CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

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

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

by

Clinton Bowen

August 2014

The thesis of Clinton Bowen is approved:

Dr. Silvia Fernandez — Date

Dr. John Dye — Date

Dr. Csaba Tóth, Chair — Date

California State University, Northridge

Table of Contents

ABSTRACT

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

By

Clinton Bowen

Master of Science in Applied Mathematics

abstract goes here

Chapter 1

Background

In this section, we cover the background subjects needed to formally pose the problem and present solutions in this thesis. We start with two types of combinatorial structures, linkages and polygonal linkages. We then discuss the configuration spaces of linkages and polygonal linkages. We then look into an alternate representation of linkages, disk arrangements and state the disk arrangement theorem. We then look at satisfiability problems and then review a framework, the logic engine, which can encode a type of satisfiability problem. Finally, we cover the basic definitions of algorithm complexity for **P** and **NP**.

1.1 Graphs

A *graph* is an ordered pair $G = (V,E)$ comprising of a set V of vertices and a set E of edges or lines. Every edge $e \in E$, is an unordered pair of distinct vertices $u, v \in V$ (the edge represents their adjacency, $e = \{u, v\}$). With this definition of a graph, there are no loops (self adjacent vertices, $\{v, v\}$) or multi-edges (several edges between the same pair of vertices).

A motivation for using graphs is modelling physical objects like molecules. This requires an embedding into the plane or $\mathbb{R}^3$. An *embedding* of the graph $G = (V,E)$ is an injective mapping $\Pi : V \mapsto \mathbb{R}^2$ (see Figure 1.1).

$G = (V, E)$
$V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$E = \{01, 12, 23, 34, 40, 58, 86, 69, 97, 75, 05, 16, 27, 38, 49\}$

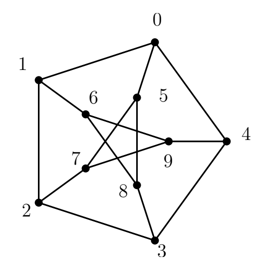


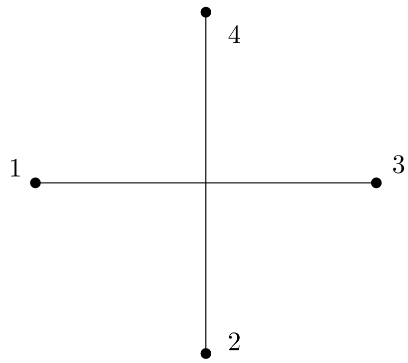
Figure 1.1: An embedding of the Peterson graph.

1.1.0.1 Edge Crossings

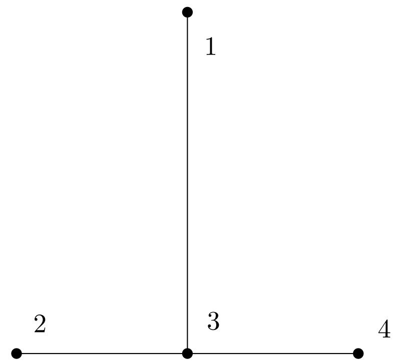
We define *plane embeddings* of a graph to be an embedding where the following degenerate configurations do not occur:

(i) the interiors of two or more edges intersect, or

(ii) an edge passes through a vertex



(a) The interior of the edges intersect.



(b) An edge passes through a vertex.

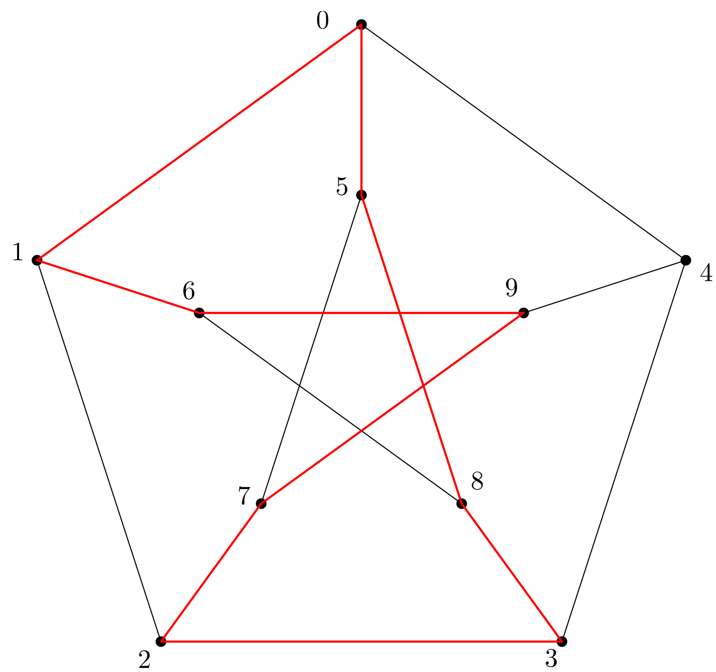
Figure 1.2: These figures exhibit the 4 types of edge crossings.

