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

ABSTRACT

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

By

Clinton Bowen

Master of Science in Applied Mathematics

Chapter 1

Background

In this thesis we consider four such decision problems surrounding graph theory and geometry. The first set of problems involve polygonal linkages and the second set of 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.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. 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). A *simple graph* has no self-adjacent vertices. In this thesis every graph is a simple graph. 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 **??** and Figure 1.21.

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 (**??**) 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 strictly work with straight line drawings

where all $c_{u,v}$ are straight line segments unless specified otherwise. Kuratowski's theorem characterizes finite

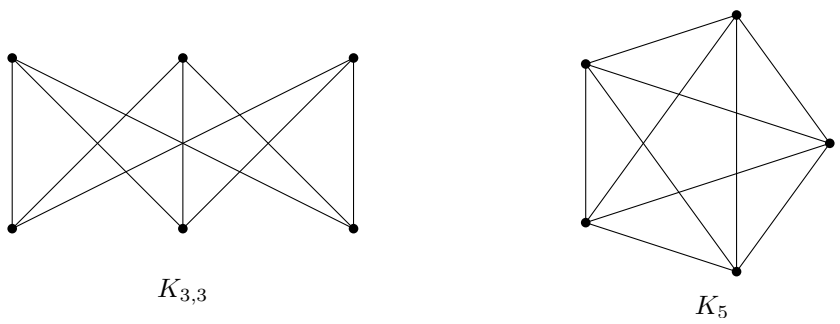


Figure 1.2: The K_5 and $K_{3,3}$ drawn in the plane.

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 $K_{3,3}$ [11]. 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 [3]. 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

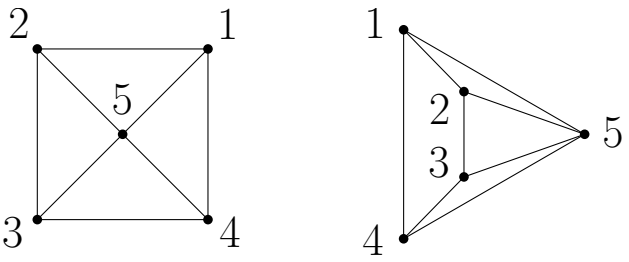


Figure 1.3: Here is a wheel graph, W_5, in two separate drawings with the same counterclockwise ordering of neighbors for each vertex.

and translation) map an embedding to 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 **??** and Figure 1.3, we realize that the two drawings of W_5 are equivalent.

1.1.1 Trees

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

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.

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

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

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.

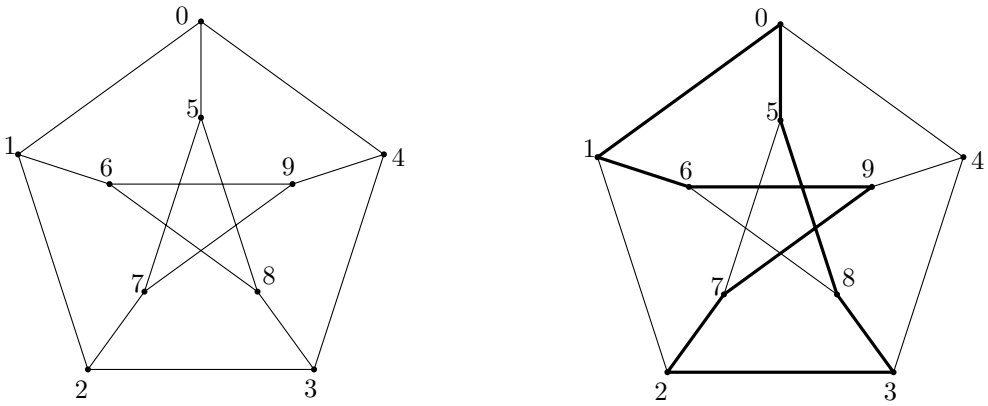


Figure 1.4: An embedding of the Peterson graph with a simple cycle of (2,7,9,6,1,0,5,8,3).

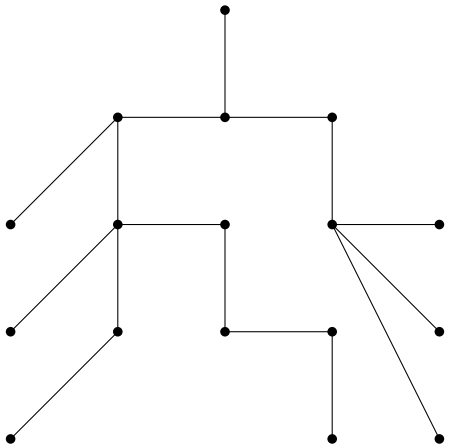


Figure 1.5: An example of a tree.

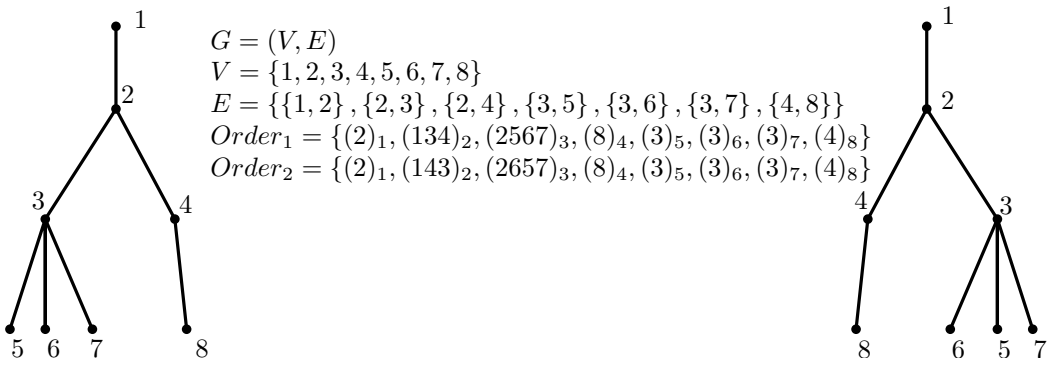


Figure 1.6: A tree with two embeddings with different cyclic orderings around vertices.

1.2 Linkages

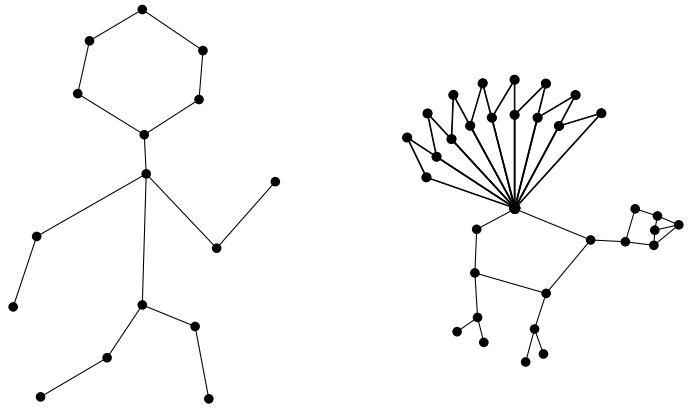


Figure 1.7: Here are skeleton drawings of a human and a turkey. When animating skeletons, one tends to make sure that the lengths of the skeleton segments are kept the same length throught the animation. Otherwise, the animation may depart from what is ideally understood of skeletal motions.

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 **??**).

