Master's Thesis

Research on Simulation of Mechanism by Linkage

1610428 Feng Tianfeng

Supervisor	Ryuhei Uehara
Main Examiner	Ryuhei Uehara
Examiners	Mineo Kaneko
	Kazuhiro Ogata
	Mizuhito Ogawa

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

February 2019

Abstract

A linkage is a collection of fixed-length 1D segments joined at their endpoints to form a graph. A segment endpoint is also called a vertex. The segments are often called links or bars, and the shared endpoints are called joints or vertices. The bars correspond to graph edges and the joints to graph nodes. The linkage can be distinguished according to their graph structure. The structure may be a general graph, a tree, a single cycle, or a simple path. A linkage whose graph is a path is called a chain, an arc, or a robot arm. In this research, we only consider such a simple linkage whose graph is a path.

The motivation for this research stems from the application of linkage. It is sometimes useful to view an articulated robotic manipulator as a chain of links. The study of linkage, and more generally, mechanisms, has long been important to engineering. The kinematics of mechanisms is often of central concern in practical applications. Therefore, the simulation problem by a linkage is considered in this research for a robot arm that modeled by a linkage P and a general mechanism modeled by a graph G. The mission is to simulate the target graph G by the given linkage P. Each vertex of the graph G should be simulated by some vertices of the linkage P, and each edge of the graph G should be simulated by a subpath of the linkage P. This problem can be formalized as finding a mapping from the path P to a path on the graph G. The decision problem asks if there is an Eulerian path of G spanned by P. To solve this problem, the weighted Eulerian path problem and its variants are investigated.

At first, we try to solve the weighted Eulerian path problem. This problem can be seen as the Eulerian path problem with edge weights for a given path P and a graph G with length function. It asks us to determine if there is an Eulerian path of G spanned by P with length consistency. This problem is linear time solvable if the path P and the graph G consist of unit length edges. However, the first interesting result is that the weighted Eulerian path problem is strongly NP-hard even if edge lengths are quite restricted. Precisely, the problem is strongly NP-hard even if P and G consist of edges of lengths only 1 and 2. We reduce the 3-Partition problem to the weighted Eulerian path problem to show the NP-hardness of this problem. The 3-Partition problem is a well-known NP-complete problem. Therefore, the weighted Eulerian path problem is tackled in two different ways, and two different simulation problems are considered: the elastic linkage problem and the traverse problem of a tree by a path.

The first problem is the elastic linkage problem, and this is an optimization version of the weighted Eulerian path problem. In weighted Eulerian path problem, the path P can only cover an edge of the graph G once, and the edge lengths of the path P are all fixed, and they cannot be changed. However, in this variant, we change the second condition. All edges in P are allowed to be elastic to simulate the target graph G by the path P, that is, we can stretch or shrink the edges in the elastic linkage P. This situation is natural not only in the context of the robot arm simulation but also in the approximation algorithm. So far, we only consider the elastic linkage problem for two paths P and G, and we use the elastic linkage P to simulate the target path G. Firstly, the elastic ratio of edges and mappings in the simulation process are defined. The elastic ratio of edges is always greater than or equal to 1, that is, the elastic ratio is the change factor of the edge in the path P, and the elastic ratio of the mapping is defined by the maximum elastic ratio of all edges in the path P. The objective of the elastic linkage problem is to minimize the elastic ratio of the mapping from the path P to the path G for given P and G. We start from a simple case which the path G consists of only one edge. In this case, we prove that the minimum elastic ratio is achieved when all ratios of edges in path P take the same value, after that, we prove that the elastic linkage problem can be solved in polynomial time by dynamic programming.

The second problem is the traverse problem of a tree by a path. In weighted Eulerian path problem, P can only cover an edge of G once, and the edge lengths of the path P are all fixed. In this variant, we change the first condition. The path P is allowed to cover an edge of the graph G twice or more, but we do not allow edges in the path P to be changed. In this situation, the path P can simulate the graph G even if G does not have an Eulerian path. In this case, we do not allow the path P to be elastic, or its ratio is fixed to 1. Firstly, the general simulation problem is proved to be weakly NP-complete even if G is an edge. It is similar to the ruler folding problem, which is weakly NP-complete. We can reduce the ruler folding problem to the general simulation problem by just letting G be an edge of length L. Thus, we consider more restricted cases. From the viewpoint of graph theory, it is natural to consider the case that G is a tree. For a given tree G and a path P with edge lengths, the traverse problem asks if G has a trail by P such that each edge of G is traversed exactly twice. When G is a tree, the problem is in a simple form, and P simulates G by traversing each edge twice in the unique spanning tree of G or G itself. However, this problem is still strongly NP-complete even in quite restricted cases; (1) G is a star, and P consists of edges of only two different lengths, and (2) G is a spider, and all edges in G and P are of length p and q, where p and q are any

two positive integers that are relatively prime, or $p = 1$ and $q = 2$. We also reduce the 3-Partition problem to the traverse problem of a tree by a path to show the NP-hardness of these restricted cases. On the other hand, when G is a star and its edge lengths are of k different values, we prove that the tree traverse problem can be solved in polynomial time by dynamic programming when k is constant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will introduce the background of the linkage simulation problem and the motivation for this research topic. We will also give the preliminaries to this research problem.

1.1 Background

Linkages are mechanisms composed of stiff, rigid chains, which are jointed at freely rotating joints. The Figure 1.1 shows a desk-lamp linkage, and the linkage flexs at the circled joints.

A *linkage* is a collection of fixed-length 1D segments joined at their endpoints to form a graph. The segments are often called *links*, and the shared endpoints are called *joints* or *vertices*. The bars correspond to graph edges and the joints to graph nodes[1].

The linkages can be distinguished according to their graph structure. The structure may be a general graph, a tree, a single cycle, or a simple path. A linkage whose graph is a path is called a chain, an arc, or a robot arm[1].

The table 1.1 shows the classification parameters for 1D linkage problems.

1.2 Motivation

The motivation for this research stems from the application of linkage. It is sometimes useful to view an articulated robotic manipulator as a chain of links. The study of linkage, and more generally, mechanisms, has long been important to engineering. The kinematics of mechanisms is often of central concern in practical applications. Therefore, the simulation problem by linkage is considered, and it also has the potential for protein folding

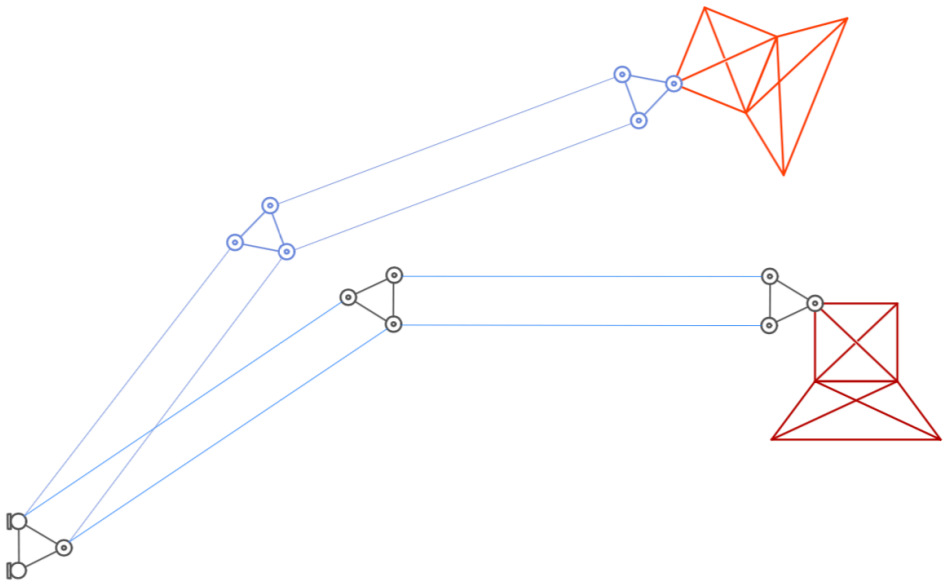

Figure 1.1: A linkage.The linkage flexes at the circled joints; the two left most joints are pinned to the plane. The shaded *lamp* structure is rigid

