Straight Line Movement in Morphing and Pursuit Evasion

by

Hamideh Vosoughpour Yazdchi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2017

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Anil Maheshwari Professor
Supervisor	Anna Lubiw Professor
Internal Member	Therese Biedl Professor
Internal Member	Timothy Chan Professor
Internal-external Member	Penny Haxell Professor

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Piece-wise linear structures are widely used to define problems and to represent simplified solutions in computational geometry. A piece-wise linear structure consists of straight-line or linear pieces connected together in a continuous geometric environment like 2D or 3D Euclidean spaces. In this thesis two different problems both with the approach of finding piece-wise linear solutions in 2D space are defined and studied: straight-line pursuit evasion and straight-line morphing.

Straight-line pursuit evasion is a geometric version of the famous cops and robbers game that is defined in this thesis for the first time. The game is played in a simply connected region in 2D. It is a full information game where the players take turns. The cop's goal is to catch the robber. In a turn, each player may move any distance along a straight line as long as the line segment connecting their current location to the new location is not blocked by the region's boundary. We first prove that the cop can always win the game when the players move on the visibility graph of a simple polygon. We prove this by showing that the visibility graph of a simple polygon is "dismantlable" (the known class of cop-win graphs). Polygon visibility graphs are also shown to be 2-dismantlable. Two other settings of the game are also studied in this thesis: when the players are free to move on the infinitely many points inside a simple polygon, and inside a splinegon. In both cases we show that the cop can always win the game. For the case of polygons, the proposed cop strategy gives an asymptotically tight linear bound on the number of steps the cop needs to catch the robber. For the case of splinegons, the cop may need a quadratic number of steps with the proposed strategy, while our best lower bound is linear.

Straight-line morphing is a type of morphing first defined in this thesis that provides a nice and smooth transformation between straight-line graph drawings in 2D. In straight-line morphing, each vertex of the graph moves forward along the line segment connecting its initial position to its final position. The vertex trajectories in straight-line morphing are very simple, but because the speed of each vertex may vary, straight-line morphs are more general than the commonly used "linear morphs" where each vertex moves at uniform speed. We explore the problem of whether an initial planar straight-line drawing of a graph can be morphed to a final straight-line drawing of the graph using a straight-line morph that preserves planarity at all times. We prove that this problem is NP-hard even for the special case where the graph drawing consists of disjoint segments. We then look at some restricted versions of the straight-line morphing: when only one vertex moves at a time, when the vertices move one by one to their final positions uninterruptedly, and when the edges morph one by one to their final configurations in the case of disjoint segments. Some of the variations are shown to be still NP-complete while some others are solvable

in polynomial time. We conjecture that the class of planar straight-line morphs is as powerful as the class of planar piece-wise linear straight-line morphs. We also explore a simpler problem where for each edge the quadrilateral formed by its initial and final positions together with the trajectories of its two vertices is convex. There is a necessary condition for this case that we conjecture is also sufficient for paths and cycles.

Acknowledgements

I would like to greatly thank my supervisor, Anna Lubiw. Truly, success in this journey might not have been possible without her guidance, mentorship, and support in all of my different situations. Meeting Anna and working with her was one of my greatest opportunities in life, both professionally and personally. She mentored me like a friend from the very early days to the last day.

I would like to thank Therese Biedl and Timothy Chan. Not only because of their great advice and comments on my thesis as committee members, but also because of all I have learned from them during my PhD studies, especially in weekly problem sessions. Also, many thanks to all other participants of the problem sessions these years. I learned a lot from them, and it was a joyful and unique experience for me.

I would like to thank the external examiner, Anil Maheshwari, and the internal/external committee member, Penny Haxell, for reading my thesis carefully and for their valuable comments and suggestions that improved the quality of the thesis. Also, many thanks to Nicole Keshav for reading my drafts and helping me to improve my writing skills.

The first part of my thesis is a joint work with Jack Snoyeink and Anna Lubiw. I would like to thank Jack for his meticulous discussions, and others who participated in discussing the problem during the CCCG2014 conference.

I would like to thank my friends in Waterloo and in the Cheriton School of Computer Science who supported me all the time and made my most memorable moments during these years. These friendships are one of my most valuable achievements during my PhD studies in Canada.

I would like to thank my family. My parents were available to help whenever we needed them. Without their boundless love and support, I would not have accomplished this. My brother, Meisam, encouraged and motivated me to begin this journey. My husband, Mohsen, supported me unconditionally all the time both technically and emotionally and I would not be here without him. And finally, my little son, Ryan, whose endless cuteness made the most joyful moments when I was writing the thesis.

Dedication

To Mohsen

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Geometric objects and geometric problems in their real representations and definitions can be too complex to represent or to solve. To simplify matters, piece-wise linear structures are widely used to define problems and represent geometric objects in computational geometry. A piece-wise linear structure consists of straight-line or linear pieces which are connected together in a continuous geometric environment like 2D or 3D Euclidean spaces. For example connecting points in 2D via a set of straight-line segments in the presence of obstacles or some other geometric constraints is a fundamental concept widely used to define and solve problems in computational geometry. Piece-wise linear structures are also called polygonal chains or poly-lines especially when the problem is defined in 2D space. Polygons and polyhedra are examples of piece-wise linear structures widely used to define the boundary of objects or regions in 2D and 3D.

Piece-wise linear solutions are also interesting in some problems because they are simpler to visualize, easier to represent, and can be computed with less complexity. However, piece-wise linear solutions are not always optimal. For example, a minimum link path between two points in a polygonal region with the presence of obstacles is a piece-wise linear path with the minimum number of bends. However, a shortest length path between two points in a region with curved boundaries is not necessarily piece-wise linear.

Even though simplifying a problem into the piece-wise linear version usually makes the problem easier, sometimes it defines a totally new problem with a different solution. A famous example is the Lion-and-Man problem, where a lion is trying to catch a man in a circular arena. In the continuous version, the man can escape indefinitely. However, if the players take turns, which means the path each player traverses is piece-wise linear, the lion may catch the man in finite time [64].

Finding shortest paths, motion planning, polygon visibility problems, and morphing graph drawings are a few more examples where polygonal chains are used to define a problem or to represent an acceptable solution for a hard problem. Sometimes we break down a complex solution, such as a solution to motion planning or morphing problems, into a series of simple steps such as linear steps. Then, an acceptable solution can be achieved in the form of a piece-wise linear structure. In problems like morphing or motion planning in which timing is included, time can be seen as an extra dimension. Then, a piece-wise linear solution in the higher dimensional space (including time as an extra dimension) might be a desirable solution for the original problem.

In this thesis, we consider two interesting but very different problems both with the approach of finding piece-wise linear solutions in 2D space: Straight-line pursuit evasion and straight-line morphing.

The first problem, straight-line pursuit evasion, is a version of the popular cops and robbers game in which the cop and robber move on straight lines inside a polygon or a region with curved boundaries. In straight-line pursuit evasion there is an evader (or robber) and a pursuer (or cop) playing in a simply connected region in 2D. Each player is modeled by a single point. The game starts with the cop choosing a point inside the region to start. Then the robber chooses its initial position accordingly. The game is full information, so both players know everything about the environment and the other player's location. They start taking turns. The cop and the robber in their turns may move on a straight-line to a new point inside the region if the new point is "visible" from the their current location. The game finishes if the cop in its turn catches the robber by moving onto the robber's location. Then the cop wins and the game is called cop-win. Otherwise, if the robber can escape indefinitely the game will be robber-win.

The version of the game we consider in this thesis is very close to the original cops and robbers game introduced by Nowakowski and Winkler [68], and Quilliot [72] in 1983. The original game is defined on undirected graphs, and the players are located on the vertices and move along the edges of the graph in their turns. The version of the game in this thesis can be seen as the geometric version of the original game played on the infinite visibility graph of all points inside a simply connected region.

There are many different variations of the cops and robbers game defined in the literature. Continuous versions of the problem in geometric environments, which are also interesting in robotics and video surveillance applications, are usually called "pursuit evasion". The Lion-and-Man problem in a circular arena from the 1930's is the oldest pursuit evasion problem in this category (see Littlewood [64]).

In continuous settings the players may move continuously with the same maximum

speed. The pursuit evasion problem in this continuous setting is a harder problem. For the simple version where the region is a circular arena in the Lion-and-Man problem, the man can escape only by moving on a precise curve which is not very intuitive. One approach to make the problem simpler is to force the players to take turns and make a limited movement in each turn. Then, the trajectory of each player will be piece-wise linear which is easier to solve and represent. However, as mentioned earlier in regards to the Lion-and-Man problem, this simplification may change the solution of the problem significantly.

In the versions of the pursuit evasion problem that are closest to our setting, the players take turns and are limited by the maximum distance the players may move in each turn. [12, 10]. In the straight-line pursuit evasion problem we defined, the limitation is actually on the link distance instead of the Euclidean distance. Each player may move to any point with link distance one on its turn. This can be a more natural limitation in real mobile robot navigation as moving forward on a straight-line can be done with higher speed than changing directions several times in each turn. Moreover, the cost and delay caused by changing directions can naturally simulate taking turns by the players. Even though this is still not a real model, it seems closer to real situations than taking turns with a maximum distance restriction in each turn.

There is no previous study on cops and robbers on visibility graphs, and this is the first time that this version of the pursuit evasion problem has been considered, even though visibility is central to many pursuit evasion problems.

In this thesis, we study the straight-line pursuit evasion problem in three different settings and show that the cop wins in all three cases. The first setting is the game on the "visibility graph" of a simple polygon. The players are on the vertices of a simple polygon. They may move from vertex v to a vertex u in their turns if and only if vertices v and u are visible inside the polygon.

In the second setting of straight-line pursuit evasion the game is extended to all the infinitely-many points inside the simple polygon. This game is also proved to be cop-win thus providing a non-trivial infinite cop-win graph. We also show that the number of steps the cop needs to capture the robber is at most n with the suggested strategy, where n is the number of vertices of the polygon. There are examples where $\Omega(n)$ steps are needed, so the bound is tight.

In the last setting of the game we prove that the game is still cop-win if the boundary of the region consists of simple curves (splines) instead of line segments. Such a region is called a "splinegon" instead of a polygon. In our suggested strategy for the cop, it takes $O(n^2 + d)$ steps for the cop to catch the robber, where n is the number of curve segments forming the boundary of the region and d is the link diameter of the region. We cannot

do better than $\Omega(n+d)$ steps.

The second problem we consider in this thesis with the approach of piece-wise linear solutions is *morphing*. By morphing we mean a smooth and continuous transformation from a (planar) drawing of a graph to another (planar) drawing of the same graph. Morphing is a relatively new term used since the 1980's and has attracted much attention in graph drawing [4, 14, 33]. The motivation of studying the special case of morphing of poly-lines is that there are some applications that can be modeled as transformation of one curve/poly-line to another curve/poly-line in a smooth and natural way. Morphing hand-writing or morphing a curve representing a road or river between geographic maps of different scales are some examples [33, 71]. In these examples the main challenge is to find a good correspondence between the points of two curves/poly-lines. However, in graph drawing morphs the correspondence of graph vertices is usually assumed to be known [21].

Planarity is usually the main property we want to preserve in the morphing of planar graph drawings. It is also desirable to be visually clear how the initial drawing is changed into the final drawing. In most studies the morph is broken into simple steps like *linear morphs* where each vertex moves along a straight line with uniform speed [4]. This leads to a piece-wise linear trajectories for the vertices. However, the trajectories might be complicated and not nicely visualizable.

In this thesis we define *straight-line morphing*. A straight-line morph from a planar straight-line drawing of a graph to another planar straight-line drawing is a morph where the trajectory of each vertex is the straight-line segment connecting its initial position to its final position. Straight-line morphs are similar to linear morphs since the vertex trajectories are the same. In straight-line morphs, however, there is freedom of how each vertex travels this simple trajectory. For example, a vertex might travel half-way to its target position and pause there while some other vertices move before it finally completes its journey to the target position. This property makes straight-line morphing more flexible than linear morphing but more constrained than piece-wise linear morphing.

Straight-line morphing with this definition is a new concept introduced for the first time in this thesis. The difference between a linear morph and a straight-line morph will be clearer if we think of time as an extra dimension. Assume that the entire morph happens in a unit time. Then in a linear morph the motion of each vertex in 3D space will be a single line segment connecting point $(x_0, y_0, 0)$ to point $(x_1, y_1, 1)$ (the first two dimensions indicate the vertex position and the last dimension is time). By contrast, in a straight-line morph the trajectory in 3D space is not limited to a single line segment. Our definition of straight-line morphs allows vertices to move along their straight-line trajectories with speeds that are arbitrary functions of time. However, all the positive solutions we give

for straight-line morphing involve piece-wise linear motions, and we conjecture that if a solution exists to an instance of straight-line morphing, then a solution exists in which the motion is piece-wise linear, i.e., the path in 3D space from $(x_0, y_0, 0)$ to $(x_1, y_1, 1)$ is a poly-line whose projection to the first two coordinates is a straight line segment.

In this thesis we explore straight-line morphs for the case of disjoint segments, paths, and cycles. We prove that deciding whether a planar straight-line morph exists is NP-hard even for a set of disjoint segments. However, we have not shown that the problem lies in NP and we still do not have any algorithm to decide the existence of straight-line morphs even for a limited category of graphs. Therefore, we had to narrow our focus to more restricted versions of the problem.

We narrow our focus in two ways: by restricting the input or by adding more limitations to the straight-line morphs. In the first part, we simplify the input, for example by eliminating the cases where some edges morph through a non-convex quadrilateral in the morph. For the second part, more specific versions of the morphs are introduced. For example, we consider straight-line morphs where only one vertex or edge is moving at each time while the rest of the graph is stationary, or even where the moving edge/vertex morphs to the final position without interruption.

Most of the restricted versions of straight-line morphs considered in this thesis are simple enough to have exponential time algorithms. For some versions we even have polynomial time algorithms, while some versions are shown to be still NP-hard. However, for the case of one-vertex-at-a-time morphs we still do not know if the additional restriction has decreased the complexity of the problem.

For the case where we simplified the input and limit it to edges with convex morphing quadrilaterals, we develop a good necessary condition that is testable in polynomial time and may rule out some negative examples. Even though this necessary condition is not sufficient in general, we conjecture that for the case of paths and cycles it is a sufficient condition.

The thesis is organized as follows. Part I (Chapters 2 to 5) is about pursuit evasion. Chapter 2 is the introduction to the problem and the background on cops and robbers, pursuit evasion, and splinegons. The cops and robbers problem on the visibility graph of the polygon and on all points inside the polygon is in Chapter 3. Chapter 4 is for the cops and robbers problem in splinegons (the regions with curved boundaries). The last chapter in Part I contains the conclusions and open problems for pursuit evasion.

Part II (Chapters 6 to 9) is about straight-line morphing. Chapter 6 is the introduction and background about morphing. In Chapter 7 straight-line morphing in its original settings is considered on disjoint segments, paths, cycles, and on a restricted category of

morphing graphs (convex morphing) and the results are discussed. In Chapter 8 the more restricted straight-line morph definitions, where only one vertex/edge moves at a time, are considered. Chapter 9 contains the conclusion and open problems for the straight-line morphing problem.

Part I

Straight Line Pursuit Evasion

Chapter 2

Pursuit Evasion: Introduction and Background[1]

Cops and robbers games, also known as pursuit-evasion games, have a rich history both for their mathematical interest and because of applications in surveillance, search-and-rescue, and mobile robotics.

In pursuit-evasion games one player, called the "evader," tries to avoid capture by "pursuers" as all players move in some domain. There are many game versions, depending on whether the domain is discrete or continuous, what information the players have, and how the players move: taking turns, moving with bounded speed, etc. The "cops and robbers game" mostly refers to discrete versions of the game on graphs, while "pursuit-evasion" often refers in the literature to the variations of the game in continuous domains such as mobile robotics and geometric environments.

Here, we consider a geometric version of the "cops and robbers game," in the domain of straight-line visibility in planar regions. Our version of the problem can also be related to "pursuit evasion" problems, since the players move in continous space in polygons or planar regions with curved boundaries. The rules of our game are the same as the original cops and robbers game first introduced in 1983 by Nowakowski and Winkler [68], and Quilliot [72]. The cop and robber are located at vertices of a graph and take turns moving along edges of the graph. The robber is caught when a cop moves to the vertex the robber is on. The standard assumption is that both players have full information about the graph and the

[1]This part of the thesis represents joint work with Anna Lubiw and Jack Snoeyink and has appeared as journal publication [65]

other player's location. The first papers on this game [68, 72] characterized the graphs in which the cop wins; they are "dismantlable" graphs that are defined in the following.

Definition 2.1 (**Dismantlable Graph**). *For a vertex v of a graph, let $N[v]$ be the* closed neighbourhood *of v, which consists of v together with the vertices adjacent to v. Vertex v* dominates *vertex u if $N[v] \supseteq N[u]$. A graph G is* dismantlable *if it has a vertex ordering $\{v_1, v_2, \ldots, v_n\}$ such that for each $i < n$, there is a vertex v_j, $j > i$ that dominates v_i in the graph G_i induced by $\{v_i, \ldots, v_n\}$.*

Since then many extensions of the cops and robbers game have been explored; see the book by Bonato and Nowakowski [16]. These extensions are mostly about making the players stronger or weaker, adding uncertainty to the game, or providing the player with partial information about the environment and other players. Note that the cops and robbers version that Seymour and Thomas [73] develop to characterize treewidth is different: the robber moves only along edges but at arbitrarily high speed, while a cop may jump to any graph vertex.

In this thesis we consider three successively more general versions of the cops and robbers game in planar regions. The first version is the cops and robbers game on the *visibility graph of a polygon*. We regard a polygon as a closed set of points, the interior plus the boundary. Two points in a polygon are *visible* or *see* each other if the line segment between them lies inside the polygon. The line segment may lie partially or totally on the boundary of the polygon. The *visibility graph* of a polygon has the same vertex set as the polygon and an edge between any pair of vertices that see each other in the polygon.

We prove that this game is cop-win by proving that visibility graphs are dismantlable. As explained below, this result is implicit in [1]. We prove the stronger result that visibility graphs are "2-dismantlable." The definition is in Chapter 3. We remark that it is an open problem to characterize or efficiently recognize visibility graphs of polygons [42, 43], and furthermore, the class of visibility graphs of polygons is not contained in any of the well-studied graph classes. Thus this result is significant in that it places visibility graphs as a subset of the class of dismantlable graphs which is known and can be recognized in polynomial time [68]. Therefore, as the first step toward recognizing visibility graphs we might test whether the graph is dismantlable (or even stronger if it is 2-dismantlable).

Our second setting is the cops and robbers game on all points inside a polygon. The cop chooses a point inside the polygon as its initial position, then the robber chooses its initial position. Then the players take turns, beginning with the cop. In each turn, a player may move to any point visible from its current location, i.e., it may move any distance along a straight-line segment inside the polygon. The cop wins when it moves to the robber's position.

For any point x in polygon P, the *visibility polygon* of x, $V(x)$, is the set of points in P visible from x. Note that $V(x)$ may fail to be a simple polygon; it may have 1-dimensional features on its boundary in certain cases where x lies on a line through a pair of vertices. We prove that the cop will win using the simple strategy of always taking the first step of a shortest path to the robber. Thus the cop plays on the reflex vertices of the polygon.

Our third setting is the cops and robbers game on all points inside a bounded simply-connected planar region. We show that if the boundary is well-behaved (see below) then the cop wins. We give a strategy for the cop to win, although the cop can no longer follow the same shortest path strategy (e.g. when it lies on a reflex curve), and can no longer win by playing on the boundary.

The cops and robbers game on all points inside a region can be viewed as a cops and robbers game on an infinite graph—the graph has a vertex for each point inside the region, and an edge when two points see each other. These "point visibility graphs" were introduced by Shermer (see [74]) for the case of polygons. Our result shows that point visibility graphs are cop-win. This provides an answer to Hahn's question [47] of finding an interesting class of infinite cop-win graphs.

The cops and robbers game on all points inside a region can be viewed as pursuit-evasion under a different metric, and could appropriately be called "straight-line pursuit-evasion." Previous work [53, 12] considered a pursuit-evasion game in a polygon (or polygonal region) where the players are limited to moving distance 1 in the Euclidean metric on each turn. In our game, the players are limited to distance 1 in the *link metric*, where the length of a path is the number of line segments in the path. This models a situation where changing direction is costly but straight-line motion is easy. Mechanical robots cannot make instantaneous sharp turns so exploring a model where all turns are expensive is a good first step towards a more realistic analysis of pursuit-evasion games with turn constraints. We also note that the protocol of the players taking alternate turns is more natural in the link metric than in the Euclidean metric.

The rest of this chapter gives the background and some related studies for this part of the thesis. This includes the background on cops and robbers and pursuit evasion problems. Moreover, we look at the background studies about regions with curved boundaries at the end of this chapter. Chapter 3 is about our first two settings of the straight-line pursuit evasion game, i.e., the cops and robbers game on the visibility graph of a polygon and on all points inside the polygon. Chapter 4 is about the cops and robbers game in planar regions with curved boundaries which is our third setting. Chapter 5 is about some open problems and discussions.

2.1 Background on Cops and Robbers

The cops and robbers game was introduced by Nowakowski and Winkler [68], and Quilliot [72] in 1983. They characterized the finite graphs where one cop can capture the robber ("cop-win" graphs) as "dismantlable" graphs, which can be recognized efficiently; see the definition of dismantlable graphs earlier in this chapter.

Theorem 2.2. *[68] A finite undirected graph is cop-win if and only if it is dismantlable.*

A year later Aigner and Fromme [2] introduced the *cop number* of a graph, the minimum number of cops needed to catch a robber. The rule with multiple cops is that they all move at once. In general, if we put one cop on each vertex of the graph, the cops will win immediately when the game begins. Thus, the cop number is $O(n)$. Among other things Aigner and Fromme proved that three cops are always sufficient and sometimes necessary for planar graphs. Beveridge et al. [11] studied what they called *geometric graphs* where vertices are points in the plane and an edge joins points that are within distance 1 and showed that 9 cops suffice, and 3 are sometimes necessary. Meyniel conjectured that $O(\sqrt{n})$ cops can catch a robber in any graph on n vertices [8]. While Meyniel's conjecture is still open, Frankl proved a sub-linear bound on the cop number is $O(n\frac{\log\log n}{\log n})$ cops [38]. Frankl's bound was then improved by Chiniforooshan to $O(\frac{n}{\log n})$ cops [26].

For any fixed k there is a polynomial time algorithm to test if k cops can catch a robber in a given graph, but the problem is NP-complete for general k [35], and EXPTIME-complete for directed graphs [45].

The cops and robbers game on infinite graphs was studied in the original paper [68] and others, e.g. [15]. Unlike finite graphs, in the case of infinite graphs being dismantlable is not sufficient for the cop to win the game. A simple counter-example is an infinite path (ray) which is dismantlable as each vertex is dominated by its next vertex in the path, but is not cop-win because the robber may escape infinitely.

In a cop-win graph with n vertices, the capture time is the maximum number of moves that the cop must take to capture the robber, where the maximum is taken over all robber strategies. It is implicit in the proof of Theorem 2.2 that the cop can win in at most n moves. In the book by Bonato and Nowakowski [16, Section 2.2] it is shown that the capture time for cop-win graphs is at most $n-3$ and they showed an example where the capture time is $n-4$. This gap was resolved by Gavenčiak by showing that the capture time is at most $n-4$ for $n \geq 7$ [41]. For three cops playing on planar graphs, Pisantechakool and Tan showed that the capture time can be at most $2n$ [70].

When k cops are playing in a graph with n vertices, a rough upper bound of n^{k+1} for the capture time is obtained by counting the number of different states of the $k+1$ players on n vertices because we may avoid a repetitive state in a winning strategy. Recently in 2017, Brandt et al. proved that this upper bound is tight for $n \geq 2$ [18]. For a fixed number of cops, the number of cop moves needed to capture a robber in a given graph can be computed in polynomial time [49], but the problem becomes NP-hard in general [15].

Many other variations of the cops and robbers game have been studied [54, 40, 55, 23, 36, 13]. Isler and Karnad [55] explore the case where the cop only knows the robber's location when the robber is within some bounded distance from the cop. The cop can still catch the robber with a randomized strategy, but the number of moves increases. Chalopin et al. [23] studied the game when the robber can hide while playing and is visible every k steps. They characterize the cop-win graphs for any value of k. Isler and Karnad [55] considered the role of information for the game of cop and robbers and showed that the game is still cop-win even if the cop does not have any information about the robber's position, but the capture time will be exponential in that case.

2.2 Background on Pursuit-Evasion

In the cops and robbers game, space is discrete. For continuous spaces, the main focus has been on polygonal regions, i.e., a region bounded by a polygon with polygonal holes removed. The seminal 1999 paper by Guibas et al. [46] concentrated on "visibility-based" pursuit-evasion where the evader is arbitrarily fast and the pursuers do not know the evader's location and must search the region until they make line-of-sight contact. This models the scenario of agents searching the floor-plan of a building to find a smart, fast intruder that can be zapped from a distance. Guibas et al. [46] showed that $\Theta(\log n)$ pursuers are needed in a simple polygon, and more generally they bounded the number of pursuers in terms of the number of holes in the region. If the pursuers have the power to make random choices, Isler et al. [53] showed that only one guard is needed for a polygon. For a survey on pursuit-evasion in polygonal regions, see [27].

The two games (cops and robbers/visibility-based pursuit-evasion) make opposite assumptions on five criteria: space is discrete/continuous; the pursuers succeed by capture/line-of-sight; the pursuers have full information/no information; the evader's speed is limited/unlimited; time is discrete/continuous (i.e., the players take turns/move continuously).

The difference between players taking turns and moving continuously can be vital, as revealed in Rado's Lion-and-Man problem from the 1930's (see Littlewood [64]) where the

two players are inside a circular arena and move with equal speed. The lion wins in the turn-taking protocol, but—surprisingly—the man can escape capture if both players move continuously.

Bhaduaria et al. [12] consider a pursuit-evasion game using a model very similar to ours. Each player knows the others' positions (perhaps from a surveillance network) and the goal is to actually capture the evader. Players have equal speed and take turns. In a polygonal region they show that 3 pursuers can capture an evader in $O(nd^2)$ moves where n is the number of vertices and d is the diameter of the polygon. They also give an example where 3 pursuers are needed. In a simple polygon they show that 1 pursuer can capture an evader in $O(nd^2)$ moves. This result, like ours, can be viewed as a result about a cop and robber game on an infinite graph. The graph in this case has a vertex for each point in a polygon, and an edge joining any pair of points at distance at most 1 in the polygon. To the best of our knowledge, the connection between this result and cops and robbers on (finite) geometric graphs [11] has not been explored. Recently in 2017, Beveridge and Cai [10] extended the result by Bhaduaria et al. [12] from polygonal regions to the regions with more general boundaries like curved boundaries. They showed that in any simple region with piece-wise analytic boundaries one cop always wins and three cops are always sufficient in the case of two dimensional regions with holes.

There is also a vast literature on graph-based pursuit-evasion games, where players move continuously and have no knowledge of other players' positions. Klein and Suri showed that if the players have a maximum unit speed and only line of sight visibility information, $\Theta(n^{1/2})$ pursuers are sufficient and sometimes necessary to catch an evader in a simple polygon [58]. They also showed that the number of pursuers grows up to a lower bound of $\Omega(n^{2/3})$ and upper bound of $O(n^{5/6})$ in the case of a polygonal environment with holes. Emadi et. al. [34] made a connection between pursuit evasion in polygons and dynamic guarding (target tracking). They showed that $\lfloor n/4 \rfloor$ pursuers are sufficient and sometimes necessary to dynamically track a target in a simple polygon (instead of $\lfloor n/3 \rfloor$ for static guarding). For surveys see [5, 37].

2.3 Background on Curved Regions

Traditional algorithms in computational geometry deal with points and piecewise linear subspaces (lines, segments, polygons, etc.). The study of algorithms for curved inputs was initiated by Dobkin and Souvaine [31], who defined the widely-used splinegon model. A *splinegon* is a simply connected region formed by replacing each edge of a simple polygon by a curve of constant complexity such that the area bounded by the curve and the edge

it replaces is convex. The standard assumption is that it takes constant time to perform primitive operations such as finding the intersection of a line with a splinegon edge or computing common tangents of two splinegon edges. This model is widely used as the standard model for curved planar environments in different studies.

Melissaratos and Souvaine [66] gave a linear time algorithm to find a shortest path between two points in a splinegon. Their algorithm is similar to shortest path finding in a simple polygon but uses a trapezoid decomposition in place of polygon triangulation. For finding shortest paths among curved obstacles (the splinegon version of a polygonal domain) there is recent work [25], and also more efficient algorithms when the curves are more specialized [24, 52].

As mentioned in previous section, there has been other recent work by Beveridge and Cai [10] in which the pursuit evasion result by Bhaduaria et al. [12] for polygonal regions is extended to regions with more general boundaries like curved boundaries. They showed that in any simple region with piece-wise analytic boundaries one cop always wins and three cops are always sufficient in the case of two dimensional regions with holes.

Chapter 3

Cops and Robbers in Polygons

In this chapter the cops and robbers problem in polygons is considered. We examine two settings: when the players move only on the vertices of a simple polygon and when they are free to move on any point inside or on the boundary of a simple polygon. We show that the game is cop-win in both settings.

Throughout this chapter the term "polygons" refers to simple polygons.

3.1 Cops and Robbers in Visibility Graphs

In this section we show that the visibility graph of any polygon is cop-win by showing that any such graph is dismantlable.

This result is actually implicit in the work of Aichholzer et al. [1]. They defined an edge uv of polygon P to be *visibility increasing* if for every two points p_1 and p_2 in order along the edge uv the visibility polyons nest: $V(p_1) \subseteq V(p_2)$. In particular, this implies that v dominates every point on the edge, and that v dominates u in the visibility graph. Aichholzer et al. showed that every polygon has a visibility-increasing edge. It is straightforward to show that visibility graphs are dismantlable based on this result.

Lemma 3.1. *The visibility graph G of any polygon P is dismantlable.*

Proof. By induction on the number of vertices of the polygon, let uv be a visibility-increasing edge, which we know exists by the result of Aichholzer et al. Then vertex v

dominates u in the visibility graph G. We will construct a dismantlable ordering starting with vertex u.

It suffices to show that $G - u$ is dismantlable. Let tu and uv be the two polygon edges incident on u. We claim that the triangle tuv is contained in the polygon: u sees t on the polygon boundary, so v must also see t as $V(u) \subseteq V(v)$. (Triangle tuv is an "ear" of the polygon.) Removing triangle tuv yields a smaller polygon whose visibility graph is $G - u$. By induction, $G - u$ is dismantlable. □

Aichholzer et al. [1] conjectured that a polygon always has at least two visibility-increasing edges. In the remainder of this section we prove this conjecture, thus giving a simpler proof of their result and also proving that visibility graphs of polygons are *2-dismantlable*. Bonato et al. [15] define a graph G to be 2-*dismantlable* if it either has fewer than 7 vertices and is cop-win or it has at least two vertices a and b such that each one is dominated by a vertex other than a, b, and such that $G - \{a, b\}$ is 2-dismantlable. They show that if an n-vertex graph is 2-dismantlable then the cop wins in at most $\frac{n}{2}$ moves by choosing the right starting point.