1.3 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 $(\mathscr{P}, \mathscr{H})$ where $\mathscr{P}$ is a finite set of polygons and $\mathscr{H}$ is a finite set of hinges; a *hinge* $h \in \mathscr{H}$ corresponds to two or more points on the boundary of distinct polygons in $\mathscr{P}$. 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 1.8). A *realization of a polygonal linkage with fixed orientation* is a realization in which each polygon is

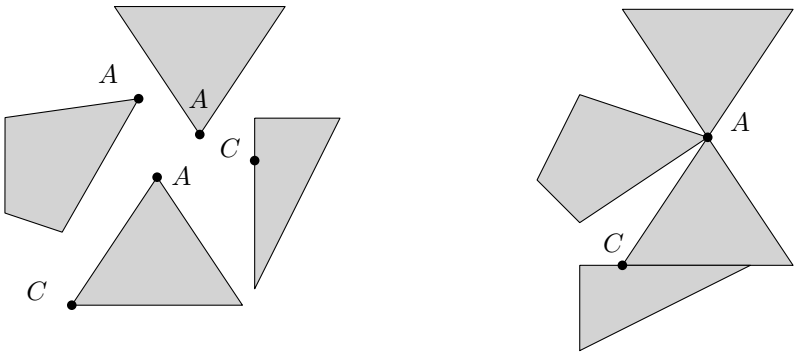


Figure 1.8: (a) A polygonal linkage with a non-convex polygon and two hinge points corresponding to three polygons. Note that hinge points correspond to two distinct polygons.(b) Illustrating that two hinge points can correspond to the same boundary point of a polygon.

translated and rotated copy of a polygon in $\mathscr{P}$; at each hinge the incident polygons are in a given counter-clockwise order (refer to Figure **??**). Note that oriented polygonal linkage realizations do not allow for

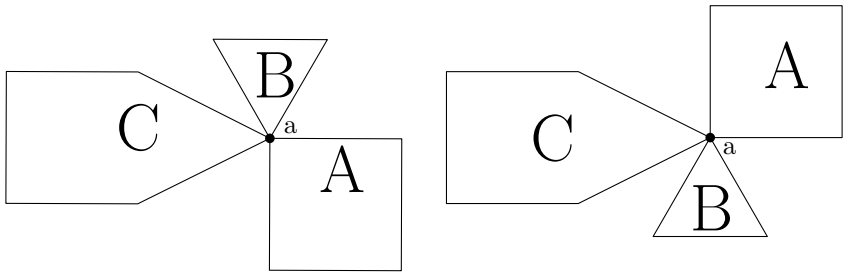


Figure 1.9: Two realizations of the same polygonal linkage with that differ in the counter-clockwise order of polygons around vertex *a*.

reflection transformations of polygons in $\mathscr{P}$.

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

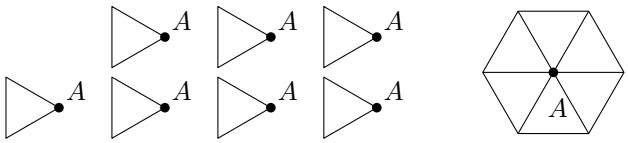


Figure 1.10: Here we have 7 congruent copies of an equilateral triangle with a hinge point of *A*. The polygonal linkage is not realizable. The best we can realize is at most 6 congruent copies of an equilateral triangle with the hinge point of *A* in the plane.

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 **??** 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 $\mathscr{P}$. 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 **??**, satisfies Problem 1 but not Problem 2. The far right is a realization

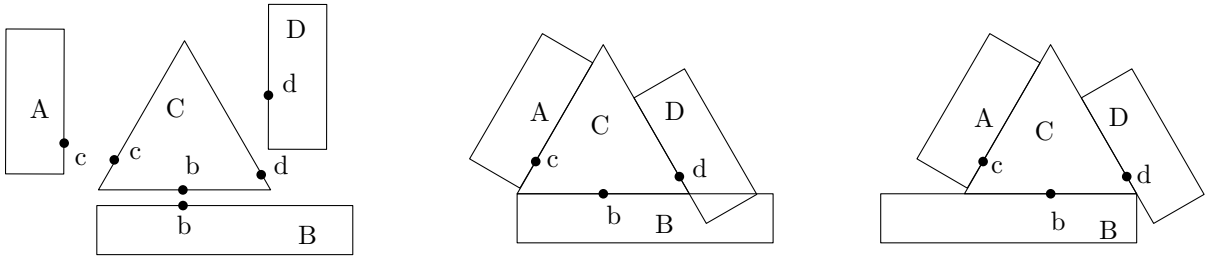


Figure 1.11: This example shows yet another example where two realizations of the same polygonal linkage. One realization where there is an intersection and another where there isn't an intersection.

but with polygon B reflected.

Figure **??** 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.

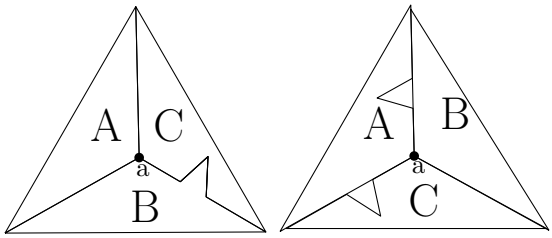


Figure 1.12: Here we have two realizations of a polygonal linkage with two different counter-clockwise order (C,B,A) and (B,C,A) respectively. Note that the placement with ordering (B,C,A) has an overlap.

Figure **??** 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 [2]? In three dimensions the answer is no however for two dimensions it is true [14].

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. [1] 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?

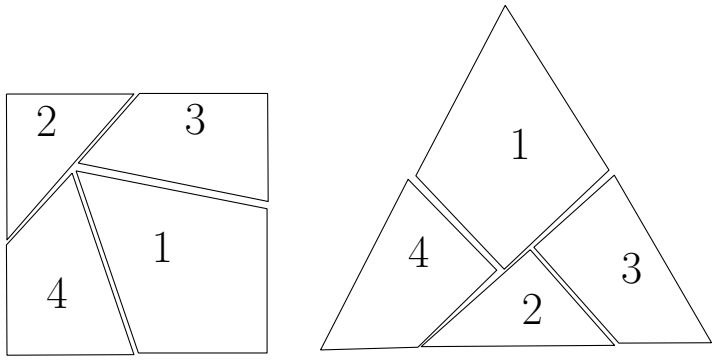


Figure 1.13: The Haberdasher Puzzle was proposed in 1902 and solved in 1903 by Henry Dudeny. The dissection is for polygons that forms a square and equilateral traingle

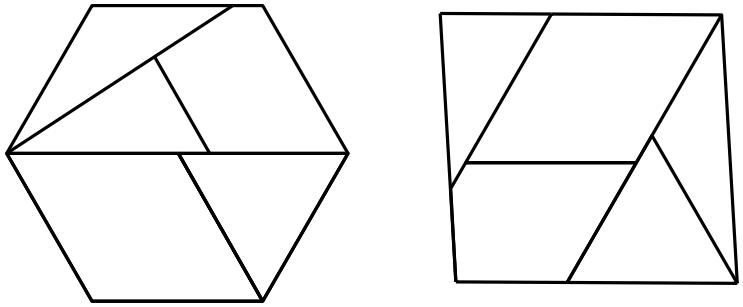


Figure 1.14: Two configurations of polygonal linkage where the polygons touch on boundary segments instead of hinges. These two realizations of the polygonal linkage are invalid to our definitions.

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 1.15, 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.