Focus	Parameter	Values
Linkage	Graph structure	General, tree, polygon, chain
	Intersection constraints	None, obstacles, simple
	Dimension	2D, 3D, 4D, kD
Problem	Geometric issue	Reconfiguration, reachability, locked
	Answer desired	Decision, path planning
	Complexity measure	Combinational, computational bounds

Table 1.1: Classification parameters for 1D linkage problems

problems. A protein is composed of a chain of amino acid and residues joined by peptide bonds. It can be modeled for many purposes as a polygonal chain representing the protein's "backbone", with three atom vertices per residue and adjacent atoms connected by bond links. The linkage simulation is possible to be used to explore the research direction of protein folding on geometry.

1.3 Preliminaries

Linkages are useful models for robot arms: a *robot arm* is a type of programmable mechanical arms, which can be modeled by a *linkage*. As we mentioned before, the linkage is a collection of fixed-length 1D segments joined at their endpoints forming a graph. In this research, we only consider simple linkages, that is, a linkage is a path $P = (v_1, v_2, \ldots, v_n)$ with length function $\ell : E \to \mathbb{R}$, where v_i is an endpoint, $e_i = \{v_i, v_{i+1}\}$ is an edge in $E = \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\}$, and its length is given by $\ell(e_i)$. Now we consider the following situation (Figure 1.2). You are given a general target mechanism which is modeled by a graph $G = (V', E')$, and a robot arm modeled by a linkage $P = (V, E)$ as above with length function $\ell : E \cup E' \to \mathbb{R}$. Our mission is *to simulate* the target graph G by the given linkage P. The joints in P are programmable, and each joint (or vertex) of G should be simulated by a joint of P. However, we can also put the joints of P on some internal points of edges of G because they can be fixed. Therefore, our problem can be formalized as finding the following mapping ϕ from P to G:

- Each vertex of G should be mapped from some vertices of P;
- Each edge of G should be mapped from a subpath of P by ϕ;
- Each edge of P should be mapped to an edge of G, which may span from one internal point to another on the edge.

The decision problem asks if there exists a mapping ϕ from P to G. That is, it asks if there is an Eulerian path of G spanned by P such that (1) when P visits a vertex in G, a vertex of P should be put on it, and (2) some vertices in P can be put on internal points of edges of G. When all edges in P and G have the same length, it is easy to solve that in linear time since the problem is the ordinary Eulerian path problem. In the context of formal languages, there are some variants of the Eulerian path problem with some constraints (see [2] for a comprehensive survey). However, so far, in the *robot arm simulation problem*, our variant of the Eulerian path problem has not been investigated, while the situation is quite natural.

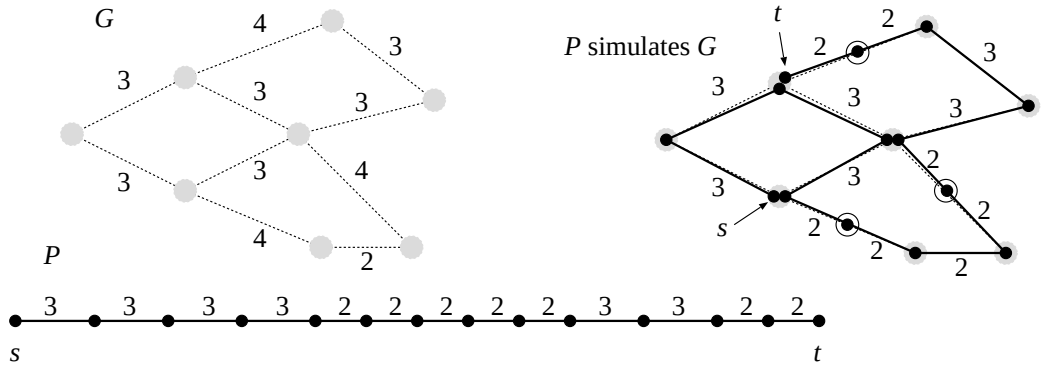

Figure 1.2: A simple example. A robot arm modeled by P can simulate a given mechanism modeled by the graph G as shown in the figure. When P simulates G, each circled joint is fixed, and two joints of the robot arm on the same vertex of G move with synchronization.

We only consider a simple undirected graph $G = (V, E)$. A *path* $P = (v_1, v_2, \ldots, v_n)$ consists of n vertices with $n-1$ edges e_i joining v_i and v_{i+1} for each $i = 1, \ldots, n-1$. The vertices v_1 and v_n of the path are called *endpoints*. Let $K_{n,m}$ denote a *complete bipartite graph* $G = (X, Y, E)$ such that $|X| = n$, $|Y| = m$, and every pair of a vertex in X and a vertex in Y is joined by an edge. A graph $G = (V, E)$ is a *tree* if it is connected and acyclic. Here a graph G is a *star* if and only if it is a complete bipartite graph $K_{1,n-1}$, and a graph G is a *spider* if and only if G is a tree that has only one vertex of degree greater than 2. In a star or a spider, the unique vertex of degree greater than or equal to 3 is called *center*. Without loss of generality, we assume that a star or a spider always has the center. Let $G = (V', E')$ and $P = (V, E)$ be a graph and a path $(v_1, v_2, \ldots, v_n)$. Let $\ell : E' \cup E \to \mathbb{R}$ be an edge-length function of them. We say the linkage P can *simulate* the mechanism G if each edge in G is spanned by at least one subpath of P, and no subpath of P properly joins two non-adjacent vertices in G. We formalize the notion of simulation by a mapping ϕ that maps each vertex V in P to a *point* in G as follows. For any edge $e = \{u, v\} \in E'$, we consider e a line segment (u, v) of length $\ell(e)$. Then the *intermediate point* p at distance $t\ell(e)$ from u is denoted by $p = tv + (1-t)u$, where $0 < t < 1$. We note that the endpoints of an edge e are not considered intermediate points of e. Now we first define a set of *points* in G by V' and all intermediate points on edges of E'. Then we define a mapping ϕ from V to points of G as follows. To make it clear, we first divide V in P into two subsets V_e and V_i such that each vertex in V_e is mapped to a vertex in V', and each vertex in V_i is mapped to an

intermediate point of G. In our problems, we assume that $\phi(v_0)$ and $\phi(v_n)$ should be in V_e. That is, P should start and end at vertices in G. Depending on the restrictions, we consider some different simulation problems and get some interesting results.

The first interesting result is that this problem is strongly NP-hard even if edge lengths are quite restricted. Precisely, the problem is strongly NP-hard even if P and G consist of edges of lengths only 1 and 2 (Theorem 1). We remind that if they consist of unit length edges, the problem is linear time solvable. We thus tackle this problem in two different ways.

The first problem is an optimization version of this problem. In this variant, we consider a linkage is *elastic*, that is, the length of one line segment is not fixed and can be changed a little bit. This situation is natural not only in the context of the robot arm simulation but also in the approximation algorithm. Formally, we allow the edges in P to be elastic to fit the vertices of P to ones of G. Our goal is to minimize the stretch/shrink ratio of each edge of P. We show that when G is a path, this can be solved in polynomial time by dynamic programming.

