Contents lists available at ScienceDirect
Computational Geometry: Theory and Applications
www.elsevier.com/locate/comgeo

Computing homotopic line simplification

Q1 M.A. Abam[a], S. Daneshpajouh[a], L. Deleuran[c], S. Ehsani[a], M. Ghodsi[a,b]

Q3 [a] Computer Engineering Department, Sharif University of Technology, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran
[c] Center for MAssive Data ALGOrithmics and Computer Science Department of Aarhus University, Denmark

ARTICLE INFO

Article history:
Received 3 December 2011
Received in revised form 29 October 2012
Accepted 7 February 2014
Available online xxxx
Communicated by P. Agarwal

Q2 *Keywords:*
Computational geometry
Simplification
Homotopy
Path
Line
Curve
Chain

ABSTRACT

In this paper, we study a variant of the well-known line-simplification problem. For this problem, we are given a polygonal path $\mathcal{P} = p_1, p_2, \ldots, p_n$ and a set $\mathcal{S}$ of m point obstacles in the plane, with the goal being to determine an optimal homotopic simplification of $\mathcal{P}$. This means finding a minimum subsequence $\mathcal{Q} = q_1, q_2, \ldots, q_k$ ($q_1 = p_1$ and $q_k = p_n$) of $\mathcal{P}$ that approximates $\mathcal{P}$ within a given error ε that is also homotopic to $\mathcal{P}$. In this context, the error is defined under a distance function that can be a Hausdorff or Fréchet distance function, sometimes referred to as the error measure. In this paper, we present the first polynomial-time algorithm that computes an optimal homotopic simplification of $\mathcal{P}$ in $O(n^6m^2) + T_F(n)$ time, where $T_F(n)$ is the time to compute all shortcuts p_ip_j with errors of at most ε under the error measure F. Moreover, we define a new concept of strongly homotopic simplification where every link q_lq_{l+1} of $\mathcal{Q}$ corresponding to the shortcut p_ip_j of $\mathcal{P}$ is homotopic to the sub-path $p_i, \ldots, p_j$. We present a method that in $O(n(m+n)\log(n+m))$ time identifies all such shortcuts. If $\mathcal{P}$ is x-monotone, we show that this problem can be solved in $O(m\log(n+m) + n\log n\log(n+m) + k)$ time, where k is the number of such shortcuts. We can use Imai and Iri's framework [24] to obtain the simplification at the additional cost of $T_F(n)$.

1. Introduction

Motivation. Visualization of a large geographical map may require different levels of simplifications. A map may consist of a collection of non-intersecting chains representing features, such as rivers or country borders, and of points representing places, such as cities, etc. A simplified map of interest resembles the original map in the following aspects: (i) the distance between each point on the original chain and its corresponding simplified chain should be within a given error tolerance and (ii) the original chain and its simplified version must be in the same homotopy class.[1] Roughly speaking, this means that if a point (a city, for instance) lies below the original chain (a river, for example), it must also remain below the simplified chain. We, however, consider a simpler variant of the above simplification criteria, which will be described below.

Problem. We are given a polygonal path $\mathcal{P} = p_1, p_2, \ldots, p_n$, a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ of m point obstacles in the plane not intersecting $\mathcal{P}$, and an error ε defined under a distance function F. The problem is to simplify the path $\mathcal{P}$ by $\mathcal{Q} = q_1, q_2, \ldots, q_k$ ($q_1 = p_1$ and $q_k = p_n$) within the given error ε so that $\mathcal{Q}$ is homotopic (to be exactly defined later) to $\mathcal{P}$.

E-mail address: daneshpajouh@ce.sharif.edu (S. Daneshpajouh).

[1] A homotopy class is the class of all paths that are homotopic to each other in a plane.

http://dx.doi.org/10.1016/j.comgeo.2014.02.002
0925-7721/© 2014 Published by Elsevier B.V.

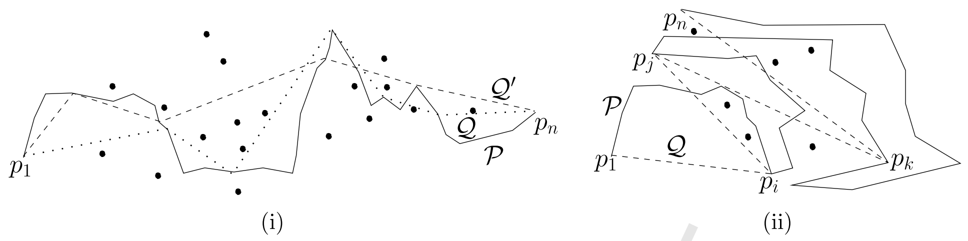

Fig. 1. (i) Two simplifications $\mathcal{Q}$ and $\mathcal{Q}'$ of $\mathcal{P}$. Only $\mathcal{Q}$ is homotopic to $\mathcal{P}$. (ii) While the simplified path $\mathcal{Q}$ and the original path $\mathcal{P}$ are homotopic, some shortcuts such as p_1p_i and p_jp_k are not homotopic to $\mathcal{P}(1,i)$ and $\mathcal{P}(j,k)$, respectively. Therefore, $\mathcal{P}$ and $\mathcal{Q}$ are not strongly homotopic.

Background. The above problem is a variant of the well-known line-simplification problem, also known in the literature as path, curve, or chain simplification, in which, for a given $\mathcal{P}$, the goal is to find the simplified path $\mathcal{Q}$ with fewer vertices approximating $\mathcal{P}$ within ε. This problem arises in many applications, such as GIS [5,6,15], image processing, and/or computer graphics [9,14], where reduction of the volume of data or lowering the complexity of the costly processing operations is important. In some of the applications, preserving the homotopy of the shape is also desirable. This ensures that the *aboveness relation*[2] of points and chains in the original and simplified maps remains unchanged.

Many variants of the line-simplification problems have been considered in the past, which can be classified into two main versions—*unrestricted* and *restricted*. In the former, the vertices of $\mathcal{Q}$ are allowed to be any arbitrary points, not just the vertices of $\mathcal{P}$ (see [20,21,23] for some results). In the restricted version, the vertices of $\mathcal{Q}$ are a subsequence of $\mathcal{P}$, and each segment q_lq_{l+1} is called a *link*. In this paper, we focus on the restricted version.

Each segment p_ip_j is called a shortcut, and each link q_lq_{l+1} of $\mathcal{Q}$ corresponds to a shortcut p_ip_j (with $j > i$). The error of such a link is defined as the distance between p_ip_j and the sub-path $p_i, \dots, p_j$ (denoted by $\mathcal{P}(i,j)$) under a desired distance function F, which is often Hausdorff or Fréchet. The total *error* of $\mathcal{Q}$ denoted by $\text{error}(\mathcal{Q},\mathcal{P})$ is also defined as the maximum error among all of its links. For each distance function, there exist two constrained optimization problems: (i) min-#: considering that $\mathcal{P}$ and ϵ are given, compute $\mathcal{Q}$ with the minimum number of vertices in such a way that $\text{error}(\mathcal{Q},\mathcal{P}) \leqslant \epsilon$, and (ii) min-$\epsilon$: considering that $\mathcal{P}$ and a maximum number of vertices k are given, compute $\mathcal{Q}$ of $\mathcal{P}$ with the smallest possible error in a way that it uses at most k vertices. The min-ϵ version is usually computed by performing a binary search over the pre-computed errors and by applying a min-# algorithm at each step. In this paper, we focus on the min-# version in the restricted model. For brevity, we use "simplification" for "min-# line simplification in the restricted model" to avoid confusion.

The simplified path $\mathcal{Q}$ is homotopic to the original path $\mathcal{P}$ (or $\mathcal{Q}$ and $\mathcal{P}$ are in the same homotopy class) if it (the simplified path $\mathcal{Q}$) is continuously deformable to $\mathcal{P}$ without passing over any points of $\mathcal{S}$ while keeping its end-vertices fixed. Precisely, the two paths α and $\beta : [0,1] \to \mathbb{R}^2$, sharing the start and end points, are homotopically equivalent with respect to $\mathcal{S}$ if there exists a continuous function $\Gamma : [0,1] \times [0,1] \to \mathbb{R}^2$ with the following properties:

(i) $\Gamma(0,t) = \alpha(t)$ and $\Gamma(1,t) = \beta(t)$ for $0 \leqslant t \leqslant 1$,
(ii) $\Gamma(s,0) = \alpha(0) = \beta(0)$, and $\Gamma(s,1) = \alpha(1) = \beta(1)$, and
(iii) $\Gamma(s,t) \notin \mathcal{S}$ for $0 \leqslant s \leqslant 1$ and $0 < t < 1$.