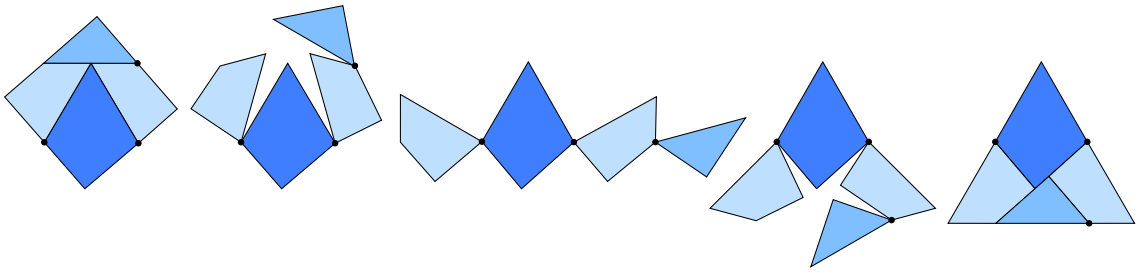


Figure 1.15: This shows the Haberdasher problem in the form of polygonal linkage [1]. This is a classic example of two polygons of equal area that have a common hinged dissection.

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). A *contact graph* $G = (V, E)$ corresponding to a given

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

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 [10]. 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.

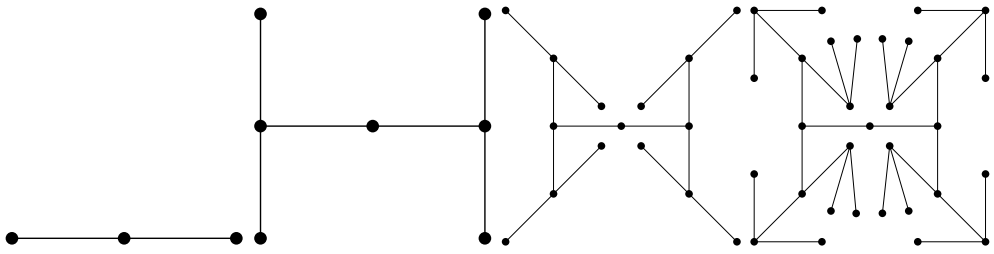
For planar graphs with positive weighted vertices, we pose two realizability problems:

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 1.18). 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



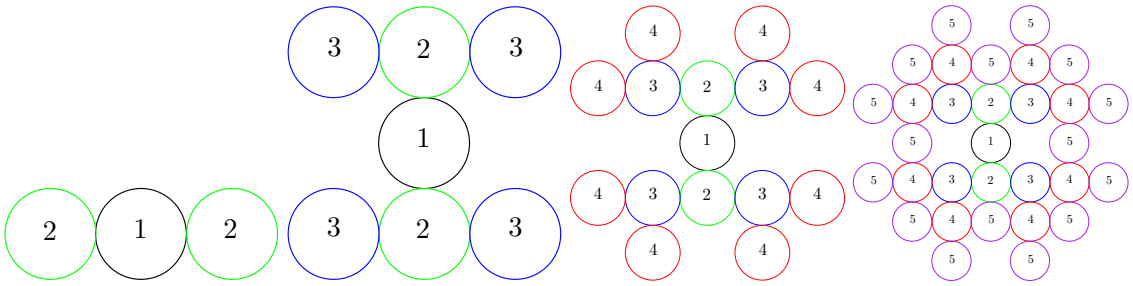
(a) Binary tree T_2. (b) Binary tree T_3. (c) Binary tree T_4. (d) Binary tree T_5.

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. The upper bound of the

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



(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.

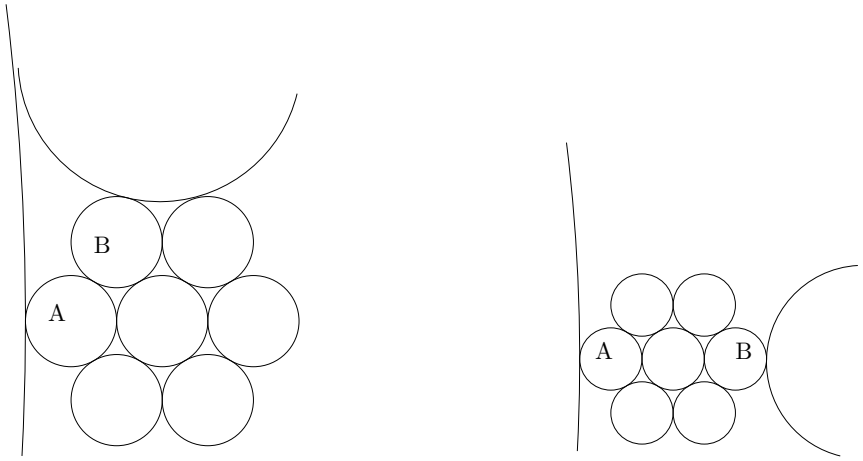


Figure 1.19: Consider these two ordered disk arrangements where A and B are in the concentric rings of disks. The large disks are in contact to A and B respectively. If A and B are adjacent, then there is a restriction of how large the size of the disks can be that are attached to them as seen in on 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.

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 **??**). Let the weight of C be $1+\varepsilon$ for sufficiently small $\varepsilon > 0$. The neighbors of C have unit weight. The weights of the two leaves have weight $\frac{1}{\varepsilon}$/ The right of Figure **??**) 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.

Suppose there is a realization where $A_1,\ldots,A_6$ are in the counter-clockwise order around C (see Figure **??**). If $\varepsilon > 0$ 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 $\frac{1}{\varepsilon}$ is sufficiently large, then the disks D_1 and D_2 also intersect. This contradicts that there is a realization. Figure **??** shows how an ordered contact graph may

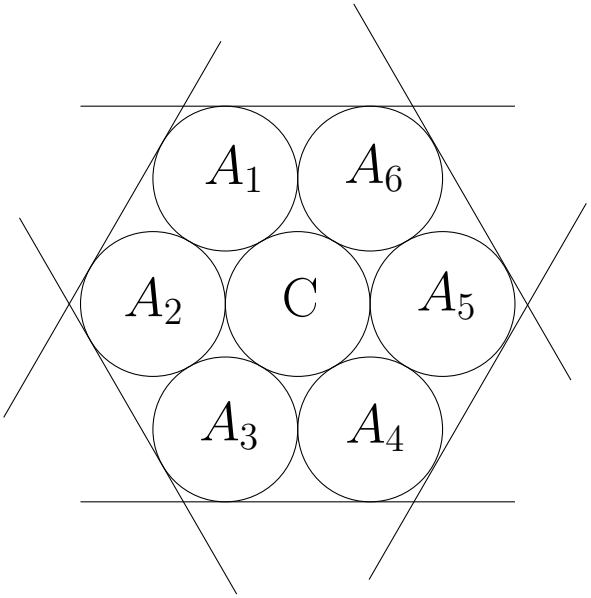


Figure 1.20: This example represents a disk arrangement and its contact graph.

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 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.

1.5.2 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.:

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

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 [7, 13]. In other words, the realization space of such a linkage is always connected.

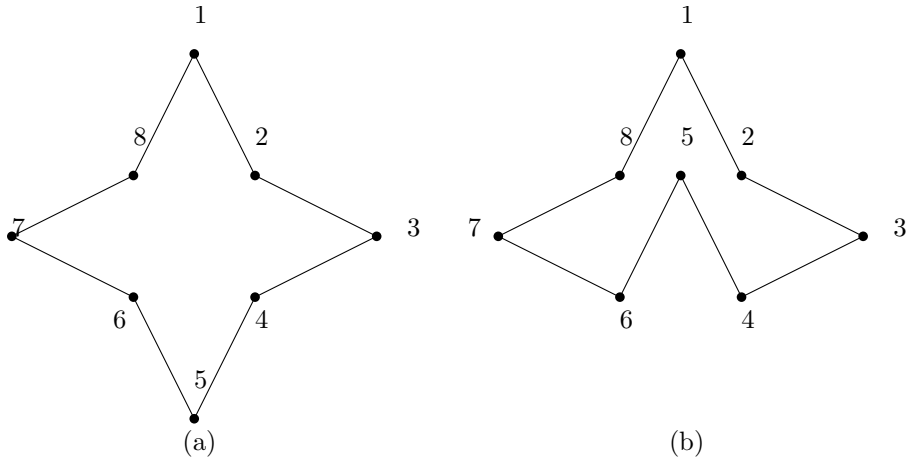


Figure 1.21: (a) and (b) show a linkage in two embeddings. Any realization of a path can be continuously moved without self-intersection to any other realizations.