A graph is called *planar* if it admits a plane embedding. A *plane graph* is a graph together with a plane embedding.

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, $(v_1, v_2, \dots, v_{t-1}, v_t)$, of distinct vertices such that every two consecutive vertices are connected by an edge, and the last vertex, v_t, connects to v_1. 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.



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

1.1.2 Ordered Trees

An *ordered tree* is a tree together with a cyclic order of the neighbors for each vertex. Embeddings of

$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$E = \{12, 23, 24, 35, 36, 37, 48, \}$
$Order_1 = \{(2)_1, (134)_2, (2567)_3, (8)_4, (3)_5, (3)_6, (3)_7, (4)_8\}$
$Order_2 = \{(2)_1, (143)_2, (2657)_3, (8)_4, (3)_5, (3)_6, (3)_7, (4)_8\}$

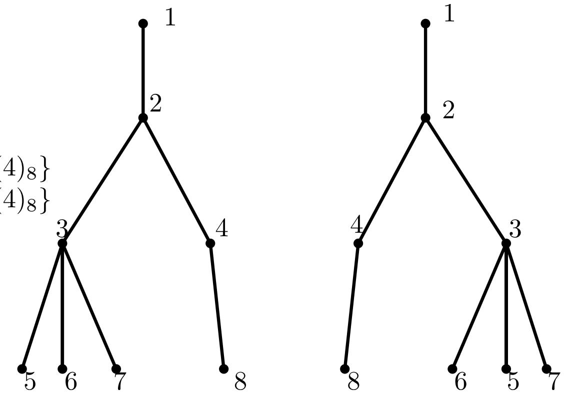


Figure 1.4: A tree with two embeddings with different cyclic orderings around vertices.

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

1.1.3 Graph Isomorphism

To determine when two graphs are equivalent, we need to define an isomorphism for graphs. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a *graph isomorphism* 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 $(f(u), f(v)) \in E_2$.

Graph	Vertices	Edges
G_1	$\{a,b,c,d,e\}$	$\{ab,(b,c),(c,d),(d,e),(e,a)\}$
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 1.5: This figure depicts the graph isomorphism shown in Table (**??**) between V_1 and V_2 in the plane.

1.2 Linkages

When graphs model physical objects, distances between adjacent vertices matter. The length assignment of a graph $G = (V, E)$ is $\ell : E \mapsto \mathbb{R}^+$. A *linkage* is a graph $G = (V, E)$ with a length assignment $l : E \mapsto \mathbb{R}^+$.

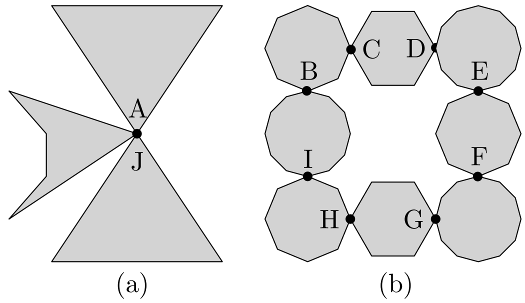


Figure 1.6: An embedded linkage

We consider embeddings of a graph that respects the length assignment. A *realization* of a linkage, G and l, is an embedding of a graph, Π, such that for every edge $\{u,v\} \in E$, $\ell(\{u,v\}) = |\Pi(u) - \Pi(v)|$. A *plane realization* is a plane embedding with the property, $l(\{u,v\}) = |\Pi(u) - \Pi(v)|$.

1.3 Polygonal Linkages

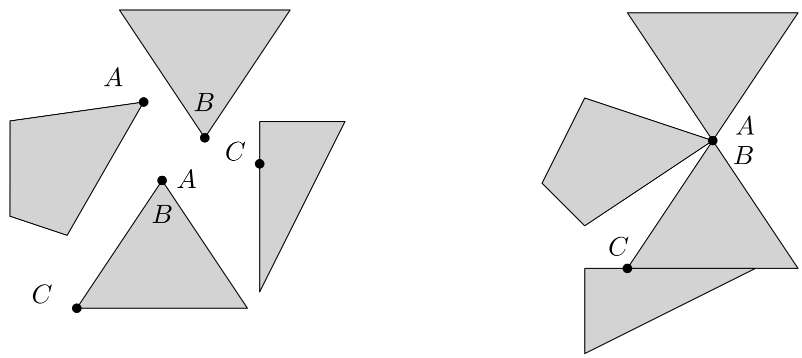


Figure 1.7: (a) A polygonal linkage with a non-convex polygon and two hinge points corresponding to three polygons. Note that hinge points correspond to two distinct polygons.(b) Illustrating that two hinge points can correspond to the same boundary point of a polygon.

