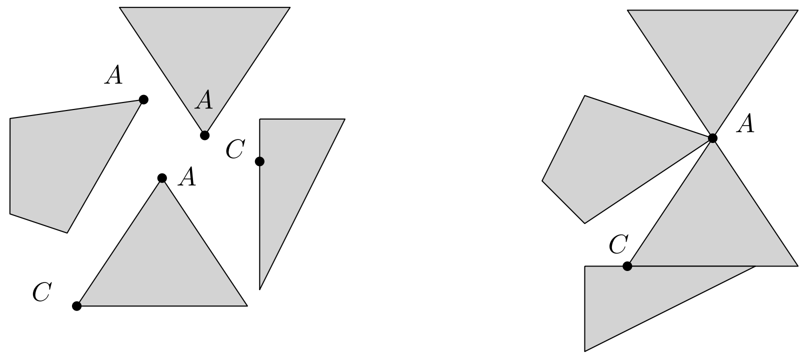


Figure 1.8: (a) A polygonal linkage with a non-convex polygon and two hinge points corresponding to three polygons. Note that hinge points correspond to two distinct polygons.(b) Illustrating that two hinge points can correspond to the same boundary point of a polygon.

realization of a polygonal linkage with fixed orientation is a realization in which each polygon is translated and rotated copy of a polygon in $\mathcal{P}$; at each hinge the incident polygons are in a given counter-clockwise order (refer to Figure 1.9).

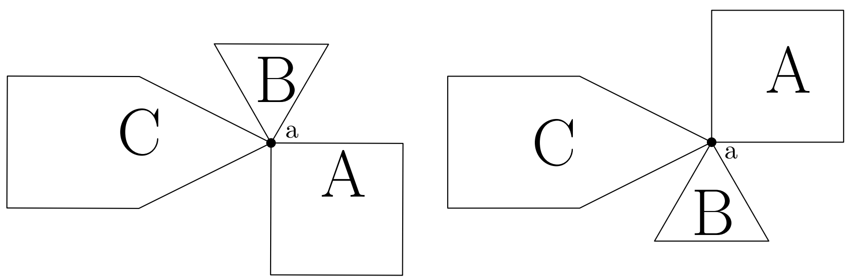


Figure 1.9: Two realizations of the same polygonal linkage with that differ in the counter-clockwise order of polygons around vertex a.

]

Note that oriented polygonal linkage realizations do not allow for reflection transformations of polygons in $\mathcal{P}$.

These two realization types allow one to pose two different problems, the realizability problem for polygonal linkages and the realizability problem for polygonal linkages with fixed orientation:

Problem 1 (Realizibility Problem for Polygonal Linkages). Given a polygonal linkage, does it have a realization?

Problem 2 (Realizibility Problem for Polygonal Linkages with Fixed Orientation). Given a polygonal linkage with fixed orientation, does it have a realization?

Not every polygonal linkage has a realization. Consider the 7 congruent copies of an equilateral triange with a common hinge point in Figure 1.10. To show it does not have a realization, suppose it is realizable. Each angle of every triangle is $\frac{\pi}{3}$ radians. The sum of 7 angles formed by the triangles is $\frac{7\pi}{3} > 2\pi$. The total radian measure around A is 2π. The contradiction is that the sum of 7 angles formed by the triangles in an interior disjoint placement is $\frac{7\pi}{3}$. The polygonal linkage of Figure 1.10 would overlap itself and does not have a realization.

There are polygonal linkages that admit realizations but every realization requires rotation. Figure 1.11

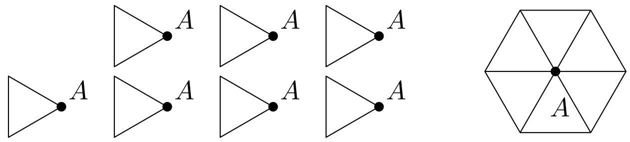


Figure 1.10: Here we have 7 congruent copies of an equilateral triangle with a hinge point of A. The polygonal linkage is not realizable. The best we can realize is at most 6 congruent copies of an equilateral triangle with the hinge point of A in the plane.

show the congruent copies of the polygons A, B, C, and D in two different configurations, the far right is a realization, the middle fails to be a realization because of the interiors of B and D intersecting and the left showing the polygons in $\mathcal{P}$. In fact, this polygonal linkage cannot admit a realization with fixed orientation. Indeed this polygonal linkage cannot satisfy Problem 2, suppose there is a realization with fixed orientation. Without loss of generality, fix the placement of C. A, B, and D have unique placement around triangle C. In this placement C and D overlap. Figure 1.11, satisfies Problem 1 but not Problem 2. The far right is a

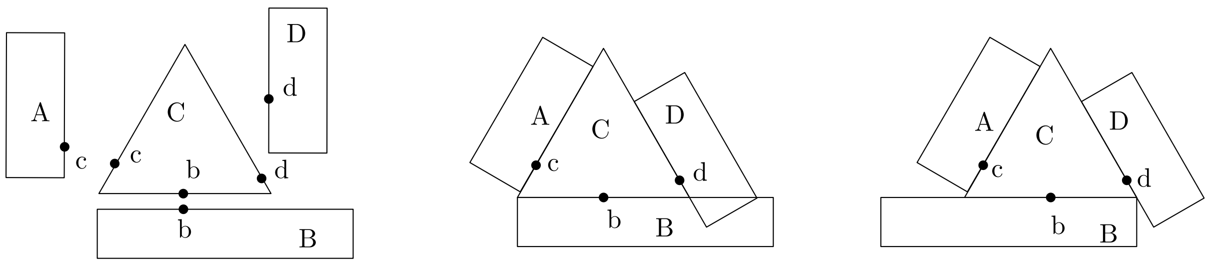


Figure 1.11: This example shows yet another example where two realizations of the same polygonal linkage. One realization where there is an intersection and another where there isn't an intersection.

realization but with polygon B reflected.