1.5.3 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 1.8). 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 $\mu(P)$.

1.5.4 Configuration Spaces of Disk Arrangements

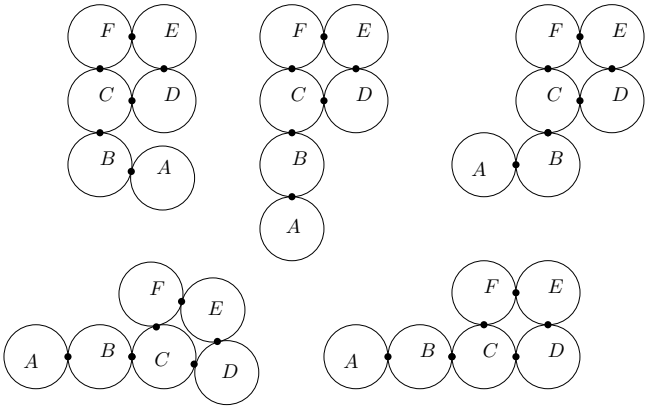


Figure 1.22: An example of a disk arrangement where A and B have a large range of freedom to move around. C, D, E, and F are limited in their range of motion to due to their hinge points.

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.

1.6 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 $\mathbb{R}$. $f(x) = O(g(x))$ 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 $|t| \leq p(|s|)$ and $B(s,t) =$ 'yes'.

The class of problems which have an efficient certifier is said to be the *nondeterministic polynomial time* class, NP. 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 = NP *cap*NP-hard.

1.6.2 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. 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 5 (Optimization of an Independent Set in G). Given a graph G, what is the largest independent set in G?

Problem 6 (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 5, B can be leveraged to solve Problem 6. Likewise, A solve Problem 6 but can be leveraged to solve Problem 5.

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 7 (Satisfiability Problem (SAT)). Given a boolean formula, is it satisfiable? [12]

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 8 (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 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*.

Problem 10 (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 7—10 are known to be NP-hard and are often used to show other problems are NP-hard as well.

1.8 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 [6, 8]. 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 [5], 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 [6, 8] and the coin graphs in [5] 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 strongly NP-hard to decide whether a polygonal linkage whose hinge graph is a **tree** can be realized.*

Theorem 3. *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 3 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 2 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 4. *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 5. *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 **??**). We prove Theorem 4 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 **??**).

Chapter 2

Decidability Problems for Hinged Polygons and Disks

2.1 The Logic Engine

The *logic engine* is a planar, mechanical device that simulates an instance NAE3SAT problem. It was introduced in Bhatt et. al. [4].

2.1.1 Construction of the Logic Engine

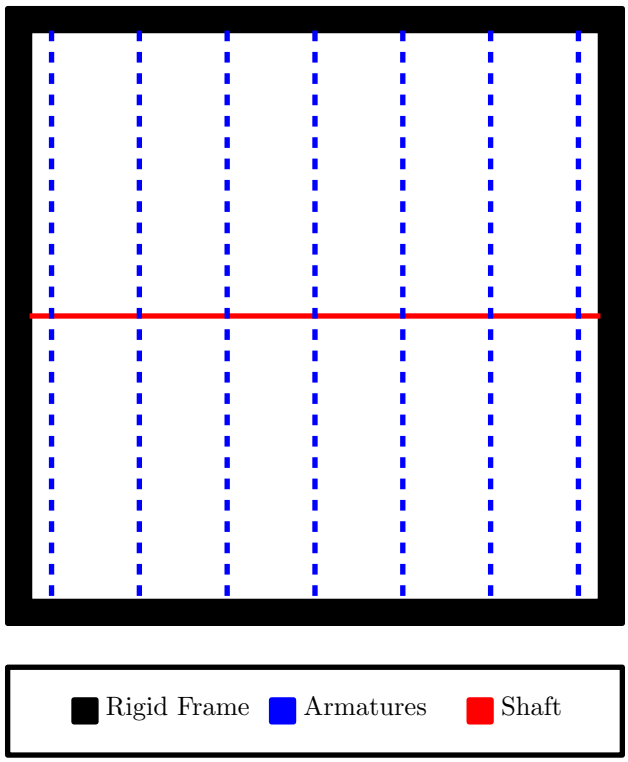


Figure 2.1: A logic engine frame with veritical armatures and a horizontal shaft.

For a given a boolean formula, Φ, in 3-CNF with n variables and m clauses, we construct a logic engine. The logic engine has a *rigid frame* which houses the mechanical components of the the logic engine. The rigid frame is the boundary of which the logic engine can operate within. The *shaft* is a horizontal line segment that is placed at mid-height of the rigid frame. The *armatures* are veritical line segments whose midpoints are on the shaft. Each armature has two orientations with respect to the shaft. There will be some flags on each armature that will be described later. The armatures each have length $2n$ units, the shaft has length m units, and the frame has a height of $2n$ and width of m units.

Each armature corresponds to a variable in Φ. There are two literals for each variable, i.e. the literal x_j and the negated literal $\bar{x}_j$. To describe the flagging arrangement, first partition each armature into $2m$ units, vertical line segments. Label the segments on the j^{th} armature starting from the shaft by $\ell_{j,1},\ldots,\ell_{j,n}$ on one side and $\bar{\ell}_{j,1},\ldots,\bar{\ell}_{j,n}$ on the other side of the shaft. Attach regular triangles, called *flags*, to some of these segments. Each segment is either flagged one or zero flags, i.e. *flagged* or *unflagged*.

1. If the literal x_j is found in clause C_k, then $\ell_{j,k}$ is unflagged.

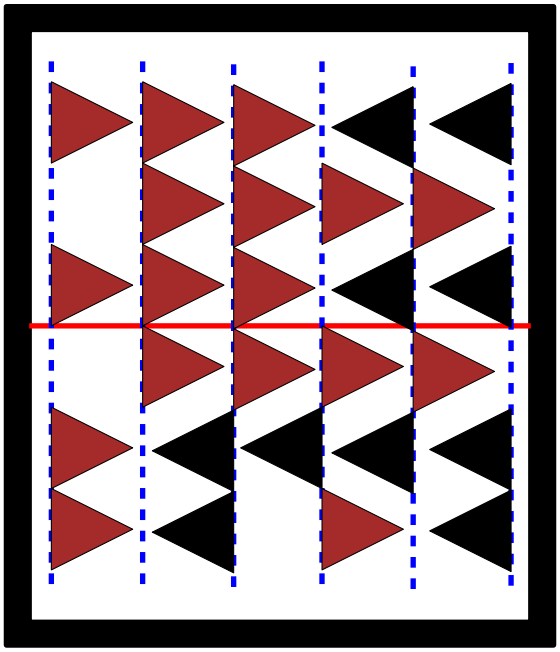
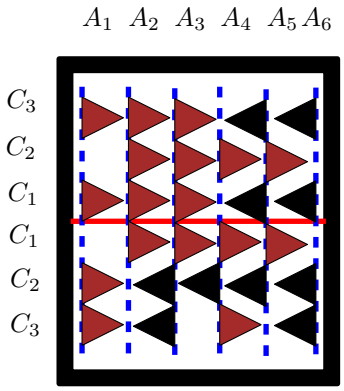


Figure 2.2: A logic engine that corresponds to a boolean formula in NAE3SAT form, Φ. The picture shows the outer rigid frame, the shaft, the armatures that correspond to the variables in Φ, with oriented flags.

