STUDY ON TWO OPTIMIZATION PROBLEMS: LINE COVER AND MAXIMUM GENUS EMBEDDING

A Thesis

by

CHENG CAO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2012

Major Subject: Computer Science

STUDY ON TWO OPTIMIZATION PROBLEMS: LINE COVER AND MAXIMUM GENUS EMBEDDING

A Thesis

by

CHENG CAO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jianer Chen
Committee Members,	Anxiao Jiang
	Hongbin Zhan
Head of Department,	Duncan Walker

May 2012

Major Subject: Computer Science

ABSTRACT

Study on Two Optimization Problems:
Line Cover and Maximum Genus Embedding. (May 2012)
Cheng Cao, B.S., Huazhong University of Science and Technology
Chair of Advisory Committee: Dr. Jianer Chen

In this thesis, we study two optimization problems which have a lot of important applications in diverse domains: Line Cover Problem (LCP) in Computational Geometry and Maximum Genus Embedding (MGE) in Topological Graph Theory.

We study LCP whose decision version is known NP-Complete from the perspective of Parameterized Complexity, as well as classical techniques in Algorithm Design. In particular, we provide an exact algorithm in time $O(n^3 2^n)$ based on Dynamic Programming and initiate a dual problem of LCP in terms of Linear Programming Duality. We study the dual problem by applying approximation and kernelization, obtaining an approximation algorithm with ratio $k-1$ and a kernel of size $O(k^4)$. Then we survey related geometric properties on LCP. Finally we propose a Parameterized Algorithm to solve LCP with running time $O^*(k^k/1.35^k)$.

We explore connections between the maximum genus of a graph and its cycle space consisting of fundamental cycles only. We revisit a known incorrect approach of finding a maximum genus embedding via computing a maximum pairing of intersected fundamental cycles with respect to an arbitrary spanning tree. We investigate the reason it failed and conclude it confused the concept of *deficiency*. Also, we characterize the upper-embeddablity of a graph in terms of maximum pairings of intersected fundamental cycles, i.e. a graph is upper-embeddable if and only if the number of maximum pairings of intersected fundamental cycles for any spanning tree

is the same. Finally, we present a lower and an upper bound of the maximum number of vertex-disjoint cycles in a general graph, $\beta(G) - 2\gamma_M(G)$ and $\beta(G) - \gamma_M(G)$, only depending on maximum genus and cycle rank.

To my family

ACKNOWLEDGMENTS

I would like to give my greatest gratitude to my advisor, Dr. Jianer Chen. Although having a hard enough time, he still would spend time every week discussing with me my research. Also, he usually taught me the way of doing research, combining with his own experiences. When I got stuck somewhere, he inspired me by raising ideas from different angles. When I obtained certain results, he encouraged me to do better. Besides guiding me on my research, he often told me stories from his own life experiences. For me, Dr. Chen is not only my academic advisor, but also my life mentor.

I would like to give many thanks to Dr. Anxiao Jiang and Dr. Hongbin Zhan. Both of them gave me valuable advice on my research and thesis. I took the course, "design and analysis of algorithms" taught by Dr. Jiang and learned a lot, which aroused my interest on Theoretical Computer Science. Dr. Zhan carefully corrected my thesis and asked questions from perspectives of other areas, which helped me to better understand specific applications of problems studied in my research.

I would like to thank all the members in our research group: Yixin Cao, Jiahao Fan, Shiyu Hu, Nikhil Pandey, and Praveen Tiwari, for all those valuable discussions.

Finally, I would like to thank the Department of Computer Science & Engineering of Texas A&M University who built a good environment to facilitate my graduate study.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Abundant important applications of NP-hard problems have attracted intensive studies for decades and quite a few techniques have been proposed to tackle NP-hard problems such as exact algorithms, approximation algorithms, randomized algorithms, heuristics, and parameterized algorithms. In the area of *Parameterized Complexity*, as it has been observed that there exist many instances of NP-hard problems whose inherent parameters are small enough compared to their input sizes, parameterized algorithms, which usually offer implementably practical alternatives to those problems having been regarded as intractable from the perspective of *Complexity Theory*, have become more and more popular and powerful in the last decade.

In this thesis, at first, we introduce an optimization problem whose decision version is known as NP-Complete in *Computational Geometry* called Line Cover Problem (LCP) (given a set of points on the Euclidean plane $\mathbb{R}^2$, find minimum straight lines on the plane to cover all the points) and then tackle this problem via developed techniques in Parameterized Complexity as well as other typical approaches in algorithm design, some of which have never been touched. Then, as the motivation of studying another classical optimization problem Feedback Vertex Set (FVS) (find a minimum set of vertices in a graph whose removal produces a graph with no cycle), we raise another optimization problem in *Topological Graph Theory* called Maximum Genus Embedding (MGE) (given a graph find the largest genus of an orientable surface on which the given graph admits a cellular embedding) [26] which is solvable in polynomial time by existed algorithms. We explore the relationship

---

This thesis follows the style of *IEEE Transactions on Computers*.

between *fundamental cycles* and *maximum genus* starting from restudying a known incorrect approach of finding a maximum genus embedding via computing a maximum pairing of intersected fundamental cycles with respect to an arbitrary spanning tree.

In this chapter, we give a brief introduction on those two optimization problems from aspects of applications, significance and previous research work. In following sections, there may be a few concepts and terms which will be introduced and explained in details in the second chapter. At the end of this chapter we describe the organization of this thesis.

A. Line Cover Problem

LCP is a fundamental problem in *Computational Geometry* and emerges in direct connections to variations of Traveling Salesman Problem (TSP) with ingredients of covering points with lines. Moreover, naturally it has a very close relationship with mathematical problems in *Linear Algebra* even when extending to instances of hyperplanes from two dimension space. Also, there are many practical applications with aim of covering or moving objects in a straight line because turns are considered costly. For instance, collecting balls by robots, dropping supplies by helicopters, and highway construction are illustrative examples since moving in a straight line could accelerate and save dramatically.

The decision version of LCP, asking if k covering lines are enough where k is a non-negative integer, was proved NP-Complete thirty years ago in [34]. Then an approximation algorithm was proposed with a factor of $\Theta(logn)$ in [30]. Then it was proved *Fixed-Parameter Tractable* (FPT) by Langerman and Morin in [33] where they also provided a *parameterized algorithm*, involving two typical techniques in designing parameterized algorithms: *bound-search tree* and *kernelization*. Based on

the algorithm given by Guibas et al. [24], their approach had the time complexity of $O(nk + k^{2(k+1)})$ where n is the input size and k is the parameter. Later, Grantson and Levcopoulos [20] solved the problem in $O^*(k^{2k}/4.84^k)$[1], along with two reduction rules presented. The current best result is $O^*(k^k/1.35^k)$ proposed by Wang et al. [46]. Generally speaking, their parameterized algorithm was derived from bound-search tree based on enumerations, which leaves certain opportunities of employing either other algorithmic methods or related geometric properties to improve it. Additionally, as a fundamental problem in Computational Geometry, in [33] LCP was abstracted as a covering problem that could model a number of concrete geometric covering problems [19]. The problem considered in [46, 33] is to cover a given set of points in the Euclidean space $\mathbb{R}^m$ by k hyperplanes of dimensions bounded by d where $d \leq m$. The parameter k is the number of hyperplanes allowed in the covering and the parameter d denotes the dimension. Also, another geometric covering problem of covering points with spheres can be fitted into the general framework. In this thesis we focus on the case of the Euclidean plane $\mathbb{R}^2$. However, extending to cases of hyperplanes from the case of two dimension space should not be too hard. We may consummate the general framework of geometric covering problems in the future work.

B. Maximum Genus Embedding

Graph embedding is also a fundamental problem with a plenty of applications in diverse domains directly such as *Computer Graphics* and *Computer Vision*. In this thesis we focus on the non-planar embedding which has been a dominating concern in Topological Graph Theory. Intuitively speaking, a non-planar embedding is an

[1]Following the convention we use $O^*(f(k))$ to denote the bound $O(f(k)n^{O(1)})$

embedding of a graph on a surface with a number of handles. As mentioned above, our biggest motivation of studying MGE is from FVS which is one of the well-known optimization problems and the history of research on FVS has been going over more than forty years. One of its important applications is about *deadlock recovery* in *Operating System*.

For the general background of Topological Graph Theory we will introduce in chapter 2 and for more details we refer books of [22, 47]. Graph embeddings, especially planar embeddings, have been studied extensively over decades. Starting from '80s of the last century people began to pay attention on non-planar embeddings. The complexity of the problem *minimum genus embedding* with aim of finding embeddings of minimum genus rather than maximum, as one of the problems in Garey and Johnson's list [18], was proved NP-Complete by Thomassen [45] for the decision version. For *maximum genus embedding*, many researchers [36, 41, 42, 50] have also proposed related theorems and concepts after [39]. One important result is that Xuong [49] characterized the maximum genus embedding in terms of components of the complements of spanning trees, which educed subsequent concepts as follows. The *deficiency* $\xi(G,T)$ of a connected graph G with respect to a spanning tree T is the number of *cotree components* in $G-T$ which contain odd edges. A spanning tree T is called a *Xuong tree* if $\xi(G,T)$ is equal to the minimum value of deficiency taking over all possible spanning trees of G [9]. Then subsequent results on deficiency were proposed and connected to *cycle rank*, which was the starting point of studying combinatorial characterizations of maximum genus. Then Furst et al. [17] presented a polynomial-time algorithm to compute a maximum genus embedding based on a reduction to *cographic matroid parity*, which is also the current best algorithm to solve MGE.

A graph is called *upper-embeddable* if it has a maximum-genus embedding with

one or two faces [17]. Kundu [31] proved that every 4-edge connected graph is upper-embeddable, while Jungerman [27] showed that there are 3-edge connected graphs which are not upper-embeddable as examples. Later Skoviera [43] presented more classes of upper-embeddable graphs.

Besides exact algorithms, many researchers have been investigating in deriving a lower bound on the maximum genus of graphs that are not upper-imbeddable [9]. From the perspective of connectivity, classifying graphs by vertex and edge connectivities below four, Chen et al. [9, 7] presented the bounds and showed the tightness. Moreover, via the known theorem that a graph is 2-edge connected if and only if it has an *ear decomposition*, Chen and Kanchi [8] showed that constructing a maximum-paired ear decomposition of a graph and constructing a maximum-genus embedding of the graph are polynomial-time equivalent. At the same time, there were studies from the perspective of topological operations. Gross and Rieper [21] revealed that although there may exist arbitrarily deep traps among local minima there cannot occur any local maxima, which coincided with the complexity gap between computing a minimum genus embedding and a maximum genus embedding. Chen [6] presented a new data structure which efficiently supports all on-line operations for general graph embeddings. Focusing on the class of cubic graphs, Huang and Liu [26] proved that the maximum genus of a cubic graph is equal to the size of a maximum *nonseparating independent set* in the graph. Then, Cao et al. [5] proposed an improved parameterized algorithm for FVS, handling the case of 3-regular by the same reduction to cographic matroid parity, which directly leads us to revisit MGE. Apart from what we mentioned above, there have been other research studying connections between maximum genus and other combinatorial characterizations of a graph such as *diameter, girth* and *chromatic number*. In this thesis we pay more attention on *fundamental cycles*. A fundamental cycle $T(e)$ with respect to a spanning tree T of a graph G

is the only cycle that passes through a cotree edge e and is contained in $T + e$ [29]. The idea of using properties related to fundamental cycles to characterize maximum genus started in [37] where the maximum pairings of intersected fundamental cycles were considered. We will discuss more details in chapter 4.

C. Organization of Thesis

This thesis is organized by five chapters as follows. Chapter I is an introductory chapter. The second chapter contains necessary background, including parameterized complexity, graph, surface, embedding, and so on. Chapter III is to discuss our research on LCP from diverse angles. We consider techniques in parameterized algorithms such as bound-search-tree and kenerlization. In the meanwhile, we also study LCP from exact algorithms, duality and geometry properties, and finally we propose a parameterized algorithm. Chapter IV is about MGE. We investigate the interrelations between maximum genus and fundamental cycles. Furthermore, we consider connections in general graphs between upper-imbeddability, maximum genus and maximum vertex-disjoint cycles. The final chapter is a succinct summary.