In the second way, we allow P to cover an edge of G twice or more. In this situation, we can simulate G by P even if G does not have an Eulerian path. In this case, we do not allow P to be elastic, or its ratio is fixed to 1. We first show that the problem is weakly NP-hard even if G is an edge (Theorem 3). In fact, this problem is similar to the ruler folding problem (see, e.g., [3, 1]). From the viewpoint of a simulation of G by P, we can take the following strategy in general. For a given G, first make a spanning tree T of G, and traverse T in the depth first manner. It is easy to see that this strategy works for any connected graph G even if G does not have an Eulerian path and the trail of the traverse of T gives us a way of simulation of G by P in a sense. In this paper, we focus on the case that the graph G is a tree. Precisely, we consider the following problem: For a given tree G and a path P (with edge lengths), the traverse problem asks if G has a trail by P such that each edge of G is traversed exactly twice. (We note that trees form a representative class of graphs that have no Eulerian paths.) We first mention that this problem is quite easy when each edge has unit length. The answer is yes if and only if $G = (V, E)$ is connected and P contains $2|E|$ edges. When G is connected, P can traverse every edge twice in the way of depth first search. This idea brought us a natural restriction that asks if P can cover G by traversing edges of G exactly twice. This idea comes from the depth first search naturally. That is, even if G has no Eulerian path and hence P cannot simulate G properly, we can find the feasible way to simulate G by P in this way. First make a spanning tree T of G and traverse T by P so that each edge of T is spanned twice by two subpaths of P. From the

practical viewpoint, it seems to be reasonable when we simulate a mechanism by a robot arm. From the viewpoint of graph theory, it is natural to consider the case that G is a tree. When G is a tree, the problem is in a simple form; P simulates G by traversing each edge twice in the unique spanning tree of G or G itself. However, this problem is still strongly NP-hard even in quite restricted cases; (1) G is a star, and P consists of edges of only two different lengths, and (2) G is a spider, and all edges are of two different lengths. On the other hand, the problem is polynomial time solvable when G is a star and its edge lengths are of k different values.

In this research problem, we will often use the following problem to show the hardness of our problems.

3-Partition Problem

Input: An integer B and a multiset A of $3m$ integers $A = \{a_1, a_2, \ldots, a_{3m}\}$ with $B/4 < a_i < B/2$ for each $i = 1, 2, \ldots, 3m$.

Output: Determine if A can be partitioned into m multisets $S_1, S_2, \ldots, S_m$ such that $\sum_{a_j \in S_i} a_j = B$ for every i.

Without loss of generality, we can assume that $\sum_{a_i \in A} a_i = mB$, and $|S_i| = 3$. It is well known that the 3-Partition problem is strongly NP-complete [1].

Chapter 2

Weighted Eulerian path problem

In this chapter, we introduce the weighted Eulerian path problem.

2.1 Description

We consider the mapping ϕ from V_e to V' that satisfies some conditions as follows; (1) for every $v' \in V'$, there is at least one vertex $v \in V_e$ with $\phi(v) = v'$; (2) for each edge $e' = \{v', u'\} \in E'$, there is a pair of vertices v_i and v_j in V_e such that (2a) $\ell(e') = \sum_{k=i}^{j-1} \ell(e_k)$, and (2b) there is no other vertex v_k is in V_e between v_i and v_j. Intuitively, each edge e' in G corresponds to a subpath in P, and vice versa. In other words, some vertices in P are mapped to some intermediate points in G, and the corresponding joints of the robot arm are fixed when the robot arm P simulates the target mechanism G. We note that by the length condition (2a), we can assume that when the subpath $(v_i, \ldots, v_j)$ in P simulates an edge $e = \{\phi(v_i), \phi(v_j)\}$, it is not allowed to span it in a zig-zag way. We also add one condition: (3) for each edge (v_k, v_{k+1}) in P, $\{\phi(v_k), \phi(v_{k+1})\}$ should be on an edge e in G. That is, there is a subpath $(v_i, \ldots, v_k, v_{k+1}, \ldots, v_j)$ with $i \leq k < k+1 \leq j$ such that (3a) $e = \{\phi(v_i), \phi(v_j)\}$ and $\phi(v_k)$ is on an intermediate point on e for each $i < k < j$. In other words, all edges in P are used for spanning some edges in G, and there is no other subpath in P joining two endpoints in G by the length condition (2a). Then we say that the linkage P can *simulate* the mechanism G if there is a mapping ϕ satisfying the conditions (1), (2), and (3). This problem can be seen as the Eulerian path problem with edge weights.

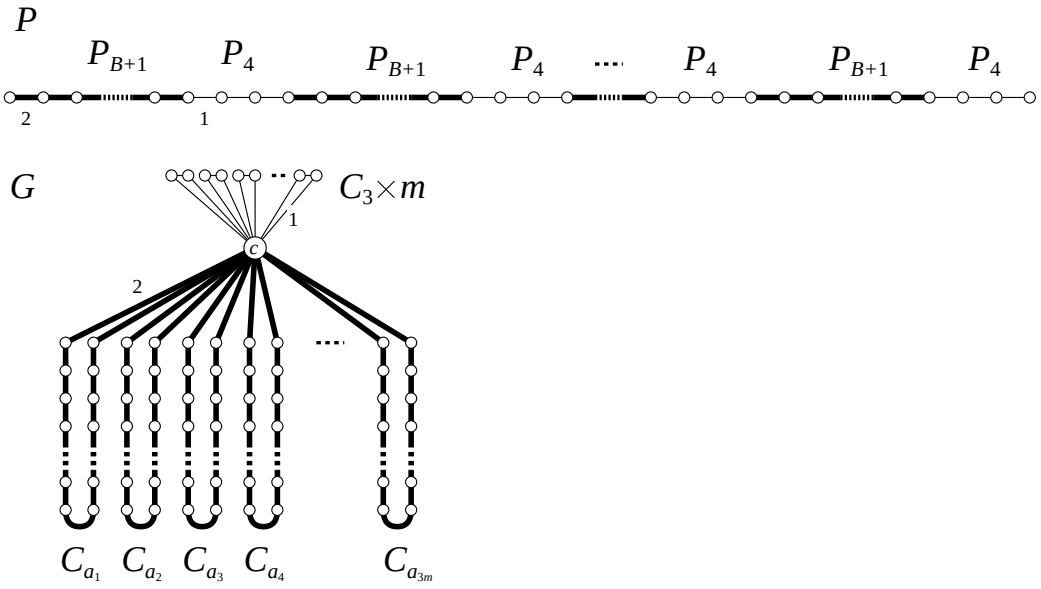

Figure 2.1: Construction of P and G; bold lines are of length 2, and thin lines are of length 1. Each P_{i+1} consists of i edges and each C_i consists of i edges.

2.2 Computational complexity of weighted Eulerian path problem

We show that the weighted Eulerian path problem is strongly NP-hard even if edge lengths are quite restricted.

Theorem 1. *Let P, G, ℓ be a path, an undirected graph, and a length function, respectively. Then the weighted Eulerian path problem is strongly NP-hard even if $\ell(e)$ is either 1 or 2 for any e in P and G.*

Proof. It is easy to see that the problem is in NP. Therefore we show the hardness. We reduce the 3-Partition problem to the weighted Eulerian path problem.

