



This document is set in Palatino, compiled with pdfL<sup>A</sup>T<sub>E</sub>X<sub>2</sub>ε and Biber.

The L<sup>A</sup>T<sub>E</sub>X template from Karl Voit is based on KOMA script and can be found online: <https://github.com/novoid/LaTeX-KOMA-template>

## Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Graz, \_\_\_\_\_  
Date Signature

## Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, am \_\_\_\_\_  
Datum Unterschrift

## Abstract

In 1973 the first art gallery problem was formulated by Viktor Klee (How many guards are needed to guard an art gallery?). Since then many visibility problems in the context of polygons have been studied in the literature. One of these is the two guards problem, first formulated and discussed by Christian Icking and Rolf Klein in 1992. In this thesis an adapted version of this problem is studied: Given a starting vertex of a polygon, how far can two guards walk on the clockwise respectively counterclockwise boundary of the polygon while maintaining mutual visibility?

To solve this maximal walk problem, existing concepts are adapted and extended. The results are used to develop an algorithm that not only solves the maximal walk problem but is formulated in more general terms so that it can be used for similar problems as well. This is demonstrated by formulating and solving an example problem.

# Contents

<b>Abstract</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Goal . . . . .	2
1.3 Outline . . . . .	2
<b>2 Preliminaries</b>	<b>4</b>
<b>3 Foundations - Existing Work and Basic Properties</b>	<b>10</b>
3.1 Discrete Straight Walks . . . . .	10
3.2 Previous Results . . . . .	14
<b>4 Characterizing Maximal Walks</b>	<b>16</b>
4.1 Induced Polygons . . . . .	16
4.2 $LR$ -Visibility With Respect to $s$ . . . . .	17
4.3 Semi-Wedges With Respect to $s$ . . . . .	23
4.4 Backward Deadlocks With Respect to $s$ . . . . .	30
4.5 Forward Deadlocks With Respect to $s$ . . . . .	37
4.6 Summary . . . . .	45
<b>5 Conditional Constraints</b>	<b>46</b>
5.1 Fulfilling Constraints . . . . .	47
5.2 Multiple Starting Pairs and Feasible Pairs . . . . .	49
5.3 Summary . . . . .	52
<b>6 Maximal Visibility</b>	<b>53</b>
<b>7 Putting Everything Together</b>	<b>59</b>
7.1 Absolute Constraints for Maximal Walks . . . . .	59

## Contents

7.2	Conditional Constraints for Maximal Walks . . . . .	60
7.3	Starting Pairs . . . . .	60
7.4	Induced Semi-Wedges . . . . .	61
7.5	Maximal Walks Problem Solution . . . . .	61
7.6	Summary . . . . .	62
<b>8</b>	<b>Other Uses of the Constraint Processing Algorithm</b>	<b>63</b>
8.1	The Maximal Same Colors Walk Problem . . . . .	63
8.2	Solving MSCW . . . . .	64
8.3	Examples . . . . .	65
8.4	Summary . . . . .	71
<b>9</b>	<b>Conclusion</b>	<b>72</b>
9.1	Review . . . . .	72
9.2	Outlook . . . . .	73
	<b>Bibliography</b>	<b>76</b>

## List of Figures

1	Backward and forward ray shots . . . . .	5
2	Violated $LR$ -visibility conditions . . . . .	6
3	(Semi-)Wedge . . . . .	8
4	Deadlocks . . . . .	8
5	Walk . . . . .	11
6	Triangulation with ear, link and branch triangles. . . . .	12
7	(Non-)Hamiltonian triangulations . . . . .	13
8	Induced semi-wedge . . . . .	18
9	$LR$ -visibility connectedness . . . . .	19
10	$LR$ -visibility ordering . . . . .	20
11	$LR$ -visibility constraints . . . . .	21
12	Semi-wedges with respect to $s$ . . . . .	24
13	Backward deadlock with respect to $s$ . . . . .	31
14	Minimal forward deadlock not unique . . . . .	38
15	$\Theta(n)$ minimal forward deadlocks . . . . .	39
16	Conditional Constraint . . . . .	47
17	Multiple upper limit pairs . . . . .	49
18	Top and bottom chains . . . . .	54
19	All shortest path vertices on $B$ . . . . .	55
20	First shortest path vertices on $B$ . . . . .	55
21	First shortest path vertex on $T$ . . . . .	56
22	Maximal same colors walk example with a convex polygon (Example 2) . . . . .	67
23	Walkthrough of convex polygon example . . . . .	68

## List of Figures

24	Maximal same colors walk example with a non-convex polygon (Example 3) . . . . .	69
25	Walkthrough of non-convex polygon example . . . . .	70
26	Walk with multiple guards . . . . .	74



# 1 Introduction

In this introductory chapter the motivation for this work is described and an outline over the rest of the thesis is provided.

## 1.1 Motivation

The first art gallery problem - with the art gallery represented by a polygon - was formulated in 1973 by Viktor Klee (see for example [11]):

*How many guards does it take to completely guard an art gallery with  $n$  vertices?*

Since then this problem as well as plenty variations have been studied in the literature. One of these variations was created and discussed by Christian Icking and Rolf Klein in 1992 (see [9]), the *two guards problem*:

*Given a polygon  $\mathcal{P}$ , a starting vertex  $s$  and a target vertex  $t$ , can two guards walking along the boundary of the polygon in clockwise respectively counterclockwise direction from  $s$  to  $t$  coordinate their walks such that they always maintain mutual visibility inside of  $\mathcal{P}$ ?*

Icking and Klein presented solutions for the two guards problem in their paper, and various other authors have since then shown improved algorithms as well as formulated slight variations of this problem. One of these variations considers only walks where the guards always move forward to their target vertex  $t$ , they always move from one vertex to the next (without “stops” along the edges) and the two guards are not allowed to move simultaneously.

These kind of walks are called *discrete straight walks* and correspond to hamiltonian triangulations (see [10]). Hamiltonian triangulations of polygons have the special property that at least one edge of each triangle lies on the boundary of the polygon (thus their dual graphs are simple paths). In the context of

## 1 Introduction

triangulation axes of polygons (see [1]) those kind of triangulations lead to favorable properties of the corresponding triangulation axis.

This thesis started with the idea to use *discrete straight walks* to prune triangulation axes by computing walks for parts of the triangulated polygon. The basic plan was to use a starting vertex  $s$  and get as far as possible with a *discrete straight walk* of two guards - in short to compute a maximal discrete straight walk from a starting point  $s$ . Soon it turned out that the scope of the thesis was filled with this problem alone and it had become a work on yet another art gallery problem.

### 1.2 Goal

Essentially this thesis is about a single problem: computing maximal walks from a starting point in a polygon. More formally:

**Definition 1** (Maximal Walks Problem). *Let  $\mathcal{P}$  be a simple polygon and let  $s$  be a vertex of  $\mathcal{P}$ . Compute all maximal discrete straight walks from  $s$ .*

Note that the *Maximal Walks Problem* asks for *all* maximal walks - indeed it turns out that there is in general no unique maximal discrete walk from  $s$ , there may be multiple maximal discrete straight walks.

The goal of this thesis is to characterize maximal walkability with respect to a starting vertex  $s$ , and to provide an efficient algorithm to solve the *Maximal Walks Problem*.

### 1.3 Outline

Chapter 2 introduces common notations, definitions and concepts that are used throughout this thesis.

In Chapter 3 previous work on walks and in particular discrete straight walks will be reviewed. This provides the basis for the new work developed in the remaining chapters.

## 1 Introduction

Chapter 4 adapts concepts that are used in existing work to characterize (discrete straight) walkable polygons to the setting of maximal walkability. It is shown that these adapted concepts can be used to characterize maximal discrete straight walks from a starting vertex.

Chapter 5 takes a detour from the maximal walks setting and develops an abstract constraint processing algorithm that is later used to solve the maximal walks problem.

In Chapter 6 tools are developed to ensure mutual visibility of the walk target vertices within the original polygon.

Finally, in Chapter 7, all the developed tools are combined into an algorithm that solves the maximal walks problem.

In Chapter 8 the constraint processing algorithm is applied to another problem (the maximal same colors walk problem) to show its generic nature and to provide insights into its workings by walking through some examples.

The concluding Chapter 9 summarizes the achievements of this thesis. In addition possible improvements as well as ideas for further research are presented.

## 2 Preliminaries

This chapter contains conventions and basic definitions that are used throughout this thesis. The notations and concepts introduced here are essential tools to describe and study visibility properties of polygons such as walkability.

For brevity we will from now on assume the following (unless stated otherwise):

- A polygon is always a *simple* polygon.
- $\mathcal{P}$  is a polygon.
- $s$  and  $t$  with  $s \neq t$  are vertices of  $\mathcal{P}$ , referred to as *start* and *target* vertex.

**Definition 2** (Convex and reflex vertices). *Let  $p$  a vertex of  $\mathcal{P}$  and  $\alpha$  the interior angle of  $p$ .  $p$  is called convex if  $\alpha < \pi$ , and reflex if  $\alpha > \pi$ .*

Note that degenerate polygons with vertices that have an interior angle equal to  $\pi$  are not discussed in this thesis.

**Definition 3** (Preceding and succeeding vertices). *Let  $p$  a vertex of  $\mathcal{P}$ . We define  $\text{Succ}_{cw}(p)$  and  $\text{Pred}_{cw}(p)$  to be the vertices of  $\mathcal{P}$  that succeed and precede  $p$  on the boundary of  $\mathcal{P}$  in clockwise direction, respectively.  $\text{Succ}_{ccw}(p)$  and  $\text{Pred}_{ccw}(p)$  are defined analogously in counterclockwise direction.*

A very useful tool in determining visibility properties of polygons are ray shots along the edges of reflex vertices:

**Definition 4** (Ray shots (as defined in [9])). *For every reflex vertex  $p$  of  $\mathcal{P}$  we define  $\text{Forw}_{cw}(p)$  and  $\text{Backw}_{cw}(p)$  as follows:*

- $\text{Forw}_{cw}(p)$  is the first intersection of the forward ray starting from  $p$  in direction  $\overrightarrow{\text{Pred}_{cw}(p)p}$  with the boundary of  $\mathcal{P}$ .

## 2 Preliminaries

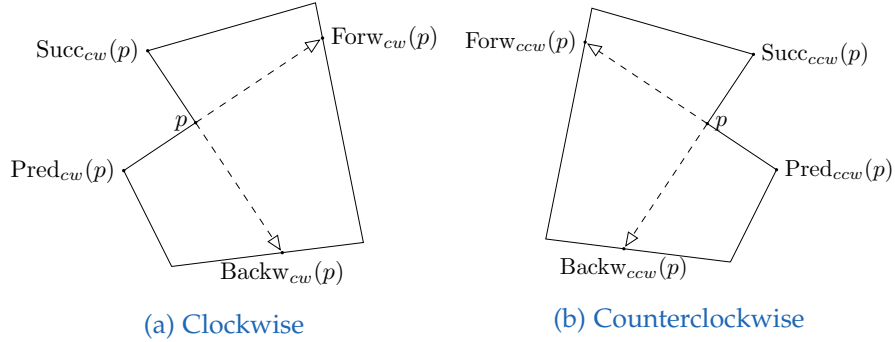


Figure 1: Backward and forward ray shots

- $Backw_{cw}(p)$  is the first intersection of the backward ray starting from  $p$  in direction  $\overrightarrow{Succ_{cw}(p)p}$  with the boundary of  $\mathcal{P}$ .

$Forw_{ccw}(p)$  and  $Backw_{ccw}(p)$  are defined analogously in counterclockwise direction. See Figure 1 for examples.

Sometimes it is necessary to indicate the polygon where the ray shooting is performed. This is done by putting the polygon in the superscript, e.g.:  $Backw_{cw}^{\mathcal{P}}(p)$ .

A concept closely tied to walkability is the mutual visibility of the boundary chains from  $s$  to  $t$  in clockwise and counterclockwise direction:

**Definition 5** (LR-visibility (see e.g. [3])). Let  $L$  and  $R$  be the clockwise and counterclockwise boundary chains of  $\mathcal{P}$  from  $s$  to  $t$ , respectively. If every point of  $L$  is visible from some point of  $R$  and vice versa, then  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$ .  $\mathcal{P}$  is called LR-visible if it is LR-visible with respect to some vertices  $s$  and  $t$ .

**Definition 6** (Ordering of vertices on the polygon boundary). Let  $p$  and  $q$  be two vertices of  $\mathcal{P}$ . The relation  $<_{cw}$  with respect to  $\mathcal{P}$  and  $s$  is defined as follows:

$p <_{cw} q \Leftrightarrow p$  is encountered prior to  $q$  when scanning the boundary of  $\mathcal{P}$  from  $s$  in clockwise direction. Note that the scan does not wrap around  $s$ .

The relation  $<_{ccw}$  is defined analogously.

**Theorem 1** (Characterization of LR-visibility (see [9])).  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$  if and only if the following conditions are satisfied:

## 2 Preliminaries

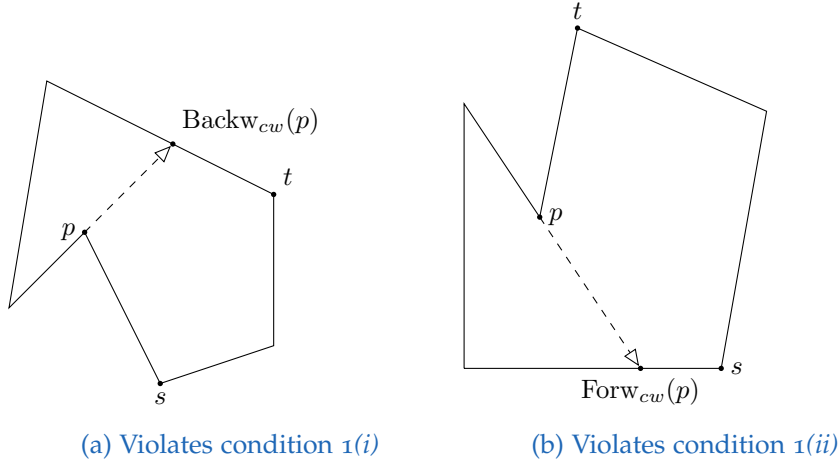


Figure 2: Backward and forward shots violating the conditions for  $LR$ -visibility from Theorem 1

1.  $L$  visible from  $R$ : There is no reflex vertex  $p \in L$  such that
  - (i)  $p \neq s$  and  $p <_{cw} \text{Backw}_{cw}(p) \in L \setminus \{t\}$ , or
  - (ii)  $p \neq t$  and  $p >_{cw} \text{Forw}_{cw}(p) \in L \setminus \{s\}$ .
2.  $R$  visible from  $L$ : There is no reflex vertex  $q \in R$  such that
  - (i)  $q \neq s$  and  $q <_{ccw} \text{Backw}_{ccw}(q) \in R \setminus \{t\}$ , or
  - (ii)  $q \neq t$  and  $q >_{ccw} \text{Forw}_{ccw}(q) \in R \setminus \{s\}$ .

See Figure 2 for polygons violating the conditions of the characterization of  $LR$ -visibility.

If the condition 1(i) is violated by some vertex  $p$ , i.e.  $p <_{cw} \text{Backw}_{cw}(p) \in L$ , then the vertex following  $p$  on  $L$  in clockwise direction can't be seen by any point on  $R$ . Similarly a vertex  $p$  with  $p >_{cw} \text{Forw}_{cw}(p) \in L$  violating condition 1(ii) implies that the vertex preceding  $p$  on  $L$  in clockwise direction can't be seen by any point on  $R$ .

The following concepts (wedges, semi-wedges and deadlocks) are essential obstacles to the walkability of polygons. In fact they are crucial to characterize walkable polygons.

## 2 Preliminaries

**Definition 7** (Wedge, semi-wedge (see [9] and [10])). *Let  $p$  and  $q$  be reflex vertices of  $\mathcal{P}$ .  $p$  and  $q$  form a left wedge if*

1. a)  $p, q \in L$ , and
- b)  $p <_{cw} q$ , and
- c)  $Backw_{cw}(p), Forw_{cw}(q) \in R$ , and
- d)  $Forw_{cw}(q) <_{ccw} Backw_{cw}(p)$ ,

*or analogously a right wedge*

2. a)  $p, q \in R$ , and
- b)  $p <_{ccw} q$ , and
- c)  $Backw_{ccw}(p), Forw_{ccw}(q) \in L$ , and
- d)  $Forw_{ccw}(q) <_{cw} Backw_{ccw}(p)$ .

*$p$  and  $q$  form a left semi-wedge if the wedge condition 1d is replaced by the following:*

- $Forw_{cw}(q) <_{ccw} Backw_{cw}(p)$ , or
- $Forw_{cw}(q) \geq_{ccw} Backw_{cw}(p)$ , but there is no vertex  $q$  of  $\mathcal{P}$  such that  $Backw_{cw}(p) \leq_{ccw} q \leq_{ccw} Forw(q)$ .

*This can be done analogously for condition 2d.*

See Figure 3 for examples of both kinds of wedges.

Note that for wedges it is necessary that the shooting rays intersect inside of  $\mathcal{P}$ . For semi-wedges this is not necessary but sufficient, thus each wedge is also a semi-wedge.

**Definition 8** (Deadlocks (see [9])). *Let  $L$  and  $R$  be the clockwise and counterclockwise boundary chains of  $\mathcal{P}$  from  $s$  to  $t$ , respectively. Let  $p \in L$  and  $q \in R$  be two reflex vertices with  $p, q \notin \{s, t\}$ .  $p$  and  $q$  form a forward deadlock if the following properties are satisfied:*

1.  $q >_{ccw} Forw_{cw}(p) \in R$  and
2.  $p >_{cw} Forw_{ccw}(q) \in L$

*$p$  and  $q$  form a backward deadlock if the following properties are satisfied:*

1.  $q <_{ccw} Backw_{cw}(p) \in R$
2.  $p <_{cw} Backw_{ccw}(q) \in L$

## 2 Preliminaries

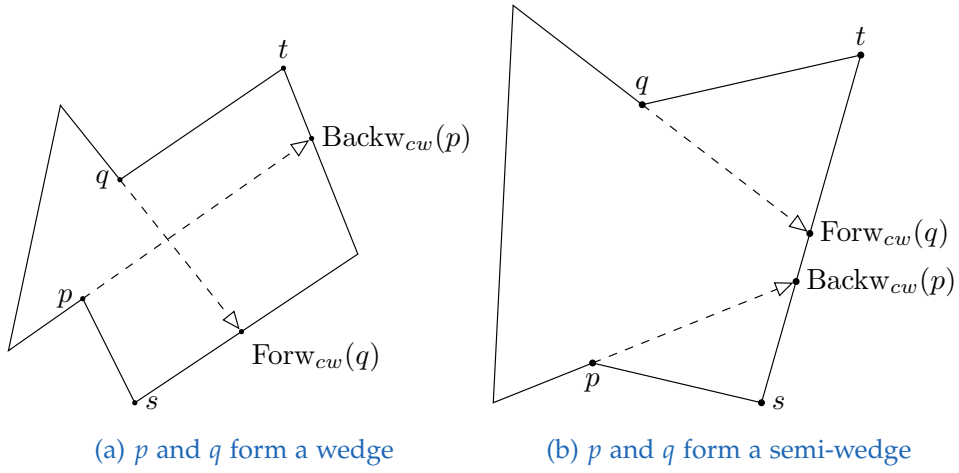


Figure 3: Wedge examples

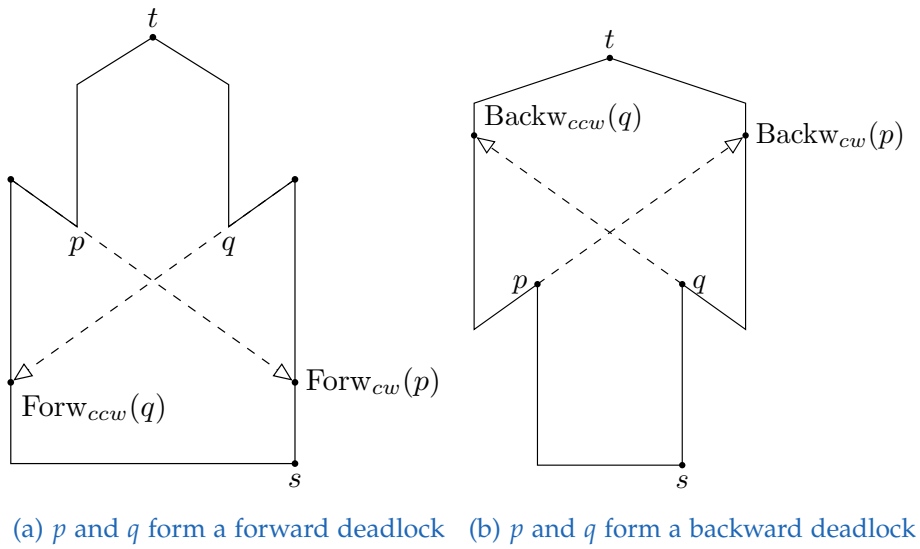


Figure 4: Deadlock examples



## 2 Preliminaries

See Figure 4 for examples of both kinds of deadlocks.

Now all notions necessary to discuss walkability of polygons are introduced.  $LR$ -visibility, wedges and deadlocks are essential tools to characterize walkability. Notice how deadlocks consist of vertex pairs with one vertex from each boundary chain  $L$  and  $R$ , whereas wedges consist of vertex pairs with both vertices from the same boundary chain.

## 3 Foundations - Existing Work and Basic Properties

The purpose of this chapter is to review existing work related to maximal walks. The discussion is mainly about properties and characterizations of walkability of polygons with respect to given start and end vertices.

### 3.1 Discrete Straight Walks

*Discrete straight walkability* was introduced in [2] and defined as follows:

**Definition 9** (Discrete straight walkability with two guards).  $\mathcal{P}$  is discretely straight walkable with two guards with respect to two vertices  $s$  and  $t$  if two guards  $a$  and  $b$  can traverse the boundary of  $\mathcal{P}$  in the following way:

- Both guards start at vertex  $s$ .
- Guards  $a$  and  $b$  move along the polygon boundary in clockwise and counter-clockwise direction, respectively.
- The two guards are always mutually visible, i.e. the segment  $\overline{ab}$  is completely contained in  $\mathcal{P}$ .
- Only one guard is moving at a time. A movement gets a guard from one vertex  $p$  to the vertex following  $p$  in the guard's moving direction.
- At the end both guards have reached the vertex  $t$ .