2. If the literal $\bar{x}_j$ is found in clause C_k, then $\bar{l}_{j,k}$ is unflagged.

Each flag has two orientations with respect to armature it is attached to. Each flag has four potential positions, the flag can reflect left or right about the armature and the armature can reflect up or down about the shaft. The *flags* are equilateral triangles attached to the armatures. The placement of the flags is dependent



$$C_3 = (x_4 \cup \bar{x}_2 \cup \bar{x}_5)$$
$$C_2 = (x_1 \cup x_6 \cup \bar{x}_2)$$
$$C_1 = (\bar{x}_1 \cup \bar{x}_5 \cup x_4)$$

Figure 2.3: A logic engine constructed from the boolean formula $\Phi = C_1 \cap C_2 \cap C_3$.

on the instance of the NAE3SAT boolean formula. Each flag as two orientations.

For the NP hardness reduction in Theorem 7 we need to make sure that all parameters in the logic engine can be specified polynomially in terms of the size of the boolean formula. Given Φ, the corresponding logic engine is constructed as follows: all components will be specified with a quantity and coordinates defined as polynomials in m and n.

Component	Quantity	Set Definition
Rigid Frame	1	$\{ (x,y) \in \mathbb{R}^2 \,\vert \text{The boundary of } [\frac{1}{2}, n+\frac{1}{2}] \times [-m,m] \}$
Shaft	1	$\{ (x,y) \in \mathbb{R}^2 \,\vert x \in [\frac{1}{2}, n+\frac{1}{2}] \text{ and } y = 0 \}$
Armatures	n	For the j^{th} armature we have $\{ (x,y) \in \mathbb{R}^2 \,\vert x = j \text{ and } y \in [-m,m] \}$
Flags	$2mn-3m$	if $\ell_{j,k}$ is flagged then the attached flag is a regular triangle with side length 1.

Table 2.1: The quantity and coordinates of the logic engine components.

2.1.2 The mechanics of the logic engine

In any of the following cases, a *collision* of flags occurs:

1. flags in the same row on adjacent armatures point toward each other.
2. a flag from the rightmost armature A_n points towards the outer rigid frame.
3. a flag from the leftmost armature A_1 points inwards of A_1.

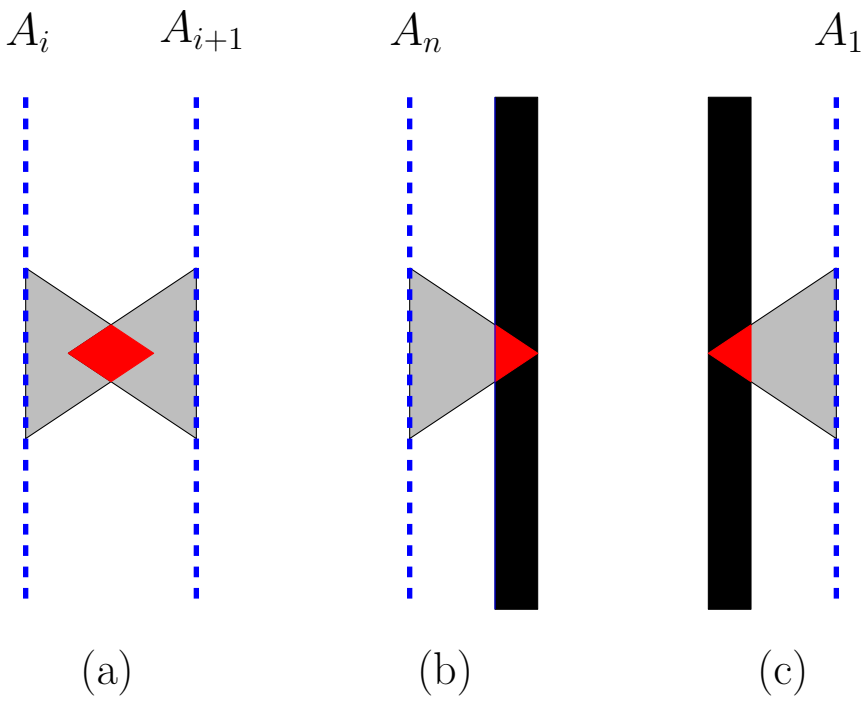


Figure 2.4: (a) Illustrates a adjacent flag collision at the same height, (b) and (c) illustrates a rigid frame collision.

The logic engine representation corresponding to Φ is to be configured such that no horizontally adjacent flags collide and flags do not collide with the rigid frame.

Lemma 1. *A row has a collision-free configuration if and only if it has at least one unglagged armature.*

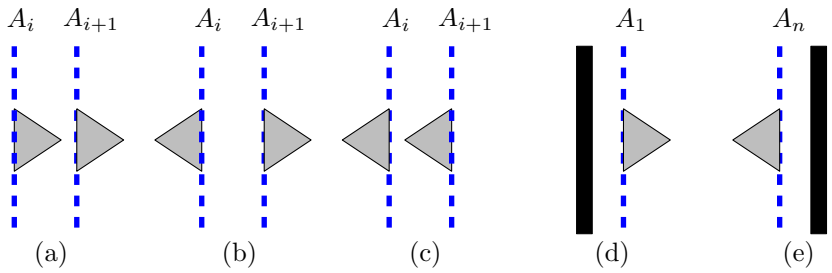


Figure 2.5: The following configuration of adjacent flags and flags that are adjacent to the rigid frame.

Proof. Suppose all armatures are flagged in a row. The flag on armature A_1 must point to the right otherwise we result in a rigid frame collision. A_2 must point to the right otherwise we result in a rigid frame collision. Without loss of generality, A_i and A_{i+1} must point to the right in order to prevent an adjacent flag collision. This implies that A_n must also point to the right which results into a rigid frame collision.

A same argument holds with the argument beginning with the flag on the armature A_n pointing to the left. Thus there is no collision-free configuration with all armatures flagged.

Suppose there is an unflagged armature in a row. Turn all flags towards the nearest unflagged armature. If there are flags on A_1 and A_n, point toward the interior thus they do not collide with the rigid frame. If there are flags on two consecutive armatures, they do not collide because the nearest unflagged armature cannot be between them. Therefore the row has a collision-free configuration. □

A logic engine is said to be *collision-free configurable* when every row has a collision-free configuration.

2.1.2.1 The Relationship of the Logic Engine and NAE3SAT

We show that given an boolean formula in 3-CNF form, Φ, and a truth assignement, τ, where the variables are given a truth assignment such that there is at least one true literal and one false literal in each clause of Φ, then the corresponding logic engine to Φ is collision-free configurable.

Theorem 6. *Given an instance of a NAE3SAT, it is a "yes" instance if and only if the corresponding logic engine is collision-free configurable.*

Proof. Suppose we have an instance of a *NAE3SAT* that is a "yes" instance. This implies that there is a truth assignment such that each clause contains a true and a false literal. Now consider the logic engine corresponding to this instance. We now show that it has a collision free configuration.

For variables that are true, configure the armatures such that the flags corresponding to the non-negated literals reside above the shaft and the flags that correspond to the negated literals reside below this shaft. For variables that are false, configure the armatures in the opposite orientation. Each clause corresponds to a pair of rows in the logic engine, one row for non-negated literals and one for negated literals. Because the *NAE3SAT* is a yes instance, every row contains at least one unflagged armature. By Lemma 1, every row has a collision-free configuration.