Fig. 1(i) illustrates two simplifications of $\mathcal{P}$ where $\mathcal{Q}$ is homotopic to $\mathcal{P}$, but $\mathcal{Q}'$ is not. We define the concept of strongly homotopic as follows: $\mathcal{Q}$ is *strongly homotopic* to $\mathcal{P}$ if for any link q_lq_{l+1} of $\mathcal{Q}$ corresponding to the shortcut p_ip_j, the sub-path $\mathcal{P}(i,j)$ and p_ip_j are also homotopic. Such a shortcut is also called a *homotopic shortcut*. Obviously, if $\mathcal{Q}$ is strongly homotopic to $\mathcal{P}$, then they are homotopic to each other, but the reverse is not necessarily true, as shown in Fig. 1(ii). However, it is easy to conclude that any x-monotone chain is strongly homotopic to any of its homotopic simplifications. We can obviously think of applications for this concept, even though its theoretical impact is of more interest to us. For example, imagine a robot and a utility wagon inside a building that includes some pillars. The robots should move along a predefined polygonal path and fix some issues in the building using the utilities in the wagon. As the wagon movement is costly, we want to minimize this cost. Furthermore, the robot cannot access the wagon if it is not within the distance ε from it. To solve this problem, we find the strongly homotopic minimum path simplification under Fréchet distance. Note that the robot cannot rotate its hand to pick up a utility from the wagon if there is a pillar between them.

Related works. There are few results on line simplification in the presence of other objects that observe the concept of homotopy type. algorithm [15,22], which is a heuristic method and does not guarantee an optimal solution. Imai and Iri [24] solved the min-ε and the min-# versions by modeling each version as a shortest-path problem in the directed

[2] Object p (point, segment or x-monotone path) is above object q if there exists a vertical line intersecting both p and q, satisfying that p is above q with respect to this vertical line.

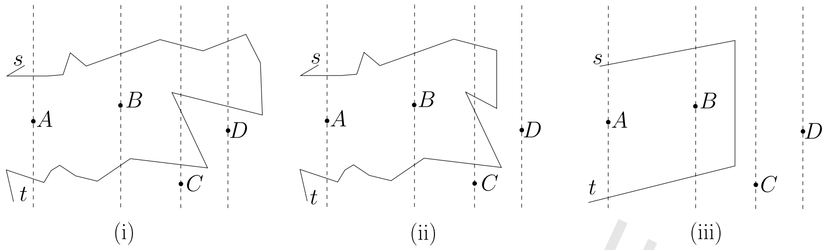

Fig. 2. (i) The path α with the sequence $\overline{ABCDDCCC}\underline{BA}$. (ii) The deformation of α that deletes $\overline{DD}$. (iii) The deformed α with the sequence $\overline{AB}\underline{BA}$, which is called $\mathrm{CS}(\alpha)$.

acyclic graphs. The running time of their method was proved to be quadratic or near quadratic by Chin and Chan [11] and by Melkman and O'Rourke [26] for the Hausdorff distance under the Euclidean metric. We refer the readers to [1,2,13,16] and to references therein for more results.

Recently, some geometers have shown interest in problems such as determining whether two paths are homotopic [10], computing the shortest homotopic path of a given path, [8] and deforming one path to another at minimal cost [7,18]. In the context of line simplification, de Berg et al. [5,6] were the first to study the homotopic line simplification in the restricted model using the Hausdorff distance measure under the Euclidean metric. Their algorithm finds the optimal simplification in $O(n(n+m)\log n)$ time, provided that $\mathcal{P}$ is x-monotone; otherwise, the simplification is not guaranteed to be optimal. Guibas et al. [21] showed that the optimal homotopic line-simplification problem in the unrestricted model is NP-hard when the simplification is forced to be simple (non-self-intersecting). A general version, a subdivision simplification, was later proved to be MIN PB-complete [19].[3] Estkowski and Mitchell [19] proposed some heuristic algorithms for this problem.

Our results. For any x-monotone path $\mathcal{P}$, we present an algorithm that computes all homotopic shortcuts of $\mathcal{P}$ in $O(m\log(n+m)+n\log n\log(n+m)+k)$ time where k is the number of such shortcuts. For simple paths, we propose an $O(n(m+n)\log(n+m))$ time algorithm to compute all homotopic shortcuts. Both algorithms can be combined with Imai and Iri's general framework to compute the optimal strongly-homotopic simplification. For the simple path case, computing all homotopic shortcuts is not sufficient to find the optimal homotopic simplification as Fig. 1(ii) illustrates. We present the first polynomial-time algorithm that computes the optimal homotopic simplification. This algorithm can be applied to non-simple paths as well.

This study is organized as follows. In Section 2, we recall some existing approaches and definitions. Section 3 presents our algorithm to compute the optimal strongly homotopic simplification. In Section 4, we present the polynomial time algorithm that computes the optimal homotopic simplification. We conclude the paper with a few remarks in Section 5.

2. Preliminaries

Imai and Iri's framework. Most simplification algorithms given in the restricted model are based on this framework where for a given ε, an unweighted directed graph $G_\varepsilon(\mathcal{P})$ (or simply G_ε) is constructed as follows: $G_\varepsilon=(V,E_\varepsilon)$ where $V=\{p_1,\dots,p_n\}$ and

$$E_\varepsilon=\bigl\{(p_i,p_j)\bigm| d_F\bigl(p_ip_j,\mathcal{P}(i,j)\bigr)\leqslant\varepsilon\bigr\}.$$

Here, $d_F(p_ip_j,\mathcal{P}(i,j))$ is the F distance of the shortcut p_ip_j and $\mathcal{P}(i,j)$ for a desired distance function F. Each $\mathcal{Q}$ with the error of, at the most, ε corresponds to a path in G_ε from p_1 to p_n. Therefore, the optimal simplification is the shortest-link path in G_ε from p_1 to p_n, which can be computed by a breadth-first search in time $O(|G_\varepsilon|)$.

Canonical sequence of a path α, denoted by $CS(\alpha)$ [10]. Let α be a path with some non-intersecting point obstacles, each of which is represented by a character. One way to represent α is to write it as a sequence of obstacle characters that α passes through either from above or below (shown by overbar or underbar). Precisely, imagine that a vertical line is drawn from each obstacle. We walk along α from its source s to its destination t. Whenever we reach a vertical obstacle line drawn from an obstacle, say A, if we are above A, we write $\overline{A}$ in the sequence, otherwise we write $\underline{A}$. For example, such a sequence for the path α illustrated in Fig. 2(i) is $\overline{ABCDDCCC}\underline{BA}$. We can use the following rule and shorten this sequence without changing its homotopy class. This means that the corresponding path of shortened sequence is homotopic to α. The shortening process is as follows. If two adjacent symbols in the sequence of α are identical (i.e., for an obstacle A, both adjacent symbols are $\overline{A}$ or $\underline{A}$), then both symbols can be deleted by deforming (or shortening) α in a way that α does not intersect with the related vertical obstacle line. Fig. 2(ii) illustrates a deformation process that deletes $\overline{DD}$ in the

[3] It is known that any MIN PB-complete problem cannot be approximated within $n^{1-\varepsilon}$ for some $\varepsilon>0$. See [25] for the precise definition of MIN PB-complete problems.