$\mathcal{P}$  is discretely straight walkable with two guards if there are two vertices  $s$  and  $t$ , such that  $\mathcal{P}$  is discretely straight walkable with two guards with respect to  $s$  and  $t$ .

In this work only *discrete straight walks* are discussed, so for ease of notation and understanding we will always write *walks* for *discrete straight walks* and *walkable* for *discretely straight walkable with two guards* (unless stated otherwise).

### 3 Foundations - Existing Work and Basic Properties

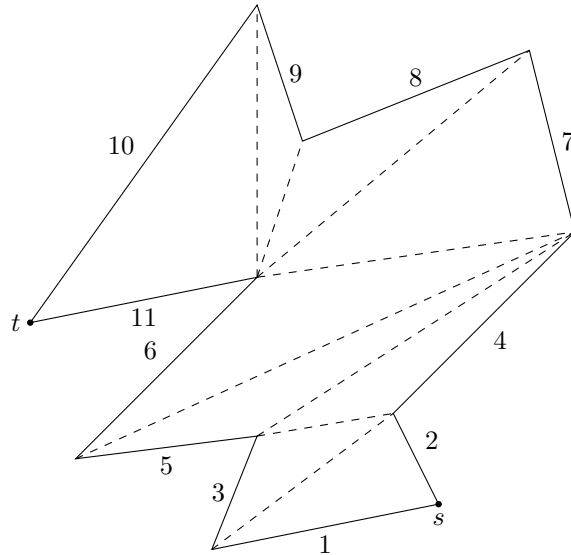


Figure 5: Walk example

**Example 1** (Example walk). Figure 5 shows an example of a walk in a polygon from a starting vertex  $s$  to a target vertex  $t$ . The walk instructions for the two guards are represented by the numbers on the edges, i.e. in step  $k$  the edge with the number  $k$  is walked by the guard associated with the corresponding boundary chain from  $s$  to  $t$ .

The dashed lines illustrate the walk by showing the “stopping points” of the guards between the walk instructions. Note that the triangulation created by these lines makes it possible to almost uniquely reconstruct the walk instructions. Ambiguity can only arise at  $s$  and  $t$ . When  $s$  is a vertex of only one triangle, then either guard can be the one that started first. Ambiguity at  $t$  can arise analogously.

There is a connection between the walkability of a polygon and the structure of its triangulation axis (as defined in [1]). To discuss this, we first recap this paper’s definition about types of triangles in a triangulation.

**Definition 10** (Types of triangles in a triangulation (adapted from [1])). Let  $T$  be a triangulation of  $\mathcal{P}$ . Let  $t \in T$  be a triangle.  $t$  is called an ear triangle if 2 of its

### 3 Foundations - Existing Work and Basic Properties

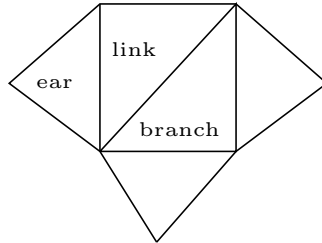


Figure 6: Triangulation with ear, link and branch triangles.

edges are from  $\mathcal{P}$ 's boundary, a link triangle if 1 of its edges is from  $\mathcal{P}$ 's boundary and a branch triangle if none of its edges are from  $\mathcal{P}$ 's boundary.

See Figure 6 for an illustration of the various triangle types.

To keep a triangulation axis structurally simple it is desirable to keep the number of branching triangles low. The next theorem shows a relationship between branching triangles and walkability.

**Theorem 2.**  $\mathcal{P}$  has a triangulation without branching triangles if and only if  $\mathcal{P}$  is walkable.

*Proof.*

$\Rightarrow$ : Let  $\mathcal{P}$  be a polygon with at least 4 vertices (other cases are trivial). Let  $T$  be a triangulation of  $\mathcal{P}$  without branching triangles. Then  $T$  has exactly two ear triangles  $E_1$  and  $E_2$ . We now construct a walk using  $T$ . Let  $s$  and  $t$  be the ear vertices of  $E_1$  and  $E_2$  respectively. Both guards  $a$  and  $b$  start at  $s$ ,  $a$  moves in clockwise direction and  $b$  in counterclockwise direction. The two guards move according to the triangulation. At first both guards walk to the next vertices of the starting ear triangle in their respective direction (one after the other), maintaining visibility because they are still on the ear triangle. Until they reach the other ear triangle  $E_2$ , the guards will always be on vertices of link triangles. That means there will always be one guard who can move to the next vertex in his direction along the edge of a link triangle that is part of the boundary of  $\mathcal{P}$ . This move maintains the guards' visibility because they are always on vertices of the same triangle. The guards continue in this way

### 3 Foundations - Existing Work and Basic Properties

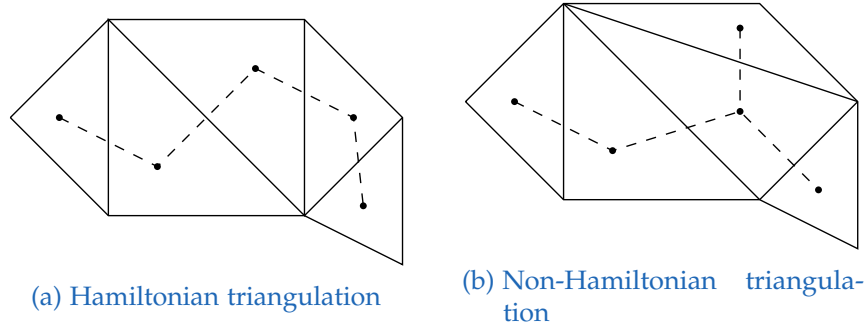


Figure 7: Examples for Hamiltonian and non-Hamiltonian triangulations

until they reach the ear triangle  $E_2$  where they simply move on to  $t$  (one after the other), again maintaining visibility because they are on the same triangle.

$\Leftarrow$ : Given a walk of two guards  $a$  and  $b$  we can construct a triangulation  $T$  without branch triangles in the following way. After each movement we add the edge  $\overline{ab}$  to  $T$  ( $\overline{ab}$  is completely contained in  $\mathcal{P}$  because  $a$  is visible from  $b$ ). The first inner edge we add to the triangulation completes the starting ear triangle  $E_1$ . Until  $a$  and  $b$  reach the vertices of the ending ear triangle  $E_2$  each new edge completes a new link triangle because only one guard moved. Thus no branch triangles are created and  $T$  is a triangulation without branch triangles.  $\square$

Walkability of  $\mathcal{P}$  turns out to be equivalent to  $\mathcal{P}$  having a Hamiltonian triangulation as has been shown in [2].

**Definition 11** (Hamiltonian triangulation). *A Hamiltonian triangulation of  $\mathcal{P}$  is a triangulation whose dual graph contains a hamiltonian path.*

See Figure 7 for examples of Hamiltonian and non-Hamiltonian triangulations.

Note that if a triangulation of a polygon is Hamiltonian, this means that its dual graph is a path.

**Theorem 3** ([2]).  *$\mathcal{P}$  is walkable if and only if  $\mathcal{P}$  has a hamiltonian triangulation.*

### 3 Foundations - Existing Work and Basic Properties

Using the concepts of *LR*-visibility, semi-wedges and forward and backward deadlocks it is possible to characterize walkable polygons as shown in [10] building on work from [9]:

**Theorem 4** (Walkability of polygons).  *$\mathcal{P}$  is walkable with respect to two vertices  $s$  and  $t$  if and only if the following conditions are met:*

- *$\mathcal{P}$  is *LR*-visible with respect to  $s$  and  $t$ .*
- *$\mathcal{P}$  has no forward or backward deadlocks with respect to  $s$  and  $t$ .*
- *$\mathcal{P}$  has no semi-wedges on the boundary chains  $L \setminus \{t\}$  and  $R \setminus \{t\}$ .*

## 3.2 Previous Results

Several works on walkability have been published, in this section we give a brief overview of their results.

It started in 1992 with the paper *The Two Guards Problem* [9] by Icking and Klein where walks were introduced. They presented solutions for the straight walk and counter-walk problem (one guard starts from  $s$  and one from  $t$ ) with a running time of  $\mathcal{O}(n \log n)$ . Note that straight walks differ from discrete straight walks in allowing the guards to stop on edges in between vertices. Further Icking and Klein showed how to solve the general walk problem (guards are allowed to backtrack) in time  $\mathcal{O}(n \log n + k)$ , where  $k$  is the size of the output which can be  $\Theta(n^2)$ .

In 1996 Heffernan [8] improved on the work by Icking and Klein and provided  $\mathcal{O}(n)$  time solutions for the straight walk and counter walk problem, as well as for the decision version of the general walk problem. He did so by reducing the number of ray shooting queries and using shortest path trees to answer them.

Discrete straight walks were introduced in 1996 by Arkin, Held, Mitchell and Skiena [2], focusing on the triangulation corresponding to a discrete straight walk. They introduced the term *Hamiltonian triangulation* for such triangulations and gave an algorithm to decide whether a polygon has a vertex pair that admits a discrete straight walk with a running time linear in the size of its visibility graph.

### 3 Foundations - Existing Work and Basic Properties

In 1998 Tseng, Heffernan and Lee [14] explored the natural generalization of finding all pairs  $(s, t)$  such that a polygon has a (straight) walk or (straight) counter-walk.  $\mathcal{O}(n \log n)$  solutions are given to compute all (straight) walkable pairs, and  $\mathcal{O}(n \log n + m)$  solutions to compute all (straight) counter-walkable pairs ( $m$  is  $\mathcal{O}(n^2)$ ).

In 1999 Narasimhan [10] presented  $\mathcal{O}(n)$  algorithms to test a vertex pair for discrete straight walkability and to compute a discrete straight walk in case of existence. Also, an algorithm to compute all vertex pairs allowing discrete straight walks in  $\mathcal{O}(n \log n)$  time is given.

Finally, in 2001 Bhattacharya, Mukhopadhyay and Narasimhan [4] presented optimal, linear-time algorithms to compute all vertex pairs that allow walks, straight walks or discrete straight walks.

## 4 Characterizing Maximal Walks

Even if a polygon is not walkable with respect to  $s$  and  $t$ , two guards can still start their walk from  $s$  and try to reach as far as possible. Eventually they will get stuck at two vertices: neither of the guards can move to the next vertex while maintaining visibility with the other guard. In this section a characterization of those maximally reachable vertex pairs is developed, relying on new extensions of the concepts  $LR$ -visibility, deadlocks and (semi)-wedges only referencing a start vertex (in contrast to a start and target vertex pair).

### 4.1 Induced Polygons

**Definition 12** (Induced polygon). *Let  $p$  and  $q$  be two vertices of  $\mathcal{P}$  such that  $s <_{cw} p <_{cw} q$  and  $\overline{pq} \subset \mathcal{P}$ . Let  $L$  be the clockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $p$  and  $R$  be the counterclockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $q$ . Let  $\mathcal{Q}$  be the polygon with the boundary consisting of  $L$ ,  $\overline{pq}$  and  $R$ . We call  $\mathcal{Q}$  the polygon induced by  $p$  and  $q$  with respect to  $s$ .*

In the following we are interested in the walkability of induced polygons. Note that a walk of an induced polygon  $\mathcal{Q}$  corresponds to a triangulation  $T$  whose dual is a path, and that  $T$  is a triangulation of a part of  $\mathcal{P}$ . If the triangulation  $T$  in  $\mathcal{P}$  can not be extended by adding another diagonal of  $\mathcal{P}$ , we say that the pair  $(l, r)$  is a maximal walkable pair of  $\mathcal{P}$  with respect to  $s$ .

The new edge  $\overline{pq}$  of an induced polygon  $\mathcal{Q}$  can prevent walkability of  $\mathcal{Q}$ . In particular, as shown in the following lemma,  $\mathcal{Q}$  can have semi-wedges that were not present in  $\mathcal{P}$ .



## 4 Characterizing Maximal Walks

**Lemma 1** (Semi-wedges in induced polygons). *Let  $\mathcal{Q}$  be the polygon induced by two vertices  $l$  and  $r$  of  $\mathcal{P}$  ( $l <_{cw} r$ ) with respect to  $s$ , such that the boundary chains from  $s$  to  $l$  in clockwise, and from  $s$  to  $r$  in counterclockwise direction do not contain any semi-wedges. Then  $\mathcal{Q}$  may have a semi-wedge but no wedge on its boundary chain.*

*Proof.* Assume that  $p$  and  $q$  ( $p <_{cw} q$ ) form a wedge on the clockwise boundary chain of  $\mathcal{Q}$  from  $s$  to  $l$ . Then  $\overline{p\text{Backw}_{cw}(p)}$  and  $\overline{q\text{Forw}_{cw}(q)}$  intersect inside  $\mathcal{Q}$  and thus also inside  $\mathcal{P}$ . Therefore  $p$  and  $q$  also form a wedge on the boundary of  $\mathcal{P}$ , a contradiction.  $\square$

To refer to these special semi-wedges more concisely we introduce a name for them:

**Definition 13** (Induced semi-wedges). *Let  $p$  and  $q$  be two vertices of  $\mathcal{P}$ ,  $p <_{cw} q$ , such that  $p$  and  $q$  don't form a semi-wedge on the boundary of  $\mathcal{P}$ . Let further  $l$  and  $r$  be two vertices of  $\mathcal{P}$ ,  $l <_{cw} r$ , such that  $p <_{ccw} r$  ( $q <_{cw} l$ ) and let  $\mathcal{Q}$  be the polygon induced by  $l$  and  $r$ . If  $p$  and  $q$  form a semi-wedge on the boundary of  $\mathcal{Q}$  then we call  $(p, q)$  an induced semi-wedge.*

See Figure 8 for an example of a semi-wedge on the boundary of  $\mathcal{Q}$  but not on the boundary of  $\mathcal{P}$ . The reason for the occurrence of this semi-wedge is that the vertex preventing the semi-wedge in  $\mathcal{P}$  is missing in  $\mathcal{Q}$ .

### 4.2 LR-Visibility With Respect to $s$

In this section the concept of  $LR$ -visibility is adapted for the purposes of maximal walks. Instead of referencing a start vertex  $s$  and a target vertex  $t$ ,  $LR$ -visibility will be defined with respect to  $s$  and two vertices  $l$  and  $r$ , the respective target points of the clockwise and counterclockwise guards.

Before introducing this adapted version of  $LR$ -visibility and studying its properties, we briefly study the original version.

#### 4 Characterizing Maximal Walks

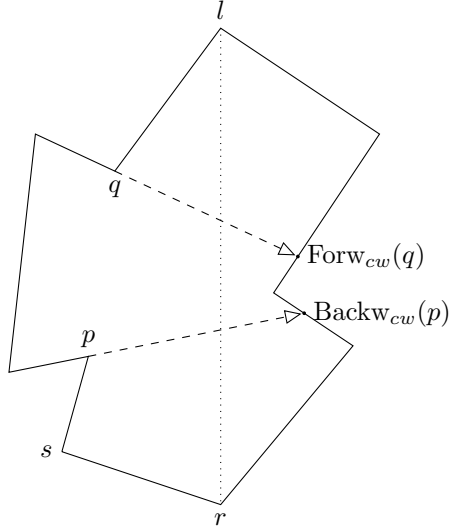


Figure 8:  $p$  and  $q$  form a semi-wedge in the polygon induced by  $l$  and  $r$  with respect to  $s$ , but not in the original polygon.

**Lemma 2** (Connectedness of LR-visible vertices). *Let  $u, v$  and  $w$  be three vertices of  $\mathcal{P}$ ,  $u <_{cw} v <_{cw} w$ . If  $\mathcal{P}$  is LR-visible with respect to  $(s$  and  $u)$  and LR-visible with respect to  $(s$  and  $w)$ , then  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $v$ .*

See Figure 9 for an illustration of this situation.

*Proof.* For  $x \in \{u, v, w\}$  let  $L_x$  be the clockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $x$  and  $R_x$  be the counterclockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $x$ .

Since  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $u$ , we know that  $R_u$  is visible from  $L_u$ .  $R_v \subset R_u$ , therefore  $R_v$  is visible from  $L_u$ . And  $L_u \subset L_v$  gives us that  $R_v$  is visible from  $L_v$ .

Similarly we get that  $L_v$  is visible from  $R_v$ , thus  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $v$ .  $\square$

**Definition 14.** *We say that a pair  $(l, r)$  of vertices of  $\mathcal{P}$  is LR-visible with respect to  $s$ , if the following properties hold:*

#### 4 Characterizing Maximal Walks

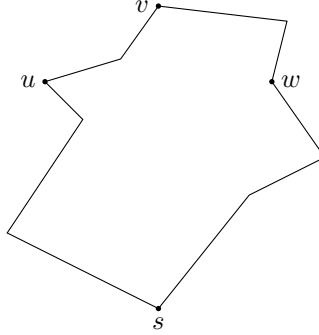


Figure 9: Illustration for Lemma 2 on the connectedness of  $LR$ -visibility

- $l <_{cw} r$
- The chains in clockwise direction from  $s$  to  $l$  and in counter-clockwise direction from  $s$  to  $r$  are  $LR$ -visible.

**Lemma 3** (Order on  $LR$ -visibility). Let  $(l, r)$  and  $(\tilde{l}, \tilde{r})$  be two different vertex-pairs of  $\mathcal{P}$  that are  $LR$ -visible with respect to  $s$ . If  $l \leq_{cw} \tilde{l} \leq_{cw} r \leq_{cw} \tilde{r}$  (note that at least one of those inequalities is strict), then the longer chains from  $s$  to  $\tilde{l}$  and  $s$  to  $r$  are also  $LR$ -visible with respect to  $s$ , i.e.  $(\tilde{l}, r)$  is  $LR$ -visible with respect to  $s$ .

See Figure 10 for an illustration of this situation.

*Proof.* Let  $L$  and  $\tilde{L}$  be the clockwise boundary chains of  $\mathcal{P}$  from  $s$  to  $l$  and  $\tilde{l}$  respectively, let  $R$  and  $\tilde{R}$  be the analogously defined counterclockwise chains.

Every point on  $\tilde{L}$  is visible from  $\tilde{R}$ . Since  $\tilde{R} \subseteq R \Rightarrow$  every point on  $\tilde{L}$  is visible from  $R$ . Every point on  $R$  is visible from  $L$ . Since  $L \subseteq \tilde{L} \Rightarrow$  every point on  $R$  is visible from  $\tilde{L}$ . Thus  $(\tilde{l}, r)$  is  $LR$ -visible with respect to  $s$ .  $\square$

The vertices  $l_{max}$  and  $r_{max}$  from the following definition indicate how far the guards starting from  $s$  can maintain  $LR$ -visibility on the polygon defined by their walks towards each other. If  $\mathcal{P}$  is not  $LR$ -visible with respect to  $s$ , then there is a maximal vertex pair that is visible with respect to  $s$ . Otherwise, if there is a vertex  $t$  such that  $\mathcal{P}$  is  $LR$ -visible with respect to  $s$  and  $t$ , then there is (due to Lemma 2) a single interval of vertices such that  $\mathcal{P}$  is  $LR$ -visible with respect to  $s$  and every vertex in this interval.

## 4 Characterizing Maximal Walks

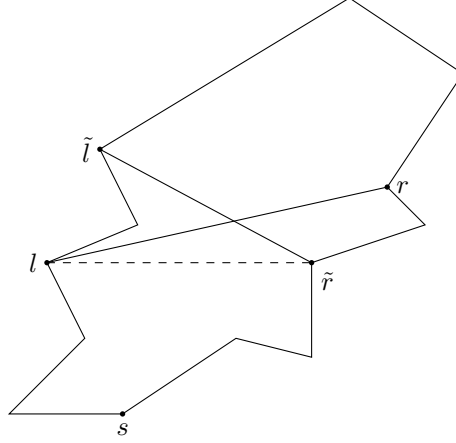


Figure 10: Illustration for Lemma 3 on the ordering of LR-visibility

**Definition 15.** We define  $l_{max}$  and  $r_{max}$  as vertices of  $\mathcal{P}$  corresponding to maximal LR-visibility of  $\mathcal{P}$ . Two cases need to be distinguished, with  $(l_{max}, r_{max})$  fulfilling different properties:

1.  $\mathcal{P}$  is not LR-visible with respect to  $s$ :
  - $(l_{max}, r_{max})$  are LR-visible with respect to  $s$ .
  - If  $l$  and  $r$  are two vertices of  $\mathcal{P}$  with  $l <_{cw} r$  such that  $(l, r)$  are LR-visible with respect to  $s$ , then  $l \leq_{cw} l_{max}$  and  $r \leq_{ccw} r_{max}$ .
2.  $\mathcal{P}$  is LR-visible with respect to  $s$ :
  - $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$  for every vertex  $t$  such that  $r_{max} \leq_{cw} t \leq_{cw} l_{max}$ .
  - If  $t$  is a vertex of  $\mathcal{P}$  such that  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$  then  $r_{max} \leq_{cw} t \leq_{cw} l_{max}$ .

Note that  $(l_{max}, r_{max})$  is well-defined because of Lemma 2 and Lemma 3.

The following lemmas provide necessary and sufficient conditions for  $l_{max}$  and  $r_{max}$ . For illustrations see Figure 11.

**Lemma 4.** Let  $p$  be a reflex vertex of  $\mathcal{P}$  satisfying  $p >_{cw} \text{Forw}_{cw}(p)$ . Then the following holds:

#### 4 Characterizing Maximal Walks

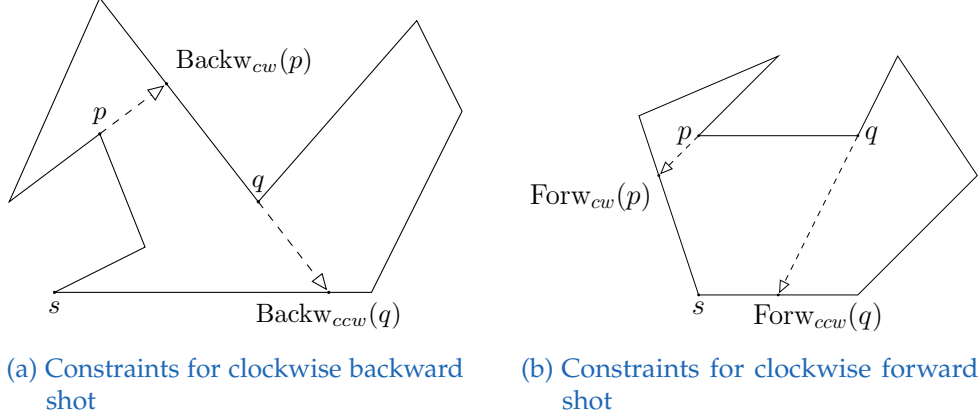


Figure 11: Illustrations for  $LR$ -visibility constraints

- (i)  $l_{max} \leq_{cw} p$ .
- (ii) If  $r_{max} <_{ccw} p$  then  $l_{max} <_{cw} \text{Pred}_{cw}(p)$ .

