Department of Mathematics and Computer Science
Algorithms

Finding Compatible Cycles in Convex Trees

Bachelor research project

Jan van Baast
David Nistor
Malcolm Vassallo

April 2024

Supervision:
B. Speckmann

Credits: 10

This is a public Bachelor research project.

This Bachelor research project has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Abstract

An interesting problem in computational geometry is that of the compatibility of graphs with a planar straight-line embedding. We consider a tree with its leaves on the convex hull of its vertices and pose the question of whether such a tree has a compatible cycle. This problem is of theoretical interest, but not yet explored thoroughly. This report intents to build intuition about the problem and explores several viable approaches to solving it. We look at a theoretical approach that uses existing theory on Hamiltonian triangulations; we look at an algorithmic approach of iteratively forming such compatible cycles and we prove the existence of a compatible cycle in a simplified scenario. The results of this report build a basis for further research to continue solving this problem.

Contents

Chapter 1

Introduction

1.1 Motivation

Trees and graphs are useful tools for visualizing data, this could either be personal data such as that found in social networks or even geographical data of locations. Such visualizations can be used to study the relationships between items in the set being considered. In the same way, reshaping such visualizations in different ways can also bring new insights or an easier way of considering the sets.

One way of reshaping such a visualization is using compatible graphs. Two graphs are compatible when the edges of the two graphs together do not intersect in the plane. The lack of intersecting edges then makes visualizations easier to make.

1.2 Problem description

Finding graphs that are compatible with each other turns out to be a challenge in certain cases. Moreover, there are many different configurations of graphs for the problem can be solved. In this report, we will focus on finding cycles that are compatible with so-called convex trees. Specifically, we ask the question: "Given a tree graph embedded in the plane, with its leaves on the convex hull, is it possible to find a compatible cycle going through all its vertices?" These terms will be more formally defined in Chapter 2.

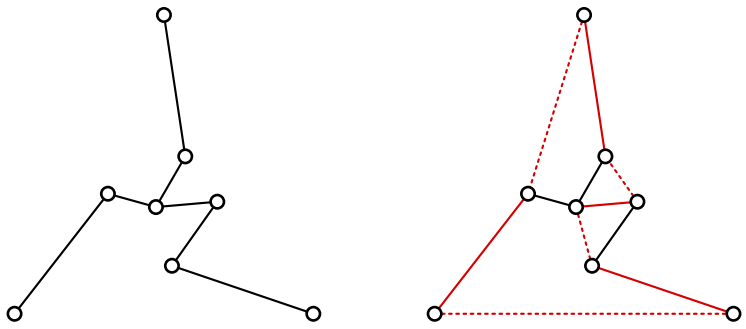


Figure 1.1: A convex tree and a cycle compatible with it (denoted in red)

1.3 Contributions and organization

We took many different approaches to approach the problem from several different angles. Each different approach led to interesting outcomes and realizations that would aid in the analysis of the research goal.

We start in Chapter 3 by exploring the problem and uncovering initial insights and topics of interest within the research. In Chapter 4, we consider the different ways in which a convex tree can be triangulated. We then discuss what properties such triangulations may have that help to provide

a Hamiltonian cycle compatible with the convex tree. In Chapter 5, instead of considering a single Hamiltonian cycle within the tree, we consider multiple cycles that together partition the vertices of the tree. We then attempt to connect these cycles to create a single Hamiltonian cycle. We also discuss an algorithm for constructing such cycles. Finally in Chapter 6, we further formalize the concept of 'visibility' between vertices and edges and we explore the possibility of extending a simple given cycle to include other vertices remaining in the tree.

1.4 Related work

The notion of graph compatibility can be found in various areas of research. Examples include compatibility of spanning trees [2] [3] or the compatibility of different graph matchings [1] [4]. Although this work is not directly related to this report, it can still help to get an intuition about graph compatibility.

More directly related work can be found in the Master's Thesis of M. van Garderen [7] where possible representations of trees as paths are discussed. This work also considers trees as visualizations of data and how they can be transformed. Within this work, we also find a counterexample to a very similar problem, where a compatible Hamiltonian cycle cannot exist for a non-convex tree. A non-convex tree being a tree where its leaves do not necessarily lie on the convex hull of its vertices. This counterexample is discussed in Chapter 4.

Chapter 2

Preliminaries

Before we start describing our results, we will outline several definition and propositions that we use.

Convexness. We start with the concept of convexness, a set of points S is called *convex* if for any two points in the set, the line between the points is also contained in the set. Using this, we can also define what a convex hull is. The *convex hull* of a set of points S is the smallest convex shape that contains all the points in S. Convexness and the convex hulls turn out to be very useful and are at the center of the most of our results.

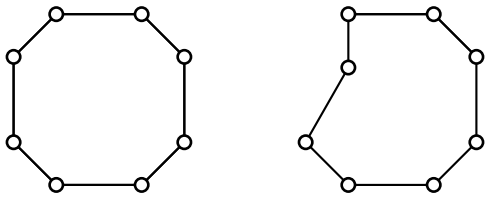


Figure 2.1: A convex and a non-convex shape

Graph types. Next, we define the types of graphs that we work with. The graphs in this report always have a planar straight-line embedding. Moreover, we define a *convex tree* to be a tree where all the leaves of the tree are on the convex hull of the vertices. We will specifically focus on triangular convex trees. A *triangular convex tree* is a convex tree that consist of only three branches connected together at one central vertex. The convex hull then only consists of the triangle formed by the three leaf vertices.

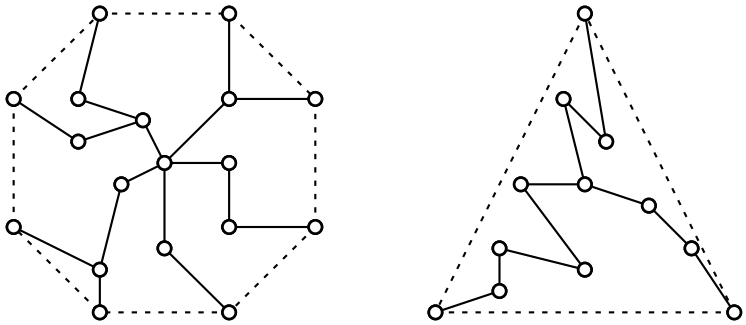


Figure 2.2: A convex tree and a triangular convex tree

Graph compatibility. Another important concept is that of graph compatibility. Consider two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ that share a vertex set. Remember that G_1 and G_2 both have a planar straight-line embedding. We say that the G_1 and G_2 are *compatible* with each other if the union $G^* = (V, E_1 \cup E_2)$ also has a valid planar straight-line embedding. In other words, G_1 and G_2 are compatible if the edges in $E_1 \cup E_2$ do not intersect each other.

It is also possible to refer to an edge or a subgraph to be compatible with a graph. In this case, the subgraph's vertex set is a subset of the graph's vertex set, but we still consider the complete edge set to determine compatibility.