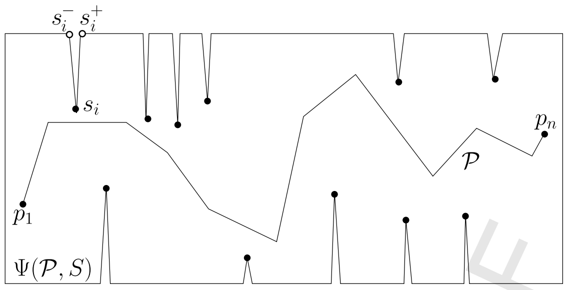

Fig. 3. A simple polygon $\Psi(\mathcal{P},\mathcal{S})$ enclosing $\mathcal{P}$.

sequence of α. This symbol deletion and path deformation is continued until no two adjacent symbols are identical. The resulting sequence is called the *canonical sequence* of α (denoted by $\mathrm{CS}(\alpha)$). Fig. 2(iii) illustrates α after several deformations, thus leading to $\mathrm{CS}(\alpha)$. It is shown in [10] that two paths α and β, sharing their endpoints, are homotopic if and only if $\mathrm{CS}(\alpha) = \mathrm{CS}(\beta)$. Clearly, the CS of a path is unique. From this point on, when we say "shortening a path", we mean to repeatedly deform the path until we end up with its canonical sequence.

3. Optimal strongly homotopic simplification

Algorithm. Our general algorithm is as follows. First, we compute G_ε in the absence of the obstacles under the given distance function F,[4] and we then test whether each edge p_ip_j in G_ε is homotopic to $\mathcal{P}(i,j)$ in the presence of the obstacles. We remove non-homotopic shortcuts from G_ε and finally compute the shortest path in G_ε from p_1 to p_n by a breadth-first search algorithm to obtain the optimal strongly homotopic simplification.

Computing homotopic shortcuts. A challenging step of our algorithms is how to efficiently compute the homotopic shortcuts. This step can be computed by applying the homotopy-testing algorithm [10] to each shortcut and its corresponding sub-path in $O((n^3 + n^2m)\log(n+m))$ time, which is far from being efficient. Here, we present two algorithms to compute the homotopic shortcuts, one for x-monotone paths and the other for the simple paths. The running time of our algorithms are $O(m\log(n+m) + n\log n\log(n+m) + k)$ and $O(n(m+n)\log(n+m))$, respectively, where k is the number of such shortcuts. The above algorithm combined with Lemmas 3 and 7 yields the following theorem.

Theorem 1. *Let $\mathcal{P}$ be a simple polygonal path with size n in a plane containing a set $\mathcal{S}$ of m point obstacles. Suppose that $T_F(n)$ is the time needed to compute G_ε under the distance function F in the absence of the obstacles, where distance function F can be a Haussdorf, Fréchet, or any other desired distance function. If $\mathcal{P}$ is x-monotone, the optimal strongly homotopic simplification of $\mathcal{P}$ can be computed in $T_F(n) + O(m\log(n+m) + n\log n\log(n+m) + k)$ time, where k is the number of homotopic shortcuts. If $\mathcal{P}$ is a simple path, this can be computed in $T_F(n) + O(n(m+n)\log(n+m))$ time.*

3.1. Computing homotopic shortcuts for x-monotone paths

The main idea behind our algorithm is to enclose $\mathcal{P}$ inside a simple polygon $\Psi(\mathcal{P},\mathcal{S})$ such that p_ip_j is a homotopic shortcut if and only if p_j is visible by p_i inside $\Psi(\mathcal{P},\mathcal{S})$.

For brevity, we use Ψ for $\Psi(\mathcal{P},\mathcal{S})$. We construct Ψ as follows. First, we determine the aboveness relation between the obstacles and $\mathcal{P}$ in $O(m\log n)$ time by locating each obstacle with respect to $\mathcal{P}$ in $O(\log n)$ time. We then compute, in $O(n+m)$ time, an axis-parallel rectangle enclosing the path vertices and the obstacles. Let λ be a sufficiently small number less than half of the x-coordinate difference of any two points of the path vertices and obstacles. Note that λ can be computed in $O((n+m)\log(n+m))$ time. For each obstacle $s_i \in \mathcal{S}$ above $\mathcal{P}$, we define two points s_i^+ and s_i^- on the upper boundary of the rectangle with x-coordinates $x(s_i)+\lambda$ and $x(s_i)-\lambda$, respectively, where $x(s_i)$ is the x-coordinate of s_i. We connect s_i to s_i^+ and s_i^- and remove the part of the rectangle joining s_i^+ to s_i^-. Similarly, we define s_i^+ and s_i^- and the associated edges for obstacles s_i below $\mathcal{P}$. Together, this gives us the simple polygon Ψ (see Fig. 3).

Lemma 2. *A shortcut p_ip_j is a homotopic shortcut if and only if p_j is visible by p_i inside $\Psi(\mathcal{P},\mathcal{S})$.*

Proof. p_i is inside Ψ by the choice of λ. If p_j is visible by p_i inside Ψ, then p_ip_j and $\mathcal{P}(i,j)$ make a cycle inside the simple Ψ, which can be deformed into a single point, thereby implying that p_ip_j and $\mathcal{P}(i,j)$ are homotopic. Assume p_ip_j is a homotopic shortcut. To the contrary, assume that p_i and p_j are not visible to each other inside Ψ. This means that

[4] For simple paths, $T_F(n)$ (the time needed to compute G_ε) is $O(n^2)$ [11] and $O(n^3)$ [2], where F is a Hausdorff or Fréchet, respectively.

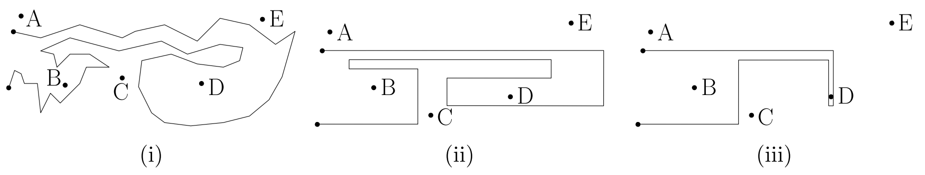

Fig. 4. (i) The original path with the sequence $\underline{A}\overline{BCD}\underline{EED}\overline{DDCBB}\underline{BA}$, (ii) the rectified path, and (iii) the shortened path with the canonical sequence $\underline{A}\overline{BCD}\underline{D}\overline{C}\underline{BA}$.

p_ip_j must intersect both $s_ks_k^-$ and $s_ks_k^+$ for some k (note that p_i and p_j are both inside Ψ). This implies that s_k is between the shortcut p_ip_j and $\mathcal{P}(i, j)$, thus contradicting the assumption. $\square$

The above lemma reduces our problem to computing the visibility graph of n points inside a simple polygon of size $3m + 4$. Ben-Moshe et al. [3] presented an $O(m + n \log n \log(m + n) + k)$-time algorithm to compute all visible pairs in $O(n + m + k)$ space, where k is the number of visible pairs. Accordingly, we obtain the following result:

Lemma 3. *Let $\mathcal{P}$ be an x-monotone polygonal path and $\mathcal{S}$ be a set of m point obstacles in a plane. All homotopic shortcuts of $\mathcal{P}$ can be computed in $O(m \log(n + m) + n \log n \log(n + m) + k)$ time and $O(n + m + k)$ space where k is the number of such shortcuts.*

3.2. *Computing all homotopic shortcuts for simple paths*

In this section, we present our algorithm for computing all homotopic shortcuts for simple paths. We first briefly describe the algorithm given in [10] and then show how to exploit it for our problem.

The algorithm by Cabello et al. [10] tests whether two simple paths α and β with common fixed endpoints and a total size of n in the presence of m point obstacles are homotopic. Their algorithm first computes $\mathrm{CS}(\alpha)$ (and $\mathrm{CS}(\beta)$) efficiently as follows. The path α is first decomposed into maximal x-monotone sub-paths. Let Υ be the set of obstacles and the maximal x-monotone sub-paths. Some members of set Υ induce an aboveness relation. Clearly, this aboveness relation is acyclic and thus defines a partial order on Υ. This partial order relation can be extended to a total order relation. The rank of x-monotone sub-paths in the total order can be used to rectify the paths. Precisely, each maximal x-monotone sub-path is treated as a horizontal segment where its y-coordinate is its rank in the total order. This new path is called the *rectified path*. The rectified path is then shortened to obtain the *canonical rectified path* (denoted by $\mathrm{CRP}(\alpha)$), which represents $\mathrm{CS}(\alpha)$ (Fig. 4 illustrates the initial steps of this algorithm).