Figure 1.11 does not quite get at the heart of the challenge with Problem 1 because the counter-clockwise order of the polygons around hinges is not considered.

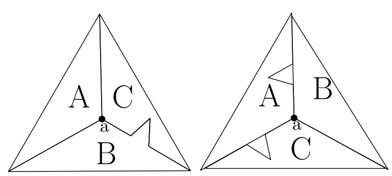


Figure 1.12: Here we have two realizations of a polygonal linkage with two different counter-clockwise order (C,B,A) and (B,C,A) respectively. Note that the placement with ordering (B,C,A) has an overlap.

Figure 1.12 shows three polygons with a common hinge. In the counter-clockwise order (A, B, C), the polygonal linkage admits a realization whereas in the counter-clockwise order (A, C, B), it does not admit a realization. The examples above show that answers to Problem 1 and 2 could be yes or no; the answer could be negative for various reasons. Sections 2 and 3 of this thesis address the computational complexity of solving Problems 1 and 2. We show that both problems are intractable.

1.3.1 Geometric Dissections

Hilbert's third problem asks: given any two polyhedra of equal volume, is it always possible to cut the first into finitely many polyhedral pieces which can be reassembled to yield the second [**?**]? In three dimensions the answer is no however for two dimensions it is true [**?**].

The Wallace-Bolyai-Gerwien Theorem simply states that two polygons are congruent by dissection iff they have the same area. A *dissection* being a collection of smaller polygons whose interior disjoint union forms a polygon. Hinged dissections of a polygon P is a polygonal linkage that admits a realization that forms P. Demaine et. al. [**?**] showed that any two polygons of the same area have a common hinged dissection where polygonal pieces must hinge together at vertices to form a connected realization and that there exists a continuous motion between the two realizations (refer to section 1.5.3). The was an outstanding problem for many years until 2007.

The Haberdasher Puzzle was proposed in 1902 by Henry Dudeney: can a square and an equilateral triangle of the same area have a common dissection into four pieces?

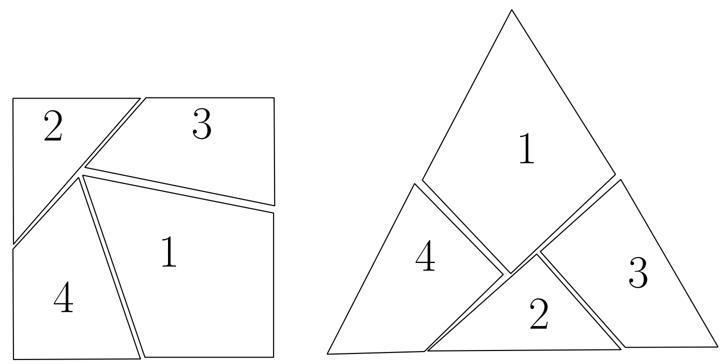


Figure 1.13: The Haberdasher Puzzle was proposed in 1902 and solved in 1903 by Henry Dudeny. The dissection is for polygons that forms a square and equilateral traingle.

Geometric dissections are closely related to polygonal linkages. Figure 1.14 shows two arrangements of the same polygons to form a hexagon and a square. The polygons are not hinged and are arranged in differing order. The polygons are merely tiled together to form the hexagon and square.

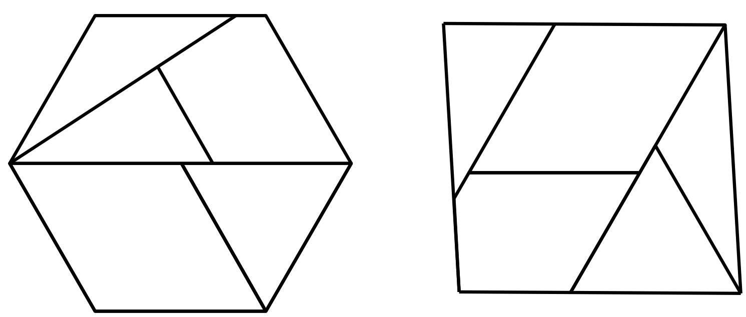


Figure 1.14: Two configurations of polygonal linkage where the polygons touch on boundary segments instead of hinges. These two realizations of the polygonal linkage are invalid to our definitions.

Figure 1.15, shows the Haberdasher problem with hinges. This makes the Haberdasher problem as a type of polygonal linkage where the polygons are free to move about their hinge points and take the form of a triangle or square.