Let P_{B+1} be a path that consists of B consecutive edges of length 2, and P_4 be a path that consists of 3 consecutive edges of length 1. Then the path P is obtained by joining m subpaths P_{B+1} and m subpaths P_4 alternatingly, that is, P is constructed by joining $P_{B+1}, P_4, P_{B+1}, P_4, \ldots, P_4, P_{B+1}$, and P_4. The graph G is constructed as follows. For each i with $1 \leq i \leq 3m$, we construct a cycle C_{a_i} of a_i edges of length 2. We also construct m cycles C_3 of 3 edges of length 1. Then these $4m$ cycles share a special vertex c in common. That is,

G is a cactus that consists of $4m$ cycles, and all vertices have degree 2 except the common vertex c that has degree $8m$. The construction is illustrated in Figure 2.1.

It is easy to see that this is a polynomial-time reduction. Thus, we show that A has a solution if and only if P can simulate G. that is, P can cover G along an Eulerian path in G with satisfying the condition of the linkage simulation. We first observe that no edge of length 2 in P_{B+1} in P can cover a cycle C_3 in G. Therefore, when P covers G, every C_3 of G has to be covered by P_4 in P. Thus, each endpoint of P_4 should be on c in G, and no edge in P_{B+1} can cover edges in C_3. Hence, each subpath P_{B+1} in P covers exactly B edges in the set of cycles C_{a_i} that consists of edges of length 2. Since $B/4 < a_i < B/2$ for each i, each subpath P_{B+1} covers exactly three cycles C_{a_i}, C_{a_j} and C_{a_k} for some i, j, k with $a_i + a_j + a_k = B$. Clearly, each cover for a subpath P_{B+1} gives a subset of A, and the collection of these subsets gives us a solution to the 3-Partition problem and vice versa.

□

Chapter 3

Elastic linkage problem

In the last chapter, we prove that the weighted Eulerian path problem is strongly NP-hard even if P and G consist of edges of lengths only 1 and 2, so we change some conditions and tackle this problem in two different ways.

In this Chapter, we consider the first version of the weighted Eulerian path problem, this is an optimization version of the weighted Eulerian path problem: the elastic linkage problem. In this problem, we allow all edges in P to be elastic to simulate the path G by the path P. The *elastic ratio* of an edge e is defined by $\max\{l'/l, l/l'\}$, where l is the length of the edge $e = \{u, v\}$ in P and l' is the length of the edge $\{\phi(u), \phi(v)\}$ in G. (Intuitively, the length of edge e is changed from l on $P$to l' on G.) For a given graph $G = (V', E')$ and a path $P = (V, E)$, it is easy to observe that P can simulate G (with elastic edges) if and only if G has an Eulerian path and $|V'| \leq |V|$. When G has an Eulerian path by P with elastic edges, the *elastic ratio* of the mapping is defined by the maximum elastic ratio of all edges in P. Then the elastic linkage problem asks to minimize the elastic ratio of the mapping from P to G for given G and P.

Elastic linkage problem from path to path

Input: Two paths $G = (V', E')$ and $P = (V, E)$ with length function ℓ.

Output: a mapping ϕ with minimum elastic (or stretch/shrink) ratio.

3.1 Elastic ratio

In this problem, we allow all edges in P to be elastic to simulate the path G by the path P. We can stretch or shrink each edge in P, thus, we define the *elastic ratio* for edges in P and mappings from path P to path G. The

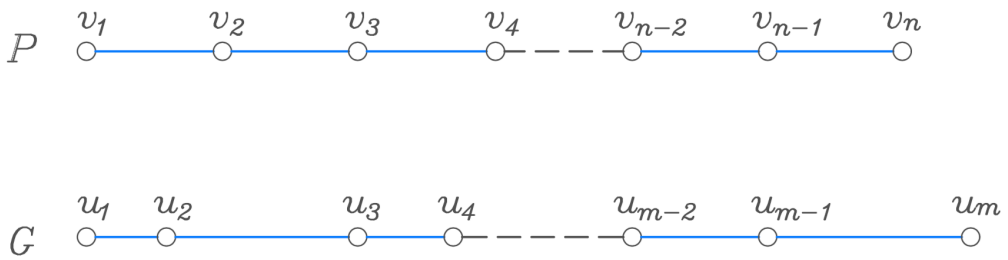

Figure 3.1: A path P and a path G, path P is an elastic linkage, edges in P are allowed to be elastic to fit the vertices of P to ones of G

objective of the elastic linkage problem is to minimize the elastic ratios of mappings from path P to path G for given G and P.

At first, we define the *ratio* of each segment in the elastic linkage P. The *ratio* of an edge e is defined by l'/l, where l is the length of the edge $e = \{u, v\}$ in P and l' is the length of the edge $\{\phi(u), \phi(v)\}$ in G. That is, the length of edge e is changed from $l$on P to l' on G.

The *elastic ratio* of an edge e in path P is defined by $\max\{r, 1/r\}$, where r is the ratio of the edge e in P, thus the *elastic ratio* is always greater than or equal to 1, that is, the *elastic ratio* is the change factor of an edge in the path P.

Let G is a path $(u_1, u_2, \ldots, u_m)$ and P is a path $(v_1, v_2, \ldots, v_n)$, see Figure 3.1.

Without loss of generality, we assume that $m \leq n$. Since each vertex in G should be mapped from only one vertex in P, it should be $\phi(v_1) = u_1$ and $\phi(v_n) = u_{n'}$, otherwise the elastic ratio will be infinity.

The elastic ratio of a mapping from P to G is the maximum among elastic ratios of all edges in P. The elastic ratio of mappings can be minimized in polynomial time, we show a polynomial-time algorithm for this elastic linkage problem based on dynamic programming.

3.2 Elastic ratio of a mapping from path to edge

At first, we consider that the path G consists of only one edge. We show a technical lemma when G is just an edge. In this case, the optimal value is achieved when all ratios are even.

Lemma 1. *Assume that G consists of an edge $e = (u_1, u_2)$. When $P = (V, E)$ is a path, the minimum elastic ratio is achieved when the ratio of each $e \in E$ takes the same value.*

Proof. Assume that the length of the edge $e = (u_1, u_2)$ in G is L, $E = \{e_1, e_2, \ldots, e_{n-1}\}$, and the length of each e_i is l_i. For a mapping ϕ, let r_i be the ratio of the edge e_i for each $i = 1, 2, \ldots, n-1$. Then we have $r_1 l_1 + r_2 l_2 + \cdots + r_{n-1} l_{n-1} = L$.

Assume the maximum among r_i for all $1 \leq i \leq n-1$ is r_k, and the minimum among r_i for all $1 \leq i \leq n-1$ is r_h. Thus, it is obvious that $r_k \geq L/(l_1 + l_2 + \cdots + l_{n-1})$, $1/r_h \geq (l_1 + l_2 + \cdots + l_{n-1})/L$.

According to the definition, the elastic ratio er of this mapping is the maximum among r_i and its reciprocal for all $1 \leq i \leq n-1$. That is, er equals the larger of r_k and $1/r_h$.

When $r_1 = r_2 = \cdots = r_{n-1}$, $\max\{r_k, 1/r_h\}$ takes the minimum. That means the minimum elastic ratio can be achieved if and only if the ratio of each $e \in E$ takes the same value. □

Now we turn to the main theorem.

3.3 Minimal elastic ratio from path to path

Theorem 2. *We can solve the elastic linkage problem from path to path in $O(n^3)$ time.*

Proof. We assume path $P = (v_1, v_2, \ldots, v_n)$, the length of each edge $\{v_i, v_{i+1}\}$ is l_i, path $G = (u_1, u_2, \ldots, u_{n'})$, the length of each edge $\{u_j, u_{j+1}\}$ is w_j, and $n \geq n' \geq 2$.