Analogous properties hold for  $r_{max}$ .

*Proof.* Let  $p >_{cw} \text{Forw}_{cw}(p)$  and  $\tilde{p} = \text{Pred}_{cw}(p)$ .

Ad (i): If  $l_{max} >_{cw} p$ , then there must be a vertex  $r \geq_{cw} l_{max}$  such that the polygon induced by  $l_{max}$  and  $r$  is  $LR$ -visible with respect to  $s$ . But because of  $p <_{cw} l_{max}$  and  $p <_{cw} \text{Forw}_{cw}(p)$  a necessary condition for  $LR$ -visibility is violated, a contradiction. Therefore we have  $l_{max} \leq_{cw} p$ . Note that  $l_{max} = p$  is possible as long as  $r_{max} \geq_{ccw} p$ .

Ad (ii): Let  $r_{max} <_{ccw} p$ . Then  $\tilde{p}$  is not visible from the counterclockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $r_{max}$  and therefore  $l_{max} <_{cw} \tilde{p}$ .  $\square$

**Lemma 5.** Let  $p$  be a reflex vertex of  $\mathcal{P}$  satisfying  $p <_{cw} \text{Backw}_{cw}(p)$ . Then the following holds:

- (i)  $l_{max} \leq_{cw} \text{Backw}_{cw}(p)$ .
- (ii) If  $r_{max} <_{ccw} \text{Backw}_{cw}(p)$  then  $l_{max} \leq_{cw} p$ .

Analogous properties hold for  $r_{max}$ .

## 4 Characterizing Maximal Walks

*Proof.* Let  $p <_{cw} \text{Backw}_{cw}(p)$  and  $\tilde{p} = \text{Succ}_{cw}(p)$ .

Ad (i): If  $l_{max} >_{cw} \text{Backw}_{cw}(p)$  holds, then there must be a vertex  $r \geq_{cw} l_{max}$  such that the polygon induced by  $l_{max}$  and  $r$  is LR-visible with respect to  $s$ . But because of  $\text{Backw}_{cw}(p) <_{cw} l_{max}$  and  $p <_{cw} \text{Backw}_{cw}(p)$ , a necessary condition of LR-visibility is violated, a contradiction. Thus  $l_{max} \leq_{cw} \text{Backw}_{cw}(p)$ .

Ad (ii): Let  $r_{max} <_{ccw} \text{Backw}_{cw}(p)$ . Then  $\tilde{p}$  is not visible from the counterclockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $r_{max}$ , therefore  $l_{max} \leq_{cw} p$ .  $\square$

**Lemma 6.** *Let  $l$  and  $r$  be two vertices satisfying the conditions in Lemma 4 and Lemma 5 for  $l_{max}$  and  $r_{max}$ , respectively. Then the following properties hold:*

1. *If  $l <_{cw} r$ , let  $\mathcal{Q}$  be the polygon induced by  $l$  and  $r$ . If  $\overline{lr} \subset \mathcal{P}$  then  $\mathcal{Q}$  is LR-visible with respect to  $(s$  and  $l)$  and  $(s$  and  $r)$ , i.e. the boundary chains from  $s$  to  $l$  and from  $s$  to  $r$  in clockwise respectively counterclockwise direction are mutually visible in  $\mathcal{Q}$ .*
2. *If  $r \leq_{cw} l$  then  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$  for every vertex  $t$  satisfying  $r \leq_{cw} t \leq_{cw} l$ .*

*Proof.* Ad 1: Assume that there exists a vertex  $p$  of  $\mathcal{Q}$  with  $s <_{cw} p <_{cw} l$  such that  $p <_{cw} \text{Backw}_{cw}^{\mathcal{Q}}(p) \leq_{cw} r$ . If  $\text{Backw}_{cw}^{\mathcal{Q}}(p) = \text{Backw}_{cw}^{\mathcal{P}}(p)$  then  $\text{Backw}_{cw}^{\mathcal{P}}(p) \leq_{cw} l \Rightarrow (s, l)$  and  $(s, r)$  are not LR-visible in  $\mathcal{P}$ , a contradiction. So let  $\text{Backw}_{cw}^{\mathcal{Q}}(p) \neq \text{Backw}_{cw}^{\mathcal{P}}(p)$ . Then  $\text{Backw}_{cw}^{\mathcal{Q}}(p) \in \overline{lr} \setminus \{l, r\}$ . So  $l <_{cw} \text{Backw}_{cw}^{\mathcal{P}}(p) <_{cw} r$  and with Lemma 5 we get  $l_{max} \leq_{cw} p$ , a contradiction to  $s <_{cw} p <_{cw} l$ .

Now let's assume that there exists a vertex  $p$  of  $\mathcal{Q}$  with  $s <_{cw} p <_{cw} l$  such that  $p >_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(p) \geq_{cw} s$ . Similar to before, if  $\text{Forw}_{cw}^{\mathcal{Q}}(p) = \text{Forw}_{cw}^{\mathcal{P}}(p) \Rightarrow (s, l)$  and  $(s, r)$  are not LR-visible in  $\mathcal{P}$ , a contradiction. So let  $\text{Forw}_{cw}^{\mathcal{Q}}(p) \neq \text{Forw}_{cw}^{\mathcal{P}}(p)$ . Then  $l <_{cw} \text{Forw}_{cw}^{\mathcal{P}}(p) <_{cw} r$ . Since  $p\text{Forw}_{cw}^{\mathcal{Q}}(p) \supset p\text{Forw}_{cw}^{\mathcal{P}}(p)$  we have that  $\text{Forw}_{cw}^{\mathcal{Q}}(p)$  lies inside of  $\mathcal{Q}$ . Therefore the boundary chain of  $\mathcal{P}$  from  $l$  to  $r$  in clockwise direction intersects with  $\overline{lr}$ , a contradiction to  $\overline{lr} \subset \mathcal{P}$ .

Ad 2: Let  $r \leq_{cw} l$ ,  $t$  a vertex of  $\mathcal{P}$  such that  $r \leq_{cw} t \leq_{cw} l$  and  $p$  a reflex vertex on  $L$ . To show that  $\mathcal{P}$  is LR-visible with respect to  $s$  and  $t$  we have to show that the conditions of Theorem 1 are satisfied. First we consider the two edge cases:

## 4 Characterizing Maximal Walks

- $t = r$ : If  $p <_{cw} \text{Backw}_{cw}(p) \in L \setminus \{t\}$ , then Lemma 5 gives us  $l \leq_{cw} \text{Backw}_{cw}(p)$ . But then  $l \leq_{cw} r$  and thus  $l = r = t = \text{Backw}_{cw}(p)$ , a contradiction. If  $p \neq t$  and  $p >_{cw} \text{Forw}_{cw}(p) \in L$ , then Lemma 4 gives us  $l \leq_{cw} p$ . But then  $l = r = p = t$ , a contradiction.
- $t = l$ : If  $p <_{cw} \text{Backw}_{cw}(p) \in L \setminus \{t\}$ , then Lemma 5 gives us  $l \leq_{cw} \text{Backw}_{cw}(p)$ . But then  $t = l = \text{Backw}_{cw}(p)$ , a contradiction. If  $p \neq t$  and  $p >_{cw} \text{Forw}_{cw}(p) \in L$ , then Lemma 4 gives us  $l \leq_{cw} p$ . But then  $t = l = p$ , a contradiction.

The analog cases for reflex vertices on  $R$  work similarly. Therefore  $\mathcal{P}$  is  $LR$ -visible with respect to  $(s$  and  $r)$  and  $(s$  and  $l)$ . Now we consider  $r <_{cw} t <_{cw} l$ . Using Lemma 2 we get  $LR$ -visibility of  $\mathcal{P}$  with respect to  $s$  and  $t$  for this case as well.  $\square$

### 4.3 Semi-Wedges With Respect to $s$

Similarly to  $LR$ -visibility, the definition of (semi-)wedges references a start vertex  $s$  and a target vertex  $t$ . Again, for the purposes of maximal walks, the definition needs to be adapted to only reference  $s$ , as there may not be a common target vertex  $t$  for both guards.

Since our maximal walks are with respect to discrete walkability, it is not necessary to distinguish between wedges and semi-wedges. So from now on only the more general term *semi-wedge* will be used. After the definition of semi-wedges is adapted, their properties are studied.

**Definition 16** (Semi-wedge (adapted from [9] and [10])). *Let  $\mathcal{P}$  be a polygon,  $p$  and  $q$  reflex vertices of  $\mathcal{P}$ .  $p$  and  $q$  form a clockwise semi-wedge with respect to  $s$  if*

1.  $p <_{cw} q$ , and
2.  $\text{Backw}_{cw}(p) >_{cw} q$ , and
3.  $\text{Forw}_{cw}(q) >_{cw} q$ , and
4. one of the following:
  - a)  $\text{Forw}_{cw}(q) >_{cw} \text{Backw}_{cw}(p)$ , or
  - b)  $\text{Forw}_{cw}(q) \leq_{cw} \text{Backw}_{cw}(p)$ , but there is no vertex  $q$  of  $\mathcal{P}$  such that  $\text{Forw}(q) \leq_{cw} q \leq_{cw} \text{Backw}_{cw}(p)$ .

#### 4 Characterizing Maximal Walks

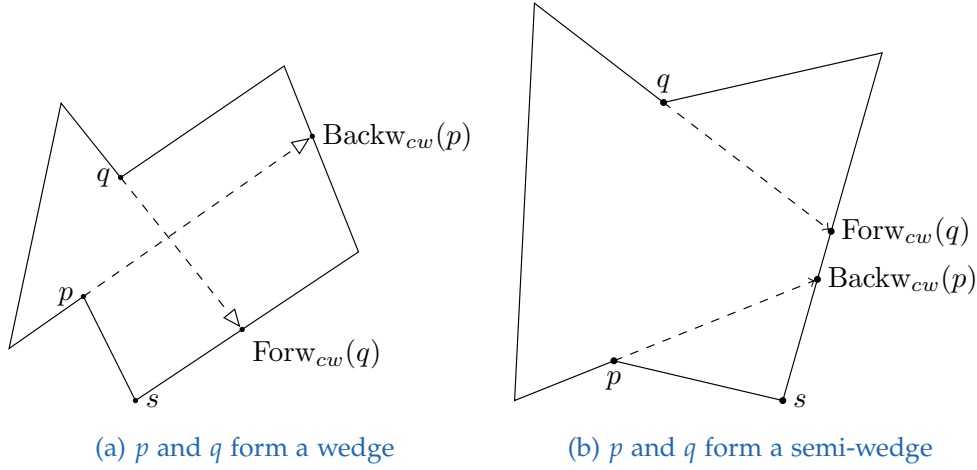


Figure 12: Semi-wedges with respect to a starting vertex  $s$

Counterclockwise semi-wedges with respect to  $s$  are defined analogously.

See Figure 12 for examples of semi-wedges with respect to a start vertex  $s$ .

**Definition 17** ((Non-)redundant semi-wedges). A semi-wedge  $(p, q)$  with respect to  $s$  is called *redundant*, if there is another semi-wedge  $(\tilde{p}, \tilde{q}) \neq (p, q)$  with respect to  $s$  with  $p \leq_{cw} \tilde{p} < \tilde{q} \leq_{cw} q$ , i.e. it contains another semi-wedge. A semi-wedge that is not redundant is called *non-redundant*.

Clearly a guard can not get past a semi-wedge, so for maximal walkability only the first semi-wedge encountered by a scan from  $s$  in a guard's moving direction matters. These semi-wedges are called *minimal*:

**Definition 18** (Minimal semi-wedges). A clockwise semi-wedge  $(p, q)$  is *minimal* with respect to  $s$ , if for any other clockwise semi-wedge  $(\tilde{p}, \tilde{q})$  one of the following holds:

$$q <_{cw} \tilde{q}, \text{ or}$$

$$\tilde{q} = q \text{ and } p >_{cw} \tilde{p}$$

The minimal counterclockwise semi-wedge is defined analogously.



## 4 Characterizing Maximal Walks

Note that this definition selects the minimal non-redundant semi-wedge out of potentially many minimal semi-wedges having the same second vertex  $q$ .

In [14] an algorithm is presented that computes all non-redundant semi-wedges (semi-wedges that do not fully contain another semi-wedge) of a polygon. The ideas of this algorithm are adapted to compute the minimal semi-wedges from a starting vertex.

**Algorithm 1** (Finding minimal semi-wedges. Adapted from [14]). We describe the algorithm to find the minimal clockwise semi-wedge. Finding the minimal counterclockwise semi-wedge works analogously.

The idea of the algorithm is to scan the boundary of  $\mathcal{P}$  in clockwise direction starting from  $s$ . During the scan, each reflex vertex with  $\text{Backw}_{cw}(p)$  (and thus a possible member of the minimal clockwise semi-wedge) is remembered in a list. When the hit point of its backward shot is encountered during the scan, it is removed from the list again. For each reflex vertex it is also checked whether it forms a semi-wedge with any of the reflex vertices in the list. If it does, the minimal clockwise semi-wedge is reported.

The pseudo code is shown in Algorithm Listing 1.

**Theorem 5** (Correctness of Algorithm 1). *Algorithm 1 finds the minimal clockwise semi-wedge in  $\mathcal{P}$  with respect to  $s$ .*

*Proof.* Let  $(\tilde{p}, \tilde{q})$  be the minimal clockwise semi-wedge in  $\mathcal{P}$  with respect to  $s$ ,  $\tilde{p} <_{cw} \tilde{q}$ . If the list *marked* contains  $\tilde{p}$  at the time the algorithm processes the vertex  $\tilde{q}$ , then the algorithm will report  $(\tilde{p}, \tilde{q})$ . Because  $\tilde{p} <_{cw} \tilde{q}$  and  $\text{Backw}_{cw}(\tilde{p}) >_{cw} \tilde{p}$  (since  $\text{Backw}_{cw}(\tilde{p}) >_{cw} \tilde{q}$ ) we know that  $\tilde{p}$  has been added to *marked*. And because  $\text{Backw}_{cw}(\tilde{p}) >_{cw} \tilde{q}$ , we know that  $\tilde{p}$  has not been removed from *marked*. Also, since  $(\tilde{p}, \tilde{q})$  is minimal, we know that the algorithm has not reported another semi-wedge before and actually reaches this state. Thus the algorithm correctly reports  $(\tilde{p}, \tilde{q})$ .

If  $\mathcal{P}$  has no clockwise semi-wedge, then the algorithm does not report a semi-wedge, therefore the algorithm is correct.  $\square$

**Theorem 6** (Running time of Algorithm 1). *Let  $\mathcal{P}$  have  $n$  vertices. Algorithm 1 can be implemented with a running time of  $\mathcal{O}(n \log n)$ .*

## 4 Characterizing Maximal Walks

---

**Algorithm Listing 1** Compute the minimal semi-wedge

---

The function  $next()$  gives the next relevant point on the boundary of  $\mathcal{P}$  in clockwise direction. If  $p$  is a reflex vertex, then  $p$  and  $Backw_{cw}(p)$  are relevant points.

```
1:  $marked \leftarrow$  empty list
2:  $start \leftarrow next()$ 
3:  $q \leftarrow start$ 
4: repeat
5:   if  $q$  is a reflex vertex and  $Backw_{cw}(q) >_{cw} q$  then
6:     Add  $q$  to  $marked$ 
7:   else if  $q$  is a reflex vertex and forms a semi-wedge with a vertex  $p$  in
    $marked$  then
8:     Find the maximal  $p$  in  $marked$  that forms a semi-wedge with  $q$ .
9:     Report  $(p, q)$  and abort.
10:  else if  $q = Backw_{cw}(p)$  for some vertex  $p$  in  $marked$  then
11:    Remove  $p$  from  $marked$ .
12:  end if
13:   $q \leftarrow next()$ 
14: until  $q = start$   $\triangleright \mathcal{P}$  does not have a semi-wedge
```

---

## 4 Characterizing Maximal Walks

*Proof.* Preprocessing to compute all necessary ray shots from reflex vertices in  $\mathcal{P}$  can be done in  $\mathcal{O}(n \log n)$  time as shown in [6].

There are at most  $n$  vertices inserted into and removed from the list *marked*, so we can maintain *marked* as a sorted list in  $\mathcal{O}(n \log n)$  time (e.g. with red-black trees). The sorting of the reflex vertices in *marked* is with respect to the clockwise order of the clockwise backward shots of those vertices.

Checking if a reflex vertex  $q$  forms a clockwise semi-wedge with a vertex  $p$  in *marked* can be done in  $\mathcal{O}(\log n)$  time using binary search in *marked*.

So every step of the loop can be done in  $\mathcal{O}(\log n)$  time, therefore the overall running time is  $\mathcal{O}(n \log n)$ .  $\square$

The vertices  $l_{max}$  and  $r_{max}$  in the following definition indicate how far two guards can walk from a starting vertex  $s$  without encountering a semi-wedge (clockwise or counterclockwise).

**Definition 19.** We define  $l_{max}$  and  $r_{max}$  to be vertices of  $\mathcal{P}$  corresponding to maximal semi-wedge free boundary chains with respect to  $s$ . Two cases need to be distinguished, with  $(l_{max}, r_{max})$  fulfilling different properties:

1.  $\mathcal{P}$  has at least two disjoint semi-wedges with respect to  $s$ :
  - $l_{max} <_{cw} r_{max}$ ,  $\overline{l_{max}r_{max}} \subset \mathcal{P}$  and the polygon induced by  $l_{max}$  and  $r_{max}$  is free of semi-wedges with respect to  $s$  except induced semi-wedges on the clockwise respectively counterclockwise boundary chains from  $s$  to  $l_{max}$  and  $r_{max}$ .
  - If  $l$  and  $r$  are two vertices of  $\mathcal{P}$  with  $l <_{cw} r$ ,  $\overline{lr} \subset \mathcal{P}$  and the clockwise respectively counterclockwise boundary chains from  $s$  to  $l$  and  $r$  are free of semi-wedges with respect to  $s$  except induced semi-wedges, then  $l \leq_{cw} l_{max}$  and  $r \leq_{ccw} r_{max}$ .
2.  $\mathcal{P}$  has at most a single non-redundant semi-wedge with respect to  $s$  (if there is a single semi-wedge, then a walk target can be placed within this semi-wedge):
  - $r_{max} \leq_{cw} l_{max}$  and for every vertex  $r_{max} \leq_{cw} t \leq_{cw} l_{max}$  the clockwise respectively counterclockwise boundary chains from  $s$  to  $Pred_{cw}(t)$  and  $Pred_{ccw}(t)$  are free of semi-wedges.

## 4 Characterizing Maximal Walks

- If  $t$  is a vertex of  $\mathcal{P}$  such that the clockwise respectively counterclockwise boundary chains from  $s$  to  $\text{Pred}_{cw}(t)$  and  $\text{Pred}_{ccw}(t)$  are free of semi-wedges, then  $r_{max} \leq_{cw} t \leq_{cw} l_{max}$ .

The following lemmas provide necessary and sufficient conditions for  $l_{max}$  and  $r_{max}$ .

**Lemma 7** (Avoidance of semi-wedges). *Let  $(p, q)$  be a clockwise semi-wedge with respect to  $s$ . Then the following holds:*

1.  $l_{max} \leq_{cw} q$ .
2. If  $r_{max} <_{ccw} q$  then  $l_{max} <_{cw} q$ .

Analogous properties hold for  $r_{max}$  with respect to counterclockwise semi-wedges.

*Proof.* Ad 1: Assume  $l_{max} >_{cw} q$  and  $r \leq \min_{<_{ccw}} \{l_{max}, r_{max}\}$ . Then  $(p, q)$  is a clockwise semi-wedge on  $L$ , a contradiction.

Ad 2: Assume  $r_{max} <_{ccw} q$ . If  $l_{max} \geq_{cw} q$  then  $(p, q)$  is again a clockwise semi-wedge on  $L$ .  $\square$

From Lemma 7 it is easy to see, that avoiding the minimal clockwise semi-wedge with respect to  $s$  avoids all clockwise semi-wedges with respect to  $s$ :

**Corollary 1.** *Let  $(p, q)$  be the minimal clockwise semi-wedge with respect to  $s$ , and let  $(\tilde{p}, \tilde{q})$  be any other clockwise semi-wedge with respect to  $s$ . Then*

$$\begin{aligned} l_{max} \leq_{cw} q &\Rightarrow l_{max} \leq_{cw} \tilde{q}, \text{ and} \\ l_{max} <_{cw} q &\Rightarrow l_{max} <_{cw} \tilde{q} \end{aligned}$$

**Lemma 8.** *Let  $l$  and  $r$  be two vertices of  $\mathcal{P}$  that satisfy the conditions of Lemma 7 for  $l_{max}$  and  $r_{max}$ , respectively. Then the following properties hold:*

1. If  $l <_{cw} r$ , let  $\mathcal{Q}$  be the polygon induced by  $l$  and  $r$ . If  $\bar{l}r \subset \mathcal{P}$ , then the clockwise boundary chain of  $\mathcal{Q}$  from  $s$  to  $l$  and the counterclockwise boundary chain of  $\mathcal{Q}$  from  $s$  to  $r$  do not contain any semi-wedges except induced semi-wedges.
2. If  $r \leq_{cw} l$ , then for every vertex  $t$  of  $\mathcal{P}$  with  $r \leq_{cw} t \leq_{cw} l$  the clockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $\text{Pred}_{cw}(t)$ , and the counterclockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $\text{Pred}_{ccw}(t)$  both do not contain semi-wedges.