A generalization of linkages are polygonal linkages where the edges of given lengths are replaced by rigid polygons. Formally, a *polygonal linkage* is an ordered pair $(\mathsf{PP}, \mathscr{H})$ where PP is a finite set of polygons and $\mathscr{H}$ is a finite set of hinges; a *hinge* $h \in \mathscr{H}$ corresponds to two points on the boundary of two distinct polygons in PP. A *realization* of a polygonal linkage is an interior-disjoint placement of congruent copies of the polygons in PP such that the points corresponding to each hinge are identified (Fig. **??**). This definition of realization rules well known geometric dissections (e.g. Fig. 1.8).

1.3.1 Disk Arrangements

It turns out the disk arrangements are an equivalent way to to represent plane graphs. By representing vertices as interioir disjoint disks and by representing edges as as points of intersections (contact), *kissing points* between two disks. The graph corresponding to a given disk arrangement, $\mathscr{D}$, is said to be the *contact graph*. A *disk arrangement* is a set, $\mathscr{D}$, of pairwise interior-disjoint disks in the plane, $\mathscr{D} = \{C_i\}_{i=1}^n$. $\{C_i\}_{i=1}^n$ such that for any circle $C \in \{C_i\}_{i=1}^n$, C

A classical result by Thurston and Koebe is that every disk arrangement embedded into the plane had a corresponding plane graph.

Theorem 1 (**??**Disk Packing Theorem)**.** *For every graph G, there is a disk arrangement in the plane whose*

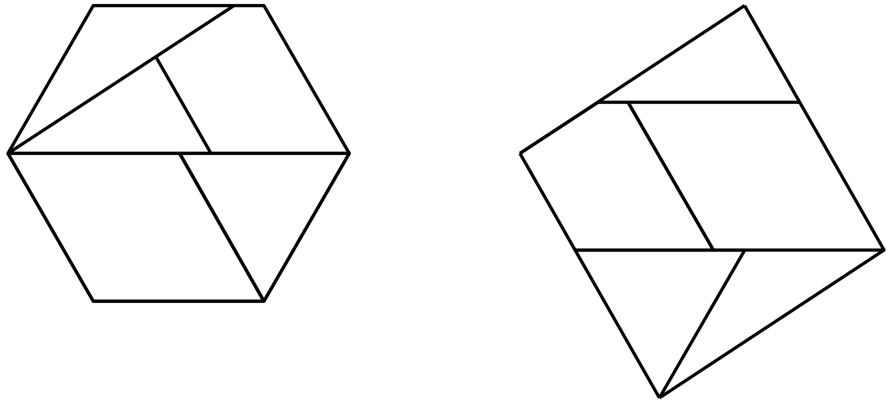


Figure 1.8: Two configurations of polygonal linkage where the polygons touch on boundary segments instead of hinges. These two realizations of the polygonal linkage are invalid to our definitions.

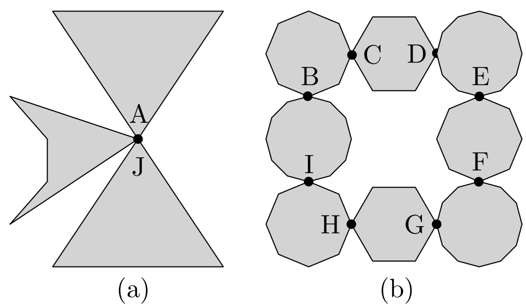


Figure 1.9: (a) A polygonal linkage with a non-convex polygon and a hinge point corresponding to three polygons. (b) A polygonal linkage with 8 regular polygons.

contact graph is isomorphic to G.

Proposition 1. *For every linkage L, there is a disk arrangement in the plane whose contact graph is isomorphic to L.*

1. Show the relation between polygonal linkages and disk arrangengements.

1.3.1.1 Ordered Disk Arrangement

Suppose we're given a tree. By the disk packing theorem we can ascertain a sense of order for the isomorphic disk packing. An *ordered disk arrangement* is a rooted tree in which the counter-clockwise ordering of adjacent vertices.

1.3.1.2 Disk Packing Confinement Problem

Given inputs of radii By adding constraints to the embeddings of disk arrangements, we can devise realizability problem by a volume argument.