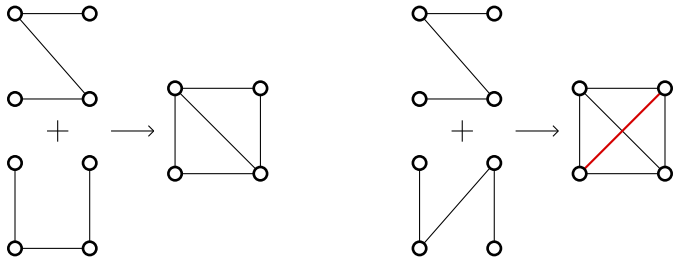


Figure 2.3: Two compatible and two non-compatible graphs

Chapter 3

Exploration

3.1 Problem description and reduction

In this report we work on a problem that was reduced from the more generalized problem. The initial question that was posed is that of: "Given an arbitrary convex tree, is it always possible to find a compatible cycle consisting of the entire vertex set?". This problem has a lot of different sides, we decided that we wanted to first focus on the convexness aspect. To this end, we reduced the problem to: "Given an arbitrary triangular convex tree, is it always possible to find a compatible cycle consisting of the entire vertex set?". This preserves the convexness aspect of the problem, whilst also simplifying the complexity of the tree as now we need to worry about only three branches. This is also the problem that we worked on for most of the project. As a last simplification, we assume throughout the entire report that no three vertices of the tree are co-linear. This is a very minor but necessary assumption as it makes certain arguments easier to make.

3.2 Initial ideas and observations

Before we start analyzing the problem using more structured approaches, we first list some properties of the problem and some approaches that we did to get intuition about the problem.

Polygons. One advantage of dealing with convex trees is that the tree divides its convex hull into simple polygons which all also have exactly one edge on the convex hull. In the case of triangular convex trees, this gives us three simple polygons to work with. This not only gives us the extra properties that simple polygons have (closed faces, connectedness, etc.), but also gives us a different angle to look at the problem.

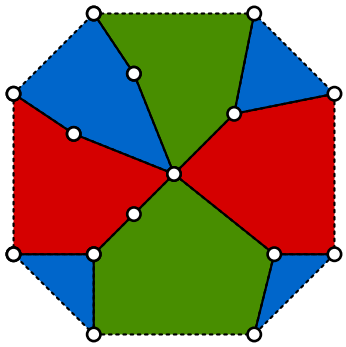


Figure 3.1: Polygons formed by a convex tree and its convex hull

Paths. Another advantage of triangular convex trees specifically is that we can also look at the three separate paths, or 'branches', and try to get results from those. Paths are simpler structures than trees, so this way we might get results more easily.

Visibility. The first intuition that we followed was to define what 'visibility' means. Intuitively, the vertices connected in the compatible cycle need to be visible to each other in the convex tree, visibility between two vertices here meaning that a compatible straight edge can be drawn between the two vertices. This is why we first tried to properly define what we meant by this 'visibility' and how making certain choices in forming the cycle affected this visibility. This turned out to be quite difficult, but later in this report we did manage to get results in this regard.

Shrinking. One of the first algorithmic approaches that we tried in order to form a compatible cycle was to 'shrink' the convex hull. This involved taking the convex hull, in our case a triangle, and iteratively finding vertices that the edges in the cycle could pick up. This seemed to always be possible when using intuition, but putting a strategy on paper proved difficult. We tried differentiating between good and bad vertices to pick up, but we could not find a proper metric for this.

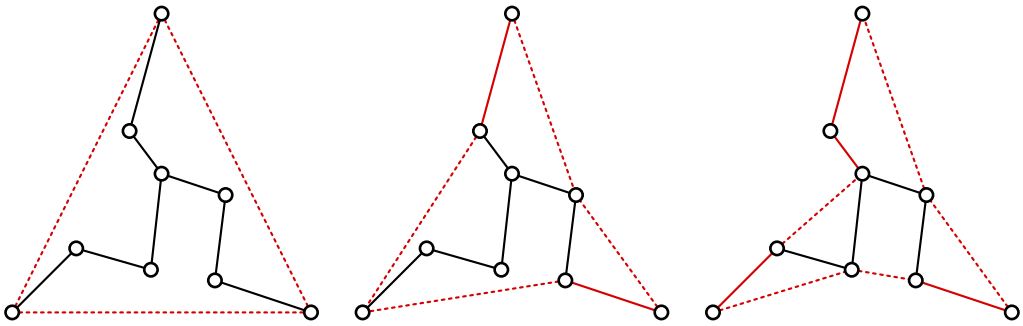


Figure 3.2: An example of shrinking the convex hull to form a cycle

Chapter 4

Hamiltonian triangulations

One way to categorize the visibility that a vertex has is using a planar triangulation. We know that every planar graph has at least one triangulation and we also know that this triangulation is compatible with the original graph. It follows that if the triangulation has a Hamiltonian cycle, then this cycle is also compatible with the original graph. This then changes the problem into proving that every convex tree has a Hamiltonian triangulation.

4.1 Separating triangles

In a triangulation, a separating triangle refers to any 3-cycle in the triangulation that, when removed, separates the graph. We also define a top-level separating triangle to be a separating triangle that is not nested inside of another separating triangle. Using these definitions, requirements can be made on planar triangulations for them to be Hamiltonian.

In 2002, B. Jackson and X. Yu [5] found such requirements. Using a tree decomposition of planar triangulations, they found that if the decomposition tree has a degree of at most 3 then the triangulation is Hamiltonian. Effectively this means that for a triangulation to be Hamiltonian, two things must hold:

i. The triangulation may contain at most 3 top-level separating triangles

ii. Any separating triangle may have at most 2 top-level separating triangles nested inside of it.

These requirements also agree with the non-convex tree found by M. van Garderen [7], as seen in Figure 4.1. This tree is a counterexample to the case where the tree does not have its leaves on the convex hull. It is not possible to triangulate this counterexample without creating at least 4 top-level separating triangles. There will be at least one in the middle of each of the three branches and there will also always be one in the middle of the tree. One possible configuration of these triangles has been marked in Figure 4.1.

In our problem, since we are dealing with convex trees, the problem of triangulating the tree reduces to triangulating the polygons formed by the tree and the convex hull. Since the triangulations of polygons cannot contain any separating triangles, they can only occur on the borders between polygons. In the case of our reduced problem with 3 branches there are three types of separating triangles, with all vertices in 1, 2 or 3 branches. These can be seen in Figure 4.2, corresponding to cases 1, 2 and 3 respectively. However, following from the requirement found by Jackson and Yu, we can ignore separating triangles with vertices in all three branches (Case 3). This is because if we only have this type of separating triangles, then all of the separating triangles must be nested inside of each other. This means that the decomposition tree of such a triangulation has at most degree 2.