Definition 3.2 (2-Dismantlable Graph). *A graph G with 7 or more vertices is* 2-dismantlable *if it has a dismantlable ordering of vertices $\{v_1, v_2, \ldots, v_n\}$ such that two vertices v_{2i-1} and v_{2i} are dominated by vertices v_j and v_k respectively, $2i < j \leq k \leq n$ in the graph G_{2i-1} induced by $\{v_{2i-1}, \ldots, v_n\}$. A graph with fewer than 7 vertices is* 2-dismantlable *if it is cop-win.*

We need a few more definitions. Let P be a simple polygon, with an edge uv where v is a reflex vertex. Extend the directed ray from u through v and let t be the first boundary point of P beyond v that the ray hits. The points v and t divide the boundary of P into two paths. Let σ be the path that does not contain u. The simple polygon formed by σ plus the edge vt is called a *pocket* and denoted Pocket(u, v). The segment vt is the *mouth* of the pocket. Note that u does not see any points inside Pocket(u, v) except points on the line that contains the mouth. See Figure 3.1 for examples, including some with collinear vertices, which will arise in our proof. Pocket(u, v) is *maximal* if no other pocket properly contains it. Note that a non-convex polygon has at least one pocket, and therefore at least one maximal pocket. This will be strengthened to two maximal pockets in Lemma 3.4 below.

To prove that the visibility graph of a polygon is 2-dismantlable we prove that a maximal pocket in the polygon provides a visibility-increasing edge and that every nonconvex polygon has at least two maximal pockets. Aichholzer et al. [1, Lemma 2] essentially proved

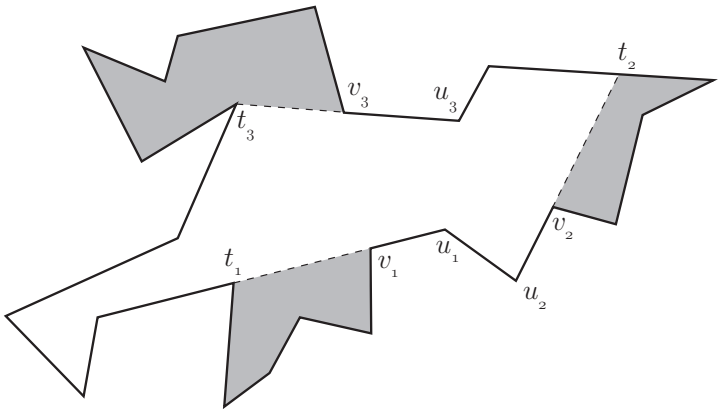


Figure 3.1: Pocket$(u_i, v_i), i = 1, 2, 3$, shaded.

this although it was not expressed in terms of maximal pockets. Also they assumed that the polygon has no three collinear vertices.

Lemma 3.3. *If uv is an edge of a polygon and Pocket(u, v) is maximal then uv is a visibility-increasing edge.*

Proof. We prove the contrapositive. Suppose that edge uv is not visibility-increasing. If u is a reflex vertex whose next neighbour on the polygon boundary is vertex w, say, then Pocket(w, u) properly contains Pocket(u, v), which implies that Pocket(u, v) is not maximal. Thus we may assume that u is convex. Flat angles (180 degrees) are considered as convex angles of the polygon. Since uv is not visibility-increasing there are two points p_1 and p_2 in order along uv such that the visibility polygon of p_1 is not contained in the visibility polygon of p_2. Thus there is a point t which is visible to p_1 but not visible to p_2. See Figure 3.2(a). By extending the segment p_1t, we may assume, without loss of generality, that t is on the polygon boundary. We claim that t lies in the closed half-plane bounded by the line through uv and lying on the opposite side of Pocket(u, v). This is obvious if p_1 is internal to edge uv, and if $p_1 = u$ it follows because u is convex. Furthermore, t cannot lie on the line through u, v otherwise p_2 would see t.

Now move point p from p_1 to p_2 stopping at the last point where p sees t. See Figure 3.2(b). There must be a reflex vertex v' on the segment tp. The points v' and t divide the polygon boundary into two paths. Take the path that does not contain v, and let u' be the first neighbour of v' along this path. It may happen that $u' = t$. Then, as shown in Figure 3.2(b), Pocket(u', v') properly contains Pocket(u, v), so Pocket(u, v) is not maximal.

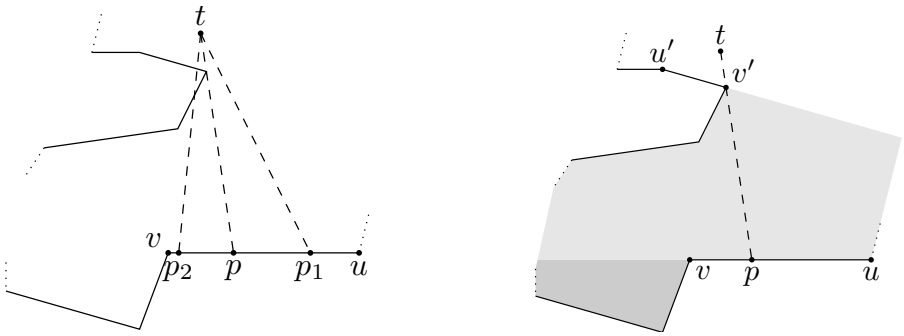


Figure 3.2: If uv is not visibility-increasing then Pocket(u, v) is not maximal.

□

Lemma 3.4. *Any polygon that is not convex has two maximal pockets Pocket(u_1, v_1) and Pocket(u_2, v_2) where u_1 does not see u_2.*

Proof. Let Pocket(u_1, v_1) be a maximal pocket. Let u be the other neighbour of v_1 on the polygon boundary. Consider Pocket(u, v_1), which must be contained in some maximal pocket, Pocket(u_2, v_2). Vertex u_1 is inside Pocket(u, v_1) and not on the line of its mouth. Therefore u_1 is inside Pocket(u_2, v_2) and not on the line of its mouth. Since u_2 cannot see points inside Pocket(u_2, v_2) except on the line of its mouth, it therefore follows that u_2 cannot see u_1. □

From the above lemmas, together with the observation that the visibility graph of a convex polygon is a complete graph, which is 2-dismantlable, we obtain the result that visibility graphs are 2-dismantlable.

Theorem 3.5. *The visibility graph of a polygon is 2-dismantlable.*

Proof. According to Lemma 3.4 any polygon has at least two distinct maximal pockets Pocket(u_1, v_1) and Pocket(u_2, v_2), and by Lemma 3.3 each of these maximal pockets corresponds to a visibility increasing edge, u_1v_1 and u_2v_2. Therefore u_1 and u_2 are candidates to be the first vertices of a 2-dismabtlable ordering. Also, as discussed in the proof of Lemma 3.1, removing the ears corresponding to vertices u_1 and u_2 of the polygon P is equivalent to removing vertices u_1 and u_2 from the visibility graph G of the polygon. By induction, the visibility graph $G - u_1 - u_2$ is 2-dismantlable. □

Consequently, the cop wins the cops and robbers game on the visibility graph of an n-vertex polygon in at most $\frac{n}{2}$ steps. There is a lower bound of $n/4$ cop moves in the worst case, as shown by the skinny zig-zag polygon illustrated in Figure 3.3. The robber can survive at least $n/4$ steps by choosing the end vertex that is farthest from the cop's initial position and staying still. In Section 3.2.1 we will prove an $\Omega(n)$ lower bound on the number of cop moves even when the cop can move on points interior to the polygon, and even when the polygon has link diameter 3, i.e. any two points in the polygon are joined by a path of at most 3 segments.

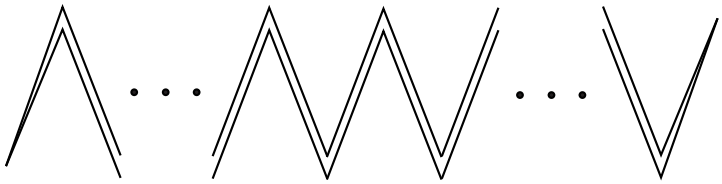


Figure 3.3: An n-vertex polygon in which the robber can survive for $n/4$ steps.

3.2 Cops and Robbers Inside a Polygon

In this section we look at the cops and robbers game on all points inside a polygon. This is a cops and robbers game on an infinite graph so induction on dismantlable orderings does not immediately apply. Instead we give a direct geometric proof that the cop always wins. Although the next chapter proves more generally that the cop always wins in any simply connected planar region with a reasonable boundary, it is worth first seeing the simpler proof for the polygonal case, both to gain understanding and because this case has a tight $\Theta(n)$ bound on the maximum number of moves, where n is the number of vertices of the polygon (discussed in Section 3.2.1).

Theorem 3.6. *The cop wins the cops and robbers game on the points inside any polygon in at most n steps using the strategy of always taking the first segment of the shortest path from its current position to the robber.*

Proof. We argue that each move of the cop restricts the robber to an ever shrinking *active region* of the polygon. Suppose the cop is initially at c_0 and the robber initially at r_0. In the i^{th} move the cop moves to c_i and then the robber moves to r_i.

Observe that for $i \geq 1$ points c_i are at reflex vertices of the polygon. To define the active region P_i containing the robber position r_i, we first define its boundary, a directed

line segment, $\bar{\ell}_i$. Suppose that the shortest path from c_{i-1} to r_{i-1} turns left at c_i, as in Figure 3.4.

Define $\bar{\ell}_i$ to be the directed segment that starts at c_i and goes through c_{i-1} and stops where the segment exits the polygon or becomes tangent to the polygon on the left side of the directed segment. (If the shortest path turns right at c_i we symmetrically define $\bar{\ell}_i$ to stop where the directed segment exits the polygon or becomes tangent to the polygon on the right side of the directed segment.)

In general, the segment $\bar{\ell}_i$ cuts the polygon into two (or more) pieces; let the *active region* P_i be the piece that contains r_{i-1}. (In the very first step, $\bar{\ell}_1$ may hug the polygon boundary, so P_1 may be all of P.)

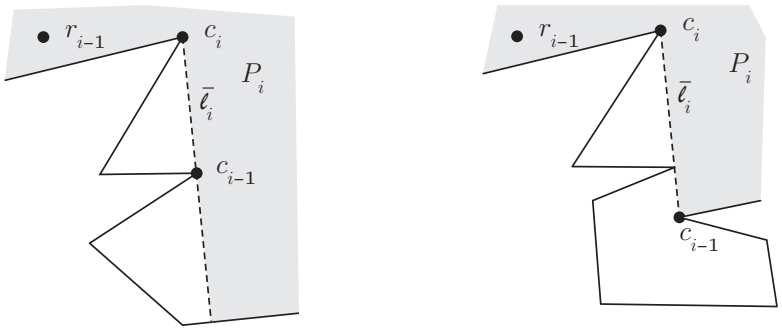


Figure 3.4: The segment $\bar{\ell}_i$ and the active region P_i (shaded) containing robber positions r_k, for all $k \geq i-1$.

We claim by induction on the (decreasing) number of vertices of P_i that the robber can never leave P_i, i.e., that r_i, r_{i+1},... are in P_i. It suffices to show that r_i is in P_i and that $P_{i+1} \subseteq P_i$ and that P_{i+1} has fewer vertices.

Suppose that the shortest path from c_{i-1} to r_{i-1} turns left at c_i. (The other case is completely symmetric.) Observe that the next robber position r_i must be inside P_i, i.e., the robber cannot move from r_{i-1} to cross $\bar{\ell}_i$. We distinguish two cases depending on whether the shortest path from c_i to r_i makes a left or a right turn at c_{i+1}.

Case 1. See Figure 3.5. The shortest path from c_i to r_i makes a left turn at c_{i+1}. We consider two subcases: (a) c_{i+1} is left of the ray $c_{i-1}c_i$; and (b) c_{i+1} is right of (or on) the ray $c_{i-1}c_i$. We claim that case (b) cannot happen because the robber could not have moved from r_{i-1} to r_i—see Figure 3.5(b). For case (a) observe that $\bar{\ell}_{i+1}$ extends past c_i and therefore P_{i+1} is a subset of P_i and smaller by at least one vertex—see Figure 3.5(a).

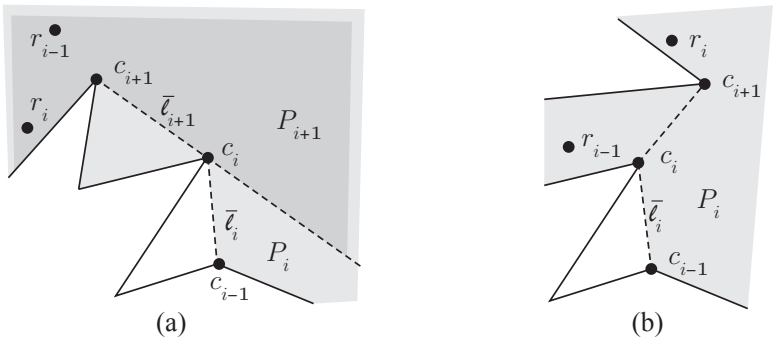


Figure 3.5: Case 1. (a) If c_{i+1} is left of the ray $c_{i-1}c_i$ then P_{i+1} (darkly shaded) is a subset of P_i (lightly shaded). (b) It cannot happen that c_{i+1} is to the right of the ray $c_{i-1}c_i$ because the robber could not have moved from r_{i-1} to r_i.

Case 2. See Figure 3.6. The shortest path from c_i to r_i makes a right turn at c_{i+1}. We consider two subcases: (a) c_{i+1} is left of the ray $c_{i-1}c_i$; and (b) c_{i+1} is right of (or on) the ray $c_{i-1}c_i$. See Figure 3.6. In case (a) $\bar{\ell}_{i+1}$ stops at c_i and in case (b) it may happen that $\bar{\ell}_{i+1}$ extends past c_i, but in either case, segment $\bar{\ell}_i$ is outside P_{i+1}, and P_{i+1} is a subset of P_i and smaller by at least one vertex. □

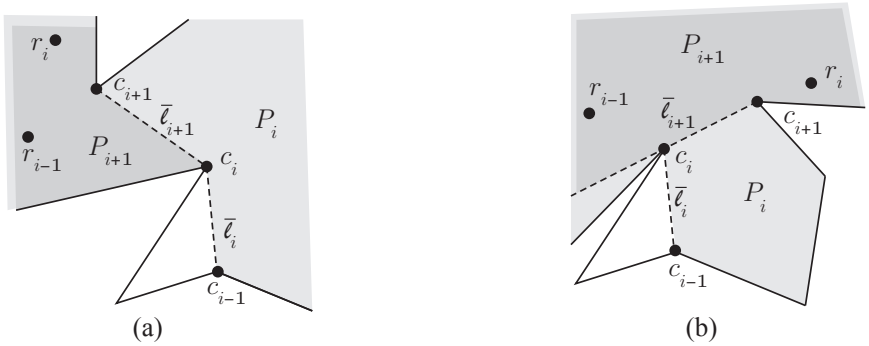


Figure 3.6: Case 2. (a) c_{i+1} is left of the ray $c_{i-1}c_i$. (b) c_{i+1} is right of the ray $c_{i-1}c_i$. In either case P_{i+1} (darkly shaded) is a subset of P_i (lightly shaded).

We note that Bhadauria et al. [12] use the same cop strategy of following a shortest path to the robber for the version of the problem where each cop or robber move is at most distance 1.

If the robber remained stationary, the cop should follow a minimum link path to the robber. What if we use such a strategy instead of our shortest path strategy? The cop would move in the same direction as in our strategy, but would move further into the interior of the polygon, rather than stopping at a reflex vertex. In fact, we will use such a strategy in the case of a region with a curved boundary, but even for a polygon, we must be careful about the how far the cop moves into the interior. Our goal in this section was to give a simple strategy that has a simple proof, restricts the cop to the polygon vertices, and yields an asymptotically optimal number of moves.

Theorem 3.6 can alternatively be proved by decomposing the polygon into $O(n^2)$ triangular regions and proving that they have an ordering with properties like a dismantlable ordering, but we do not pursue that approach (it is not any easier).

3.2.1 Lower Bounds

In this subsection we discuss lower bounds on the worst case number of cop moves. The example in Figure 3.3 shows that the cop may need $\Omega(n)$ moves even when it may move on interior points of the polygon. We give an example to show that this lower bound holds even when the polygon has small link diameter. The link distance between two points inside the polygon is the minimum number of bends of a path between them through the inside area of the polygon. The *link diameter* of a polygon is the maximum link distance between two points inside the polygon. If the link distance between the cop's position and the robber's position is d, it takes d steps for the cop to catch the robber if the robber does not move. Therefore, the half the link diameter is a lower bound on the length of the game.

Theorem 3.7. *There is an n-vertex polygon with link diameter 3 and a robber strategy that forces the cop to use $\Omega(n)$ steps.*

Proof. We modify the zig-zag polygon from Figure 3.3 to decrease the link diameter to 3 as shown in Figure 3.7. The polygon consists of $k = n/3$ similar sections concatenated, where n is the number of vertices of the polygon. We will show that the robber has a strategy to survive at least $k/2$ steps in such a polygon.

Robber Strategy: The robber plays on points x_i, $1 \leq i \leq k$. Initially the robber chooses the closest point x_i to $x_{\lfloor k/2 \rfloor}$ such that x_i is not visible to the cop's initial position. Observe that x_i is only visible to the gray area in Figure 3.7 and the line segments $x_{i-1}x_i$ and x_ix_{i+1}, so the cop can only see at most three of the x_i's from its initial position.

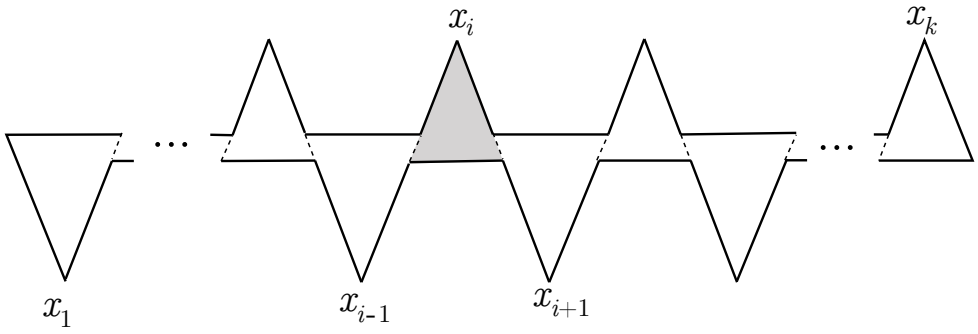


Figure 3.7: A polygon with constant link diameter in which the robber can survive for $\Omega(n)$ steps.

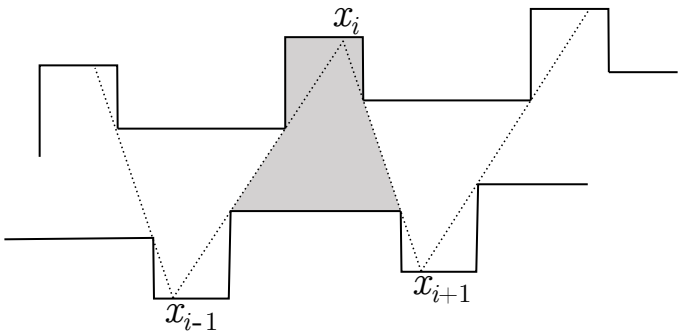


Figure 3.8: We can adjust the edges and the dashed-line path such that the polygon becomes non-degenerate.

The robber remains stationary until it is visible to the cop, i.e., when the cop enters the gray area or along the segments $x_{i-1}x_i$ and x_ix_{i+1}. Then, the robber moves to one of the neighbors x_{i-1} or x_{i+1}, the one that is not visible to the cop. At least one of x_{i-1} and x_{i+1} is safe for the robber to move to in the next step as we observe that the only point visible to all three of x_{i-1}, x_i, and x_{i+1} is x_i, and the cop may not be at point x_i otherwise the robber would have been visible to the cop in the previous step.

The robber can survive at least $k/2 - 1$ steps with this strategy as the game may only be terminated at either x_1 or x_k.

Note that the polygon in Figure 3.7 is degenerate, with edges that lie on the same line, but we can adjust it a little bit to resolve all degeneracies, as shown in Figure 3.8, at the expense of decreasing the lower bound to $n/8$. We need to be careful not to increase the link diameter of the polygon through the edge level changes. More precisely, assume

that in Figure 3.7 there are k horizontal edges at y-coordinate 0 and k horizontal edges at y-coordinate 1. The horizontal edges at $y = 0$ in Figure 3.7 are evenly spread from $y = 0$ to $y = \frac{k-1}{3k}$ in Figure 3.8, and the horizontal edges at $y = 1$ are evenly spread from $y = \frac{2k+1}{3k}$ to $y = 1$. This will allow a corridor of width $\frac{k}{3k}$ and the link diameter is still 3. Also, the triangular dents are replaced with rectilinear ones to remove the collinear diagonal edges. □

Chapter 4

Cops and Robbers Inside a Splinegon

In this chapter we consider the cops and robbers game in a simply connected region with curved boundary, specifically a *splinegon* R whose boundary consists of n smooth curve segments that each lie on their own convex hull. Other natural assumptions (such as algebraic curves or splines of bounded degree, or other curves of constant complexity) give regions that can be converted to splinegons with a constant factor overhead by cutting at points of inflection and points with vertical tangents. Assume that tangents in a given direction and common tangents between curve segments can be computed. A *vertex* is an endpoint between two curve segments.

We need another assumption to avoid an infinite game where the cop gets closer and closer to the robber but never reaches it. This occurs, for example, when two curves meet tangentially at a vertex as in Figure 4.1—in fact, in this situation a robber at a vertex avoids capture by remaining stationary. One possibility is to assume that the robber is captured when the cop gets sufficiently close (within some ϵ). Instead, we will make the assumption that the link distance between any two points in the splinegon R is finite, and bounded by d.

With these assumptions—a splinegon R of n curved segments with link diameter d—we prove that the cop always wins, and does so in $O(n^2 + d)$ steps. But first, we show through additional examples that the strategy must be a little more complex than in the polygonal case.

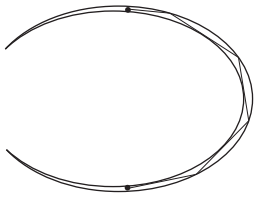


Figure 4.1: The number of cop moves may be infinite even when $n = 2$. Truncating the vertices makes the game finite but the number of moves may depend on the link diameter.

4.1 The Cop Strategy

A major difference from the polygonal case is that the cop may need to move to interior points in order to win. Figure 4.2, for example, shows a region in which the robber can win if the cop always stops on the boundary.

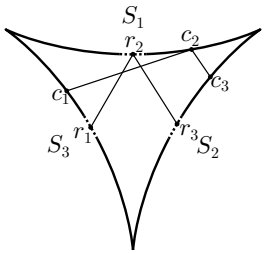


Figure 4.2: If the cop plays only on the boundary then the robber can win: the robber's strategy is to play on the middle dashed portions of the boundary and always move to the same curve S_i that the cop is on. In our winning cop strategy the cop would move to the endpoint tangents (drawn as thin lines).

Our strategy is that the cop starts off along the first straight segment of the Euclidean shortest path [28, Chapter 15] to the robber's current position. However, if this segment is tangent to a concave curve of the shortest path then the cop should move further, into the interior of the polygon. How far should the cop go? It is tempting to stop the cop when it can see the robber, but Figure 4.3 shows that this strategy fails—the cop should move farther. Figure 4.4 shows there is also a danger of moving the cop too far.

In our strategy the cop will stop at certain lines inside the splinegon. We first state the cop strategy in terms of these lines, and then define the lines. We use the notation c_{i-1} for the cop's position and r_{i-1} for the robber's position at the start of round i. Their initial positions are c_0 and r_0. Recall that each round begins with a cop move.

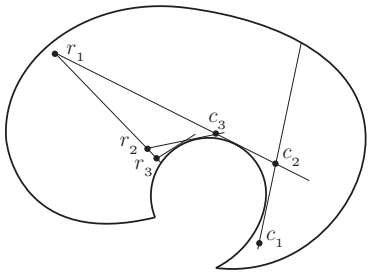


Figure 4.3: If the cop moves only far enough to see the robber then the robber can win because it can force the cop to take smaller and smaller steps.

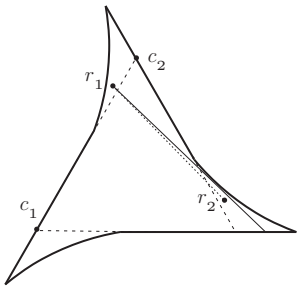


Figure 4.4: If the cop moves too far then the robber can win: the cop moves from c_1 to c_2, the robber moves from r_1 to r_2 (dotted line) and then this can be repeated around the polygon. In our proposed strategy the cop would not move from c_1 all the way to c_2—it would stop at a robber exit line (drawn as a thin line).

Cop Strategy for Round i. If the cop sees the robber, it moves to the robber's position and wins. Otherwise, define the cop's next position, c_i, as follows: Compute the shortest path from the cop's current position, c_{i-1}, to the robber's current position, r_{i-1}. Let ℓ_i be the ray along the first straight segment of this shortest path, or, if the shortest path begins with a curve, let ℓ_i be the tangent to this curve at the point c_{i-1}. Let b_i be the first point where the shortest path diverges from ℓ_i. Then b_i lies on the boundary of the splinegon R. If b_i is not a splinegon vertex, let γ_i be the boundary curve containing b_i. If b_i is a splinegon vertex then there are two boundary curves incident to b_i, and we let γ_i be the boundary curve segment containing b_i such that in a neighbourhood of b_i, the diverging shortest path lies in the portion of the region bounded by γ_i, and the part of ray ℓ_i past b_i.

By reflection if necessary, assume that the path starts upward and turns left, as depicted in Figure 4.5. If $b_i \neq c_{i-1}$ and b_i is a vertex, then define c_i to be b_i. (This matches the polygonal case.) Otherwise ℓ_i is tangent to γ_i, so define c_i to be the first point on the

ray ℓ_i, past b_i, where ℓ_i intersects a *common tangent* or a *robber exit line* or touches the splinegon boundary.

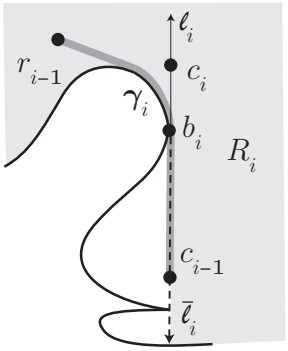


Figure 4.5: The cop move, showing the shortest path from c_{i-1} to r_{i-1} (thick grey path), the first straight segment of this path $c_{i-1}b_i$ upward along ray ℓ_i, the new cop position c_i, the downward segment $\bar{\ell}_i$ (dashed) and the active region R_i (lightly shaded).

We now define *common tangents* and *robber exit lines*. Refer to Figure 4.6. A *common tangent* is a line segment that is tangent to R at two points and extends in both directions until it exits R. At each endpoint of each curve we have an *endpoint tangent*—the tangent to the curve through the endpoint. An endpoint tangent extends in both directions until it exits R. We count endpoint tangents as common tangents. There are $O(n^2)$ common tangents, because a curve has at most four common tangents with any other curve or vertex.

We define *robber exit lines* relative to the current robber and cop positions, using the notation from the cop strategy above. See Figure 4.6. Consider segments that start at r_{i-1} and are tangent to R, ending at the tangent point. Among these, a *robber exit line* is a segment that crosses ray ℓ_i such that the tangent point is on the far side of the segment with respect to the direction of ℓ_i. To be precise, the infinite line containing the segment divides the plane into two parts, and the *far side* is the side not containing c_{i-1}, the source of ray ℓ_i.

If we extend a robber exit line past its tangent point to the region boundary we obtain a *bay* of points not visible from the robber position. Note that every bay (including the tangent point) contains a vertex of the region —either the tangent point itself is a vertex or the tangent point is on a reflex curve, and we must change curves before the end of the bay.

With these definitions of common tangents and robber exit lines, we have completely

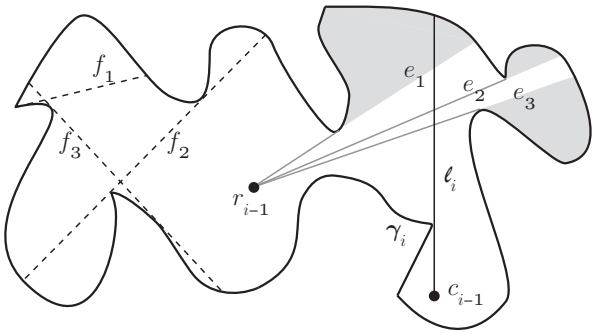


Figure 4.6: Lines f_1, f_2 and f_3 are three of the many common tangents. Segments e_1, e_2 and e_3 go through r_{i-1} and are tangent to R. Segment e_1 is not a robber exit line because it does not cross ℓ_i. Segment e_3 is not a robber exit line because its tangent point is not on the far side with respect to the direction of ray ℓ_i. Segment e_2 is a robber exit line. Lightly shaded regions are the bays.

specified the cop strategy. We note that the cop's move can be computed in polynomial time assuming we have constant time subroutines to compute common tangents and tangents at a given point. We can preprocess to find all common tangents. For a given robber position, we can find all robber exit lines in polynomial time. We can find shortest paths in the splinegon R in linear time using the algorithm of [66]. With this information, we can find the next cop position. A straightforward implementation takes $O(n^2)$ time per move, though this can probably be improved.

4.2 The Cop Wins

In order to prove that the cop wins using the strategy specified in the previous section, we first show that each cop move restricts the robber to a smaller subregion. Then we show that the number of steps the cop needs to win is $O(n^2 + d)$ where n is the number of segments and d is the link diameter of the region.

We begin by defining the subregion that the robber is restricted to during and after round i. Define $\bar{\ell}_i$ to be the directed segment that starts at b_i (the first point where the shortest path diverges from ℓ_i) and goes through c_{i-1} and stops where the segment exits the splinegon or becomes tangent to the splinegon on the side of the directed segment opposite to the tangency point with γ_i (e.g., where part of the boundary is to the left of

the downward directed segment $\bar{\ell}_i$ in Figure 4.5).

The segment $\bar{\ell}_i$ starts and ends on the boundary so it cuts the region into two (or more) pieces; define the *active region*, R_i, to be the piece that contains r_{i-1}. Define the *exclusion region* to be its complement in R. See Figure 4.5. Observe that any line segment inside the region R with one endpoint at r_{i-1} has its other endpoint inside R_i. In particular, this means that r_i is inside R_i, i.e. that the robber cannot exit R_i in round i.