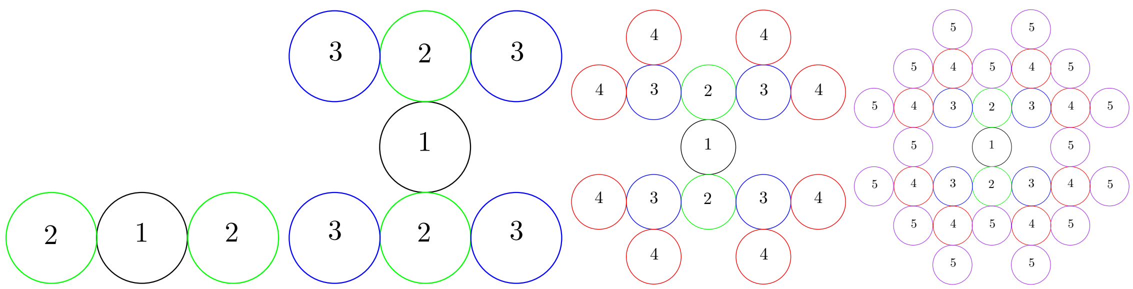
1. Round 1: Start with a disk of unit radius.
2. Round 2: Add two kissing disks, each of diameter 2, that do not intersect with any other disk (they may kiss other disk).

Figure 1.10: This example represents a disk arrangement transformed to and from its corresponding graph G_2

3. Round 3 and Higher: For each new kissing disk added, add two more non-intersecting kissing disks of diameter 2 to it.

For each round i we are adding $2^{(i-1)}$ disks, each with an of π. The area that the disk arrangement is bounded by at round i is a box of length $2 \cdot (2 \cdot (i-1)+1)$ totalling to an area of $(4 \cdot i^2 - 4 \cdot i + 1)$. Meanwhile the total area of the disk arrangement at round i is $\pi \cdot (2^i - 1)$. The exponential growth rate of the disk packing will exceed its bounded area for sufficiently large i.

Figure (1.11) illustrates the iterative problem. The problem with this is that the area in which is necessary to contain this disk growing disk arrangement will exceed the area needed to contain it.



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

1.4 Configuration Spaces

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

Alexander Calder, 1933

We'd like to describe motions and range of motions of embedded graphs, linkages, polygonal linkages, and disk arrangements. Table 1.4 provides the definition of *reconfiguration* for each type of object covered so far:

Object Type	Definition of Reconfiguration
Graph Embeddings	a continuous motion of the vertices that never causes the edges to intersect.
Linkage	a continuous motion of the vertices that preserves the lengths of the edges and never causes the edges to intersect.
Polygonal Linkage	a continuous motion of polygons that preserves shapes of polygons, hinge point pairings, and never causes the polygonal sides to intersect.
Disk Arrangement	a continuous motion of disks that preserves disk radii, pairs of contact points, and never causes disks to intersect.

1.4.1 Configuration Spaces of Linkages

Let's focus on the space of embeddings of a linkage. If there are n vertices of a linkage, the *configuration space* of a linkage is said to be a vector space of dimension $2 \cdot n$ where edge length is preserved. A *configuration space* for a linkage G and corresponding proper embedding, L_1 is said to be for any other

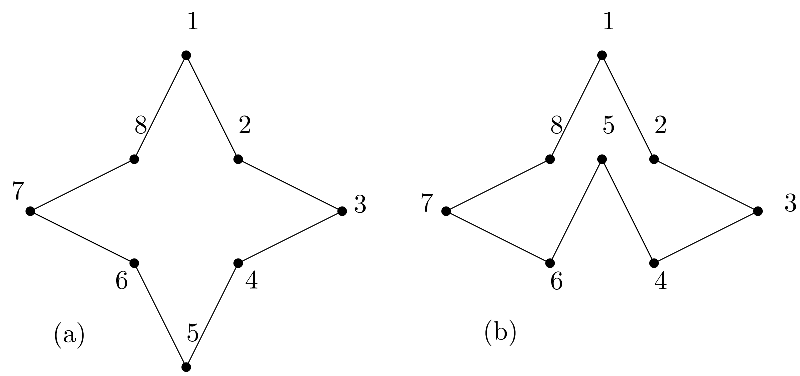


Figure 1.12: (a) and (b) show a linkage in two embeddings.

proper embedding of a linkage G, L_2, such that the lengths of every edge of G is preserved between the two embeddings, i.e.:

$$l((u,v)) = |L_1(u) - L_1(v)| = |L_2(u) - L_2(v)|$$

Equivalent embeddings include translations and rotations about the center of mass on $L(V)$. We further our embeddings by requiring that one vertice is pinned to the point of origin on the plane as well as a neighboring vertex.

Theorem 2 ([**?**, **?**]Carpenter's Rule Theorem)**.** *Every realization of a 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.*