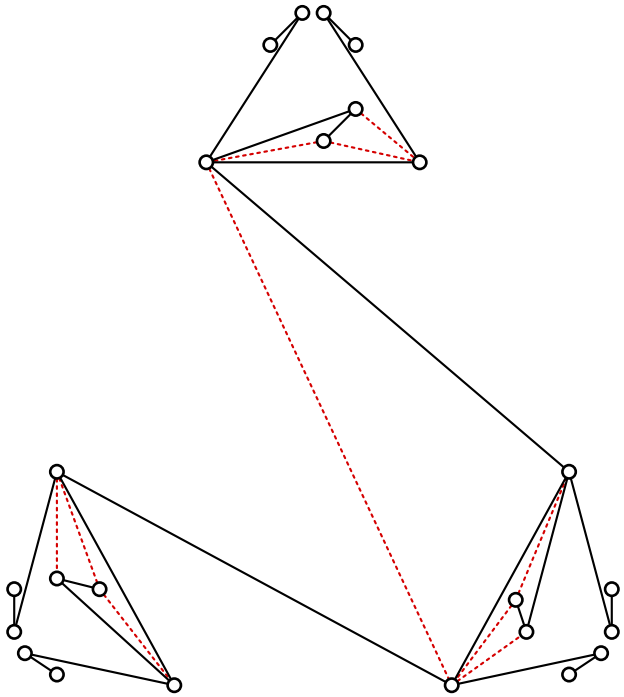


Figure 4.1: A simplified version of the tree found by M. van Garderen

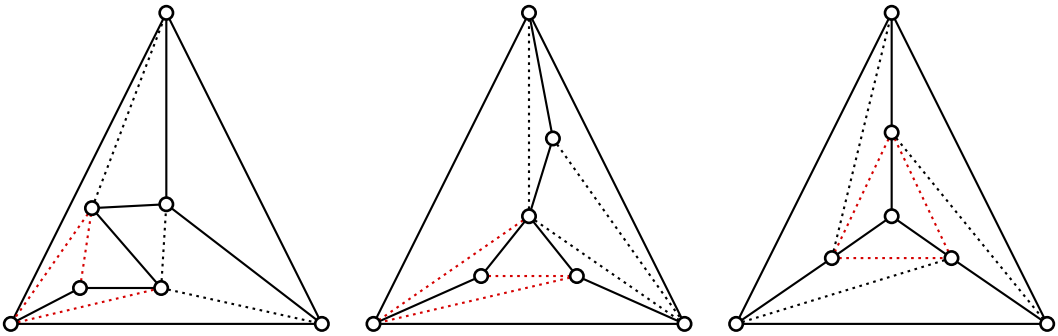


Figure 4.2: The three possible cases for separating triangles

4.1.1 Flipping edges

One particular way of moving between triangulations is by flipping edges. It has been shown that any triangulation is reachable from any other triangulation by simply flipping edges [6]. This makes it a good operation to find the triangulation that we want. By flipping an edge, it is possible to remove a separating triangle from a triangulation.

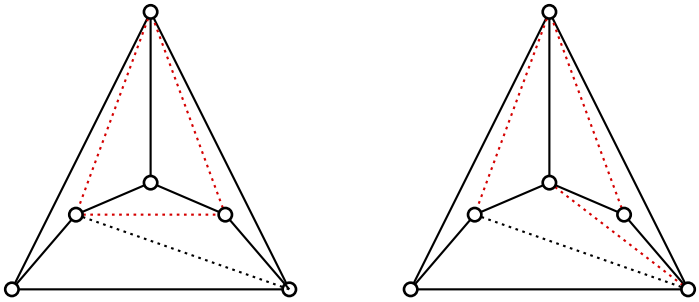


Figure 4.3: Removing a separating triangle using an edge flip

However, it is not always possible to flip an edge directly in the separating triangle. In this case, it is needed to chain multiple edge flips to make the desired edge flip available.

Another problem is that flipping an edge might introduce a different separating triangle. In this case, we need to also remove this new separating triangle using edge flips. However, this is not always possible. It is possible to create a convex tree where a separating triangle is forced, as seen in Figure 4.4. We were not able to find a convex tree with more than 5 vertices where this is the case, which leads us to conjecture this is not possible.

Conjecture 1. *Let $T = (V, E)$ be a convex tree, if $|V| \geq 5$ then there exists a triangulation that is 4-connected.*

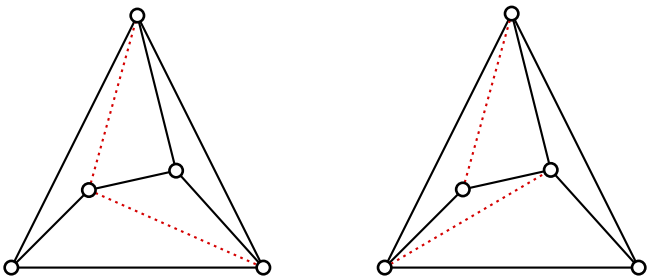


Figure 4.4: The only two possible triangulations of this convex tree, both leading to separating triangles

Chapter 5

Vertex claiming

To further explore the potential of identifying compatible Hamiltonian cycles within triangular convex trees, we use a strategy that first involves identifying smaller cycles contained in the trees. Our main intention is to find a way of assigning each vertex in a triangular convex tree to one of the smaller cycles, so that we can afterwards link them into a Hamiltonian cycle. This chapter will explore a decomposition technique for the triangular convex tree, a coloring technique for its vertices and then the subsequent creation of compatible cycles over that coloring.

5.1 Maximal convex polygon decomposition

Our goal is to fully split the triangular convex tree $G = (V, E)$ into smaller convex polygons. Starting with the three edges on the convex hull of G, we would go from the periphery, inward.

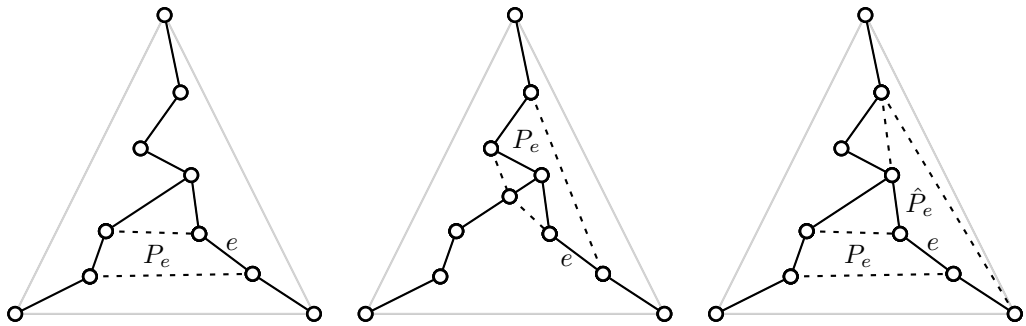
Definition 1 (Convex polygon with respect to e)**.** Let P_e be a polygon and G be a triangular convex tree in the Euclidean plane. We say that P_e is a convex polygon with respect to edge e if it complies to the following:

i. Edge e is part of the edge set of P_e.

ii. P_e is convex.

iii. P_e is compatible with G.

iv. P_e's interior face contains no vertices from V other than those on the boundary of P_e.



Valid P_e | Invalid P_e (violates iv) | $\hat{P}_e$ compared to a P_e

Figure 5.1: Convex polygons with respect to edge e

Furthermore, in a graph $G = (V, E)$, we say that $\hat{P}_e$ is a maximal convex polygon with respect to edge e if out of all possible P_e polygons, it encloses the largest geometrical area. One thing to note is that there may be more than one P_e and $\hat{P}_e$ in G.

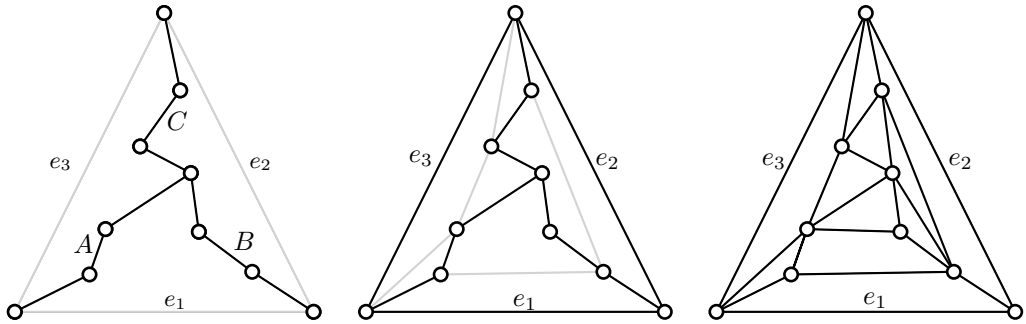
Let $A = (V_A, E_A)$, $B = (V_B, E_B)$ and $C = (V_C, E_C)$ be the three branches leaving the root of the triangular convex tree G, where each branch consists of a simple path extending from the root to a leaf as from the definition of the triangular convex tree. Consider the following algorithm to fully partition any given triangular convex tree into convex polygons:

Step 1: In the triangular convex tree, we start with the three edges e_1, e_2, and e_3 on the boundary of the convex tree. We color them gray, meaning we have discovered them. We also color every edge in $A \cup B \cup C$ black.