CHAPTER II

PRELIMINARIES

In this chapter, we introduce preliminary background which is related to subject topics of this thesis, *Parameterized Complexity* and *Graph Embedding*. We will not go to deep levels but all the necessary concepts and notations mentioned in this thesis. First, we introduce several fundamental definitions in Parameterized Complexity and two frequently-used methods, kernelization and bound-search tree. Then, we describe notations and concepts in *Topological Graph Theory* such as graph, surface, and embedding.

A. Parameterized Complexity

Definition A.1. [38] *A* parameterized problem *is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a finite alphabet. The second component is called the parameter of the problem.*

Research over a decade have observed that quite a few optimization problems can be converted to corresponding parameterized problem with inherent parameters in many instances. Furthermore, those parameters usually are small enough compared to the input size. We have the following definition.

Definition A.2. [38] *A parameterized problem L is* fixed-parameter tractable *if it can be determined in $f(k) \cdot n^{O(1)}$ time whether or not $(x, k) \in L$, where f is a computable function only depending on k. The corresponding complexity class is called FPT.*

If a parameterized problem has been proved FPT, the corresponding algorithms are called *parameterized algorithms* which usually offer implementably practical alternatives to those problems regarded as intractable. For example, the current best parameterized algorithm of LCP [46] has complexity of $O^*(k^k/1.35^k)$ where the star

here means we only care about the part of "$f(k)$" only depending on the parameter k. Since k is relatively much smaller than the input size n, a parameterized algorithm can become relatively efficient. On the other hand, given an algorithm whose running time is $O(n^k)$, it may still run slowly as n is fairly large and k is not so small as a constant. Therefore, a parameterized algorithm can be practical for instances with small parameters. We introduce two typical techniques of designing parameterized algorithms mentioned in this thesis.

Definition A.3. Kernelization [38] *Let $\mathcal{L}$ be a parameterized problem, that is, $\mathcal{L}$ consists of input pairs (I, k), where I is the problem instance and k is the parameter. Kernelization means to replace the instance (I, k) by a reduced instance (I', k') (called problem kernel) such that $k' \leq k$, $|I'| \leq g(k)$ for some function g depending only on k and $(I, k) \in \mathcal{L}$ if and only if $(I', k') \in \mathcal{L}$. Furthermore, the reduction from (I, k) to (I', k') must be computable in polynomial time $T_K(|I|, k)$. The function $g(k)$ is called the size of the problem kernel.*

By the definition, the key step in kernelization is to derive reduction rules to make sure the "if and only if" condition holds between original instance and reduced instance. In the meanwhile, the reduction step should be computable in polynomial time and the size of reduced instance should only depend on the given parameter k. Take LCP as an example again, in [20] a kernel of size $O(k^2)$ was provided based on several reduction rules. The most powerful part of kernelization is to compute a much smaller instance, i.e. only depending on the small parameter k, in polynomial time and then we can only focus on the small kernel which may be tackled easily and solved quickly.

Although there are other algorithmic approaches in design of parameterized algorithms such as *iterative compression*, *greedy localization* and *coloring coding*, *bound-*

search tree may be the most widely used one which can be also applied to derive exact algorithms. Given an instance of a problem, an algorithm can be derived by abstracting it onto a tree structure as follows. If current instance satisfies certain conditions, solves it directly. Otherwise, the instance is reduced into smaller sub-instances and then recursively each sub-instance is called. The process can be abstracted as a recursive tree, i.e. a DFS tree. Every node represents an instance and the root is the original instance whose children are all the reduced sub-instances. The running time of the whole recursive algorithm is equal to the product of number of leaves and the maximum summation of time for all all nodes in a path from root to a leaf taking over all paths. In any parameterized algorithm, the number of leaves can be bounded by some measurement depending on k only, and the summation of time for nodes in any path from root to a leaf can be bounded in polynomial time. We enumerate all the possible cases and consider every path in the search tree. If it shows that there is no solution after solving each leaf then we output "no solution", and otherwise it brings an exact solution.

In [5], a parameterized algorithm on a variation of FVS called *disjoint-FVS* was provided based on bound-search tree. After preprocessing all the vertices with degree no larger than two, based on bound-search tree, branching on all the vertices whose degree is larger than three, leaving only vertices with degree exact three. Now every node in the search tree represents a sub-instance of a cubic graph, which can be solved directly in polynomial time as same as existed algorithms of MGE. The relationship between FVS and MGE motivates us to convert to restudy MGE, a different optimization problem in Topological Graph Theory.

B. Graph Embedding of Maximum Genus

1. Graphs, Surfaces and Embeddings

In Discrete Mathematics, a graph is an abstract representation of a set of objects and pairs of the objects which are connected by links [3]. In Computer Science, these interconnected objects are usually called nodes or *vertices*, and the links, i.e. line segments or curves, connecting some pairs of vertices are called *edges*. In particular, in this thesis, a graph G is represented by a set $V(G)$ of vertices and a set $E(G)$ of edges consisting of unordered pairs of vertices. Given an edge $e = (u, v)$, we call the vertices u, v are two *ends* of e and e *joins* vertices u and v. The edges may be *directed* or *undirected*. An edge joining a vertex with itself is called a *self-loop*. If more than one edges join the same two vertices we call them *multi-edges*. The degree of a vertex is the number of edges incident to it. If two edges are incident to a common vertex, we call them *adjacent edges*. A graph is called a *cubic graph* or *3-regular graph* if every vertex has degree three. In an undirected graph, two vertices are called *connected* if the graph contains a path between those two vertices and called *disconnected* otherwise. A graph is *connected* if each pair of vertices is connected and called *disconnected* otherwise. A graph is *simple* if it contains neither multi-edges nor self-loops. A vertex in a graph is called a *cutpoint* of the graph if removal of that vertex disconnects the graph. Similarly, an edge is a *cutedge* if the graph is disconnected after the removal of that edge. A graph is *k-edge-connected* (*k-vertex-connected*, respectively) if for any $0 < h < k$, removing arbitrary h edges (any h vertices) can not disconnect the given graph. Other terminology and notations not explicitly explained in this thesis generally conform to those in [3]. The graphs considered in this thesis are undirected, connected and may contain multi-edges and self-loops unless stated explicitly.

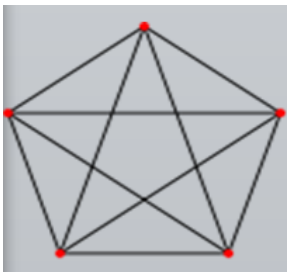


Fig. 1. K_5

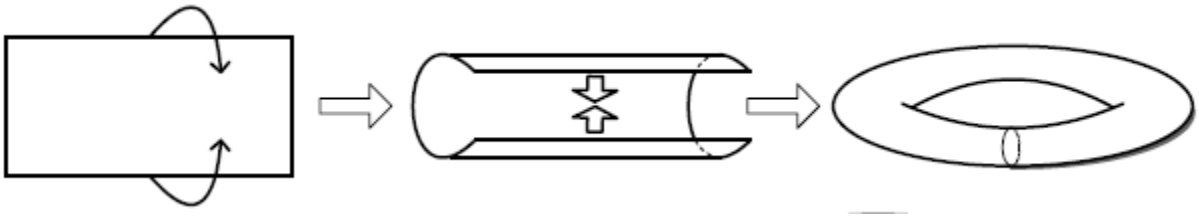


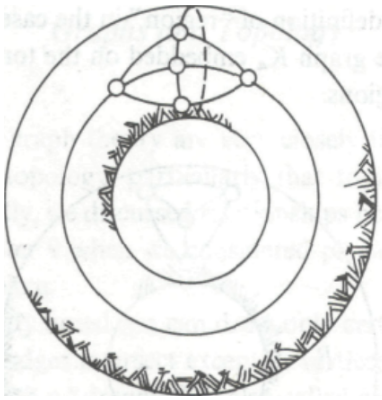
Fig. 2. [32] Construction of a torus from a rectangle

To make the contents to be discussed below understandable to those who are not familiar with Topological Graph Theory, here we illustrate *graph planar embedding* and *graph non-planar embedding* by examples and figures. Take K_5 shown in Fig. 1 as the example.

It is known that K_5 cannot be embedded on a plane, or intuitively speaking, cannot be drawn on a plane without any crossing of edges. However, starting from a sheet of rectangle paper, we can make it by some tricks as shown in Fig. 2.

Now we can draw K_5 on the surface in Fig. 2 with no edge crossing as shown in Fig. 3. Therefore, although K_5 does not have any planar embedding, there are non-planar embeddings. As a matter of fact, K_5 has an embedding with only one face on certain surfaces, though the embedding in Fig. 3 contains five faces.

Our terminology in this thesis is compatible with books of [22, 47]. In Topological Graph Theory, a *surface*, denoted by S, is a compact two-manifold without



[1]

Fig. 3. An embedding of K_5 on a torus without crossing of edges

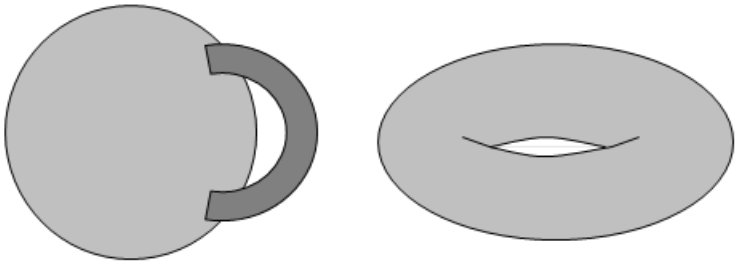


Fig. 4. A handle attached on a sphere and a torus

boundaries. Surfaces can be divided into two classes of *orientable* and *non-orientable*. A surface is orientable if it does not contain a Möbius band [17]. In this thesis we only deal with orientable surfaces which can be viewed as a sphere attached with cylinders called *handles* as shown in Fig. 4. The *genus* of an orientable surface is the number of attached handles.

A *cellular embedding* of a graph G on a surface S is a one-to-one mapping $\phi : G \to S$ such that each component of $S \setminus \phi(G)$ is homeomorphic to an open disc which is defined as a *face* of G respect to the embedding ϕ [25]. In this thesis we only discuss cellular embeddings on orientable surfaces. The *Euler polyhedral equation*

[1]The figures on this page are quoted from Internet

$|V(G)| - |E(G)| + |F| = 2 - 2\gamma(S)$ holds for all cellular embeddings where $|F|$ is the number of faces of the corresponding embedding ϕ and $\gamma(S)$ denotes the genus of surface S. In this thesis, we may call "the genus of a graph is g" or "a graph has genus g", which means the graph can be embedded on a surface of genus g.

2. Maximum Genus Embedding

The *maximum genus* of a graph G, commonly denoted by $\gamma_M(G)$, is the maximum integer g such that there exists a cellular embedding of G on the orientable surface S of genus g. Xuong [48] characterized a maximum genus embedding in terms of *cotree components* which are components of the complements of a spanning tree. Xuong proved that computing the maximum genus can be reducible to computing a combinatorial invariant [17] called *deficiency*. The *deficiency* $\xi(G,T)$ of a connected graph G with respect to a spanning tree T is the number of components of odd edges in the cotree $G - T$. The deficiency $\xi(G)$ of a graph G is the minimum value of deficiency taking over all spanning trees of G [8]. A spanning tree T is called a *Xuong tree* if $\xi(G,T) = \xi(G)$. Let $\beta(G)$ denote the cycle rank of G which is the number of edges in any cotree, that is, always $|E(G)| - |V(G)| + 1$.

Theorem B.1. [48] *Let G be a connected graph. We have the following equation:*

$$\gamma_M(G) = \frac{\beta(G) - \xi(G)}{2}$$

By Euler's polyhedral equation, taking over all the embeddings of a given graph, the minimum faces is equal to the deficiency plus one. A graph G is said to be *upper-embeddable* if $\gamma_M(G) = \lfloor \frac{\beta(G)}{2} \rfloor$ [25].

Theorem B.2. [49] *Let G be a connected graph. G is upper-embeddable if and only if $\xi(G) = 0$ or 1.*

A *rotation* of a vertex is a cyclic permutation of the edge-ends incident on that vertex [6]. A list of rotations, one for each vertex of the graph, is called a *rotation system* [6]. The rotation at one vertex is the cyclic permutation corresponding to the order in which the edge-ends are traversed in an orientation-preserving tour around it. Therefore, An embedding of a graph in an orientable surface induces a rotation system [6]. Conversely, every rotation system also induces a unique embedding of the graph onto an orientable surface [22].

Insertions and deletions of edges are both fundamental topological operations in a graph embedding. More details can be found in [8]. In general, given an embedding of graph G, assuming an edge e whose two ends are vertices of G is to be inserted to or deleted from the current embedding, the following two propositions summarize the effects of those two operations.