A *reconfiguration* of a linkage whose graph is $G = (V,E)$ and length assignment is ℓ is a continuous function $f : [0,1] \mapsto \mathbb{R}^{2\cdot|V|}$ specifying a configuration of the linkage for every $t \in [0,1]$ where length assignment ℓ is preserved, edges do not cross and for every $\varepsilon > 0$, there exists a $\delta > 0$ such that $|t_1 - t_2| < \delta$ implies

$$|f(t_1) - f(t_2)| < \varepsilon$$

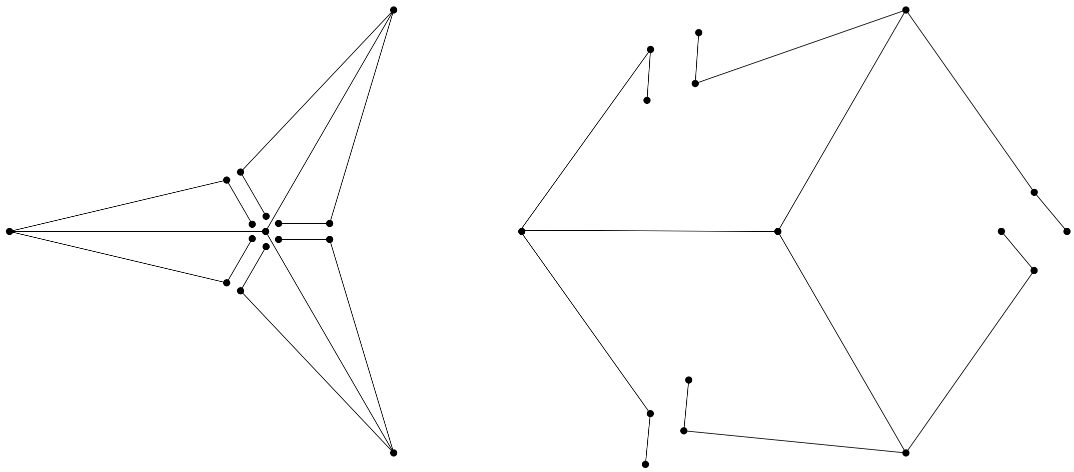


Figure 1.13: A linkage whose complete configuration space is discontinuous. These two examples above are two configurations of the same linkage that cannot continuously transform into the other without edge crossing.

1.4.2 Configuration Spaces of Polygonal Linkages

Some text goes here

1.4.3 Configuration Spaces of Disk Arrangements

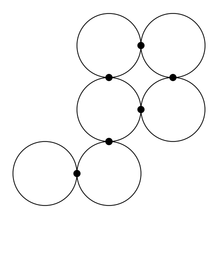


Figure 1.14: A linkage whose complete configuration space is discontinuous. These two examples above are two configurations of the same linkage that cannot continuously transform into the other without edge crossing.

1.5 Algorithm Complexity

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

1.5.1 Qualitative Analysis of Algorithms

Determining the time and space that algorithms use determine their efficiency. The *worst-case* running time is the largest possible running time that an algorithm could have over all inputs of a given size N. *Brute force* is when an algorithm tries all possibilities to see if any formulates a solution. An algorithm is said to be *efficient* if it achieves qualitatively better worst-case performance, at an analytical level, than brute force search.

1.5.2 Categorization of Algorithms

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

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

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

for all $x \geq x_0$

1.5.2.1 P and NP

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

To categorize problems [**?**], we ask the following:

Problem 1. Can arbitrary instances of problem Y by solved using a polynomial number of standard computational steps, plus a polynomial number of calls to an algorithm that solves X?

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

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

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

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

1.5.3 Reduction

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. subsubsectionIndependent 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 3. *Let $G = (V,E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.*

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

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

Theorem 3 allows for problem reductions for independent set and vertex cover problems.

1.5.3.1 Reduction of the Independent Set and Vertex Cover Problem

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

Problem 2 (Optimization of an Independent Set in G). Given a graph G, what is the largest independent set in G?

Problem 3 (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?

An algorithm that solves the optimization problem automatically solves the decision problem of the independent set. An algorithm that solves the decision problem for all size k solves the optimization problem where the decision is "yes" for the largest value of k. This establishes a reduction of the optimization problem to the decision problem and vice versa.

1.6 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 boolean formula is *satisfiable* if one can assign true or false value to each variable so that the formula is true.

Problem 4 (Satisfiability Problem (SAT)). Given a boolean formula, decide whether it is satisfiable.

[**?**] It is known that every boolean formula can be rewritten in *conjunctive normal form* (CNF), a conjunction of clauses. A *clause* is a disjunction of distinct literals. A *literal* is a variable or a negated variable, x_i or $\bar{x}_i$, for $i = 1, \ldots, n$. 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. Given a boolean formula in 3-CNF, decide whether it is satisfiable is a *3-SAT problem*.

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 5 (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 6 (Not All Equal 3 SAT Problem (NAE3SAT)). Given a boolean formula in 3-CNF, decide whether it is satisfiable so that each clause contains a true and a false literal.

3-SAT, PLANAR 3-SAT, NAE3SAT are NP hard [ADD REFERENCE]. THE PROOF FOR THESE REDUCTIONS ARE NOT OBVIOUS BUT ARE SHOWN [HERE, THERE, AND OVER THERE] respectively.

1.7 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 [**?**, **?**]. 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 paper, 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 PP, and edges represent a hinge between two polygons). Our main result is that realizability remains NP-hard when restricted to simply connected structures.