Step 2: We calculate a maximal convex polygon $\hat{P}_e$ for each grey edge e (previously discovered edge). $\hat{P}_e$ with the set of previously discovered maximal convex polygons must simultaneously be compatible with G.

Step 3: We color each grey edge black (the already discovered edges), meaning that there are no maximal convex polygons to be calculated that contain these edges.

Step 4: We color each non-black edge in $\hat{P}_e$ grey, meaning that we have just discovered it.



Step 1 of first iteration | Step 2, 3, 4 of first iteration | Finalisation of algorithm

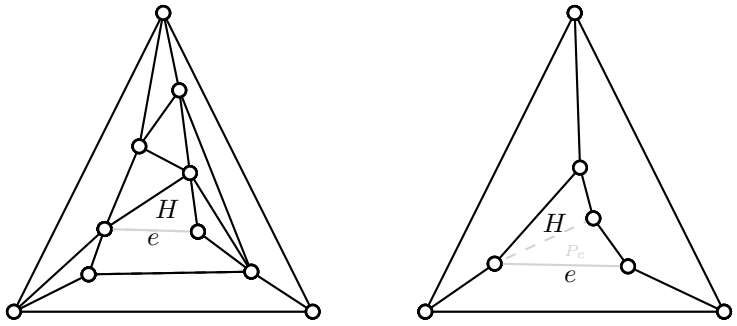
Figure 5.2: Decomposition Algorithm

Step 5: Repeat Steps 2 through 4, looping back to Step 2 with the current set of gray edges. The loop continues until the set of gray edges is empty, indicating that all maximal convex polygons have been calculated and therefore the convex tree has been fully partitioned into convex polygons.

Proposition 1. *The application of the decomposition algorithm on a triangular convex tree $G = (V, E)$ always returns a new $\hat{P}_e$ for a non-black edge e at any iteration of the algorithm.*

Proof. Since the algorithm considers edge e, G is not fully decomposed into convex polygons. Assume it was fully decomposed, e would have been black as it would have been contained in exactly one more maximal convex polygon $\hat{P}_e$'s boundary, by the definition of the algorithm. Thus there is some simple polygon H that has e as an edge in its boundary and is not yet decomposed. We also know by algorithm construction (coloring A, B and C black in $Step$ 1) that H lies completely between two of the branches of the triangular convex tree (A, B and C), being able to have common edges with the two branches.

Case 1: H is convex and contains no vertices from V: Assume H is convex and its interior face contains no vertices from V other than those on its boundary. The proof here is trivial, as this can become $\hat{P}_e$, having the necessary properties.



H is convex H is non-convex

Figure 5.3: Case 1 and 2 - Proposition 1

Case 2: H is non-convex and contains no vertices from V: Assume H is non-convex and its interior face contains no vertices from V other than those on its boundary. Since H is a simple non-convex polygon there is always a triangulation for it as mentioned in the previous chapter. Thus we can split H such that e will be part of the boundary of a convex polygon (a triangle). Similar to case A, we can form at least one $\hat{P}_e$.

Case 3: H contains other vertices from V: We will now show that this is impossible. By triangular convex tree construction all vertices are part of one of the branches A, B and C. Without loss of generality assume that H is contained in-between branches A and B. We also assume that there is at least one vertex from V inside H that is not contained in A or B, since algorithm construction would not allow this vertex to be part of A or B. This would mean that the vertex is contained in C. But this is impossible as H would not be contained between only two branches of G anymore. Thus, H contains no other vertices from V.

□

We start our algorithm with a finite number of edges: e_1, e_2, and e_3. The algorithm continues to calculate all maximal convex polygons, until none are left to be found. Consider the step where a gray edge e is reached. Since e is not yet black, the algorithm will seek $\hat{P}_e$. By Proposition 1, $\hat{P}_e$ will be found. The algorithm will stop decomposing when there are no gray edges to be applied on. This happens when there are no more maximal convex polygons that can be found in G. Thus the algorithm fully decomposes the graph G into maximal convex polygons.

5.2 Claiming vertices

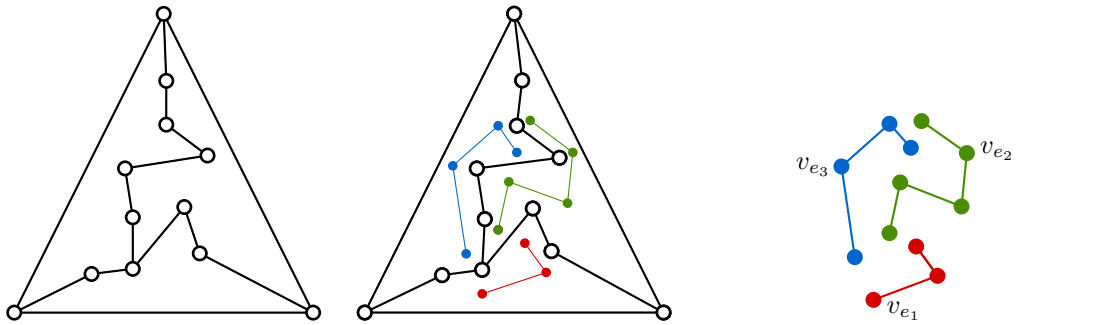
Our defined goal was to assign each vertex in a triangular convex tree to a smaller cycle. Assume we apply the algorithm described in the previous section on a convex triangular tree G. Edges e_1, e_2 and e_3 together with the branches A, B and C are the same with the definitions in the previous section. Using the resulting decomposition we will assign all vertices to compatible cycles that are enclosed inside the area formed by the graph's convex hull.

We delimit the three main simple polygons in G: P_A delimited by the edges in $A \cup B \cup e_1$, P_B delimited by the edges in $B \cup C \cup e_2$ and P_C delimited by the edges in $C \cup A \cup e_3$. We define the dual maximal polygon graph H of the triangular convex tree G to be a disconnected graph that for each maximal convex polygon in the maximal polygon decomposition of G, contains a vertex.

We modify Step 2 of our algorithm. For each previously discovered (grey) edge e, we calculate $\hat{P}_e$ and correspondingly add a vertex v_e in graph H. Then, if the algorithm has calculated other maximal convex polygons in a previous iteration, we review them with the purpose of finding the first $\hat{P}_{e'}$, where e' was considered in the previous iteration of the algorithm, that has at least one common boundary edge with $\hat{P}_e$. We make the condition stronger by requiring both $\hat{P}_{e'}$ and $\hat{P}_e$ to be contained between the same

two out of the three branches (A, B and C) of G. Once found, we connect the vertices corresponding to these polygons, v_e and $v_{e'}$, with an edge in H. If no $v_{e'}$ exists, then v_e is a root in its respective connected component.