#### 4 Characterizing Maximal Walks

*Proof.* Ad 1: Let  $\bar{l}r \subset \mathcal{P}$ . Assume that the boundary chain of  $\mathcal{Q}$  from  $s$  to  $l$  contains a semi-wedge  $(p, q)$  that is not an induced semi-wedge, so  $s <_{cw} p <_{cw} q <_{cw} l$ . We claim that  $(p, q)$  is a semi-wedge of  $\mathcal{P}$  as well. There are two cases to consider:  $(p, q)$  can be either a semi-wedge ( $\text{Backw}_{cw}^{\mathcal{Q}}(p) <_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(q)$ ) or a ‘‘proper’’ wedge ( $\text{Backw}_{cw}^{\mathcal{Q}}(p) \geq_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(q)$ ).

Let  $(p, q)$  be a wedge of  $\mathcal{Q}$ :  $\text{Backw}_{cw}^{\mathcal{Q}}(p) <_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(q)$ . We need to show that all properties of the definition of a wedge can be extended from  $\mathcal{Q}$  to  $\mathcal{P}$ :

- (a)  $p <_{cw} q$ : This trivially holds in  $\mathcal{P}$  as well.
- (b)  $\text{Backw}_{cw}^{\mathcal{P}}(p) >_{cw} q$ : If  $\text{Backw}_{cw}^{\mathcal{Q}}(p) = \text{Backw}_{cw}^{\mathcal{P}}(p)$  we are obviously done, so let  $\text{Backw}_{cw}^{\mathcal{Q}}(p) \neq \text{Backw}_{cw}^{\mathcal{P}}(p)$ . Then we need to have  $\text{Backw}_{cw}^{\mathcal{Q}}(p) \in \bar{l}r \Rightarrow \text{Backw}_{cw}^{\mathcal{P}}(p) >_{cw} l >_{cw} q$ .
- (c)  $\text{Forw}_{cw}^{\mathcal{P}}(q) >_{cw} q$ : If  $\text{Forw}_{cw}^{\mathcal{Q}}(q) = \text{Forw}_{cw}^{\mathcal{P}}(q)$  we are done again, so let  $\text{Forw}_{cw}^{\mathcal{Q}}(q) \neq \text{Forw}_{cw}^{\mathcal{P}}(q)$ . Then we need to have  $\text{Forw}_{cw}^{\mathcal{Q}}(q) \in \bar{l}r \Rightarrow \text{Forw}_{cw}^{\mathcal{P}}(q) >_{cw} l >_{cw} q$ .
- (d)  $\text{Backw}_{cw}^{\mathcal{P}} <_{cw} \text{Forw}_{cw}^{\mathcal{P}}(q)$ :  $p <_{cw} q$  and  $\text{Backw}_{cw}^{\mathcal{Q}}(p) <_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(q)$ , so the ray shots must intersect inside of  $\mathcal{Q}$ :  $p\text{Backw}_{cw}^{\mathcal{Q}}(p) \cap q\text{Forw}_{cw}^{\mathcal{Q}}(q) = \{v\}$ , with  $v$  somewhere inside of  $\mathcal{Q}$ . Since  $\mathcal{Q} \subset \mathcal{P}$ , the ray shots of  $\mathcal{Q}$  must be part of the ray shots of  $\mathcal{P}$ :  $p\text{Backw}_{cw}^{\mathcal{Q}}(p) \subseteq p\text{Backw}_{cw}^{\mathcal{P}}(p)$  and  $q\text{Forw}_{cw}^{\mathcal{Q}}(q) \subseteq q\text{Forw}_{cw}^{\mathcal{P}}(q)$ . Therefore  $v$  must be inside of  $\mathcal{P}$  and thus  $\text{Backw}_{cw}^{\mathcal{P}}(p) <_{cw} \text{Forw}_{cw}^{\mathcal{P}}(q)$ .

So  $(p, q)$  is a wedge of  $\mathcal{P}$ , a contradiction to  $l_{max} <_{cw} q$  from Lemma 7.

Now let  $\text{Backw}_{cw}^{\mathcal{Q}}(p) \geq_{cw} \text{Forw}_{cw}^{\mathcal{Q}}(q)$ . Since  $(p, q)$  is a semi-wedge of  $\mathcal{Q}$  there exists no vertex  $u \in \mathcal{Q}$  such that  $\text{Forw}_{cw}^{\mathcal{Q}}(q) \leq_{cw} u \leq_{cw} \text{Backw}_{cw}^{\mathcal{Q}}(p)$ . Again we show that all necessary properties of  $(p, q)$  can be extended from  $\mathcal{Q}$  to  $\mathcal{P}$  such that  $(p, q)$  is a semi-wedge of  $\mathcal{P}$ :

- (a)  $p <_{cw} q$ : Obvious.
- (b)  $\text{Backw}_{cw}^{\mathcal{P}}(p) >_{cw} q$ : Identical to the wedge-case above.
- (c)  $\text{Forw}_{cw}^{\mathcal{P}}(q) >_{cw} q$ : Also identical to the wedge-case above.
- (d)  $\text{Forw}_{cw}^{\mathcal{Q}}(q)$  and  $\text{Backw}_{cw}^{\mathcal{Q}}(p)$  lie on the same edge  $e$ . If  $e$  is also an edge of  $\mathcal{P} \Rightarrow (p, q)$  is a wedge of  $\mathcal{P}$  and we are done. So suppose that  $e$  is not an edge of  $\mathcal{P} \Rightarrow e = \bar{l}r$ . If  $\text{Forw}_{cw}^{\mathcal{P}}(q) >_{cw} \text{Backw}_{cw}^{\mathcal{P}}(p)$ , then  $(p, q)$  is a wedge in  $\mathcal{P}$  and we are done again. So  $\text{Forw}_{cw}^{\mathcal{P}}(q) \leq_{cw} \text{Backw}_{cw}^{\mathcal{P}}(p)$ . If

## 4 Characterizing Maximal Walks

$\nexists u \in \mathcal{P}$ :  $\text{Forw}_{cw}^{\mathcal{P}}(q) \leq_{cw} u \leq_{cw} \text{Backw}_{cw}^{\mathcal{P}}(p)$ , then  $(p, q)$  is a semi-wedge in  $\mathcal{P}$  and we are done. So assume there is such a vertex  $u \in \mathcal{P}$ . But then  $(p, q)$  is an induced semi-wedge of  $\mathcal{Q}$ , contradicting our assumption.

So  $(p, q)$  is a semi-wedge of  $\mathcal{P}$  as well. But like above in the case of the wedge, this contradicts Lemma 7 ( $l_{max} <_{cw} q$ ), concluding the proof.

Ad 2: Assume that the clockwise boundary chain of  $\mathcal{P}$  from  $s$  to  $\text{Pred}_{cw}(t)$  contains a semi-wedge  $(p, q)$ . Then Lemma 7 demands that  $l \leq_{cw} q$ . But this gives  $q <_{cw} t \leq_{cw} l \leq_{cw} q$ , a contradiction. The case for the counterclockwise boundary chain can be proven analogously.  $\square$

### 4.4 Backward Deadlocks With Respect to $s$

For our purposes of maximal walks we need to get rid of the reference to  $t$  (and consequently of references to  $L$  and  $R$ ) in the definition of backward deadlocks. To achieve this we slightly alter the definition from above:

**Definition 20** (Backward deadlocks with respect to a start vertex). *Two vertices  $p$  and  $q$  of  $\mathcal{P}$  with  $p <_{cw} q$  form a backward deadlock with respect to  $s$  if the following conditions are fulfilled:*

1.  $\text{Backw}_{cw}(p) >_{cw} p$
2.  $\text{Backw}_{ccw}(q) >_{ccw} q$
3.  $\text{Backw}_{cw}(p) >_{ccw} q$
4.  $\text{Backw}_{ccw}(q) >_{cw} p$

**Lemma 9** (Order on backward deadlocks). *Let  $(p, q)$  and  $(\tilde{p}, \tilde{q})$  be two backward deadlocks in  $\mathcal{P}$  with respect to  $s$ . If  $p <_{cw} \tilde{p}$  and  $\tilde{q} <_{ccw} q$ , then  $p$  and  $\tilde{q}$  form a backward deadlock with respect to  $s$ .*

See Figure 13 for an example.

*Proof.* To show that  $p$  and  $\tilde{q}$  form a backward deadlock with respect to  $s$ , we have to show that the following properties are satisfied:

- (i)  $\text{Backw}_{cw}(p) >_{cw} p$
- (ii)  $\text{Backw}_{ccw}(\tilde{q}) >_{ccw} \tilde{q}$

#### 4 Characterizing Maximal Walks

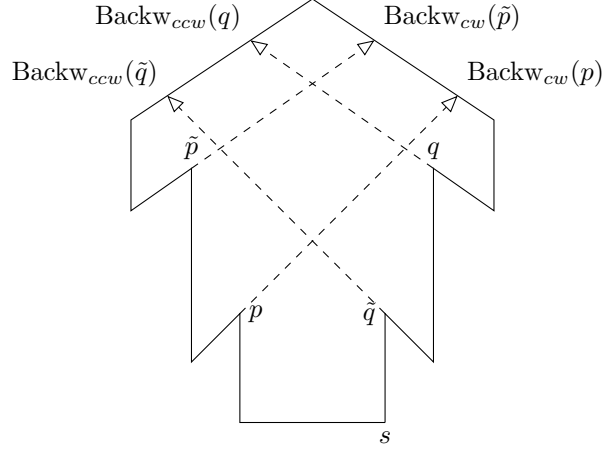


Figure 13:  $p$  and  $\tilde{q}$  form a backward deadlock with respect to  $s$ .

(iii)  $\text{Backw}_{cw}(p) >_{ccw} \tilde{q}$

(iv)  $\text{Backw}_{ccw}(\tilde{q}) >_{cw} p$

Ad (i): Since  $p$  and  $q$  form a backward deadlock with respect to  $s$ , we have  $\text{Backw}_{cw}(p) >_{cw} p$ .

Ad (ii):  $\tilde{p}$  and  $\tilde{q}$  form a backward deadlock with respect to  $s$ , so we have  $\text{Backw}_{ccw}(\tilde{q}) >_{ccw} \tilde{q}$ .

Ad (iii):  $p$  and  $q$  form a backward deadlock with respect to  $s$ , so we have  $\text{Backw}_{cw}(p) >_{ccw} q$ . With  $q >_{ccw} \tilde{q}$  we get  $\text{Backw}_{cw}(p) >_{ccw} \tilde{q}$ .

Ad (iv):  $\tilde{p}$  and  $\tilde{q}$  form a backward deadlock with respect to  $s$ , so we have  $\text{Backw}_{ccw}(\tilde{q}) >_{cw} \tilde{p}$ . With  $\tilde{p} >_{cw} p$  we get  $\text{Backw}_{ccw}(\tilde{q}) >_{cw} p$ .  $\square$

As an important consequence of this lemma there is always a unique minimal backward deadlock with respect to  $s$  (given that backward deadlocks with respect to  $s$  exist). Later it will be shown that for maximal walkability only this unique minimal backward deadlock needs to be considered.

**Corollary 2** (Minimal backward deadlock). *If there exists a backward deadlock with respect to  $s$ , then there exists a minimal backward deadlock  $(\hat{p}, \hat{q})$  with respect to  $s$ , i.e.: if  $p$  and  $q$  form a backward deadlock with respect to  $s$  then  $\hat{p} \leq_{cw} p$  and  $\hat{q} \leq_{ccw} q$ .*

## 4 Characterizing Maximal Walks

*Proof.* Suppose  $\mathcal{P}$  has no such minimal backward deadlock with respect to  $s$ . Let

$$B = (p_1, q_1), \dots, (p_k, q_k)$$

be the set of all backward deadlocks of  $\mathcal{P}$  with respect to  $s$  (note that  $k \geq 2$ , the other cases are trivial). Let

$$p_l = \min_{<_{cw}} \{p_i \mid i = 1, \dots, k\}$$

$$q_r = \min_{<_{ccw}} \{q_i \mid i = 1, \dots, k\}$$

Thus  $(p_l, q_l)$  is the first backward deadlock with respect to  $s$  from  $s$  in clockwise direction and  $(p_r, q_r)$  correspondingly the first backward deadlock with respect to  $s$  from  $s$  in counterclockwise direction. Note that  $(p_l, q_l) \neq (p_r, q_r)$ , otherwise our assumption would be violated (we would have a minimal backward deadlock with respect to  $s$ ). However, according to Lemma 9  $p_l$  and  $q_r$  also form a backward deadlock with respect to  $s$ . Since  $(p_l, q_r) \notin B$ , this is a contradiction.  $\square$

**Algorithm 2** (Find the minimal backward deadlock with respect to  $s$ ). The boundary of  $\mathcal{P}$  is scanned from  $s$  simultaneously in clockwise and counterclockwise direction using vertices  $p$  and  $q$  on the respective boundary chains.  $p$  and  $q$  take turns in being advanced on reflex vertices towards each other, each one being advanced until its backward shot is placed between the current vertices of  $p$  and  $q$ , e.g.  $p <_{cw} \text{Backw}_{cw}(p) <_{cw} q$ .

The detailed algorithm is presented in Algorithm Listing 2.

**Theorem 7** (Correctness of Algorithm 2). *If there is a backward deadlock with respect to  $s$ , then Algorithm 2 finds the minimal backward deadlock with respect to  $s$ . If there is no backward deadlock with respect to  $s$ , then the algorithm reports nothing.*

*Proof.* Let  $(\tilde{p}, \tilde{q})$  be the minimal backward deadlock with respect to  $s$ . Since the algorithm only returns when it finds a backward deadlock with respect to  $s$  or when  $p = q$ , there must be some stage where either

- (i)  $p = \tilde{p}$  and  $q <_{ccw} \tilde{q}$ , or
- (ii)  $q = \tilde{q}$  and  $p <_{cw} \tilde{p}$ .



**Algorithm Listing 2** Computing the minimal backward deadlock

At first we introduce two helper functions:

```

1: function NEXT_CANDIDATE_CW(start, q)
2:    $p \leftarrow start$ 
3:   while  $Backw_{cw}(p) \leq_{ccw} q$  or  $Backw_{cw}(p) \leq_{cw} p$  do
4:     if  $p = q$  then
5:       return  $p$ 
6:     else
7:        $p \leftarrow$  next reflex vertex of  $\mathcal{P}$  in clockwise direction after  $p$ 
8:     end if
9:   end while
10:  return  $p$ 
11: end function
12: function NEXT_CANDIDATE_CCW(start, p)
13:   $q \leftarrow start$ 
14:  while  $Backw_{ccw}(q) \leq_{cw} p$  or  $Backw_{ccw}(q) \leq_{ccw} q$  do
15:    if  $q = p$  then
16:      return  $q$ 
17:    else
18:       $q \leftarrow$  next reflex vertex of  $\mathcal{P}$  in counterclockwise direction after  $q$ 
19:    end if
20:  end while
21:  return  $q$ 
22: end function

```

Now we can formulate the algorithm:

```

1:  $p \leftarrow s$ 
2:  $q \leftarrow s$ 
3: repeat
4:    $p \leftarrow next\_candidate\_cw(p, q)$ 
5:   if  $(p, q)$  backward deadlock with respect to  $s$  then
6:     return  $(p, q)$ 
7:   else if  $p \neq q$  then
8:      $q \leftarrow next\_candidate\_ccw(q, p)$ 
9:     if  $(p, q)$  backward deadlock with respect to  $s$  then
10:      return  $(p, q)$ 
11:    end if
12:   end if
13: until  $p = q$  ▷ There is no backward deadlock with respect to  $s$ 

```

## 4 Characterizing Maximal Walks

Assume (i) holds (the case (ii) is analogous). Then we have  $\text{Backw}_{cw}(p) >_{ccw} q$  and  $\text{Backw}_{cw}(p) >_{cw} p$ . So the algorithm searches with  $\text{next\_candidate\_ccw}(q, p)$  for an appropriate  $\hat{q}$  with  $\text{Backw}_{ccw}(\hat{q}) >_{cw} p$  and  $\text{Backw}_{ccw}(\hat{q}) >_{ccw} \hat{q}$ . Since  $(\tilde{p}, \tilde{q})$  is the minimal backward deadlock with respect to  $s$ , the algorithm picks  $\tilde{q}$  as the first appropriate  $\hat{q}$  in counterclockwise direction. Thus the algorithm sets  $q$  to  $\tilde{q}$  and has successfully detected the minimal backward deadlock with respect to  $s$ .

If no backward deadlock with respect to  $s$  exists then the algorithm obviously does not report any backward deadlock (before a vertex pair  $(p, q)$  is reported it is always checked whether  $(p, q)$  actually forms a backward deadlock with respect to  $s$ ).

Finally we need to establish that the algorithm always terminates. If a backward deadlock with respect to  $s$  exists, then it is reported and the algorithm terminates. So assume there is no backward deadlock with respect to  $s$ . The functions  $\text{next\_candidate\_cw}$  and  $\text{next\_candidate\_ccw}$  always push  $p$  and  $q$  further towards each other in clockwise respectively counterclockwise direction. Thus eventually we get  $p = q$  and the termination condition of the loop is satisfied, resulting in the termination of the algorithm.  $\square$

**Theorem 8** (Running time of Algorithm 2). *Let  $\mathcal{P}$  be a polygon with  $n$  vertices. Then the running time of Algorithm 2 is  $\mathcal{O}(n)$  if the backward ray shots of reflex vertices are precomputed.*

*Proof.* In the algorithm each reflex vertex of  $\mathcal{P}$  is visited once in constant time, leading to  $\mathcal{O}(n)$  overall running time.  $\square$

The vertices  $l_{max}$  and  $r_{max}$  in the following definition indicate how far two guards can walk without encountering a backward deadlock.

**Definition 21.** *We define  $l_{max}$  and  $r_{max}$  to be vertices of  $\mathcal{P}$  satisfying the following properties:*

1.  $\mathcal{P}$  has at least one backward deadlock:
  - $l_{max} <_{cw} r_{max}$ ,  $\overline{l_{max}r_{max}} \subset \mathcal{P}$  and the polygon  $\mathcal{Q}$  induced by  $l_{max}$  and  $r_{max}$  with respect to  $s$  does not contain any backward deadlock.

## 4 Characterizing Maximal Walks

- Let  $l$  and  $r$  be two vertices of  $\mathcal{P}$  with  $l <_{cw} r$ , and let  $\mathcal{Q}$  be the polygon induced by  $l$  and  $r$ . If  $\overline{lr} \subset \mathcal{P}$  and  $\mathcal{Q}$  does not contain any backward deadlock with respect to  $s$ , then  $l \leq_{cw} l_{max}$  and  $r \leq_{ccw} r_{max}$ .
2.  $\mathcal{P}$  has no backward deadlock:  
 $l_{max} = r_{max} = s$ , i.e. for any vertex  $t \neq s$  it holds that  $\mathcal{P}$  has no backward deadlocks with respect to  $s$  and  $t$ .

The following lemmas give necessary and sufficient conditions for  $l_{max}$  and  $r_{max}$ .

**Lemma 10** (Avoidance of backward deadlocks). *If  $p$  and  $q$  form a backward deadlock with respect to  $s$ , then the following holds:*

1.  $l_{max} \leq_{cw} p$ .
2.  $r_{max} \leq_{ccw} q$ .

*Proof.* Ad 1:  $\text{Succ}_{ccw}(p)$  is not visible from any point  $u$  with  $s \leq_{ccw} u \leq_{ccw} q$ .

Ad 2:  $\text{Succ}_{ccw}(q)$  is not visible from any point  $u$  with  $s \leq_{cw} u \leq_{cw} p$ . □

**Lemma 11.** *Let  $l$  and  $r$  be two vertices of  $\mathcal{P}$  that satisfy the conditions of Lemma 10 for  $l_{max}$  and  $r_{max}$ , respectively. If  $l <_{cw} r$ , then the polygon  $\mathcal{Q}$  induced by  $l$  and  $r$  with respect to  $s$  contains no backward deadlocks.*

*Proof.* Let  $(\tilde{p}, \tilde{q})$  be the minimal backward deadlock of  $\mathcal{P}$  with respect to  $s$ . Since  $l$  and  $r$  satisfy the conditions of Lemma 10, we have  $l \leq_{cw} \tilde{p}$  and  $r \leq_{ccw} \tilde{q}$ . Assume that  $\mathcal{Q}$  contains a backward deadlock with respect to  $s$  formed by two vertices  $p$  and  $q$ . Then the following holds in  $\mathcal{Q}$ :

- $p <_{cw} \text{Backw}_{cw}^{\mathcal{Q}}(p)$
- $q <_{ccw} \text{Backw}_{ccw}^{\mathcal{Q}}(q)$
- $q <_{ccw} \text{Backw}_{cw}^{\mathcal{Q}}(p)$
- $p <_{cw} \text{Backw}_{ccw}^{\mathcal{Q}}(q)$

We show that all of these properties can be extended to  $\mathcal{P}$ . To simplify the discussion we distinguish the following cases with respect to the relative position of  $p$  and  $q$  to  $l$  and  $r$ .

## 4 Characterizing Maximal Walks

1.  $p \leq_{cw} l$  and  $q \leq_{ccw} r$

Note that  $(p, q) \neq (l, r)$ , since ray shots can't hit the incident edge  $\bar{l}r$ .