We prove below in Lemma 4.3 that $R_{i+1} \subsetneq R_i$, i.e., the active region shrinks. Following that, we show that the cop wins in a finite number of steps. The proofs are similar to the analogous results for polygons, and involve handling four cases for the left/right configuration of the cop and the robber. Suppose that the shortest path from c_{i-1} to r_{i-1} makes a left turn at b_i. (The other case is completely symmetric.) We distinguish the following cases:

Case 1. The shortest path from c_i to r_i makes a left turn at b_{i+1}.

(a) c_{i+1} is left of the ray $c_{i-1}c_i$—more precisely, in moving from c_{i-1} to c_i to c_{i+1} the cop turns left by an angle in the range $(0, 180°)$. (Turning by $0°$ will be handled in case (b).)

(b) c_{i+1} is right of the ray $c_{i-1}c_i$—more precisely, the cop turns right at c_i by an angle in $[0, 180°)$.

Case 2. The shortest path from c_i to r_i makes a right turn at b_{i+1}.

(a) c_{i+1} is left of the ray $c_{i-1}c_i$—more precisely, the cop turns left at c_i by an angle in $(0, 180°)$.

(b) c_{i+1} is right of the ray $c_{i-1}c_i$—more precisely, the cop turns right at c_i by an angle in $[0, 180°)$.

Note that the cop never turns by an angle of $180°$ (doubling back) because then b_{i+1} would be on the line segment between c_i and b_i or further along, on the ray $\bar{\ell}_i$. In the first case, b_{i+1} would provide a stopping point for c_i according to the rule that the cop stops on the boundary. The second case is impossible because the robber can never move from r_{i-1} to a position that would cause the cop to move onto $\bar{\ell}_i$.

We begin by showing that Case 2(a) can happen only in special circumstances and that Case 1(b) cannot happen at all.

Lemma 4.1. *In Case 2(a) the segment c_ib_{i+1} is tangent to the boundary on its left side (as well as tangent to the boundary on its right side at b_{i+1}).*

Proof. See Figure 4.7. The segment c_ib_{i+1} is tangent to the boundary curve γ_{i+1} on its right side at point b_{i+1}. If $c_i = b_i$, then the segment c_ib_{i+1} touches the boundary at its right side at b_i. Otherwise, c_i is past b_i. Suppose that segment c_ib_{i+1} is not tangent to the boundary on its left side. We show that the cop has passed a common tangent, which is a contradiction. Move c_i back towards c_{i-1} while maintaining tangency with the curve γ_{i+1} We can move some positive amount. Either we reach the tangent at an endpoint of γ_{i+1} or the segment c_ib_{i+1} hits a boundary point on its left side (possibly because c_i reaches b_i). In either case, we have arrived at a common tangent, so c_i should have been placed here rather than further along. Note that $c - i$ may not reach c_{i-1}, otherwise, $b_i = c_{i-1}$ and the tangent at b_i will be a vertical line. □

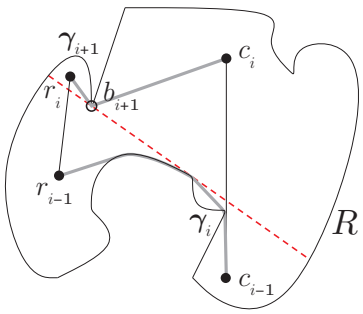


Figure 4.7: In case 2(a) if the segment c_ib_{i+1} is not tangent to the boundary on its left side then there is an earlier choice for c_i (on the dashed red common tangent).

Lemma 4.2. *Case 1(b) cannot occur, i.e., it cannot happen that the shortest path from c_i to r_i makes a left turn at b_{i+1} and c_{i+1} is right of the ray $c_{i-1}c_i$ by an angle in $[0, 180°)$.*

Proof. Suppose the situation does occur. See Figure 4.8(a). We show that the cop has passed a common tangent or a robber exit line, which gives a contradiction. Because c_{i+1} is to the right of the ray $c_{i-1}c_i$, the robber's move $r_{i-1}r_i$ must have crossed the line through $c_{i-1}c_i$, say at point x. We claim that segments $r_{i-1}r_i$ and $c_{i-1}c_i$ intersect. First note that x lies after b_i along the ray $c_{i-1}c_i$. We must show that c_i lies after x along this ray. If c_i lies before x, then there is a two-link path inside the region, c_i, x, r_i that turns right at x. Shortening this to a locally shortest path, we obtain the shortest path from c_i to r_i that makes a first turn to its right, contradicting our assumption.

Define σ to be the shortest path from r_{i-1} to c_i. The segment c_ib_{i+1} is tangent to the curve γ_{i+1}. We will now move point p from c_i towards b_i, maintaining a segment τ through p tangent to the curve γ_{i+1}. In the other direction, τ extends to its intersection point with

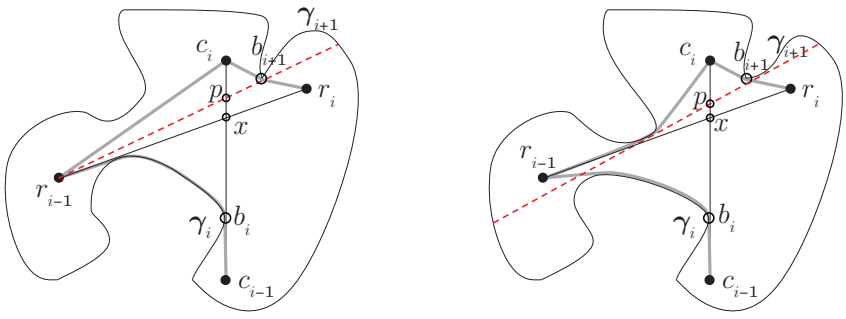


Figure 4.8: Case 1(b) cannot occur in these situations: moving p from c_i towards b_i while maintaining tangency of τ (dashed red) with γ_{i+1} we encounter: (a) a robber exit line or (b) a common tangent.

σ. See Figures 4.8 and 4.9, where τ is drawn as a dashed red line. If τ reaches an endpoint tangent of γ_{i+1} then we have a common tangent and the cop should have stopped at point p. Otherwise, the segment τ must at some point lose contact with σ and we claim that this can happen only because of one of the following:

- The segment τ intersects σ at r_{i-1}; this is a robber exit line. See Figure 4.8(a).
- The segment τ becomes tangent to σ; this is a common tangent. See Figure 4.8(b).
- The segment bumps into the region boundary (possibly at point b_i); this is a common tangent. See Figure 4.9.

In all cases, the cop should have stopped at point p because of the common tangent or robber exit line τ. □

We are now ready to show that the active region shrinks.

Lemma 4.3. *The active regions satisfy* $R_{i+1} \subsetneq R_i$.

Proof. Assume that the shortest path from c_{i-1} to r_{i-1} makes a left turn at b_i, and consider the cases as listed above.

Case 1(a). The shortest path from c_i to r_i makes a left turn at b_{i+1} and c_{i+1} is left of the ray $c_{i-1}c_i$. See Figure 4.10. The ray $\bar{\ell}_{i+1}$ from b_{i+1} through c_i intersects ℓ_i at c_i, and

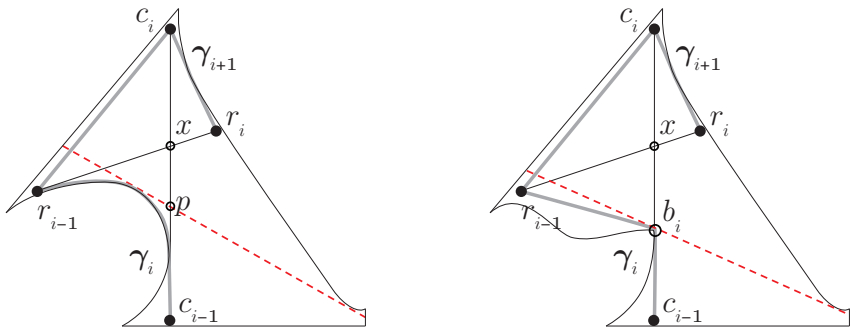


Figure 4.9: Case 1(b) cannot occur in these situations: moving p from c_i towards b_i while maintaining tangency of τ (dashed red) with γ_{i+1} we encounter a common tangent by bumping into the region boundary (a) before p reaches b_i, or (b) at b_i.

is therefore completely contained in the active region R_i. Furthermore, the open segment $c_{i-1}c_i$ is outside R_{i+1} but inside R_i. Thus $R_{i+1} \subsetneq R_i$.

Case 1(b). The shortest path from c_i to r_i makes a left turn at b_{i+1} and c_{i+1} is right of the ray $c_{i-1}c_i$. This case cannot occur by Lemma 4.2.

Case 2(a). The shortest path from c_i to r_i makes a right turn at b_{i+1} and c_{i+1} is left of the ray $c_{i-1}c_i$. See Figure 4.11(a). By Lemma 4.1, the segment c_ib_{i+1} is tangent to the boundary on its left side, say at point p. The ray $\bar{\ell}_{i+1}$ that defines the active region extends from b_{i+1} to p. Its extension goes through c_i, so it is contained in R_i. Furthermore, the open segment $c_{i-1}c_i$ is outside R_{i+1} but inside R_i. Therefore $R_{i+1} \subsetneq R_i$.

Case 2(b). The shortest path from c_i to r_i makes a right turn at b_{i+1} and c_{i+1} is right of the ray $c_{i-1}c_i$ (or on the ray). See Figure 4.11(b). The ray $\bar{\ell}_{i+1}$ intersects ℓ_i at c_i, and is contained in R_i. Furthermore, the open segment $c_{i-1}c_i$ is outside R_{i+1} but inside R_i. Thus $R_{i+1} \subsetneq R_i$. □

Finally we prove our main result.

Theorem 4.4. *Inside a splinegon of n curve segments with link diameter d, the cop wins the cops and robbers game in $O(n^2 + d)$ moves.*

Proof. We argue that at each step between the first and the last, the active region shrinks in some discrete way. Define the *newly excluded region*, E_i, to be $R_i - R_{i+1}$. In order to take care of collinearities, we will include the boundary of R_{i+1} but exclude the boundary

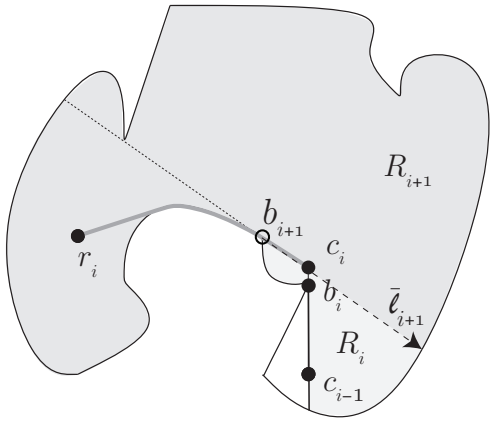


Figure 4.10: Case 1(a).

of R_i. If the two boundaries intersect, the intersection point is not included in E_i. In Figures 4.10, 4.11(a), and 4.11(b) the region E_i is lightly shaded. By Lemma 4.3, the E_i's are disjoint.

Let σ be a minimum link path from the initial to the final cop position. Then σ has at most d bends. Note that there are at most $O(n^2)$ common tangents because there are at most n^2 pairs of curves, and each pair has at most four common tangents. Our bounds derive from these two facts.

Our plan is to show that at each step we make progress in one of the following ways:

1. c_i is a vertex
2. c_i and b_i are the tangent points of a common tangent
3. E_i contains a vertex of the region
4. E_i contains an endpoint of a common tangent with both endpoints in R_i
5. E_i contains a bend of σ.

We begin by bounding the number of events of each of the above types. After that we will show that one of these events occurs in each step.

According to Lemma 4.3 c_i and b_i never repeat, so event (1) happens at most n times and event (2) happens at most n^2 times. Because the E_i's are disjoint, event (3) happens

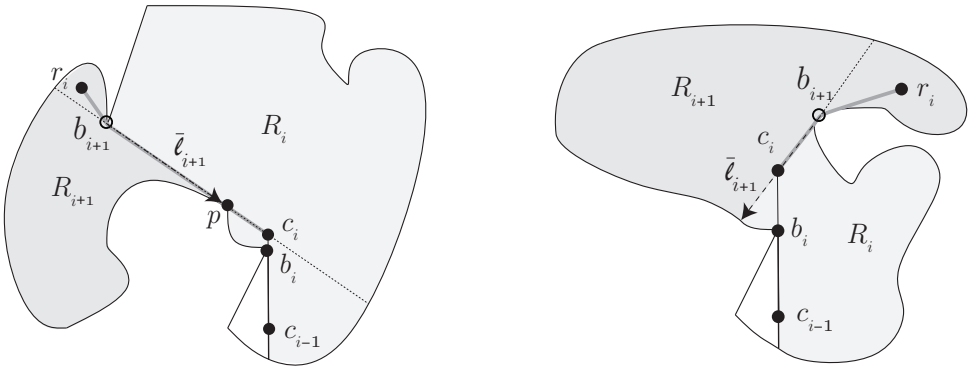


Figure 4.11: (a) Case 2(a). (b) Case 2(b).

at most n times, event (4) happens at most n^2 times, and event (5) happens at most d times.

It remains to prove that at each step, one of the above 5 events occurs.

Recall the conditions for defining the cop's position c_i. If $c_i = b_i$ because b_i is a vertex then we have event (1). Otherwise c_i stops at a common tangent or a robber exit line or on the boundary of the region.

Suppose first that c_i stops on a common tangent that crosses ℓ_i. (If c_i is on a common tangent that does not cross ℓ_i then c_i is one of the tangent points, and hence on the boundary, and that case is dealt with below.) Since the common tangent crosses ℓ_i, both tangent points must lie in R_i. The boundary of R_{i+1} is a line segment that goes through c_i and therefore one endpoint of the common tangent must lie inside R_{i+1}, possibly on its boundary. This endpoint is in E_i. Thus we have event (4).

We next consider the case where c_i stops on a robber exit line of r_{i-1}. Let the robber exit line be the segment $r_{i-1}t$. We will prove that E_i contains a vertex, i.e., event (3). We will do this by proving that the tangent point, t, of the robber exit line, and its bay, must lie in E_i. Then, as noted when we defined robber exit lines, this bay contains a vertex, so E_i contains a vertex. It remains to prove that t lies in E_i. As observed when the active region R_i was defined, any line segment inside the region R with one endpoint at r_{i-1} has its other endpoint inside R_i. Applying this to the segment $r_{i-1}t$ shows that t lies in R_i. It remains to show that t is outside R_{i+1}. Applying the observation that line segments incident to r_i remain inside R_{i+1} to the line segment $r_i r_{i-1}$ shows that r_{i-1} is inside R_{i+1}. Finally, there are two lines that cross at point c_i: the robber exit line $r_{i-1}t$, and the ray ℓ_{i+1}

which marks the boundary of R_{i+1}. Since r_{i-1} is inside R_{i+1} the other end of the segment, t, must be outside R_{i+1}.

Finally, we must consider the possibility that c_i stops on the region boundary, i.e. c_i is at a point where ℓ_i exits the whole region, or where ℓ_i is tangent to the boundary of the region. When can this happen? By Lemma 4.2, Case 1(b) never occurs. By Lemma 4.1, the cop always stops at a common tangent in Case 2(a), and this tangent crosses ℓ_i, so this is a case we already dealt with.

Thus we must be in Case 1(a) or 2(b). If c_i is not at a point where ℓ_i exits the region then ℓ_i is tangent to the boundary at c_i. Since ℓ_i is also tangent to the boundary at b_i, thus c_i and b_i are the tangent points of a common tangent, i.e. event (2). We are left with the case where c_i is at a point where ℓ_i exits the region. We first claim that this cannot happen in Case 2(b), because in this case (see Figure 4.11(b)) the robber's move $r_{i-1}r_i$ must cross line ℓ_i beyond c_i, which is impossible if ℓ_i exits the region at c_i. Thus we must be in Case 1(a). See Figure 4.12. The minimum link path σ from the initial cop position (outside R_i, or possibly on the boundary of R_i) to the final robber position (inside R_{i+1}) must include a bend point in E_i. This is event (5). □

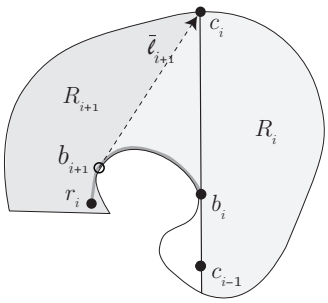


Figure 4.12: When the cop stops on the region boundary in Case 1(a), the minimum link path from the initial cop position (in the white region) to the final robber position (in the darkly shaded region) must include a bend point in E_i (the lightly shaded region).

Therefore, the cops and robbers game in its most general settings, i.e., when the cop and the robber are playing inside or on the boundary of a splinegon, is also cop-win and the suggested strategy guarantees that the cop will catch the robber after $O(n^2 + d)$ steps.

Chapter 5

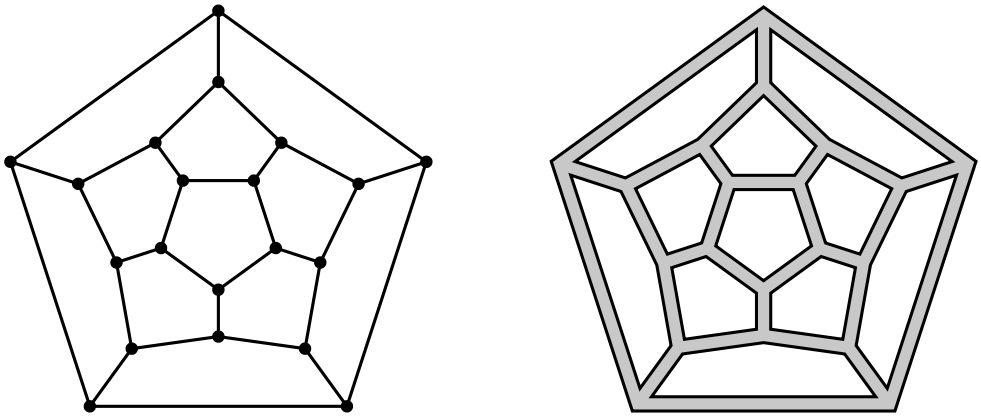
Pursuit Evasion: Conclusions and Open Problems

We explored the straight-line pursuit evasion problem and the challenges to catch an evader in three different settings: on the visibility graph of a simple polygon, on all the infinitly many points inside a simple polygon, and on all the infinitely many points inside a spline-gon. All three settings are shown to be cop-win, and all games are played in simply connected regions in 2D.

Therefore, a natural next step is to consider the problem in regions with holes. A lower bound of three cops may be achieved using the cop-number of planar graphs. Aigner and Fromme showed that the cop number of planar graphs is three and there are examples where three cops are required to catch a single robber [2]. Using such examples, we may construct examples of the straight-line pursuit evasion problem on all points inside a polygon with holes. This is by taking a straight-line planar drawing of the graph and cutting out polygonal holes to leave narrow corridors for the graph edges. Figure 5.1 shows a 3-cop win planar graph and the instance of the straight-line cops and robbers game constructed from it. It is an open question whether three cops suffice for any polygon with holes.

Some other open problems and discussion about the straight-line pursuit evasion problem are as follows:

1. What is the complexity of finding how many moves the cop needs for a given polygon/region? The graph version of this problem is solvable in polynomial time for cop-win graphs [49]. For the cops and robbers game on the points inside a polygon we conjecture that the problem is solvable in polynomial time if the cop is restricted



(a) (b)

Figure 5.1: (a) A 3-cop win planar graph and (b) the cops and robbers game inside a polygonal region with holes constructed from the graph; the game inside this polygonal region needs at least three cops.

to the reflex vertices of the polygon. However, the cop may save by moving to an interior point—for example in a star-shaped polygon whose kernel is disjoint from the polygon boundary—so the problem seems harder if the cop is unrestricted.

2. Is there a lower bound of $\Omega(n^2 + d)$ on the worst case number of cop moves in a splinegon of n curve segments and link diameter d? From results in Section 3.2.1 we have a lower bound of $\Omega(n + d)$.

3. What if the evader can move distance 2 (or even 3) in the link metric? Does one cop suffice? In the graph setting this problem has been considered by Frieze et al.[40]. They show that the cop number increases in general as the robber's speed increases.

Part II

Straight Line Morphing

Chapter 6

Straight-Line Morphing: Introduction and Background

The transformation of shapes or drawings under different assumptions and constraints is widely studied in computational geometry. "Morphing" has been the standard term used since the 1980's to address the continuous transformation between shapes while some properties are maintained during this transformation. All 2D reconfiguration problems, i.e., to change an object or shape from one configuration to another configuration, can be defined as "morphing". Research has been done to look at different constraints or properties to be maintained according to different applications and problems. These problems may vary from combinatorial problems such as linkage folding reconfigurations, morphing between two different triangulations, and morphing of graph drawings, to transforming between different shapes in computer graphics and animation applications. Some of these problems are discussed in Section 6.3.

Preserving planarity is a common assumption in different morphing problems of graph drawings. It is also visually desirable that vertices move on "nice" paths during the morph to resemble a more natural transformation. "Nice" paths are generally defined as paths that are not too long and do not have many bends.

In this part of the thesis we define a type of morphing in which the vertex trajectories are simple, and yet the morphing method is powerful enough to preserve planarity.

6.1 Linear Morphing

The simplest way to morph with the optimal vertex trajectories would be to move each vertex on the straight line connecting its initial position to the final position. This gives the shortest trajectory for all vertices with no bends. When each vertex also moves with a uniform speed from the beginning to the end of the morph, it is called "linear morphing."

Definition 6.1 (Linear Morphing). *Let $G = (V, E)$ be a planar graph on n vertices, with two planar straight-line drawings given by initial vertex positions $P_0 = \langle p_1^0, p_2^0, \ldots, p_n^0 \rangle$ and final vertex positions $P_1 = \langle p_1^1, p_2^1, \ldots, p_n^1 \rangle$. Without loss of generality, we assume that the morph happens in time interval $[0, 1]$. The drawing of G at time $t \in [0, 1]$ is captured by the vertex positions P_t.*

In linear morphing, the position of each vertex $v_i \in G$ at time t, denoted by p_i^t, can be viewed as a function defined by:

$$p_i^t = (1 - t)p_i^0 + tp_i^1, \tag{6.1}$$

As we can see, all vertices have deterministic position functions in linear morphing. This makes it possible to verify whether a linear morph preserves planarity throughout the morph. Figure 6.1 shows linear morph of a path in three different times.

Essentially, if planarity is violated at some point during the linear morph, there is a time t in which two edges start to intersect. Time t can be computed in polynomial time by solving quadratic equations of segment movements in a linear morph.

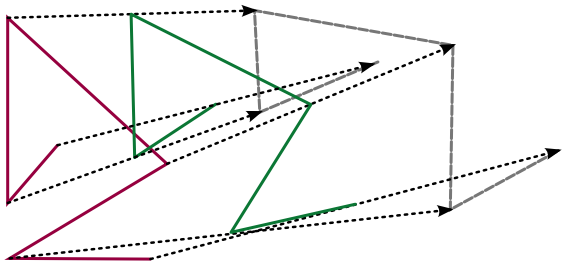


Figure 6.1: Linear morphing of a path from the initial configuration (solid red lines) to the final configuration (gray dashed lines). The arrow lines show the trajectories of each vertex. The path is in initial configuration at time $t = 0$, in the final configuration at time $t = 1$, and the solid (dark green) lines show the path configuration at time $t = 0.5$. All vertices traveled half of the way at $t = 0.5$.

Lemma 6.2. *There is an $O(|V| \cdot |E|)$ time algorithm to check whether a linear morph of a planar straight-line graph maintains planarity.*

Proof. The vertex positions at time t are determined by equation 6.1 which is a linear function of t. Any edge $e(v_i, v_j)$ of G at time t is a line segment connecting p_i^t and p_j^t. To check planarity, we only need to test if all edges are disjoint for all $t \in [0, 1]$. In other words, is there a time t, $0 \leq t \leq 1$ such that two segments (p_i^t, p_j^t) and $(p_{i'}^t, p_{j'}^t)$, corresponding to edges $e(v_i, v_j)$ and $e(v_{i'}, v_{j'})$ in G, intersect? The parametric equation of the line passing through a pair of points p_i^t and p_j^t has the form:

$$q_k^t = kp_i^t + (1 - k)p_j^t, k \in [0, 1],$$

which is a quadratic formula in terms of t and k. To check if two segments intersect during the morph we should check if the line segment formula for two segments has a common point, i.e., to solve the equation $q_k^t = q_{k'}^t$ where, $q_{k'}^t$ is the equation of line segment of edge $e(v_{i'}, v_{j'})$. This equation is quadratic with three variables k,k',and t that should be in the range of $[0, 1]$.

Since the vertices move continuously during the morph, if two segments intersect in some time t, there should be a time $t_0 \leq t$ when one endpoint of one of the two segments lies on the other segment. That is because the two segments were disjoint initially at time $t = 0$ and we only need to find when they start to intersect.

With this observation, the number of variables in the equation will be reduced to two, because we only need to check if there is some point $p_{i'}^t$ that lies on line segment (p_i^t, p_j^t). We must solve the equation $q_k^t = p_{i'}^t$ for any vertex $v_{i'}$ and any edge $e(v_i, v_j)$, that forms the quadratic formula for q_k^t. The quadratic equation can be solved in constant time and we have $|E| \times |V|$ of these equations to solve to ensure that none of them have a solution. Note that to ensure that a vertex/edge coincidence will lead to an intersection we also need to check if the vertex $v_{i'}$ is on opposite sides of edge $e(v_i, v_j)$ just before and after time t. This test can be done in constant time. Therefore, we can check in time $O(|E||V|)$ if the given linear morph preserves planarity. □

Figure 6.2 shows an example where a linear morph does not preserve planarity. At time $t = 0.5$ two edges intersect. The intersection first occurs shortly before $t = 0.25$.

Linear morphing is a very restrictive category of morphs and rarely maintains planarity, however it is widely used as a basic step in different morphing algorithms. Next, we introduce a new class of morphs which is much more powerful than the class of linear morphs, but the vertex trajectories are still simple.

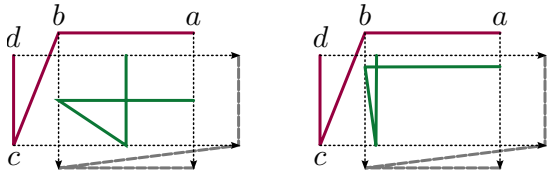


Figure 6.2: Linear morph of a path that does not preserve planarity: (a) $t = 0.5$: the path (dark green lines) is self crossing and violates the planarity condition; (b) $t = 0.25$: the moments after the crossing starts to happen.

6.2 Straight-line Morphing

Linear morphing is restrictive in two ways: straight-line vertex trajectories and uniform speed. In this thesis we focus on "straight-line morphing" or simply "SL-morphing", as a more powerful class of morphs, by relaxing the uniform speed condition in linear morphing:

An SL-morph is a continuous transformation of a planar graph drawing from an initial straight-line drawing to a final straight-line drawing in which each vertex moves on the straight-line segment connecting its initial position to its final position. A planar SL-morph or PSL-morph is an SL-morph that preserves planarity. The detailed definitions of SL-morphing and Planar SL-morphing will come later.

Compared to linear morphing, there is much more freedom on vertex movements in SL-morphing; few or many vertices may move at once, they may move at different speeds, or may pause while other vertices move, and so on. Figure 6.3 shows the same example shown in Figure 6.2 with a planar SL-morph. As shown in Figure 6.3, if vertices a and b morph first to the final configuration, and then the two vertices c and d morph, the planarity is preserved.

Definition 6.3 (SL-morphing). *Let $G = (V, E)$ be a planar graph on n vertices, with two planar straight-line drawings given by initial vertex positions $P_0 = \langle p_1^0, p_2^0, \ldots, p_n^0 \rangle$ and final vertex positions $P_1 = \langle p_1^1, p_2^1, \ldots, p_n^1 \rangle$. Without loss of generality, we assume that the morph happens in time interval $[0, 1]$. The drawing of G at time $t \in [0, 1]$ is captured by the vertex positions P_t.*

In an SL-morph*, the position of each vertex $v_i \in G$ at time t, denoted by p_i^t, can be viewed as the following function:*

$$p_i^t = (1 - d_i(t))p_i^0 + d_i(t)p_i^1,$$

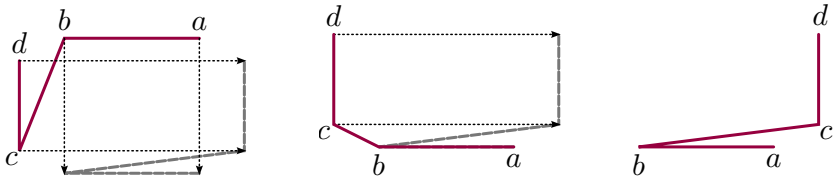


Figure 6.3: The path from the initial configuration may morph to the final configuration through the following steps: (a) The graph in the initial configuration; (b) The intermediate state after a and b have morphed; (c) The graph in the final configuration

where d_i is a continuous non-decreasing parameterization of time, $d_i : [0,1] \to [0,1]$ with fixed points of 0 and 1, i.e., $d_i(0) = 0$ and $d_i(1) = 1$.

Definition 6.4 (Planar SL-morphing). *An SL-morph is called a planar SL-morph if planarity is preserved throughout the morphing time $[0,1]$.*

A *linear morph* is a special case of SL-morphing, where d_i is the identity for all i, i.e., $d_i(t) = t$, for all i and t.

The problem of finding a planar SL-morph, or deciding if there is no such SL-morph for a given graph from an initial drawing to a final drawing, does not seem to be easy even for specific categories of planar graphs, e.g., for disjoint segments, paths, or cycles. It is also interesting to see if more specific kinds of morphs exist between two given planar drawings of a graph. For example, the category of solutions in which the vertices move one by one from their initial positions to final positions is easier to check and also interesting, because it is in NP whereas the general SL-morphing problem is not known to be in NP. More problems and challenges in SL-morphing are listed and discussed in Section 6.3.

For any SL-morph, we will have a sub-morph if we only look at the time interval $[t_s, t_f] \subset [0,1]$. A sub-morph can be seen as an SL-morph, where the initial vertex positions are defined by P_{t_s}, and the final vertex positions are defined by P_{t_f}. We only need to scale the time interval $[t_s, t_f]$ into $[0,1]$ to be consistent with the definition.