We can easily see that each connected component in H is a tree, since vertices added later in the algorithm are connected to at most one vertex calculated earlier. The earliest calculated vertex is the root of the tree. After the creation of the dual disconnected graph H, we assign three different colors: red, blue and green to each of the three connected trees' vertices. Now each tree's vertices have the same color. Let the roots of the three trees in H be vertices v_{e_1}, v_{e_2} and v_{e_3} corresponding to $\hat{P}_{e_1}$, $\hat{P}_{e_2}$ and respectively $\hat{P}_{e_3}$ in G.



Graph G | Coloring polygons | Dual maximal polygon graph H of G

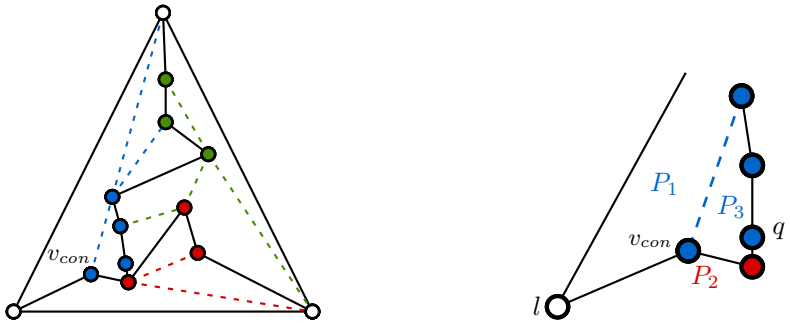
Figure 5.4: Claiming steps

We color every vertex v in G with the following vertex coloring strategy:

Case 1: If v is a leaf part of the convex hull of of G, it is not assigned any color.

Case 2: If v is on the boundary of polygons with their respective vertices in H having the same color, v takes the color of the polygons.

Case 3: If v is a vertex on the boundary of polygons where each polygon's respective vertex in H has a different color compared to the others, v takes the color of the vertex with the lowest depth in their respective connected tree component in H. If there are multiple choices with this criteria, but only one of these vertices is not a leaf in its respective connected component in H, we strengthen our choice by allowing v to take the color of this vertex. Furthermore, we have the unresolved conflict when two or more of these vertices are not leaves in their respective connected components in H.



Triangular convex tree with conflict | Conflict in assigning vertices to cycles

Figure 5.5: Coloring vertices and conflict in coloring

Consider a new triangular convex tree G with a conflict in coloring verticess. Notice vertex v_{con}. It is part of the boundaries of three maximal convex polygons P_1, P_2 and P_3 that are found together

compatible in G after applying our decomposition algorithm. Since both the corresponding vertices of P_1 and P_2 in H have depth 0, as they are roots in their respective trees, we are in the unresolved conflict situation of Case 3 of our vertex coloring strategy. In the Figure 5.5, we assign the vertex the color blue. Consider vertex q that by our strategy is also assigned the color blue. The blue cycle needs to contain both q and l. Thus, if v_{con} was red, edge ql would not be compatible in G. On the other hand, by assigning v_{con} blue we can have edges qv_{con} followed by $v_{con}l$ in our blue cycle, connecting q to l, successfully completing the cycle.

Now that we have assigned colors to all vertices in G except the convex leaves, we are left with assigning the vertices of each color into smaller cycles.

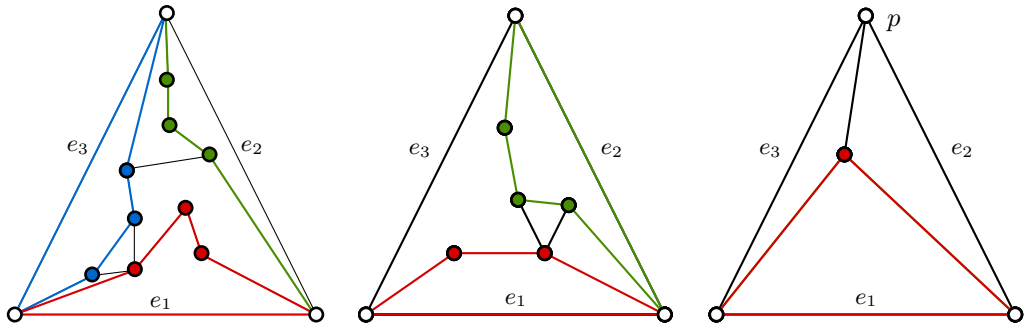
Conjecture 2. *It is always possible to construct a compatible cycle in G by using exactly all vertices of one color and two convex leaves.*

Conjecture 3. *There is always a split of a triangular convex tree G into compatible cycles containing all vertices of graph G except possibly a leaf part of the convex hull of G, where each cycle contains all vertices of exactly one color in G and two leaves that are on the convex hull of G. No cycle can contain the same pair of leaves on the convex hull as another cycle.*

The split of G into compatible cycles as mentioned in Conjecture 3 can result in the following three cases:

Case 1: All cycles contain each exactly one of $G's$ convex hull's edges, different from the others. Removing the convex hull's boundary edges from their respective cycles results in three simple paths. Let $P = P_1 \cup P_2 \cup P_3$ denote this set of paths. Each leaf part of the convex hull of G, due to its membership in two compatible cycles, is an endpoint of exactly two different paths in P. Since each pair of paths in P must share a leaf endpoint with a different path, these shared endpoints allow us to connect the paths sequentially, forming a Hamiltonian cycle.

Case 2: Two cycles are enough to pick the vertices of G, that case is analogous to **Case 1**. Removing the convex hull's boundary edges from their respective cycles results in two simple paths. Let P_1 and P_2 denote the paths. We use the third boundary edge that is not part of any cycle as an addition to the compatible paths P_1 and P_2 to form the compatible Hamiltonian cycle.



Case 1 Case 2 Case 3

Figure 5.6: The three possible cases for compatible cycle split

Case 3: One cycle contains all vertices in G except for one leaf part of the convex hull of G: p. The cycle contains one edge that is part of the convex hull of G. Let this edge be e_1. Removing this edge we get a simple path P_1. Similar to the other cases, to form a compatible Hamiltonian cycle, picking up the remaining p vertex, we add edges e_2 and e_3 to P_1 recompleting the cycle.

5.3 Filling the gaps

In this chapter we have seen how to decompose a triangular convex tree into cycles and then claim the vertices of the cycles to ultimately form a Hamiltonian cycle.

We still have two conjectures in our approach. The reason for this is that the current coloring strategy contains conflicts. Delving deeper into strengthening the coloring strategy may provide the solution to our problem. For Conjecture 2 it was always the case that we could color the vertices in such a way that we could form the smaller cycles with the existing coloring rules and, only when there would be a conflict our intuition would need to be used. Finally, for Conjecture 3, if all non-leaf vertices of a triangular tree would be colored correctly with one of the three colors resulting in the smaller cycles being formed, our problem would be solved as shown at the end of the section before.

Chapter 6

Adjacent polygons

A previously mentioned proposal to find a Hamiltonian cycle in the tree consisted of finding compatible Hamiltonian cycles for each of the three branches. Using these Hamiltonian cycles, it might have been easier to then find a compatible Hamiltonian cycle for the entire graph. When we considered a branch together with its convex hull, it turned out that the problem was once again that of finding a compatible Hamiltonian cycle in a graph formed by adjacent polygons. In this chapter, we consider simplifications of this problem.