- a)  $p <_{cw} \text{Backw}_{cw}^Q(p) \Rightarrow p <_{cw} \text{Backw}_{cw}^P(p)$ :  
 If  $\text{Backw}_{cw}^Q(p) = \text{Backw}_{cw}^P(p)$  we are done, so let  $\text{Backw}_{cw}^Q(p) \neq \text{Backw}_{cw}^P(p)$ . Assume  $p \geq_{cw} \text{Backw}_{cw}^P(p)$ . There is some  $k$  such that  $\text{Backw}_{cw}^P(p) \in \overline{v_k v_{k+1}}$ ,  $v_{k+1} \leq_{cw} l$ . Let  $\vec{r}$  be the ray used for the clockwise ray shots from  $p$  (the rays for the ray shots in  $\mathcal{P}$  and  $\mathcal{Q}$  are identical). Then  $\vec{r} \cap \overline{v_k v_{k+1}} = \text{Backw}_{cw}^P(p)$  in  $\mathcal{Q}$  as well. But  $\vec{r}$  first hits  $\bar{l}r$  in  $\mathcal{Q}$ , so now the ray is outside of  $\mathcal{Q}$ . The hit at  $\text{Backw}_{cw}^P(p)$  must be from the inside of  $\mathcal{Q}$ , so  $\vec{r}$  must intersect another edge  $e$  of  $\mathcal{Q}$  before. But this edge would also be present in  $\mathcal{P}$ , a contradiction to the value of  $\text{Backw}_{cw}^P(p)$ .
- b)  $q <_{ccw} \text{Backw}_{ccw}^Q(q) \Rightarrow q <_{ccw} \text{Backw}_{ccw}^P(q)$ : Symmetrical to the property before.
- c)  $q <_{ccw} \text{Backw}_{cw}^Q(p) \Rightarrow q <_{ccw} \text{Backw}_{cw}^P(p)$ :  
 Analogous to the property 1a
- d)  $p <_{cw} \text{Backw}_{ccw}^Q(q) \Rightarrow p <_{cw} \text{Backw}_{ccw}^P(q)$ : Symmetrical to the property before.

2.  $p <_{cw} q \leq_{cw} l$

- a)  $p <_{cw} \text{Backw}_{cw}^Q(p) \Rightarrow p <_{cw} \text{Backw}_{cw}^P(p)$ :  
 If  $\text{Backw}_{cw}^Q(p) = \text{Backw}_{cw}^P(p)$  we are done, so let  $\text{Backw}_{cw}^Q(p) \neq \text{Backw}_{cw}^P(p)$ . Then  $\text{Backw}_{cw}^Q(p) \in \bar{l}r \Rightarrow l \leq_{cw} \text{Backw}_{cw}^P(p)$ , so  $p <_{cw} \text{Backw}_{cw}^P(p)$ .
- b)  $q <_{ccw} \text{Backw}_{ccw}^Q(q) \Rightarrow q <_{ccw} \text{Backw}_{ccw}^P(q)$ :  
 If  $\text{Backw}_{ccw}^Q(q) = \text{Backw}_{ccw}^P(q)$  we are done, so let  $\text{Backw}_{ccw}^Q(q) \neq \text{Backw}_{ccw}^P(q)$ . Then  $\text{Backw}_{ccw}^Q(q) \in \bar{l}r$  and therefore  $\text{Backw}_{ccw}^Q(q) \leq_{ccw} l$ . So  $q >_{ccw} \text{Backw}_{ccw}^P(q)$ , a contradiction to our assumption.
- c)  $q <_{ccw} \text{Backw}_{cw}^Q(p) \Rightarrow q <_{ccw} \text{Backw}_{cw}^P(p)$ :  
 If  $\text{Backw}_{cw}^Q(p) = \text{Backw}_{cw}^P(p)$  we are done, so let  $\text{Backw}_{cw}^Q(p) \neq \text{Backw}_{cw}^P(p)$ . Then  $\text{Backw}_{cw}^Q(p) \in \bar{l}r$ . So  $l \leq_{cw} \text{Backw}_{cw}^Q(p)$  and therefore  $q <_{cw} \text{Backw}_{cw}^Q(p)$ , contradicting our assumption.
- d)  $p <_{cw} \text{Backw}_{ccw}^Q(q) \Rightarrow p <_{cw} \text{Backw}_{ccw}^P(q)$ :  
 If  $\text{Backw}_{ccw}^Q(q) = \text{Backw}_{ccw}^P(q)$  we are done, so let  $\text{Backw}_{ccw}^Q(q) \neq \text{Backw}_{ccw}^P(q)$ . Then  $\text{Backw}_{ccw}^Q(q) \in \bar{l}r$ . So  $l \leq_{cw} \text{Backw}_{ccw}^P(q)$  and

## 4 Characterizing Maximal Walks

thus  $p <_{cw} \text{Backw}_{ccw}^{\mathcal{P}}(q)$ .

3.  $r \leq_{cw} p <_{cw} q$ . This case is symmetrical to the previous one.

So  $p$  and  $q$  form a backward deadlock of  $\mathcal{P}$  with  $p <_{cw} \tilde{p}$  or  $q <_{ccw} \tilde{q}$ , a contradiction to  $(\tilde{p}, \tilde{q})$  being the minimal backward deadlock of  $\mathcal{P}$ .

Note that we do not need  $p <_{cw} \tilde{p}$  and  $q <_{ccw} \tilde{q}$  because of Corollary 2.  $\square$

### 4.5 Forward Deadlocks With Respect to $s$

Like for backward deadlocks, it is necessary to get rid of the reference to  $t$  in the definition of forward deadlocks. So here is the adapted definition of forward deadlocks:

**Definition 22** (Forward deadlocks with respect to a start vertex). *Two vertices  $p$  and  $q$  of a polygon  $\mathcal{P}$  with  $p <_{cw} q$  form a forward deadlock with respect to  $s$  if the following conditions are fulfilled:*

1.  $\text{Forw}_{ccw}(q) <_{cw} p$
2.  $\text{Forw}_{cw}(p) <_{ccw} q$

As shown above there is always a minimal backward deadlock which is convenient in the maximal walks setting. For forward deadlocks the situation is less fortunate: it is shown next that there is always a unique maximal forward deadlock, but in general there is not a unique minimal forward deadlock. In fact, the number of minimal forward deadlocks can be linear in relation to the number of vertices of  $\mathcal{P}$ .

**Theorem 9** (Maximal forward deadlock). *Let  $(p, q)$  and  $(\tilde{p}, \tilde{q})$  be two forward deadlocks with respect to  $s$ . If  $p <_{cw} \tilde{p}$  and  $\tilde{q} <_{ccw} q$  then  $(\tilde{p}, q)$  is a forward deadlock with respect to  $s$ .*

*Proof.* To show that  $(\tilde{p}, q)$  is a forward deadlock, the following properties have to be shown:

- (i)  $\text{Forw}_{ccw}(q) <_{cw} \tilde{p}$ .
- (ii)  $\text{Forw}_{cw}(\tilde{p}) <_{ccw} q$ .

## 4 Characterizing Maximal Walks

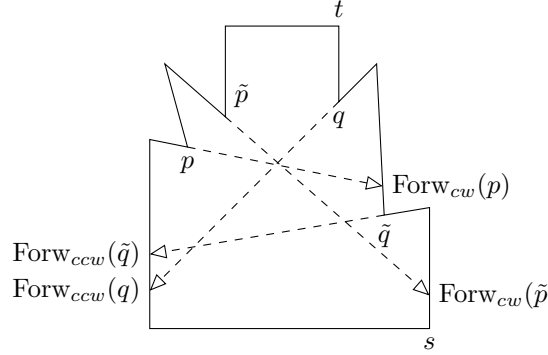


Figure 14: Counterexample used in proof of Theorem 10 with  $\tilde{q} <_{ccw} \text{Forw}_{cw}(p)$

Ad (i):  $\text{Forw}_{ccw}(q) <_{cw} p <_{cw} \tilde{p}$ , therefore  $\text{Forw}_{ccw}(q) <_{cw} \tilde{p}$ .

Ad (ii):  $\text{Forw}_{cw}(\tilde{p}) <_{ccw} \tilde{q} <_{ccw} q$ , therefore  $\text{Forw}_{cw}(\tilde{p}) <_{ccw} q$ .  $\square$

**Definition 23** (Minimal forward deadlock). A forward deadlock  $(p, q)$  with respect to  $s$  is minimal if there is no other forward deadlock  $(\tilde{p}, \tilde{q}) \neq (p, q)$  with respect to  $s$  such that  $\tilde{p} \leq_{cw} p$  and  $\tilde{q} \leq_{ccw} q$ .

**Theorem 10** (There is no unique minimal forward deadlock). Let  $(p, q)$  and  $(\tilde{p}, \tilde{q})$  be two forward deadlocks with respect to  $s$ . Let further be  $p <_{cw} \tilde{p}$  and  $\tilde{q} <_{ccw} q$ . Then  $(p, \tilde{q})$  is not necessarily a forward deadlock with respect to  $s$ .

*Proof.* If  $p <_{cw} \text{Forw}_{ccw}(\tilde{q})$  or  $\tilde{q} <_{ccw} \text{Forw}_{cw}(p)$  then  $(p, \tilde{q})$  is not a forward deadlock with respect to  $s$ . See Figure 14 for an example.  $\square$

**Theorem 11** (Number of minimal forward deadlocks). If  $\mathcal{P}$  has  $n$  vertices, there can be  $\mathcal{O}(n)$  minimal forward deadlocks with respect to  $s$ .

*Proof.* See Figure 15 for a construction with  $\mathcal{O}(n)$  minimal forward deadlocks with respect to  $s$ .  $\square$

To characterize maximal walks, *all* minimal forward deadlocks need to be known. The following algorithm computes all minimal forward deadlocks.

#### 4 Characterizing Maximal Walks

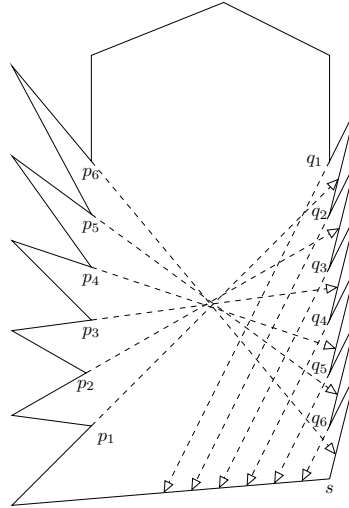


Figure 15: The minimal forward deadlocks with respect to  $s$  are  $(p_1, q_1), \dots, (p_6, q_6)$ . This gives  $\frac{n}{4} - 1 = \mathcal{O}(n)$  minimal forward deadlocks.

**Algorithm 3** (All minimal forward deadlocks). The algorithm performs a counterclockwise scan of the boundary of  $\mathcal{P}$  starting from  $s$ . When the hit point of a clockwise forward shot is encountered, its originating reflex vertex is inserted into a list  $F$  that is sorted with respect to  $<_{cw}$ . When a reflex vertex is encountered, it is checked whether it forms a forward deadlock with any vertex in  $F$ . If it does, the minimal such vertex (with respect to  $<_{cw}$ ) is selected and the corresponding forward deadlock reported. Vertices from  $F$  that form a part of a reported forward deadlock need to be marked to avoid reporting of non-minimal forward deadlocks (no deadlock with a marked vertex is reported). Finally, during the scan the list  $F$  needs to be updated to remove all reflex vertices that have been passed.

See Algorithm Listing 3 for the detailed algorithm.

**Theorem 12** (Correctness of Algorithm 3). *Algorithm 3 reports all minimal forward deadlocks of  $\mathcal{P}$  with respect to  $s$ .*

*Proof.* Claim 1: If  $(p, q)$  is reported then  $(p, q)$  is a minimal forward deadlock

## 4 Characterizing Maximal Walks

---

**Algorithm Listing 3** Compute all minimal forward deadlocks

---

$F$  is a list sorted with respect to  $<_{cw}$ ,  $next(p)$  is a function that returns the next relevant point on the boundary of  $\mathcal{P}$  after  $p$  in counterclockwise direction. Relevant points are:

- Reflex vertices of  $\mathcal{P}$ ,
- $Forw_{cw}(p)$  if  $p$  is a reflex vertex of  $\mathcal{P}$  and
- $s$ .

Now we can formulate the algorithm:

```
1:  $F \leftarrow$  empty list
2:  $a \leftarrow next(s)$ 
3: while  $a \neq s$  do
4:   if  $a = Forw_{cw}(p) \wedge p <_{cw} a$  then
5:     Add  $p$  to  $F$ .
6:   else if  $a$  is a reflex vertex  $q$  then
7:     Binary search in  $F$  for the smallest vertex  $\tilde{p}$  with respect to  $<_{cw}$ 
     such that  $Forw_{ccw}(q) <_{cw} \tilde{p}$ .
8:     if There is such a  $\tilde{p}$  and  $\tilde{p}$  is unmarked then
9:       Report the forward deadlock  $(\tilde{p}, q)$ .
10:      Mark  $\tilde{p}$ .
11:     end if
12:   end if
13:    $a \leftarrow next(a)$ 
14:   Delete all vertices  $p$  from  $F$  with  $p \leq_{ccw} a$ 
15: end while
```

---



## 4 Characterizing Maximal Walks

with respect to  $s$ .

**Proof of Claim 1:** Since  $F$  contains  $p$  we know that  $\text{Forw}_{cw}(p) <_{ccw} q$  and  $\text{Forw}_{ccw}(q) <_{cw} p$ . We also know that  $p <_{cw} q$ , thus  $(p, q)$  is a forward deadlock with respect to  $s$ .

**Assumption:**  $(\tilde{p}, \tilde{q})$  is a forward deadlock with respect to  $s$ , satisfying  $\tilde{p} <_{cw} p$  and  $\tilde{q} \leq_{ccw} q$ .

When  $(p, q)$  is reported  $F$  contains  $\tilde{p}$  because  $\text{Forw}_{cw}(\tilde{p}) <_{ccw} \tilde{q} \leq_{ccw} q$ . Since  $\tilde{p} <_{cw} p$  holds, the algorithm would report  $(\tilde{p}, q)$ , or if  $\tilde{p}$  is marked it would not report anything, a contradiction.

**Assumption:**  $(\tilde{p}, \tilde{q})$  is a forward deadlock with respect to  $s$ , satisfying  $\tilde{p} = p$  and  $\tilde{q} <_{ccw} q$ .

$(\tilde{p}, \tilde{q})$  has already been reported and therefore  $\tilde{p} = p$  is already marked. Thus  $(p, q)$  can't be reported, a contradiction.

This means that  $(p, q)$  is a minimal forward deadlock with respect to  $s$ .  $\square_{\text{Claim 1}}$

**Claim 2:** If  $(p, q)$  is a minimal forward deadlock with respect to  $s$  then the algorithm reports it.

**Proof of Claim 2:** Let  $(p, q)$  be a minimal forward deadlock with respect to  $s$ . This means that  $q$  is a reflex vertex and  $\text{Forw}_{cw}(p) <_{ccw} q$ . Therefore  $F$  contains  $p$  when the algorithm processes  $q$ . Since  $p$  is the smallest vertex in  $F$  with respect to  $<_{cw}$  satisfying  $\text{Forw}_{ccw}(q) <_{cw} p$  and  $p$  is unmarked (because  $(p, q)$  is minimal), the algorithm reports  $(p, q)$ .  $\square_{\text{Claim 2}}$

$\square$

**Theorem 13** (Running time of Algorithm 3). *Let  $\mathcal{P}$  be a polygon with  $n$  vertices. Then Algorithm 3 has running time  $\mathcal{O}(n \log n)$ .*

*Proof.* First note that the sorted list  $F$  can be maintained in  $\mathcal{O}(n \log n)$  time overall. Also, the binary search step is done at most  $\mathcal{O}(n)$  times, resulting in  $\mathcal{O}(n \log n)$  time overall. For each reflex vertex there is at most one forward deadlock reported by the algorithm, i.e. reporting takes  $\mathcal{O}(n)$  time. Thus the running time for Algorithm 3 is  $\mathcal{O}(n \log n)$ .  $\square$

The vertices  $l_{max}$  and  $r_{max}$  in the following definition indicate how far two guards can walk from  $s$  without encountering a forward deadlock.

## 4 Characterizing Maximal Walks

**Definition 24.** We define  $l_{max}$  and  $r_{max}$  to be vertices of  $\mathcal{P}$  satisfying the following properties:

1.  $l_{max} <_{cw} r_{max}$ :
  - $\overline{l_{max}r_{max}} \subset \mathcal{P}$  and the polygon  $\mathcal{Q}$  induced by  $l_{max}$  and  $r_{max}$  does not contain any forward deadlock with respect to  $s$ .
  - Let  $l$  and  $r$  be two vertices of  $\mathcal{P}$  with  $\overline{lr} \subset \mathcal{P}$ ,  $l <_{cw} r$ ,  $l \geq_{cw} l_{max}$ ,  $r \geq_{ccw} r_{max}$  and let  $\mathcal{Q}$  be the polygon induced by  $l$  and  $r$ . If  $\mathcal{Q}$  does not contain any forward deadlocks with respect to  $s$ , then  $l = l_{max}$  and  $r = r_{max}$ .
2.  $r_{max} \leq_{cw} l_{max}$ :
  - For every vertex  $t$  of  $\mathcal{P}$  with  $r_{max} \leq_{cw} t \leq_{cw} l_{max}$ , there are no forward deadlocks in  $\mathcal{P}$  with respect to  $s$  and  $t$ .
  - Let  $l$  and  $r$  be two vertices of  $\mathcal{P}$  with  $l \geq_{cw} l_{max}$ ,  $r \geq_{ccw} r_{max}$  such that for every vertex  $t$  of  $\mathcal{P}$  with  $r \leq_{cw} t \leq_{cw} l$ , there are no forward deadlocks in  $\mathcal{P}$  with respect to  $s$  and  $t$ . Then  $l = l_{max}$  and  $r = r_{max}$ .

Note that in contrast to preceding definitions of  $l_{max}$  and  $r_{max}$  (for LR-visibility, semi-wedges and backward deadlocks), the values for  $l_{max}$  and  $r_{max}$  with respect to forward deadlocks are not unique.

The following lemmas give necessary and sufficient conditions for  $l_{max}$  and  $r_{max}$ .

**Lemma 12** (Avoidance of a single forward deadlock). *Let  $(p, q)$  be a forward deadlock with respect to  $s$ . Then at least one of the following holds:*

- (i)  $l_{max} \leq_{cw} p$
- (ii)  $r_{max} \leq_{ccw} q$

*Proof.* Our goal is to get a polygon  $\mathcal{Q}$  induced by two vertices  $l$  and  $r$  of  $\mathcal{P}$  such that  $\mathcal{Q}$  is free of the forward deadlock  $(p, q)$ . Recall that  $(p, q)$  is a forward deadlock of  $\mathcal{Q}$  with respect to  $s$  if:

1.  $p <_{cw} q$
2.  $s <_{cw} \text{Forw}_{ccw}(q) <_{cw} p$
3.  $q <_{cw} \text{Forw}_{cw}(p) <_{cw} s$

## 4 Characterizing Maximal Walks

Thus at least one of these conditions has to be violated to avoid having the forward deadlock  $(p, q)$  in  $\mathcal{Q}$ . The first condition can't be changed, therefore only the two others remain. The only way to violate the chain of inequalities  $s <_{cw} \text{Forw}_{ccw}(q) <_{cw} p$  is to get rid of either  $p$  or  $q$ , thus setting  $l_{max} <_{cw} p$  (or  $r_{max} <_{ccw} q$ ). The remaining condition can be handled analogously. However, in the context of walkability,  $l_{max}$  and  $r_{max}$  represent walk targets for the guards. Thus  $l_{max} = p$  (or  $r_{max} = q$ ) is actually allowed, since the walk target is not relevant for forward deadlocks: the two guards can't walk beyond  $p$  and  $q$ , but the clockwise guard can walk beyond  $p$  while the counterclockwise guard stays below  $q$ . If the clockwise guard reaches  $q$ , then it may be possible that the counterclockwise guard can also reach  $q$ , thus  $q$  is a possible walk target (considering only the forward deadlock  $(p, q)$ ).

Also note that trying to place the walk target  $t$  below the forward deadlock (either  $t <_{cw} p$  or  $t <_{ccw} q$ ) does not change anything, since this would imply either  $l_{max} \leq_{cw} p$  or  $r_{max} \leq_{ccw} q$  - cases we already considered above.  $\square$

**Lemma 13** (Avoidance of all minimal forward deadlocks). *Let  $(p_1, q_1), \dots, (p_k, q_k)$  be all minimal forward deadlocks with respect to  $s$ . We assume without loss of generality that  $p_1 <_{cw} p_2 <_{cw} \dots <_{cw} p_k$  (this implies  $q_1 >_{ccw} q_2 >_{ccw} \dots >_{ccw} q_k$ ). To avoid all these forward deadlocks one of the following has to hold:*

- $l_{max} \leq_{cw} p_1$
- $l_{max} \leq_{cw} p_2$  and  $r_{max} \leq_{ccw} q_1$
- $\vdots$
- $l_{max} \leq_{cw} p_k$  and  $r_{max} \leq_{ccw} q_{k-1}$
- $r_{max} \leq_{ccw} q_k$

*Proof.* Using Lemma 12 it is easy to see that all minimal forward deadlocks (and thus all forward deadlocks) are avoided if any of these conditions holds. For example  $l_{max} \leq_{cw} p_i$  avoids the forward deadlocks  $(p_i, q_i), \dots, (p_k, q_k)$  and  $r_{max} \leq_{ccw} q_{i-1}$  avoids the forward deadlocks  $(p_1, q_1), \dots, (p_{i-1}, q_{i-1})$ . The edge cases can also be verified easily.  $\square$

**Lemma 14.** *Let  $D = \{(p_1, q_1), \dots, (p_k, q_k)\}$  be all minimal forward deadlocks of  $\mathcal{P}$  with respect to  $s$ . We assume without loss of generality that  $p_1 <_{cw} p_2 <_{cw} \dots <_{cw} p_k$ . Let  $i \in \{2, \dots, k-1\}$  and  $l$  and  $r$  be two vertices of  $\mathcal{P}$  such that  $l \leq_{cw} p_i$  and*

#### 4 Characterizing Maximal Walks

$r \leq_{ccw} q_{i-1}$  (the special cases  $l \leq_{cw} p_1$  and  $r \leq_{ccw} q_k$  are similar) and  $\bar{l}r \subset \mathcal{P}$ . Then the polygon  $\mathcal{Q}$  induced by  $l$  and  $r$  contains no forward deadlocks with respect to  $s$ .