We define two functions as follows for $i > i' \geq j$:

$$dist(v_{i'}, v_i) = l_{i'} + l_{i'+1} + \cdots + l_{i-1}, \text{ and}$$
$$Ser(v_{i'}, v_i, w_j) = \max\left\{\frac{w_j}{dist(v_{i'}, v_i)}, \frac{dist(v_{i'}, v_i)}{w_j}\right\}.$$

That is, $dist(v_{i'}, v_i)$ is the length of the path $(v'_i, \ldots, v_i)$, and $Ser(v_{i'}, v_i, w_j)$ is the minimum elastic ratio of all edges in the subpath $P' = (v_{i'}, v_{i'+1}, \ldots, v_i)$ of P that covers the edge $\{u_j, u_{j+1}\}$. We first precompute these functions as tables which will be referred in our polynomial-time algorithm. The computation of the corresponding table $Ser[(v_{i'}, v_i), w_j]$ can be done as follows: (1) for each $(v_{i'}, v_i)$ with $i' < i$, compute $dist(v_{i'}, v_i)$ and fill in the table $dist[v_{i'}, v_i]$, (2) for each $j = 0, 1, \ldots, n'$, compute $Ser(v_{i'}, v_i, w_j)$ and fill in

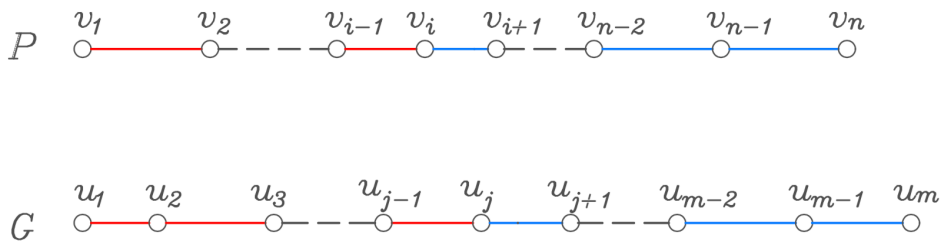

Figure 3.2: We consider to use the subpath $P' = (v_1, v_2, \ldots, v_i)$ of path P to simulate the subpath $G' = (u_1, u_2, \ldots, u_j)$ of path G, and we have $\phi(v_1) = u_1$, $\phi(v_i) = u_j$

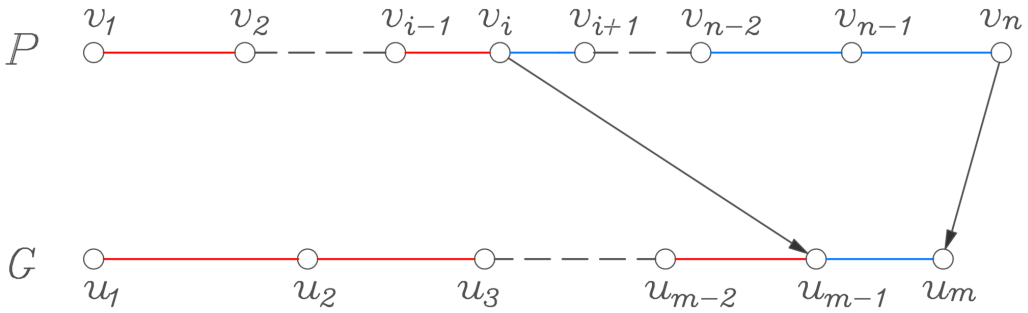

Figure 3.3: We consider to use the subpath $P' = (v_1, v_2, \ldots, v_i)$ of path P to simulate the subpath $G' = (u_1, u_2, \ldots, u_{m-1})$ of path G, and we have $\phi(v_1) = u_1$, $\phi(v_i) = u_{m-1}$

the table $Ser[(v_{i'}, v_i), w_j]$. In (1), each $dist(v_{i'}, v_i)$ can be computed in a constant time by using $dist(v_{i'}, v_i) = dist(v_{i'}, v_{i-1}) + \ell(e_{i-1})$ when we compute the values of this table in the order of $(i - i') = 1, 2, 3, \ldots$. On the other hand, in (2), each $Ser(v_{i'}, v_i, w_j)$ can be computed in a constant time. Therefore, the precomputation can be done in $O(n^3)$ time in total.

To solve the elastic linkage problem efficiently, we define two more functions $ER(v_i, u_j)$ and $M(v_i, u_j)$ as follows. First, $ER(v_i, u_j)$ is the minimum elastic ratio of the mappings from the subpath $P' = (v_1, v_2, \ldots, v_i)$ of P to the subpath $G' = (u_1, u_2, \ldots, u_j)$ of G, see Figure 3.2, Figure 3.3

Then we have the following:

$$ER(v_i, u_j) = \begin{cases} Ser(v_1, v_i, w_1) & \text{when } j = 2 \\ \min_{j-1 \le k \le i-1}\{\max\{ER(v_k, u_{j-1}), Ser(v_k, v_i, w_{j-1})\}\} & \text{when } j > 2 \end{cases}$$

Our goal is to obtain the mapping from P to G with elastic ratio $ER(v_n, u_{n'})$.

Next, $M(v_i, u_j)$ is a sequence of j vertices of path P that represents the mapping with minimum elastic ratio from the subpath P' to the subpath G'. The first and last vertices in $M(v_i, u_j)$ are v_1 and v_i. Then we have the following:

$$M(v_i, u_j) = \begin{cases} (v_1, v_i) & \text{when } j = 2 \\ (M(v_\tau, u_{j-1}), v_i) & \text{when } j > 2, \end{cases}$$

where τ is determined by the following equation;

$$ER(v_i, u_j) = \max\{ER(v_\tau, u_{j-1}), Ser(v_\tau, v_i, w_{j-1})\}.$$

Then our goal is to obtain $M(v_n, u_m)$. The $ER(v_n, u_m)$ and $M(v_n, u_m)$ can be obtained simultaneously by the dynamic programming technique. In the tables of $ER(v_n, u_m)$ and $M(v_n, u_m)$, $ER(v_n, u_m)$ and $M(v_n, u_m)$ are easy to get if the values in the $(m-2)$-nd row are available. The table $ER(v_n, u_m)$ is filled from $j = 2$, that is, for each $v_i = v_1, v_2, \ldots, v_n$, $ER(v_i, u_1) = Ser(v_1, v_i, w_1)$ has already been computed, and accordingly, the first row of table $M(v_n, u_m)$ is $M(v_i, u_1) = (v_1, v_i)$. After filling in the first row of the tables, it is easy to get the values in the second row, the third row, up to the $(m-2)$-nd row and finally get $ER(v_n, u_m)$ and $M(v_n, u_m)$. Each element of the table $ER(v_n, u_m)$ can be computed in $O(n)$ time, and each element of the table $M(v_n, u_m)$ can be computed in constant time. Therefore, the computation of $ER(v_n, u_m)$ and $M(v_n, u_m)$ can be done in $O(n^3)$ time, and the precompution also can be done in $O(n^3)$ time. Thus, the algorithm runs in $O(n^3)$ time, which means the elastic linkage problem can be solved in polynomial time. □

Chapter 4

Traverse problem of a tree by a path

In this chapter, we focus on the traverse problem of G by P. This is another variant of the weighted Eulerian path problem. In this variant, we allow P to cover an edge of G twice which means we allow the mapping to map two subpaths of P to an edge of G, however, we do not allow P to be elastic, or its ratio is fixed to 1. For a given graph $G = (V', E')$ and a path $P = (V, E)$, when all edges have a unit length, it is easy to observe that P can simulate G in this manner if and only if G is connected and $2|E| = |E'|$. We first perform the depth first search on G, and traverse this search tree. We also consider its edge-weighted version as the *traverse problem* for a graph G and a path P.