To test whether paths α and β are homotopic, the algorithm first tests whether the turn obstacles of α and β define the same set where a *turn obstacle* O of α is an obstacle at which $\mathrm{CRP}(\alpha)$ makes a U-turn, i.e., either $\underline{O}\overline{O}$ or $\overline{O}\underline{O}$ exists in $\mathrm{CS}(\alpha)$. If not, α and β are not homotopic. Otherwise, both $\mathrm{CRP}(\alpha)$ and $\mathrm{CRP}(\beta)$ are broken at the turn obstacles and are, consequently, decomposed into x-monotone sub-paths. Then, each sub-path of $\mathrm{CRP}(\alpha)$ and its corresponding sub-path in $\mathrm{CRP}(\beta)$ (i.e., the one ending at the same turn obstacles) are tested to see whether or not they are homotopic. These tests are performed together by simultaneously sweeping the x-monotone sub-paths of $\mathrm{CRP}(\alpha)$ and $\mathrm{CRP}(\beta)$ and the set of obstacles from the left to the right—see [10] for more details.

3.2.1. *Our algorithm*

We now show how to exploit the algorithm in [10] and compute all homotopic shortcuts p_hp_i in $O(n(m+n)\log(n+m))$ time. Let $C_h = \{p_hp_i \mid 1 \leqslant h < i \leqslant n\}$. We show how to compute all homotopic shortcuts in C_h for a fixed h in $O((m + n)\log(n + m))$ time. This process can be repeated for every $1 \leqslant h \leqslant n$, thus yielding our main result.

From now on, due to the ease of presentation, we fix $h = 1$ and compute all homotopic shortcuts in C_1. Our global strategy is as follows. We compute $\mathrm{CRP}(\mathcal{P}(1, i))$'s ($\mathrm{CRP}(i)$'s, for short) and maintain them in a plane $\mathcal{H}_1$. We begin this step by rectifying $\mathcal{P}$ (Fig. 5(ii)). We then inductively compute $\mathrm{CRP}(i)$ for all i's. If $\mathrm{CRP}(i)$ has a turn obstacle, it cannot be homotopic to the shortcut p_1p_i as the CRP of p_1p_i does not have any turn obstacle. Therefore, we maintain $\mathrm{CRP}(i)$, provided that it is x-monotone. However, a separate maintenance of all x-monotone $\mathrm{CRP}(i)$'s may require $O(n^2)$ space and time, which results in an inefficient $O(n^3)$-time algorithm to compute all homotopic shortcuts p_ip_j. Hence, we encode all x-monotone $\mathrm{CRP}(i)$'s into a geometric tree $\mathcal{T}$ of size $O(n)$ in the plane $\mathcal{H}_1$ where each x-monotone $\mathrm{CRP}(i)$ corresponds to a path from the root to a node or leaf in $\mathcal{T}$ (Fig. 5(iii)). In a different plane $\mathcal{H}_2$, we rectify all shortcuts p_1p_i $(2 \leqslant i \leqslant n)$ whose $\mathrm{CRP}(i)$ is x-monotone in the presence of the point obstacles (Fig. 5(iv)). We finally follow the main step, which is testing whether $\mathrm{CRP}(i)$'s encoded in $\mathcal{T}$ in plane $\mathcal{H}_1$ and the corresponding rectified shortcuts in plane $\mathcal{H}_2$ are homotopic. Note that, in Fig. 5, $\mathrm{CRP}(5)$ and $\mathrm{CRP}(11)$ are not maintained in $\mathcal{T}$ as they are not x-monotone, and thus the shortcuts p_1p_5 and p_1p_{11} are not rectified, while the others are rectified and shown in Fig. 5(iv). As the rectifying operations change the position of points, in Fig. 5, for any point p, we use $\widehat{p}$ and $\widetilde{p}$ instead of p in the rectified forms—note that a rectifying operation just changes the y-coordinate of a point and the x-coordinate remains unchanged. The following sketches our algorithm.

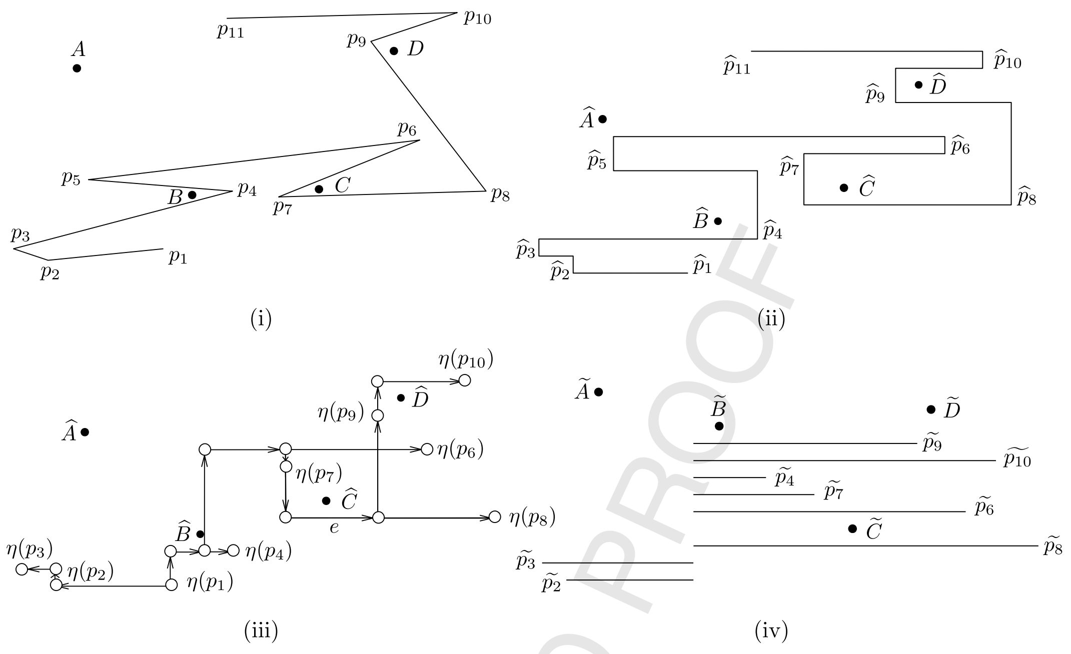

Fig. 5. (i) The original path $\mathcal{P} = p_1, \dots, p_{11}$. (ii) The rectified path. (iii) The tree $\mathcal{T}$ in plane $\mathcal{H}_1$. The nodes of $\mathcal{T}$ are denoted by empty circles. (iv) The rectified shortcuts in plane $\mathcal{H}_2$.

Algorithm Computing Homotopic Shortcuts

```
1. Rectify $\mathcal{P}$.
2. Construct a segment-dragging queries structure over point obstacles.
3. for $h = 1$ to $n$
4.    do Let $\mathcal{H}_1, \mathcal{H}_2, \mathcal{T}$ be empty.
5.       Compute CRP($\mathcal{P}(h,i)$), $h < i \leqslant n$ and encode all in a geometric tree $\mathcal{T}$ in plane $\mathcal{H}_1$.
6.       Rectify all shortcuts $p_h p_i$, $h < i \leqslant n$ whose CRP($\mathcal{P}(h,i)$) is $x$-monotone in plane $\mathcal{H}_2$.
7.       Test whether each CRP($\mathcal{P}(h,i)$), $h < i \leqslant n$ in $\mathcal{H}_1$ and the corresponding rectified shortcuts in $\mathcal{H}_2$ are homotopic
         and output them.
```

Next, we explain the details of each step of our algorithm for $h = 1$.

Rectifying $\mathcal{P}$. Because $\mathcal{P}$ is simple, the edges of $\mathcal{P}$ and the obstacles of $\mathcal{S}$ induce the aboveness relation, which is acyclic and computable in time $O((n+m)\log(n+m))$ [27]. In fact, the aboveness relation defines a partial order that can be easily extended to a total order. Let $\text{rank}_p(\mathcal{O})$ be the rank of an object $\mathcal{O}$ (obstacle or edge) in the total order. By letting the y-coordinate of any point of object $\mathcal{O}$ be $\text{rank}_p(\mathcal{O})$, the path $\mathcal{P}$ becomes rectified, i.e., each edge is represented as a horizontal segment. We join two horizontal segments corresponding to two consecutive edges in $\mathcal{P}$ by a vertical segment to maintain the original connectivity. Note that for each segment $p_i p_{i+1}$, there is a horizontal segment in the rectified $\mathcal{P}$. We denote the endpoint of this horizontal segment corresponding to p_{i+1} by $\widehat{p}_{i+1}$, see Fig. 5(ii).