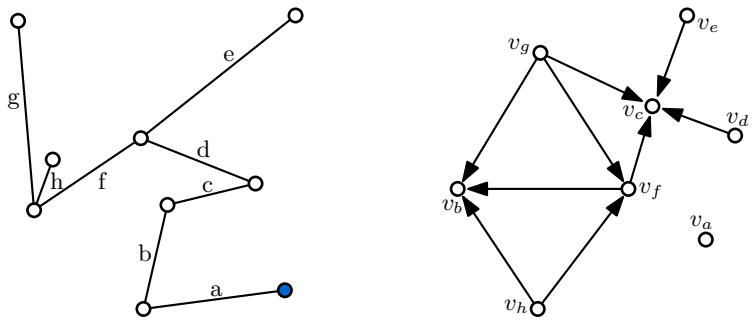
6.1 The convex quadrilateral

First, we consider a convex quadrilateral split by a path. This is equivalent to a path where the beginning and the end of the path are connected with an extra vertex on both sides of the path. This divides the quadrilateral into two adjacent polygons.

We first wish to formalize the situation where an edge can 'see' a vertex, intuitively this refers to when uninterrupted edges can be drawn between the ends of the edge and the vertex itself. This is a sufficient requirement as this means that uninterrupted lines can be drawn between the vertex and the ends of the edge, and so the vertex can be 'collected' by the edge. Hence in a given graph $G = (V, E)$, we say that a vertex $w \in V$ is *visible* to an edge $uv \in E$ if the triangle uvw is compatible with G. We also say that uv can see w. For completeness, we also consider that edges of the form xw can see w.

In the same way, it is useful to be able to refer to other edges that do not allow or *block* an edge from seeing vertices. Therefore we say that an edge $p \in E$ blocks an edge $q = uv \in E$ from seeing vertex $w \in V$ if p intersects the edges of triangle uvw. (and $p \notin \{q, wu, wv\}$)

Having defined blocking edges, we are interested in starting from an edge that does not have visibility of a vertex, and by considering its blocking edges, finding an edge that does. We construct the *blocking graph* $H(v) = (V_{H(v)}, E_{H(v)})$ of a vertex $v \in V$. For all edges $e \in E$ there exists a vertex $v_e \in V_{H(v)}$. There exists a directed edge from $v_p \in V_{H(v)}$ to $v_q \in V_{H(v)}$ if q blocks p from seeing v in graph G. An example of the construction of such a graph can be found in Figure 6.1.



Original graph Blocking graph w.r.t. the blue vertex

Figure 6.1: Blocking Graph Example

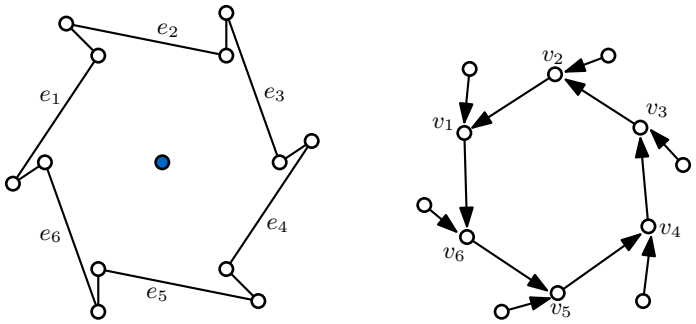


Figure 6.2: Cyclic Blocking Graph (not all edges and vertices are labeled)

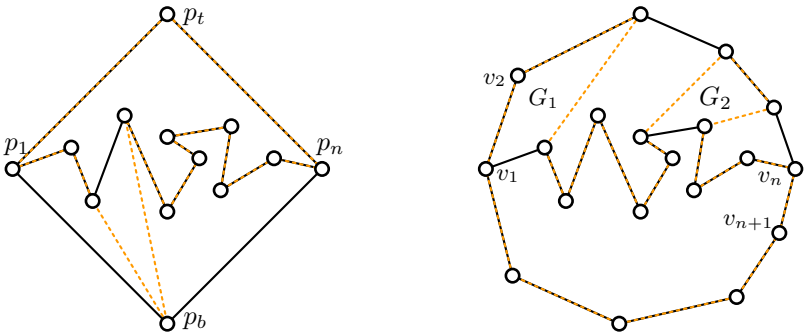
Proposition 2. *Let $G = (V, E)$ be a quadrilateral that encloses a path P. A blocking graph of a vertex $v \in V$ is acyclic.*

Proof. Let $H(v) = (V_{H(v)}, E_{H(v)})$ be the blocking graph of $v \in V$. Suppose $H(v)$ has a cycle $v_1, \ldots, v_n, v_1$. Let $v_i \in V_{H(v)}$ correspond to an edge $e_i \in E$. We know that for all $1 < i \leq n$, e_i intersects the triangle formed by edge e_{i-1} and vertex v. Additionally, e_1 intersects the triangle between e_n and v. For this to happen we understand in an intuitive sense that the edge e_i must 'block' part of the edge e_{i-1} from seeing v, while itself being partially blocked by e_{i+1}, (where e_n 'blocks' part of e_{n-1} from vertex v, and is blocked partially by e_1). Because all triangles are connected to v, we know that for such a cycle to occur, this cycle will have to surround the vertex v (An example of this can be seen in Figure 6.2). However, since v is part of the convex quadrilateral, and all other vertices are contained within this quadrilateral, and so they cannot surround v and so we arrive at a contradiction. Therefore, $H(v)$ cannot contain a cycle. □

Proposition 3. *Let $G = [p_1, p_t, p_n, p_b]$ be a convex quadrilateral that encloses a path $P = (V_P, E_P)$ where $V_P = \{p_i : 1 \leq i \leq n\}$ and $E_P = \{p_i p_{i+1} : 1 \leq i < n\}$. There exists a Hamiltonian cycle containing all vertices in G and P, that is compatible with G and P.*

Proof. Let $P = (V_P, E_P)$ be the path within G. We prove there is an edge $e \in E_P$ that can see p_b. Consider the blocking graph $H(p_b)$. Since $H(p_b)$ is a directed acyclic graph, it must have a sink. This sink represents an edge in P that has no blocking edges between itself and p_b and is therefore able to see p_b. Let this sink be the edge $p_j p_{j+1} \in E_P$ where $1 \leq j < n$.

Consider the Hamiltonian cycle that starts at vertex p_1, and traverses P until vertex p_j, it is now possible due to the vertex visibility to visit vertex p_b, and then return to vertex p_{j+1} on P. The cycle then finishes traversing P by reaching vertex p_n, moves on to vertex p_t, and finally returns to vertex p_1. Since



Convex Quadrilateral - Proposition 3 Convex Polygon - Proposition 4

Figure 6.3: Convex Shapes with internal Paths

the triangle $p_j p_b p_{j+1}$ is compatible, by definition of visibility, with the combined convex quadrilateral G and path P, we have that the entire cycle is compatible with $G \cup P$ proving the proposition.

An example of the graph defined within this proposition can be seen in Figure 6.3. The constructed cycle described is highlighted as the dotted yellow edges. □

6.2 The convex polygon

Instead of a quadrilateral, we now consider an arbitrary convex polygon. We extend the notion of visibility. Previously we discussed the visibility of a vertex to an edge, now we discuss the visibility of one or more connected edges from a single edge. In a graph $G = (V, E)$, we say that a path $P = p_1 \dots p_n$ in G is visible to an edge $v_1v_2 \in E$ if the polygon $v_1p_1 \dots p_nv_2$ is compatible with G (we also allow the case where $v_1 = p_1$ or $v_2 = p_n$). We also say that the edge v_1v_2 can see P, and that $v_1p_1 \dots p_nv_2$ is the polygon formed between the path P and the edge v_1v_2.