*Proof.* Assume that two vertices  $p$  and  $q$  form a forward deadlock with respect to  $s$  in  $\mathcal{Q}$ . Then the following properties hold in  $\mathcal{Q}$ :

1.  $p <_{cw} q$
2.  $\text{Forw}_{ccw}^{\mathcal{Q}}(q) <_{cw} p$
3.  $\text{Forw}_{cw}^{\mathcal{Q}}(p) <_{ccw} q$

We show that these properties can be extended to  $\mathcal{P}$ :

1.  $p <_{cw} q$ : Trivial.
2.  $\text{Forw}_{ccw}^{\mathcal{P}}(q) <_{cw} p$ : If  $\text{Forw}_{ccw}^{\mathcal{P}}(q) = \text{Forw}_{ccw}^{\mathcal{Q}}(q)$  then we are done, so let  $\text{Forw}_{ccw}^{\mathcal{P}}(q) \neq \text{Forw}_{ccw}^{\mathcal{Q}}(q)$ . Then  $\text{Forw}_{ccw}^{\mathcal{Q}}(q) \in \bar{l}r \setminus \{r\}$ , therefore  $p >_{cw} r$  and  $\text{Forw}_{ccw}^{\mathcal{P}}(q) <_{cw} r$ . Combining the last two inequalities we get  $\text{Forw}_{ccw}^{\mathcal{P}}(q) <_{cw} p$ .
3.  $\text{Forw}_{cw}^{\mathcal{P}}(p) >_{cw} q$ : If  $\text{Forw}_{cw}^{\mathcal{Q}}(p) = \text{Forw}_{cw}^{\mathcal{P}}(p)$  then we are done, so let  $\text{Forw}_{cw}^{\mathcal{Q}}(p) \neq \text{Forw}_{cw}^{\mathcal{P}}(p)$ . Then  $\text{Forw}_{cw}^{\mathcal{Q}}(p) \in \bar{l}r \setminus \{l\}$ , therefore  $q <_{cw} l$  and  $\text{Forw}_{cw}^{\mathcal{P}}(p) >_{cw} l$ . Combining the last two inequalities we get  $\text{Forw}_{ccw}^{\mathcal{P}}(p) >_{cw} q$ .

So all properties needed for a forward deadlock with respect to  $s$  can be extended from  $\mathcal{Q}$  to  $\mathcal{P}$  and  $(p, q)$  forms indeed a forward deadlock with respect to  $s$  in  $\mathcal{P}$ . We now proceed to show that this leads to a contradiction.

If  $p <_{cw} l$  and  $q <_{cw} l$  (the case for  $p <_{ccw} r$  and  $q <_{ccw} r$  works analogously), then  $(p, q)$  form a forward deadlock of  $\mathcal{P}$  that has not been avoided by  $l$  and  $r$ , a contradiction. So we can assume that  $p <_{cw} l$  and  $q <_{ccw} r$ . Therefore there exists an  $i$  such that  $p <_{cw} p_i$  and  $q <_{ccw} q_{i-1}$  (note that  $(p, q) \notin D$ ). If  $q \leq_{ccw} q_i \Rightarrow (p_i, q_i)$  is not minimal, a contradiction to our assumption that  $D$  contains all minimal forward deadlocks. Similarly, if  $p \leq_{cw} p_{i-1} \Rightarrow (p_{i-1}, q_{i-1})$  is not minimal, a contradiction. Thus we have  $p_{i-1} <_{cw} p <_{cw} p_i$  and  $q_i <_{ccw} q <_{ccw} q_{i-1}$ , so  $(p, q)$  is a minimal forward deadlock of  $\mathcal{P}$ , but  $(p, q) \notin D$ , a contradiction.

In conclusion,  $\mathcal{Q}$  does not contain a forward deadlock with respect to  $s$ .  $\square$

## 4 Characterizing Maximal Walks

Note that the vertex pair  $(l_{max}, r_{max})$  is not uniquely defined for forward deadlocks (as opposed to the previous sections on  $LR$ -visibility, semi-wedges and backward deadlocks). There can actually be  $\mathcal{O}(n)$  many such vertex pairs.

### 4.6 Summary

In this chapter the concepts of  $LR$ -visibility, semi-wedges, backward deadlocks and forward deadlocks were adapted to drop the reference to a fixed target vertex  $t$ . Now only the start vertex  $s$  is fixed. This allows the formulation of conditions for maximal walkability starting from  $s$  in terms of these concepts. Later (in Chapter 7) these conditions will be used to compute maximal walkable vertex pairs.

The next chapters take a detour from the maximal walks setting, developing tools that are needed later to compute maximal walkable vertex pairs.

## 5 Conditional Constraints

This chapter takes a detour from the polygon setting into a more abstract setting: processing simple conditional constraints on two variables. An algorithm is developed to efficiently process such constraints. In Chapter 7 the conditions for maximal walkability will be transformed into constraints usable by the constraint processing algorithm to solve the maximal walks problem.

We start by formally describing our constraint satisfaction problem:

Let  $l$  and  $r$  be two integer variables, both bounded below by 0 and above by individual limits, i.e.  $l \leq a$  and  $r \leq b$ ,  $a, b \in \mathbb{N}$ . Let  $C = C_L \cup C_R$  be a set of constraints on  $l$  and  $r$ . A constraint  $c \in C_L$  has the form

$$l \leq x_i \Rightarrow r \leq y_i$$

and a constraint  $c \in C_R$  has the form

$$r \leq x_j \Rightarrow l \leq y_j$$

where  $x_i, y_i, x_j, y_j \in \mathbb{N}$ .

See Figure 16 for an illustration of the setting, showing one constraint in  $C_L$ .

Given the limits  $a$  and  $b$  as well as the set of constraints  $C$ , what are the maximal values for  $l$  and  $r$  satisfying the limits and the constraints in  $C$ ? More formally, we want to compute the following:

$$\max\{(\tilde{l}, \tilde{r}) \mid \tilde{l} \leq a \wedge \tilde{r} \leq b \wedge \forall c \in C : c \text{ is fulfilled}\}$$

Here we use the following partial ordering on  $\mathbb{N} \times \mathbb{N}$ . Let  $(l_1, r_1), (l_2, r_2) \in \mathbb{N} \times \mathbb{N}$ , then:

$$(l_1, r_1) \leq (l_2, r_2) \Leftrightarrow l_1 \leq l_2 \wedge r_1 \leq r_2$$

## 5 Conditional Constraints

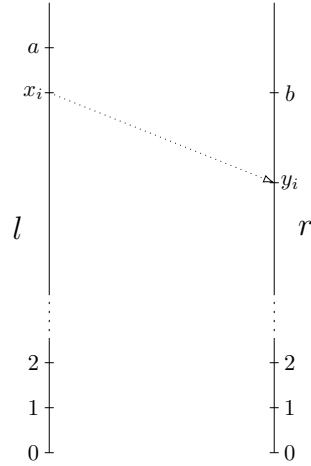


Figure 16: The variables  $l$  and  $r$  are bounded below by 0 and above by  $a$  and  $b$ , respectively.  $x_i$  and  $y_i$  represent a constraint from  $C_L$ , so if  $l \leq x_i \Rightarrow r \leq y_i$ .

To facilitate the following discussion, we introduce some notation. For a constraint  $c \in C_L$  of the form  $l \leq x_i \Rightarrow r \leq y_i$  we denote  $y_i$  as  $c_{max}^r$  (the maximal value allowed for  $r$ ) and  $x_i$  as  $c_{act}^l$  (the maximal value for  $l$  when the constraint is active). Analogously we use  $c_{max}^l$  and  $c_{act}^r$  for  $c \in C_R$ . We say that a constraint  $c$  is *activated* at  $c_{act}^r$  ( $c_{act}^l$ ).

### 5.1 Fulfilling Constraints

Given limits  $a$  and  $b$  for  $l$  and  $r$ , respectively, we can start to fulfill constraints from the top down. Let us assume for now that there is no constraint activated above  $a$  or  $b$  (this limitation will be removed later).

Let us denote the current maximal values by  $l$  and  $r$ , and use  $l^{old}$  and  $r^{old}$  to keep previous values of  $l$  and  $r$ . Now there may be constraints active on  $l$  ( $r$ ) that are not fulfilled by  $r$  ( $l$ ). To fulfill the constraints active on  $l$ , all these constraints are processed and the value of  $r$  is updated accordingly. We do this by looking at each  $\tilde{l} \in \{l, l+1, \dots, l^{old}\}$  and process the constraints activated

## 5 Conditional Constraints

at  $\tilde{r}$ . Now that all constraints active on  $l$  are fulfilled we can analogously fulfill the constraints active on  $r$ . By comparing  $l^{old}$ , the value of  $l$  before processing the constraints active on  $r$ , with  $l$ , we can determine whether this procedure needs to be iterated. If  $l = l^{old}$ , then there is no constraint  $c$  active on  $r$  with  $c_{max}^l < l$ . Since for all constraints  $c$  active at  $l$  the inequality  $r \leq c_{max}^r$  holds, it follows that all constraints are fulfilled and the algorithm can report  $(l, r)$  and stop. If  $l < l^{old}$  we need to iterate, since there may be a constraint  $c$  activated on a value  $\tilde{l} \in \{l \leq \tilde{l} < l^{old}\}$  with  $c_{max}^r < r$ .

The described algorithm is listed in Algorithm Listing 4. Correctness follows from the discussion above. Running time is  $\mathcal{O}(\max\{n, m\})$ , where  $n = a + b$  and  $m = |C|$ .

If there are constraints with  $c_{act}^l > a$  or  $c_{act}^r > b$ , then these constraints can be processed before entering the main loop.

---

### Algorithm Listing 4 Fulfill constraints

---

Input: Limits  $a$  and  $b$ , the set of constraints  $C$ .

```

1:  $(l, r) \leftarrow (a, b)$ 
2:  $(l^{old}, r^{old}) \leftarrow (l, r)$ 
3: repeat
4:   for  $\tilde{l} \in \{l, \dots, l^{old}\}$  do
5:     for  $c \in \{c \in C_L \mid c_{act}^l = \tilde{l}\}$  do
6:        $r = \min\{r, c_{max}^r\}$ 
7:     end for
8:   end for
9:    $l^{old} \leftarrow l$ 
10:  for  $\tilde{r} \in \{r, \dots, r^{old}\}$  do
11:    for  $c \in \{c \in C_R \mid c_{act}^r = \tilde{r}\}$  do
12:       $l = \min\{l, c_{max}^l\}$ 
13:    end for
14:  end for
15:   $r^{old} \leftarrow r$ 
16: until  $l^{old} = l$ 

```

---



## 5 Conditional Constraints

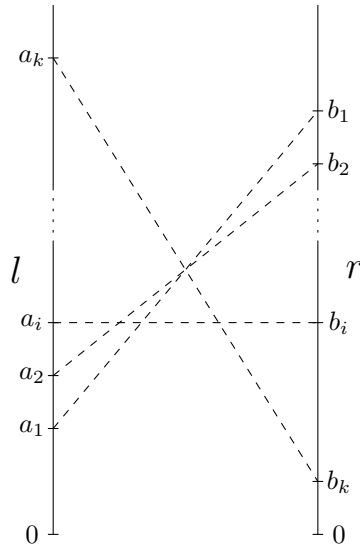


Figure 17: Multiple upper limit pairs, each upper limit pair is assumed to be maximal.

### 5.2 Multiple Starting Pairs and Feasible Pairs

In this section we extend our algorithm to handle multiple upper limit pairs and to report only feasible pairs (note that for our maximal walkability algorithm, feasible will mean visible within the polygon, but the discussion in this section is more generic, thus the broader term). We assume that there are  $k$  upper limit pairs  $(a_1, b_1), \dots, (a_k, b_k)$  ordered by increasing  $a_i$  values (this implies that the  $r_i$  values are decreasing, otherwise not all starting pairs would be maximal - see Figure 17). Deciding on feasibility of a pair is delegated to a *max\_feasible* function that computes, for a given value  $l$  and an upper limit  $r$ , the highest value  $\tilde{r}$  such that  $(l, \tilde{r})$  is feasible, or reports that no such  $\tilde{r}$  exists. The multiple upper limit pairs are handled simultaneously, such that the running time is still linear in the number of processed values, constraints and calls to *max\_feasible*.

**Theorem 14.** *Algorithm Listing 5 is correct, i.e. all maximal pairs are reported and all reported pairs are maximal.*

## 5 Conditional Constraints

---

**Algorithm Listing 5** Fulfill constraints with multiple upper limit pairs and infeasible pairs.

---

Input: Limits  $(a_1, b_1), \dots, (a_k, b_k)$ , the set of constraints  $C$ .

---

```
1:  $r_{rep} \leftarrow -1$ 
2:  $l \leftarrow a_k + 1$ 
3:  $r \leftarrow b_1$ 
4: while  $r_{rep} < r \wedge l > 0$  do
5:    $l \leftarrow l - 1$ 
6:    $(l, r) \leftarrow \text{fulfill\_constraints}(l, r, C)$ 
7:    $a_i \leftarrow \min\{a_\lambda \geq l\}$ 
8:    $r_{cand} = \min\{b_i, r\}$ 
9:    $r_{feas} \leftarrow \text{max\_feasible}(l, r_{cand})$ 
10:   $r_{proc} = \min\{\tilde{r} \mid \nexists c \in C_R \text{ active on } \tilde{r} : l_{max}^c < l\}$ 
11:  if  $r_{feas} \neq \text{null} \wedge r_{feas} \geq r_{proc} \wedge r_{feas} > r_{rep}$  then
12:     $r_{rep} \leftarrow r_{feas}$ 
13:    report $(l, r_{rep})$ 
14:  end if
15: end while
```

---

## 5 Conditional Constraints

*Proof.* First, note that the value of  $r$  is only changed by calling the algorithm `fulfill_constraints` (see Algorithm Listing 4). The first call is with  $(a_k, b_1)$  and subsequent calls with  $(l, r)$ , where  $l$  is monotonically decreasing. Thus `fulfill_constraints` $(l, r) = \text{fulfill\_constraints}(l, b_1)$ .

Let us look at one iteration of the while loop under the assumption that the algorithm worked correctly so far, i.e. all maximal pairs  $(\tilde{l}, \tilde{r})$  with  $\tilde{l} \geq l$  have been reported and no other pairs. Let  $l^{old}$  be the value of  $l$  before the iteration.

$$(l, r) = \text{fulfill\_constraints}(l^{old} - 1, b_1)$$

So  $(l, r) = \text{fulfill\_constraints}(l', b_1)$  for all  $\tilde{l}$  with  $l \leq l' < l^{old}$  and there is no pair  $(\tilde{l}, \tilde{r})$  with  $l < \tilde{l} < l^{old}$  that fulfills all constraints.

$$\begin{aligned} a_i &= \min\{a_\lambda \geq l\} \\ r_{cand} &= \min\{b_i, r\} \end{aligned}$$

Thus there can be no maximal pair  $(l, \tilde{r})$  with  $\tilde{r} > r_{cand}$ .

$$\begin{aligned} r_{proc} &= \min\{\tilde{r} \mid \exists c \in C_R : c_{act}^r \geq \tilde{r} \wedge c_{max}^l < l\} \\ r_{feas} &= \max\_feasible(l, r_{cand}) \end{aligned}$$

If  $r_{feas} = null$ , then there can be no maximal pair  $(l, \tilde{r})$  for  $\tilde{r} \leq r_{cand}$  as there is no such feasible  $\tilde{r}$ .

So now assume that  $r_{feas} \neq null$ . Thus there is no maximal pair  $(l, \tilde{r})$  with  $r_{cand} \geq \tilde{r} > r_{feas}$  (would not be feasible).

If  $r_{feas} < r_{proc}$ , then there is a constraint  $c \in C_R$  with  $c_{act}^r > r_{feas}$  and  $c_{max}^l < l$ , so  $(l, r_{feas})$  is violating constraints.

If  $r_{feas} \leq r_{rep}$ , then there is a maximal pair  $(\tilde{l}, \tilde{r})$  already reported in a previous iteration with  $\tilde{l} > l$  and  $\tilde{r} = r_{rep}$ , so  $(l, r_{feas})$  is not maximal.

So now let  $r_{feas} \neq null$ ,  $r_{feas} \geq r_{proc}$  and  $r_{feas} > r_{rep}$ . Then  $(l, r_{feas})$  is maximal, since we showed that there is no feasible constraint-fulfilling pair  $(\tilde{l}, \tilde{r})$  with  $l < \tilde{l} < l^{old}$  or  $(\tilde{l} = l \text{ and } \tilde{r} > r_{feas})$ .

Also it is clear from the discussion above that no pair  $(\tilde{l}, \tilde{r})$  is reported that is not maximal with  $l \leq \tilde{l} < l^{old}$ . Thus we can conclude that the algorithm finishes an iteration correctly if all iterations before were done correctly.

## 5 Conditional Constraints

Since any maximal pair  $(\tilde{l}, \tilde{r})$  must have  $\tilde{l} \leq a_k$  and  $\tilde{r} \leq b_1$ , the initialization before entering the while loop is correct as well. So the algorithm reports all maximal pairs and only maximal pairs.

Finally, we observe that the algorithm always terminates, since  $l$  is decreased in every iteration and thus eventually the condition  $l > 0$  is violated.  $\square$

**Theorem 15.** *Let  $n = a_k + b_1$  and  $\mathcal{O}(g(n))$  the running time of a call to `max_feasible`. If there are  $\mathcal{O}(n)$  constraints, then the running time of the algorithm in Algorithm Listing 5 is  $\mathcal{O}(n + ng(n))$ .*

*Proof.* The function `fullfill_constraints` (Algorithm Listing 4) can be implemented such that the resulting pair of the last call is remembered. This way each constraint is handled only once (if a call results in  $(l, r)$ , then the following call will be with  $(\tilde{l}, \tilde{r}) \neq (l, r)$  where  $\tilde{l} \leq l$  and  $\tilde{r} \leq r$ ). So if there are  $\mathcal{O}(n)$  constraints, then this takes  $\mathcal{O}(n)$  time.

Computing  $r_{proc}$  can also be implemented in  $\mathcal{O}(n)$  time. The last value of  $r_{proc}$  is remembered, and we scan down from this last value as long as all active constraints are fulfilled by  $l$ . As soon as a violated active constraint is encountered, we stop and remember this constraint: This is where the next  $r_{proc}$  computation starts. So only the last (violated) constraint is checked multiple times, but only once in each of the at most  $n$  iterations. So for  $\mathcal{O}(n)$  constraints we get a running time of  $\mathcal{O}(n)$ .

If feasibility checks are done at cost  $\mathcal{O}(g(n))$ , they contribute  $\mathcal{O}(ng(n))$  to the running time since there is only one feasibility check per iteration. Thus the complete running time is  $\mathcal{O}(n + ng(n))$ .  $\square$

### 5.3 Summary

In this chapter an algorithm was developed to solve a constraint satisfaction problem for simple conditional constraints that also handles multiple starting pairs and (in)feasible pairs efficiently. Later this algorithm will be used to solve the maximal walks problem.

## 6 Maximal Visibility

A maximal walkable vertex pair  $(l, r)$  of  $\mathcal{P}$  must also be visible inside of  $\mathcal{P}$ . That is, the part of  $\mathcal{P}$  from  $l$  to  $r$  (the non-walked part) must not obstruct the view between the clockwise chain from  $s$  to  $l$  and the counterclockwise chain from  $s$  to  $r$ . In this section an algorithm is developed to decide for a vertex pair if it is visible, and if not, to compute a maximal visible vertex pair below the original one. This algorithm will later be used in *max\_feasible* calls from the constraint processing algorithm.

Formally, given two vertices  $l, r \in \mathcal{P}$ , we want to compute

$$r_{max} = \max_{<_{ccw}} \{ \tilde{r} \in \mathcal{P} \mid \tilde{r} \leq_{ccw} r \wedge \overline{l\tilde{r}} \subset \mathcal{P} \}$$

Note that  $r_{max} = r$  if and only if  $l$  and  $r$  are mutually visible. We use ray shots and the hourglass data structure (developed in [7] for shortest path computations between two vertices of a polygon) to compute  $r_{max}$ .

Also note that the treatment of the problem of computing maximal visible vertex pairs in this section is self-contained and does not take into account properties of maximal walks (in particular the illustrating figures are not concerned with walkability).

Let  $g(l, r) = v_0 v_1 \dots v_k v_{k+1}$  be the shortest path from  $l$  to  $r$  in  $\mathcal{P}$ , where  $v_0 = l$  and  $v_{k+1} = r$ . Let  $T$  ( $B$ ) be the clockwise (counterclockwise) boundary chain from  $l$  to  $r$  on  $\partial\mathcal{P}$  (illustrated in Figure 18).

As it will turn out below, for all relevant shortest paths it suffices to compute the first vertex on the shortest path after the starting vertex. This reduces the time needed to compute the necessary information from linear to logarithmic time.

## 6 Maximal Visibility

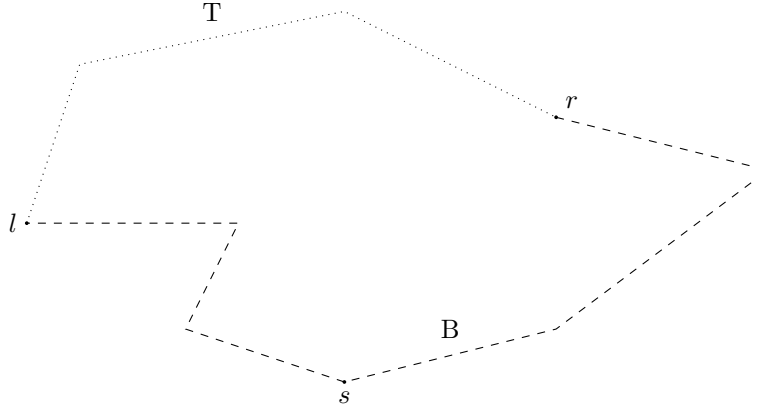


Figure 18: Top and bottom chains. The top chain ( $T$ ) is drawn in dotted style, the bottom chain ( $B$ ) in dashed style.

Suppose  $v_i \in B$  for all  $i$  (illustrated in Figure 19). Then  $g(l, r)$  is a convex polyline. The only vertices of  $B$  visible from  $l$  are between  $l$  and  $v_1$ . Since  $v_1$  is visible from  $l$  we have that  $(l, v_1)$  is maximal visible if  $v_1 \geq_{ccw} s$ , otherwise no  $v$  with  $\overline{lv} \subset \mathcal{P}$  and  $s \leq_{ccw} v \leq_{ccw} r$  exists.