Proposition B.1. [8] *If the two ends of e are inserted between two corners of the same face in the given embedding, e is inserted to split the face into two but the genus does not change. If the two ends are between two corners of different faces, e is inserted to merge two faces and the genus is increased by 1.*

Proposition B.2. [8] *If the two sides of e are the same face in the given embedding, e is deleted to split the face into two and the genus is decreased by 1. If the two sides belong to different faces, e is deleted to merge those two faces and the genus does not change.*

CHAPTER III

LINE COVER PROBLEM

In this chapter, we studied LCP on exact algorithm, dual problem, geometry properties, and parameterized algorithm. In section A we present an exact algorithm based on dynamic programming, which also helps in the parameterized algorithm to be mentioned in section D. In section B, we raise a problem MAXIMUM NON-COLLINEAR PROBLEM and prove it is the dual problem of LCP. Related geometric properties of LCP are presented in section C. Finally, we propose a parameterized algorithm in section D.

A. Exact Algorithm

The decision problem of LCP was proved NP-Complete in [34], which implies it cannot be solved exactly in polynomial time unless $NP = P$. The proof is a reduction from the known NP-Complete problem 3-SAT. Moreover, it is not hard to observe that LCP is the special case of the general SET COVER PROBLEM with the constraint of exact one intersection. On approximation, in [30], it was proved that LCP cannot be approximated within a ratio $o(logn)$ in random polynomial time unless $NP \subseteq ZTIME(nO(loglogn))$. However, we still have to solve LCP via exact algorithms in certain circumstances. For those readers who are interested in designing exact algorithms to solve NP-hard problems, we recommend the book [15].

Let P be the given set of n points in any instance of LCP. We can draw all the straight lines determined by all those points in P. Clearly this process can be done in polynomial time of n via checking each pair of points in P. Let $L = l_1, l_2, \ldots, l_m$ be the set of m lines determined by all points in P. We are asked to find out minimum lines among m lines in L to cover all points in P. Checking all subsets of L, clearly

we can find the minimum lines in time $O(n2^m)$. However, in our case $m = O(n^2)$ and the running time becomes $O(n2^{n^2})$ with unacceptable complexity. Given an instance of LCP and we construct L in polynomial time as described above. Now we consider the instance (P, L) of LCP and we have the following theorem.

Theorem A.1. *There exists an $O(nm2^n)$ time algorithm to solve any instance (P, L) of LCP where $|P| = n$ and $|L| = m$.*

Proof. Let $cover(l)$ be the set of all points in P covered by the line $l \in L$. Therefore, we have a collection $\mathcal{C} = \{cover(l_1), cover(l_2), \ldots, cover(l_m)\}$ after arbitrarily assigning orders to all those lines. For each nonempty subset of $U \subseteq P$ and for every $j = 1, 2, \ldots, m$ we use $OPT[U, j]$ to denote the minimum size of a subset of $\{cover(l_1), cover(l_2), \ldots, cover(l_j)\}$ that covers all points in U. Now we compute all values $OPT[U, j]$ for every $U \subseteq P$ and $j = 1, 2, \ldots, m$ as follows. Initially for every subset U consisting of exact one point u we set $OPT[\{u\}, 1] = 1$ if $u \in cover(l_1)$, and otherwise $OPT[\{u\}, 1] = \infty$. In $j + 1$ step, $OPT[U, j + 1]$ can be computed for all $U \subseteq P$ in $O(n)$ as follows:

$$OPT[U, j + 1] = \min\{OPT[U, j], OPT[(U \setminus cover(l_{j+1})), j] + 1\}$$

After computing $OPT[U, j]$ for all $U \subseteq P$ and every $j \in \{1, 2, \ldots, m\}$, clearly $OPT[P, m]$ is the result of the given instance of LCP. The complete algorithm *LCP(P, L)* is described in Fig. 5:

As the detailed algorithm shown in Fig. 5, it is not hard to conclude the running time of the algorithm *LCP(P, L)* is $O(nm2^n)$. □

Even though the exact algorithm provided in Fig. 5 is an exponential-time algorithm, in section D we will show that it is beneficial when combing this algorithm with parameterized approaches applied on the decision version of LCP. In particular,

LCP(P, L)
Input: A set P of n points in the plane and a set L of m lines determined by all points in P
Output: The minimum number of covering lines

```
1   C = ∅
2   for each l ∈ L
3       cover(l) = {all points in P covered by l}
4       add cover(l) into C
5   for each u ∈ P
6       if u ∈ cover(l_1)
7           OPT[{u}, 1] = 1
8       else OPT[{u}, 1] = ∞
9   for j = 1 to m − 1
10      for each U ⊆ P
11          OPT[U, j+1] = min{OPT[U, j], OPT[(U \ cover(l_{j+1})), j] + 1}
12  return OPT[P, m]
```

Fig. 5. Exact exponential algorithm LCP(P, L) for LCP

when the cardinality of the given set of points is small enough, i.e. $|P| \leq 5k$ with a small integer k, we can apply this algorithm directly and get an exact solution in time of $O^*(2^{5k})$ which is acceptable from Parameterized Complexity.

B. The Dual Problem

1. Duality

Many computational optimization problems can be formulated as linear programming known as a powerful technique to solve hard problems. Moreover, the concept of linear programming duality has been shown more and more powerful and important [10]. Given an optimization problem to be solved, the identification of a dual problem is often coupled with the discovery of an exciting algorithm. Duality is also powerful on providing a proof that a solution is indeed optimal. Here we skip more details on

weak linear programming duality and linear programming duality. We define a new problem called Non-collinear Packing Problem (NPP) as the following:

Definition B.1. Non-collinear Packing Problem *Given a set P of n points on the Euclidean plane R^2, find a maximum subset $S \subseteq P$ of non-collinear points, i.e. any three points are not collinear.*

Before proving the duality between NPP and LCP, we need to show how to formulate both problems to linear programming. To formulate the form of linear programming for instances of LCP and NNP, we use a few denotations as the following:

- An $\binom{n}{2} \times n$ matrix A, in which each column corresponds to a point in P and each pair of point $p_i, q_i \in P$ determines a row, such that $a_{i,j} = 1$ iff $p_j \in \{p_i, q_i\}$ or collinear with p_i, q_i.
- A vector b of size $\binom{n}{2}$, with each element being 2.
- A vector c of size n, with each element being 1.
- Two vectors x, y of indeterminates with size n and $\binom{n}{2}$ respectively.

The form of linear programming for NNP is as the following:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=0}^{n-1} c_j x_j \\ \text{subject to} \quad & \sum_{j=0}^{n-1} a_{i,j} x_j \leq b_i \quad \forall i = 0, 1, \ldots, \binom{n}{2} - 1 \\ & x_j \in \{0, 1\} \end{aligned}$$

The form of linear programming for LCP is as the following:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=0}^{\binom{n}{2}-1} b_i y_i \\ \text{subject to} \quad & \sum_{i=0}^{\binom{n}{2}-1} a_{i,j} y_i \geq c_j \quad \forall j = 0, 1, \ldots, n-1 \\ & y_i \in \{0, 1\} \end{aligned}$$

Both problems are formulated as integer linear programming as shown above. By the definition of linear programming duality, NPP is the dual problem of LCP.

2. Approximation

GreedyNoncollinear(P)
Input: A set P of n points in the plane
Output: a subset of non-collinear points in P

```
1   S = ∅
2   if P = ∅
3      return ∅
4   if |P| = 1 or |P| = 2
5      return P
6   while P ≠ ∅
7      pick an arbitrary point p from P
8      if S ≠ ∅
9         draw lines determined by p and all points in S
10        remove all points on those lines from P
11     add p into S
12  return S
```

Fig. 6. Greedy approximation algorithm for Non-collinear Packing Problem

We consider approximations on NPP. A greedy approximation algorithm is proposed as shown in Fig. 6.

Clearly this approximation algorithm can be done in polynomial time. Within each loop, we may get at most $n-1$ lines, find all points on those lines, and remove

them from P. This can be done in $O(n)$ or $O(logn)$ via certain data structures and we may remove at least two points from P. In conclusion, the running time is bounded by $O(n^2/2)$. Now suppose the optimal solution is S^* and we consider the approximation ratio. Every time one point is added into S, we find out all lines determined by all points in S and remove all points on them from P, which makes sure that no three collinear points can be added into S in later steps. Hence on each of lines determined by points in S, at most two points in $P \setminus S$ may appear in S^*, implying $|S^*| \leq 2 * \binom{|S|}{2}$ and the approximation ratio $|S| - 1$.

From now on, we consider the parameterized version of NPP: given a set P of n points in the plane and a non-negative integer k, find a subset of k non-collinear points (i.e. any three points are not collinear) in P or report such k points do not exist.

Given any instance (P, k) of NPP, we apply the approximation algorithm *GreedyNoncollinear(P)* in polynomial time and get a set of S as defined in the algorithm. We have the following theorem.

Theorem B.1. *Given an instance (P, k) of NPP. After applying the algorithm GreedyNoncollinear(P), we get S as a solution of (P, k) if $|S| \geq k$. Otherwise, $\forall p \in P$, p is on at least one of $\binom{|S|}{2}$ lines determined by all points in S. Moreover, there exist at most $2 * \binom{|S|}{2}$ non-collinear points in P.*

Proof. Within each loop of the algorithm *GreedyNoncollinear(P)*, every time when one point is added into S, all those lines determined by all the points in S are found and all those points on them are removed to makes sure that no three collinear points can be added into S in the later steps. Therefore, in the set S returned by *GreedyNoncollinear(P)*, it is easy to observe that any three points are not collinear. Thus, if $|S| \geq k$ we already find at least k non-collinear points in P. Otherwise,

assume there exist one point p in P which is not on any of $\binom{|S|}{2}$ lines determined by all points in S. Then after applying the algorithm *GreedyNoncollinear* we know we never removed p from P, which contradicts with the termination condition of that algorithm. Moreover, suppose there exist more than $2 * \binom{|S|}{2}$ non-collinear points in P, more than $\binom{|S|}{2}$ lines are needed to cover these non-collinear points as well as all points, which contradicts $\binom{|S|}{2}$ lines are enough to cover all the points. Thus, either the solution can be found in polynomial time, or the size of solution is always bounded by $2 * \binom{k-1}{2}$. □

Therefore, by theorem B.1, all discussions below are based on the assumption that after applying algorithm *GreedyNoncollinear(P)* the size of returned set is smaller than k given in the instance. Therefore, we can always use $\binom{k-1}{2}$ straight lines to cover all the points, and the number of non-collinear points is always bounded by $(k-1)(k-2)$.

3. Kernelization

In this section, we study NPP by kernelization. We present a reduction rule as follows, which directly implies a problem kernel.

Rule B.1. *Given an instance (P, k) of NPP. If more than $\binom{k-2}{2} + 1$ collinear points are found, then*

- *Remove them from P*
- $k := k - 2$

Lemma B.1. *Rule B.1 can be done in $O(n^3)$ time*

Proof. We can pick every pair of points in P and draw a line, and check how many points are on each line. If we find a line, denoted by l, on which at least $\binom{k-2}{2}+2$ points

are, we remove all points on l from P and get P' whose size is at most $|P| - \binom{k-2}{2} - 2$ and then do $k := k - 2$. Since at most $\frac{n(n-1)}{2}$ lines we can check and on each line at most n points exist, the running time is bounded by $O(n^3)$. □

Theorem B.2. *Let (P, k) be an instance of NPP and let (P', k') be the reduced instance after having applied rule B.1 to (P, k). Then (P', k') has a solution iff (P, k) has a solution.*

Proof. We prove there exist k' non-collinear points in P' iff there exist k non-collinear points in P. After applying rule B.1, we have (P', k') with $|P'| \leq |P| - \binom{k-2}{2} - 2$ and $k' = k - 2$. If there exist k' non-collinear points in P', let S' be the set of k' non-collinear points and A' be the set of lines determined by all points in S', so $|S'| = k' = k - 2$ and $|A'| = \binom{k-2}{2}$. Since all the lines in A' may intersect with l at no more than $\binom{k-2}{2}$ points, there must exist two points on l, say p_1 and p_2, each of which is not on any line in A'. We can find out p_1 and p_2 via checking each point on l and each line in A', which can be done in $O(k^2 n)$. Hence $S' \cup \{p_1 \cup p_2\}$ is a set of k non-collinear points in P, and we know there exist k non-collinear points in P.