Suppose we have an instance of a *NAE3SAT* such that the corresponding logic engine has a collision-free configuration. By Lemma 1 every row at least one unflagged armature. The k^{th} clause is represented by the k^{th} rows above and below the shaft. If the literal x_j is found in clause C_k, then the armature is unflagged in that row. If the literal $\bar{x}_j$ is found in clause C_k, then $\bar{l}_{j,k}$ is unflagged. All flags corresponding to negated literals reside below the shaft and flags corresponding to non-negated literals reside above the shaft. All

together we have that every clause has a true literal and a false literal. Thus, we have a 'yes' instance of the *NAE3SAT*. □

Theorem 7. *Deciding whether a logic engine is collision-free configurable is NP-Hard.*

Proof. In table **??**, we defined the components of the logic engine in terms of polynomials in m and n. If there were a polynomial time algorithm that decides whether a given logic engine is collision-free configurable, then by Theorem 6 we would have a polynomial time algorithm to decide whether an instance of the NAE3SAT is a 'yes' instance. Since NAE3SAT is NP-Hard [9], there is no such algorithm unless $P = NP$. □

]

2.2 Logic Engines Represented as Polygonal Linkages

In the previous section, we introduced the logic engine. This section builds an analogous structure that is formed from a polygonal linkage and we may interchangeably say subcomponent for polygon. We can modify the mechanical structure of the logic engine to form a polygonal linkage. For a given a boolean formula, Φ, in 3-CNF with n variables and m clauses, the rigid frame is broken into two polygons, each polygon on the extremity of the structure. The shaft is broken into n polygons. Each armature is broken into two parts, each part containing m subcomponents. In figure 2.7, each flag becomes a rectangle.

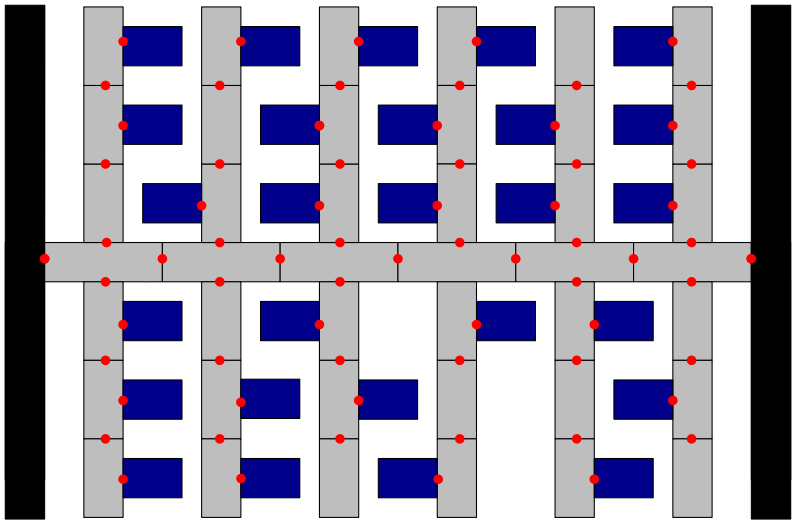


Figure 2.7: A logic engine realized as a polygonal linkage.

2.2.1 Construction of the Polygonal Linkage Logic Engine

Suppose we are given an boolean formula with m clauses and n variables in 3-CNF form, Φ, we construct the polygonal linkage similarly to the logic engine. The corresponding polygonal linkage $P_\ell = (\mathscr{P}, \mathscr{H})$ has the following polygon components:

The large frame subcomponents are hinged on the left most and right most shaft subcomponents. Each adjacent shaft subcomponents are hinged and each shaft subcomponent has two orientations, a reflecion up and a reflection down about the shaft hinge points. On each shaft subcomponent there are two armature shaft subcomponents, one above the shaft subcomponent and one below the shaft subcomponent. By the two orientations of the shaft subcomponent, each armature subcomponent has to possible positions. Each

Component	Height	Width	Quantity
Large Frame Subcomponent	$2 \cdot m$	1	2
Shaft Subcomponent	1	3	n
Armature Subcomponent	2	1	$2 \cdot m$
Flag	1	1.5	$2mn - 3m$

Table 2.2: The components of $\mathscr{P}$ specified polynomially in terms of the size of the boolean formula Φ.

armature comprises of m armature subcomponents that are hinged together; in total there are $2n$ armatures. Each armature subcomponent has two orientations, a reflection left and a reflection right about the armature hinge points. Label the armature subcomponents on the j^{th} armature starting from the shaft by $\ell_{j,1},\ldots,\ell_{j,n}$ on one side and $\bar{\ell}_{j,1},\ldots,\bar{\ell}_{j,n}$ on the other side of the shaft. Attach a rectangular flag specified in Table **??**, to some of these segments. Each segment is either flagged one or zero flags.

1. If the literal x_j is found in clause C_k, then $\ell_{j,k}$ is unflagged.
2. If the literal $\bar{x}_j$ is found in clause C_k, then $\bar{l}_{j,k}$ is unflagged.

Each flag has two orientations with respect to armature it is attached to. Each flag has four potential positions, the flag can reflect left or right about the armature and the armature can reflect up or down about the shaft.

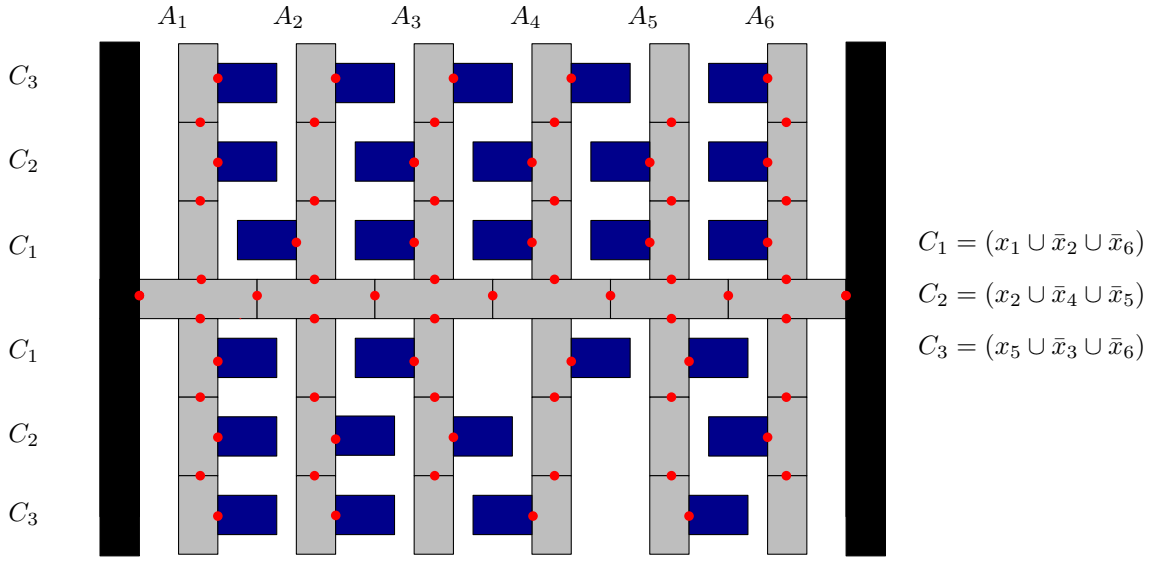


Figure 2.8: A polygonal linkage logic engine that corresponds to the boolean formula $\Phi = C_1 \cap C_2 \cap C_3$.