Theorem 4. *It is NP-complete 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-complete to decide whether a given tree (resp., plane tree) with positive vertex weights is the contact graph (resp., ordered contact graph) of a disk arrangements with specified radii.*

The unoriented versions, where the underlying graph (hinge graph or contact graph) is a tree can easily be handled with the logic engine method (Section **??**). We prove Theorem 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 **??**).

1.7.1 Problem Statement

Problem 7 (Unordered Realizibility Problem for Linkages). For a linkage, it asks whether its corresponding graph is the contact graph (resp., ordered contact graph) of some disk arrangement with specified radii.

Problem 8 (Unordered Realizibility Problem for Polygonal Linkages). The *realizability* problem for a polygonal linkage asks whether a given polygonal linkage has a realization.

Problem 9 (Ordered Realizibility Problem for Linkages). For a linkage, it asks whether its corresponding graph is the ordered contact graph of some disk arrangement with specified radii.

Problem 10 (Ordered Realizibility Problem for Polygonal Linkages). The *realizability* problem for a ordered polygonal linkage asks whether a given polygonal linkage has a realization with respect to order.

1.7.2 Decidability of Problem

test

Given a boolean formula in NAE3SAT form,

Chapter 2

Decidability Problems for Hinged Polygons and Disks

2.1 The Logic Engine

2.1.1 Construction of the Logic Engine and Encoding of an NAE3SAT Problem on a Logic Engine

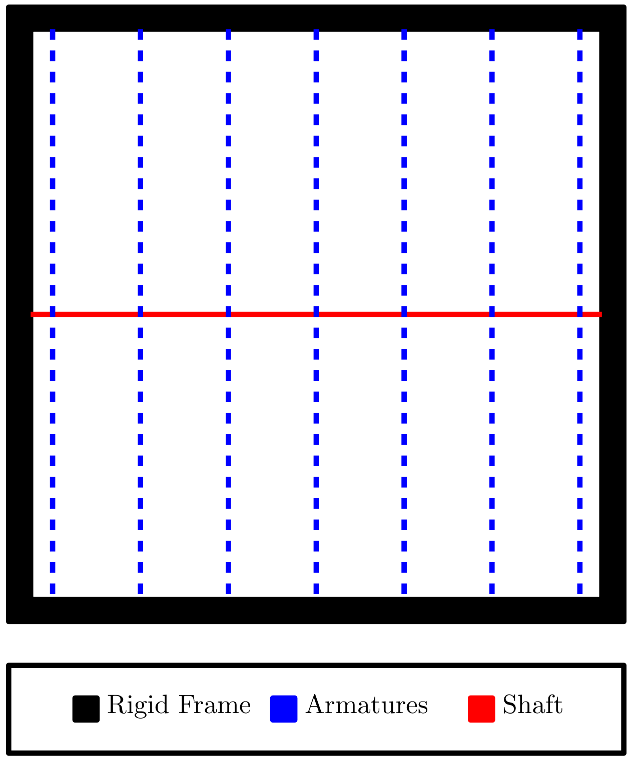


Figure 2.1: A logic engine frame with veritical armatures and a horizontal shaft.

The logic engine is a planar model which can encode instances of the NAE3SAT problem. The components of the logic engine are as follows: the rigid frame, the shaft, the armatures, the chains, and the flags. Figure 2.1 shows the a rigid frame, armatures, and shaft. Each armature represents a boolean variable. The rigid frame defines the border of the model and the armatures has two orientations with respect to the shaft. Flagging arragement indicates the relationship of the boolean literal's existence within a clause. There are two literals for each variable.

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

Negated literals reside below the shaft and non-negated literals reside above the shaft.

A *collision* of flags occur if either of the following occurs:

1. flags in the same row on adjacent armatures point toward each other.
2. a flag from the outermost armature A_n points towards the outer rigid frame.

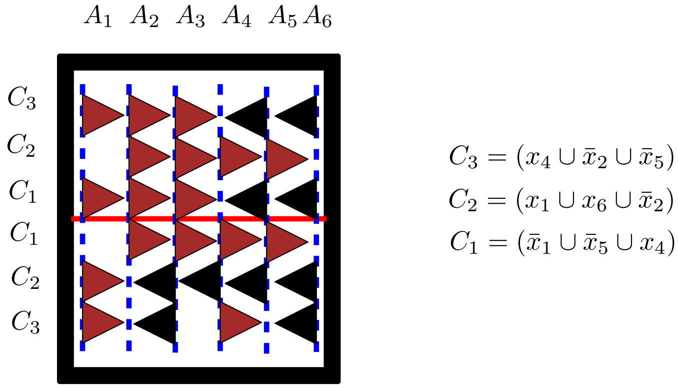


Figure 2.2

3. a flag from the innermost armature A_1 points inwards of A_1.