If there don't exist k' non-collinear points in P', we know at most $k' - 1$ non-collinear points exist in P' and at most two points on l may appear in any solution of (P, k). Therefore, there exist at most $k - 1$ non-collinear points in P. □

Theorem B.3. *For any instance (P, k) which cannot be solved by algorithm GreedyNoncollinear(P), we can find an instance (P', k') with $k' \leq k$ in polynomial time so that (P, k) has a solution iff (P', k') has a solution. Moreover, there can be at most $O(\frac{k^4}{4})$ points in P'.*

Proof. By theorem B.2, given an instance (P, k), we can recursively apply rule B.1 until can not be applied further, and get a reduced instance (P', k') such that (P', k') has a solution iff (P, k) has a solution. By lemma B.1, each time we carry out rule

B.1 in time $O(n^3)$ and the recursion may be executed at most $O(k/2)$ times, so we can find (P', k') in polynomial time with $|P'| \leq |P|$ and $k' \leq k$.

For the reduced instance (P', k'), there may exist at most $\binom{k'-2}{2}$ collinear points in P'. Based on the assumption mentioned in the previous subsection, all points in P' can be covered by $\binom{k-1}{2}$ lines. Note that applying the approximation algorithm described in the last subsection takes only polynomial time. Therefore, the size of P' can be bounded by $O(k^4/4)$. □

By theorem B.3, given an instance of NPP, we can compute a kernel of size $O(k^4/4)$ in polynomial time. Compared to the existed quadratic kernel of LCP, the kernel size we get of NPP is not exciting, probably due to a natural gap of quadratic size between points and lines. We may further study on kernelization of NPP in future work. Besides kernelization, more discoveries on NPP that was never touched before are about to be presented in the following subsection.

4. Further Study on Noncollinear Packing Problem

Lemma B.2. *Given an instance (P, k) of NPP and a solution S, there exist at most $\left\lfloor \binom{k}{2}/2 \right\rfloor$ collinear points in $P \setminus S$, such that the following two conditions hold at the same time:*

(1) Each point is an intersection point of at least two of $\binom{k}{2}$ lines determined by all points in S

(2) No two points are on the same line of $\binom{k}{2}$ lines determined by all points in S

Proof. Assume there exist at least $\left\lfloor \binom{k}{2}/2 \right\rfloor + 1$ points in $P \setminus S$ satisfying those two conditions. Let A be the set of all those points and suppose all points in A are collinear on a line l, so $|A| \geq \left\lfloor \binom{k}{2}/2 \right\rfloor + 1$. Let L be the set of $\binom{k}{2}$ lines determined by all points in S. According to those two conditions, we know each point in A is on at

least two lines of L, and $l \notin L$. Therefore, A contains at least $\left\lfloor \binom{k}{2} \right\rfloor + 2$ points which is strictly larger than $|L|$, implying there exists at least one point in A not on any line in L, which contradicts with the first condition. Thus, it shows that there are at most $\left\lfloor \binom{k}{2}/2 \right\rfloor$ points in $P \setminus S$ satisfying those two conditions at the same time. □

Lemma B.3. *Given an instance (P, k) of NPP and a solution S, for any point in $P \setminus S$ which is collinear with at most two points in S, we can get another solution S' including that point in polynomial time.*

Proof. Let L be the set of $\binom{k}{2}$ lines determined by all points in S. Arbitrarily pick one point in $P \setminus S$ collinear with at most two points in S, denoted by p, and we have two cases: it is either on a line in L, or not on any line in L.

If p is not on any line in L, $S \cup p$ is a set of $k+1$ non-collinear points. Arbitrarily removing one point p' from S, we can get a new solution $S' = \{S \setminus p'\} \cup p$ for the same instance in linear time.

If p is on one line in L, suppose determined by p_1 and p_2, since p is not collinear with any other two points in $S \setminus \{p_1 \cup p_2\}$, we can get another solution either $S' = \{S \setminus p_1\} \cup p$ or $S' = \{S \setminus p_2\} \cup p$ for the same instance in linear time.

To check p is on how many lines in L can be done in $O(k^2)$. Combing the discussion above, we complete the proof. □

Lemma B.4. *Given an instance (P, k) of NPP and a solution S, if there are at least two points in $P \setminus S$ collinear with at most two points in S, we can get another solution S' including two of them in polynomial time.*

Proof. Let C be the set of those points in $P \setminus S$ collinear with at most two points in S. Any solution can include at most two points in C. Without loss of generality, suppose we want to find a solution including both p_1 and p_2 in C. By lemma B.3, we can find a solution S' containing p_1 in polynomial time. If p_2 is collinear with at most

two points in S', applying lemma B.3 again we can find a solution S'' containing p_1 and p_2 in polynomial time. If p_2 is collinear with more than two points in S', since p_2 is collinear with at most two points in $S' \setminus p_1$, it must be collinear with p_1 and another point p_3 in $S' \setminus p_1$. We replace p_3 by p_2 and get a solution containing both of p_1 and p_2 in polynomial time. □

Lemma B.5. *Given an instance (P, k) of NPP and a solution S. If there exist $\lfloor \binom{k}{2}/2 \rfloor + 1$ points in $P \setminus S$ which are collinear, assuming on a line l, we can find a solution including one point on l in polynomial time. If there exist at least $\lfloor \binom{k}{2}/2 \rfloor + 2$ points in $P \setminus S$ which are collinear, assuming on a line l', we can find a solution including two points on l' in polynomial time.*

Proof. Let L be the set of $\binom{k}{2}$ lines determined by all points in S. If $l \in L$, S already includes two points on l. If $l \notin L$, by lemma B.2 we know there is a point p on l collinear with at most two points in S, and then by lemma B.3 we can get another solution S' including p. Similarly, if $l' \in L$, S already includes two points on l'. If $l' \notin L$, by lemma B.2 we know there are at least two points collinear with at most two points in S, and then by lemma B.4 we can get another solution S'' including two points on l'. Clearly seeking those points can be done in $O(k^2 n)$, so totally the whole process can be done in polynomial time. □

Rule B.2. *Given an instance of NPP (P, k). If more than $\lfloor \binom{k}{2}/2 \rfloor + 1$ collinear points are found, remove them from P.*

Theorem B.4. *Let (P, k) be an instance of NPP and (P', k) be the reduced instance after repeatedly applying rule B.2 to (P, k). Given a solution S' of reduced instance (P', k), we can find a solution S of (P, k) containing two points in $P \setminus P'$ in polynomial time.*

Proof. Analogous to the proof of lemma B.1, rule B.2 can be applied in $O(n^3)$. Let C be the set of all those collinear points we removed via rule B.2, so $|C| \geq \left\lfloor \binom{k}{2}/2 + 2 \right\rfloor$ and $P' = P \setminus C$. If (P', k) has a solution S' of size k, S' is also a solution for the instance (P, k). Since $C \subseteq P \setminus S'$, by lemma B.5 we can find a solution including two points in C in polynomial time. □

C. Geometric Properties

One weakness of all previous research on LCP is the lack of exploiting geometric properties. As mentioned before, this problem is essentially a fundamental problem in Computational Geometry. Specifically, the Euclidean plane $\mathbb{R}^2$ we focus on in LCP may bring several meaningful geometric properties to handle this problem. Even though [33] abstracted LCP on the geometry level and emphasized the role geometry may play, all the existed algorithms are directly derived from combinatorics. Therefore, we try to studied LCP from geometric perspectives starting from surveying related theorems.

A point in the plane has two natural parameters, x-coordinate and y-coordinate [12]. Also, a line which is not a vertical line has two parameters, slope and intersection with y-axis. Thus, a set of points in the plane and a set of lines can be systematically mapped between each other, called *duality transforms* as the following [12].

Let $p = (p_x, p_y)$ be a point in the given set of points. The dual of p denoted by l_p is the the line defined as:

$$l_p := (y = p_x x - p_y)$$

Let $l : y = mx + b$ be a line in the plane. The dual of l is the point defined as:

$$p_l := (m, -b)$$

Note that this duality transform described above is not well-defined for vertical lines. However, usually we can do preprocessing for vertical lines, i.e. rotating the x-axis and y-axis in order to eliminate the cases of vertical lines.

Certain properties hold after duality transform: (1) incidence preserved: that is, $p \in l$ iff $p_l \in l_p$. (2) order preserved: p lies above l iff p_l lies above l_p.

Basically the duality transform can convert a problem of point configuration, i.e. LCP, to a problem of line arrangement. Essentially these two types of geometric problems are the same problems. However, the advantage here is that it provides a new perspective. Still, if a problem transformed from the given primary problem is solved, it may benefit the given problem. Note that the duality transform described above clearly can be done in polynomial time. Transform LCP to a problem of line arrangement, and we can define the dual as the following:

Definition C.1. Minimum Point Across Problem (PAP) *Given a line arrangement of n straight lines in the Euclidean plane $\mathbb{R}^2$, find the minimum number of intersection points such that any line passes through at least one of them.*

[4] The most famous question concerning the set of lines spanned by finite points from Sylvester is that if it is true that given a bunch of points in the Euclidean plane, not all on a line, there are two points whose connecting line called *ordinary line* does not pass through a third [44]. The history of the research on this question is fairly long [4]. Finally, it ended with the famous *Sylvester-Gallai theorem*: given a finite number of points in the Euclidean plane, either all the points are collinear, or there is a line containing exactly two of the points. The next natural question is to find the minimum number t_2 of ordinary lines. The best known lower bound found by Csima and Sawyer [11] is $t_2 \geq 6n/13$. Trying to dig out any potential usefulness, we enlist several related theorems and status as follows. Most results are surveyed from the

book [4].

Every configuration of $n \geq 11$ points in the plane, not all on one line and not forming a near-pencil, determines at least $2n-4$ lines [28]. On the other hand, Erdos [13] showed that all values from $cn^{\frac{3}{2}}$ to $\binom{n}{2}$ are reachable, except two cases of $\binom{n}{2}-1$ and $\binom{n}{2}-3$. Similar results are found by Beck [2]. All these theorems suggest a general lower bound of number of all the spanning lines.

Erdos-Beck Theorem is summarized as the following[1]: let P be a set of n points on the plane. If no more than $n-k$ points are collinear for some $0 \leq k < n-2$, there are $\Omega(nk)$ lines determined by points in P. Applying to our problem LCP, suppose no more than k points are collinear where $0 \leq k < n-2$, then we know there are at least $n(n-k)$ lines spanned by all given points.

In [1], it also proved the "weak Dirac conjecture", stating that there exists a constant $\epsilon > 0$ such that we can always have a point incident to at least ϵn lines determined by any point configuration of n points in the plane where not all are collinear [4]. Back to our problem LCP, this theorem shows that we can always find a point which connects lines between that point and other ones, and we can get ϵn lines. For the dual of line arrangement, similarly, the theorem suggests there must be one line on which there are ϵn intersection points.

[4] For any $k \geq 2$, let t_k denote the number of lines passing through precisely k points. As mentioned at the beginning of this section, the current lower bound of t_2 is $\frac{6}{13}n$. For the dual version, i.e. the line arrangement $\mathcal{A}$, $t_2(\mathcal{A}) \geq \frac{6}{13}$. Extending to larger numbers, the Erdos-Purdy inequalities [14] suggests that: if $t_n = 0$, $\max(t_2, t_3) \geq n-1$ and $\forall i, \max(t_2, t_3) \geq t_i$. If $t_2 \leq n-2$ and $t_n = 0$, $t_3 \geq cn^2$. If $t_k = 0$ for all $k \geq \frac{1}{100}n$, total number of spanned lines satisfies $\sum_k t_k \geq cn^2$, for a suitable constant $c > 0$. Back to LCP, intuitively speaking, all the theorems suggest that quite a few lines pass through exact two or three points.

[4] Let $t_k^{orchard}(n)$ denote the max of t_k over all sets of n points in a plane containing no more than k collinear points. For any fixed $k \geq 4$, $t_k^{orchard}(n) = o(n^2)$. Via an ingenious recursive construction, Grunbaum [23] gave a lower bound: $t_k^{orchard}(n) \geq c_k n^{1+\frac{1}{k-2}}$.

Although there is no affirmative outcome obtained from those related geometric theorems enlisted above, we still survey them in this thesis in order to inspire readers. It is possible that they may benefit on other problems in Computational Geometry.