Segment-dragging queries. Our algorithm relies on a segment-dragging data structure defined over m point obstacles to compute CRP(i). This data structure is used to find the first obstacle hit when the given vertical segment is translated vertically. From this point forward, we call such an obstacle the closest obstacle to a given vertical segment. We use Chazelle's data structure [12] that preprocesses the obstacles in time $O(m\log m)$ while using $O(m)$ space to answer segment-dragging queries in $O(\log m)$ time.

Canonical rectified paths. We maintain two stacks $\mathcal{S}_c$ and $\mathcal{S}_u$. Upon processing $\widehat{p}_{i+1}$, stack $\mathcal{S}_c$ maintains CRP(i) as a sequence of horizontal and vertical segments, and stack $\mathcal{S}_u$ maintains U-turns as a sequence of vertical segments at which CRP(i) makes U-turns. Moreover, $\mathcal{T}$ contains all CRP(j), $1 < j < i+1$. In the beginning, $\mathcal{S}_u$ is set to be empty, and $\mathcal{S}_c$ is set to be the first edge of the rectified $\mathcal{P}$. We also initialize the tree $\mathcal{T}$ by adding new nodes $\eta(p_1)$ and $\eta(p_2)$, as well as the directed edge $(\eta(p_1), \eta(p_2))$ to the empty tree $\mathcal{T}$ and set $\eta(p_1)$ to be the root of $\mathcal{T}$, where $\eta(p)$ denotes the node with label p placed at the position of point p. Now, CRP($i+1$) is computed as follows. Consider the horizontal segment of the rectified $\mathcal{P}$ ending at $\widehat{p}_{i+1}$. Let the other endpoint of this segment be q. We pop $\mathcal{S}_c$, which is a segment ending at $\widehat{p}_i$ and starting at another point, say r. If $r\widehat{p}_i$ is vertical, we should pop one more segment from $\mathcal{S}_c$. Without loss of generality, assume that

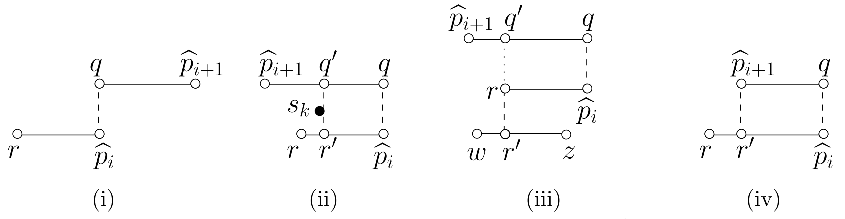

Fig. 6. Different cases depending on the positions of $r\widehat{p}_i$ and $q\widehat{p}_{i+1}$ and the obstacles when $\widehat{p}_i$ is to the right of r.

$r\widehat{p}_i$ is horizontal and $\widehat{p}_i$ is to the right of r—the cases, where $\widehat{p}_i$ is to the left of r, can be handled similarly. Note that $\widehat{p}_i$ and q are connected in the rectified $\mathcal{P}$ by a vertical segment. We distinguish four cases based on being either true or false according to the following three propositions: (Q_1) $\widehat{p}_{i+1}$ is to the right of $\widehat{p}_i$; (Q_2) there is at least one obstacle above segment $r\widehat{p}_i$ and below segment $q\widehat{p}_{i+1}$; and (Q_3) the length of $q\widehat{p}_{i+1}$ is greater than the length of $r\widehat{p}_i$—see Fig. 6, which depicts all four cases. It is easy to verify whether Q_2 is true or false using the segment-dragging structure

(i) Q_1: the segments $r\widehat{p}_i$, $\widehat{p}_i q$, and $q\widehat{p}_{i+1}$ are simply pushed into $\mathcal{S}_c$. Moreover, if $\mathcal{S}_u$ is empty, we update $\mathcal{T}$ by adding new nodes $\eta(p_{i+1})$ and $\eta(q)$, as well as new edges $(\eta(p_i), \eta(q))$, and $(\eta(q), \eta(p_{i+1}))$ to $\mathcal{T}$.
(ii) Not (Q_1) but Q_2: $\widehat{p}_{i+1}$ is to the right of $\widehat{p}_i$ and let $r'q'$ be the vertical segment touching s_k from right where s_k is the right-most obstacle on the left side of $\widehat{p}_i q$. We push rr', $r'q'$, and $q'\widehat{p}_{i+1}$ into $\mathcal{S}_c$ (we indeed erase $r'\widehat{p}_i$, $\widehat{p}_i q$, and qq'). We also push $r'q'$ into $\mathcal{S}_u$ as a new U-turn. Because $\mathcal{S}_u$ is not empty, we do not update $\mathcal{T}$.
(iii) Not (Q_1), not (Q_2), but Q_3: let rr' be the vertical segment hanging from r, which can be obtained by popping $\mathcal{S}_c$ one more time. We glue $r'r$ and rq' and push $r'q'$ into $\mathcal{S}_c$ as well as $q'\widehat{p}_{i+1}$. Moreover, if the top of $\mathcal{S}_u$ is $r'r$, we pop $r'r$ from $\mathcal{S}_u$. If $\mathcal{S}_u$ is empty, we update tree $\mathcal{T}$ by inserting new nodes $\eta(r')$ (if it does not exist), $\eta(q')$ and $\eta(p_{i+1})$ as well as the new directed edges $(\eta(r'), \eta(q'))$, and $(\eta(q'), \eta(p_{i+1}))$ to tree $\mathcal{T}$. If $\eta(r')$ does not exist, we split the edge $(\eta(z), \eta(w))$ of $\mathcal{T}$, whose embedding in the plane (i.e., segment zw) contains point r', into the new directed edges $(\eta(z), \eta(r'))$ and $(\eta(r'), \eta(w))$. We can test in constant time whether $r'q'$ can still be moved to the left, by considering the last three segments of $\mathcal{S}_c$. If it can, we pop $q'\widehat{p}_{i+1}$ and consider it as a new horizontal segment to be inserted, and then repeat step (iii).
(iv) Not (Q_1), not (Q_2), and not (Q_3): we push rr' and $r'\widehat{p}_{i+1}$ into $\mathcal{S}_c$. Moreover, if $\mathcal{S}_u$ is empty, we update $\mathcal{T}$ by adding new nodes $\eta(r')$ and $\eta(p_{i+1})$, removing the edge $(\eta(p_i), \eta(r))$ and adding new directed edges $(\eta(r), \eta(r'))$, $(\eta(r'), \eta(p_i))$ and $(\eta(r'), \eta(p_{i+1}))$.

Because in each of the n iterations, a constant number of edges are added to $\mathcal{T}$, the size of $\mathcal{T}$ is $O(n)$. Note that, in some iterations, step (iii) may be processed recursively several times, but it is only for the last recurrence that we may have to update $\mathcal{T}$. Furthermore, the number of such recurrences in total is $O(n)$, as each recurrence can be charged to a horizontal segment of the path and each horizontal segment can be charged at most twice. Therefore, we have the following lemma:

Lemma 4. *All x-monotone* CRP(i) *can be encoded in* $O(n \log m)$ *time into a geometric tree* $\mathcal{T}$ *of size* $O(n)$*, where each x-monotone* CRP(i) *corresponds to a path from the root to a node or leaf in tree* $\mathcal{T}$*.*

Rectifying shortcuts. Let $\mathcal{I}$ be the set of indices i such that CRP(i) is x-monotone. In fact, $\mathcal{I}$ presents all remaining shortcut candidates to be homotopic shortcuts after computing all CRP(i)'s. Consider all shortcuts p_1p_i and m obstacles in $\mathcal{S}$, where $i \in \mathcal{I}$. They induce an aboveness relation defining a partial order, which can be simply extended to a total order. Let $\text{rank}_s(\mathcal{O})$ be the rank of an object $\mathcal{O}$ (obstacle or shortcut) in this total order. We set the y-coordinate of any point of object $\mathcal{O}$ to be $\text{rank}_s(\mathcal{O})$. This rectifies all shortcuts. We denote the rectified shortcut p_1p_i by $\widetilde{p}_1\widetilde{p}_i$, see Fig. 5(iv). Note that, to reduce the confusion, $\widetilde{p}_1$ is not illustrated in this figure. Also, in this figure, there is no relation between $\text{rank}_p(s_i)$ and $\text{rank}_s(s_i)$ for obstacle s_i, as they come from two different total orders. For instance, the y-coordinates of obstacle B in Fig. 5(iii) and Fig. 5(iv) are different.