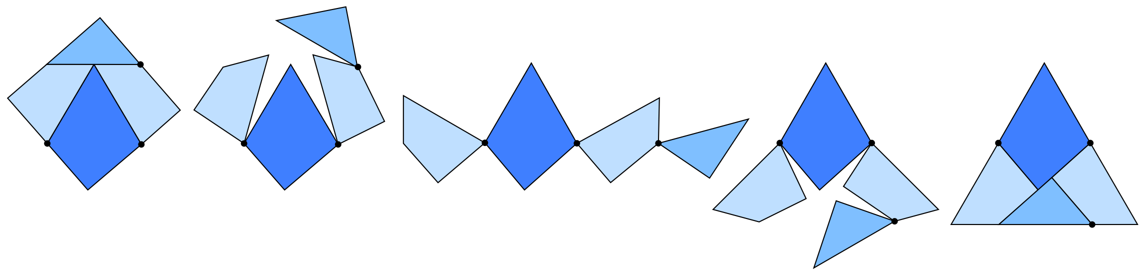


Figure 1.15: This shows the Haberdasher problem in the form of polygonal linkage [**?**]. This is a classic example of two polygons of equal area that have a common hinged dissection.

1.4 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).

Figure 1.16: This example represents a disk arrangement and its contact graph.

A *contact graph* $G = (V, E)$ corresponding to a given 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, $d_j \in D$ (see Figure 1.16). 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. 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.

Koebe's theorem states that for every planar graph G, there exists a planar disk arrangement whose contact graph is G [**?**]. This motivates the question of whether a planar graph G is a contact graph of a disk arrangement with given radii. The radii can be given by a weight function. Let $\omega : V \mapsto \mathbb{R}^+$ be the *weight function*. ω assigns a weight to each vertex in V. Let $\Pi : V \mapsto \mathbb{R}^2$ be that planar mapping of vertices.

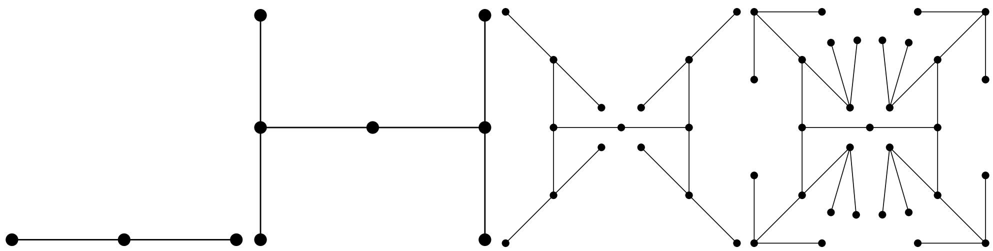
For planar graphs with positive weighted vertices, we pose two realizability problems:

Problem 3 (Unordered Realizibility Problem for a Contact Graph). Given a planar graph with positive weighted vertices, is it a contact graph of some disk arrangement where the radii equal the vertex weights?

Problem 4 (Ordered Realizibility Problem for a Contact Graph). Given a planar graph with positive weighted vertices and a combinatorial embedding, is it a contact graph of some disk arrangement where the radii equal the vertex weights and the counter-clockwise order of neighbors of each disk is specified by the combinatorial embedding?

An instance of Problem 3 is shown in Figure 1.16 where the cycle graph C_5 is the contact graph of unit disks. It is not difficult to see that there exists a planar graph with positive weights with no realizable disk arrangement. Consider the a star graph with 6 leafs, each vertex with unit weight. In any realization, the angle between two consecutive edges must be greater than $\frac{\pi}{3}$. The sum of 6 angles is 2π however, the sum of 6 consecutive angles is greater than 2π. The contradiction shows that no realization is possible (refer to Figure **??**). Note that with the wheel graph W_7 is realizable as a contact graph of unit disks.

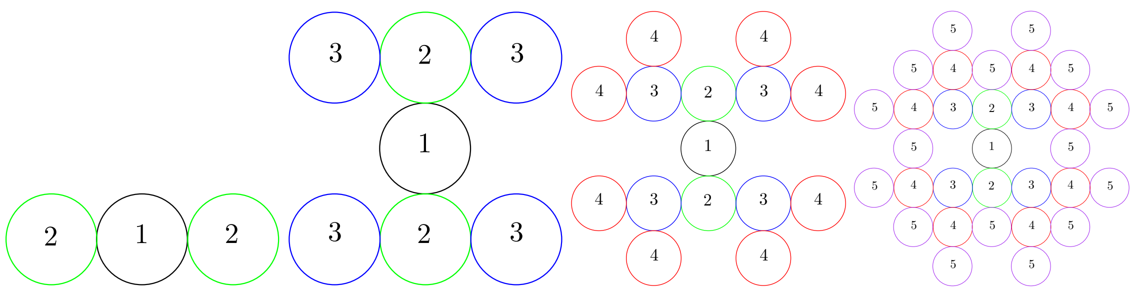
Every path with arbitrary positive radii is realizable as a contact graph, place the vertices on a line. We show that not all binary trees are realizable, even with unit disks.Consider the balanced binary trees of depth i $\{T_i\}_{i=1}^{\infty}$ with unit weights on the vertices (see Figure 1.17). These trees are not realizable for sufficiently large i. Let i be a positive integer and suppose that T_i is a contact graph of unit disks. The balanced binary



(a) Binary tree T_2. (b) Binary tree T_3. (c) Binary tree T_4. (d) Binary tree T_5.