Definition 6.5 (Piecewise Linear Morphing). *A morph is "piecewise linear" if the morph time interval $[0,1]$ can be divided into a finite number of sub-intervals, with dividers $t_0 = 0 < t_1 < t_2 < \cdots < t_k = 1$, such that, the morph is linear in each of the sub-intervals $[t_i, t_{i+1}], 0 \le i \le k-1$.*

For intuition about SL-morphs it is helpful to think of the *"configuration space"*. The configuration space is an n-dimensional space, where n is the number of vertices of the morphing graph or a sub-graph. Each axis represents the position of one vertex parameterized from 0 (the initial position) to 1 (the final position). Each point in the configuration space represents a possible configuration of the graph in an SL-morph. An SL-morph is a path from point $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$ that is non-decreasing according to all axes. Some points/areas in the configuration space are not valid in a planar SL-morph because some edges are crossing. If we block out the invalid areas, a planar SL-morph is a monotone path from point $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$ through the free space.

Figure 6.4 shows the configuration space of a morphing path. The blue non-decreasing path that is shown in the configuration space corresponds to a planar SL-morph while the red path corresponds to an SL-morph that does not preserve planarity.

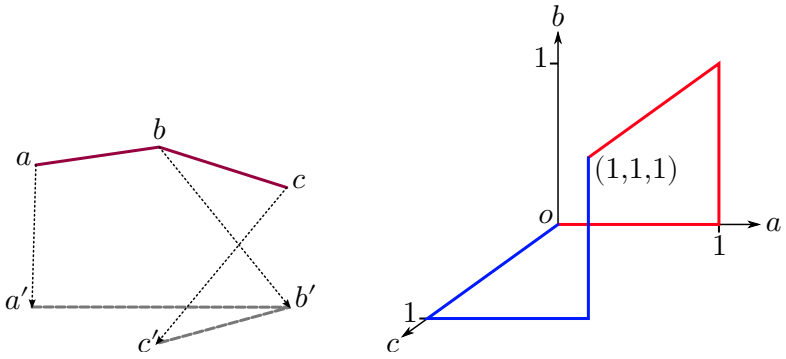


Figure 6.4: (a) A path with three vertices to be morphed from initial configuration a, b, c to final configuration a', b', c' (b) The corresponding configuration space with two non-decreasing paths from point $(0, 0, 0)$ to point $(1, 1, 1)$. In the blue path the vertices morph to their final configuration one by one in the order c, a, b. This SL-morph preserves planarity. In the red path the vertices morph one by one in the order a, b, c. This SL-morph does not preserve planarity because the path of vertex c' crosses the segment $a'b'$.

The blocked area in the configuration space is the union of the blocked regions corresponding to the crossing of two edges. We do not know the characterization of these simple blocked regions, but we conjecture that the boundary of the region corresponding to the crossing of two straight line segments should not be too complicated. For example, it would be very nice if each such region were convex. Unfortunately, that is not true. However, we still think the concave boundaries of the blocked regions are simple enough to

ensure the existence of a piece-wise linear path through the free space if there is any path.

Conjecture 6.6. *If there is a planar SL-morph for a planar graph, G, from initial vertex positions, P_0, to final vertex positions, P_1, then there exists a piece-wise linear SL-morph for G from P_0 to P_1.*

Later in Chapter 8 we will rule out the existence of some specific piece-wise linear planar SL-morphs such as the planar SL-morph when the path in the configuration space consists of segments parallel to axes, i.e., when the vertices move one-at-a-time to their final configurations. Specifically, we show examples where a planar SL-morph exists but there is no uninterrupted-one-vertex-at-a-time morph.

6.3 Background on Morphing

The first result on morphing planar graphs was proved by Cairns in 1944 [21]. It was a constructive proof that for every planar triangulation with two different planar drawings with the same faces and the same outer face, there always exists a morph from initial vertex positions to final vertex positions that preserves planarity using a finite sequence of linear morphs. Later, this result was extended by Thomassen from planar triangulations to general planar straight-line graphs [76].

Cairns's constructive proof has two drawbacks. First, the number of steps is exponential due to two recursions on morphing with one smaller size. Second, the constructed morph is not satisfactorily visualizable. Cairns constructs a morphing in which vertices may move very close together, and they may also move on complicated trajectories from their initial to final positions.

The existence result by Cairns and Thomassen was improved in a succession of papers [4, 6, 9] culminating in a paper by Alamdari et al. [3] showing that any straight-line drawing of a planar graph can be morphed to any other straight-line drawing of the graph (with the same faces) using a sequence of $O(n)$ linear morphs, where each linear morph is "unidirectional", i.e., all vertices move on parallel lines. The bound of $O(n)$ is tight in the worst case [3].

Angelini et al. also worked on morphing between convex graph drawings [7]. They show that we can morph any convex drawing of a planar graph to another convex drawing while convexity and planarity preserved at all times. This morph consists of $O(n)$ linear morph steps and this bound is tight in the worst case.

There are also results on morphing planar graphs (or sub-classes such as cycles, paths, or trees) while preserving additional properties, such as edge lengths [22], or edge directions [14]. When the lengths of edges is fixed the problem is called linkage reconfiguration. We may think of the edges as rigid bars and the vertices as flexible joints. The edges may rotate at the vertices as long as they do not cross other edges. Any path or cycle linkage configuration in 2D can transform (morph) to the canonical configuration–straight path or convex cycle–while planarity and the edge lengths are preserved at all times [22]. This is not necessarily true for paths in 3D space [30]. See the book by Demaine and O'Rourke for more interesting problems and results about folding and linkage reconfiguration [30].

Biedl et al. [14] worked on morphing from a planar graph drawing to another planar graph drawing where each edge has the same initial and final direction and such that the directions of all the edges (as well as planarity) are preserved all the time in the morph. They prove that such morphs always exist in the case of orthogonal drawings. However, the problem is NP-hard if there are three or more different edge directions.

There are studies also in graphics and modeling about morphing, with a slightly different focus. In animation and graphics, morphing is mostly about finding the correspondence between the different parts of two shapes to obtain a satisfactory transformation. Efrat et al. [33] defined the best morph between two poly-lines (paths) as the correspondence which gives the Fréchet distance between the polylines. The Fréchet distance is the minimum leash length we need to connect a man that walks forward on the first poly-line and a dog that walks forward on the second poly-line. At any time during the morph the man's and the dog's locations give us a correspondence between the two poly-lines. After finding the best correspondence, a linear morph can be computed and the maximum distance each point travels will be minimized. Planarity might be lost in this algorithm. Nöllenburg et al. [67] also use the term "morphing" to find the correspondence between the images of a river or a road that are in the form of poly-lines in different maps with different scales.

There are studies on visualizability of graph drawings [39, 19] that can be applied to the morphing of graphs too. One criterion for the visualizability of a morph is to see how the intermediate frames satisfy the parameters of a nice drawing. For example, is the graph simple in all intermediate steps? Does any vertex/edge coincidence happen, or how close are the vertices in intermediate steps? How different are the intermediate frames from the source and the target drawings?

Friedrich and Eades in 2002 [39] enumerate some criteria that make a transition from one drawing of a graph to the other drawing smooth. According to their model, preserving the mental map of a viewer and showing the changes in graph structure are two important factors. Although these factors are not well defined, Friedrich and Eades suggest that

making the path each node travels as short and simple as possible helps in maintaining the structure of the graph in the viewer's mind. This is one of our motivations for studying straight-line morphing.

Chapter 7

Straight-Line Morphing in the Continuous Setting

In this chapter we consider the SL-morphing problem: given two planar straight-line drawings of a graph, is there a planar straight-line morph between them? We consider this problem for special classes of graphs, namely disjoint edges, paths, or cycles. For a set of disjoint edges (segments) we prove that the problem is NP-hard. However, we still do not have any algorithm that decides the existence of a planar SL-morph even for a set of disjoint segments, so we do not know if the problem lies in NP.

For paths and cycles we look at some necessary conditions we can test in polynomial time. We also show some negative examples that refute some conjectures about specific categories of solutions for the case of paths and cycles.

We also explore SL-morphing in other restrictive settings when each edge of the graph in its initial position and final position together with the trajectories of its two vertices form a convex quadrilateral. With this limiting input condition the NP-hardness proof is no longer valid. However, we show that even for disjoint segments with convex morphing quadrilaterals the existence of local solutions (where all subsets of the problem have solutions) does not guarantee a global solution.

7.1 Preliminaries and Definitions

In SL-morphing we can define *morphing quadrilaterals* by looking at the morph of a single edge. Defining and looking at the properties of morphing quadrilaterals will help us later

in extracting properties and defining challenges in straight-line morphing.

Definition 7.1 (**Morphing Quadrilateral**). *Consider a single edge of the graph, drawn as initial segment ab and final segment $a'b'$. Then one vertex travels on segment aa', and the other travels on segment bb'. The quadrilateral formed by ab, bb', $b'a'$, $a'a$ is called the* morphing quadrilateral. *See Figure 7.1.*

The morphing quadrilateral of each edge has one of the following forms; see Figure 7.1:

- Convex: when the morphing quadrilateral is convex
- Simple Non-Convex: when the morphing quadrilateral does not cross itself but is not convex
- Crossing Morph Lines: when the two morph line segments (the segments each vertex traverses in an SL-morph) cross
- Crossing Edges: when the initial and final segments of the edge cross

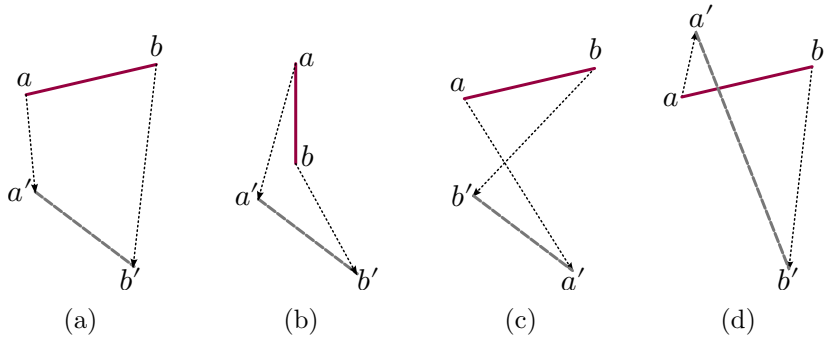


Figure 7.1: Four different types of morphing quadrilaterals: (a) convex; (b) simple non-convex; (c) with crossing morph lines; (d) with crossing edges

Each of the morphing quadrilateral types has its own properties and difficulties to morph: In *convex* morphing quadrilaterals, regardless of how the end vertices travel along their morphing paths, all of the area inside the quadrilateral is always passed over by the edge.

In *simple non-convex* morphing quadrilaterals it is still possible to morph by only using the inside area of the quadrilateral. However, we might need to use the outside area in

a planar SL-morph. Figure 7.2(a) shows an example in which we cannot have any SL-morph that preserves planarity unless edge x goes through the outside of its morphing quadrilateral at some time during the morph. Otherwise, edge y, which has a convex morphing quadrilateral, cannot morph while preserving planarity.

In morphing quadrilaterals with *crossing morph lines* or *crossing edges*, even though it may be possible to morph using only the inside area of the quadrilateral, the SL-morph that only uses the inside area is very restrictive. In the case of crossing morph lines, two vertices must morph to meet at the crossing point; then, they finish the morph by moving to their final positions. This scenario is not acceptable as a planar morph, because a segment would shrink to zero length, so two vertices would become coincident which is not allowed in a graph drawing. Therefore, some area outside of the morphing quadrilateral is always passed over in the case of crossing morph lines. Lemma 7.2 describes what the extra area will look like. This property of segments with crossing morph lines will be used later in designing Boolean gadgets. In the case of crossing edges, there is only one SL-morph scenario that keeps the edge inside the quadrilateral, i.e., when the two endpoints morph synchronously such that the edge always passes through the crossing point of the initial and final configurations of the segment.

Lemma 7.2. *Let s be a segment with initial position a_0b_0 and final position a_1b_1, and suppose its morphing quadrilateral has morph lines that cross at point o. Then, either some points of the triangle $\triangle a_0ob_1$ or triangle $\triangle b_0oa_1$ must be used in any PSL-morph of s. Furthermore, no PSL-morph of s uses points of both triangles.*

Proof. A scenario to morph a_0b_0 to a_1b_1, as discussed earlier, is to move both vertices at any speed to meet at point o, then continue to their final positions. See Figure 7.6. However, this scenario is not acceptable as the vertices pass each other and violate the planarity condition. Therefore, two vertices do not reach point o at the same time.

Assume, without loss of generality, that vertex a is the first vertex that meets the crossing point o. Then, no points of triangle $\triangle a_0ob_1$ will be passed by the segment and some points in triangle $\triangle b_0oa_1$ will definitely be passed by the segment in any planarity preserving morph scenario. Symmetrically, if vertex b meets the crossing point o first, no points of triangle $\triangle b_0oa_1$ will be passed and some points of triangle $\triangle a_0ob_1$ will be passed during an SL-morph that preserves planarity. □

Figure 7.2(b) and Figure 7.2(c) show the situations where we need to use the outside of the morphing quadrilateral for the cases of crossing morph lines and crossing edges, respectively. Similar to the simple-non-convex case, segment y needs segment x to completely clear segment y's morphing quadrilateral to morph successfully.

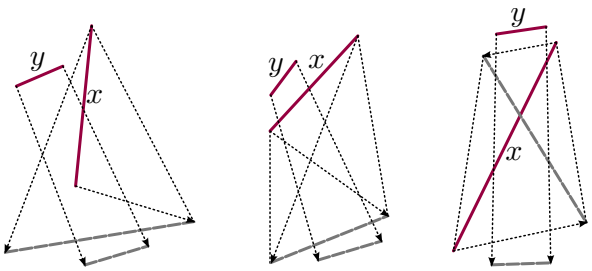


Figure 7.2: Sometimes it is not possible to morph without using the outside area of the morphing quadrilaterals. In all cases segment x must leave its morphing quadrilateral to allow segment y to morph: (a) segment x with simple non-convex quadrilateral; (b) segment x with crossing morph lines; (c) segment x with crossing edges

For any planar graph, some negative instances may be ruled out by looking at the edges as independent morphing elements. In general if there is no planar SL-morph for a subgraph of the original morphing graph, then there is no planar SL-morph for the original graph. This necessary condition in its simplest form is described in the following.

The Basic Necessary Condition. If there is a planar SL-morph for a graph G, then for any pair of edges e_1 and e_2, there is a planar SL-morph that morphs both e_1 and e_2. In other words, if there is a pair of edges e_1 and e_2 that may not be morphed through a planar SL-morph, then there is no planar SL-morph for the entire graph G.

Figure 7.3 shows some examples that pass the basic necessary condition and some examples in which the basic necessary condition fails. In Figure 7.3(a) segment x may morph to stand outside of the morphing quadrilateral of segment y (either on the right or the left side). Then segment y morphs to the final configuration. Segment x morphs into the final configuration last. In Figure 7.3(b) segment x may morph to the final configuration and clear the morphing quadrilateral of y first. Then segment y morphs to the final configuration. In Figure 7.3(c) there is no planar SL-morph because segment x may never clear the morphing quadrilateral of y. In Figure 7.3(d) also there is no planar SL-morph. In this example, segment x may clear the morphing quadrilateral of y, but may not do this without crossing segment y.

The basic necessary condition for a graph $G = (V, E)$ can be tested in time $O(|E|^2)$. That is because the basic necessary condition for a pair of segments is testable in constant time since the size of the problem is constant and the segments may only have a constant number of different relative positions. However, the basic necessary condition is not always

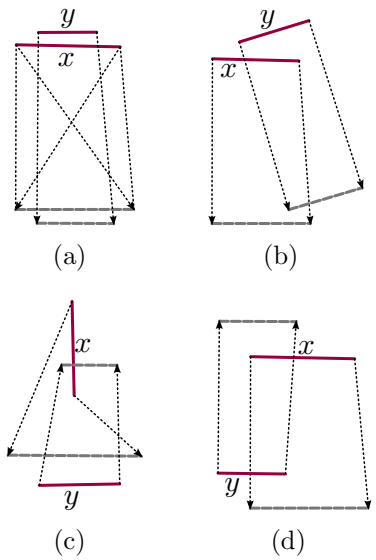


Figure 7.3: (a), (b) examples of pairs of segments where the basic necessary condition holds; (c), (d) the basic necessary condition does not hold for these pairs of segments.

sufficient. Figure 7.4 shows a counter example involving three segments. Each pair of the segments passes the condition and may morph successfully, but there is no planar SL-morph for the three of them. For segment y to pass the central common area of the three morphing quadrilaterals, segment z should have passed the area and exited the morphing quadrilateral of y. This means that it enters the morphing quadrilateral of segment x and blocks it. Segment x may not morph before segment y either because of their special configuration. In Section 7.4 we will see why this example does not have a planar SL-morph using the dependency graph that will be defined.

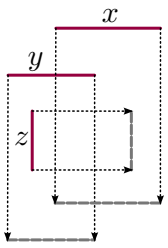


Figure 7.4: Each pair of these three segments passes the necessary condition, but there is no planar SL-morph for the three of them.

7.2 Planar SL-morphing of Disjoint Segments is NP-hard

The morphing problem is simpler on disjoint segments because each segment's movement is independent of the rest of the graph. On the other hand, this might lead to exponentially many ways to morph the entire set of segments when each segment has a few different ways to morph. Also, the planarity constraint means that choices made for one segment affect choices for other segments.

In this section we show that SL-morphing of disjoint segments is NP-hard. The reduction is from a version of the 3-SAT problem that is called the planar monotone 3-SAT problem. The 3-SAT problem is a Boolean satisfiablity problem in which the formula to be satisfied is in the form of the conjunction of a set of clauses, where each clause is the disjunction of three literals or fewer. Each literal is either one of the n Boolean variables or the negation of a Boolean variable. A 3-SAT problem is called monotone when for each clause, all the literals in the clause are either positive or negative. Both 3-SAT and monontone 3-SAT problems are known to be NP-hard [44].

Essentially, the planar 3-SAT problem is a 3-SAT problem when the following graph, G, is planar: G has a vertex for each clause and each variable in the 3-SAT problem. There is an edge between each clause vertex and all the three (or fewer) variables in it. It is shown that the 3-SAT problem is still NP-complete if the graph G is planar [63]. There are many geometrical problems shown to be NP-hard by reduction from the planar 3-SAT problem. In fact, even more restrictive versions of planar 3-SAT remain NP-complete.

Knuth and Raghunatan introduced the *rectilinear configuration* of graph G of a planar 3-SAT instance. A rectilinear configuration is a planar drawing of graph G such that all the variable vertices are located in a horizontal row and all the clause vertices are axis-aligned rectangles that are connected to their variables by vertical edges [60]. The planar 3-SAT problem is NP-complete even if the rectilinear representation of the 3-SAT problem is given.

Figure 7.5 shows a rectilinear representation of the graph G of a planar monotone 3-SAT instance (with dashed lines indicating the rectangles corresponding to the clauses). The figure also shows a planar drawing of G with a node for each clause inside its rectangle and with a one bend orthogonal path for each edge. We use the representation shown in Figure 7.5 later for our reduction.

Recently, in 2012, De Berg and Khosravi showed that the planar 3-SAT problem is still NP-hard even if it is monotone and the rectilinear representation is given [29].

Planar Monotone 3-SAT.

Input: A rectilinear representation of a planar monotone 3-SAT problem in which all positive clauses (the clauses with all positive variables) are located above the row of the variables, and all negative clauses (the clauses with all negative variables) are located below the row of the variables.

Output: Is the 3-SAT problem satisfiable?

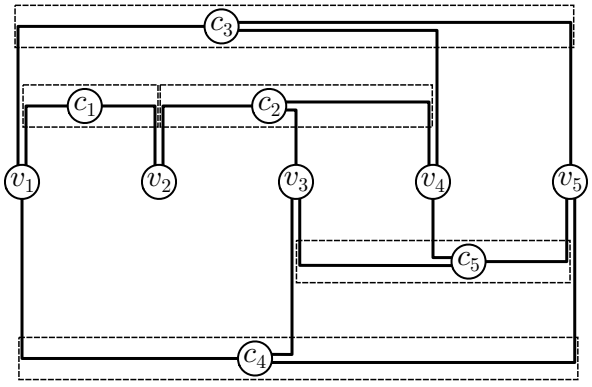


Figure 7.5: A rectilinear representation of a planar monotone 3-SAT instance: $(v_1 \vee v_2) \wedge (v_2 \vee v_3 \vee v_4) \wedge (v_1 \vee v_4 \vee v_5) \wedge (\neg v_1 \vee \neg v_3 \vee \neg v_5) \wedge (\neg v_3 \vee \neg v_4 \vee \neg v_5)$.
Dashed lines indicate the rectangle corresponding to each clause.

Theorem 7.3. *The planar SL-morphing problem is NP-hard even when the graph consists of disjoint segments, i.e., the graph is a matching.*

In the rest of this section, we show how to construct an instance of morphing of disjoint segments from an instance of monotone planar 3-SAT. The reduction involves designing variable gadgets that choose True/False values, clause gadgets, and wire gadgets that propagate values from the variable gadgets to the clause gadgets.

7.2.1 Boolean Gadget

Figure 7.6(a) shows a gadget consisting of two disjoint segments, s_1 and s_2. Segment s_1 in this gadget has a morphing quadrilateral with crossing morph lines most of which is located inside the convex morphing quadrilateral of segment s_2. We call s_1 the *crossing segment* of the gadget and s_2 the *straight segment* of the gadget. We show in this section

how this gadget can be used as a Boolean gadget in a 3-SAT construction. The idea is that s_1 must move out of the way to the left or right in order for s_2 to morph in a planar SL-morph.

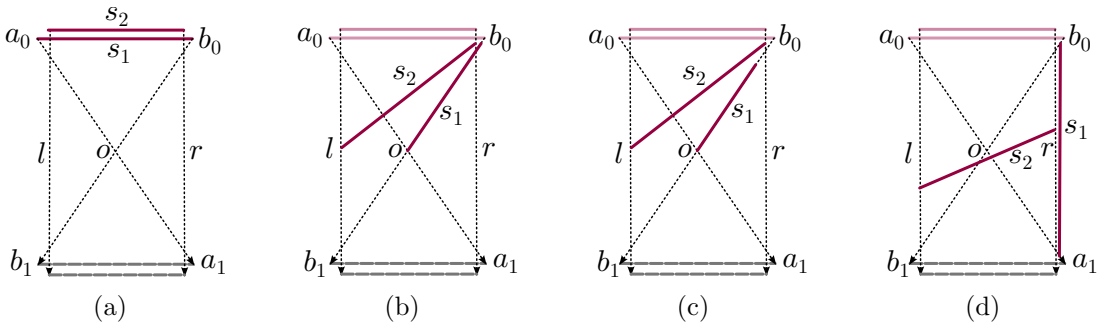


Figure 7.6: Boolean Gadget: (a) Initial configuration; (b)-(d) Illustrations for the proof of Lemma 7.4: (b) Time t' when the first endpoint of s_1 reaches point o; s_2 is behind point o in its morph journey, (c) We will not have a planar SL-morph if the other endpoint of s_1 enters the morphing quadrilateral of s_2 before s_2 passes over point o, (d) For s_2 to morph in a planar way, s_1 has to fully clear the morphing quadrilateral of s_2.

Figure 7.7 shows the configuration space of morphing segment s_1 in the Boolean gadget shown in Figure 7.6. The horizontal axis shows the configuration of the endpoint a and the vertical axis is the configuration of endpoint b; on each axis 0 is the initial configuration and 1 is the final configuration. Point t, $0 \leq t \leq 1$ along the horizontal axis corresponds to a being at position $(1-t)a_0 + ta_1$. Each point in the space corresponds to a possible configuration of segment s_1 during the morph. According to the definition of SL-morphs, a morph of segment s_1 corresponds to a path from the initial configuration, point $(0, 0)$, to the final configuration, point $(1, 1)$, that is non-decreasing according to both axes, i.e., xy-monotone. The two shaded areas in Figure 7.7 show the configurations of s_1 in which segment s_1 lies outside the morphing quadrilateral of segment s_2.

We prove below in Lemma 7.4 that in any PSL-morph solution of this gadget, the curve λ representing the morph of segment s_1 enters one of these shaded areas in the morph; i.e., there must be some time during the morph when s_1 lies outside of the morphing quadrilateral of segment s_2. If λ enters the top left shaded area in an SL-morph, this means that s_1 lies on the left side of morphing quadrilateral of segment s_2 at some time during the morph. We map this scenario to the "false" value of the Boolean gadget. Otherwise, λ enters the bottom right shaded area, which is equivalent to lying on the right side of morphing quadrilateral of segment s_2. We map this scenario to the "true" value

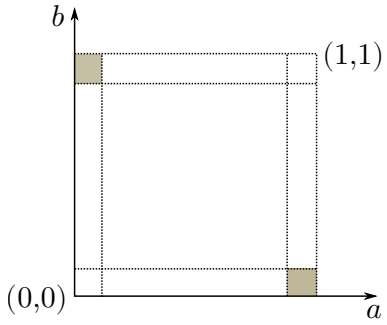


Figure 7.7: The configuration space of segment s_1 in Boolean gadget, shown in Figure 7.6. Any PSL-morph will enter one of the gray areas.

of the Boolean gadget. These two scenarios are mutually exclusive since an xy-monotone path cannot enter both shaded regions.

Lemma 7.4. *In the gadget shown in Figure 7.6, in any planar SL-morph there is some time point when segment s_1 lies outside of the morphing quadrilateral of segment s_2, either to the left or the right of the morphing quadrilateral of s_2. Furthermore, these two possibilities are mutually exclusive.*

Proof. In a PSL-morph of the gadget, consider time t when segment s_2 first reaches through the point o, the crossing point of segments a_0a_1 and b_0b_1. At least one of the endpoints of s_1 should have passed point o before t, as otherwise segment s_1 would have blocked segment s_2 from reaching point o. On the other hand, it is not possible that both a and b have passed o before time t, because otherwise segment s_1 would lie on the bottom half of the morphing quadrilateral of segment s_2 for the rest of the time, blocking s_2 from finishing the morph to its final configuration. Therefore, at time t, one but not both of the endpoints of s_1 has passed o. By symmetry, assume that a has passed o.

Now consider the first time $t' < t$ when a reaches point o. At time t' segment s_2 is above point o and even above segment ab, and endpoint b is on the right side; see Figure 7.6(b). At time t' point b may be outside or inside of the morphing quadrilateral of segment s_2; see Figure 7.6(b), and 7.6(c), respectively. We consider these two cases. If b is inside at time t' (Figure 7.6(c)) then for all times $t'' \geq t'$ part of the segment connecting a and b will lie in the morphing quadrilateral of segment s_2 making it impossible for s_2 to finish the morph.

Thus, b must be outside the morphing quadrilateral of s_2 at time t' (Figure 7.6(b)).The

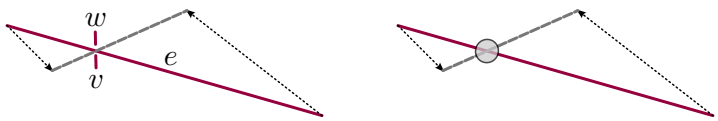
only way for segment s_2 to pass segment s_1 is for endpoint a of segment s_1 to exit the morphing quadrilateral of segment s_2 before the endpoint b enters it. Therefore, there will be a moment when segment s_2 is lying on the right side of the morphing quadrilateral of segment s_2; see Figure 7.6(d).

Symmetrically, if the endpoint b passes the intersection point o first, segment s_2 will lie on the left side of the morphing quadrilateral of segment s_2 at some time. These two possibilities are mutually exclusive, because each one corresponds to a shaded area shown in the configuration space in Figure 7.7 and any xy-monotone path in the configuration space may not pass through both shaded areas. □

We will use a rotated Boolean gadget as the core of each variable gadget.

7.2.2 Switch and Wire Gadgets

In this section, we introduce an important small gadget that works as a switch; see Figure 7.8. This switch will be used in making OR gadgets. Also, a chain of switches can be used as a wire in propagating choices about morphing of one segment to another part of the construction.



(a) (b)

Figure 7.8: Switch gadget: Two endpoints of segment e must morph simultaneously to avoid crossing with the stationary segments v and w. This gadget will be drawn as shown in part (b) for simplicity.

The switch gadget consists of a main segment, e, and two small segments, v and w. The segment e has a morphing quadrilateral with crossing edges. The two small segments, v and w, are stationary, i.e., the initial and final configurations are the same. These two segments, v and w, play the role of stationary obstacles that limit the morph of segment e. Figure 7.9 shows the configuration space of segment e's morph. The boundary of the configuration space is not part of the configuration space of PSL-morphs, because segment e crosses the segments v or w which are stationary. Therefore, to morph from point $(0, 0)$ to point $(1, 1)$ in the configuration space, segment e has to follow the narrow shaded area shown in Figure 7.9. In other words, in any PSL-morph of the gadget, moving one endpoint

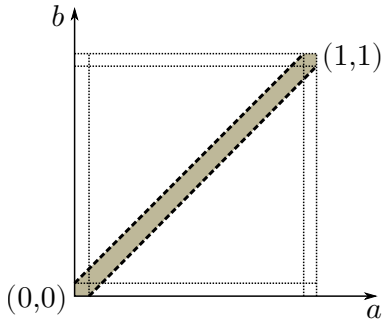


Figure 7.9: Configuration space (shaded) of segment e in the switch gate shown in Figure 7.8

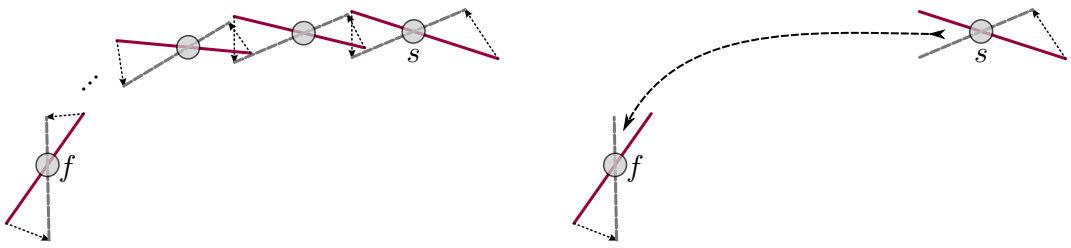
of segment e will cause the other endpoint to move as well. We will use this property to propagate choices made in variable gadgets to the clause gadgets.

Using switch gadgets, we can construct a chain that works as a directed wire that propagates the morph of a segment to some other part of the plane; see Figure 7.10. In Figure 7.10(a), when the main segment in switch s, as the first element of the chain, is near its final configuration at time t, the next switch in the chain must be near its final configuration too. Then, as we continue on, the last segment, f, will be near its final configuration at time t. Note that this is a directed dependency from s to f, not the other way around. In other words, segment f might be near its final configuration while switch s, or some other switch element of the chain before f, is at any valid configuration.

We can use smaller size switches in the chain to make the chain pass the narrower parts of the construction. Also, the chain may bend when necessary. The only essential property we should preserve is that each switch element of the chain must be near its final configuration when it clears the morphing quadrilateral of the main edge in the previous switch element. Figure 7.10(b) symbolizes how we show the chains in the constructions. The arrow goes from the start of the chain, s, to the end of it, f.