Testing homotopy for tree $\mathcal{T}$ *and the rectified shortcuts.* For a horizontal edge e of $\mathcal{T}$, let above(e) be the set of the point obstacles above edge e. Precisely, above(e) is the set of obstacles s_j, thus satisfying (i) $\text{rank}_p(s_j) > \text{rank}_p(e)$, and (ii) the x-coordinate of s_j lies between the x-coordinates of the endpoints of e. In a similar way, we define below(e) to denote all obstacles below edge e. As every obstacle above (or below) CRP(i) is above (or below) exactly one edge of all edges appearing in CRP(i), the homotopy test of p_1p_i and CRP(i) can be divided into some homotopy tests in which edges appearing on CRP(i) are involved. For any edge e appearing on CRP(i), every obstacle in above(e) must be above $\widetilde{p}_1\widetilde{p}_i$, and any obstacle in below(e) must be below $\widetilde{p}_1\widetilde{p}_i$; otherwise, p_1p_i and CRP(i) cannot be homotopic. Therefore, for each edge e and any obstacle s_j, we have the following condition:

Condition 1. $(s_j \in \text{above}(e) \wedge \text{rank}_s(p_1p_i) < \text{rank}_s(s_j)) \vee (s_j \in \text{below}(e) \wedge \text{rank}_s(p_1p_i) > \text{rank}_s(s_j))$.

The following simple lemma states the case where the rectified shortcut $\widetilde{p}_1\widetilde{p}_i$ and CRP(i) are homotopic:

Lemma 5. *For any $i \in \mathcal{I}$, the rectified shortcut $p_1 p_i$ and* CRP(i) *are homotopic if and only if, for any horizontal edge e of $\mathcal{T}$ appearing on* CRP(i) *and any obstacle $s_j \in \text{above}(e) \cup \text{below}(e)$, Condition 1 holds.*

This lemma implies that any edge e and any object $s_j \in \text{above}(e) \cup \text{below}(e)$ violating Condition 1 removes $p_1 p_i$ of being a homotopic shortcut, and if there is no such pair e and s_j, the shortcut $p_1 p_i$ is definitely a homotopic shortcut. However, testing Condition 1 for each edge e and any $s_j \in \text{above}(e) \cup \text{below}(e)$ is costly in total. We can reduce the total cost using the following lemma.

For an edge e, let $s_j \in \text{above}(e)$ ($s_j \in \text{below}(e)$) be the obstacle with the minimum (maximum) rank_s. We define $\min(\text{above}(e)) = \text{rank}_s(s_j)$ ($\max(\text{below}(e)) = \text{rank}_s(s_j)$). The following lemma states that from among all obstacles, either above or below e, at most two obstacles together with e must be tested to determine whether they satisfy Condition 1.

Lemma 6. *It suffices to verify Condition 1 for each horizontal edge e of $\mathcal{T}$ and the obstacles with ranks* $\min(\text{above}(e))$ *and* $\max(\text{below}(e))$ *in plane $\mathcal{H}_2$, if they exist.*

Proof. We claim if there is an obstacle violating Condition 1 for an edge e, then the obstacle with the rank of either $\min(\text{above}(e))$ or $\max(\text{below}(e))$ also violates Condition 1 for the edge e. The lemma follows immediately from the claim. Next, we prove the claim.

Let $\mathcal{O}$ be an obstacle violating Condition 1 for edge e. Assume $\mathcal{O}$ is in above(e) (if $\mathcal{O}$ is in below(e), the proof can be done similarly). Since $\mathcal{O}$ is in above(e), $\text{rank}_s(O) \geqslant \min(\text{above}(e))$. On the other hand, since $\mathcal{O}$ violates Condition 1, we know $\text{rank}_s(\mathcal{O}) < \text{rank}_s(p_1 p_i)$ for some i. All this together implies $\min(\text{above}(e)) < \text{rank}_s(p_1 p_i)$ which implies the object with the rank of $\min(\text{above}(e))$ also violates Condition 1. □

Knowing Lemma 6, the main question is how quickly we can perform the verification of Condition 1 for any horizontal edge e of $\mathcal{T}$ to remove non-homotopic shortcuts. Note that above(e) or below(e) may contain many obstacles and that $\mathcal{I}(e)$ may contain several indices, where $\mathcal{I}(e)$ is the set of all indices $i \in \mathcal{I}$ for which e appears in CRP(i). Therefore, the size of $\mathcal{I}(e)$ can be $O(n)$ in the worst case, which requires $O(n^2)$ homotopy tests where each one takes $O(1)$ time. We show that this process can be performed in $O(n \log n)$ time. From this point forward, we direct our attention to above(e), as below(e) can be handled similarly.

Computing $\min(\text{above}(e))$. One easy way to compute $\min(\text{above}(e))$ for any horizontal edge e of $\mathcal{T}$ is to build a range-search [4] query data structure $\mathcal{T}_{\text{obs}}$ over the obstacles in $\mathcal{H}_1$, which is a two-level tree. For each node ν in the second level of $\mathcal{T}_{\text{obs}}$, we maintain extra information, that is, the minimum rank_s of the obstacles lying at the subtree rooted at ν. Hence, $\min(\text{above}(e))$ can be computed in $O(\log^2 m)$ time. However, this can be performed more quickly as we know all edges in advance. We sweep the horizontal edges and the obstacles from top to bottom and maintain a binary tree over the swept obstacles based on their x-coordinates. When the sweep line reaches an edge e, above(e) is considered as the union of $O(\log m)$ subtrees in the binary tree, and consequently, $\min(\text{above}(e))$ can be computed in $O(\log m)$ time.

Verification of Condition 1. To collectively verify Condition 1 in which an edge e is involved, we define a new ordering of elements in $\mathcal{I}$ such that elements in $\mathcal{I}(e)$ become consecutive. This ordering is obtained by an in-order traversal of $\mathcal{T}$—note that for any $i \in \mathcal{I}$, there is a node in $\mathcal{T}$ labeled $\eta(p_i)$. Let $\sigma(i)$ be the rank of $i \in \mathcal{I}$ in this ordering. As an example, $\langle 3, 2, 4, 8, 9, 10, 6, 7 \rangle$ is the new ordering of the indices in $\mathcal{T}$ of Fig. 5(iii). For the edge e specified in this figure, $\mathcal{I}(e)$ is $\{8, 9, 10\}$ in which indices in $\mathcal{I}(e)$ are consecutive in the new ordering, i.e., they are not fragmented into chunks. This property reduces the verification time of Condition 1 in which an edge e is involved in a 3-sided range query over new points. With each $i \in \mathcal{I}$, we associate point $\check{p}_i = (\sigma(i), \text{rank}_s(p_1 p_i))$ in a new plane $\mathcal{H}$—see Fig. 7. Each edge e defines a segment on the x-coordinate, precisely, σ-coordinate of plane $\mathcal{H}$, which includes all indices in $\mathcal{I}(e)$. Note that the segment corresponding to e can be computed in the in-order traversal of $\mathcal{T}$. Also, $\min(\text{above}(e))$ defines a half-segment in the y-coordinate. Both the segment corresponding to e and the half segment corresponding to $\min(\text{above}(e))$ define a 3-sided range that identifies all indices that violate Condition 1. Fig. 7 illustrates the 3-sided range for edge e specified in Fig. 5(iii). Therefore, every $i \in \mathcal{I}$ whose corresponding point lies in this range violates Condition 1 and, consequently, must be removed from $\mathcal{I}$. As these range-searches are available in advance, we can sweep points and ranges in $\mathcal{H}$ from top to bottom and maintain a binary tree $\mathcal{T}_{\text{bin}}$ over the points based on their x-coordinates, i.e., their σ-coordinate, such as the one described in the previous paragraph. Upon processing a three-sided range, we simply remove $O(\log n)$ subtrees from the binary tree $\mathcal{T}_{\text{bin}}$. Therefore, the sweeping takes $O(n \log n)$ time in total. After handling below(e) for horizontal edges e of $\mathcal{T}$ in a similar manner, any remaining shortcut is a homotopic shortcut. Putting this together, we obtain the following result:

Lemma 7. *Let $\mathcal{P} = p_1, \ldots, p_n$ be a non-self-intersecting polygonal path, and let $\mathcal{S}$ be a set of m point obstacles in a plane. All homotopic shortcuts $p_i p_j$ can be computed in $O(n(m+n)\log(n+m))$ time and $O(n+m+k)$ space where k is the number of homotopic shortcuts.*

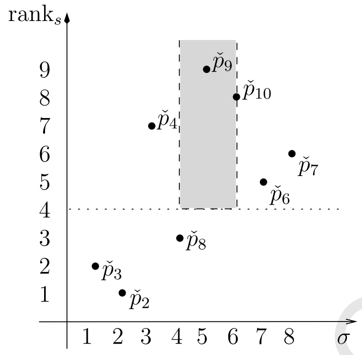

Fig. 7. The new points $(\sigma(i), \mathrm{rank}_s(p_1 p_i))$ in plane $\mathcal{H}$ for the indices appear in Fig. 5(iii). The gray area is the 3-sided range corresponding to $\mathcal{I}(e) = \{8, 9, 10\}$. As points $\check{p}_9$ and $\check{p}_{10}$ lie in the gray area, they violate Condition 1. Thus, $p_1 p_9$ and $p_1 p_{10}$ cannot be homotopic shortcuts. Note that, in the new ordering, the segment corresponding to e in $\mathcal{H}$ is $[4, 6]$. Also, the $\min(\mathrm{above}(e))$ is 4, which is $\mathrm{rank}_s(\widetilde{C})$.

Remark 1. The running time can be improved to $O(m \log m + nm + n^2 \log(n+m))$, if we are allowed to use extra $O(m \log m)$ space. To obtain this improvement, at the beginning, we compute the aboveness relation between the obstacles and edges of $\mathcal{P}$ as well as the aboveness relation between the obstacles and all shortcuts, and we exploit these relations in each iteration, where we fix i to be $1, \ldots, n$. Moreover, we maintain a persistent binary tree [17] over the obstacles to compute all $\min(\mathrm{above}(e))$ in total time $O(n \log m)$ rather than $O(m \log m + n \log m)$ in each iteration. This persistent binary tree occupies $O(m \log m)$ space.

4. Optimal homotopic simplification

In this section, we present the first algorithm that computes the optimal simplification for general paths (the path can be self-intersecting). Our method exploits the observation that the path $\mathcal{P}$ and every simplification of $\mathcal{P}$ can be thought of as strings, namely, their canonical sequence. Therefore, the goal is to find a simplification $\mathcal{Q} = q_1, q_2, \ldots, q_k$ ($q_1 = p_1$ and $q_k = p_n$) with the minimum number of links satisfying $\mathrm{CS}(\mathcal{Q}) = \mathrm{CS}(\mathcal{P})$. We use the dynamic-programming paradigm to find such a simplification. The sub-problem is defined as finding $\mathcal{Q}_{ij}$ to be the optimal simplification for $\mathcal{P}(i, j)$, the path that begins at p_i and ends at p_j. Obviously, $\mathcal{Q}_{1n}$ is the final answer, and each sub-problem should also be solved optimally.

We first compute G_ε under a given distance function F in time $T_F(n)$ (see Section 2). We then compute $\mathrm{CS}(\mathcal{P})$ and $\mathrm{CS}(p_i p_j)$ for every edge $p_i p_j$ in $G_\varepsilon ilon$. Because $p_i p_j$ is just a segment, $\mathrm{CS}(p_i p_j)$ can be easily computed in $O(m)$ time. In fact, for each obstacle, we only need to determine whether it is above or below the shortcut $p_i p_j$, and then sort the resulted sequence based on the x-coordinates of the obstacles. For ease of presentation, we denote the first b symbols of $\mathrm{CS}(p_i p_j)$ by $\mathrm{CS}(b, p_i p_j)$ and the last b symbols of $\mathrm{CS}(p_i p_j)$ by $\mathrm{CS}(p_i p_j, b)$. Moreover, we denote the kth symbol of $\mathrm{CS}(\alpha)$ by $\mathrm{CS}(\alpha)[k]$. When all $\mathrm{CS}(p_i p_j)$s are available, the $\mathrm{CS}(\mathcal{Q})$ can be computed by concatenating the CSs of all links of $\mathcal{Q}$ and then removing the adjacent repeated symbols. Accordingly, we have:

$$\mathrm{CS}(\mathcal{Q}_k) = \begin{cases} \mathrm{RM}(\mathrm{CS}(\mathcal{Q}_{k-1}) \oplus \mathrm{CS}(q_{k-1} q_k)) & \text{if } k > 2, \\ \mathrm{CS}(q_1 q_2) & \text{if } k = 2, \end{cases}$$

where $\mathcal{Q}_i = q_1, q_2, \ldots, q_i$ (the first $i-1$ links of $\mathcal{Q}$), RM is the operator that removes repeated adjacent symbols and $\oplus$ is the concatenation operator.

Our dynamic programming is based on the observation that if $\mathrm{CS}(\mathcal{Q}) = \mathrm{CS}(\mathcal{P})$, any prefix of $\mathrm{CS}(\mathcal{P})$ must be produced by concatenating the CS of the first i links of $\mathcal{Q}$ and the first j symbols of the CS of $(i+1)$th link of $\mathcal{Q}$ for some i and j and then removing the repeated adjacent symbols. Indeed, the last symbol of the prefix of $\mathrm{CS}(\mathcal{P})$ must match a symbol in $\mathrm{CS}(\mathcal{Q})$, as we know $\mathrm{CS}(\mathcal{Q}) = \mathrm{CS}(\mathcal{P})$. Therefore, this symbol must exist in the CS of a link ($(i+1)$th link for some i) of $\mathcal{Q}$, as $\mathrm{CS}(\mathcal{Q})$ is initially obtained by concatenating the CSs of its links. If the position of this symbol in the CS of $(i+1)$th link of $\mathcal{Q}$ is j, then the prefix of $\mathrm{CS}(\mathcal{P})$ is equal to $\mathrm{RM}(\mathrm{CS}(\mathcal{Q}_{i+1}) \oplus \mathrm{CS}(j, q_{i+1} q_{i+2}))$.

The above prefix property guides us to define $\mathrm{seq}(\mathcal{Q}_{1i}, b)$ as follows:

$$\mathrm{seq}(\mathcal{Q}_{1i}, b) = \mathrm{RM}\big(\mathrm{CS}(q_1 q_2) \oplus \mathrm{CS}(q_2 q_3) \oplus \cdots \oplus \mathrm{CS}(q_{r-2} q_{r-1}) \oplus \mathrm{CS}(b, q_{r-1} q_r)\big),$$

where $q_1 = p_1$ and $q_r = p_i$.