Figure 1.17: We show the linkages T_2 through T_5 with distance 2 between adjacent vertices.

tree T_i has $2^i - 1$ vertices. The total area of the disks is $\left(2^i - 1\right)^2 \cdot \pi$. We now derive an upper bound for this area. Suppose the disk corresponding to the root of the tree is centered at the origin. The centers of the disks at level j are at a distance at most $2 \cdot (j-1)$ away from the origin. The centers of all disks are at distance at most $2 \cdot (i-1)$ away from the origin. All unit disks are contained in a disk of radius $2i - 1$ centered at the origin. The total area of the disks is at most $(2i-1)^2$. A upper bound of the total area of the disk arrangement is the area of the bounding box of the disks.



(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 1.18: For $i = 2, 3, 4, 5$ the tree T_i is a contact graph of unit disks.

Figure **??** shows the first four non-trivial trees as a contact graph of unit disks. 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 $i \geq 8$ we have a contradiciton.

There are instances where a planar graph with weights admits a realization but the cyclic order of neighbors may not be the same as the combinatorial embedding. Define G as follows: start with a star centered at C and with 6 leafs, A_1 through A_6; attach two leafs, B_1 and B_2, to A_1 and A_2 respectively (see Figure 1.19). Let the weight of C be $1 + \epsilon$ for sufficiently small $\epsilon > 0$. The neighbors of C have unit weight. The weights of the two leaves have weight $\frac{1}{\epsilon}$. The right of Figure 1.19) shows a realization where A_1 and A_2 are in opposite position of the counter-clockwise order around C. If A_1 and A_2 are required to be consecutive in the counter-clockwise order around C, there is no realization.

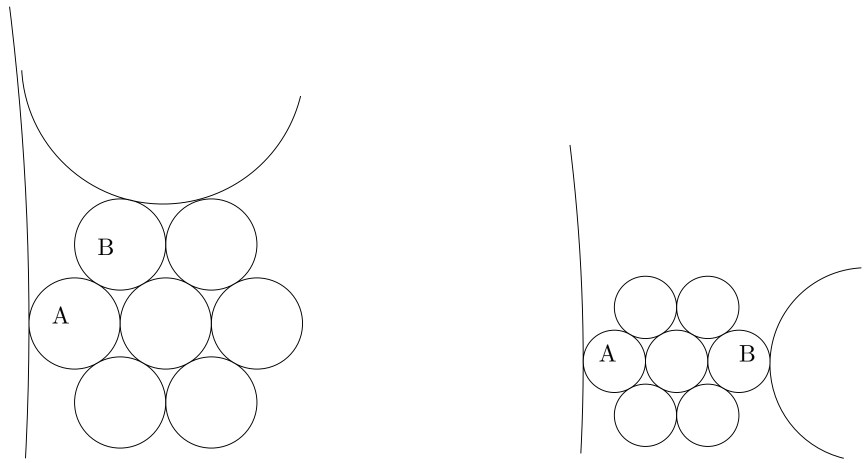


Figure 1.19: Consider these two ordered disk arrangements where A and B are in the concentric rings of disks. The large disks are in contact to A and B respectively. If A and B are adjacent, then there is a restriction of how large the size of the disks can be that are attached to them as seen in on the left. Whereas if A and B are not adjacent in this disk arrangment as shown on the right, the size of the kissings disks could be arbitrarily large.

Suppose there is a realization where $A_1, \ldots, A_6$ are in the counter-clockwise order around C (see Figure 1.20). If $\epsilon > 0$ is sufficiently small, then the centers of of $A_1, \ldots, A_6$ are arbitrarily close to the vertices of a regular hexagon. Consider the common tangent lines between A_1 and B_1 and A_2 and B_2. The possible position of tangent line between A_1 and B_1 ranges from the common tangent line of A_1 and A_6 to the common tangent line of A_1 and A_2. Similarly, The possible position of tangent line between A_2 and B_2 ranges from the common tangent line of A_2 and A_3 to the common tangent line of A_1 and A_2. In any position, the common tangent lines between A_1 and B_1 and A_2 and B_2 intersect. If $\frac{1}{\epsilon}$ is sufficiently large, then the disks D_1 and D_2 also intersect. This contradicts that there is a realization.

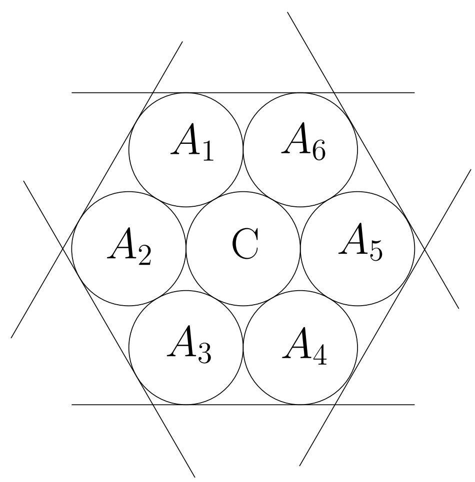


Figure 1.20: This example represents a disk arrangement and its contact graph.