7.2.3 Clause Gadget

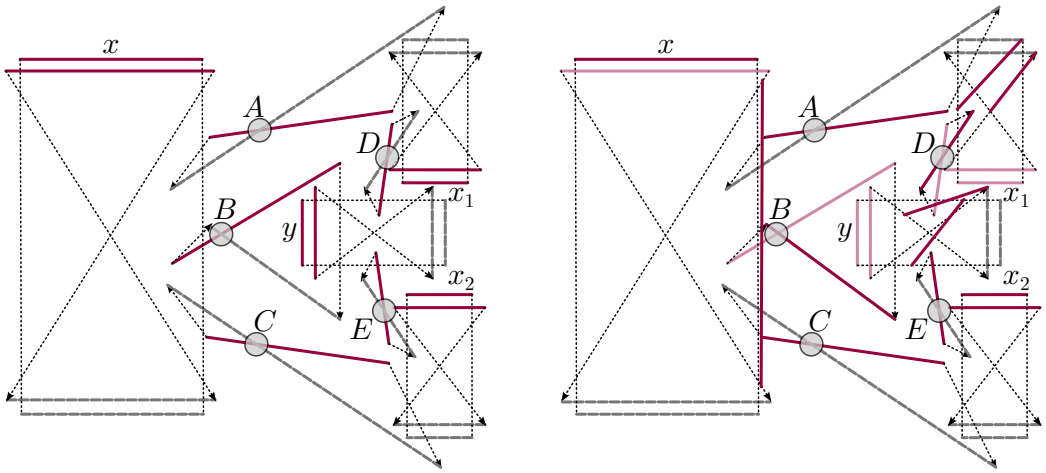
In this section we show how the straight line morph of a set of disjoint segments can form a clause gadget. Figure 7.11 shows a construction of an OR gadget using the Boolean gadget and the switch gadget. We will prove that this gadget implements $x = x_1 \vee x_2$, where x, x_1, and x_2 are the values of the Boolean gadgets. This means that for the Boolean gadget



(a) (b)

Figure 7.10: Switch chain gadget: The morph of one end, the switch f, must occur before the morph of the other end of the chain, switch s.

x to have "true" value, at least one of the Boolean gadgets, x_1 and x_2, must have "true" values. By cascading two of these OR gadgets, we will have a clause gadget for 3-SAT.



(a) (b)

Figure 7.11: OR gadtet: (a) the gadget in initial state; (b) the intermediate state when x_1 has true value.

Lemma 7.5. *In Figure 7.11, in order for the Boolean gadget x to have true value (where the crossing segment morphs to the right side), at least one of the Boolean gadgets x_1 and x_2 must have true value.*

Proof. According to the definition of the "true" value for Boolean gadget x, described in Section 7.2.1, if x has true value then there is a time, t, such that the crossing segment in

x lies on the right side of the morphing quadrilateral of the straight segment. To preserve planarity, switches A and C must be near their initial state, and the switch B must be near its final state at time t. For the main segment in switch B to morph, the following event should have happened before time t: In the middle Boolean gadget, y, both segments have cleared the area that the right end of switch B needs to pass. Consequently, there was a time, $t' < t$, when the crossing segment in y was on the top or the bottom of the convex morphing quadrilateral in y (instead of right or left, since the gadget is rotated). So, at time t' at least one of the switches, D and E, has morphed close to their final configurations, depending on which intermediate state (top or bottom) the gadget y chose.

Consider the case where the gadget y uses the top intermediate state to morph. The other case is symmetric. Then switch D is near its final configuration at time t'. Now we focus on the Boolean gadget x_1 and the two switches connecting to it, D and A, and claim that it is not possible for x_1 to morph through its left side. Assume, to the contrary, that the Boolean gadget x_1 morphs with the left intermediate state. Therefore, there is a time t'' such that the crossing segment in gadget x_1, called s, is lying on the left side of the convex morphing quadrilateral in x_1. Observe that $t'' < t'$ since after t' the main edge of the switch D will stay inside the convex morphing quadrilateral in x_1 permanently. Also observe that from t'' onward the top endpoint of s (at time t'') will stay in the morphing quadrilateral of the main edge of switch A. This leads to a contradiction, because $t'' < t$ and the switch A is near its initial configuration at time t. So, the Boolean gadget x_1 may not have false value in any PSL-morph of the OR gadget. □

Lemma 7.6. *In Figure 7.11, any valuation of Boolean gadgets, x_1 and x_2, that satisfies $x_1 \vee x_2$ can be part of an SL-morph solution of the entire gadget.*

Proof. In any valuation that satisfies $x_1 \vee x_2$ at least one of x_1 or x_2 has the true value, i.e., the gadget uses the right intermediate state to morph. Now we suggest a morphing scenario for the entire OR gadget. For the Boolean gadget(s) x_1, x_2, or both, which have true values, follow these steps: first morph the crossing segment to lie on the right side of the convex morphing quadrilateral; then, morph the straight segment to the middle of its way, passing the area of the first switch, but not entering the area of the second switch; then, morph the first (small) switch to its final configuration; for example see the state of the Boolean gadget x_1 in Figure 7.11(b).

Now, in the Boolean gadget y, at least one of the top and bottom sides (or both sides if both x_1 and x_2 have true values) is clear. Then, we can morph the gadget y to the intermediate state shown in Figure 7.11(b) in which the area of switch B is clear. We then morph switch B; this makes the right (true) side of the Boolean gadget x clear.

After completing the morph of x to its final configuration, switches A and C morph to their final configurations. Then, the Boolean gadgets x_1 and x_2, that are either in the intermediate state described above, or in the initial configuration, complete their morphs. Finally, either of the switches, D or E, which is not morphed yet, morphs to its final configuration, clearing the area of the gadget y to finish the morph. □

Note that in the OR gadget, the Boolean gadget x_1 is upside down. To fix that into a normal form of a Boolean gadget we suggest the *copy gadget*, shown in Figure 7.12. This gadget copies the right side value of a Boolean gadget, x, into the right side value of one or more Boolean gadgets which are flipped vertically.

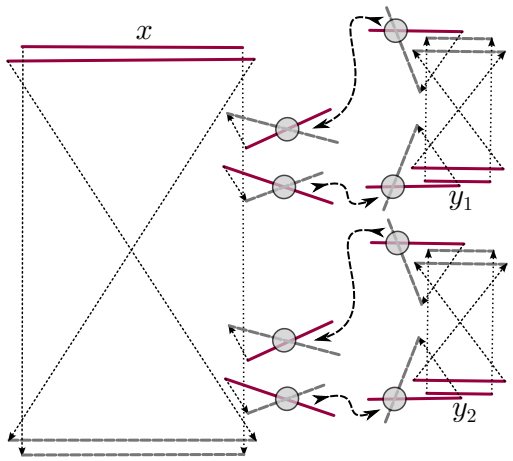


Figure 7.12: Copy Gadget

Lemma 7.7. *In Figure 7.12, if the Boolean gadget x uses the right intermediate state to morph in a planar SL-morph, then all the flipped Boolean gadgets y_i's (y_1 and y_2 in this figure) will use the right intermediate state to morph.*

Proof. First, we show that it is not possible for any of y_i's to morph using the left intermediate state. The Boolean gadget x uses its right intermediate state to morph, so there is a time t in which the crossing segment in x lies on the right side of the vertical dashed line in gadget x. At time t, the upper switch chains of all y_i's are near their initial configuration while the lower switch chains of all y_i's are near their final configuration. Assume, to the contrary, that one of y_i's morph using the left intermediate state. So there must be a time t' when the crossing segment stands on the left side of the vertical dashed line in y_i. At time t' the upper switch chain must be near its final configuration, because this segment

never leaves the area that the first switch of the upper chain needs to morph through after time t', so $t' > t$. On the other hand, the lower switch chain of y_i may not morph before time t' since one endpoint of the crossing segment is blocking the area all the time before t', so $t' < t$, which is a contradiction.

Second, we show that there is a planar SL-morph if all y_i's use their right intermediate states to morph. This can be done by following these steps: In all y_i's, the crossing segments go to their right intermediate states, and the straight segments travel half way, waiting in the middle of the Boolean gadget, clearing the area of the lower switch, and not entering the upper switch; then, the gadget x morphs into its right intermediate state, meaning that the lower switch in all y_i's has morphed near its final configuration. Then, the gadget x morphs to its final configuration, all upper switch chains morph to their final configurations, then, all y_i's complete their morphs to their final configurations. □

Note that the copy gadget, shown in Figure 7.12, only copies the right value of the Boolean gadget x, not the left value. The following lemma shows how it may occur.

Lemma 7.8. *In Figure 7.12, if the Boolean gadget x uses the left intermediate state to morph, then each of the flipped Boolean gadgets y_i (y_1 and y_2 in this figure) may use either the right or the left side to morph.*

Proof. The Boolean gadget x morphs through its left side, so only the morph of the straight segment in x may conflict with the morph of the two switch chains connected to each y_i. The following scenario shows how the flipped Boolean gadgets y_i's may morph. The straight segment in x passes the area of the first switch chain of y_1, but does not enter the area of the second switch chain yet; the right endpoint waits in the small area between two switches. Then, the first chain morphs to its final configuration. This removes any limitation from the Boolean gadget y_1, so it may morph to its final configuration through either the right or the left side. Then, the second switch chain from x to y_1 may morph to its final configuration, clearing the area that the waiting segment in x needs to proceed. Now, the Boolean gadget x is finished with y_1 and will repeat these steps with y_2, and continues on. □

To construct a 3-clause-gadget we cascade two OR gadgets and a copy gadget, as shown in Figure 7.13. Note that the switches in the OR gadget can be replaced with switch chains, as shown in this figure when the variables of the clause are located far from each other in the 3-SAT construction.

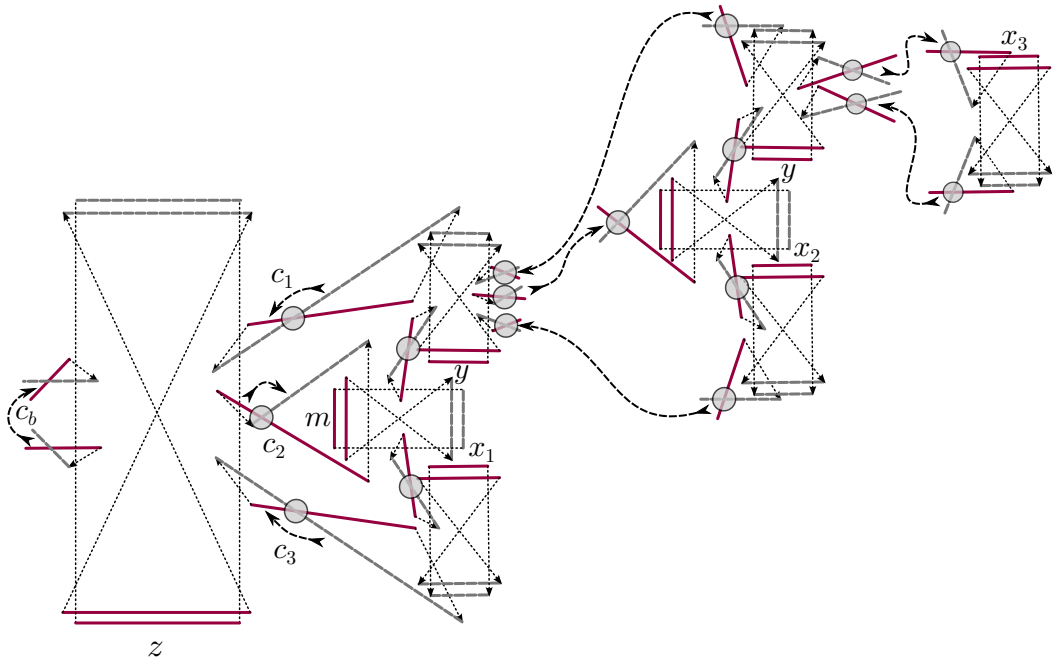


Figure 7.13: 3-Clause Gadget

Finally, we want to force the clause to be true, and to do that we must force a Boolean gadget to have a specific (true or false) value. This can be done by blocking the other side of the Boolean gadget, as shown in Figure 7.14.

Lemma 7.9. *In Figure 7.14 the Boolean gadget z may only have true value, i.e., for any planar SL-morph, z morphs through its right-side. Furthermore, z can morph through its right side.*

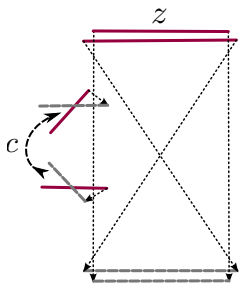


Figure 7.14: A Boolean gadget which may only have true value

Proof. Assume, to the contrary, that the Boolean gadget z uses the left intermediate state (false value) to morph. Then, there is some time when the crossing segment in z lies on the left side of the left vertical dashed line. This means that the first element of the switch chain c is near its final configuration to clear the area while the last element of the chain must be still near its initial configuration. This is not possible according to the property of switch chains discussed earlier.

To morph through the right side, the crossing segment in the Boolean gadget z goes into the right intermediate state. Then, the straight segment morphs half way and stops between the two endpoint of chain c. Then, the chain c morphs into its final configuration, and the straight segment may complete its morph into the final position. The crossing segment may finish its morph last. □

Note that in the complete clause construction, shown in Figure 7.13, the small chain c_b added to the left side of the Boolean z makes the entire gadget always true, as proved in Lemma 7.9. Now, we must ensure that all appearances of the variables in clauses have consistent true/false values. In the following section, we show how to join a variable gadget to all appearances of that variable in the clause gadgets.

7.2.4 The Final Construction

In this section, we show how to connect all appearances of the same variable in the 3-SAT problem together, ensuring that they all have consistent values, i.e., all identical literals have the same value and the literals which are the negations of other literals have opposite values.

Recall that we use one Boolean gadget for each variable v_i. We then use the copy gadget to copy the right (true) value of this Boolean gadget into all appearances of the positive form of variable v_i and copy the left (false) value into all appearances of the negative form of v_i ($\neg v_i$). Figure 7.15 illustrates this.

As shown in Figure 7.15, there is a pair of switch chains between the variable gadget and each appearance of the variable in clauses. One chain is directed from the variable to the initial part of the literal in the clause, and the other chain is directed from the final part of the literal to the variable, as shown in the figure. Recall that in the monotone 3-SAT problem the input is assumed to be a rectilinear representation with the variable gadgets lined up in the middle, all clauses with positive literals above, and all clauses with negative literals below. Then we connect each clause to its three variables with an edge, which is actually a pair of switch chains following the path in the rectilinear representation. These

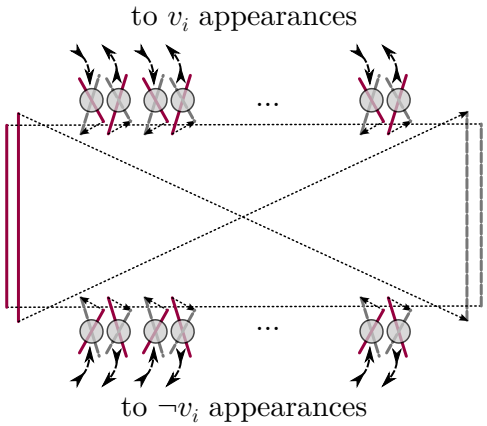


Figure 7.15: Variable Gadget: A rotated Boolean gadget that is connected to all its appearances in the clauses using the copy gadget.

chains do not cross, because the rectilinear representation is planar. Figure 7.16 shows the planar graph and the corresponding SL-morph construction of a monotone 3-SAT example.

The ordering of the chains in the variable side is also important. This will be clearer after the following lemmas are given about the entire 3-SAT construction.

7.2.5 Proof of Theorem 7.3

For an instance I of the monotone 3-SAT problem, let M denote the morphing instance as constructed above. To complete the reduction from the monotone planar 3-SAT problem to the PSL-morph of disjoint segments, we prove that I is satisfiable if and only if M has a PSL-morph. We separate the proof into two parts.

Lemma 7.10. *If M has a PSL-morph, then I has a satisfying truth value assignment.*

Proof. If M has a PSL-morph solution, then all main Boolean gadgets in all clauses, denoted by z in Figure 7.13, morph through their right sides, since we have shown in Lemma 7.9 that it is not possible for them to successfully morph through their left sides. Then, according to Lemma 7.5, at least one of the Boolean gadgets y_1 and x_1 has morphed through the right side. If y_1 has morphed through its right side, for the same reason, at least one of the Boolean gadgets y_2 and x_2 has morphed through their right sides. If y_2 has morphed through its right side, x_3 has morphed through its right side, as shown in the copy gadget in Lemma 7.7. Consolidating these, at least one of the Boolean gadgets x_1,

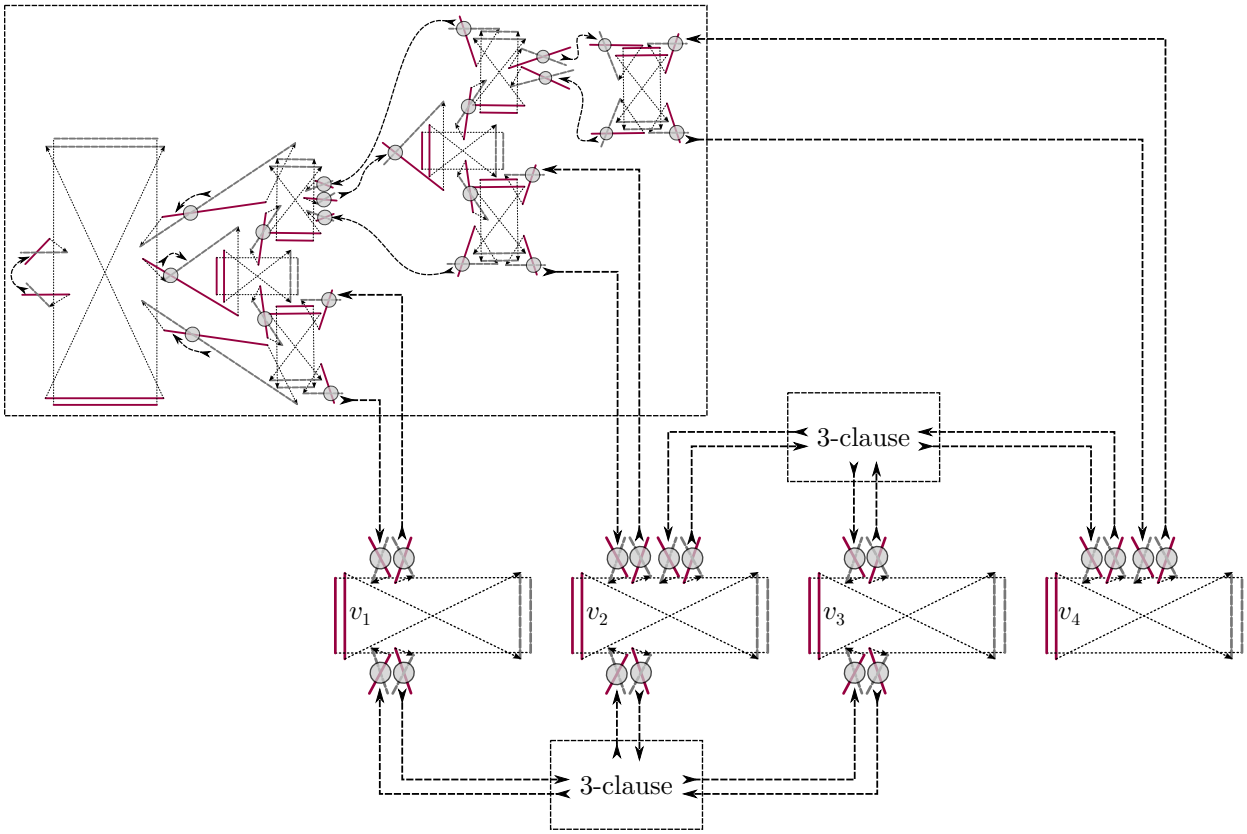


Figure 7.16: The full construction of a 3-SAT problem $(v_1 \vee v_2 \vee v_4) \wedge (v_2 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3)$; The first clause is shown with all details and the others are shown with a box.

x_2 and x_3 has true value. According to Lemma 7.7, the Boolean gadgets that have true values force the corresponding variable gadgets to morph accordingly by two switch chains connecting them to the corresponding variables. Therefore, all clauses are satisfied and, also, all appearances of the variables in clauses have consistent values. □

Lemma 7.11. *If I has a satisfying truth value assignment, then M has a PSL-morph.*

Proof. Consider the valuation of variables which satisfies the 3-SAT problem. We show how the entire construction morphs according to this valuation. The value (true/false) of each variable determines whether the corresponding variable gadget morphs through the top or the bottom side (if a variable has true value it morphs through the bottom side; otherwise, it morphs through the top side). Use the following step-by-step scenario for each variable gadget, v_i, that morphs through the bottom side (v_i takes the true value): The top

endpoint of the crossing segment morphs to the crossing point o. Now, the top endpoint of the other segment, a, may pass through all switches one by one without crossing this segment; see Figure 7.17.

Consider the clauses in which v_i appears positively. These clause gadgets are connected to v_i's gadget by chains starting from the top side of v_i's gadget. Let $C_{i_1}, \ldots, C_{i_k}$ be the left to right ordering of these clauses along the top of v_i's gadget. Then, the endpoint a stops between two chains connecting v_i's gadget to C_{i_1}. The first chain then morphs to its final configuration; this will clear v_i's final position in C_{i_1}, so v_i's Boolean gadget in C_{i_1} is free to finish its morph. Now, v_i's Boolean gadget in clause C_{i_1} morphs through its right side to the specific configuration shown in Figure 7.18. This intermediate configuration allows the other chain (between the top side of variable gadget v_i to clause gadget C_{i_1}) to morph; also it allows the clause to morph successfully later. Then, the other chain morphs and clears the way for endpoint a to proceed to the pair of chains connecting v_i to C_{i_2}. We now repeat the same steps for the next pair of switches on the path of a and continue on.

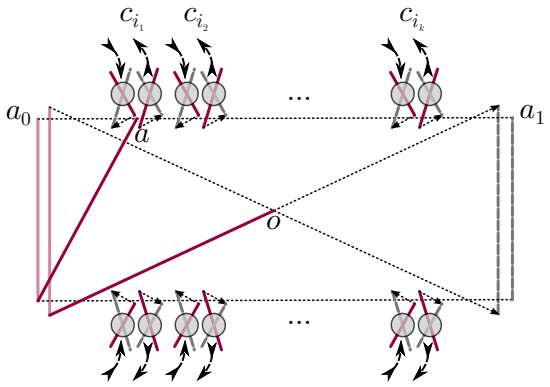


Figure 7.17: The intermediate state of the gadget of variable v_i to pass through the top switch chains one by one.

We repeat this scenario for all v_i's that take false value, as well. The only difference in such variables is that the process is vertically mirrored. Now all true literals in clauses are in the intermediate configuration that is shown in Figure 7.18. All false literals, on the other hand, may not have right-side values because they are connected to the opposite sides of the variable gadgets as proved in Lemma 7.7.

Now, we can morph the rest of the clauses according to the scenario described in the proof of Lemma 7.6 to the point where all false literals are in the morphing configuration shown in Figure 7.19. Note that, this can be done because in the 3-SAT solution each

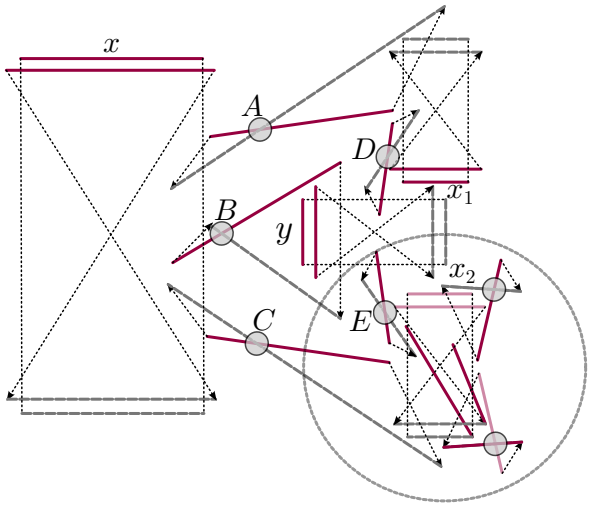


Figure 7.18: The state of the OR gadget when the true valued literals, x_2 in this example, are morphed to the intermediate state.

clause has at least one true literal. Now, all variable gadgets complete their morphs to their final configuration; then, each clause morphs to its final configuration. □

To complete the proof of Theorem 7.3 we also need to show that the reduction can be done in polynomial time.

Lemma 7.12. *The construction of the planar SL-morphing instance M from a planar monotone 3-SAT instance I takes polynomial time.*

Proof. The planar SL-morphing instance M consists of clause gadgets, variable gadgets, and wire gadgets connecting the variables to all their appearances in clauses. The complexity of drawing each clause gadget is constant, because the number of segments in each clause is constant, so the segments endpoints can lie on a constant-size grid; say a $k \times k$ grid. Each variable gadget may have $O(n)$ segments (depending on the number of clauses that include the variable) and can lie on a $k \times nk$ grid.

Wires consist of switch gadgets. Each switch has a constant number of segments and the number of switches in each wire depends on the length, width, and the number of bends in the wire. There are constant number of switches per length unit and in each bend. We need more switches when the width of the wire decreases, but it is still constant for wires with constant width. Figure 7.20 shows the wire gadget in bends and in straight

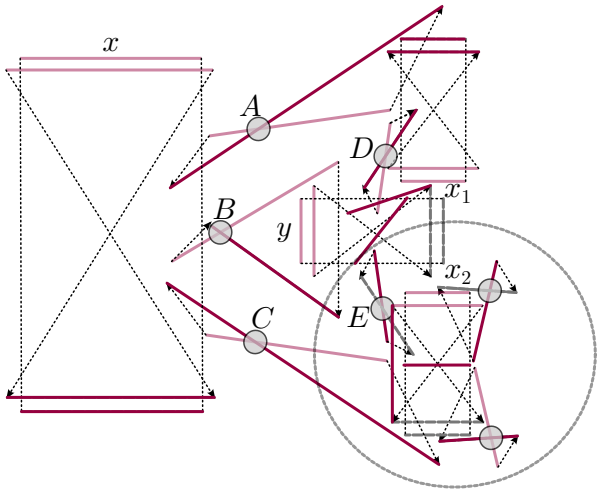


Figure 7.19: The state of the OR gadget when the false valued literals are morphing to the intermediate state.

parts with different widths. When the width of the wire decreases by half, the number of switches will be almost doubled.

Now we take a rectilinear representation of the planar monotone 3-SAT instance I as shown in Figure 7.5. Observe that this drawing lies on an $O(n) \times O(n)$ grid where n is the number of clauses and variables in I. We can expand this grid by a factor of nk and thicken every edge to a constant width strip. Now we replace each variable and clause node by the corresponding gadget and replace each edge by a wire made of switches. Note that each edge has length $O(n)$ and $O(1)$ bends in the original rectilinear representation and so the number of switches needed for each wire is $O(n)$. □

7.3 Morphing of Paths and Cycles

As shown in the previous section, the planar SL-morphing problem is hard even for the special class of disjoint segments. In this section we explore SL-morphing of paths and cycles and observe some of the properties and challenges in planar SL-morphing of these two simple classes of graphs. SL-morphing of cycles and paths, together with the case of disjoint edges, gives us a good understanding of the challenges of the morphing problem for general graphs. However, we do not yet know the complexity status of the planar SL-morphing problem for cycles and paths, in particular, we do not know if it is NP-hard.

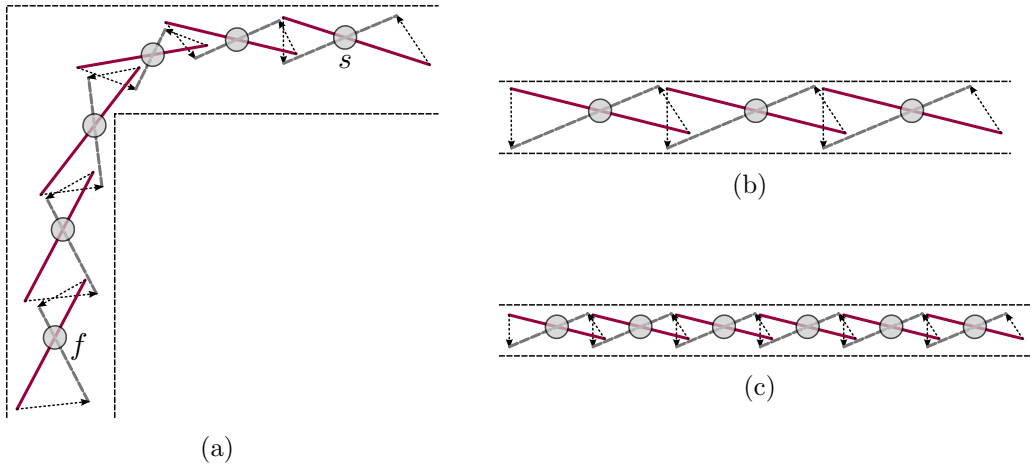


Figure 7.20: (a) The wire gadget in bends; (b) Wire gadget in straight parts with unit width; (c) Wire gadget in straight parts with half width

Figure 7.21 shows an example of a morphing path. A possible planar SL-morph for this path is to morph vertex c from its initial position to its final position (point c') first. Then the sub-paths on the right and the left side of vertex c can move independently without any conflict. Note that the linear morph does not preserve planarity for this example because of the edge cd that will be shrunk to zero length in the linear morph. Figure 7.22 shows an example of a morphing cycle. A planar SL-morph for this cycle is to first morph the edge ed to its final configuration. Then morph vertex c to its final position. Last, we can morph edge ab to its final configuration. In this example also the linear morph does not work because of the edge bc that will be shrunk to zero length in the linear morph.

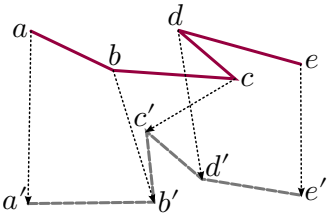


Figure 7.21: A path that has a planar SL-morph from the initial configuration (solid red lines) to the final configuration (dashed gray lines).

Planar SL-morphing of paths and cycles seems harder than planar SL-morphing of disjoint segments. When each vertex of a cycle or a non-end vertex of a path moves, two

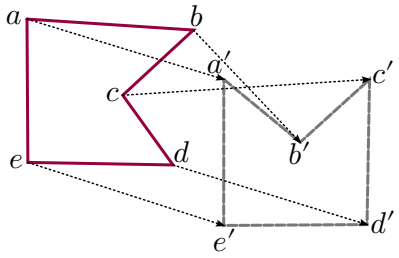


Figure 7.22: A cycle morphs from the initial configuration (solid red lines) to the final configuration (dashed gray lines).

incident edges move and sweep some area while in the case of disjoint segments moving each vertex only affects the movement of one edge. On the other hand, the denser the graph the more restrictions there are on the movement of the vertices and there is hope that the more limited solution space makes the problem easier to test. However, we do not know of any algorithm that decides the existence of planar SL-morphs for paths or cycles.