D. Parameterized Algorithm

The parameterized algorithm proposed in this thesis is based on the idea of bound-search tree introduced in chapter 2. We abstract LCP as a combinatorial problem of putting n points into k boxes as follows, similar to what have been mentioned in [19]. There are k boxes totally and each box has room for exactly two points, so each box can correspond to a straight line determined by the two points in that box. A box is called closed if it contains exact two point and called open otherwise. Each closed box, containing two points, defines a unique line that passes through two points in the box and hence covers all points in the given set which are collinear with those two points. We can abstract each node in the search tree as a ways of placement of some points in those k boxes. The root of the search tree represents the scenario that all boxes are empty. All the children of a node v correspond to several possible ways of picking the next point from the given set and putting it into an open box in the configuration of v. The ways of picking which point and putting it into which box both require consideration. Take the next point and put it into an open box b, a new child of v is created. If b is closed, all points in the given set that are covered by the corresponding line l_b will be removed from the given set of points and we put l_b into

the solution as a covering line. Then keeping doing branchings to find other covering lines by the same way. If all boxes are closed, we stop the whole searching process. Now it is not hard to check whether all points are covered by those k boxes in time $O(kn)$.

In particular, similar to the algorithm in [46], we define a configuration of a placement of points in k boxes as (n_0, n_1, n_2), such that for $i = 0, 1, 2$ there are n_i boxes that contains exactly i points. Note that two different placements may have the same configuration. For each node in the search tree, we have a corresponding configuration, and we use n_0, n_1, n_2 as three parameters in our algorithm. Clearly, $0 \leq n_0, n_1, n_2 \leq k$, and initially $n_0 = k, n_1 = n_2 = 0$. Additionally, in the algorithm to be shown in Fig. 7, we define $cover(l)$ as the set of points covered by l.

Recall the exact algorithm $LCP(P, L)$ described in section A. Even though the running time of algorithm $LCP(P, L)$ is $O^*(2^n)$, still it is applicable in the parameterized algorithm to be shown in Fig. 7. Whenever the number of remaining points that have not been covered becomes small enough, i.e. $5k$, we can immediately run the algorithm $LCP(P, L)$ in $O^*(2^{5k})$ to deal with all the remaining points and make sure an exact result will be returned.

Theorem D.1. *The algorithm LC(P, (n_0, n_1, n_2)) will return a correct solution for every instance (P, k) of LCP.*

Proof. When a box becomes closed as defined above, we get a line as a covering line via connecting those two points in the box, and remove all points in the given set P covered by that covering line. Hence when P is empty, the configuration (n_0, n_1, n_2) that corresponds n_2 covering lines is the correct solution of the given instance. In the meanwhile, if k lines have been found while P is not empty, the corresponding branch fails. Otherwise, in step 5 and 6, we draw all lines and find every set of points

covered by each line. In the step 8, if no line that covers more than 6 points can be found, it means that in the set of uncovered points P, $|P| \leq 5k$ if all points of P can be covered by k lines. Clearly we can check the cardinality of P in polynomial time. If $|P| > 5k$ then we know no solution for this branch. Otherwise, we call the exact algorithm described in section A to find the minimum number of covering lines, which can be done in $O^*(2^{5k})$. Now we have a claim and proof as the following.

Claim. *Through step 9 to step 11 in $LC(P, (n_0, n_1, n_2))$ shown above, the variable a is no larger than $k - n_2$ if and only if in this branch we can find a solution of the given instance (P, k).*

To prove this claim, first we prove the following statement: the variable a is no larger than $k - n_2$ if and only if the remaining set of points whose cardinality is smaller than $5k$, denoted by P', can be covered by $k - n_2$ lines. If $a \leq k - n_2$, points of P' can be covered by a lines, and then definitely can be covered by $k - n_2$ lines. If $a > k - n_2$, according to the optimality of the a, points of P' have to be covered by more than $k - n_2$ lines. Then, we prove all points in P' can be covered by $k - n_2$ lines if and only if in the current branch we can find k lines to cover all points of P. Suppose the algorithm has proceeded to certain configuration (n_0, n_1, n_2) and let P' be the set of all uncovered points with $|P'| \leq 5k$. Based on our algorithm, points of $P \setminus P'$ can be covered by n_2 lines. If all points of P' can be covered by $k - n_2$ lines, since points of $P \setminus P'$ can be covered by n_2 lines, clearly all points of P can be covered by k lines. On the other hand, if more than $k - n_2$ lines are required to cover all points of P', for the current branch, more than k covering lines are needed for P. Hence our algorithm returns "this branch fails" and tries another branch. If over all possible branches enumerated, all the given points have to be covered by more than k lines, our algorithm reports such k covering lines do not exist. Otherwise, combining

current configuration and a covering lines for P', we get the solution.

In step 12, if more than $k - n_2$ points on the line l and if it is not a covering line in the final solution, then at least $|cover(l)|$ lines are needed to cover all points on l, which exceeds the number of required lines. Thus, l must be a covering line and we remove one empty box from the configuration and all points on l from P. Otherwise, $|cover(l)| \leq k - n_2$, we branch by two case on l: either l is a covering line, or it is not and all points in $cover(l)$ must be covered by $|cover(l)|$ different covering lines. Therefore, from step 17 to step 29, we assume l is not a covering line in the final solution. We add each point on l one by one into $|cover(l)|$ different boxes since those points must be covered by distinct lines in any solution if it exists. The underlying idea is to enumerate all possible cases, and when a branch is reached to be a closed box we find the line spanned by points in that box, and the same operations as described above.

If $n_1 = 0$, it means that all the open boxes are empty. Then for each point on l we just put it into an empty box. If $n_1 = k - n_2$, it means that each open box contains exact one point. Then we branch by adding each point into one of n_1 boxes and for each branch we get $|cover(l)|$ lines as shown from step 21 to 23. Otherwise, among all those open boxes, there are some empty ones and some containing one point. In step 25, we partition the points on l into two parts, one is about to be put in empty boxes and another will be added to those boxes with one room. Adding to empty boxes is always trivial without branching, while we need to branch for the other part from step 27 to 29.

The whole process makes sure that for each line spanned by points in the input set of points, both cases of either being a covering line or not are considered. Moreover, if it is not a covering line, each point on the line is used to seek another covering line which must pass through it. Combining with part of the proof of theorem 3.1 in [46],

we finish the proof of correctness. □

Theorem D.2. *The algorithm LC(*$P, (n_0, n_1, n_2)$*) is a parameterized algorithm.*

Proof. As described in the proof of theorem D.1, in the algorithm LC($P, (n_0, n_1, n_2)$) shown in Fig. 7, we do branching on different ways of placements for only $2k$ points. By step 3, if there is no open box left, the algorithm stops. By step 12, if the line we picked is the covering line, we remove points on it and remove one empty box, without placing those points into boxes. Moreover, as n_0 is bounded by k, the execution times of recursions in step 13 and 15 are bounded by k since every time n_0 is decreased by 1. If the line is not a covering line in the final solution then we put each point on it into a distinct box. The total points we need to place cannot be more than $2k$ since $2k$ is the total rooms of k boxes. By lemma 2.1 in [46], the number of leaves in the search tree of the whole algorithm proposed in [46] is exactly equal to the number of different placements of all the points we put in. Therefore, via the same analysis in the proof of theorem 2.2 in [46], we get the same running time $O^*(k^k/1.35^k)$ which shows the algorithm LC($P, (n_0, n_1, n_2)$) is a parameterized algorithm. □

LC$(P, (n_0, n_1, n_2))$
Input: A set P of points in the plane, and a box placement with configuration (n_0, n_1, n_2)
Output: An extension of the given configuration to k lines corresponding to k boxes that cover all points in P if such an extension exists.

if $P = \emptyset$
  then return the input configuration of placement
if ($n_2 = k$ and $P \neq \emptyset$) or $n_0 < 0$
  then return "this branch fails"
Draw all lines $L = \{l_1, l_2, \ldots, l_m\}$ spanned by all points in P
Compute $cover(l_i)$ for $i = 1, 2, \ldots, m$
Pick a line l in L with $|cover(l)| \geq 6$
if no line covering more than 6 points can be found
  Call $LCP(P, \{l_1, l_2, \ldots, l_m\})$, assign returned value to a variable a
  if $a > k - n_2$, return "this branch fails"
  else combine current configuration and the solution found in $LCP(P, \{l_1, l_2, \ldots, l_m\})$, and return it.
if $|cover(l)| > k - n_2$
  LC$(P \setminus cover(l), ((n_0 - 1), n_1, (n_2 + 1)))$
case 1
  LC$(P \setminus cover(l), ((n_0 - 1), n_1, (n_2 + 1)))$
case 2
  if $n_1 = 0$
    Add each point $p \in cover(l)$ into an empty box
    LC$(P \setminus cover(l), ((n_0 - |cover(l)|), (n_1 + |cover(l)|), n_2))$
  if $n_1 = k - n_2$
    Branch by adding each point $p \in cover(l)$ into one of n_1 boxes
    for each branch we get $|cover(l)|$ lines: $l'_1, l'_2, \ldots, l'_{|cover(l)|}$
      $P' := cover(l'_1) \cup cover(l'_2) \cup \cdots cover(l'_{|cover(l)|})$
      LC$(P \setminus P', (n_0, (n_1 - |cover(l)|), (n_2 + |cover(l)|)))$
  else
    for each of $2^{|cover(l)|}$ partitions of $cover(l)$: (P_0, P_1)
    if $|P_0| \leq n_0$ and $|P_1| \leq n_1$
    Add each $p \in P_0$ into an empty box, branch by adding each $p \in P_1$ into one of n_1 boxes, for each branch we get $|P_1|$ lines: $l'_1, l'_2, \ldots, l'_{|P_1|}$, and **for** each branch
      $P' := cover(l'_1) \cup cover(l'_2) \cup \cdots cover(l'_{|P_1|})$
      LC$(P \setminus \{P_0 \cup P'\}, ((n_0 - |P_0|), (n_1 + |P_0| - |P_1|), (n_2 + |P_1|)))$
return "no solution"

Fig. 7. Parametrized algorithm for Line Cover Problem

CHAPTER IV

MAXIMUM GENUS EMBEDDING

In this chapter, we investigate interrelations between maximum genus of a graph and its cycle space consisting of fundamental cycles. Specifically, we restudy a known incorrect approach of finding a maximum genus embedding by computing a maximum pairing of intersected fundamental cycles with respect to an arbitrary spanning tree. We point out the reason that it does not work. Furthermore, we consider the interplay between upper-embeddablity and the maximum parings of intersected fundamental cycles. As a byproduct of exploring properties of fundamental cycles, in terms of maximum genus and cycle rank, we show a lower bound and an upper bound on MAXIMUM VERTEX-DISJOINT CYCLES which is a known optimization problem with the goal of packing maximum vertex-disjoint cycles in a graph.

A. Intersection Graph and Maximum Matching

Definition A.1. [40] *A graph $G_M = (V_M, E_M)$ is the* intersection graph *of G where V_M is a set of nodes representing all the fundamental cycles with respect to a spanning tree of G and any two nodes in V_M are connected by an edge in E_M if and only if those two corresponding fundamental cycles have at least one vertex in common.*

In [40], an approach to solve MGE was proposed, which heavily depended on intersection properties of fundamental cycles. It stated that the maximum genus of a graph is equal to the size of any maximum matching of an intersection graph with respect to an arbitrary spanning tree, which seems to improve the result in [16] and also attracts us to explore the intersection graph and maximum matching. The algorithm runs fairly simple as follows. Start from an arbitrary spanning tree

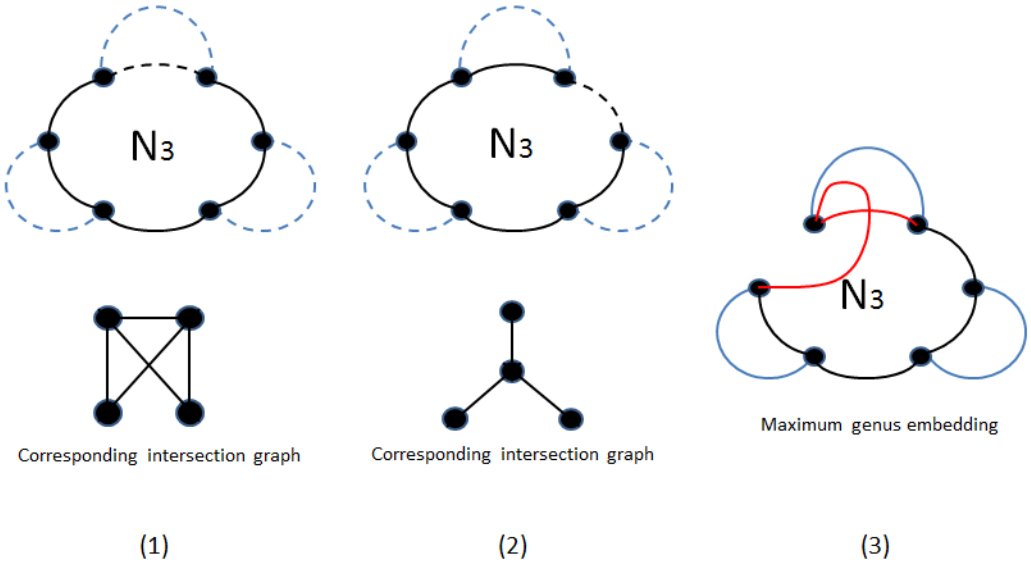


Fig. 8. N_3 and corresponding intersection graphs

of the given graph and the corresponding intersection graph can be constructed in polynomial time. Then run the existed best algorithm to find a maximum matching [35]. It claimed that the cardinality of resulting matchings always equals to the maximum genus of the given graph. What is more, suppose the size of a maximum matching is g, then the graph can be embedded on a surface of genus at least g. However, counterexamples were provided in [29]. One counterexample was on a class of 2-edge connected graphs, and another was derived from a class of 3-edge connected graphs.