Figure 1.19 shows how an ordered contact graph may not be realizable. On the left, it shows a limitation on the weights of the disks that are in contact with disks A and B. On the right, the figure shows the order where A and B are on opposing ends of the ring of disks and can allow of arbitrary size of wieghted disks in contact with A and B.

1.5 Configuration Spaces

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

Alexander Calder, 1933

Recall Figure 1.15 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.

1.5.1 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. The mapping Π uniquely determines each edge. An edge $\{u, v\} \in E$, is mapped to 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. Let $\mathcal{D}_G$ be the set of all drawings of the graph G. By labelling the vertices of G, e.g. $v_1, v_2, \ldots, v_k, \ldots, v_n$, we can create a mapping from $\mu : \mathcal{D}_G \mapsto \mathbb{R}^{2|V|}$ where the coordinates of $\Pi(v_k)$ are the $(2k)^{\text{th}}$ and $(2k + 1)^{\text{st}}$ coordinates in $\mathbb{R}^{2|V|}$. $\mu(\Pi)$ is a configuration. The configuration space is the set of $\mu(\Pi)$ for all drawings Π.

1.5.2 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.:

$$P_{(G,\ell)} = \{\, \Pi \in \mathcal{D}_G \,|\forall \{u, v\} \in E, \ell(\{u, v\}) = |\Pi(u) - \Pi(v)|\, \}$$

With respect to $P_{(G,\ell)}$, 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 mapping from $\mu : P_{(G,\ell)} \mapsto \mathbb{R}^{2|V|}$ where the corresponding coordinates of $\Pi(v_k)$ are in the $(2k)^{\text{th}}$ and $(2k + 1)^{st}$ coordinates in $\mathbb{R}^{2|V|}$. The configuration space is $\mu\left(P_{(G,\ell)}\right)$.

Using standard definitions from real analysis, we can begin to pose problems about linkages with respect to a corresponding configuration space. A continuous function $\gamma : [0, 1] \mapsto \mu\left(P_{(G,\ell)}\right)$ is a path from a realization $\gamma(0)$ to another realization $\gamma(1)$. γ can be thought of as an animation of drawings that starts at $\gamma(0)$ and ends at $\gamma(1)$. Any two realizations in the same path-connected component can be animated from one to the other continuously. The Carpenter's Rule states that every realization of a path linkage can be continuously moved (without self-intersection) to any other realization [**?**, **?**]. To ask if $\mu\left(P_{(G,\ell)}\right)$ is a connected space, is to ask if $\mu(P)$ is connected in $\mathbb{R}^{2|V|}$. In other words, the realization space of such a linkage is always path-connected.

1.5.3 Configuration Spaces of Polygonal Linkages

The placement of a polygon is described by an isometry of Euclidian plane. An isometry is a composition of a translation, a rotation, and a possible reflection. As such, the isometry can be descibed as three parameters: (a, b, θ) where (a, b) is a translation vector; if $\theta \geq 0$, it describes a counter-clockwise rotation by θ and if $\theta < 0$, it descibes a reflection in the x-axis followed by a counter-clockwise rotation θ. For m polygons there will be $3m$ parameters. Recall a realization of a polygonal linkage is an interior-disjoint placement of congruent copies of the polygons in $\mathcal{P}$ such that the copies of a hinge are mapped to the same point (e.g., Figure 1.8). First consider the set of all realizations for the polygonal linkage $(\mathcal{P}, \mathcal{H})$ and call it P.

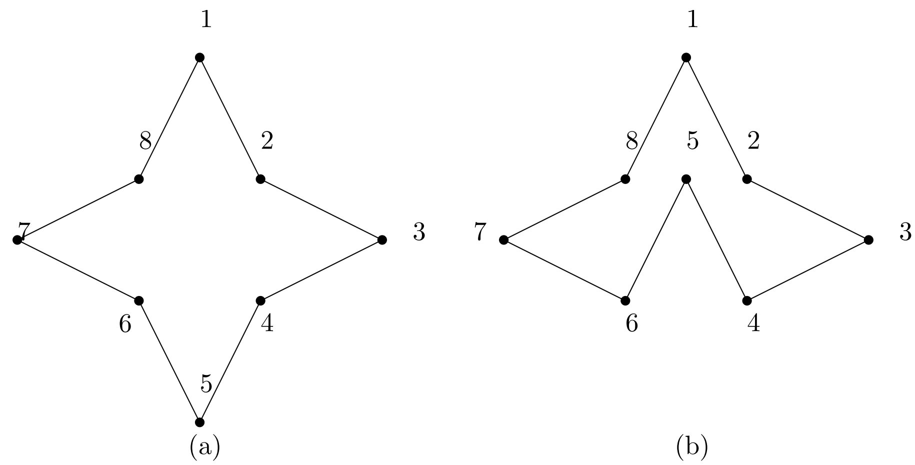


Figure 1.21: (a) and (b) show a linkage in two embeddings. Any realization of a path can be continuously moved without self-intersection to any other realizations.

$\mu : P \mapsto \mathbb{R}^{3m}$ where m is the number of polygons in $\mathcal{P}$ is the configuration space function and the configuration space is the set $\mu(P)$.