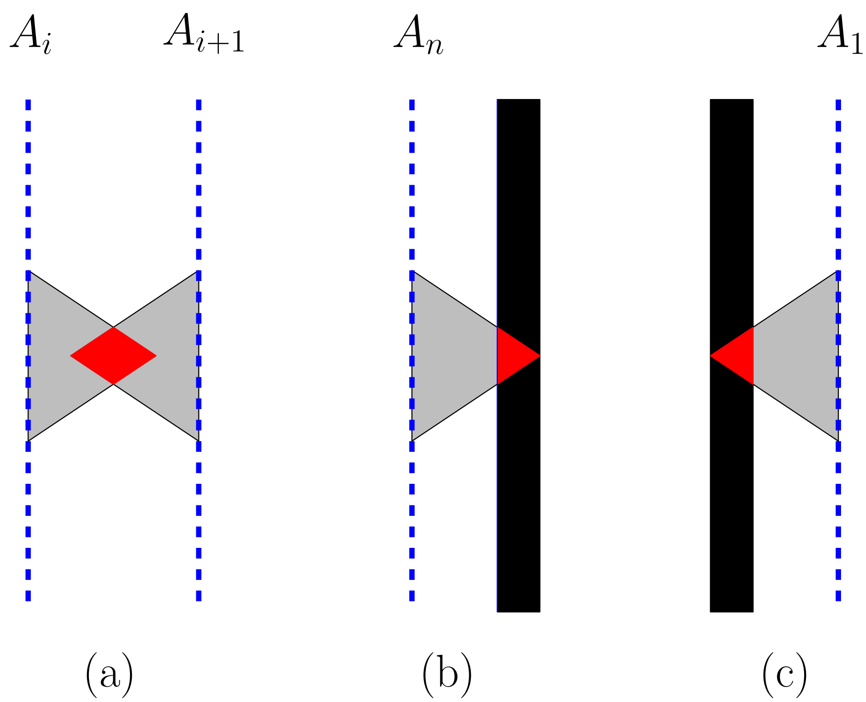


Figure 2.3: (a) Illustrates a adjacent flag collision at the same height, (b) and (c) illustrates a rigid frame collision.

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

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.

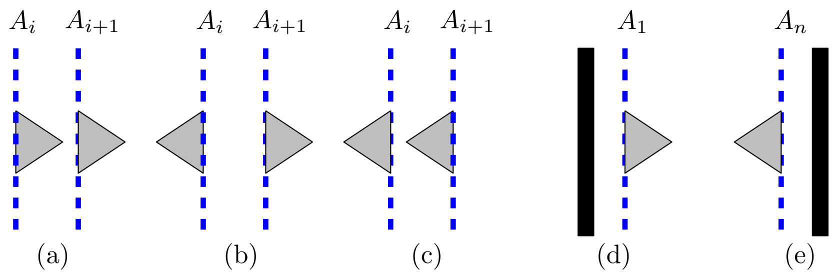


Figure 2.4: The following configuration of adjacent flags and flags that are adjacent to the rigid frame.