In this thesis, firstly, we identify a kind of substructure where the bugs may emerge. As shown in Fig. 8, N_3 is a cubic graph consisted of 6 vertices and 9 edges. The solid edges constitute a spanning tree of N_3 and dashed edges are cotree edges. Thus, fundamental cycles can be identified and the corresponding intersection graph can be constructed. In Fig. 8 there are two different spanning trees and two distinct intersection graphs, as shown in (1) and (2). After running the existed algorithm

of computing the maximum matching on those two intersection graphs, we get two maximum matchings of different size, 2 for (1) and 1 for (2). Since maximum genus is an inherent property of the given graph, clearly the scheme in [40] does not work for the case of N_3. In fact, the maximum genus of N_3 is one and a maximum genus embedding is shown in (3) of Fig. 8. We can change the rotation of the common vertex of that pair of adjacent red edges and let one red edges cross along a handle, increasing the genus by 1.

Although N_3 is a substructure which is not a simple graph, counterexamples on simple graphs and cubic graphs can also be derived from N_3 as shown in Fig. 9. Note that (1) of Fig. 9 is a 2-edge connected graph which may contain the same base of fundamental cycles with N_3. We are interested in cubic simple graphs since the cotree components with respect to any spanning tree become special, i.e. every component is either a simple path or a simple cycle. The graph of (2) in Fig. 9 is a cubic simple graph, and it contains three cut edges which must be included in any spanning tree. Therefore, there exist spanning trees such that parts of the corresponding cotree components and intersection graphs are exactly as same as those shown in Fig. 8.

Perhaps the most special part of the approach proposed in [40] is the starting point: an arbitrary spanning tree. Because we have found so many counterexamples, we check if the approach works on special spanning trees, i.e. depth-first search (DFS) trees. DFS is a widely-used algorithm for traversing the whole graph starting at an arbitrary vertex. After the searching process, a spanning tree is formed based on the order of the vertices reached during the search. The most important trait of DFS trees is that all the cotree edges are *back edges* each of which always connects a vertex and one of its ancestors. Hence, the two ends of each cotree edge should be in the same subtree. Intuitively speaking, compared to other kinds of spanning trees, there will be less intersections of fundamental cycles with respect to a DFS tree. For

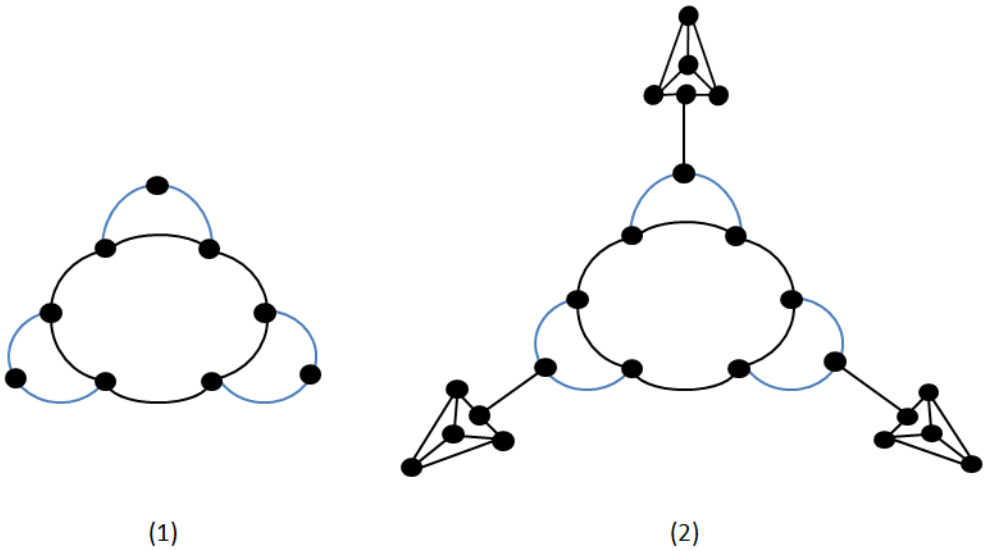


Fig. 9. Counterexamples on simple graph and cubic graph

some spanning trees there are so many edges in the corresponding intersection graphs that the size of maximum matching can be very large, but the edges can become less in intersection graphs of DFS trees. In Fig. 10, we list three examples, assuming the DFS trees are given. Solid edges are tree edges and dashed edges denoted by lowercase letters are cotree edges. Recall that we denote a fundamental cycle $T(e)$ uniquely determined by the cotree edge e.

For the first one, constructing the corresponding intersection graph, we get one as same as the previous example shown in (1) of Fig. 8, and we get a maximum matching of size 2. Without loss of generality, suppose we get two pairs $(T(b), T(c))$ and $(T(a), T(d))$. We illustrate the process of embedding those four cotree edges in Fig. 11. After embedding two edges b and c, shown as (1) of Fig. 11, where the edge c can be embedded in a way of crossing along a handle, the number of faces of that embedding is one and the genus is increased by one. Then, as shown in (2) of Fig. 11, since two ends of edge a are both at two corners of the same face, we embed a and

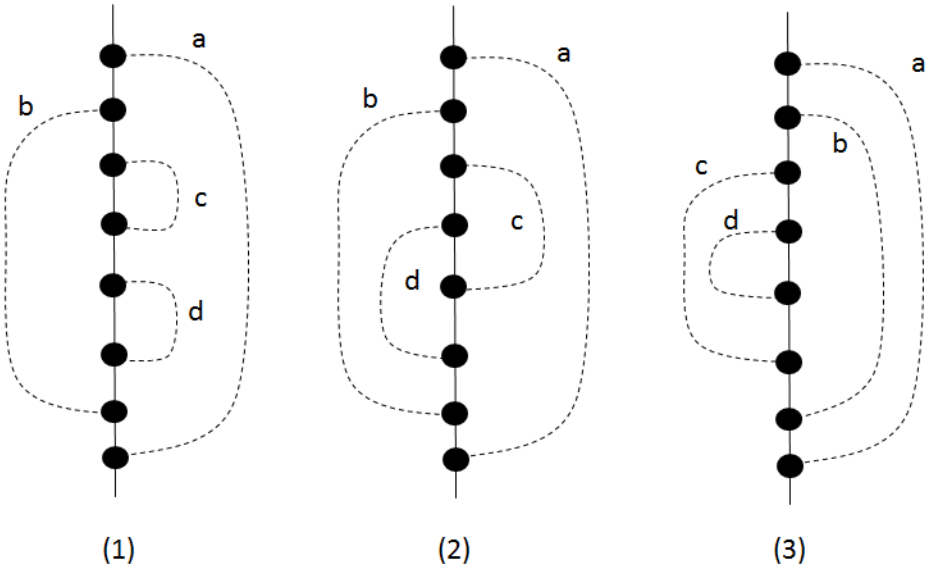


Fig. 10. DFS trees and cotree edges

it splits face 1 into two faces. Finally, when inserting d, we can see the two ends of d are still at two corners of the same face as shown in (3), without increasing genus. Note that no matter how to embed edge a, two ends of d are still at two corners of the same face. Therefore, even though we get a maximum matching of size two, only one genus can be created, which directly contradicts the first theorem in [40]. In fact, (4) of Fig. 11 is the maximum matching with size one which leads to a maximum genus embedding with three faces.

Back to Fig. 10. In (2), any pair of fundamental cycles are adjacent, i.e. at least one vertex in common. The intersection graph is a complete graph of four nodes and hence it includes a perfect matching. Similar to the process illustrated in Fig. 11, it is not difficult to check that the maximum genus of the subgraph in (2) is equal to the size of every maximum matching. Moreover, each pair of any maximum matching can create one more handle. In fact, the second graph is upper-embeddable. We will discuss more details on upper-embeddablity in the next section.

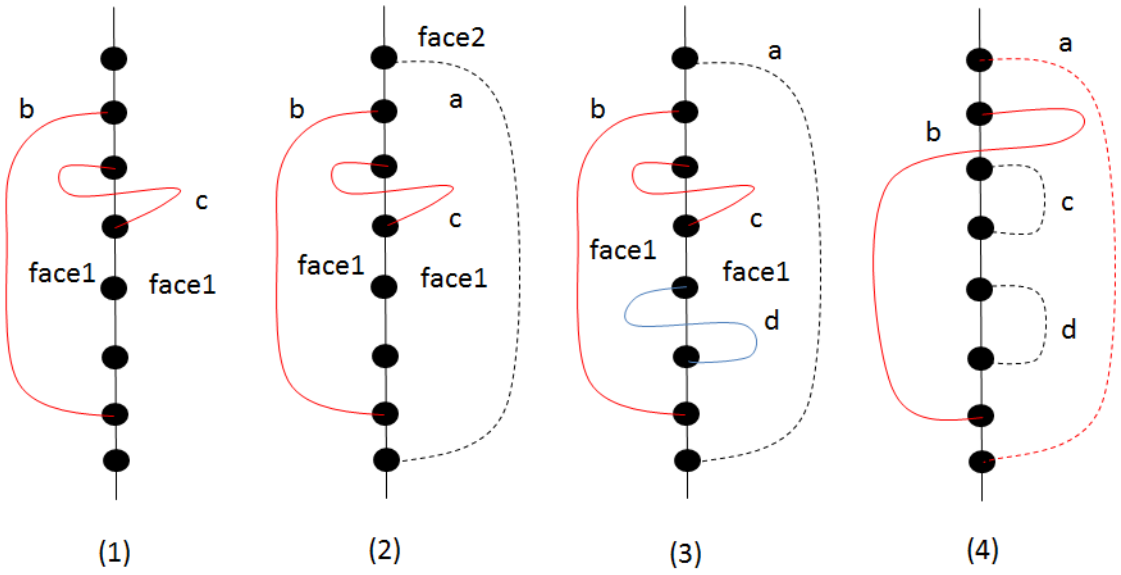


Fig. 11. The process of embedding the subgraph shown in (1) of Fig. 11

Although [29] provided counterexamples, the source of bugs in [40] have not been presented clearly. In this thesis, we show it starting from the example of (3) in Fig. 10. Similar to (2), it is an upper-embeddable graph and the corresponding intersection graph has a perfect matching with size two. Suppose we get the matching $(T(a), T(b))$ and $(T(c), T(d))$, then we can construct the embedding shown in (1) of Fig. 12 with genus two which is already maximum. Note that the order of embedding those two pairs does not matter. It seems that the algorithm works for all the upper-embeddable graphs. Nevertheless, we find that it is a different story if we get another maximum matching $(T(b), T(c))$ and $(T(a), T(d))$. If we embed b and c first as shown in (2) of Fig. 12, we can get an embedding with one more genus. Suppose we have face 1 after embedding the pair (b, c), now it is not hard to see that both ends of a and d are at corners of face 1, and then no matter how to embed a and d we get an embedding of three faces with genus unchanged. If we embed a and d first, it is not difficult to check that embedding c and b cannot increase genus neither. Also, if we