1.5.4 Configuration Spaces of Disk Arrangements

rewrite where theta is not needed. disks are symmetric and rotation does not help. The placement of a polygon is described by an isometry of Euclidian plane. An isometry is a composition of a translation, a rotation, and a possible reflection. As such, the isometry can be descibed as three parameters: (a, b, θ) where (a, b) is a translation vector; if $\theta \geq 0$, it describes a counter-clockwise rotation by θ and if $\theta < 0$, it descibes a reflection in the x-axis followed by a counter-clockwise rotation θ. For m polygons there will be $3m$ parameters. Recall a realization of a polygonal linkage is an interior-disjoint placement of congruent copies of the polygons in $\mathcal{P}$ such that the copies of a contact are mapped to the same point (e.g., Figure 1.8). First consider the set of all realizations for the polygonal linkage $(\mathcal{P}, \mathcal{H})$ and call it P. $\mu : P \mapsto \mathbb{R}^{3m}$ where m is the number of polygons in $\mathcal{P}$ is the configuration space function and the configuration space is the set $\mu(P)$.

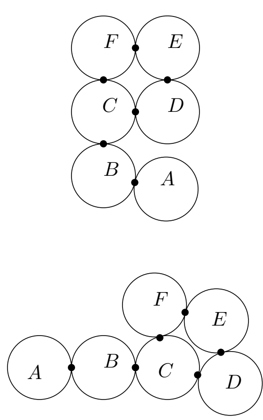


Figure 1.22: An example of a disk arrangement where A and B have a large range of freedom to move around. C, D, E, and F are limited in their range of motion to due to their contact points.

Consider the set of realizations P for a given disk arrangement $\mathcal{D} = \{D_i\}_{i=1}^{n}$. For any realization $R \in P$, there exists a corresponding contact graph, C. The configuration spaces of $\mathcal{D}$ 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.

1.6 Algorithm Complexity

Algorithms are a list of instructions executed with a given input. The efficiency of an algorithm can be measured in terms of the amount of resources it uses such as time, memory, and power. Ideally, a desirable algorithm would primarily have a small run time and secondarily utilize a small amount of resources.

The time and space used by an algorithm is measured with units defined by a model of computation. The actual running time of an algorithm depend on a variety of factors for example: the processor, the hardware, the temperature, etc. Mathematical models of computation have been developed to measure running time of algorithms independent of the machine it runs on. One of the oldest and most popular models is the random access machine (RAM) model. RAM measures the unit of space in the number of words used where each word can store an arbitrary integer. In the real RAM, each word can store an arbitrary real number. The units of time is measured in the number of arithmetic operations and number of memory accesses (read or write).

The *running time* of an algorithm on a given input, is the time it takes to terminate.The *worst-case* running time is the largest running time over all inputs of a given size N. It is a function of N, it is usually monotonically increasing function since larger inputs tend to take more time to process. The key parameter of the efficiency of an algorithm is the growth rate of its worst case running time in terms of N. An algorithm is said to be *efficient* if the time needed to perform the list of instructions can be determined from a polynomial. Devising an efficient algorithm for a given problem is often a difficult task.

The growth rate of running times are typically compared upto constant vectors. Let f and g be defined on some subset of $\mathbb{R}$. $f(x) = O\left(g(x)\right)$ if and only if there exists a constant M and x_0 such that

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

for all $x \geq x_0$

1.6.1 Complexity Classes

Problems can be categorized by their running times. Each algorithm computes a function $f(I)$ on an input I, however many different algorithms can compute the same function. Algorithms are differentiated by their running times but the function is characterized by the fastest algorithm that can compute it. A problem can be formulated as follows, given input I find $f(I)$.

Problems can be categorized into complexity classes based on the fastest algorithms that solve them. 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 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 $|t| \leq p\left(|s|\right)$ and $B(s,t) =$ 'yes'.

The class of problems which have an efficient certifier is said to be the *nondeterministic polynomial time* class, NP. We continue with the definitions for NP-hard and NP-complete. A problem is NP-hard if every problem in NP can be reduced to it in polynomial time. 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, i.e. Y is reduced in polynomial time to X. A problem is NP-complete if it NP and NP-hard, i.e. NP-complete = $\mathsf{NP} \cap \mathsf{NP}$-hard.

1.6.2 RSA Cryptosystem

Cryptography is the study of secure communication between parties in an untrusted or unsecure communication channel. Cryptography has three primary purposes for secure communications: provide confidentiality, authenticate entities, and verification of data. Modern cryptography is based on hard math problems such as integer factorization, discrete logarithmic problem, and pre-image problems. Hard math problems are not found in P and found in NP. In most forms of modern cryptography, *keys*, are data parameters used to form function outputs for the use in a communication channel. There are three common types of keys in cryptography:

1. A *secret key* is known by certain entities. Typically a secret key is used by entities to encrypt and decrypt data that is communicated over an untrusted channel. Secret keys require a secure channel to exchange between all entities.
2. When there are no means to exchange secret keys, one can use a private and public key scheme. A *public key* is a key that can be shared with any entity, i.e. trusted and untrusted entities can know it. The *private key* is a key that is kept to one entity. It is treated like a secret key and should only be known to that entity.