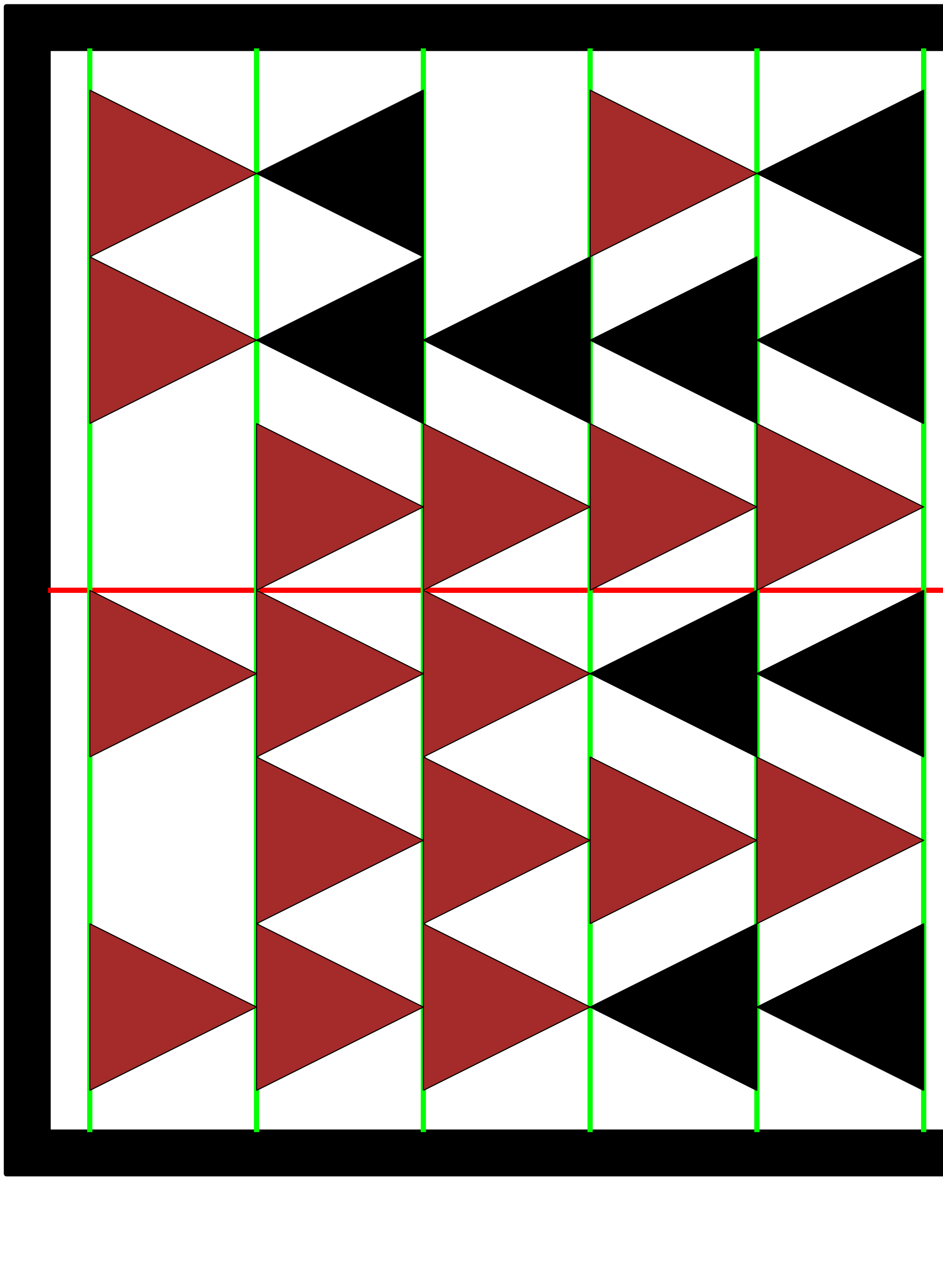
Suppose there is an unflagged armature in a row. Turn all flags towards the nearest unflagged armature. If there are flags on A_1 and A_n, point toward the interior thus they do not collide with the rigid frame. If there are flags on two consecutive armatures, they do not collide because the nearest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

Theorem 6. *Given an instance of a $NAE3SAT$, it is a "yes" instance if and only if the corresponding 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 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 $NAE3SAT$. □



2.2 Construction of the NAE3SAT Problem over Hinged Polygons

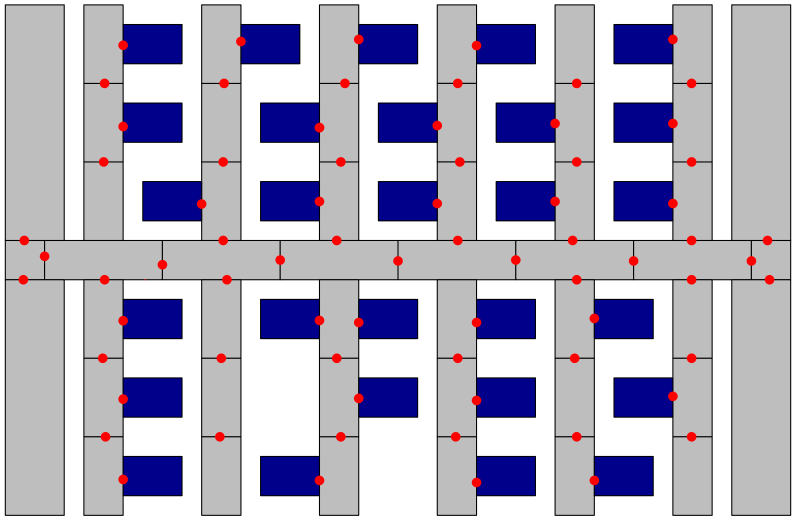


Figure 2.6: The logic engine realized as a polygonal linkage.

Suppose we have a NAE3SAT with m clauses and n variables in 3-CNF form. The polygonal linkage logic engine that corresponds to this boolean formula has the following dimensions:

Component	Height	Width	Quantity
Shaft Subcomponent	1	3	n
Armature Subcomponent	2	1	$2 \cdot m$
Flag	1	1.5	–
Large Frame SubComponent	$2 \cdot m$	1.75	4
Small Shaft Subcomponent	1	1	2

2.3 Construction of the NAE3SAT Problem over Disks

2.4 Logic Engines Represented as Polygonal Linkages

Theorem 7. *It is NP-Complete to decide whether a polygonal linkage is whose hinge graph is a tree can be realized*

2.5 Logic Engines Represented as Unit Disk Contact Graphs