Suppose  $v_j \in T$  and  $v_0, v_1, \dots, v_{j-1} \in B$  with  $j > 1$  (illustrated in Figure 20).  $v_j$  is the end of the convex chain  $v_0v_1 \dots v_j$ , so a ray  $\rho$  from  $v_1$  in direction  $\overrightarrow{v_0v_1}$  is above this chain, and  $\rho$  intersects  $T$  before  $v_j$ . So no vertex on  $B$  beyond  $v_1$  can be seen from  $l$ . Thus, like in the previous case,  $(l, v_1)$  is maximal visible if  $v_1 \geq_{ccw} s$ , otherwise no  $v$  with  $\overline{lv} \subset \mathcal{P}$  and  $s \leq_{ccw} v \leq_{ccw} r$  exists.

Now suppose  $v_1 \in T$  (illustrated in Figure 21). Let  $\rho$  be the ray starting from  $v_1$  in direction  $\overrightarrow{v_0v_1}$ . This ray intersects  $B$ , let  $\rho \cap \partial\mathcal{P} = b$ . Clearly, no vertex on  $B$  after  $b$  is visible from  $l$ . Let  $w$  be the vertex on  $B$  before  $b$  ( $b$  is not a vertex). Let  $g(l, w)$  be the shortest path from  $l$  to  $w$  in  $\mathcal{P}$ ,  $g(l, w) = w_0w_1 \dots w_mw_{m+1}$  with  $w_0 = l$  and  $w_{m+1} = w$ . Since  $\overline{lw} \subset \mathcal{P}$ , we have that  $w_i \in B$  for all  $i$ . Thus  $(l, w_1)$  is maximal visible (using the same reasoning as above) if  $w_1 \geq_{ccw} s$ , otherwise no  $w$  with  $\overline{lw} \subset \mathcal{P}$  and  $s \leq_{ccw} w \leq_{ccw} r$  exists.

The complete algorithm to compute maximal visible vertex pairs is presented in Algorithm Listing 6. Its correctness follows from the discussion above.

## 6 Maximal Visibility

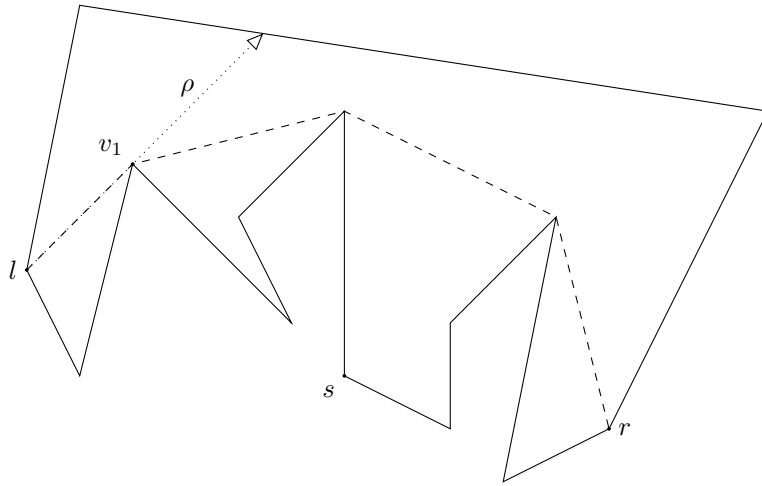


Figure 19: The shortest path lies completely on  $B$ , creating a convex polyline from  $l$  to  $r$ . No vertex on  $B$  beyond  $v_1$  is visible from  $l$ . Thus  $v_1$  is maximal if  $s \leq_{ccw} v_1$ .

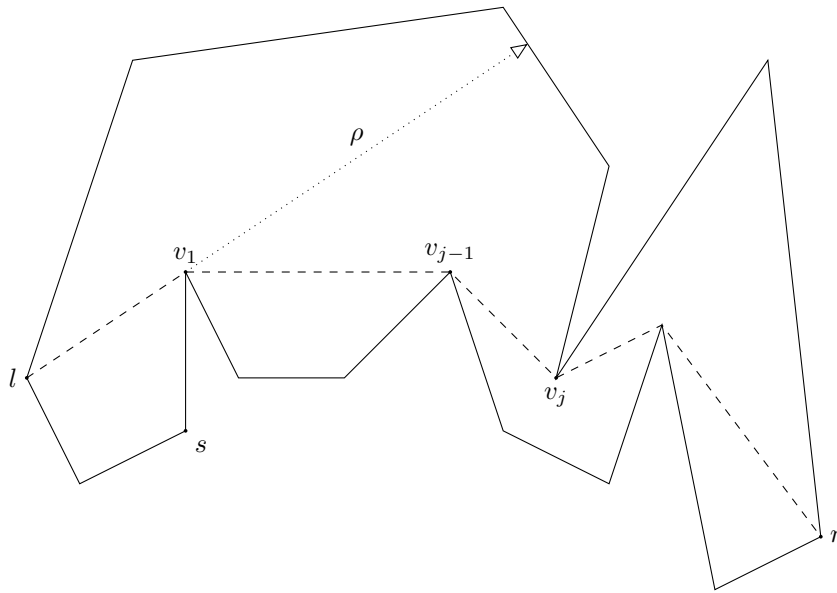


Figure 20: The first vertices of the shortest path from  $l$  to  $r$  lie on  $B$ . No vertex beyond the ray  $\rho$  is visible from  $l$ . Thus  $v_1$  is maximal if  $s \leq_{ccw} v_1$ .

## 6 Maximal Visibility

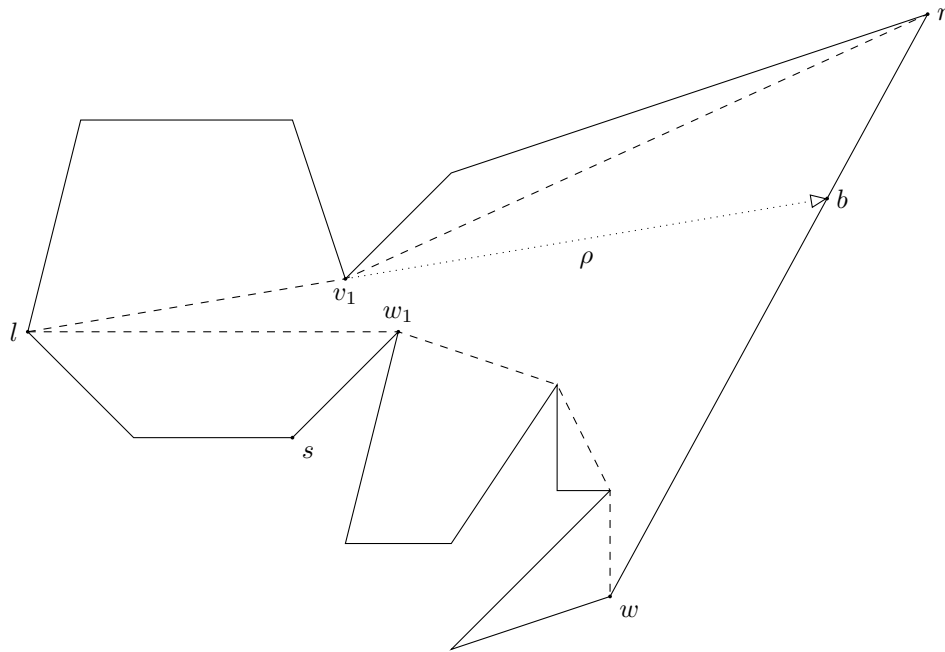


Figure 21: The first vertex after  $l$  on the shortest path from  $l$  to  $r$  lies on  $T$ . Any maximal visible vertex from  $l$  must lie in counterclockwise direction before  $b$ , the intersection of the ray  $\rho$  with  $\partial\mathcal{P}$ . The first candidate is  $w$ , using the same steps as above, the shortest path from  $l$  to  $w$  reveals the maximal visible vertex  $w_1$ .



## 6 Maximal Visibility

---

**Algorithm Listing 6** Compute maximal visible vertex pair.

---

```
1: compute hourglass for  $(l, r)$ 
2:  $v \leftarrow$  first vertex after  $l$  on shortest path to  $r$ 
3: if  $s \angle_{cw} v \angle_{cw} l$  then
4:   Report that no visible vertex pair exists.
5: else if  $s \leq_{ccw} v \leq_{ccw} r$  then
6:   Report  $(l, v)$ .
7: else
8:    $\rho \leftarrow$  ray from  $v$  in direction  $\overrightarrow{lv}$ 
9:    $b \leftarrow \rho \cap \partial\mathcal{P}$ 
10:   $w \leftarrow$  vertex in ccw direction before  $b$ 
11:  compute hourglass for  $(v, b)$ 
12:   $x \leftarrow$  first vertex after  $l$  on shortest path to  $w$ 
13:  if  $s \leq_{ccw} x \leq_{ccw} r$  then
14:    Report  $(l, x)$ .
15:  else
16:    Report that no visible vertex pair exists.
17:  end if
18: end if
```

---

## 6 Maximal Visibility

The running time is dominated by 2 hourglass computations (and querying them to retrieve the second vertex of shortest paths) and one ray shooting query. Ray shooting queries in a simple polygon can be answered in  $\mathcal{O}(\log n)$  time (see [5]). Computing hourglasses can also be done in  $\mathcal{O}(\log n)$  time (see [7]). In [7] shortest path queries using (already computed) hourglasses are answered in additional  $\mathcal{O}(k)$  time, where  $k$  is the length of the shortest path. For our purposes, the reporting step can be adapted to stop after the second vertex (the first after the starting vertex). This leads to constant additional time, thus overall  $\mathcal{O}(\log n)$  time is needed for the computation of hourglasses and shortest path vertices. Therefore also the complete algorithm takes  $\mathcal{O}(\log n)$  time.

To summarize, we have the following theorem:

**Theorem 16.** *For a polygon with  $n$  vertices, Algorithm Listing 6 correctly computes a maximal visible vertex pair in  $\mathcal{O}(\log n)$  time.*

## 7 Putting Everything Together

All that is left to do is to tie all the pieces together. The lemmas developed in Chapter 4 provide absolute and conditional constraints to use in the constraint processing algorithm from Chapter 5, and the algorithm to compute maximal visible vertex pairs from Chapter 6 is used to provide the correct feasible vertex pairs. An extension of the constraint processing algorithm to avoid new semi-wedges in the induced subpolygon will finally provide a solution to the maximal walks problem.

First of all, to be able to use the constraint processing algorithm from Chapter 5, the maximal walks problem must be translated into the terms of the constraint processing algorithm. In the constraint processing algorithm, the two variables  $l$  and  $r$  have as domains the integer intervals from 0 to  $a$  and from 0 to  $b$ , respectively. The walk targets of the guards are the vertices of the polygon, so they get labelled from 0 (the start vertex  $s$ ) to  $n$  (the start vertex  $s$  may be reached again if one guard walks around the whole polygon), once in clockwise direction for the guard walking in clockwise direction, and once in counterclockwise direction for the other guard. We denote the clockwise mapping by  $\tau_L$  and the counterclockwise mapping by  $\tau_R$ , with  $\tau_L^{-1}$  and  $\tau_R^{-1}$  being the inverse mappings. The relations  $<_{cw}$  and  $<_{ccw}$  can now be simply translated to the natural ordering on the intervals from 0 to  $n$ .

### 7.1 Absolute Constraints for Maximal Walks

The characterization of maximal walks by  $LR$ -visibility, semi-wedges and deadlocks, provides the following absolute constraints on  $l$  (constraints on  $r$  are analogous):

- For each reflex vertex  $p$  with  $p >_{cw} \text{Forw}_{cw}(p)$ :  $l \leq_{cw} p$ .

## 7 Putting Everything Together

- For each reflex vertex  $p$  with  $p <_{cw} \text{Backw}_{cw}(p)$ :  $l \leq_{cw} \text{Backw}_{cw}(p)$ .
- If  $(p, q)$  is the minimal clockwise semi-wedge, then  $l \leq_{cw} q$ .
- If  $(p, q)$  is the minimal backward deadlock, then  $l \leq_{cw} p$ .

Note that the first two constraints come from  $LR$ -visibility.

These constraints can be combined into a single absolute constraint of the form  $l \leq_{cw} x$ , which in turn can be turned into a conditional constraint:

$$r \leq_{ccw} s \Rightarrow l \leq_{cw} x$$

Absolute constraints on  $r$  can be handled analogously.

### 7.2 Conditional Constraints for Maximal Walks

The conditional constraints for maximal walks can be used directly. The characterization by  $LR$ -visibility, semi-wedges and deadlocks provides the following conditional constraints on  $l$  (constraints on  $r$  are analogous):

- Let  $p$  be a reflex vertex with  $p >_{cw} \text{Forw}_{cw}(p)$ . If  $r <_{ccw} p$ , then  $l <_{cw} \text{Pred}_{cw}(p)$ .
- Let  $p$  be a reflex vertex with  $p <_{cw} \text{Backw}_{cw}(p)$ . If  $r <_{ccw} \text{Backw}_{cw}(p)$ , then  $l \leq_{cw} p$ .
- Let  $(p, q)$  be the minimal clockwise semi-wedge. If  $r <_{ccw} q$ , then  $l <_{cw} q$ .

Note that the first two conditional constraints come from  $LR$ -visibility.

### 7.3 Starting Pairs

The multiple starting pairs for the constraint processing algorithm are provided by the minimal forward deadlocks. If  $(p_1, q_1), \dots, (p_k, q_k)$  are the minimal forward deadlocks with respect to  $s$ , then the starting vertex pairs are  $(p_1, s), (p_2, q_1) \dots (p_k, q_{k-1}), (s, q_k)$  (see Lemma 13).

## 7.4 Induced Semi-Wedges

As Lemma 1 states, a subpolygon  $Q$  of  $\mathcal{P}$  may have a semi-wedge that is not a semi-wedge of  $\mathcal{P}$ . These semi-wedges prevent walkability of  $Q$  because we are only allowed to walk in a discrete way.

Let us look more closely at such an induced semi-wedge  $(p, q)$  with  $s <_{cw} p <_{cw} q <_{cw} l$ . Then  $l <_{cw} \text{Backw}_{cw}(p) <_{cw} r$ , but recalling one condition for  $LR$ -visibility from Lemma 5 we know that if  $p < \text{Backw}_{cw}(p)$  and  $r <_{ccw} \text{Backw}_{cw}(p)$ , then we must have  $l \leq_{cw} p$ . This means that  $l <_{cw} q$  and thus there can be no induced semi-wedge  $(p, q)$  if  $LR$ -visibility conditions are fulfilled. Analogous arguments hold for counterclockwise semi-wedges, thus the problem of induced semi-wedges disappears in the presence of  $LR$ -visibility.

## 7.5 Maximal Walks Problem Solution

We now describe the complete solution to the maximal walks problem.

As a first preprocessing step all forward and backward ray shots are computed for all reflex vertices (note that a clockwise forward ray shot equals a counterclockwise backward ray shot). As shown in [5] a single ray shooting query can be performed in  $\mathcal{O}(\log n)$ , thus this preprocessing can be done in  $\mathcal{O}(n \log n)$  time.

Using the ray shots and the mappings  $\tau_L$  and  $\tau_R$  from polygon vertices to integers, all absolute and conditional constraints for maximal walks, as well as the starting vertex pairs to avoid all forward deadlocks, can be transformed to inputs for the constraint processing algorithm from Chapter 5.

To answer maximal visibility queries as *max\_feasible* calls from the constraint processing algorithm, the polygon needs to be preprocessed to answer shortest path queries as shown in [7]. This again takes  $\mathcal{O}(n \log n)$  time.

Finally, the results from the constraint processing algorithm are transformed back into the maximal walks domain by  $\tau_L^{-1}$  and  $\tau_R^{-1}$  to get the maximal

## 7 Putting Everything Together

walkable vertex pairs of  $\mathcal{P}$ . This completes our solution for the maximal walks problem.

Since the constraint processing algorithm with  $\mathcal{O}(n)$  constraints and a running time of  $\mathcal{O}(\log n)$  for a single feasibility query takes  $\mathcal{O}(n \log n)$  time, we get the following main result:

**Theorem 17.** *The maximal walks problem can be solved in  $\mathcal{O}(n \log n)$  time.*

### 7.6 Summary

In this chapter all the building blocks were combined to solve the maximal walks problem with a running time of  $\mathcal{O}(n \log n)$ .

This concludes the core part of this thesis, the next chapter will demonstrate the generic nature of the constraint processing algorithm by solving another problem (the maximal same colors walk problem), and illustrate the workings of the algorithm for better understanding.

## 8 Other Uses of the Constraint Processing Algorithm

In this chapter it is demonstrated that the developed constraint processing algorithm can not only be used to solve the maximal walk problem. In Section 8.1 a new problem (the “maximal same colors walk problem”) is defined and solved using the constraint processing algorithm in Section 8.2. The following Section 8.3 presents example instances of this problem and walks through the constraint processing algorithm solving these examples.

### 8.1 The Maximal Same Colors Walk Problem

In the context of the maximal same colors walk problem, there will again be two guards (the term *guard* may not be entirely appropriate for this problem, but keeping familiar terms should make it easier for the reader) walking along the two boundary chains of a polygon  $\mathcal{P}$ , starting from the same vertex  $s$ . Each vertex is assigned a color and both guards need to pass vertices of the same colors (the attentive reader might see conditional constraints) and are blocked by vertices of a poisonous color (this looks suspiciously like an opportunity for absolute constraints). Of course the goal is to send the two guards as far as possible along their respective boundary chains.

So here is the formal definition of the maximal same colors walk problem:

**Definition 25** (Maximal same colors walk problem (MSCW)). *Let  $\mathcal{P}$  be a polygon,  $s \in \mathcal{P}$  a starting vertex. Let further be  $C = \{c_1, \dots, c_k\}$  a set of colors and define a function  $c$  that assigns a color to each vertex of  $\mathcal{P}$ :*

$$\forall v \in \mathcal{P} : c(v) \in C$$

## 8 Other Uses of the Constraint Processing Algorithm

Let  $c_p \in C$  be a poisonous color.

The rules for a same colors walk are the following:

- Both guards  $A$  and  $B$  start at  $s$ .
- $A$  and  $B$  walk along the clockwise and counterclockwise boundary chains of  $\mathcal{P}$ , respectively.
- Only one guard moves at a time, moving from one vertex to the next in its direction.
- Let  $C_A$  and  $C_B$  be the set of colors of the vertices on  $A$ 's walk and  $B$ 's walk, respectively. Then  $C_A = C_B$  must hold.
- A vertex  $v$  with  $c(v) = c_p$  (the poisonous color) can not be passed by any guard.

The maximal same colors walk problem is to find the maximal same colors walk.

### 8.2 Solving MSCW

We can use the event processing algorithm developed in the previous chapters to solve MSCW.

The poisonous vertices (those vertices  $v$  with  $c(v) = c_p$ , the poisonous color) provide absolute constraints. Let

$$a = \min_{<_{cw}} \{v \in \mathcal{P} \mid c(v) = c_p\}$$
$$b = \min_{<_{ccw}} \{v \in \mathcal{P} \mid c(v) = c_p\}$$

Then  $l <_{cw} a$  and  $r <_{ccw} b$  (note that we use  $l$  and  $r$  as in the previous chapters).

For each color  $c \in C \setminus \{c_p\}$  we get conditional constraints. Let

$$x_c = \min_{<_{cw}} \{v \in \mathcal{P} \mid c(v) = c\}$$
$$y_c = \min_{<_{ccw}} \{v \in \mathcal{P} \mid c(v) = c\}$$



## 8 Other Uses of the Constraint Processing Algorithm

Thus, we get the conditional constraint

$$\text{if } l <_{cw} x_c \text{ then } r <_{ccw} y_c,$$

and analogously

$$\text{if } r <_{ccw} y_c \text{ then } l <_{cw} x_c.$$

The rest of the work is done by the constraint processing algorithm. Note that mutual visibility of  $l$  and  $r$  was not requested, so we can use the constraint processing algorithm without calls for feasibility checks. Also the absolute constraints provide a unique starting vertex pair, so the simple constraint fulfillment algorithm suffices.










To take advantage of the full capability of the constraint processing algorithm we could alter MSCW in the following way:

- A same colors walk must have  $\overline{lr} \subset \mathcal{P}$
- For each walk the poisonous color  $c_p$  can be chosen from a set of poisonous colors  $G \subseteq C$ .
- Vertices  $v$  with  $c(v) \in G$  are excluded from the same colors restriction.

Now we get different absolute constraints for each choice of  $c_p$ , thus we have multiple starting vertex pairs. So now with the added visibility requirement and multiple starting vertex pairs we can take full advantage of the constraint processing algorithm.

### 8.3 Examples

In the following examples we will use the terms *color* and *symbol* interchangeable, as it is easier to use symbols than colors.

**Example 2** (Constraint processing algorithm walk-through convex). As a first example we walk through the algorithm as it processes the constraints given by the convex example polygon in Figure 22a. The poisonous colors are represented by the symbols ,  and , the non-poisonous colors by the symbols , , , ,  and . The constraints and the starting pairs with respect to the start vertex  $s$ , as well as the mapping from vertices to integers,

## 8 Other Uses of the Constraint Processing Algorithm

are shown in Figure 22b. Note that the symbol  $\checkmark$  does not contribute to the constraints, as it is the symbol of the start vertex  $s$ . The conditional constraints in this case are bidirectional, since they are symmetric for the maximal same colors walk problem.

In Figure 23 all values of the constraint processing algorithm at the end of all relevant iterations are visualized. The value for  $r_{feas}$  is not shown, since the example polygon is convex, thus  $r_{feas}$  is always equal to  $r_{cand}$ . The constraints processed by the call to *fulfill\_constraints* are shown in green, the resulting pair  $(l, r)$  is shown in blue (the old values of  $(l, r)$  are indicated by a dashed blue line segment). The current values for  $(a_i, b_i)$  are shown in orange, the values for  $r_{cand}$ ,  $r_{proc}$  and (once it has become relevant)  $r_{rep}$  are shown on a separate scale on the right hand side.