A *cryptosystem* is a suite of algorithms used to establish secure communication channel between parties. The RSA cryptosystem is the first practical cryptosystem in modern cryptography that allows for encryption of data (confidentiality), authentication of entites, and verify message integrity using just one underlying hard math problem, integer factorization.

RSA is named after its second inventors, Ron Rivest, Adi Shamir, and Leonard Adelmen. These three individuals devised and published the algorithm in 1977. The original inventor of RSA was Clifford Cocks in 1973 however, it was only known to the public since 1997 that Clifford Cocks was the original inventor because his was was classified by Government Communication Headquaters, an intelligence agency of the United Kingdom.

For the RSA cryptosystem, we first want to pose the communication security problem. Suppose we have two entities, Alice and Bob, that wish to communicate over an unsecure channel. Should Alice and Bob agree to using RSA, each entity will have a pair of keys, a private key and a public key. Most implementations of RSA use the following key derivation [**?**]:

1. Let $n = p \cdot q$ where p and q are randomly chosen prime numbers.
2. Choose an integer e such that $1 < e \leq \phi(n)$ and $gcd\,(e, \phi\,(n)) = 1$ where ϕ is the Euler totient function.
3. Let $d \equiv e^{-1} \mod \phi(n)$.

The RSA cryptosystem is based around the following formula:

$$(m^e)^d \mod n = \left(m^d\right)^e \mod n \equiv m \mod n$$

where we have natural numbers e, d, m, and n, such that $m < n$ and $gcd(m, n) = 1$.

If Alice derives a public and private key in the manner described, her public key is (n, e) and her private key is d. Suppose Bob wants to send Alice a message M. Bob will represent M in binary form, m. Alice sends her public key (n, e) to Bob and keeps her private key d secret. If $gcd(m, n) \neq 1$, then Bob modifies m by padding m with additional digits so that m becomes co-prime with n. There are efficient padding schemes to modify m such that m and n are co-prime [**?**]. Bob sends the following value, c, to Alice:

$$c \equiv m^e \mod n$$

c is said to be a ciphertext. Alice can decrypt the ciphertext and recover m by computing:

$$m \equiv c^d \mod n = (m^e)^d \mod n$$

The RSA cryptosystem is based on integer factorization and the RSA problem, given n where $n = p \cdot q$ where p and q are prime numbers, find p and q. For sufficiently large integers, factorization and the RSA problem becomes very difficult. If there were an algorithm that solved integer factorization in polynomial time, then one can also solve the RSA problem as well. Suppose a third party listens into the conversation and knows Alice's public key (n, e) and the ciphertext c. If the the third party can factor $n = p \cdot q$ then the attacker can computer $\phi(n)$ and compute $d \equiv e^{-1} \mod \phi(n)$ using the extended euclidian algorithm. Once they have d, the attacker can compute $m \equiv c^d \mod n$.

Currently, the most efficient factorization algorithm is the general number sieve algorithm [**?**]. It's an exponential running time algorithm. If a polynomial running time algorithm for integer factorization existed, it would allow for compromise of security that is provided to communicating parties by the RSA cryptosystem. It would allow for a reduction of the integer factorization problem, a problem that exists in NP but not P. The algorithm would allow for an adversary to attack RSA [**?**] in polynomial time.

1.7 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. Here are the problem statements for satisfiability:

Problem 5 (Satisfiability Problem (SAT)). Given a boolean formula, is it satisfiable? [**?**]

Brute force is when an algorithm tries all possibilities to see if any formulates a satisfiable solution. It is clear that SAT is decidable in exponential time by testing all possibilities. This is callled a brute force solution. It is not known whether SAT admits a polynomial time solution, that is whether it is in P.

Problem 6 (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, Φ, with n variables and m clauses, we define the *associated graph* $A(\Phi)$ as follows: the vertices correspond to the variables and clauses in Φ. We place an edge in the graph if variable x_i appears in clause C_j.

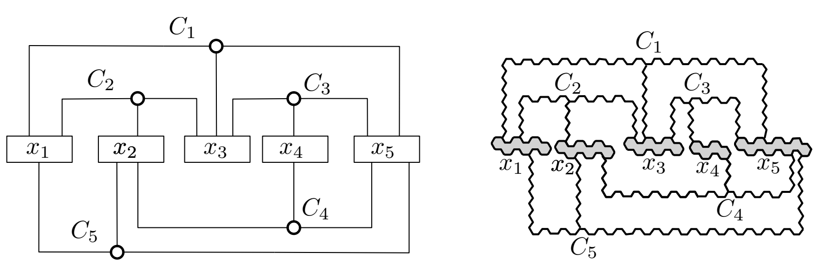


Figure 1.23: Left: the associated graph $A(\Phi)$ for a Boolean formula Φ. Right: the schematic layout of the variable, clause, and transmitter gadgets in the auxiliary construction showing in Section **??**

Problem 7 (Planar 3-SAT). Given a boolean formula Φ in 3-CNF such that its associated graph is planar, decide whether it is satisfiable is a *3-SAT problem*.

Figure 1.23 is associated to a family of boolean formulas. One such associated boolean formula is:

$$(x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_2 \vee x_5)$$

Note that the figure establishes an edge relation between variables and clauses whereas the clauses in boolean formulas do not have variables but literals of variables.

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?