7.3.1 Applying the Basic Necessary Condition

As the first step toward deciding if a morphing instance involving a path or cycle has a planar SL-morph, we may apply the basic necessary condition discussed in Section 7.1. According to this necessary condition, for a global planar SL-morph solution all pairs of edges of the graph must have a planar SL-morph. Figure 7.23 shows a path example that does not have any planar SL-morph, because it fails the basic necessary condition.

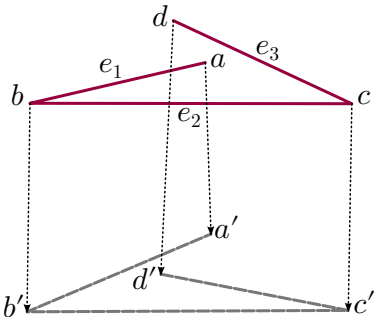


Figure 7.23: A path that cannot morph into the final configuration through a planar SL-morph, because two edges e_1 and e_3 do not pass the basic necessary condition. In particular, vertex a may never leave the morphing quadrilateral of e_3.

Even though the basic necessary condition is helpful in ruling out some negative examples, it does not guarantee the existence of a planar SL-morph for a path or cycle. Figures 7.24, and 7.25 show negative examples that pass the basic necessary condition but do not have any planar SL-morph.

Claim. *The example in Figure 7.24 has no planar SL-morph.*

Proof. In Figure 7.24 the morphing quadrilateral of edge ab is convex. Assume that endpoint b reaches the crossing point with ee' at time t. Then edge de must be in the lower part of the morphing quadrilateral of edge bc at time t, because otherwise it crosses the edge ab (no matter where the endpoint a is). Now, look at the configuration of edge bc. The endpoint b is at the crossing of ee' and bb'. Depending on the position of endpoint c at time t, the edge de either crosses bc or will be in the morphing quadrilateral of bc permanently after time t. Therefore, there is no planar SL-morph for this example, even though it passes the basic necessary condition. □

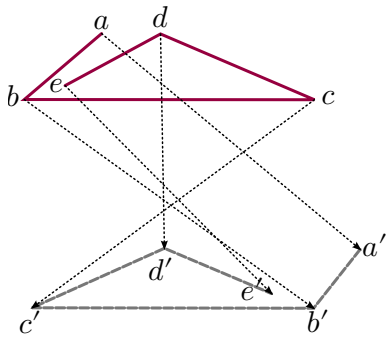


Figure 7.24: A morphing path that passes the basic necessary condition, but that cannot morph through a planar SL-morph.

The example in Figure 7.25 has no planar SL-morph for a more general reason. Observe that any cycle must have the same clockwise direction in the initial and final configurations in order for a planar SL-morph to exist. In other words, a planar SL-morph cannot model mirroring. This property has been observed for planar morphing, but we include an explicit proof below. Note that the cycle is mirrored in the counter example in Figure 7.25 that shows the basic necessary condition is not sufficient.

Lemma 7.13. *When traversing a cycle clockwise in the initial configuration starting at some vertex v, we visit the vertices in the same order as when the cycle is traversed clockwise in the final configuration starting at the same vertex v.*

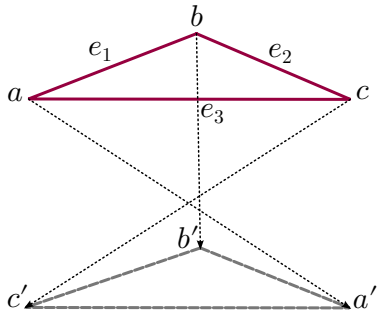


Figure 7.25: A cycle that cannot morph into a mirrored drawing but still passes the basic necessary condition.

Proof. Assume, to the contrary, that there is a planar SL-morph for a cycle mirrored in the final configuration. Since the SL-morphing is a continuous transformation from the initial configuration to the final configuration, there must be a time t in which the direction of the cycle flipped during the morph. This can only happen if the area inside the cycle shrinks to zero at time t and then increases in the flipped direction. A configuration in which the cycle has zero inside area is not a valid planar configuration, so this is a contradiction. □

7.3.2 Special Morphs

In this subsection we show through examples that for planar SL-morphs of paths and cycles it is not enough to consider simple morphs like linear morphs or the morphs in which the vertices morph one-by-one to their final positions (uninterrupted one-vertex-at-a-time morphs). Figures 7.26 and 7.27 show examples of a path and a cycle that have a planar SL-morph but there is no uninterrupted one-vertex-at-a-time or linear morph, as we now show.

In Figure 7.26 the linear morph does not preserve planarity because of two edges ab and bc that will pass over each other during the linear morph. Also there is no uninterrupted one-vertex-at-a-time morph that preserves planarity because vertex c must move together with vertices d and e to avoid crossings. However, a planar SL-morph is to morph vertices c, d, and e together into their final positions. Then, move vertices b and a in any order. Note that in this example all morphing quadrilaterals are convex.

In Figure 7.27 the cycle has a planar SL-morph: vertices a, g, and h may move together through a linear morph to their final positions. Then the rest of the vertices may move one by one in the following order from the initial position to the final position: b, f, e, d, c. The

linear morph does not preserve planarity for this example. Also, there is no uninterrupted one-vertex-at-a-time morph that preserves planarity because vertex b is the only vertex that may morph into the final position without crossing. After morphing b to b', there are no other vertices that can move to their final positions without losing planarity.

Figure 7.28 shows an example where the morph is a simple translation of a path, and the linear morph preserves planarity. However, we may not morph any single vertex to its final position without crossing. In this example also the morphing quadrilaterals are all convex.

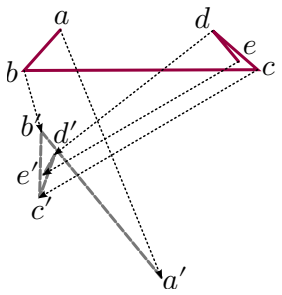


Figure 7.26: A path that has a planar SL-morph, but no uninterrupted one-vertex-at-a-time or linear morph preserves planarity.

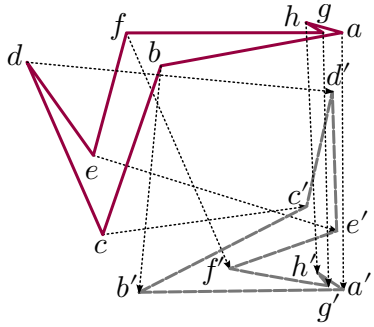


Figure 7.27: A cycle that has a planar SL-morph, but no uninterrupted one-vertex-at-a-time or linear morph preserves planarity.

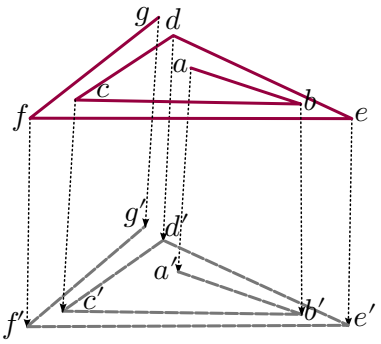


Figure 7.28: A path that has a planar linear morph, but no uninterrupted one-vertex-at-a-time morph.

7.4 Morphing of Graphs with Convex Morphing Quadrilaterals (Convex Morphing)

In this section we focus on planar SL-morphs for the specific case when the morphing quadrilateral of each edge is convex. We call this specific case of planar SL-morphing, *convex morphing*. We explore some properties of convex morphing when the graph is a matching (disjoint segments), a path, or a cycle. When all morphing quadrilaterals are parallelograms in a connected graph (such as a path or a cycle) the morph will be a simple translation in the plane and a linear morph is always a solution. Note that even the case of morphing disjoint segments will morphing quadrilaterals that are parallelograms is not trivial, because the edges may morph in different directions. For an example see Figure 7.4.

The main motivation for convex morphing comes from the properties of different types of morphing quadrilaterals. Among the four possible types of morphing quadrilaterals listed in Section 7.1, the case of convex morphing quadrilaterals is the only one in which the area of the plane that is passed over by the edge is fixed in all different morphing scenarios. Moreover, the NP-hardness proof of PSL-morphing mainly relies on morphing quadrilaterals that may sweep different areas in different morphing scenarios. It seems that the point that makes the problem hard is to decide which area is swept by each segment. Thus, convex morphing is a natural simplified PSL-morphing problem. In other words, an interesting question is: can we find a polynomial time (or even exponential time) algorithm for the planar SL-morph problem if the input only includes segments with convex morphing quadrilaterals?

7.4.1 A Necessary Condition for Convex Morphing

In this section we show a stronger version of the basic necessary condition for convex morphing. We have the first observation as follows:

Lemma 7.14. *For each pair of segments (s_1, s_2), if the two convex morphing quadrilaterals intersect (the quadrilaterals have a common point), then for any point p in the intersection, s_1 and s_2 pass over p in some order and this order is the same for all points in that common area.*

Proof. The intersection of the two convex quadrilaterals is a convex polygon R. Assume, to the contrary, that there are two points, p and q, in polygon R such that s_1 reaches p at time t_1 before s_2 reaches p, and s_2 reaches q at time t_2 before s_1 reaches q, where $t_1 \leq t_2$. Note that two segments may not reach a point at the same time because a planar morph is assumed.

Consider the sub-morph of the two segments in the time interval $[t_1, t_2]$. This means that the morphing quadrilaterals of s_1 and s_2 are cut at the times t_1 and t_2. We show that both the initial and final positions of segment s_1 in this sub-morph intersect the morphing quadrilateral of s_2. See Figure 7.29 for an example.

At the initial time, t_1, segment s_1 intersects the morphing quadrilateral of s_2, because s_1 passes through point p at this time and s_2 will reach p later. At the final time, t_2, segment s_2 passes through point q and segment s_1 will reach it later, so segment s_1 still crosses the segment pq at time t_2. Thus, considering the sub-morph in $[t_1, t_2]$ both the initial and final configurations of s_1 are inside the quadrilateral of s_2 making it impossible for them to have a planar SL-morph. □

Note that Lemma 7.14 does not imply that one segment passes all the common area before the other segment enters it, but it states that if we observe any single point p in the common region, morphing segments s_1 and s_2 pass over p in the same order.

The condition described in this observation can be extended to the case when more than two segments have overlapping morphing quadrilaterals. For any region that two or more segments must pass over, the segments pass over all points in the region in the same order. Some of these orderings may not be possible in a planar SL-morph because of the initial or final configurations of the segments. Here are some conditions that we can check to rule out some of these orderings for segments s_1 and s_2 with overlapping morphing quadrilaterals. The conditions are symmetric if we swap the names of s_1 and s_2, so the conditions must be checked for both pairs (s_1, s_2) and (s_2, s_1).

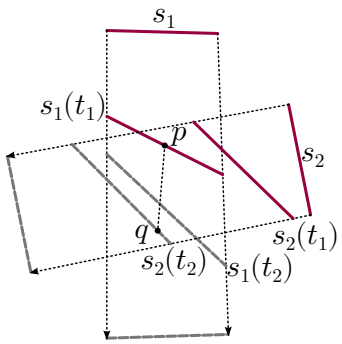


Figure 7.29: An example of two segments with convex morphing quadrilaterals: there is no planar SL-morph if s_1 reaches point p before s_2 and s_2 reaches q before s_1.

Let R be the intersection of the morphing quadrilaterals of s_1 and s_2:

1. If some point of the initial configuration of segment s_1 is in R, then s_1 must precede s_2 in any valid ordering according to Lemma 7.14; see Figure 7.30(a).

2. If some point of the final configuration of segment s_1 is in R, then s_2 must precede s_1 in any valid ordering according to Lemma 7.14; see Figure 7.30(b).

Lemma 7.15. *Two segments s_1 and s_2 with convex morphing quadrilaterals have a planar SL-morph iff the two conditions (1) and (2) (and these two conditions with swapped s_1 and s_2) allow it.*

Proof. If the two above conditions imply that no ordering of s_1 and s_2 is valid (s_1 must precede s_2 and s_2 must precede s_1), then there is no planar SL-morph solution because it will contradict Lemma 7.14 otherwise; see Figure 7.30(c) for an example.

On the other hand, if conditions (1) and (2) allow an ordering such as (s_1, s_2), it means that the initial position of s_2 and the final position of s_1 do not intersect R. Then, we can morph segment s_1 to exit R. We then morph s_2 to its final position. If none of the conditions (1) and (2) apply to the segments s_1 and s_2 (and none of the conditions apply to the swapped segments, s_2 and s_1), it means that R contains no point of the initial/final configuration of either segment, then both orderings are possible; see Figure 7.30(d). □

The above conditions force some ordering on the edges passing common regions. We may construct a *dependency* graph G_d based on these conditions as follows: Graph G_d has a vertex for each edge of the convex morphing graph G. There is an edge in G_d between two

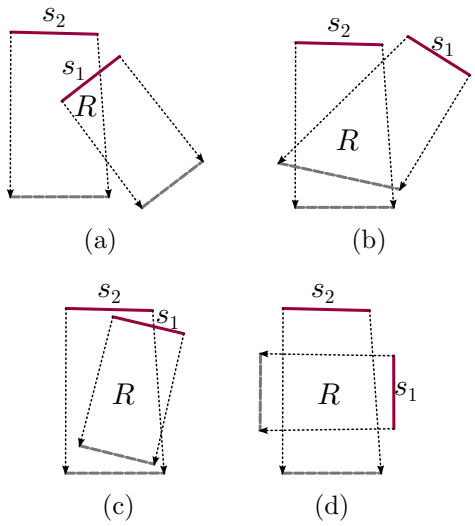


Figure 7.30: Two convex morphing segments with overlapped morphing quadrilaterals: (a) s_1 must pass over the points of the common region first; (b) s_2 must pass over the points of the common region first; (c) no planar SL-morph exists; (d) there is no constraint on the ordering of s_1 and s_2 passing over points of the common region.

vertices iff the morphing quadrilaterals of the two edges overlap. The two above conditions (1) and (2) determine the direction of some of the edges. An edge $(s_1, s_2) \in G_d$ is directed from s_1 to s_2 if s_1 must pass the points in R first. Let $\overrightarrow{G_d}$ denote the partially directed version of graph G_d formed by the two conditions (1) and (2). Figure 7.31 shows a convex morphing path G and the corresponding $\overrightarrow{G_d}$ graph. In this example all edges of $\overrightarrow{G_d}$ are directed, but in general, some may be undirected.

A planar SL-morph M yields a fully directed version of G_d that we denote by $\overrightarrow{G_d}(M)$. Note that this is well defined by Lemma 7.14. The dependency graph $\overrightarrow{G_d}(M)$ may have directed cycles. However, by Lemma 7.14, the morphing quadrilaterals of the segments involved in such a directed cycle may not share a common point. If all the morphing quadrilaterals contain a common point p, then the order in which segments pass p is a total order.

We may try to find a planar SL-morph by looking at $\overrightarrow{G_d}$ and by orienting the rest of the edges. Figure 7.32 shows a convex morphing example with n disjoint segments ($n = k+2$) that does not have any planar SL-morph because the dependency graph G_d always contains a directed cycle corresponding to a shared common point.

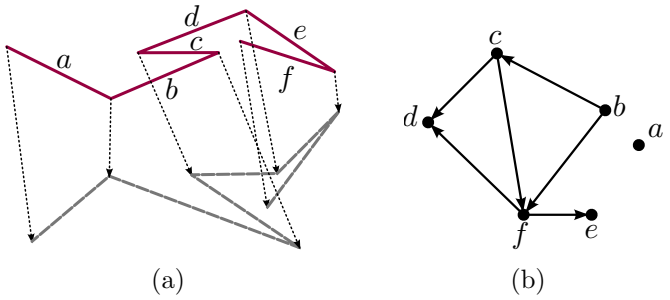


Figure 7.31: A convex morphing path and the corresponding dependency graph $\overrightarrow{G_d}$

Figure 7.32(b) shows the dependency graph $\overrightarrow{G_d}$. Vertex b in this graph has edges to all other vertices, because the morphing quadrilateral of segment b overlaps with the morphing quadrilateral of all other segments. The other vertices in G_d form a path because the morphing quadrilateral of each segment a_i only overlaps with the morphing quadrilateral of the two segments before and after a_i in the order. All edges (a_i, a_{i+1}), $1 < i < k-1$, are directed from a_{i+1} to a_i since the initial configuration of segment a_{i+1} is inside the morphing quadrilateral of segment a_i. The edge c, a_1 is directed from a_1 to c because the final configuration of segment c is inside the quadrilateral of segment a_1. The edges from b to c and a_k are also determined in $\overrightarrow{G_d}$ as follows: edge (b, a_k) is directed from b to a_k and edge (c, b) is directed from c to b.

We observe that any directions the remaining edges take, we will have a directed triangle of the form (a_i, a_{i+1}), (a_{i+1}, b), (b, a_i). This triangle corresponds to a shared common area between the morphing quadrilaterals of three segments a_i, a_{i+1}, and b. Therefore, this example does not have a planar SL-morph. We also observe that there is a planar SL-morph for any proper subset of the segments in this example. This shows that the existence of a planar SL-morph cannot be tested by its sub-problems.

We now turn to one positive result. From the discussion above, we must orient the undirected edges of $\overrightarrow{G_d}$ so that there is no directed cycle corresponding to morphing quadrilaterals that share a common point. We can show that it is enough to impose this condition on the cycles of size three (triangles).

Lemma 7.16. *The dependency graph $\overrightarrow{G_d}(M)$ has a directed cycle corresponding to morphing quadrilaterals that share a common point iff it has such a cycle of size three (triangle).*

Proof. Any directed cycle of size $k > 3$ in $\overrightarrow{G_d}(M)$ that corresponds to a common region among k morphing quadrilaterals forms a clique in G_d, because any two morphing quadri-

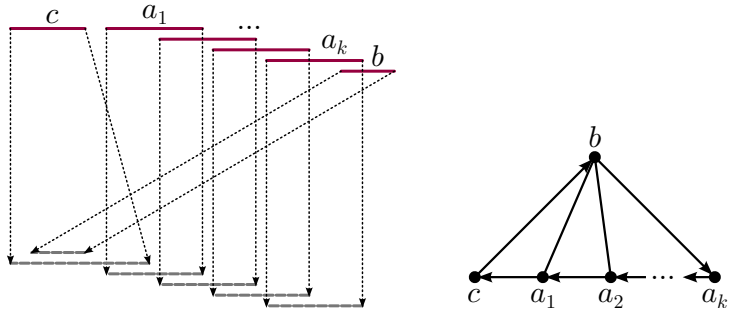


Figure 7.32: A convex morphing example that does not have a planar SL-morph because the necessary condition fails.

laterals in the cycle share a common region. Thus, if we have a directed cycle C of size k in $\overrightarrow{G_d}(M)$, all other edges between any two vertices in the cycle are also present in G_d and are directed. To find a directed cycle of a smaller size we only need to take any edge e connecting two non-consecutive vertices in C. Depending on the direction of e we have a directed cycle on one side of e (including e) or on the other side. Both cycles are smaller in size than k. Repeating this process, we will end up with a directed triangle. □

So, we only need to check if all the triangles in $\overrightarrow{G_d}$ corresponding to the quadrilaterals that share a common point are acyclic. This condition is summarized as follows:

Necessary Condition for Convex Morphing. In any planar SL-morph M of a convex morphing graph G, the dependency graph $\overrightarrow{G_d}(M)$ does not include any directed triangle that corresponds to three morphing quadrilaterals that share a common point.

Some of the edge directions are fixed in all planar SL-morphs; these are captured in the partially directed graph $\overrightarrow{G_d}$. All other edges of G_d must be oriented in any planar SL-morph M. If there is no orientation of the edge that satisfies the necessary condition for convex morphing, then there is no planar SL-morph.

However, this condition is not sufficient and there are some acyclic orientations of G_d that do not represent any planar SL-morph. Figure 7.33 shows an example. The morphing quadrilaterals of every two of the three edges x, y, and z intersect, so G_d has all three edges. However, none of the edges are directed in $\overrightarrow{G_d}$. Also, the three morphing quadrilaterals do not share a common point so any orientation of G_d will pass the necessary condition. Let R_{xy} be the intersection of the morphing quadrilaterals of x and y, R_{xz} be the intersection of the morphing quadrilaterals of x and z, and R_{yz} be the intersection of the morphing

quadrilaterals of y and z. The orientation shown in Figures 7.33(b) is when the points in R_{xy} are passed by x first, the points in R_{yz} are passed by y first, and the points in R_{xz} are passed by z first. There is no planar SL-morph with this orientation, because any of the segments are dependent on some other segment to reach the region in which it must pass the points first. Figure 7.33(c) shows another possible orientation (again with a cycle) that corresponds to a planar SL-morph. In this orientation the points in R_{xz} are passed by segment x first, so segment x can pass over R_{xz} and wait in the gap between R_{xz} and R_{xy}. Similarly, segments y and z can pass over R_{xy} and R_{yz}, respectively, and wait before entering their second intersection region. Then, they all can finish their morphs into the final positions.

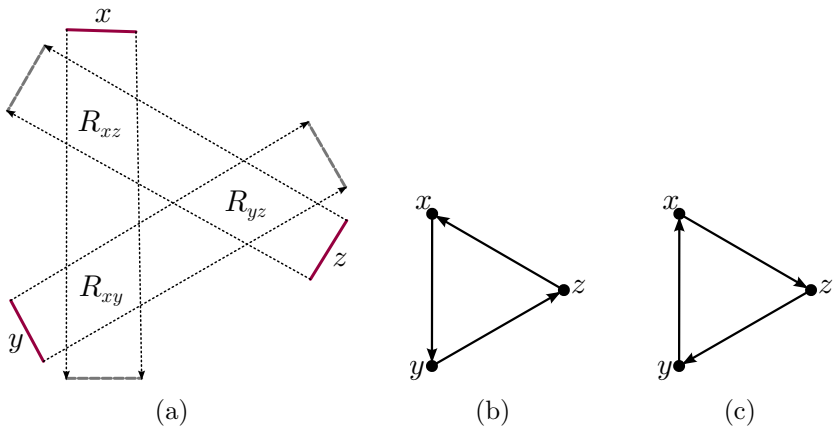


Figure 7.33: Convex morphing of a set of disjoint segments with two possible orientations of $\overrightarrow{G_d}$ that both pass the necessary condition for convex morphing. The orientation in (c) represents a planar SL-morph but the orientation in (b) does not.

As shown in Figure 7.33 the necessary condition for convex morphing is not sufficient at least for disjoint segments. We could not find a counterexample for paths or cycles, so we make the following conjecture:

Conjecture 7.17. *A convex morphing path or cycle has a planar SL-morph iff the dependency graph $\overrightarrow{G_d}$ has an orientation that does not include any directed triangle that corresponds to three morphing quadrilaterals that share a common point.*

Chapter 8

Morphing One Vertex/Edge at a Time

In the previous chapter we considered the PSL-morphing problem: given initial and final drawings of a graph, is there a PSL-morph between them? We showed that this problem is NP-hard for a set of disjoint segments. We do not know of any, even exponential time, algorithm that decides the PSL-morphing problem for disjoint segments. So, we do not know if the problem in general is NP-complete. We conjecture that the PSL-morphing problem is PSPACE-hard. That is because the *sliding tokens* problem which has some similarities to the PSL-morphing problem, is PSPACE-hard [50].

In this chapter we narrow our focus and study more specialized morph definitions each specifying a more restricted category of morphs. We explore planar morphs where only one vertex or one edge moves at a time while the rest of the graph is stationary. We use "1V" to denote one vertex moving at a time and "1E" to denote one edge moving at a time. Moving only one edge is not possible in general, but makes sense when the graph is a matching graph. We further distinguish whether the vertex/edge that moves travels from its initial to final position in one step; we call this "uninterrupted" vertex/edge morphing, and we use "U1V" and "U1E" to denote uninterrupted one-vertex-at-a-time and uninterrupted one-edge-at-a-time morphings, respectively.

We show that the PSL-morphing problem for disjoint segments remains hard for some of these restricted categories of morphs, but becomes testable in polynomial time for others.

Furthermore, we observe through examples that these different restricted categories of planar morphs are not equivalent (i.e., do not solve the same set of inputs) even for the case of disjoint segments. The diagram in Figure 8.1 shows the relation between the variations

of PSL-morphs. Except for the border between Planar SL-morphing and 1V morphing where we are not sure if they really define two different classes of morphs, the examples in Figure 8.2 show that they solve different sets of inputs.

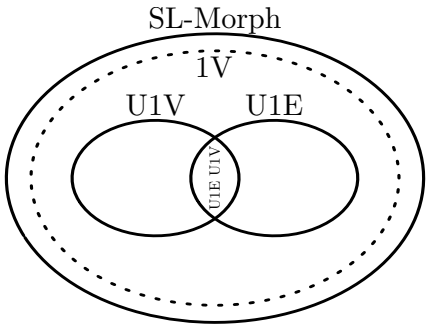
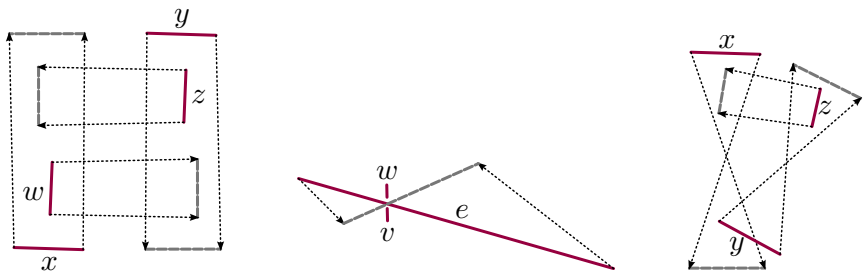


Figure 8.1: The containment diagram of sets of inputs solved by the restricted versions of planar SL-morphing.

Figure 8.2(a) shows a set of disjoint line segments in initial and final positions such that there is a planar SL-morph but no U1E or U1V morph. In this example, at least one of the segments w and z must pause its morph in the area between x and y. This scenario may not arise in any U1E or U1V morph. The next example in Figure 8.2(b) (which is also the switch gadget in Chapter 7) can be morphed through a U1E morph. However, it has no U1V morph since the endpoints of the segment e must move together or must take tiny 1V steps. The last example in Figure 8.2(c) shows the other direction: there is a U1V morph but no U1E morph. It has no U1E morph because segments x and y may not morph first, and if segment z morphs first it will block segment x permanently. However, it has a U1V morph by morphing the right endpoint of segment x to its final position first. Then segments y and z may morph to their final configurations and the other endpoint of segment x will morph to its final position.

Table 8.1 shows the summary of our results about restricted categories of planar SL-morphs for disjoint segments. For NP-hardness results, similar to the general problem discussed in the previous section, the reductions are from planar 3-SAT and planar monotone 3-SAT problems. Each of these special categories of morphs is explained in more detail in the following sections.

The NP-hardness reduction for uninterrupted one-vertex-at-a-time (U1V) morphing is explained in Section 8.1. Uninterrupted-one-edge-at-a-time (U1E) morphs and three variations (U1E 1V, U1E Linear, and U1E U1V) are described in Section 8.2. U1E morphing and the two first variations are shown to have polynomial time algorithms while the U1E



(a) (b) (c)

Figure 8.2: (a) A set of disjoint segments with a planar SL-morph but no U1V or U1E morph; (b) A set of disjoint segments with a U1E morph but no U1V morph; (c) A set of disjoint segments with a U1V morph but no U1E morph

U1V	NP-complete	Section 8.1
U1E	P	Section 8.2
Linear U1E	P	Section 8.2.1
U1E 1V	P	Section 8.2.1
U1E U1V	NP-complete	Section 8.2.2

Table 8.1: Summary of results for one-at-a-time planar SL-morphing of disjoint segments

U1V morph problem is proved to be NP-complete in Section 8.2.2. The last column of Table 8.1 shows which results are in which section.

8.1 Uninterrupted One-vertex-at-a-time Morphing (U1V)

In this section the *uniterrupted one-vertex-at-a-time* morphs, a special category of the SL-morphs, are defined and studied in detail. In this category of morphs, one vertex of the graph morphs from its initial configuration to its final configuration at once while the rest of the graph is stationary. Even though this category of morphs is more restricted and we can test the existence of such morph for a given input in exponential time, we prove that the decision problem remains NP-complete. Uninterrupted one-vertex-at-a-time or U1V morphing is precisely defined as follows:

Definition 8.1 (One-Vertex-at-a-Time Morph (1V)). *An SL-morph is* one-vertex-at-a-time *or 1V when the morph time interval* $[0, 1]$ *can be divided into a finite number of sub-intervals delimited by* $t_0 = 0 < t_1 < t_2 < \cdots < t_k = 1$*, such that in any time interval*

$[t_i, t_{i+1}], 0 \leq i \leq k-1$, *all vertices are stationary except one that is moving. Without loss of generality, we can assume that the non-stationary vertex moves with uniform speed in* $[t_i, t_{i+1}]$, *i.e., with a linear morph.*

Definition 8.2 (**Uninterrupted One-Vertex-at-a-Time Morph (U1V)**). *An SL-morph is* uninterrupted one-vertex-at-a-time *or U1V if the movement of any vertex is done at once without interruption. Formally, the morph time interval* $[0, 1]$ *can be divided into n sub-intervals, where n is the number of vertices of G, such that in each sub-interval all vertices are stationary except one that moves from its initial position to its final position.*

Theorem 8.3. *For a set of disjoint segments, finding a U1V morph that preserves planarity is NP-Complete.*

Finding a U1V morph that preserves planarity for a given morphing graph simply involves checking if there is an order of the vertices of the graph such that the vertices may morph one by one in that order without losing planarity. This problem is in NP as we can check all different orderings of the vertices in the worst case. Similar to the PSL-morphing problem, the NP-hardness reduction is from the monotone planar 3-SAT problem. Recall that monotone planar 3-SAT means that every clause has all positive or all negative literals and the graph of variables in clauses has a planar rectilinear drawing where clauses with positive literals are above, variables are in the middle, and clauses with negative literals are located below [29]. See Figure 7.5 in Chapter 7 for the drawing of the graph for an instance of a planar monotone 3-SAT problem.

The rest of this section provides materials we need to construct an instance of a U1V morphing of disjoint segments from any instance of planar monotone 3-SAT. This involves designing variable gadgets, clause gadgets, and wires to connect each variable to all its appearances in the clauses and propagate the chosen True/False values.

8.1.1 Boolean Gadget

A single segment with crossing morph lines can work as a Boolean gadget in the U1V problem since there are two different ways to morph it. Once each vertex starts moving, it must go to the end of its trajectory to the final configuration. Therefore, we only may determine the ordering in which the vertices of the segment morph. The two different orderings, as shown in Figure 8.3, lead to different areas of the plane to be swept.