4.1 General cover problem

Before the traverse problem, we consider the more general case that allows P to cover an edge of G twice or more. This general simulation problem is similar to the following ruler folding problem:

Ruler Folding: Given a polygonal chain with links of integer length $\ell_1, \ldots, \ell_{n-1}$ and an integer L, can the chain be folded flat(reconfigured so that each joint angle is either 0 or π), so that its total folded length is L?

The details of this problem and related results can be found in [3]. In our context, we have the following theorem:

Theorem 3. *The general simulation problem of G by P is NP-complete even if G is an edge.*

Proof. We can reduce the ruler folding problem to our problem by just letting G be an edge of length L. □

We note that the ruler folding problem is weakly NP-complete, and we have a simple pseudo-polynomial-time algorithm that runs in $O(nL)$ time as follows;

Input: Set of integers $S = \{\ell_1, \ldots, \ell_{n-1}\}$ and an integer L
Output: Determine if there is $I \subseteq \{1, \ldots, n-1\}$ with $\sum_{i \in I} \ell_i == L$
begin
  Initialize array $a[0], \ldots, a[L]$ by 0;
  Set $a[0] = 1$;
  foreach $i = 1, \ldots, n-1$ **do**
    foreach $j = 0, \ldots, L$ **do**
      if $a[j] == 1$ *and* $j + \ell_i \leq L$ **then** $a[j + \ell_i] = 1$;
    end
  end
  if $a[L] == 1$ **then** output "Yes";
  else output "No";
end

4.2 Tree traversal problem

4.2.1 NP-completeness results

Now we turn to the traverse problem. Even if a connected graph G has no Eulerian path, when we allow to visit each edge in G twice, we can visit all vertices of G by a path in the depth first search manner. Therefore, we consider the following *traversal problem* as a kind of the robot arm simulation problem:

Input: A path $P = (V, E)$ that forms a path $(v_1, v_2, \ldots, v_n)$, and a graph $G = (V', E')$ with length function $\ell : E \cup E' \to \mathbb{R}$.

Output: A mapping ϕ from P to G such that each edge in G is mapped from exactly two subpaths of P, or "No" if it does not exist.

We first observe that it is linear time solvable when each edge has the unit length just by depth first search. Therefore, it is an interesting question that asks the computational complexity when ℓ maps to few distinct values, especially, ℓ maps to two distinct values.

We give three hardness results about the traversal problem even if the graph G is a simple tree T and the edge lengths are quite restricted.

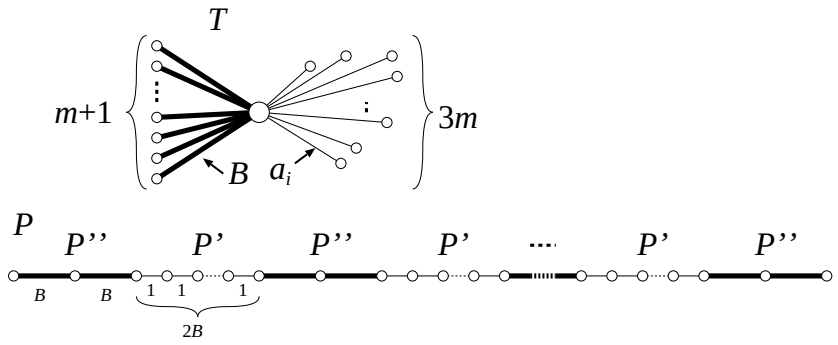

Figure 4.1: Reduction to $K_{1,4m+1}$ and a path P.

Theorem 4. *The traversal problem of a tree T by a path P is strongly NP-complete in each of the following cases: (1) T is a star $K_{1,n-1}$, and P consists of edges of two different lengths. (2) T is a spider, and all edges in G and P are of length p and q, where (2a) p and q are any two positive integers that are relatively prime, or (2b) $p = 1$ and $q = 2$.*

Proof. Since it is clear that each of the problems is in NP, we show their hardness. We will give polynomial-time reductions from the 3-Partition to our problems.
(1) T is a star $K_{1,4m+1}$. Among $4m+1$ edges, the length of $m+1$ edges is B, and the other $3m$ edges have length a_i for each $i = 1, 2, \ldots, 3m$ (Figure 4.1). The construction of P is as follows. Let P' be a path that consists of $2B$ edges of length 1, and P'' be a path that consists of 2 edges of length B. Then the path P is obtained by joining $m+1$ subpaths P'' and m subpaths P' alternatingly, that is, P is constructed by joining $P'', P', P'', P', P'', \ldots, P', P''$ as shown in Figure 4.1.

The construction is done in polynomial time. Thus, we show that the 3-Partition problem has a solution if and only if the constructed cover problem has a solution. We first observe that P'' cannot cover any short edge of length a_i in T. Therefore, each P'' should cover each edge of length B in T twice. Hence all of the endpoints of P''s (and hence P') are on the central vertex of T. Therefore, if P can cover T properly, it is easy to see that each P' should cover three edges of length a_i, a_j, and a_k with $a_i + a_j + a_k = B$ exactly twice. This concludes the proof of (1).
(2a) This reduction is similar to (1). Let P' be a path that consists of $2B$ edges of length p, and P'' be a path that consists of 2 edges of length q. Then the path P is obtained by joining $m+1$ subpaths P'' and m subpaths P' alternatingly. On the other hand, the spider T is obtained by sharing the central vertex of $4m+1$ subpaths (Figure 4.2). Among $4m+1$ subpaths,

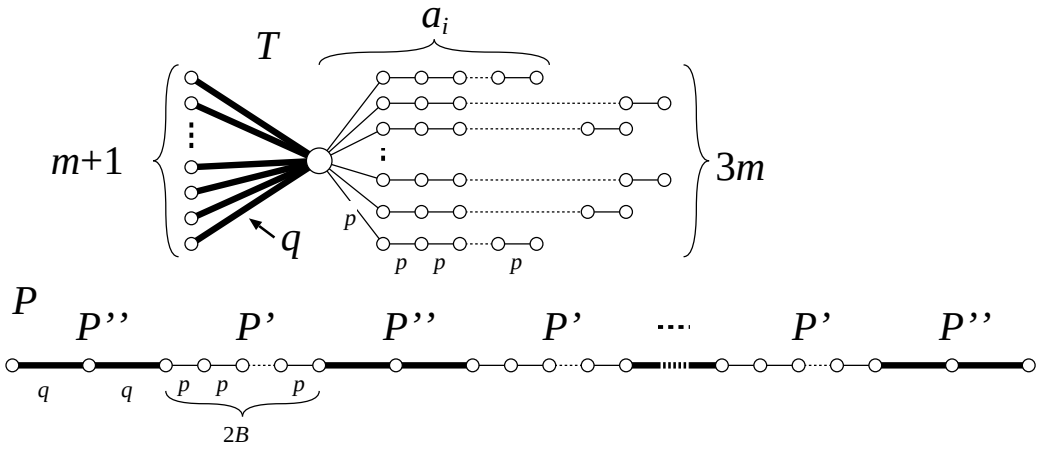

Figure 4.2: Reduction to spider of two different lengths.

$m + 1$ paths are just edges of length q. The other $3m$ subpaths are of a_i edges for each $1 \leq i \leq 3m$, and each edge has length p. Since p and q are relatively prime, P'' cannot cover each of the edges of length p. Therefore, their endpoints (and the endpoints of P') share the central vertex of T. Thus, each P' gives us the solution of the 3-Partition as in (1), which completes the proof of (2a).
(2b) The reduction itself is the same as (2a) except $p = 1$ and $q = 2$. In this case, we observe that no edge of length 1 can be covered by any edge of length 2 in P''. Therefore, each edge of P'' of length 2 should cover the edges of T of length 2. Thus, each P' gives us the solution of the 3-Partition as in (2a), which completes the proof of (2b).