Theorem 8. *Given an instance of a NAE3SAT, it is a "yes" instance if and only if the corresponding polygonal linkage logic engine has a collision-free configuration.*

Proof. Suppose we have an instance of a *NAE3SAT* that is a "yes" instance. This implies that there is a truth assignment such that each clause contains a true and a false literal. Now consider the polygonal linkage logic engine corresponding to this instance. We now show that it has a collision free configuration.

For variables that are true, configure the armatures such that the flags corresponding to the non-negated literals reside above the shaft and the flags that correspond to the negated literals reside below this shaft. For variables that are false, configure the armatures in the opposite orientation. Each clause corresponds to a pair of rows in the polygonal linkage logic engine, one row for non-negated literals and one for negated literals. Because the *NAE3SAT* is a yes instance, every row contains at least one unflagged armature. By Lemma 1, every row has a collision-free configuration.

Suppose we have an instance of a *NAE3SAT* such that the corresponding polygonal linkage logic engine has a collision-free configuration. By Lemma 1 every row at least one unflagged armature. The k^{th} clause is represented by the k^{th} rows above and below the shaft. If the literal x_j is found in clause C_k, then the armature is unflagged in that row. If the literal $\bar{x}_j$ is found in clause C_k, then $\bar{l}_{j,k}$ is unflagged. All flags corresponding to

negated literals reside below the shaft and flags corresponding to non-negated literals reside above the shaft. All together we have that every clause has a true literal and a false literal. Thus, we have a 'yes' instance of the *NAE*3*SAT*. □

2.2.2 Properties for Weighted Trees and Polygonal Linkages

In order to perform our analysis for weighted trees and polygonal linkages, we'll want to use a suitable metric. The usual Euclidian distance will not suffice for this analysis and so we turn to the Hausdorff distance.

Hausdorff Distance Let A and B be sets in the plane. The *directed Hausdorff distance* is

$$d(A,B) = \sup_{a\in A} \inf_{b\in B} ||a-b|| \tag{2.1}$$

$h(A,B)$ finds the furthest point *ainA* from any point in B. *Hausdorff distance* is

$$D(A,B) = \max\{d(A,B), d(B,A)\} \tag{2.2}$$

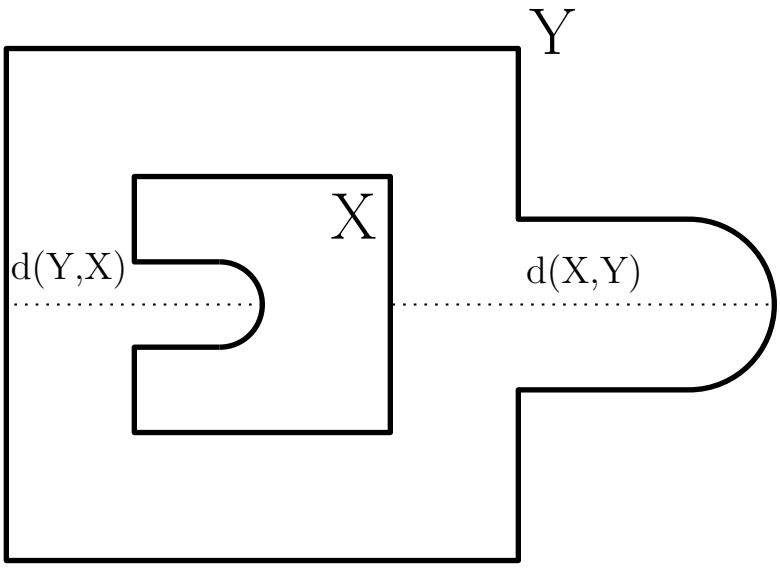


Figure 2.9: An illustrative example of $d(X,Y)$ and $d(Y,X)$ where X is the inner curve, and Y is the outer curve.

ε-approximation The weighted graph, G, is an *ε-approximation* of a polygon P if the Hausdorff distance between every realization such realization of G as a contact graph of disks and a congruent copy of P is at most epsilon. A weighted graph G is said to be a *$O(f(x))$-approximation* of a polygon P if there is a positive constant M such that for all sufficiently large values of x the Hausdorff distance between every realization such realization of G as a contact graph of disks and a congruent copy of P is at $M \cdot |f(x)|$. A weighted graph G is said to be a *stable* if it has the property that for every two such realizations of G, the distance between the centers of the corresponding disks is at most ε after a suitable rigid transformation.

2.3 Realizability Problems for Weighted Trees

In figure 2.10, we have a set of unit radius disks (circles) arranged in a manner that outlines regular, concentric hexagons.

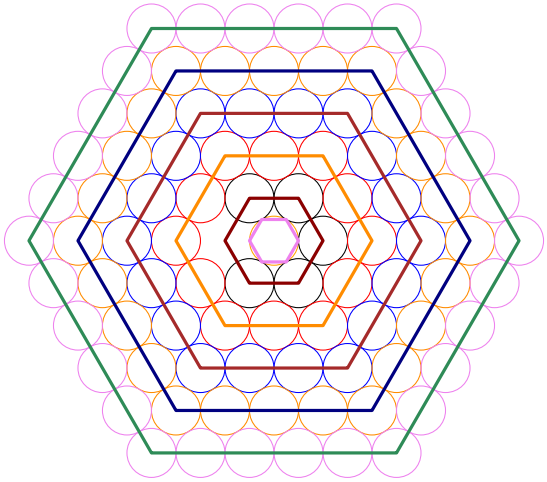


Figure 2.10: A contact graph that resembles the shape of concentric hexagons.

Problem 11 (Appoximating Polygonal Shapes with Contact Graphs). For every $\varepsilon > 0$ and polygon P, there exists a contact graph $G = (V,E)$ such that the Hausdorff distance $d(P,G) < \varepsilon$

Recall problems (**??**) and (**??**): given a positive weighted tree, T, is T the (ordered) contact graph of some disk arrangement where the radii are equal to the vertex weights. For now, we'll focus on a particular family of this problem space where the weighted trees can be realized as a *snowflake*. For $i \in \mathbb{N}$, the construction of the snowflake tree, S_i, is as follows:

- Let v_0 be a vertex that has six paths attached to it: p_1, p_2, ..., p_6. Each path has i vertices.
- For every other path p_1, p_3, and p_5:
  - Each vertex on that path has two paths attached, one path on each side of p_k.
  - The number of vertices that lie on the path attached to the j^{th} vertex of p_k is $i - j$.

A *perfectly weighted snowflake tree* is a snowflake tree with all vertices having weight $\frac{1}{2}$. A *perturbed snowflake tree* is a snowflake tree with all vertices having weight of 1 with the exception of v_0; in a perturbed snowflake tree, v_0 will have a radius of $\frac{1}{2} + \gamma$. For our analysis, all realizations of any snowflake, perfect or perturbed, shall have v_0 fixed at origin. This is said to be the canonical position under Hausdorff distance of the snowflake tree.