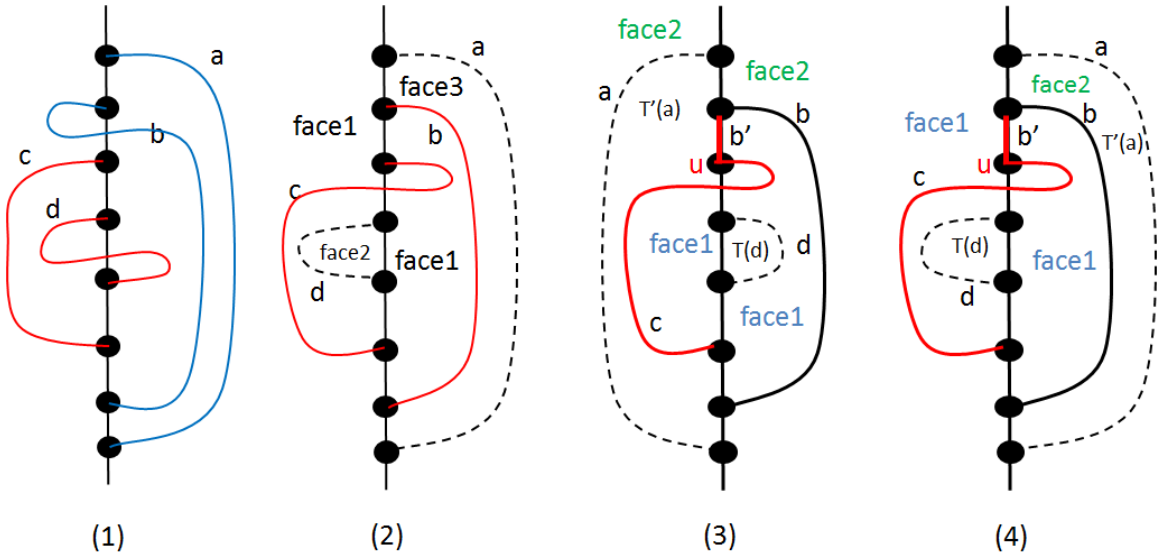


Fig. 12. Different matchings create embeddings with different genus

get a maximum matching of $(T(b), T(d))$ and $(T(a), T(c))$, it is the same result that no matter how to embed those two pairs we can only create an embedding of three faces with only one genus.

We investigate the reason by illustrating examples shown in (3) and (4) of Fig. 12. Given two intersected fundamental cycles, we know there are exact two intersection vertices even though they may have lots of vertices in common. When we embed those two cotree edges with aim of increasing genus, the only way is to change the rotation on either intersection vertex, not ends of cotree edges. Suppose $T(b)$ and $T(c)$ are incident at a vertex u as shown in (3) and (4). Changing the rotation of u as shown can keep only one face with genus increased, while neither of the rotations on two ends of b is changed. Now we insert edge a whose two ends are at corners of the same face and the only face is split into two faces. Then we can see that no matter how to embed a, both faces obtained contain a and b, and two ends of d are always in the same face. Thus, inserting d can only generate an embedding with three faces

with genus one. It is easy to check that embedding d at first will bring the same result.

From the perspective of spanning tree and fundamental cycle, the series of topological operations described above is equivalent to the operations of changing current pair of cotree edges to adjacent cotree edges whose incident vertex is one of two intersection vertices of a pair of incident fundamental cycles. It is the adjacent pair (c, b') rather than (c, b) to increase genus. For the next pair to be embedded, the spanning tree has actually been changed: the previous cotree edge b has become a tree edge and the previous tree edge b' now is a cotree edge, causing $T(a)$ changed to $T'(a)$ as shown in (3) and (4). Surprisingly now $T'(a)$ and $T(d)$ do not have any vertex in common anymore, and we have no way to insert a and d to increase genus. Thus, the maximum matching computed at the beginning on an arbitrary intersection graph cannot guarantee that each pair of matchings may be converted to an adjacent pair of edges to increase genus, because the spanning tree has been changed after each pair are embedded and previous intersected cycles may become disjoint now.

In conclusion, bugs occurred in the proof of theorem 1 of [40] where it claimed that given g pairs of intersected fundamental cycles with respect to a spanning tree T of G then there exists an embedding of G with genus at least g. Given a spanning subgraph H of G where $H \supseteq T$ and two intersected fundamental cycles $T(e)$ and $T(f)$ where $e, f \in E(G)-H$, the proof confused $\xi(H)$ and $\xi(H, T)$, even though it correctly showed that $\xi(H, T) \geq \xi(H+e+f)$. Starting from T, $\xi(T) = \xi(T, T) \geq \xi(T+e+f)$ holds, so $\gamma_M(T + e + f) > \gamma_M(T)$ is true. However, then when starting from the subgraph $G_0 = T + e + f$, $\xi(G_0) = \xi(G_0, T)$ may not necessarily hold unless T is a Xuong tree of G_0, i.e. e and f are a pair of adjacent cotree edges with respect to T. Thus, given two intersected fundamental cycles $T(g)$ and $T(h)$ with respect to T where $g, h \in E(G) - G_0$, $\xi(G_0) \leq \xi(G_0, T) \geq \xi(G_0 + g + h)$ and $\gamma_M(G_0 + g + h) > \gamma_M(G_0)$

may not be necessarily true. It was mentioned that a new spanning tree T' can be constructed such that $\xi(T) = 0 = \xi(T, T) = \xi(T + e + f, T') = \xi(T + e + f)$, by the same way shown in (3) and (4) of Fig. 12. Then T' is a Xuong tree of G_0. However, now a pair of intersected fundamental cycles $T(g)$ and $T(h)$ may not be incident anymore with respect to T', so inserting g and h may not be able to increase genus.

In a word, we have the following result, and the equality may not necessarily hold.

$$\xi(H) \leq \xi(H, T) \geq \xi(H + e + f, T') \geq \xi(H + e + f)$$

Therefore, constructing an arbitrary spanning tree T at the beginning cannot guarantee that T is a Xuong tree of both H and $H + e + f$. Also, even though we can find another spanning tree T' which may be a Xuong tree of $H + e + f$, the next pair of intersected fundamental cycles with respect to T may not be incident with respect to T'.

B. Upper-embeddablity

As we mentioned in previous chapters, upper-embeddablity is always the property people are interested in not only because quite a few classes of graphs has been proved upper-embeddable but also the maximum genus is equal to the cycle rank. Suppose the graphs we discuss here are connected simple graphs. We use notations in this section as the following: let $G\sharp T$ denote the intersection graph G_M of G with respect to a spanning tree T and denote the size of a maximum matching of $G\sharp T$ by $\nu(G\sharp T)$. In [16], as mentioned in chapter 1, an important theorem indicated the relationship between maximum genus and the size of any maximum matching of corresponding intersection graph as follows.

Corollary B.1. [16] *Let G be a connected graph, then $\gamma_M(G) = \min \nu(G\sharp T)$ taking*

over all the spanning trees of G. Moreover, there exists a spanning tree T such that $\gamma_M(G) = \nu(G\sharp T)$.

Now we present a theorem characterizing upper-embeddbablity in terms of the size of maximum matchings of intersection graphs.

Theorem B.1. *If G is an upper-embeddable graph, then $\nu(G\sharp T) = \lfloor \beta(G)/2 \rfloor$ for any two spanning trees T of G*

Proof. By corollary B.1, we have $\min \nu(G\sharp T) = \gamma_M(G)$ taking over all possible spanning trees of G. Since G is upper-embeddable, by definition $\gamma_M(G) = \lfloor \beta(G)/2 \rfloor$. Thus, for an arbitrary fixed spanning tree T, we know $\nu(G\sharp T) \geq \min \nu(G\sharp \mathcal{T}) = \gamma_M(G) = \lfloor \beta(G)/2 \rfloor$.

On the other hand, for any spanning tree T, the graph $G\sharp T$ has $\beta(G)$ vetices and the cardinality of maximum matching can be at most half of number of vertices. Hence we get $\nu(G\sharp T) \leq \lfloor \beta(G)/2 \rfloor$. Therefore, $\nu(G\sharp T) = \lfloor \beta(G)/2 \rfloor$ for every spanning tree T. □

Theorem B.2. *A graph G is an upper-embeddable graph if and only if every intersection graph of G has a maximum matching of size $\lfloor \beta(G)/2 \rfloor$.*

Proof. We show if G is not upper-embeddable, not every intersection graph of G has a maximum matching of $\lfloor \beta(G)/2 \rfloor$. If G is not upper-embeddable, by definition we have $\gamma_M(G) < \lfloor \beta(G)/2 \rfloor$. Via corollary B.1, we know there exists a spanning tree T such that $\nu(G\sharp T) = \gamma_M(G) < \lfloor \beta(G)/2 \rfloor$. Thus for the intersection graph $G\sharp T$ there is no perfect matching. Combing theorem B.1 we finish the proof. □

By theorem B.2, the upper-embeddablity of a graph can be checked via constructing intersection graphs and computing maximum matchings. Although it is known

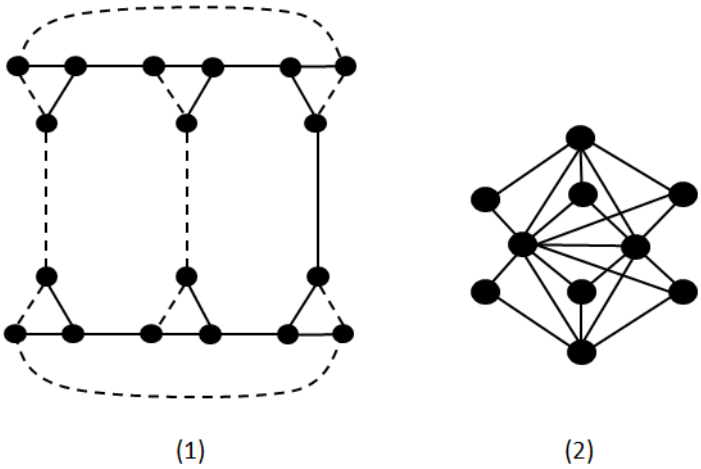


Fig. 13. 3-edge connected cubic graphs

that there exist infinite 3-edge connected cubic graphs that are not upper-embeddable [7], here we give an example based on theorem B.2.

The graph of (1) in Fig. 13 is a 3-edge connected cubic simple graph. It contains 18 vertices and 27 edges and we investigate if it is upper-embeddable. In (1) solid edges form a spanning tree and dashed edges are cotree edges. For that spanning tree, it is not hard to see that the intersection graph shown in (2) of Fig. 13 does not have a perfect matching, i.e. the size of any maximum matching is no more than 4. By theorem B.1, we can immediately conclude that the graph (1) in Fig. 13 is not an upper-embeddable graph.

In [32], an approach of finding a maximum genus embedding on *signed* graphs was proposed. Since unsigned graphs discussed in this thesis is a subclass of signed graphs, that method can be extended to tackle MGE. The main idea is as the following: start with an arbitrary spanning tree T of G and greedily find a maximal set A of adjacent pairs of cotree edges with respect to T, leaving a set S of cotree edges called "single edges". Let $H_0 = T + A$, and we know H_0 is an upper-embeddable spanning

subgraph, i.e. $\xi(H_0) = 0$. Then find a sequence of subgraphs of G with zero deficiency $H_0 \subset H_1 \subset H_2 \cdots \subset H_i \cdots \subset H_j$ such that for all $i = 0, 1, \ldots, j-1$, $H_{i+1} = H_i + e + f$ where $e, f \in S$ and $\gamma_M(H_j) = \gamma_M(G)$.

Nevertheless, we find the proof of theorem 4.3.2 in [32] is not correct. In any *regular frame decomposition* of G defined by the author, there exist a *free leaf* L and a pair of adjacent edges (e, f) such that $e \in L, f \notin L, e, f \in H$ where H is a spanning subgraph of G. However, even though $\xi(G) = \xi(G - e - f)$, $\xi(H - e - f) = \xi(H)$ may not be true. The reason is that a regular frame decomposition of G may not be necessarily a regular frame decomposition of its spanning subgraph H. In terms of Xuong tree, even if e, f are adjacent cotree edges with respect to a Xuong tree T of G, T may not be a Xuong tree of the spanning subgraph H of G, so $\xi(H - e - f) = \xi(H)$ may not hold.