Figure 23a shows the first iteration, *fulfill\_constraints* is called with  $(a_3, b_1)$  and the conditional constraints for  $\blacksquare$  and  $\star$  become active, resulting in  $(l, r) = (7, 8)$ .  $(a_i, b_i) = (a_3, b_3) = (9, 2)$ , so  $r_{cand} = 2 (= r_{feas})$ . But  $r_{proc} = 8$ , thus no maximal pair is reported. In the second iteration (see Figure 23b)  $l$  is pushed down by one,  $(a_i, b_i)$  becomes  $(a_2, b_2)$  and  $r_{cand} = 6$ , but  $r_{proc} = 8$  is still higher and nothing is reported. The next relevant iteration is shown in Figure 23c. *fulfill\_constraints* is called with  $(l, r) = (4, 8)$ , activating the constraints for  $\blacktriangle$  and  $\blacklozenge$ , resulting in  $(l, r) = (1, 6)$ .  $(a_i, b_i) = (a_1, b_1) = (3, 11)$ , thus  $r_{cand} = r = 6$ .  $r_{proc}$  goes down to 4 (staying above the only remaining constraint, the one for  $\bullet$ ). Now the conditions for reporting are satisfied, and  $(l, r_{cand}) = (1, 6)$  is reported. For the next iteration in Figure 23d we now also have a value for  $r_{rep} = 6$ . The final constraint (for  $\bullet$ ) is activated by  $l$  moving down to 0, resulting in  $(l, r) = (0, 3)$ . But now  $r_{rep} > r_{cand}$ , thus no pair is reported and also the looping condition is false and the algorithm ends.

**Example 3** (Constraint processing algorithm walk-through non-convex). In this example we walk through the algorithm processing the constraints of a non-convex polygon. The polygon and the corresponding constraints are shown in Figure 24. The poisonous colours are represented by the symbols  $\text{\textcircled{X}}$  and  $\text{\textcircled{+}}$ , the non-poisonous colors by the symbols  $\blacksquare$  and  $\checkmark$ . The conditional constraints, starting vertex pairs and the mapping from vertices to integers are visualized in Figure 24b.

## 8 Other Uses of the Constraint Processing Algorithm

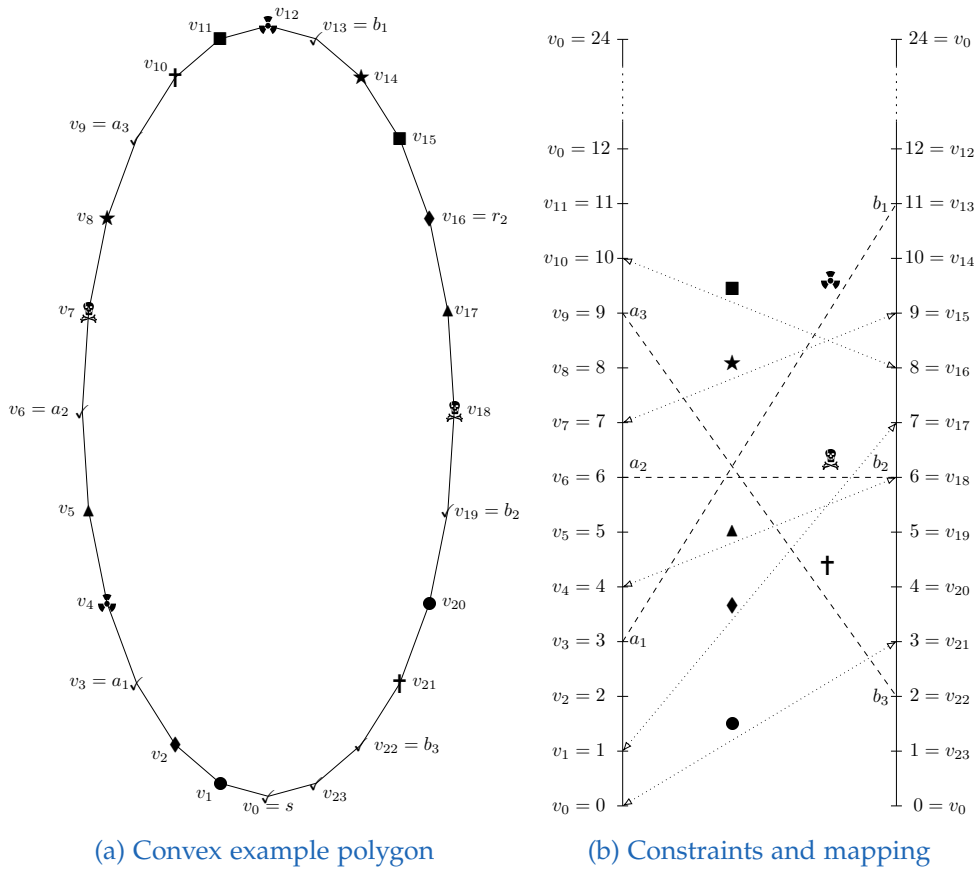


Figure 22: Maximal same colors walk example with a convex polygon (Example 2)

## 8 Other Uses of the Constraint Processing Algorithm

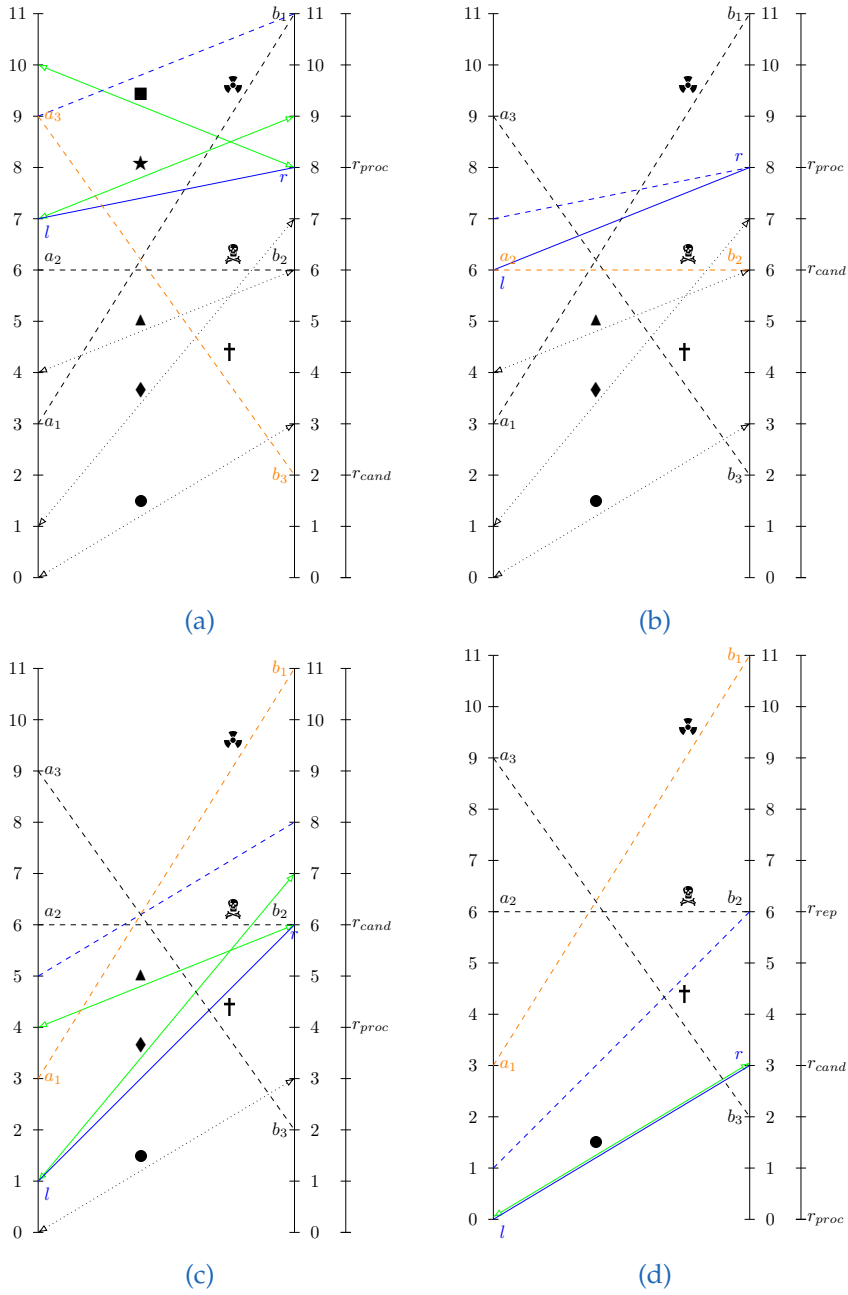


Figure 23: Walkthrough of Example 2

## 8 Other Uses of the Constraint Processing Algorithm

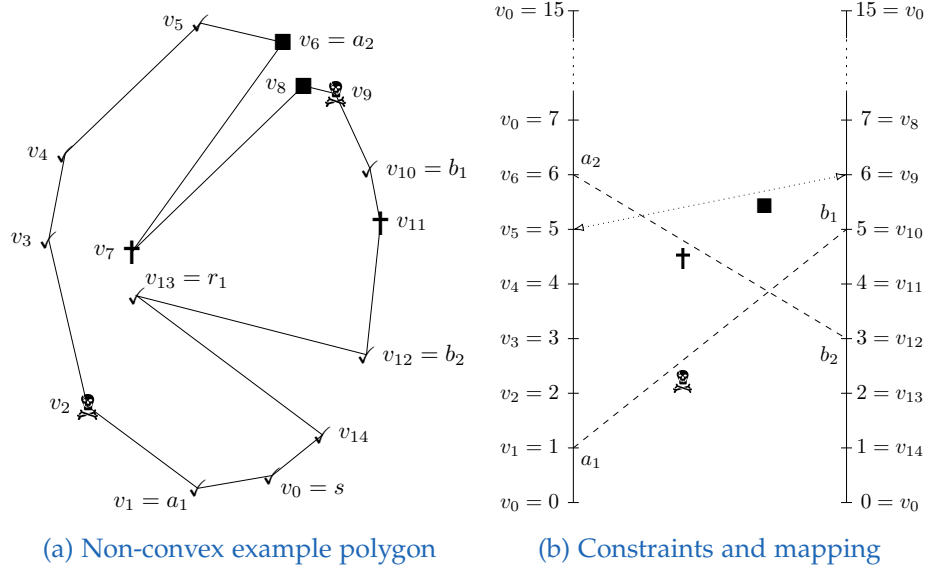


Figure 24: Maximal same colors walk example with a non-convex polygon (Example 3)

In Figure 25 the most interesting iterations of the algorithm are illustrated (analogous to Example 2 and Figure 23). Since the polygon is non-convex this time, visibility computations are necessary to ensure mutual visibility of the reported pairs. In the first iteration (see Figure 25a)  $r_{cand} = 3$ , but for  $l = 5$  there is no vertex between  $s$  and  $v_{12}$  (the vertex corresponding to  $r_{cand}$ ) that is visible from  $v_5$  (the vertex corresponding to  $l$ ), and thus no pair is reported. Note that already  $r_{proc} = 0$ , since there are no more inactive conditional constraints. Thus only  $l$  will be decreased further, while  $r = 5$  will stay until the algorithm is finished. In the second iteration (see Figure 25b)  $l$  is decreased to 4, and now  $r_{feas} = 2$  since  $\overline{v_4 v_{13}} \subset \mathcal{P}$ , so  $(4, 2)$  is reported. The next iteration (see Figure 25c) is similar,  $l$  is decreased again to 3, and  $r_{feas} = 3$ . Since  $2 = r_{rep} < r_{feas} = 3$  the maximal pair  $(3, 3)$  is reported. All other vertices below  $v_3$  can't see beyond  $v_{13}$ , thus no more pairs are reported and the algorithm ends.

## 8 Other Uses of the Constraint Processing Algorithm

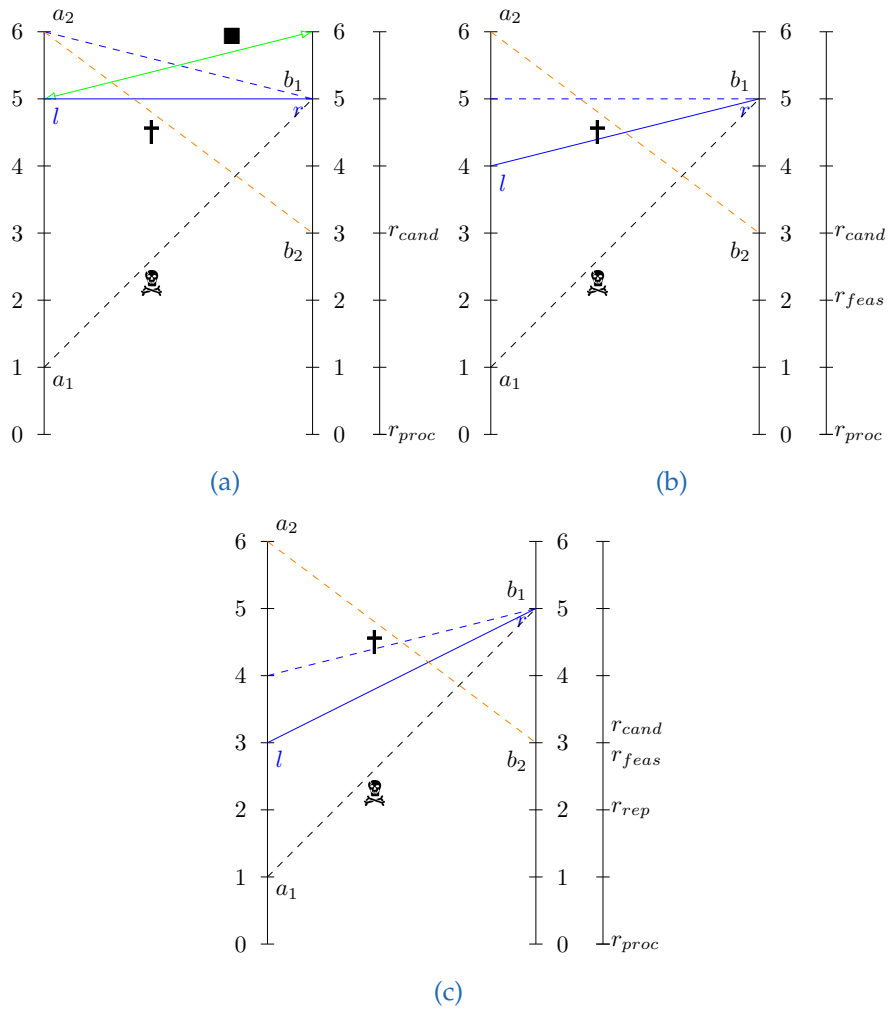


Figure 25: Walkthrough of Example 3

### 8.4 Summary

In this chapter the developed algorithm was put into action. To illustrate the general nature of the constraint processing algorithm, a new problem was formulated (the maximal same colors walk problem) and the algorithm's workings demonstrated on two example polygons.

This concludes the main work of this thesis. In the next chapter the whole thesis will be briefly reviewed and an outlook into possible future work will be provided.

## 9 Conclusion

In this final chapter the achievements of this thesis are reviewed and a short outlook to possible future work is done.

### 9.1 Review

The main goal of this thesis was to solve the maximal walks problem. This has been achieved by developing a general constraint processing algorithm that can be used to solve the maximal walks problem in  $\mathcal{O}(n \log n)$  time. To take advantage of this event processing algorithm, the geometric requirements have to be translated into absolute and conditional constraints. This is achieved by adapting and extending existing concepts from the theory about walkability of simple polygons (*LR*-visibility, semi-wedges and deadlocks).

Throughout the whole thesis the natural notion of *maximality* is used. As the next theorem shows, our algorithm solves the maximal walks problem for other notions of maximality as well. The only necessary requirement is the following: Let  $W$  be a walk for two guards, ending in  $p$  and  $q$  for the clockwise and counterclockwise guards, respectively. Then a walk  $\tilde{W}$  is larger than  $W$ , if the guards reach  $p$  and  $q$ , respectively, and (at least) one of them also reaches a vertex beyond  $p$  or  $q$ .

**Theorem 18.** *Let*

$$M = \{(p_1, q_1), \dots, (p_m, q_m)\}$$

*be all maximal walkable vertex pairs as reported by our algorithm. Then for any notion of maximality (conforming to the remark above), all maximal walkable vertex pairs are included in  $M$ .*



## 9 Conclusion

*Proof.* Suppose for some notion of maximality a maximal walkable vertex pair  $(p, q)$  is not included in  $M$ .  $(p, q)$  is walkable, so there is a walkable vertex pair  $(\tilde{p}, \tilde{q}) \in M$  such that  $p \leq_{cw} \tilde{p}$  and  $q \leq_{ccw} \tilde{q}$ . At least one of these inequalities is strict. But then a walk reaching  $(\tilde{p}, \tilde{q})$  is larger than a walk reaching  $(p, q)$ , a contradiction to  $(p, q)$  being a maximal walkable vertex pair.  $\square$

Example measures for maximality would be to maximize the number of vertices passed by a single guard or the sum of vertices passed by both guards.

### 9.2 Outlook

The final running time of the complete algorithm is  $\mathcal{O}(n \log n)$ , which should be good enough for most applications. As many walkability problems have been solved in linear time (such as reporting all walkable start and target pairs of a polygon - see [4]), it may be possible to improve the running time further.

Another interesting topic are generalizations of two guard walkability. One approach would be to extend the guards' abilities to overcome the obstacles posed by semi-wedges and deadlocks. Imagine for example that multiple guards walk along the boundary chains. When e.g. a semi-wedge occurs, one guard can stop at the "start" vertex of the semi-wedge, while another guard continues along the boundary chain, now being visible from the first guard (but not from the guards on the other boundary chain). Note that each such separation of guards on one boundary chain introduces a branch triangle in the triangulation associated with the walk.

See Figure 26 for an example walk with multiple guards from  $s$  to  $t$  that passes a wedge. Two guards start at  $s$  in clockwise direction, one guard in counterclockwise direction. At  $p_1$  one of the clockwise guards stops while the other one continues his walk. So the guard at  $p_1$  maintains visibility with both other guards. When the first guard reaches  $p_4$  the wedge is passed and the second guard joins by walking inside the polygon from  $p_1$  to  $p_4$ .

## 9 Conclusion

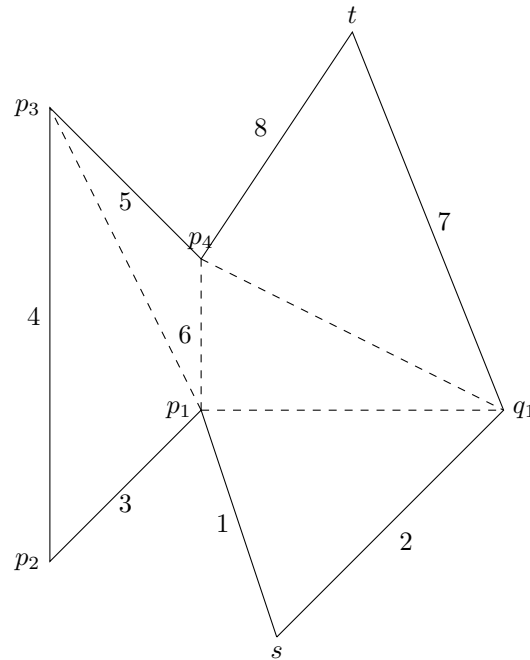


Figure 26: Walk with multiple guards

Multiple questions arise for this problem, e.g.: What is the minimum number of guards such that the polygon is walkable for them? If the number of guards is fixed, how far can they get from a starting vertex?

A different kind of generalization has been proposed by Tan in [12]. Here three guards are used to walk the polygon. Two guards walk on the boundary chains as in the two guard problem, while the third guard walks inside the polygon while maintaining visibility with the two other guards. In the paper algorithms are presented that decide whether such a walk from a starting vertex  $s$  to a target vertex  $t$  exists and construct an optimal walk in case of existence. As for the two guards problem it would be interesting to explore maximal walks with respect to a starting point in case of non-walkability.

Another way to generalize is to change the notion of visibility from having the straight line segment between the guards within the polygon to something else, for example to allow circular arcs instead of straight line segments, or

## 9 Conclusion

to allow the shortest path between the two guards to have one or more links instead of zero (similar to [13]).

A possible application of maximal walkability brings us back to the introduction where it is mentioned that the motivation for this work was to prune triangulation axes. In this setting there may be multiple parts of the triangulation that could be pruned by using walks. This means that maximal walks need to be computed for multiple starting vertices. It may be possible to use knowledge of maximal walks from one starting vertex to construct maximal walks from another starting vertex to reduce the overall running time. This would parallel the work to report all walkable pairs of a polygon to report all maximal walks from multiple or all start vertices. It is however not even clear that the output size of such a problem stays linear.

## Bibliography

- [1] Wolfgang Aigner, Franz Aurenhammer, and Bert Jüttler. On triangulation axes of polygons. *Information Processing Letters*, 115(1):45–51, 2015.
- [2] Esther M Arkin, Martin Held, Joseph SB Mitchell, and Steven S Skiena. Hamiltonian triangulations for fast rendering. *The Visual Computer*, 12(9):429–444, 1996.
- [3] David Avis and Godfried T Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transactions on Computers*, 100(12):910–914, 1981.
- [4] Binay Bhattacharya, Asish Mukhopadhyay, and Giri Narasimhan. Optimal algorithms for two-guard walkability of simple polygons. In *Workshop on Algorithms and Data Structures*, pages 438–449. Springer, 2001.
- [5] Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.
- [6] Bernard Chazelle and Leonidas J Guibas. Visibility and intersection problems in plane geometry. *Discrete & Computational Geometry*, 4(1):551–581, 1989.
- [7] Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126 – 152, 1989.
- [8] Paul J Heffernan. An optimal algorithm for the two-guard problem. *International Journal of Computational Geometry & Applications*, 6(01):15–44, 1996.
- [9] Christian Icking and Rolf Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.

## Bibliography

- [10] Giri Narasimhan. On hamiltonian triangulations in simple polygons. *International Journal of Computational Geometry & Applications*, 9(03):261–275, 1999.
- [11] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*, volume 57. Oxford University Press Oxford, 1987.
- [12] Xuehou Tan. The two-guard problem revisited and its generalization. In *Algorithms and Computation*, pages 847–858. Springer, 2004.
- [13] Xuehou Tan. Sweeping simple polygons with the minimum number of chain guards. *Information Processing Letters*, 102(2-3):66–71, 2007.
- [14] LH Tseng, P Heffernan, and DT Lee. Two-guard walkability of simple polygons. *International Journal of Computational Geometry & Applications*, 8(01):85–116, 1998.