If the left endpoint of the segment morphs first (Figure 8.3(b)), the morphing of the left endpoint sweeps the right triangle $\triangle oba'$ and the segment lies on the right-side of

the morphing quadrilateral in the intermediate state. Then, we say the segment morphs through its right-side area and map it to the "true" Boolean value. Otherwise, if the right endpoint of the segment morphs first (Figure 8.3(c)), symmetrically, the segment morphs through its left-side area, and we map it to the "false" Boolean value. The shaded areas in Figure 8.3(b) and 8.3(c) show the area that will be used when the segment morphs through the right-side or the left-side area, respectively. A single Boolean gadget will simply work as a variable gadget in the construction.

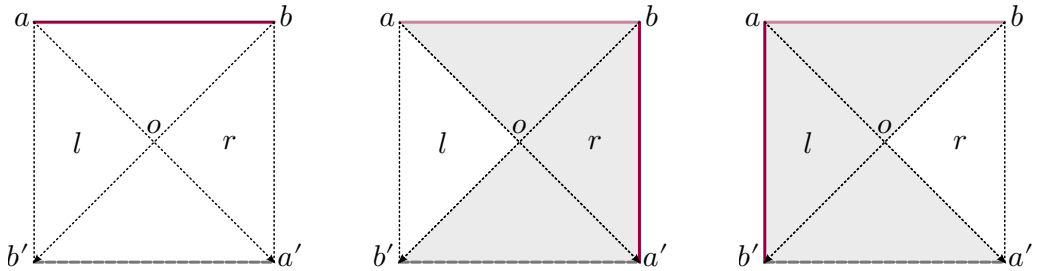


Figure 8.3: (a) A single segment in the U1V problem works as a Boolean gadget and thus as the variable gadget in the construction. (b) The Boolean gadget in the right (true) intermediate state. The gray area shows the area that will be swept when the segment morphs through its right side. (c) The Boolean gadget in the left (false) intermediate state. The gray area shows the area that will be swept when the segment morphs through its left side.

8.1.2 Clause Gadget

Figure 8.4 shows a gadget that works as an OR-gate in the U1V morph problem. By cascading two of these OR gadgets, and also by blocking the left side of the Boolean gadget, we construct a clause that always has true value; see Figure 8.5.

Lemma 8.4. *In the gadget shown in Figure 8.4, if the Boolean gadget (segment) z has true value (morphs through its right-side), then at least one of the Boolean gadgets x_1 and x_2 has morphed through its right-side.*

Proof. It is sufficient to prove that it is not possible that the segment z morphs through its right-side, and both x_1 and x_2 morph through their left-sides. Assume, to the contrary, that it occurs in some planar U1V morph. Segment z morphs through its right-side, so

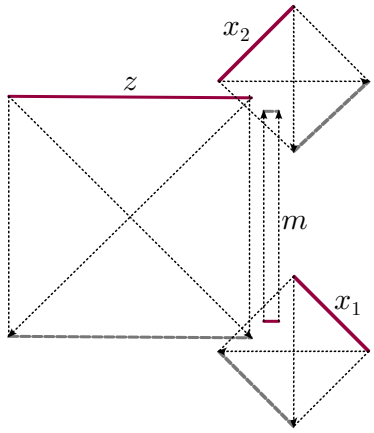


Figure 8.4: The OR gadget: In any planar U1V morph, if Boolean gadget z is true, then $x_1 \vee x_2$ is true.

there is some time point t in which the segment stands vertically on the right side of the morphing quadrilateral. Both segments x_1 and x_2 need some of the area occupied by z at its right intermediate state to morph. Therefore, both endpoints of segment x_1 should have morphed before time t, because after time t segment z will never clear the area that segment x_1 needs to morph. So, both endpoints of segment m have morphed before time t to clear the area segment x_1 needs to morph. On the other side, both endpoints of segment x_2 are in their initial positions at time t. That is because the right endpoint of segment z must clear the area that segment x_2 needs to morph. Segment x_2 may not finish its morph after time t either, because of the segment m which is in its final configuration and will occupy the left side of segment x_2 permanently after time t. □

Lemma 8.5. *In the gadget shown in Figure 8.4, any true/false valuation of Boolean gadgets x_1 and x_2 that satisfies $x_1 \vee x_2$, can be a part of a planar U1V morph when the Boolean gadget (segment) z has true value (morphs through its right-side).*

Proof. In the valuation that satisfies $x_1 \vee x_2$, if x_1 has true value (morphs through its right-side), then the area that segment x_1 needs to morph does not overlap with any of m and z. This works for any U1V morph scenario in which x_2 morphs after the morph of segment z is finished and before the morph of segment m is started. Segment x_1 in this case may morph any time without any conflict. If x_2 is assigned true value in the given valuation, for a similar reason, x_2 may morph any time without any conflict with other segments in the gadget. A guaranteed successful scenario is to morph segment m first,

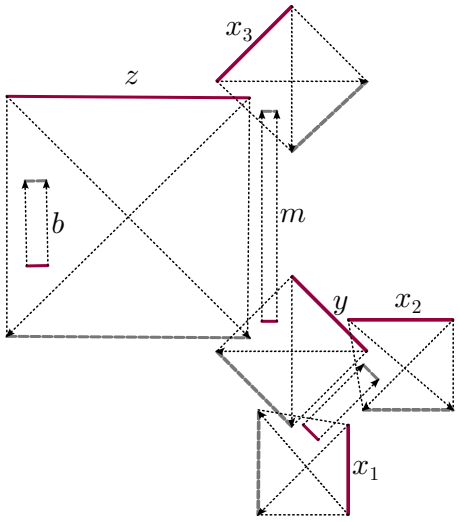


Figure 8.5: The clause gadget in the planar U1V morph problem

then morph x_1, through either the right or the left side (based on the given valuation), and to morph segment z last. □

We now show how to construct an OR gadget for 3 variables by cascading two basic OR gadgets. See Figure 8.5

Lemma 8.6. *In any planar U1V morph of the gadget shown in Figure 8.5, at least one of the Boolean gadgets x_1, x_2, and x_3 morphs through its right-side (takes true value).*

Proof. Segment z in this gadget, may only morph through its right-side because of segment b that permanently blocks the area segment z needs when it morphs through the left-side. Therefore, according to Lemma 8.4, at least one of the Boolean gadgets x_3 and y morphs through the right-side. If y morphs through its right-side, for a similar reason, at least one of x_1 and x_2 morphs through the right-side. □

Lemma 8.7. *In the gadget shown in Figure 8.5, any valuation that satisfies $x_1 \vee x_2 \vee x_3$ can be a part of a planar U1V morph for the entire gadget.*

Proof. This gadget is the cascade of two OR gadgets that is shown in Figure 8.4. If the given valuation satisfies $x_1 \vee x_2$, it means that Boolean gadget y takes true value and there is a U1V morph scenario for both OR gadgets according to Lemma 8.5. Otherwise, the Boolean gadget y takes false (left) value and the Boolean gadget x_3 must take true value.

Similar to the scenario given in the proof for Lemma 8.5, a U1V morph solution is to morph segment m first, then the entirety of the smaller OR gadget, and segments x_3 and z last. □

Note that the Boolean gadgets of false x_i's morph first. We will need this property later in the proof of Theorem 8.3.

In the next section, we introduce the variable gadget and the materials we need to connect different appearances of the same variables in clauses.

8.1.3 Copy and Wire Gadgets

The gadget shown in Figure 8.6 works as an AND operator that copies the right (true) value of the Boolean gadget z into all n Boolean gadgets connecting to its right side. We will use the AND gadget later as the core of the variable gadget.

Lemma 8.8. *In the gadget shown in Figure 8.6, if segment z morphs through its right-side, all variables $x_1, \ldots, x_n$ ($n = 2$ in this figure) should morph through their right sides.*

Proof. Segment z morphs through its right-side, so there is a time t in which the left endpoint of segment z has morphed into its final position while the right endpoint is still in its initial position. This means that the segment stands vertically where it crosses the initial and final configurations of all the x_i's at time t. So, all x_i's should be either in the right or the left intermediate position at time t. Being at left intermediate position is not possible because it blocks the area that the left endpoint of segment z needs to morph, so all x_i's should morph through their right-sides. Moreover, they are all standing vertically on the right side of their morphing quadrilaterals when segment z stands vertically on the right-side of its morphing quadrilateral. They all may finish their morphs to the final configuration after segment z finishes the morph and clears the area. □

Using the same idea in the copy gadget, we design a chain of segments to copy the value of a Boolean segment into some other parts of the plane. The chain is essentially the AND-gate with a single entry which is cascaded several times, as shown in Figure 8.7. When the first element of the chain, x, is in the right position, the next element of the chain must be in the right position as well for the same reason discussed in Lemma 8.8. As we continue on, all elements of the chain to the last one, y, are in the right positions. Unlike the chain in the general settings of the SL-morph problem, this chain is bidirectional, which

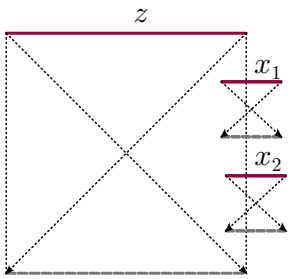


Figure 8.6: The gadget that works as an AND-gate: if segment z morphs through the right-side x_1 and x_2 have to morph through their right-sides.

means that it is symmetric in both directions and may transfer the left value of the Boolean segment y into the left value of the Boolean segment x as well.

As shown in Figure 8.7, we can make the chain of smaller segments pass through the narrow parts, and the chain may also bend when it is necessary. Figure 8.7(b) shows how the chain is symbolized for simplicity.

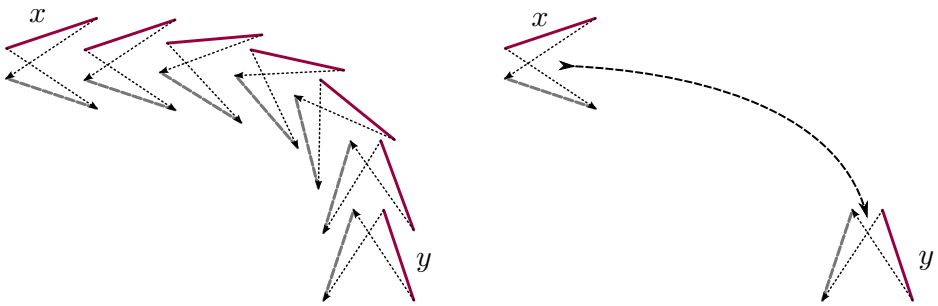


Figure 8.7: Chain gadget that transfers the right value of the Boolean segment x into the other Boolean segment y as directed with arrow. The same chain may also transfer the left value of the Boolean segment y into x.

We use these chains to connect a variable to all its appearances in the clauses. The variable gadget of v_i in the U1V problem is a Boolean gadget such that its "true" side is connected to all the appearances of $\neg v_i$ in clauses using the AND-gate, and its "false" side is connected to all the appearances of v_i in clauses. Figure 8.8 shows an example.

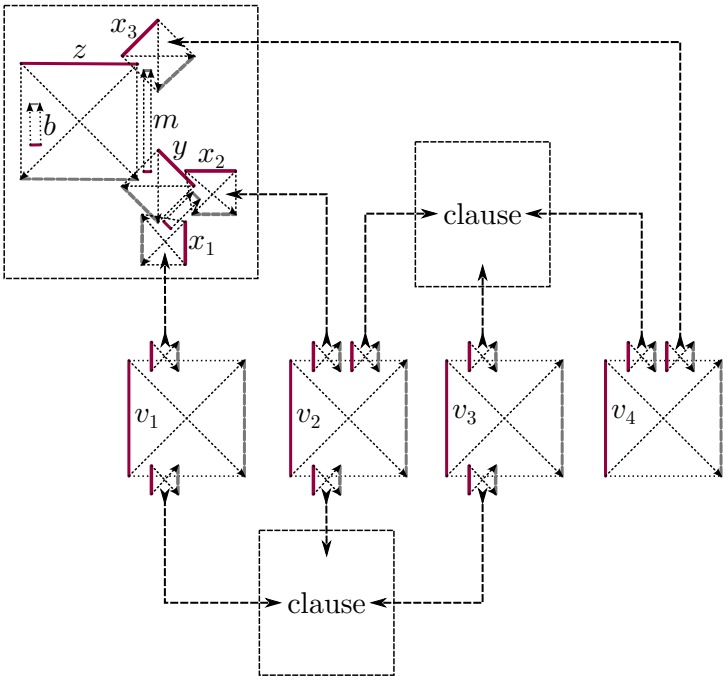


Figure 8.8: The full construction of a monotone planar 3-SAT instance $(v_1 \vee v_2 \vee v_4) \wedge (v_2 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3)$. The first clause is shown with all details and the others are shown with a box.

8.1.4 Proof of Theorem 8.3

For an instance I of the monotone 3-SAT problem let M denote the morphing instance as constructed above and shown in Figure 8.8. To complete the reduction from the monotone planar 3-SAT problem to the U1V morph of disjoint segments, we prove that I is satisfiable if and only if M has a U1V morph that preserves planarity (a planar U1V morph). We separate the proof into two parts.

Lemma 8.9. *If M has a planar U1V morph, then I is satisfiable.*

Proof. In the planar U1V morph of M, each variable gadget $v_1, \ldots, v_n$ morphs either through the top or the bottom intermediate state. Define a valuation V of variables in I in which a variable v in V has true value if the variable gadget v in the planar U1V morph of M morphs through its bottom intermediate state, and v has false value otherwise, i.e., the variable gadget v morphs through its top intermediate state. We show that the valuation V satisfies I.

Lemma 8.6 states that all the clauses are satisfied in any planar U1V morph of M. It is sufficient to show that the value of each literal in the clauses is compatible with the valuation V we defined. For positive clauses (the clauses above the variables in M), the true literals are connected to the corresponding variables with the wires, ensuring that all are assigned true. Therefore, the valuation V satisfies all the positive clauses. It is symmetric for the negative clauses since the satisfied literals in negative clauses are connected to the bottom of the corresponding variables, making it impossible to morph them through the bottom intermediate state. Therefore, the valuation that comes from a planar U1V morph satisfies I. □

Lemma 8.10. *If I is satisfiable, then M has a planar U1V morph.*

Proof. Assume that the valuation V satisfies I. We give a scenario to morph M based on the valuation V. For each variable v_i which is assigned "true" in V, we morph the Boolean segment v_i in M through its bottom-side, and for each variable v_i which is assigned "false" in V, we morph the Boolean segment v_i through its top-side.

A tricky point about the AND-gate and the wires in U1V morphing is that at the time t in which the Boolean segment of a variable gadget v_i is in top/bottom intermediate state, all the Boolean segments connected to the top/bottom side of the variable gadget v_i (and thus all the elements in the wires) are in their top/bottom intermediate state. Thus, the Boolean segments x_i in clauses connected to the top/bottom side of a variable gadget (the literals corresponding to $\neg v_i/v_i$) which take false value in the valuation V, must all be in their left intermediate states at the same time. Moreover, we observe in the proof of Lemma 8.7 that the literals with false values in a clause gadget cannot morph in any ordering. A possible order of the false valued literals in a clause gadget is the order from the lower indices to the higher ones.

This special point may cause two dangers in the morphing. First, if we have a clause with repeated literals such as $v_1 \vee v_1 \vee v_2$ then the morphing scenario in Lemma 8.7 does not work if v_1 takes false value, because the two Boolean segments in the clause connected to the variable gadget v_1 must be in their left-side state synchronously which is not possible. The scenario that handles such situations is to first eliminate repeated literals by simplifying the 3-SAT instance I before the construction.

The other danger, or issue, is that the wires force an interaction between the allowed morphing order inside the variable gadgets and the allowed morphing order inside the clause gadgets. According to the proof of Lemma 8.7, false valued literals among x_1, x_2, and x_3 in each clause morph from the lowest index to the highest index, and the true valued literals will morph afterwards. For example, if x_1 and x_3 have false values according

to the valuation V, then the Boolean segment x_1 morphs through its left-side first, then x_3 morphs through its left-side, and x_2 will morph through its right-side last.

To resolve this issue, it is sufficient to morph the variable gadgets from right to left in the morphing construction M while the three variables appear in each clause from right to left are connected to x_1, x_2 and x_3 in this order. Also note that in this scenario the Boolean segments corresponding to literals that take true values are free to morph after the false literals.

The entire scenario can be as follows: morph the variable gadgets in the middle row of the construction from the right-most one to the left-most one according to the valuation V. If variable v_i in V has true value, morph the Boolean segment v_i through its bottom-side. Then all the elements of the wires connecting the bottom side of the Boolean segment v_i to the negative clause gadgets (the clauses that have $\neg v_i$ as a literal) will be in their left intermediate states when the Boolean segment v_i is in its bottom intermediate state. This is symmetric if variable v_i has false value. We then morph it through its top-side and it forces the Boolean segments corresponding to all appearances of v_i in the clauses to morph through their left-sides at the same time.

Then, we finish morphing of the Boolean segment v_i to its final position and the Boolean segments representing the literals in clauses are free to finish their morphs. According to the internal scenario for each clause gadget in the proof of Lemma 8.7, we will have a global U1V morph that preserves planarity for M.

The valuation V satisfies I, so at least one of the literals in each clause has true value and by Lemma 8.7 there is a planar U1V morph for the clause. □

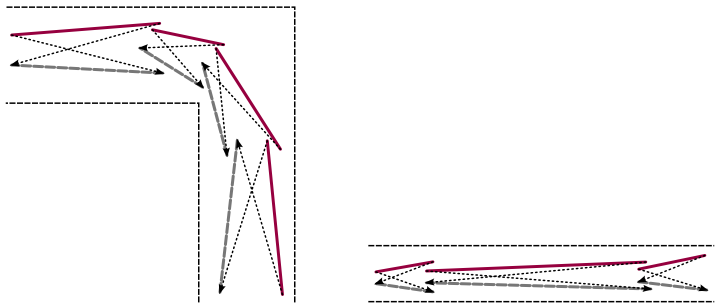
Lemma 8.11 will complete the proof of Theorem 8.3 by showing that the reduction can be done in polynomial time.

Lemma 8.11. *The construction of the planar SL-morphing instance M from a planar monotone 3-SAT instance I takes polynomial time.*

Proof. The planar SL-morphing instance M consists of clause gadgets, variable gadgets, and wire gadgets connecting the variables to all their appearances in clauses. Similar to the case of planar SL-morph, the complexity of drawing each clause gadget is constant and it can lie on a constant-size grid; say a $k \times k$ grid. Each variable gadget may have $O(n)$ segments (depending on the number of clauses that include the variable) and can lie on a $k \times nk$ grid.

The number of segments in wire gadget in planar U1V morphing only depends on the number of bends, because the number of segments in each bend and in each straight part of the wire are constant. See Figure 8.9.

Now we take a rectilinear representation of the planar monotone 3-SAT instance I as shown in Figure 7.5. Observe that this drawing lies on an $O(n) \times O(n)$ grid where n is the number of clauses and variables in I. We can expand this grid by a factor of nk and thicken every edge to a constant width strip. Now we replace each variable and clause node by the corresponding gadget and replace each edge by a wire made of switches. Note that each edge has $O(1)$ bends in the original rectilinear representation and so the number of segments needed for each wire is $O(1)$. □



(a) (b)

Figure 8.9: (a) U1V wire gadget in bends; (b) U1V wire gadget in straight parts

8.2 Uninterrupted One-Edge-at-a-Time Morphing (U1E)

In this section we define *uninterrupted one-edge-at-a-time morphing*. Similar to U1V morphing in which the vertices morph one by one from the initial position to the final position, here the edges of the graph morph one by one to the final position without interruption. However, we will show that unlike U1V morphing we can test in polynomial time if there is a U1E morph that preserves planarity.

Definition 8.12 (Uninterrupted One-Edge-at-a-Time Morphing (U1E)). *An SL-morph of a set of disjoint line segments is* uninterrupted one-edge-at-a-time *or U1E if the movement of any edge is done at once without interruption. Formally, the morph time interval* $[0, 1]$ *can be divided into* $|E|$ *sub-intervals delimited by* $t_0 = 0 < t_1 < \cdots < t_{|E|} = 1$,

such that in each sub-interval $[t_i, t_{i+1}]$, $0 \leq i \leq |E|$, *all vertices are stationary except for the two endpoint vertices of one edge, which move from their initial positions to their final positions.*

Note that this category of morphs is only definable on disjoint segments, because otherwise the edges may not move independently which is essential in the U1E morph definition. In U1E morphs, unlike U1V morphs, there are morphs that cannot be realized if the two endpoints of the moving edge must morph linearly. This is because different ways of morphing the endpoints of a single segment may cause different areas in the plane to be swept, and some of these may violate planarity. In this section we show how to test if there is a planar U1E morph for a set of disjoint segments.

First, we review the properties of each type of morphing quadrilateral which were discussed earlier in Section 7.1. See Figure 7.1. For all types of morphing quadrilaterals, all the area inside the quadrilateral will be swept regardless of how the endpoints of the segment morph. However, the segment may need extra area for some types of morphing quadrilaterals to morph in a planar way. For U1E morphs, it is enough to morph each segment with the *minimum area*. The properties of each type of morphing quadrilateral are summarized as follows:

- For a segment with *convex morphing quadrilateral* (Figure 7.1(a)), the area that is swept in a U1E morph is fixed regardless of how the segment endpoints morph.
- For a segment with *simple non-convex morphing quadrilateral* (Figure 7.1(b)), the minimum area that will be swept in all U1E morphing scenarios is the area inside the quadrilateral. There is a U1E morph that only uses this minimum area. This is where linear morphing does not always suffice and may use some extra area.
- In the case of a morphing quadrilateral with *crossing edges* (Figure 7.1(d)), a U1E morph will use the area inside the quadrilateral. However, only one specific morph uses only the inside area, namely, when the two endpoints of the segment move synchronously such that the segment always passes through the point where the initial and the final configurations cross. In this case also the linear morph may use extra area outside the morphing quadrilateral. See Figure 8.10 for an example.
- In the case of a morphing quadrilateral with *crossing morph lines* (Figure 7.1(c)) it again happens that any U1E morph will use the area inside the quadrilateral. However, to use only the inside area the two endpoints must meet at the crossing point of the morph lines and pass over each other which is not accepted as it violates

the planarity condition. We can avoid this scenario by letting the segment use some area outside the morphing quadrilateral as long as the extra area does not introduce any morphing dependency with other segments.

Morphing of each segment s interferes with only the other segments that intersect the morphing quadrilateral of s at their initial or final position. Thus, exiting the morphing quadrilateral does not change the solution as long as the extra area does not make any new intersection with other segments in their initial or final configurations.

For this type of morphing quadrilateral, we can always find a circle around the crossing point of the morph lines such that the circle is empty of any new initial or final segments that does not intersect the morphing quadrilateral already. Figure 8.11 shows the empty circle that will be used to morph a segment with crossing morph lines in a planar way.

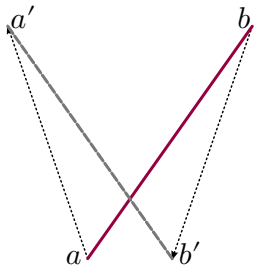


Figure 8.10: A segment with crossing edges in which the linear morph uses some area outside of the morphing quadrilateral.

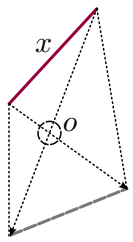


Figure 8.11: The small circle around crossing point o, such that it crosses the same set of segments as the morphing quadrilateral of x, guarantees a planar U1E morph for segment x. The morph will only use the morphing quadrilateral and some area in the small circle.

Now, we can compute how the morph of each segment depends on the morph of other segments based on the minimum area each segment needs to morph. This forms a dependency graph, G_d, that tells us which segment has to morph before some other segments:

There is a vertex s in G_d for each edge (segment) s in the morphing problem. There is a directed edge $e(s_1, s_2)$ in E_{G_d} if one of the following conditions happens:

- The initial configuration of segment s_1 overlaps with the minimum area that segment s_2 needs to morph
- The final configuration of segment s_2 overlaps with the minimum area that segment s_1 needs to morph

Theorem 8.13. *For a given set of disjoint segments, there is a planar U1E morph iff the dependency graph G_d is acyclic. Furthermore, there is an $O(n^2)$ to test this.*

Proof. $\Rightarrow$ First, we prove that if G_d is acyclic, then there is a planar U1E morph. The acyclic G_d has at least one source s, i.e., a vertex with no edges directed toward it. We can morph s first because there is no other segment in its initial position overlapping with the area s needs to morph, and also the final configuration of s is not overlapped by the minimum morph area of any other segment. This follows from the way G_d is defined. This procedure uses a topological ordering of the segments which guarantees that if the segments are morphed in this order in a U1E morph and each segment only uses the minimum area to morph, then planarity is preserved.

$\Leftarrow$ Second, we prove that if G_d is cyclic, there is no planar U1E morph for the set of disjoint segments. We can simply observe from the construction of G_d that if there is a directed edge (s_1, s_2), the segment s_1 should morph before the segment s_2 in any planar U1E morph because otherwise if s_1 morphs after s_2, either s_2 crosses the initial configuration of s_1 or s_1 crosses the final configuration of s_2. If G_d has a cycle, then in any U1E morph at least one edge of the cycle must go backward in the morphing order of the edges, i.e., at least one (s_1, s_2) edge appears in the wrong order which violates planarity.

The following algorithm will find a planar U1E morph for a given set of disjoint segments, or decide that there is no such morph:

- For each segment s compute the minimum morphing area that has the least overlap with other segments' initial and final configurations. ($O(n^2)$)
- Construct the dependency graph G_d. ($O(n^2)$)
- Test if G_d is acyclic. ($O(n + |E(G_d)|)$)
- If G_d is acyclic, find a topological order of the segments to morph. This is the order by which the U1E morph preserves planarity. ($O(n + |E(G_d)|)$)

- If G_d has a cycle, there is no planar U1E morph.

The most time consuming step of the algorithm is to construct G_d in which we need to test each pair of segments, in a constant time. Therefore, the algorithm takes time $O(n^2)$. □

8.2.1 U1E Morphs with Extra Conditions

In the previous section we showed that if each segment morphs using the minimum area, which is the area with the least overlap with other segments in their initial or final configurations, the problem of finding a planar U1E morph will be equivalent to finding a topological order of the segments based on the dependency graph. We show in this section how adding some extra conditions to the morphing of each segment in U1E morphs affects the complexity of the problem. We consider the U1E linear morphs, U1E 1V morphs, and U1E U1V morphs. While the first two variants of U1E morphs can be treated with the same technique of finding the minimum area, we prove that it is NP-hard to decide if there is a U1E U1V morph that preserves planarity for a given set of disjoint segments. It is especially interesting that going from U1E 1V morphs to U1E U1V morphs, the complexity of the problem changes from $O(n^2)$ to exponential.

Definition 8.14 (U1E Linear Morphing). *An SL-morph is* U1E Linear *when it is U1E according to Definition 8.12 and also the endpoints of each segment move with uniform speed in the segment's time interval, i.e., with a linear morph.*

In U1E linear morphs, the area that each segment needs to morph is not the same as the minimum area we had for U1E morphs. However, the area that each segment sweeps is deterministic and can be computed in constant time for each segment. Therefore, the algorithm for finding a planar U1E linear morph will be as follows:

- For each segment s compute the area that segment s sweeps in a linear morph. ($O(n)$)
- Construct the dependency graph G_d. ($O(n^2)$)
- Test if G_d is acyclic. ($O(n + |E(G_d)|)$)
- If G_d is acyclic, then find a topological order of the segments to morph. This is the order by which the U1E linear morph preserves planarity. ($O(n + |E(G_d)|)$)

- If G_d has a cycle, there is no planar U1E linear morph.

The complexity of the algorithm for planar U1E linear morph is still $O(n^2)$.

Definition 8.15 (U1E 1V Morph). *A* U1E 1V *morph is a U1E morph when the morph of each segment is also 1V, i.e., only one endpoint moves at a time. Note that the two endpoints may move alternately in multiple steps.*

In U1E 1V morphs we may also define the minimum morph area each segment needs. The minimum morph area is the same as for the U1E morph problem, except for the case of a crossing-edges morphing quadrilateral. In this case the minimum morphing area in U1E is achieved when both endpoints of the edge move together. For U1E 1V morphs we may need a little extra area when the two endpoints of the segments take turns and move a little bit in each turn. The steps must be small enough such that it causes no extra dependency edge in the dependency graph G_d. This is possible when the input has no degeneracies and can be computed in $O(n)$ for each segment. Thus, the algorithm that decides if there is a planar U1E 1V morph for a set of disjoint segments will be the same as the U1E morph case when the minimum morph area for each segment is adjusted for 1V for segments with crossing-edges morphing quadrilaterals. Comparing to the two previous cases, this planar U1E 1V morph may lead to many steps though.

Definition 8.16 (U1E U1V Morph). *A* U1E U1V *morph is a U1E 1V morph in which each vertex of the edge morphs at once without interruption.*

In U1E U1V morphs, unlike the previous variations of the U1E morphs, we may not define a minimum area for each segment to use in the morph. The segments may use different areas in two different orderings of their endpoints in U1V morphs. This causes the problem to be harder. In the following section we show that going from U1E 1V morphs to U1E U1V morphs, increases the complexity of the problem from polynomial to NP-complete.

8.2.2 Planar U1E U1V Morphing is NP-Complete

U1E U1V morphing is a more restricted version of U1V morphing, so finding a planar U1E U1V morph is in NP. In Section 8.1, we show that finding a planar U1V morph for a set of disjoint segments is NP-Complete. In this section, we prove that finding a planar U1E U1V morph, unlike other versions of U1E morphs, is NP-complete even for a

set of disjoint segments. Note that in U1E U1V morphing, as in other U1E morphs, any morphing solution is an ordering of the edges. Moreover, in the morphing of each segment we must decide in which order the endpoints of the segment morph.

Theorem 8.17. *For a set of disjoint segments, finding a U1E U1V morph that preserves planarity is NP-Complete.*

As in the two other previous NP-hardness proofs for PSL-morphing and planar U1V morphing, the reduction is from the planar monotone 3-SAT problem. See Figure 7.5 for an example of a planar monotone 3-SAT drawing. Now, we show how we construct the variable gadgets, clause gadgets and the wires connecting each variable to all its appearances in the clauses in U1E U1V morphing.

Boolean Gadget

Similar to U1V morphing, a single segment with crossing morph lines works as a Boolean gadget in U1E U1V morphing because it may morph in two different possible ways that need different areas to be swept. If the left endpoint of the segment morphs first, the morphing of the left endpoint sweeps the right triangle $\triangle oba'$ and the segment lies on the right-side of the morphing quadrilateral in the intermediate state. Then, we say the segment morphs through its right-side area and map it to the "true" Boolean value. Otherwise, if the right endpoint of the segment morphs first, symmetrically, the segment morphs through its left-side area, and we map it to the "false" Boolean value; see Figure 8.12. A single Boolean segment will be used as a Boolean variable in the construction.