C. Bounds on Maximum Vertex-Disjoint Cycles

Maximum Vertex-Disjoint Cycles Problem is a known optimization problem with the goal of packing maximum vertex-disjoint cycles in a given graph. The deterministic version has been proved NP-Complete. As a byproduct of exploring properties of fundamental cycles, from the perspectives of maximum genus and fundamental cycles, we present a lower bound and an upper bound on the maximum number of vertex-disjoint cycles. By existed algorithm in [17], the bounds only depending on maximum genus and cycle rank can be computed in polynomial time. We note that the condition of vertex-disjoint cycles coincides with what we investigate on fundamental cycles: intersection. Hence we have the following theorem.

Theorem C.1. *Given a graph G, the number of maximum vertex-disjoint cycles in G is lower-bounded by $\beta(G) - 2\gamma_M(G)$ and upper-bounded by $\beta(G) - \gamma_M(G)$.*

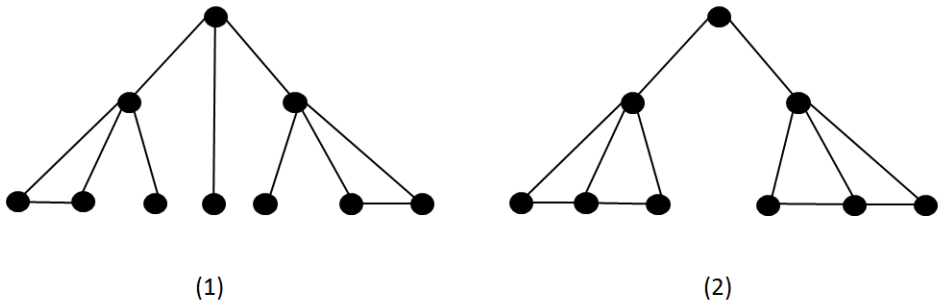


Fig. 14. Cases of tight lower bound and tight upper bound

Proof. First we prove the lower bound by showing there exist $\beta(G) - 2\gamma_M(G)$ vertex-disjoint cycles. By corollary B.1 in the last section, there exists a spanning tree T such that $\gamma_M(G) = \nu(G\sharp T)$. We start from T and construct the intersection graph $G\sharp T$. Then we compute a maximum matching M whose size is equal to $\gamma_M(G)$. Note that for all the unmatched vertices in $G\sharp T$, they correspond to fundamental cycles in G with respect to T which are also cycles in G. Since M is a maximum matching, it is a maximal matching. Thus, any pair of unmatched vertices can not be adjacent, that is, vertex-disjoint. Since the number of vertices in any intersection graph of G is always equal to the cycle rank $\beta(G)$, the number of unmatched vertices by M is equal to $\beta(G) - 2|M|$, that is, $\beta(G) - 2\gamma_M(G)$.

For the upper bound, we recall Xuong tree introduced in chapter 2. Let T be a Xuong tree of G. The number of cotree components of odd edges is the minimum taking over all the spanning trees of G. Given T we can find a maximum pairing of adjacent cotree edges by the manner described in [17] in polynomial time. Let A denote the set of maximum pairs of adjacent cotree edges, and by definition of Xuong tree we know $|A| = \gamma_M(G)$. For each pair $(e, f) \in A$, e is incident with f. Thus, $T(e)$ and $T(f)$ must have at least one vertex in common, i.e. the intersection vertex of e and f. Therefore, at most one of $T(e)$, $T(f)$ and $T(e) \cup T(f)$ may appear in any

set of maximum vertex-disjoint cycles of G. There are $\beta(G)$ fundamental cycles with respect to T and $|A| = \gamma_M(G)$, so we get the upper bound $\beta(G) - \gamma_M(G)$.

Now we show the cases that the bounds are tight. Clearly both of the lower and upper bounds can be tight at the same time for all those graphs whose maximum genus equals zero, i.e. trees. In Fig. 14, (1) is the case that a graph with maximum genus zero and both bounds become tight, and (2) is the case that upper bound $\beta(G) - \gamma_M(G)$ can be reached. □

CHAPTER V

SUMMARY

In this thesis, we study two optimization problems: Line Cover Problem (LCP) in Computational Geometry and Maximum Genus Embedding (MGE) in Topological Graph Theory. Both problems have quite a few important practical applications directly in many domains.

We further study LCP from different aspects including classical techniques in algorithm design and parameterized algorithms. We propose an exact algorithm and survey related geometric properties. Then we initiate another optimization problem which is the dual problem of LCP in terms of linear programming duality. We do kernelization and approximation on the dual problem and present some observations. Finally, we propose a parameterized algorithm to solve LCP. One possible work in future is to study two variants of LCP, i.e. to find either covering "bands" with a fixed width or covering curves rather than straight lines.

We investigate the relationship between the maximum genus of a graph and its cycle space consisting of fundamental cycles. We restudy an approach of computing a maximum pairing of intersected fundamental cycles with respect to an arbitrary spanning tree. It is known incorrect and we investigate the reason it fails. Also, we characterize the upper-embeddability of a graph in terms of maximum pairings of intersected fundamental cycles. Finally, we present a lower and an upper bound of the maximum number of vertex-disjoint cycles in general graphs, in terms of maximum genus and cycle rank. One possible future work is to explore any other polynomial approach to solve MGE instead of using reduction to matroid parity.

REFERENCES

[1] J. Beck, "On the lattice property of the plane and some problems of dirac, motzkin, and erdos in combinatorial geometry," *Combinatorica*, vol. 3, pp. 281–297, 1983.

[2] ——, "Remarks on combinatorial geometry 1," *Stud. Sci. Math. Hungar.*, vol. 20, pp. 249–254, 1985.

[3] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. New York: North-Holland, 1976.

[4] P. Brass, W. Moser, and J. Pach, *Research Problems in Discrete Geometry*. Springer, 2005.

[5] Y. Cao, J. Chen, and Y. Liu, "On feedback vertex set: new measure and new structures," Proc. 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2010). Lecture Notes in Computer Science 6139, 2010.

[6] J. Chen, "Algorithmic graph embeddings," *Theoretical Computer Science*, vol. 181, pp. 247–266, 1997.

[7] J. Chen, D. Archdeacon, and J. L. Gross, "Maximum genus and connectivity," *J. Discrete Math.*, vol. 149, pp. 19–29, 1996.

[8] J. Chen and S. P. Kanchi, "Graph ear decomposition and graph embeddings," *SIAM J. Discrete Math.*, vol. 12, pp. 229–242, 1999.

[9] J. Chen, S. P. Kanchi, and J. L. Gross, "A tight lower bound on the maximum genus of a simplicial graph," *J. Discrete Math.*, vol. 156, pp. 83–102, 1996.

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms (3rd ed.)*. Massachusetts: MIT Press, 2009.

[11] J. Csima and E. Sawyer, "The $6n/13$ theorem revisited," *Graph Theory, Combinatorics, Algorithms and Applications*, vol. 1, pp. 235–249, 1995.

[12] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications (3rd ed.)*. Springer, 2008.

[13] P. Erdos, "On a problem of grunbaum," *Can. Math. Bull.*, vol. 15, pp. 23–25, 1972.

[14] P. Erdos and G. Purdy, "Some combinatorial problems in the plane," *J. Combin. Theory*, vol. A 20, pp. 205–210, 1978.

[15] F. Fomin and D. Kratsch, *Exact Exponential Algorithms*. Springer, 2010.

[16] H. Fu, M. Skoviera, and M. Tsai, "The maximum genus, matchings and the cycle space of a graph," *J. Czechoslovak Math.*, vol. 48, pp. 329–339, 1998.

[17] M. L. Furst, J. Gross, and L. A. McGeoch, "Finding a maximum genus graph imbedding," *J. Asso. Comp. Mach.*, vol. 35, pp. 523–534, 1988.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Fressman, 1979.

[19] P. Giannopoulos, C. Knauer, and S. Whitesides.

[20] M. Grantson and C. Levcopoulos, "Covering a set of points with a minimum number of lines," *Lecture Notes in Computer Science*, vol. 3998, pp. 6–17, 2006.

[21] J. L. Gross and R. G. Rieper, "Local extrema in genus-stratified graphs," *J. Graph Theory*, vol. 15, pp. 159–171, 1991.

[22] J. L. Gross and T. W. Tucker, *Topological Graph Theory.* New York: Wiley-Interscience, 1987.

[23] B. Grunbaum, "New view of some old questions of combinatorial geometry," *Colloq. Int. Theorie Comb. Roma 1973*, vol. 1, pp. 451–468, 1976.

[24] L. Guibas, M. Overmars, and J. Robert, "The exact fitting problem in higher dimensions," *Computational Geometry Theory Application*, vol. 6, pp. 215–230, 1996.

[25] Y. Huang, "Maximum genus of a graph in terms of its embedding properties," *J. Discrete Math.*, vol. 262, pp. 171–180, 2003.

[26] Y. Huang and Y. Liu, "Maximum genus and maximum nonseparating independent set of a 3-regular graph," *J. Discrete Math.*, vol. 176, pp. 149–158, 1997.

[27] M. Jungerman, "A characterization of upper embeddable graphs," *Trans. Am. Math. Soc.*, vol. 241, pp. 401–406, 1978.

[28] L. M. Kelly and W. O. J. Moser, "On the number of ordinary lines determined by n points," *Can. J. Math.*, vol. 10, pp. 210–219, 1958.

[29] M. Kotrbcik and M. Skoviera, "Fundamental cycles and the maximum genus of a graph," 2011, presented on the webpage of authors' department.

[30] V. Kumar, S. Arya, and H. Ramesh, "Hardness of set cover with intersection 1," Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP'00). Springer, Berlin, 2000.

[31] S. Kundu, "Bounds on number of disjoint spanning trees," *J. Combin. Theory*, vol. B 17, pp. 199–203, 1974.

[32] B. Kusý, "Algorithms to embed graphs into surfaces," Master's thesis, Comenius University, Bratislava, Slovakia, 2002.

[33] S. Langerman and P. Morin, "Covering things with things," *Discrete Computational Geometry*, vol. 33, pp. 717–729, 2005.

[34] M. Megiddo and A. Tamir, "On the complexity of locating linear facilities in the plane," *Operations research letter*, vol. 1, pp. 194–197, May 1982.

[35] S. Micali and V. V. Vazirani, "An $o(\sqrt{|V|}|e|)$ algorithm for finding maximum matching in general graphs." Proc. 21th IEEE Symp. Foundations of Computer Science (FOCS 1980), 1980.

[36] L. Nebeský, "Every connected, locality connected graph is upper imbeddable," *J. Graph Theory*, vol. 5, pp. 205–207, 1981.

[37] ——, "Characterizing the maximum genus of a connected graph," *J. Czechoslovak Math.*, vol. 43, pp. 177–183, 1993.

[38] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*. New York: Oxford University Press, 2006.

[39] E. Nordhaus, B. Stewart, and A. T. White, "On the maximum genus of a graph," *J. Comb. Theory Ser.*, vol. B 11, pp. 258–267, 1971.

[40] H. Ren, H. Zhao, and H. Li, "Fundamental cycles and graph embeddings," *Sci. China Ser. A*, vol. 52, pp. 1920–1926, 2009.

[41] R. Ringeisen, "The maximum genus of a graph," Ph.D. dissertation, Michigan State University, East Lansing, Michigan, 1970.

[42] ——, "Determining all compact orientable 2-manifolds upon which $k_{m,n}$ has 2-cell imbeddings," *J. Comb. Theory*, vol. B 12, pp. 101–104, 1972.

[43] M. Skoviera, "The maximum genus of graphs of diameter two," *Discrete Math.*, vol. 87, pp. 175–180, 1991.

[44] J. J. Sylvester, "Mathematical question 11851," *Educational Times*, vol. 46, p. 156, 1893.

[45] C. Thomassen, "The graph genus problem is np-complete," *J. Algorithms*, vol. 10, pp. 568–576, 1989.

[46] J. Wang, W. Li, and J. Chen, "A parameterized algorithm for the hyperplane-cover problem," *Theoretical Computer Science*, vol. 411, pp. 4005–4009, 2010.

[47] A. T. White, *Graphs, Groups, and Surfaces*. Amsterdam: North-Holland, 1984.

[48] N. H. Xuong, "How to determine the maximum genus of a graph," *J. Combin. Theory*, vol. B 26, pp. 217–225, 1979.

[49] ——, "Upper-embeddable graphs and related topics," *J. Combin. Theory*, vol. B 26, pp. 226–232, 1979.

[50] J. Zaks, "The maximum genus of cartesian products of graphs," *Can. J. Math.*, vol. 26, pp. 1025–1035, 1974.