It is quickly noted from this definition that it is not necessary to consider all edges in the created polygon for compatibility with G. We can instead say that an edge $uv \in E$ can see the path $P = p_1 \dots p_n$ if the edges v_1p_1 and p_nv_2 are compatible with G. Similarly to Proposition 2, we see that once again the blocking graph of a vertex in G is acyclic.

Corollary 1. *Let $G = (V, E)$ be a convex polygon with $V = \{v_1, \dots, v_n, \dots, v_m\}$ where $3 \leq n < m$, and edges $e_i = v_iv_{i+1} \in E$ for all $1 \leq i < m$, as well as $e_m = v_mv_1 \in E$. Let $P = (V_P, E_P)$ be a path enclosed within G, that connects vertices v_1 and v_n. A blocking graph of a vertex $v \in V$ is acylic.*

The proof for this corollary is the same as that of Proposition 2. An example of the graph defined within the corollary can be seen in Figure 6.3.

Proposition 4. *Let $G = (V, E)$ be a convex polygon with $V = \{v_1, \dots, v_n, \dots, v_m\}$ where $3 \leq n < m$, and edges $v_iv_{i+1} \in E$ for all $1 \leq i < m$, as well as $v_mv_1 \in E$. Let $P = (V_P, E_P)$ be a path enclosed within G, that connects vertices v_1 and v_n. There exists a subset of edges $E_s \subseteq E_P$, and a set of paths P_s that are part of G, such that the following hold:*

i. Each path in P_s is visible to exactly one edge in E_s, and each edge in E_s sees exactly one path in P_s.

ii. The paths within P_s partition the set of vertices $\{v_1, \dots, v_{n-1}\}$.

iii. The polygons formed by an edge in E_s and its visible path in P_s are all simultaneously compatible with each other and to $G \cup P$.

Proof. Consider an edge $e_1 = p_1p_1' \in P$ that can see the path $v_1 \dots v_i$, but cannot see the vertex v_{i+1} due to some blocking edge f. Therefore $p_1v_1 \dots v_ip_1'$ is a polygon G_1 created by this visibility, (an example can be seen in Figure 6.3) and by the definition of path visibility, is compatible with $G \cup P$. Using the blocking graph $H(v_{i+1})$, we find a sink connected to v_{e_1}. We know this is possible through Corollary 1. This sink, v_{e_2} corresponding to the edge $e_2 = p_2p_2'$, can see v_{i+1}.

Suppose e_2 can see the path $v_{i+1} \dots v_j$ ($j \geq i + 1$), we once again know that the resulting polygon from this visibility, G_2, is compatible with $G \cup P$. However, now we must ensure that G_1 and G_2 are compatible with each other also. As e_1 sees the path $v_1 \dots v_i$, this visibility could only have been broken by the new edges forming G_2, namely p_2v_{i+1} and v_jp_2'. We therefore have to show that these edges do not intersect G_1.

To do this, we consider the vertices visited in the blocking graph $H(v_{i+1})$ from v_{e_1} to the sink v_{e_2}. Let the set of such edges that correspond to these vertices be defined as $E_B \subseteq E_P$. We note that for all $e \in E_b$, e cannot exist within the polygon G_1 since this implies the existence of a blocking edge between e_1 and the path $v_1 \dots v_i$, which we know cannot exist due to the assumption of this visibility. Furthermore, for any such e, the polygon created between e and the path $v_{i+1} \dots v_j$ does not intersect G_1 (although visibility of e to this path is not assumed as it may not be the sink in the blocking graph). We see this since f blocks the visibility of e_1 to v_{i+1}, so the polygon formed from f cannot intersect

G_1 due to the convexity of G and since f already does not intersect G_1. Hence if there now exists a blocking edge $g \in E_b$ such that the polygon formed between g and v_{i+1} intersects with G_1, then g must pass 'behind' (with respect to v_{i+1}) the other edges in E_b that have been already visited to reach such a position. This would also imply that some previously visited edge in E_b is now a blocking edge to g, thus creating a cycle with the blocking graph which we know cannot be the case, hence no such edge g exists in E_b. Figure 6.4 illustrates a simple example of this fact.

We now proceed and in a similar manner, we find an edge e_3 that can see some path $v_{j+1} \dots v_k$ ($k \geq j+1$). With similar reasoning we conclude that the polygon G_3 formed by the visibility of $v_{j+1} \dots v_k$ to e_3 is compatible with both G_1 and G_2. We can continue this process until we have found a set of edges E_s from E_p that together can see paths (together making the set P_s) that partition the vertices $v_1 \dots v_n$.

We note that each edge in E_s is related to exactly one path in P_s, and that the polygons formed between an edge in E_s and its corresponding visible path in P_s are simultaneously compatible. □

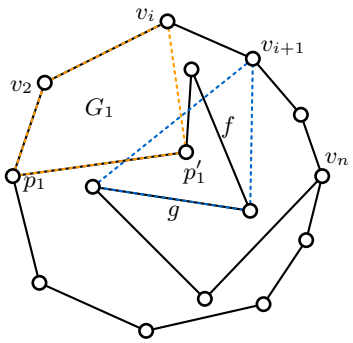


Figure 6.4: Here we see that the polygon formed between g and v_{i+1} (in blue) does intersect G_1 (bounded by yellow). However, we note that $g \notin E_b$, and moreover f sees v_{i+1}. The fact that $g \notin E_b$ holds by Corollary 1, as discussed in the proof for Proposition 4.

With this proposition, similarly as was done for Proposition 3, we can conclude that a Hamiltonian cycle containing all vertices in G and P that is compatible with G and P exists. This can easily be done by starting at v_n and following edges in E to v_{n+1} until v_m and finally to v_1. From v_1, we then follow the path P back to v_n, whilst collecting all vertices v_i ($1 < i < n$) by using the visibilities discussed in Proposition 4. In Figure 6.3 an example of the constructed cycle described is highlighted with dotted yellow edges. Examples of the mentioned polygons G_1 and G_2 are also labeled.

6.3 Pushing non-convexity

The natural continuation at this point is to consider whether such a technique can be applied to a polygon G and enclosed path P when we loosen the convexity requirement previously imposed on G.

We begin by noting that the convexity requirement within Proposition 4 is too strong for what is necessary, specifically that the convexity requirement on G does not have to be strict on the path $v_{n+1} \dots v_m$. We see that this is the case since we only require 'convexity' to hold on the edges of G that we wish to be visible from the path P. However then what sort of properties must hold to ensure such path visibilities exist?

Definition 2 (Pseudo-Convex)**.** In a polygon $G = (V, E)$, we say that a path on the polygon $P = p_1 \dots p_n$ is pseudo-convex on G if and only if P is a subset of the edges that define the convex hull of G.

Proposition 5. *Let $G = (V, E)$ be a polygon with $V = \{v_1, \dots, v_n, \dots, v_m\}$ where $3 \leq n < m$, and edges $v_i v_{i+1} \in E$ for all $1 \leq i < m$, as well as $v_m v_1 \in E$. Let $P = (V_P, E_P)$ be a path enclosed within G, that connects vertices v_1 and v_n. Where the path $P_C = v_1 v_2 \dots v_n$ is pseudo-convex on the polygon enclosed by*

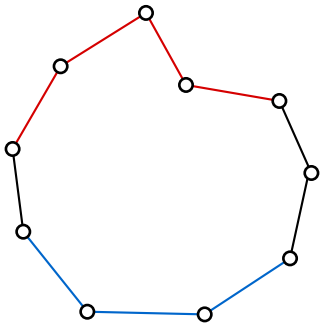


Figure 6.5: In the polygon G we see the pseudo-convex path P on G (in blue) and the not pseudo-convex path Q on G (in red)

P and P_C. There exists a subset of edges $E_s \subseteq E_P$, and a set of paths P_s that are part of G, such that the following hold:

i. Each path in P_s is visible to exactly one edge in E_s, and each edge in E_s sees exactly one path in P_s.

ii. The paths within P_s partition the set of vertices $\{v_1, \dots, v_{n-1}\}$.

iii. The polygons formed by an edge in E_s and its visible path in P_s are all simultaneously compatible with each other and with $G \cup P$.