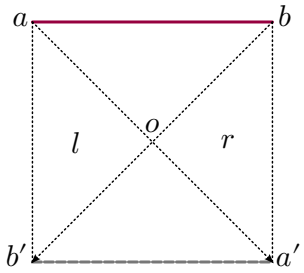


Figure 8.12: A Boolean variable in U1E U1V morph: a single segment with crossing morph lines

Clause Gadget

The gadget shown in Figure 8.13 works as an OR gadget in the planar U1E U1V morph problem. This gadget is different from the OR gadget for U1V morph, because here both endpoints of each segment must morph immediately after each other without interruption. The following Lemma 8.18 and Lemma 8.19 explain why the gadget shown in Figure 8.13 is an OR gadget.

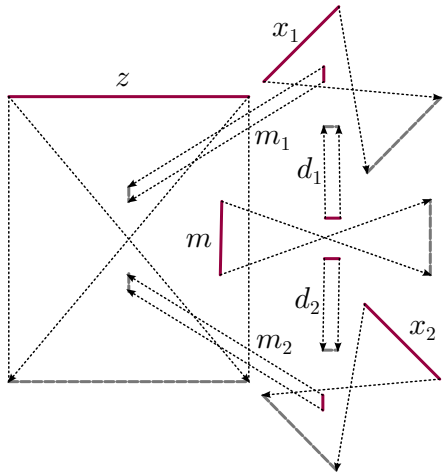


Figure 8.13: U1E U1V morph OR gadget: In any planar U1E U1V morph if Boolean segment z is true, then $x_1 \vee x_2$ is true.

Lemma 8.18. *In the gadget shown in Figure 8.13, if the Boolean gadget (segment) z has true value (morphs through its right-side), then at least one of the Boolean gadgets x_1 and x_2 has morphed through its right-side.*

Proof. First note that in any planar U1E U1V morph of this gadget, segment m_1 and m_2 morph after segment z and before segments x_1 and x_2 respectively. That is because regardless of the true/false (right/left) valuation of the Boolean segments x_1 and x_2, segments m_1 and m_2 in their initial configuration conflict with the planar morph of x_1 and x_2 respectively. If we denote this partial ordering with "$<$" symbol, we have $z < m_1 < x_1$ and $z < m_2 < x_2$. Also, if Boolean segment z morphs through the right side, we will have $m < z$ with the same reasoning. Boolean segment m morphs either through its top or bottom, so we will have $d_1 < m$ or $d_2 < m$. Thus, either $d_1 < m < z < x_1$ or $d_2 < m < z < x_2$ is true in a planar U1E U1V morph. If the first one holds ($d_1 < x_1$), it means that when x_1 is going to morph its left side is occupied with segment d_1 in its final configuration.

Therefore, x_1 may only morph through the right side. It is symmetric for d_2 and x_2, so at least one of the x_1 and x_2 morphs through the right side. □

Lemma 8.19. *In the gadget shown in Figure 8.13, any true/false valuation of Boolean gadgets x_1 and x_2 that satisfies $x_1 \vee x_2$, can be a part of a planar U1E U1V morph when the Boolean segment z has true value (morphs through its right-side).*

Proof. In the valuation that satisfies $x_1 \vee x_2$, if x_1 has true value we can morph segment d_1 first. Symmetrically, if x_2 has true value we can morph d_2 first. Now Boolean segment m has at least one of the top or bottom areas free to morph (depending on which d_1 or d_2 or both have morphed already). After segment m morphs, segment z is free to morph through the right side. Then, both segments m_1 and m_2 morph. Now, segments x_1 and x_2 morph according to their valuation. If one of the Boolean segments x_1 and x_2 has to morph through the left side, the corresponding d_1 or d_2 segments has not morphed yet and will morph last. □

In Figure 8.14 we show how to construct a clause gadget by cascading two of the OR gadgets. The following Lemmas 8.20 and 8.21 prove that this clause gadget works.

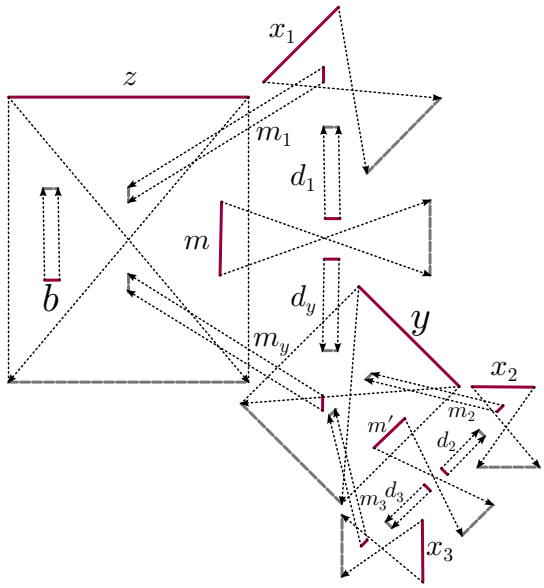


Figure 8.14: U1E U1V morph clause gadget: In any planar U1E U1V morph $x_1 \vee x_2 \vee x_3$ is always true.

Lemma 8.20. *In any planar U1E U1V morph of the gadget shown in Figure 8.14, at least one of the Boolean gadgets x_1 , x_2 , and x_3 morphs through its right-side (takes true value).*

Proof. The clause gadget shown in Figure 8.14 is a cascade of two OR gadgets and a little segment b added in the left triangle of Boolean segment z. Boolean segment z may only morph through the right side, because the segment b blocks its left side area permanently. Then, one of the Boolean segments x_1 and y must morph through the right side as we proved in Lemma 8.18. If y has true value with the same reasoning at least one of the Boolean segments x_1 and x_2 morphs through the right side. □

Lemma 8.21. *In the gadget shown in Figure 8.14, any valuation that satisfies $x_1 \vee x_2 \vee x_3$ can be part of a planar U1E U1V morph for the entire gadget.*

Proof. If in the given valuation, $y = x_2 \vee x_3$ is true, then a planar U1E U1V morph is to morph z, x_1 and y according to the scenario shown in the proof of Lemma 8.19. In the suggested scenario in Lemma 8.19 when y (x_2 in Lemma 8.19) morphs, m_y has already morphed, so the OR gadget that consists of y, x_2 and x_3 will be clear of extra segments and with the same scenario the entire OR gadget may successfully morph. Otherwise, if y is false (both x_2 and x_3 are false), x_1 is true. Therefore, d_1 may morph first. Then, the segment m morphs through its top side. Then, after the morph of Boolean segment z, segments m_1 and m_y morph. After the morph of m_1, Boolean segment x_1 is free to morph through the right side. After the morph of m_y, Boolean segment y is free to morph through the left side. Then, segments m_2 and m_3 may morph, and enable Boolean segments x_2 and x_3 to morph through either side (actually left side because we assumed y is false). Finally, segments d_2, d_3, and m' morph. □

In the next section, we show how the clause gadgets are connected to their variables and what the entire construction looks like. To do that we need to show how to copy the valuation of a variable and spread it into all its appearances in the clauses using wire gadgets.

Copy and Wire Gadgets

Figure 8.15 shows an AND gadget that will be used as the core of the variable gadget in planar U1E U1V morphing to copy the valuation of each variable in a 3-SAT problem to all its appearances in the clauses. Lemma 8.22 explains how this gadget works.

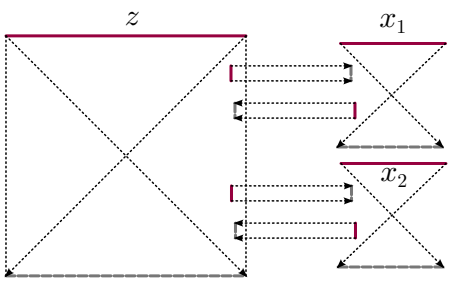


Figure 8.15: AND-gate in a planar U1E U1V morph that is used as the core of variable gadgets in the construction (in rotated form) and copies the true value of a Boolean segment z into all Boolean segments x_i.

Lemma 8.22. *In the gadget shown in Figure 8.15, if segment z is true (morphs through its right side) in a planar U1E U1V morph, then all variables x_i (x_1 and x_2 in this figure) are true (morph through the right side).*

Proof. Boolean segment z morphs through the right side, so to preserve planarity the top segments between z and the x_i's must morph before z. These segments will block the left side of the corresponding x_i's, so all x_i's must morph through their right sides. The bottom segments between the x_i's and z prevents x_i's from morphing before segment z. If a segment x_i morphs through the left side before segment z morphs, the bottom segment should have morphed and blocked the right side of segment z. □

The two segments connecting each Boolean segment x_i to the Boolean segment z in the AND-gate can be replaced by the chain of segments (wire gadget) shown in Figure 8.16(a). Using this wire gadget, a Boolean segment can be copied and propagated to different areas of the plane where the clauses are located.

In the chain of segments shown in Figure 8.16(a), each element of the chain morphs after the next element, so when segment x morphs, segment y has already morphed. Similar to the wire gadgets in previous sections, the wire gadget may bend or pass through narrower parts by adjusting the element angles and sizes. It is enough to make the initial configuration of each element of the chain overlap with the morphing quadrilateral of the previous element of the chain. Figure 8.16(b) shows how the wire gadget is symbolized for simplicity.

Now we have all the materials to construct a morph instance corresponding to an instance of the monotone planar 3-SAT problem. Each variable v_i in this construction is a rotated Boolean segment which is connected from the top side to all the clauses with v_i as

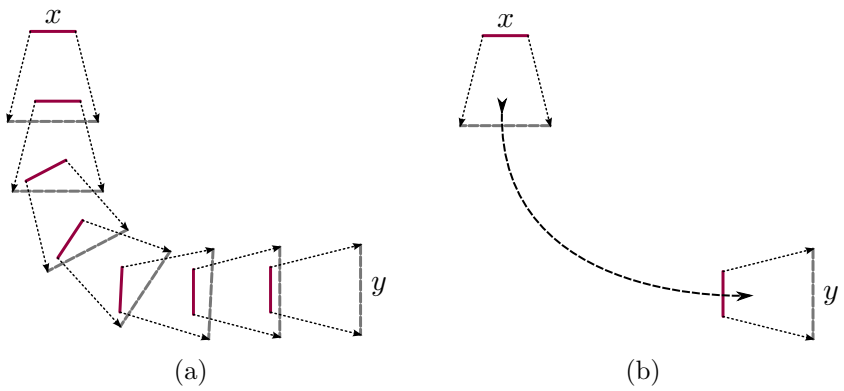


Figure 8.16: A chain of segments works as a wire. The last element y always morphs before the first element x in a planar U1E U1V morph.

a literal using AND-gates and from the bottom side to all the clauses with $\neg v_i$ as a literal. See Figure 8.17 for an example.

Proof of Theorem 8.17

In this section we complete the reduction from the monotone planar 3-SAT problem to the planar U1E U1V morphing problem by showing that any solution to an instance of the monotone planar 3-SAT problem is equivalent to an instance of the planar U1E U1V morph problem and vice versa. For an instance I of the monotone 3-SAT problem let M denote the morphing instance as constructed above. We prove through the following two lemmas that I is satisfiable if and only if M has a U1E U1V morph that preserves planarity.

Lemma 8.23. *If M has a planar U1E U1V morph, then I is satisfiable.*

Proof. In a planar U1E U1V morph of M, each Boolean segment v_i morphs either through the top or the bottom side. We construct a valuation V of the variables based on how v_i morphs. For each v_i, assign v_i true value if the Boolean segment v_i morphs through the bottom side. Otherwise, assign v_i false value. We then prove that the valuation V satisfies I.

According to Lemma 8.20 all the clauses are satisfied in a planar U1E U1V morph. Now we only need to show that the values of the literals in all clauses are compatible with the valuation V. The pair of wires that connects each literal in the top (positive)

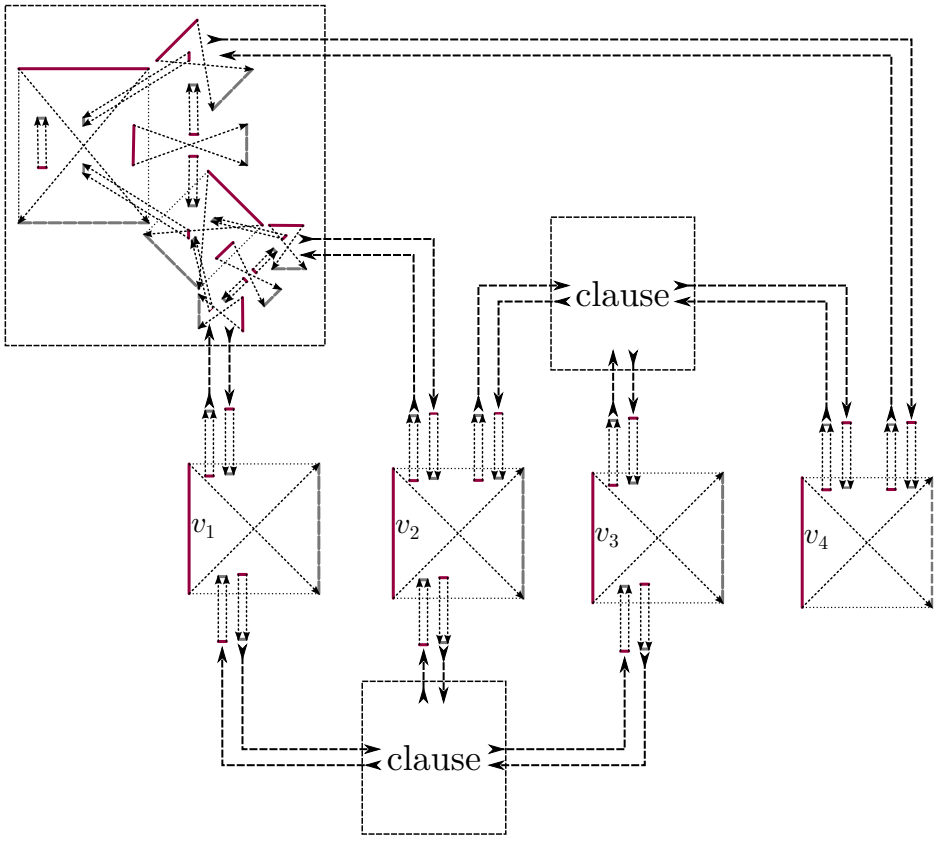


Figure 8.17: The full construction of a monotone planar 3-SAT instance $(v_1 \vee v_2 \vee v_4) \wedge (v_2 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3)$. The first clause is shown with all details and the others are shown with a box.

clauses to the corresponding variable guarantees by Lemma 8.22 that if the literal in the clause is true, the corresponding variable morphs through the bottom side and has true value in valuation V. This also holds for the false variables symmetrically. The satisfied literals in the bottom (negative) clauses force the corresponding variable gadgets to morph through the top side. Therefore, a planar U1E U1V morph of this construction represents a valuation that satisfies I. □

Lemma 8.24. *If I is satisfiable, then M has a planar U1E U1V morph.*

Proof. Assume that a valuation V satisfies I. For each Boolean segment v_i in M if the variable v_i in V is true we morph the Boolean segment v_i through the bottom side. Otherwise, if the variable v_i in V is false, we morph the Boolean segment v_i through the top side.

In order to do that, the bottom side of all true v_i's and the top side of all false v_i's must be cleared first. Thus, one of each such pairs of the wires connecting variables and the literals in the clauses must morph and the other wire in the pair must wait. Now all the Boolean segments v_i morph according to the valuation, and the other wires in all pairs of wires can morph now. The wires that morphed in the first round block the right side of the negation of corresponding literals, preventing them from having true values in the clauses. However, there still exists a planar U1E U1V morph for M because all clauses have at least one true literal and according to Lemma 8.21 there is a planar U1E U1V morph for the clause corresponding to any satisfying valuation of the literals. After morphing all the clauses (or any time in the middle of that if needed) we can morph the remaining wires. □

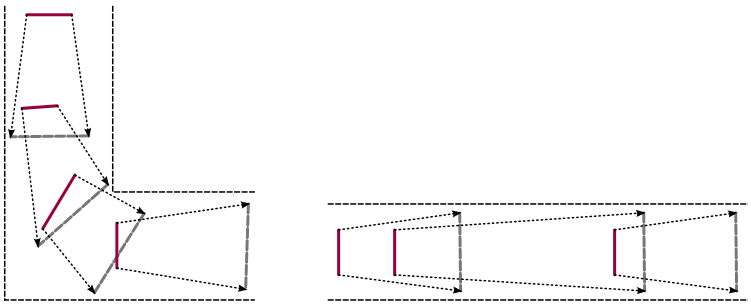
Lemma 8.25 will complete the proof of Theorem 8.17 by showing that the reduction can be done in polynomial time. The proof is similar to the case of planar U1V morphing in Lemma 8.11.

Lemma 8.25. *The construction of the planar SL-morphing instance M from a planar monotone 3-SAT instance I takes polynomial time.*

Proof. The planar SL-morphing instance M consists of clause gadgets, variable gadgets, and wire gadgets connecting the variables to all their appearances in clauses. Similar to the case of planar SL-morph, the complexity of drawing each clause gadget is constant and it can lie on a constant-size grid; say a $k \times k$ grid. Each variable gadget may have $O(n)$ segments (depending on the number of clauses that have the variable) and can lie on a $k \times nk$ grid.

The number of segments in wire gadget in planar U1E U1V morphing only depends on the number of bends, because the number of segments in each bend and in each straight part of the wire are constant. See Figure 8.18.

Now we take a rectilinear representation of the planar monotone 3-SAT instance I as shown in Figure 7.5. Observe that this drawing lies on an $O(n) \times O(n)$ grid where n is the number of clauses and variables in I. We can expand this grid by a factor of nk and thicken every edge to a constant width strip. Now we replace each variable and clause node with the corresponding gadget and replace each edge by a wire made of switches. Note that each edge has $O(1)$ bends in the original rectilinear representation and so the number of segments needed for each wire is $O(1)$. □



(a) (b)

Figure 8.18: (a) U1E U1V wire gadget in bends; (b) U1E U1V wire gadget in straight parts

Chapter 9

Straight-Line Morphing: Conclusions and Open Problems

Straight-line morphing as defined in this thesis can model non-rigid transformation, as well as all translations and some rotations in 2D space. However, like any other continuous planar morphing in 2D, straight-line morphing may not perform mirroring. Our research on this problem, which includes studying different settings of the problem and also looking at some simplified versions, shows that testing the existence of a planar straight-line morph is a hard problem.

There are many related open problems on straight-line morphing. Some are as follows:

1. How hard is the planar straight-line morphing problem for disjoint segments, cycles, or paths? We proved that the planar straight line morphing problem for disjoint segments is NP-hard. However, we still do not know if it is in NP or maybe belongs to a harder class of problems such as PSPACE-hard.

2. How hard is the convex morphing problem (the simplified version of the problem when all morphing quadrilaterals are convex) for disjoint segments, cycles, or paths? For the case of cycles and paths we conjecture that the necessary condition for convex morphing discussed in Section 7.4.1 is sufficient; see Conjecture 7.17. However, it is not proved yet.

3. We showed in this thesis that some discrete versions of straight-line morphing are NP-complete for a set of disjoint segments. However, we do not know if they can be solved in polynomial time for cycles or paths.

4. How hard is 1V SL-morphing? Among the simplified versions of SL-morphing we still do not know anything about the version when only one vertex moves at a time. We conjecture that 1V SL-morphing is equivalent to the original SL-morphing because if we do not have degeneracy in the input we can always take tiny 1V steps.

5. Are SL-morphs equivalent to piece-wise linear morphs? We conjecture that they are equivalent; see Conjecture 6.6. In other words, if there is a planar SL-morph there always exists a piece-wise linear planar SL-morph.

We can also go forward and define new versions of morphing. One direction is to make straight-line morphing more visualizable. Sometimes it is desirable not only to preserve planarity, but also to preserve the global structure of the graph as much as possible. We may formalize this parameter as the maximum distance from each configuration during the morph to the most similar configuration that occurs during a linear morph. We can then try to minimize this distance through the entire morph.

Although vertex/edge coincidence is not allowed in the SL-morphing definition, the result might not satisfactorily preserve the global structure of the graph because of vertices/edges that move very close together and form sharp angles. Then, another criterion would be to prohibit vertices/edges from moving close to each other, thus preventing the formation of sharp angles.

Another direction is to achieve a nice morph by allowing a few (constantly many) bends in the vertex trajectories, allowing a few crossings per segment through the morph, or letting vertices move a bit out of their standard trajectories. These variations might make the problem easier in two ways. First, it might be easier to develop an efficient algorithm with these relaxations. Second, it leads to a more powerful morphing definition that can model more transformations of the graphs.

References

[1] O. Aichholzer, G. Aloupis, E. D. Demaine, M. L. Demaine, V. Dujmovic, F. Hurtado, A. Lubiw, G. Rote, A. Schulz, D. L. Souvaine, and A. Winslow. Convexifying polygons without losing visibilities. In *Canadian Conference on Computational Geometry (CCCG)*, pages 229–234, 2011.

[2] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[3] S. Alamdari, P. Angelini, F. Barrera-Cruz, T. M Chan, G. Da Lozzo, G. Di Battista, F. Frati, P. Haxell, A. Lubiw, M. Patrignani, et al. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.

[4] S. Alamdari, P. Angelini, T. M. Chan, G. Di Battista, F. Frati, A. Lubiw, M. Patrignani, V. Roselli, S. Singla, and B. T. Wilkinson. Morphing planar graph drawings with a polynomial number of steps. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1656–1667, 2013.

[5] B. Alspach. Searching and sweeping graphs: a brief survey. *Le Matematiche*, 59:5–37, 2004.

[6] P. Angelini, F. Frati, M. Patrignani, and V. Roselli. Morphing planar graphs drawings efficiently. In *International Symposium on Graph Drawing*, pages 49–60. Springer, Cham, 2013.

[7] P. Angelini, G. Da Lozzo, F. Frati, A. Lubiw, M. Patrignani, and V. Roselli. Optimal Morphs of Convex Drawings. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 126–140, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] W. Baird and A. Bonato. Meyniel's conjecture on the cop number: a survey. *Journal of Combinatorics*, 3:225–238, 2012.

[9] F. Barrera-Cruz, P. E. Haxell, and A. Lubiw. Morphing planar graph drawings with unidirectional moves. In *Mexican Conference on Discrete Mathematics and Computational Geometry*, pages 57–65, Oaxaca, Mexico, 2013.

[10] A. Beveridge and Y. Cai. Pursuit-evasion in a two-dimensional domain. *Ars Mathematica Contemporanea*, 13(1):187–206, 2017.

[11] A. Beveridge, A. Dudek, A. M. Frieze, and T. Müller. Cops and robbers on geometric graphs. *Combinatorics, Probability & Computing*, 21(6):816–834, 2012.

[12] D. Bhadauria, K. Klein, V. Isler, and S. Suri. Capturing an evader in polygonal environments with obstacles: The full visibility case. *The International Journal of Robotics Research*, 31(10):1176–1189, 2012.

[13] S. Bhattacharya, G. Paul, and S. Sanyal. A cops and robber game in multidimensional grids. *Discrete Applied Mathematics*, 158(16):1745–1751, August 2010.

[14] T. Biedl, A. Lubiw, and M. J Spriggs. Morphing planar graphs while preserving edge directions. In *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2006.

[15] A. Bonato, P. A. Golovach, G. Hahn, and J. Kratochvíl. The capture time of a graph. *Discrete Mathematics*, 309(18):5588–5595, 2009.

[16] A. Bonato and R. J. Nowakowski. *The Game of Cops and Robbers on Graphs*. American Mathematical Society, 2011.

[17] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry Theory and Applications*, 23(3):313–335, 2002.

[18] S. Brandt, Y. Emek, J. Uitto, and R. Wattenhofer. A tight lower bound for the capture time of the cops and robbers game. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[19] J. Branke. Dynamic graph drawing. In *Drawing Graphs*, volume 2025 of *Lecture Notes in Computer Science*, pages 228–246. Springer Berlin Heidelberg, 2001.

[20] G. R. Brightwell and P. Winkler. Gibbs measures and dismantlable graphs. *Journal of Compinatorial Theory (Series B)*, 78:141–166, 1999.

[21] S. S. Cairns. Deformations of plane rectilinear complexes. *The American Mathematical Monthly*, 51(5):247–252, 1944.

[22] J. H. Cantarella, E. D. Demaine, H. N. Iben, and J. F. O'Brien. An energy-driven approach to linkage unfolding. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 134–143. ACM, 2004.

[23] J. Chalopin, V. Chepoi, N. Nisse, and Y. Vaxès. Cop and robber games when the robber can hide and ride. *SIAM Journal of Discrete Mathematics*, 25(1):333–359, 2011.

[24] D. Z. Chen, J. Hershberger, and H. Wang. Computing shortest paths amid convex pseudodisks. *SIAM Journal of Computing*, pages 1158–1184, 2013.

[25] D. Z. Chen and H. Wang. Computing shortest paths among curved obstacles in the plane. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 369–378, New York, NY, USA, 2013. ACM.

[26] E. Chiniforooshan. A better bound for the cop number of general graphs. *Journal of Graph Theory*, 58(1):45–48, 2008.

[27] T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31:299–316, 2011.

[28] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational Geometry: Introduction*. Springer, 2008.

[29] M. De Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry and Applications*, 22(03):187–205, 2012.

[30] E. D. Demaine and J. ORourke. *Geometric Folding Algorithms*. Cambridge University Press, 2007.

[31] D. P. Dobkin and D. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(1-4):421–457, 1990.

[32] A. Efrat, L. J. Guibas, S. Har-Peled, J. S. B. Mitchell, and T. M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28(4):535–569, 2002.

[33] A. Efrat, S. Har-peled, L. J. Guibas, and T. M. Murali. Morphing between polylines. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 680–689, 2000.

[34] H. Emadi, T. Gao, and S. Bhattacharya. Partitioning strategies and task allocation for target-tracking with multiple guards in polygonal environments. *arXiv preprint arXiv:1611.05092*, 2016.

[35] F. V. Fomin, P. A. Golovach, and J. Kratochvíl. On tractability of cops and robbers game. In *Fifth IFIP International Conference On Theoretical Computer Science*, pages 171–185, 2008.

[36] F. V. Fomin, P. A. Golovach, and D. Lokshtanov. Cops and robber game without recharging. In *Proceedings of the 12th Scandinavian Conference on Algorithm Theory*, SWAT'10, pages 273–284, Berlin, Heidelberg, 2010. Springer-Verlag.

[37] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

[38] P. Frankl. Cops and robbers in graphs with large girth and Cayley graphs. *Discrete Applied Mathematics*, 17(3):301–305, 1987.

[39] C. Friedrich and P. Eades. Graph drawing in motion. *Journal of Graph Algorithms and Applications*, 6(3):353–370, 2002.

[40] A. M. Frieze, M. Krivelevich, and P. Loh. Variations on cops and robbers. *Journal of Graph Theory*, 69(4):383–402, 2012.

[41] T. Gavenčiak. Cop-win graphs with maximum capture-time. *Discrete Mathematics*, 310(10):1557–1563, 2010.

[42] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[43] S. K. Ghosh and P. P. Goswami. Unsolved problems in visibility graphs of points, segments and polygons. *ACM Computing Surveys (CSUR)*, 46(2):22, 2013.

[44] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[45] A. S. Goldstein and E. M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science*, 143(1):93–112, 1995.

[46] L. J. Guibas, J. Latombe, S. M. Lavalle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry & Applications*, 09(04n05):471–493, 1999.

[47] G. Hahn. Cops, robbers and graphs. *Tatra Mountains Mathematical Publications*, 36:163–176, 2007.

[48] G. Hahn, F. Laviolette, N. Sauer, and R. E. Woodrow. On cop-win graphs. *Discrete Mathematics*, 258(1-3):27–41, 2002.

[49] G. Hahn and G. MacGillivray. A note on k-cop, l-robber games on graphs. *Discrete Mathematics*, 306(19-20):2492–2497, 2006.

[50] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.

[51] J. Hershberger and L. J Guibas. An $O(n^2)$ shortest path algorithm for a non-rotating convex body. *Journal of Algorithms*, 9(1):18–46, 1988.

[52] J. Hershberger, S. Suri, and H. Yildiz. A near-optimal algorithm for shortest paths among curved obstacles in the plane. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 359–368, New York, NY, USA, 2013. ACM.

[53] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884, 2005.

[54] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with local visibility. *SIAM Journal of Discrete Mathematics*, 20(1):26–41, 2006.

[55] V. Isler and N. Karnad. The role of information in the cop-robber game. *Theoretical Computer Science*, 3(399):179–190, 2008.

[56] K. Klein. *Geometric Pursuit Evasion*. PhD thesis, University of California, Santa Barbara, 2014.

[57] K. Klein and S. Suri. Catch me if you can: Pursuit and capture in polygonal environments with obstacles. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[58] K. Klein and S. Suri. Capture bounds for visibility-based pursuit evasion. *Computational Geometry*, 48(3):205–220, 2015.

[59] K. Klein and S. Suri. Pursuit evasion on polyhedral surfaces. *Algorithmica*, 73(4):730–747, 2015.

[60] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

[61] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[62] S. M. LaValle and J. E. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. *IEEE Transactions on Robotics and Automation*, 17(2):196–202, 2001.

[63] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[64] J. E. Littlewood. *Littlewood's Miscellany*. Cambridge University Press, 1986.

[65] A. Lubiw, J. Snoeyink, and H. Vosoughpour. Visibility graphs, dismantlability, and the cops and robbers game. *Computational Geometry*, 66:14–27, 2017.

[66] E. Melissaratos and D. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992.

[67] M. Nöllenburg, D. Merrick, A. Wolff, and M. Benkert. Morphing polylines: A step towards continuous generalization. *Computers, Environment and Urban Systems*, 32(4):248–260, 2008.

[68] R. J. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[69] J. C. Onofre, S. Rajsbaum, and M. Raynal. A topological perspective of recursion in distributed computing. In *Mexican Conference on Discrete Mathematics and Computational Geometry*, 2013.

[70] P. Pisantechakool and X. Tan. On the capture time of cops and robbers game on a planar graph. In *International Conference on Combinatorial Optimization and Applications*, pages 3–17. Springer, 2016.

[71] S. Pomm, C.and Werlen. Smooth morphing of handwritten text. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 328–335, New York, NY, USA, 2004. ACM.

[72] A. Quilliot. Homomorphismes, points fixes dans les graphes, les ensembles ordonnés et les espaces métriques. *Ph.D. Thesis, These de Doctorat d'Etat, Université de Paris VI*, 1983.

[73] P. D Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.

[74] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[75] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal of Computing*, 21(5):863–888, 1992.

[76] C. Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983.