Problems 5—8 are known to be NP-hard and are often used to show other problems are NP-hard as well [**?**, **?**].

1.8 Contribution

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 [**?**, **?**]. 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 [**?**], 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 [**?**, **?**] and the coin graphs in [**?**] 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 $\mathcal{P}$, and edges represent a hinge between two polygons). Our main result is that realizability remains NP-hard when restricted to simply connected structures.

Theorem 1. *It is strongly NP-hard to decide whether a polygonal linkage whose hinge graph is a* ***tree*** *can be realized.*

Theorem 2. *It is strongly NP-hard to decide whether a polygonal linkage whose hinge graph is a* ***tree*** *can be realized with fixed orientation.*

Our proof for Theorem 2 is a reduction from PLANAR-3-SAT (P3SAT): decide whether a given Boolean formula in 3-CNF with a planar associated graph is satisfiable. Our proof for Theorem 1 is a reduction from NOT-ALL-EQUAL-3-SAT (NAE3SAT): decide whether a given Boolean formula in 3-CNF is it satisfiable so that each clause contains a true and a false literal?

Theorem 3. *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 4. *It is NP-Hard to decide whether a given tree (resp., plane tree) with positive vertex weights is the contact graph (resp., 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 1.7). We prove Theorem 3 for *oriented* realizations with a reduction from PLANAR-3SAT (Section 1.3), and then reduce the realizability of ordered contact trees to the oriented realization of polygonal linkages by simulating polygons with arrangements of disks (Section 1.4).

1.9 Related Work and Results

Previous research has established NP-hardness in several easy cases, but realizability for simply connected structures remained open. Polygonal linkages (or body-and-joint frameworks) are a generalization of classical linkages (bar-and-joint frameworks) in rigidity theory. A linkage is a graph $G = (V, E)$ with given edge lengths [**?**]. A realization of a linkage is a (crossing-free) straight-line embedding of G in the plane. Based on ideas developed by Bhatt and Cosmadakis [**?**], who proved that the realizability of linkages is NP-complete on the integer grid, the *logic engine* method [**?**, **?**, **?**, **?**] has become a standard tool for proving NP-hardness in graph drawing. The logic engine is a graph composed of rigid 2-connected components, where two possible realizations of a 2-connected component encode a binary variable.

However, the logic engine method is **not** applicable to problems with fixed embedding or orientation, where the circular order of the neighbors of each vertex is part of the input. Cabello et al. [**?**, **?**] used a significantly more elaborate reduction to show that the realizability of 3-connected linkages (where the orientation is unique by Whitney's theorem [**?**]) is NP-hard. This problem is efficiently decidable, though, for near-triangulations [**?**, **?**].

Note that every *tree* linkage can be realized in $\mathbb{R}^2$ with almost collinear edges. According to the celebrated *Carpenter's Rule Theorem* [**?**, **?**], every realization of a path (or a cycle) linkage can be continuously moved (without self-intersection) to any other realization. In other words, the realization space of such a linkage is always connected. However, there are trees of maximum degree 3 with as few as 8 edges whose realization space is disconnected [**?**]; and deciding whether the realization space of a tree linkage is connected is PSPACE-complete [**?**]. (Earlier, Reif [**?**] showed that it is PSPACE-complete to decide whether a polygonal linkage can be moved from one realization to another among polygonal obstacles in $\mathbb{R}^3$.) Cheong et al. [**?**] consider the "inverse" problems of introducing the minimum number of point obstacles to reduce the configuration space of a polygonal linkage to a unique realization.

Connelly et al. [**?**] showed that the Carpenter's Rule Theorem generalizes to certain polygonal linkages obtained by replacing the edges of a path linkage with special polygons (called *slender adornments*). Our Theorem 1 indicates that if we are allowed to replace the edges of a linkage with arbitrary convex polygons, then deciding whether the realization space is empty or not is already NP-hard.

Recognition problems for intersection graphs of various geometric object have a rich history [**?**]. Breu and Kirkpatrick [**?**] proved that it is NP-hard to decide whether a graph G is the contact graph of unit disks in the plane, i.e., recognizing *coin graphs* is NP-hard; see also [**?**]. Recognizing outerplanar coin graphs is already NP-hard, but decidable in linear time for caterpillars [**?**]. It is also NP-hard to recognize the contact graphs of pseudo-disks [**?**] and disks of bounded radii [**?**] in the plane, and unit disks in higher dimensions [**?**, **?**]. Eades and Wormald [**?**] showed that it is NP-hard to decide whether a given tree is a *subgraph* of a coin graph. All these hardness reductions produce graphs with a large number of cycles, and do not apply to trees. Note that the contact graphs of disks *of arbitrary radii* are exactly the planar graphs (by Koebe's circle packing theorem), and planarity testing is polynomial. Consequently, every tree is the contact graph of disks of *some* radii in the plane. However, deciding whether a given star is realizable as a contact graph of disks of given radii but arbitrary embedding is already NP-hard [**?**].

Schaefer [**?**] proved that deciding whether a graph with given edge lengths can be realized by a straight-line drawing (possibly with crossing edges) has the same complexity as the existential theory of the reals. Both reductions crucially rely on a large number of cycles. Our work is the first to simulate rigid polygons with truly flexible combinatorial structures that have simply connected topology.