Proof. We first note, that such a graph G, (an example of which can be seen in Figure 6.6), this is comparable to the graph considered in Proposition 4, except the implied pseudo-convexity on G of the path $v_n v_{n+1} \dots v_1$ along the polygon is now lost. With this change however, we see that the visibility of subpaths in P_C (that we wish to collect) to edges within P does not change by this lost pesudo-convexity.

We see that this is the case since this path visibility is only determined by the edges that make up the polygon enclosed between P and P_C. Therefore, as the edges of G that are no longer stated to be convex are not part of this polygon, we conclude that such path visibilities are once again possible by the same reasoning used for Proposition 4. □

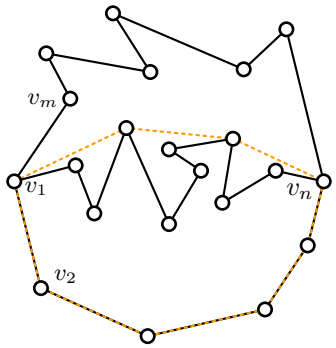


Figure 6.6: Example of a Graph considered in Proposition 5. The Convex hull of the polygon enclosed by P_C and P is illustrated by the yellow dotted shape.

6.4 Application to the Triangles

Having discussed path visibility and its power in finding Hamiltonian cycles under conditions, we now wish to apply this to triangular convex trees.

Corollary 2. *A triangular convex tree has a compatible Hamiltonian cycle when one branch is pseudo-convex on the polygon $\hat{G}$ enclosed by itself, another branch, and the edge connecting the leaves of these two branches.*

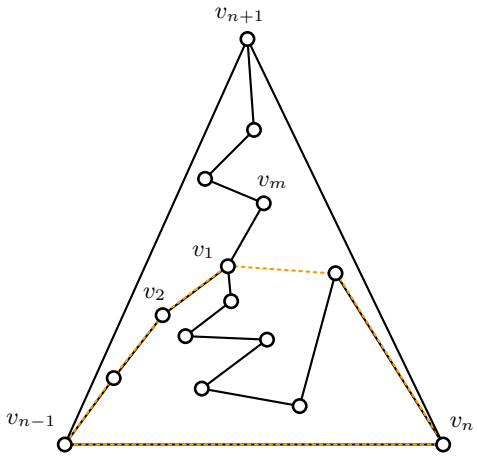


Figure 6.7: Labeling scheme on a triangular convex tree. The dotted yellow line demonstrates the convex hull of the polygon enclosed by the bottom branches, and the pseudo-convexity of $v_1 \dots v_{n-1}$

Proof. The proof of this follows directly from Proposition 5, and the labeling scheme previously defined.

Label the central vertex of the tree (with degree three) as v_1, following the branch that is pseudo-convex on $\hat{G}$ towards the leaf, label each vertex as v_i with increasing index i, the label at the leaf as v_{n-1}. Then label the leaf of the other branch that encloses $\hat{G}$ as v_n and the final remaining leaf as v_{n+1}. Finally, the vertices on the branch connected to v_{n+1} are labeled with increasing index once again in decreasing depth until reaching v_1 once again, with the label of the final vertex as v_m. An example of this labeling scheme is shown in Figure 6.7.

We can consider, using this labeling scheme, a polygon $G = (V, E)$ with $V = \{v_1, \dots, v_n, \dots, v_m\}$ where $3 \leq n < m$, and edges $v_i v_{i+1} \in E$ for all $1 \leq i < m$, as well as $v_m v_1 \in E$. Additionally, we find the path $P = (V_P, E_P)$ connecting $v_1 v_n$, as the other branch used to enclose the polygon $\hat{G}$.

This labeling of the triangular convex tree can now directly be compared to the graph G and path P considered in Proposition 5. Hence we can conclude that there exists a Hamiltonian cycle compatible with the tree. □

Chapter 7

Conclusion

In conclusion, this report explored the feasibility of finding compatible Hamiltonian cycles within convex trees and especially, triangular convex trees. We simplified the problem, while keeping its original significance. Firstly, our research builds intuition for understanding the problem. We explored multiple ways of approaching the problem. One way is by trying to prove the existence of a Hamiltonian triangulation of the convex tree. We also explored an algorithmic approach using vertex claiming and a tree decomposition.

Our main result is that we managed to prove that compatible cycles exist in the cases of the convex quadrilateral and the convex polygon that are split by simple paths. We applied these results in finding a compatible Hamiltonian cycle in a triangular convex tree under certain conditions. However, the results are limited by the required condition that one of the branches needs to be pseudo-convex, which is not universally applicable.

Future studies could continue with any of the proposed methods. One could try to prove the existence of a Hamiltonian triangulation. It is also an option to aim to generalize the successful findings in the simpler configurations to less constraint convex tree structures. Additionally, exploring algorithmic approaches to find these compatible Hamiltonian cycles could bridge the gap between theoretical existence and practical computability.

Thus, our investigation delved into potential methods to solving the problem and identifying key features for successful cycle construction. It also proposes methods for subsequent research in this problem in the field of computational geometry.

Bibliography

[1] O. Aichholzer, S. Bereg, A. Dumitrescu, A. García, C. Huemer, F. Hurtado, M. Kano, A. Márquez, D. Rappaport, S. Smorodinsky, D. Souvaine, J. Urrutia, and D.R. Wood. Compatible geometric matchings. *Electronic Notes in Discrete Mathematics*, 31:201–206, 2008.

[2] O. Aichholzer, K. Knorr, W. Mulzer, N. El Maalouly, J. Obenaus, R. Paul, M.M. Reddy, B. Vogtenhuber, and A. Weinberger. Compatible spanning trees in simple drawings of K_n. *International Symposium on Graph Drawing and Network Visualization*, 30, 2022.

[3] A. García, C. Huemer, F. Hurtado, and J. Tejel. Compatible spanning trees. *Computational Geometry*, 47(5):563–584, 2014.

[4] M. Ishaque, D. Souvaine, and C.D. Tóth. Disjoint compatible geometric matchings. *ACM Symposium on Computational Geometry*, 27, 2011.

[5] B. Jackson and X. Yu. Hamilton cycles in plane triangulations. *Journal of Graph Theory*, 41:138–150, 2002. https://doi.org/10.1002/jgt.10057.

[6] E. Osherovich and A.M. Bruckstein. All triangulations are reachable via sequences of edge-flips: an elementary proof. *Computer Aided Geometric Design*, 25(3):157–161, 2007. https://www.sciencedirect.com/science/article/pii/S0167839607000763.

[7] M. van Garderen. How path-like is a tree? Compatible transformations from tree to path and vice versa. Master's thesis, Technische Universiteit Eindhoven, 2013. https://research.tue.nl/en/studentTheses/how-path-like-is-a-tree.