□

4.2.2 polynomial time solvable case

In Theorem 4, we show that the traversal problem of a tree by a path is NP-hard even if we strictly restrict ourselves. Now we turn to show a polynomial-time algorithm for the case that we furthermore restrict. The Figure 4.3 shows the constructions of path P and tree T in this case, T is a star and its edge lengths are of k different values.

Theorem 5. *Let T be a star $K_{1,n'}$ and the number of distinct lengths of its edges is k. Let P be any path of length n. Without loss of generality, we suppose $2n' \leq n$. Then the traversal problem of T by P can be solved in*

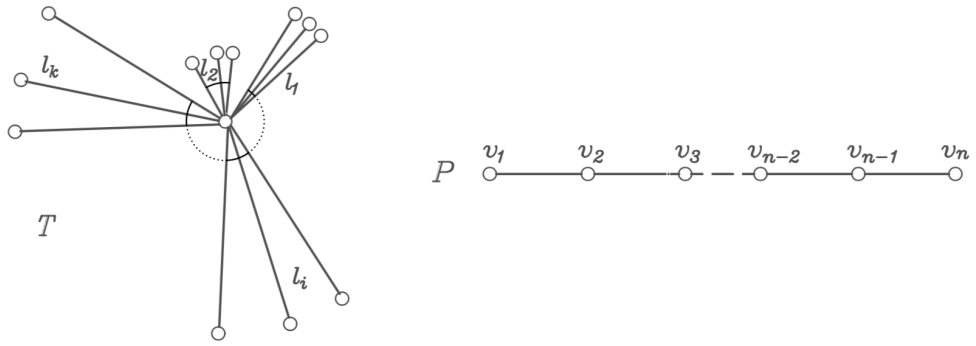

Figure 4.3: Reduction to star T with k different edge lengths and a path P

$O(n^{k+1})$ time and $O(n^k)$ space. That is, it is polynomial time solvable when k is a constant.

Proof. We suppose that each edge of T has a length in $L = \{\ell_1, \ell_2, \ldots, \ell_k\}$, and T contains L_i edges of length ℓ_i for each i. For a vertex v_i in P and length ℓ_j in L, we define a function $\mathrm{pre}(v_i, \ell_j)$ as follows;

$$\mathrm{pre}(v_i, \ell_j) = \begin{cases} v_k & \text{there is a vertex } v_k \text{ with } k < i \text{ on } P \text{ s. t. } \ell(e_k) + \cdots + \ell(e_i) = \ell_j, \\ \phi & \text{otherwise.} \end{cases}$$

We first precompute this function as a table which will be referred to in our polynomial-time algorithm. To distinguish the function $\mathrm{pre}(v_i, \ell_j)$, we refer to this table as $\mathrm{pre}[v_i, \ell_j]$ which uses $O(nk)$ space. The computation of pre[] can be done as follows; (0) initialize pre[] by ϕ in $O(nk)$ time, (1) sort L in $O(k \log k)$ time, and (2) for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k-1$, the vertex v_i fills the table $\mathrm{pre}[v_{i'}, \ell_j] = v_i$. In (2), the vertex v_i can fill $\mathrm{pre}[v_{i'}, \ell_j] = v_i$ in $O(n + k)$ time. Therefore, the precomputing takes $O(n(n + k) + k \log k)$ time in $O(nk)$ space.

Now we turn to the computation for the traversal problem. To do that, we define a predicate $F(d_1, d_2, \ldots, d_k, v_i)$ which is defined as follows: When there is a cover of a subtree T' of T that consists of d_1 edges of length ℓ_1, d_2 edges of length ℓ_2, ..., and d_k edges of length ℓ_k by the subpath $P' = (v_1, v_2, \ldots, v_i)$ when v_1 and v_i are put on the center of T, $F(d_1, d_2, \ldots, d_k, v_i)$ is *true*, and *false* otherwise. (For notational convenience, we define that $F(d_1, d_2, \ldots, d_k, \phi)$ is always false.) Thus, our goal is to determine if $F(L_1, L_2, \ldots, L_k, v_n)$ is true or false. The predicate $F(d_1, d_2, \ldots, d_k, v_i)$ is determined by the following

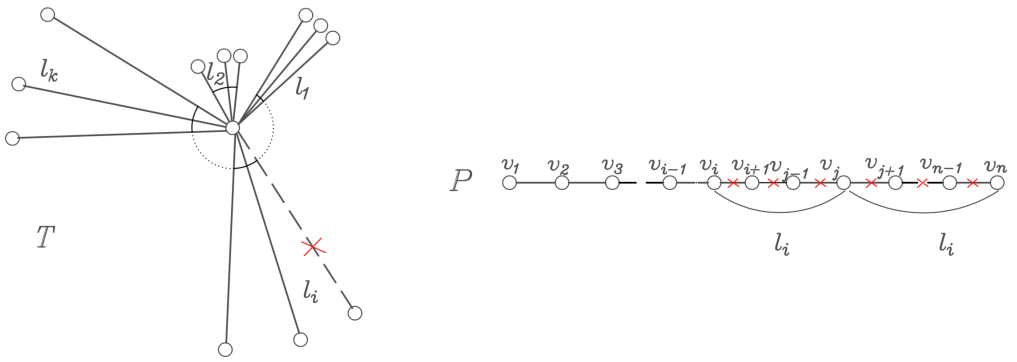

Figure 4.4: For each edge in T, find a subpath of P which covers this edge exactly twice

recursion;

$$F(d_1, d_2, \ldots, d_k, v_i)$$
$$= \bigvee_{1 \le j \le k} ((\mathrm{pre}(v_i, \ell_j) \neq \phi) \wedge F(d_1, \ldots, d_j - 2, \ldots, d_k, \mathrm{pre}(\mathrm{pre}(v_i, \ell_j), \ell_j))).$$

That is, for the vertex v_i, we have to have two vertices $v_{i'} = \mathrm{pre}(v_i, \ell_j)$ and $v_{i''} = \mathrm{pre}(\mathrm{pre}(v_i, \ell_j), \ell_j)$ such that $\ell(e_{i'}) + \ell(e_{i'+1}) + \cdots + \ell(e_i) = \ell_j$ and $\ell(e_{i''}) + \ell(e_{i''+1}) + \cdots + \ell(e_{i'}) = \ell_j$ for some j with $1 \le j \le k$. The correctness of this recursion is trivial. This idea is shown in Figure 4.4.

The predicate $F(L_1, L_2, \ldots, L_k, v_n)$ is computed by a dynamic programming technique. That is, the table $F[d_1, d_2, \ldots, d_k, v_i]$, corresponding to the predicate $F(L_1, L_2, \ldots, L_k, v_n)$, is filled from $d_1 = 0, d_2 = 0, \ldots, d_k = 0$ for the center vertex c, which is true. Then, we increment in the bottom up manner; that is, we increment as $(d_1, d_2, \ldots, d_k) = (0, 0, \ldots, 0, 1)$, $(0, 0, \ldots, 1, 0)$, $\ldots$, $(0, 1, \ldots, 0, 0)$, $(1, 0, \ldots, 0, 0)$, $(0, 0, \ldots, 0, 2)$, $(0, 0, \ldots, 1, 1)$, $\ldots$, $(0, 1, \ldots, 0, 1)$, $(1, 0, \ldots, 0, 1)$, and so on. The number of combinations of $(d_1, d_2, \ldots, d_k)$ is $L_1 \cdot L_2 \cdot \cdots \cdot L_k \le n'^k = O(n^k)$, and the computation of $F[d_1, d_2, \ldots, d_k, v_i]$ for the $(d_1, d_2, \ldots, d_k)$ can be done in linear time. Therefore, the algorithm runs in $O(n^{k+1})$ time and $O(n^k)$ space. □

Chapter 5

Conclusion

In this research of the linkage simulation problem, we first consider the following situation: a general target mechanism which is modeled by a graph $G = (V', E')$, and a robot arm modeled by a linkage $P = (V, E)$ with length function $\ell : E \cup E' \to \mathbb{R}$. The mission is to simulate the target graph G by the given linkage P. That is, we should find the mapping ϕ from P to G which satisfies the following conditions: each vertex of G should be mapped from some vertices of P and each edge of G should be mapped from a subpath of P by ϕ.