Consider the graph of the triangular lattice with unit distant edges:

$$\begin{aligned} V &= \left\{ a \cdot (1,0) + b \cdot \left(\frac{1}{2}, \frac{\sqrt{3}}{2} \right) : a,b \in \mathbb{Z} \right\} \\ E &= \{\{u,v\} : ||u-v|| = 1 \text{ and } u,v \in V\} \end{aligned}$$

The following graph, $G = (V,E)$ is said to be the *unit distance graph* of the triangular lattice. We can show that no two distinct edges of this graph are non-crossing. First suppose that there were two distinct edges that crossed, $\{u_1,v_1\}$ and $\{u_2,v_2\}$. With respect to u_1, there are 6 possible edges corresponding to it, with each

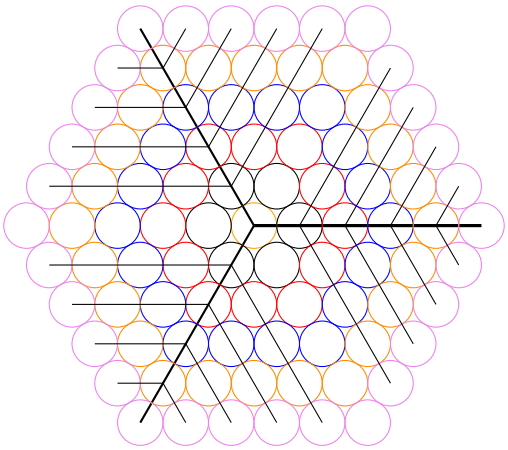


Figure 2.11: The same contact graph as in figure 2.10 overlayed with the a perfectly weighted snowflake tree.

edge $\frac{\pi}{3}$ radians away from the next. Neither edge crosses another; and so we have a contradiction that there are no edge crossings with $\{u_1, v_1\}$.

The perfectly weighted snowflake tree that is a subgraph over the *unit distance graph*, $G = (V,E)$, of the triangular lattice. To show this, for any S_i, fix $v_0 = 0 \cdot \cdot (1,0) + 0 \cdot \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) = (0,0) \in V$ at origin. Next consider the six paths attached from origin. Fix each consecutive path $\frac{\pi}{3}$ radians away from the next such that the following points like on the corresponding paths: $(1,0) \in p_1, \left(\frac{1}{2}, \frac{\sqrt{2}}{3}\right) \in p_2, \left(-\frac{1}{2}\mathrm{p}_4, \frac{\sqrt{3}}{2}\right) \in p_3, (-1,0) \in p4, \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right) \in p_5, \left(\frac{1}{2}, -\frac{\sqrt{3}}{2}\right) \in p_6$. For S_i, there are i vertices on each path.

We define the six paths from origin as follows:

$$
\begin{aligned}
p_1 &= \left\{ a \cdot (1,0) = \vec{v} \middle| a \in \mathbb{R}^+ \right\} \\
p_2 &= \left\{ a \cdot \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) = \vec{v} \middle| a \in \mathbb{R}^+ \right\} \\
p_3 &= \left\{ -a \cdot (1,0) + a \cdot \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) = a\left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right) = \vec{v} \middle| a \in \mathbb{R}^+ \right\} \\
p_4 &= \left\{ a \cdot (-1,0) = \vec{v} \middle| a \in \mathbb{R}^+ \right\} \\
p_5 &= \left\{ a \cdot \left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right) = \vec{v} \middle| a \in \mathbb{R}^+ \right\} \\
p_6 &= \left\{ a \cdot (1,0) - a \cdot \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right) = a \cdot \left(\frac{1}{2}, -\frac{\sqrt{3}}{2}\right) \middle| a \in \mathbb{R}^+ \right\}
\end{aligned}
$$

For S_i there exists i vertices on each path. We shall denote the i^{th} vertex on the j^{th} path as $v_{j,i}$. For each path defined above, the paths are defined as a set of vectors, $\vec{v} = a \cdot \vec{p}$ for some $a \in \mathbb{R}^+$ and $\vec{p} \in \mathbb{R}^2$. By setting $a = 1,2,\ldots,i$, we obtain points that are contained in V. For $j = 1,3,5$ and $l = 1,...,i$, there exists two paths attached to each vertex $v_{j,l}$. For S_i, each path attached to the k^{th} vertex of p_j, there are $i-k$ vertices. We will need to show that each of the $i-k$ vertices on each corresponding path are also in V.

The triangular lattice is symmetice under rotation about v_0 by $\frac{\pi}{3}$ radians. For each vertex $v_{1,l}$ for $l = 1,2,...,i-k$, we place two paths from it; the first path $\frac{\pi}{3}$ above p_1 at $v_{1,l}$ and $\frac{-\pi}{3}$ below p_1 at $v_{1,l}$ and call these paths $p^+_{1,l}$ and $p^-_{1,l}$ respectively. With respect to $v_{1,l}$, one unit along $p^+_{1,l}$ is a point on the triangular lattice

and similarly so on $p^-_{1,l}$. Continuing the walk along these paths, unit distance-by-unit distance, we obtain the next point corresponding point on the the triangular lattice up to $i-k$ distance away from $v_{1,l}$. This shows that each of the $i-k$ vertices on $p^-_{1,l}$ and $p^+_{1,l}$ are in V. By rotating all of the paths along p_1 by $\frac{2\pi}{3}$ and $\frac{4\pi}{3}$, we obtain the the paths along p_3 and p_5 respectively, completing the construction.

2.3.1 On the Decidability of Problem (??)

Recall that problem (**??**) states: given a positive weighted tree, T, is T the contact graph of some disk arrangement where the radii are equal to the vertex weights?

Proof. Suppose we are given a positive weighted tree, $T = (V_1, E_1)$. By the Disk Packing Theorem, there is a disk arrangement in the plane, D, whose contact graph, $G = (V_2, E_2)$ is isomorphic to T. We need to so that $G = T$ and the radii of the disks in D are equal to the vertex weights of T.

To show that $G = T$, we need to show that $V_1 = V_2$ and $E_1 = E_2$.

To show that the radii of the disks in D are equal to the vertex weights of T, we first consider .... □

2.4 chapter 3

The *graph associated* to a Boolean formula in 3-CNF is a bipartite graph where the two vertex classes correspond to the variables and to the clauses, respectively; there is an edge between a variable x and a clause C iff x or $\neg x$ appears in C. See Fig. 2.12 (left).

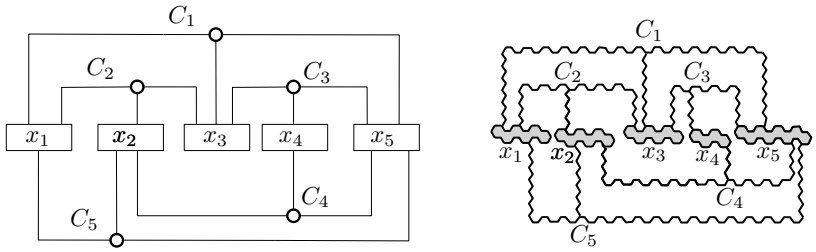


Figure 2.12: Left: the associated graph $A(\Phi)$ for a Boolean formula Φ. Right: the schematic layout of the variable, clause, and transmitter gadgets in our construction.

Bibliography

[1] Timothy G Abbott, Zachary Abel, David Charlton, Erik D Demaine, Martin L Demaine, and Scott Duke Kominers. Hinged dissections exist. *Discrete & Computational Geometry*, 47(1):150–186, 2012.

[2] Martin Aigner and Günter M Ziegler. Hilbert's third problem: decomposing polyhedra. *Proofs from the book*, pages 53–61, 2010.

[3] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[4] S. N. Bhatt and S. S. Cosmadakis. The complexity of minimizing wire lengths in VLSI layouts. *Inform. Process. Lett 25*, 4:263–267, 1987.

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

[6] 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.

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

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

[9] Michael R Garey and David S Johnson. *Computers and Intractibility*. WH Freeman and company New York, 1979.

[10] Paul Koebe. *Kontaktprobleme der konformen Abbildung*. Hirzel, 1936.

[11] Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 1(15):271–283, 1930.

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

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

[14] E. C. Zeeman. On hilbert's third problem. *The Mathematical Gazette*, 86(506):241–247, 2002.