The decision problem of linkage simulation problem asks if there is an Eulerian path of G spanned by P. This problem is a generalization of the Eulerian path problem, which we call the weighted Eulerian path problem. In this problem, when we use the linkage P to simulate the target mechanism G, P can only cover an edge of G once and each edge length is fixed. This problem is linear time solvable if P and G consist of unit lengths edges. However, we first prove that this problem is strongly NP-hard even if edge lengths are quite restricted. Actually, this problem is strongly NP-hard even if P and G consist of edges of lengths only 1 and 2. We reduce the 3-Partition problem to the weighted Eulerian path problem to show the NP-hardness of this problem, where the 3-Partition problem is well-known NP-complete. We show the constructions of path P and graph G, the path P consists of two different kinds of subpath with edges of lengths 1 and 2, and the graph G is a cactus that consists of $4m$ cycles, m cycles consist of edges of length 1, $3m$ cycles consist of edges of length 2, we show that the multiset A of the 3-Partition problem has a solution if and only if P can simulate G, which means path P can cover the target graph G along an Eulerian path in G with satisfying the condition of the linkage simulation.

Therefore, the weighted Eulerian path problem is tackled in two different ways, two different simulation problems are considered: the elastic linkage

problem and the traverse problem of a tree by a path.

The first problem is the elastic linkage problem, this is an optimization version of the weighted Eulerian path problem. In weighted Eulerian path problem, the edge lengths of P are all fixed, they cannot be changed, but in this variant, all edges in P are allowed to be elastic to simulate the target graph G by the path P, that is, we can stretch or shrink the edges in the elastic linkage P. This situation is natural not only in the context of the robot arm simulation but also in the approximation algorithm, so far, we consider the elastic linkage problem for two paths P and G, we use the elastic linkage P to simulate the path G. Firstly, the elastic ratio of edges and mappings in the simulation process are defined, the elastic ratio of an edge e is defined by $\max\{l'/l, l/l'\}$, where l is the original length of the edge in P and l' is the length of the edge in G. Intuitively, the length of the edge is changed from l on P to l' on G. Thus the elastic ratio is always greater than or equal to 1, that is, the elastic ratio is the change factor of the edge in the path P. When P simulates G with elastic edges, the elastic ratio of the mapping is defined by the maximum elastic ratio of all edges in P. The objective of the elastic linkage problem is to minimize the elastic ratio of the mapping from a path P to a path G for given P and G. We start from a simple case which the path G consists of only one edge, and in this case, we proved that the minimum elastic ratio is achieved when all ratios of edges in path P take the same value. Then we proved that the elastic linkage problem can be solved in polynomial time by dynamic programming.

The second problem is the traverse problem of a tree by a path. In weighted Eulerian path problem, P can only cover an edge of G once, in this variant, P is allowed to cover an edge of G twice or more. In this situation, P can simulate G even if G does not have an Eulerian path. In this case, we do not allow P to be elastic, or its ratio is fixed to 1. Firstly, the general simulation problem is proved to be weakly NP-hard, even if G is an edge. It is similar to the ruler folding problem, which is weakly NP-complete. Thus, we consider more restricted cases. From the viewpoint of graph theory, it is natural to consider the case that G is a tree. For a given tree G and a path P with edge lengths, the traverse problem asks if G has a trail by P such that each edge of G is traversed exactly twice. When G is a tree, the problem is in a simple form, P simulates G by traversing each edge twice in the unique spanning tree of G, or G itself. However, this problem is still strongly NP-hard even in quite restricted cases; (1) G is a star, and P consists of edges of only two different lengths, and (2) G is a spider, and all edges are of two different lengths. We also reduce the 3-Partition problem to the traverse problem of a tree by a path to show the NP-hardness of these restricted cases, and we also give the constructions of path P and tree T. On the other

Problem	Conditions	Results
Weighted Eulerian Path Problem	fixed edge-length,cover once	Strongly NP-hard in quite restricted case
Elastic Linkage Problem	non-fixed edge length, cover once	Polynomial-time solvable
Covering problem of a tree by a path:		
General cover problem	fixed edge-length, cover twice or more	NP-complete
Tree traversal problem	fixed edge-length, cover exactly twice	NP-completeness results;Polynomial-time algorithm

Table 5.1: Current results for the linkage simulation problems

hand, the problem is polynomial time solvable when G is a star and its edge lengths are of k different values.

The current results that we obtained for linkage simulation problems are as shown in Table 5.1. When we use path P to simulate target graph G, we just allow P to cover an edge in G once and all edge-lengths are fixed. We show that this problem is strongly NP-hard even if edge lengths are quite restricted. We thus consider two variants of the weighted Eulerian path problem. The first variant is the elastic linkage problem. In this variant, P can only cover an edge of G once, but each edge length is not fixed, we allow edges in P to be elastic to fit vertices in G, the goal is to minimize the elastic ratio. This is an optimization problem, we show that it can be solved in polynomial time by dynamic programming when G is a path. The other variant is covering problem of a tree by a path. In this variant, each edge length is fixed but we allow P to cover an edge of G twice or more. We first show that the general cover problem of G by P is NP-complete even if G is an edge, we thus consider more restricted cases, we assume P cover each edge in tree T exactly twice, and each edge length is fixed. We show this problem is still strongly NP-hard in two restricted cases and we also show another case that can be solved in polynomial time by dynamic programming.

In fact, there are more variants of weighted Eulerian path problem to be solved, for example, advanced elastic linkage problem. In this problem, when we use path P to simulate target graph G, we not only allow P to cover an edge of G twice or more but also allow the edges in P to be elastic. The goal is to minimize the elastic ratio of P, but so far there is no good way to solve this problem.

However, the research results that we got provide some help or progress for other research in this field. In this research, our focus is on one-dimensional (1D) linkage problems, as we mentioned before, the linkages are useful models for robot arms and they also have the potential for protein folding problems. A protein is composed of a chain of amino acid and residues joined by peptide bonds. It can be modeled for many purposes as a polygonal chain representing the protein's "backbone", with three atom vertices per residue and adjacent atoms connected by bond links. The linkage simulation is possible to be used to explore the research direction of protein folding on geometry. We will continue our research on linkage and try to investigate more applications for it.

Bibliography

[1] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*, Cambridge University Press,(2007)

[2] Orna Kupferman and Gal Vardi. Eulerian Paths with Regular Constraints, *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 58, pp. 62:1–62:15(2016)

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*, Freeman,(1979)

Publication

[1] Tianfeng Feng, Takashi Horiyama, Yoshio Okamoto, Yota Otachi, Toshiki Saitoh, Takeaki Uno, and Ryuhei Uehara . Computational Complexity of Robot Arm Simulation Problems, *29th International Workshop on Combinational Algorithms (IWOCA 2018)*, Lecture Notes in Computer Science, Vol. 10979, pp. 177-188, 2018/07/16-2018/07/19, Singapore.