We define the main table of our dynamic programming, namely, $\mathrm{OptHS}[i, j, b, l]$. Roughly speaking, $\mathrm{OptHS}[i, j, b, l]$ specifies the minimum number of links needed to construct the first l symbols of $\mathrm{CS}(\mathcal{P})$ such that the last link is $p_i p_j$ and the first b symbols of $\mathrm{CS}(p_i p_j)$ are used to construct the first l symbols of $\mathrm{CS}(\mathcal{P})$. More precisely, for numbers l and b, let $p_i p_j$ be the last link of $\mathcal{Q}_{1j}$ and let $\mathcal{Q}_{1,j}$ be a simplification of $\mathcal{P}(1, j)$ with the minimum number of links such that $\mathrm{seq}(\mathcal{Q}_{1j}, b)$ is equal to exactly the first l symbols of $\mathrm{CS}(\mathcal{P})$. We then store the size of $\mathcal{Q}_{1j}$ (i.e., the number of its links)

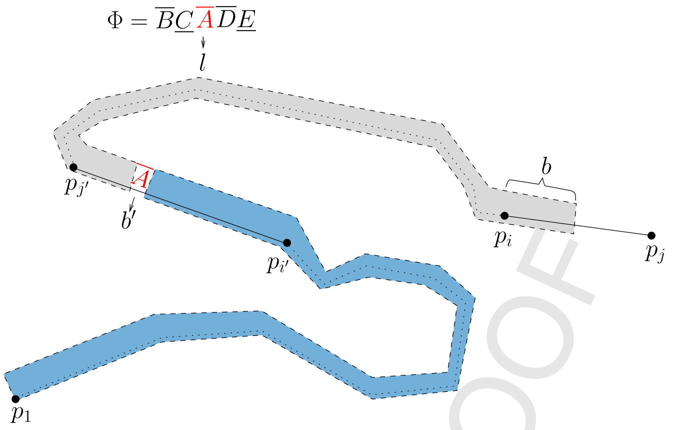
$\Phi = \overline{B}\underline{C}\overline{A}\overline{D}\underline{E}$
l
A
b
$p_{j'}$
b'
$p_{i'}$
p_i
p_j
p_1

Fig. 8. Let $\Phi = \mathrm{CS}(\mathcal{P})$ and $\mathcal{Q}$ specify OptHS$[i, j, b, l]$. There must be a link $p_{i'}p_{j'}$ in $\mathcal{Q}$ such that $\mathrm{CS}(p_{i'}p_{j'})[b'] = \mathrm{CS}(\mathcal{P})[l]$ for some b' and $\mathrm{seq}(i', j', i, j, |\mathrm{CS}(p_{i'}p_{j'})| - b', b)$ is empty.

in OptHS$[i, j, b, l]$. It can easily be shown that $\min_{i=1}^{n}$ OptHS$[i, n, |\mathrm{CS}(p_i p_n)|, |\mathrm{CS}(\mathcal{P})|]$ is equal to the number of links of the optimal homotopic simplification where $|.|$ denotes the size of the sequence.

To complete the OptHS matrix, we often need to know whether the $\mathrm{CS}(\mathcal{Q}_{ij})$ is empty for some i and j. Therefore, we define a new table OptNHS$[i_1, j_1, i_2, j_2, a, c]$ as follows. Consider $\mathcal{Q}_{i_1 j_2}$, which is a simplification of $\mathcal{P}(p_{i_1}, p_{j_2})$. Let $p_{i_1}p_{j_1}$ and $p_{i_2}p_{j_2}$ be the first and the last links of $\mathcal{Q}_{i_1 j_2}$, respectively, and let $\ell_1, \dots, \ell_r$ be the middle links of $\mathcal{Q}_{i_1 j_2}$. We define

$$\mathrm{seq}(i_1, j_1, i_2, j_2, a, c) = \mathrm{RM}\big(\mathrm{CS}(p_{i_1}p_{j_1}, a) \oplus \mathrm{CS}(\ell_1) \oplus \cdots \oplus \mathrm{CS}(\ell_r) \oplus \mathrm{CS}(c, p_{i_2}p_{j_2})\big).$$

If $\mathrm{seq}(i_1, j_1, i_2, j_2, a, c)$ of $\mathcal{Q}_{i_1 j_2}$ is empty and the number of its links is minimum, we store the number of its links in OptNHS$[i_1, j_1, i_2, j_2, a, c]$.

Now, we describe how to fill in the matrices OptHS and OptNHS. The process for computing OptHS$[i, j, b, l]$ and OptNHS$[i_1, j_1, i_2, j_2, a, c]$ is as follows. Let $\mathcal{Q}_{1j}$ specify OptHS$[i, j, b, l]$. There must be a link $p_{i'}p_{j'}$ in $\mathcal{Q}_{1i}$ such that $\mathrm{CS}(p_{i'}p_{j'})[b'] = \mathrm{CS}(\mathcal{P})[l]$ for some b' and $\mathrm{seq}(i', j', i, j, |\mathrm{CS}(p_{i'}p_{j'})| - b', b))$ is empty—see Fig. 8. Thus,

$$\begin{aligned}\mathrm{OptHS}[i, j, b, l] = \min_{i' < j' \leqslant i} \Big(&\mathrm{OptHS}\big[i', j', b' - 1, l - 1\big] \\ &+ \mathrm{OptNHS}\big[i', j', i, j, \big|\mathrm{CS}(p_{i'}p_{j'})\big| - b', b\big] - 1\Big),\end{aligned}$$

where the minimum is taken over all links $p_{i'}p_{j'}$ where $\mathrm{CS}(p_{i'}p_{j'})[b'] = \mathrm{CS}(\mathcal{P})[l]$ for some b'. In a similar way, if $\mathcal{Q}_{i_1 j_2}$ specifies OptNHS$[i_1, j_1, i_2, j_2, a, c]$, there is a link $p_{i'}p_{j'}$ in $\mathcal{Q}_{i_1 j_2}$ such that $\mathrm{CS}(p_{i'}p_{j'})[a'] = \mathrm{CS}(p_{i_1}p_{j_1})[|\mathrm{CS}(p_{i_1}p_{j_1})| - a + 1]$ for some a'. Indeed, $\mathrm{CS}(p_{i'}p_{j'})[a']$ and $\mathrm{CS}(p_{i_1}p_{j_1})[|\mathrm{CS}(p_{i_1}p_{j_1})| - a + 1]$ can eliminate each other if OptNHS$[i_1, j_1, i', j', a - 1, a' - 1]$ is empty—see Fig. 9. Thus,

$$\begin{aligned}\mathrm{OptNHS}[i_1, j_1, i_2, j_2, a, c] = \min_{j_1 \leqslant i' < j' \leqslant i_2} \Big(&\mathrm{OptNHS}\big[i_1, j_1, i', j', a, a' - 1\big] \\ &+ \mathrm{OptNHS}\big[i', j', i_2, j_2, \big|\mathrm{CS}(p_{i'}p_{j'})\big| - a', c\big] - 1\Big).\end{aligned}$$

We refer readers to Appendix A for the precise computations of OptHS$[i, j, b, l]$ and OptNHS$[i_1, j_1, i_2, j_2, a, c]$.

Reporting. OptHS stores the size of the optimal homotopic simplification. Therefore, to report the optimal homotopic simplification, we may need to spend more space and time. As we only need to report the optimal homotopic simplification specifying $\min_{i=1}^{n}$ OptHS$[i, n, |\mathrm{CS}(p_i p_n)|, |\mathrm{CS}(\mathcal{P})|]$, we find it by passing over the matrices once more, without asymptotically increasing the time and space complexities.

Time and space complexity. In matrix OptNHS$[i_1, j_1, i_2, j_2, a, c]$, we have $1 \leqslant i_1, i_2, j_1, j_2 \leqslant n$ and $0 \leqslant a, c \leqslant m$. Furthermore, in matrix OptHS$[i, j, b, l]$, we have $1 \leqslant i, \ j \leqslant n$, and $0 \leqslant b \leqslant m$, and $0 \leqslant l \leqslant nm$. Therefore, OptNHS and OptHS occupy $O(n^3m^2)$ and $O(n^4m^2)$ space, respectively. Note that the size of $\mathrm{CS}(p_i p_j)$ is, at most m, and $|\mathrm{CS}(\mathcal{P})|$ can be, at most, mn as the number of the edges of $\mathcal{P}$ is, at most, $n - 1$. To fill one entry of OptNHS, we take *min* operation over, at most, $O(n^2)$ links $p_{i'}p_{j'}$. As OptNHS has $O(n^4m^2)$ entries, it takes $O(n^6m^2)$ time to fill the whole matrix. Similarly, it takes $O(n^5m^2)$ time to fill the OptHS matrix. Note that finding the symbol of $\mathrm{CS}(p_{i'}p_{j'})$ that matches the lth symbol of $\mathcal{P}$ takes $O(m)$ time, but we can perform it in $O(1)$ time by constructing an auxiliary 2-dimensional array of size $O(n^2m)$ to store the position of each symbol in CS of each link. After computing the matrices, $\min_{i=1}^{n}$ OptHS$[i, n, |\mathrm{CS}(p_i p_n)|, |\mathrm{CS}(\mathcal{P})|]$ can be computed in $O(n)$ time, which does not asymptotically increase the time complexity of our algorithm. Accordingly, we have the following theorem.

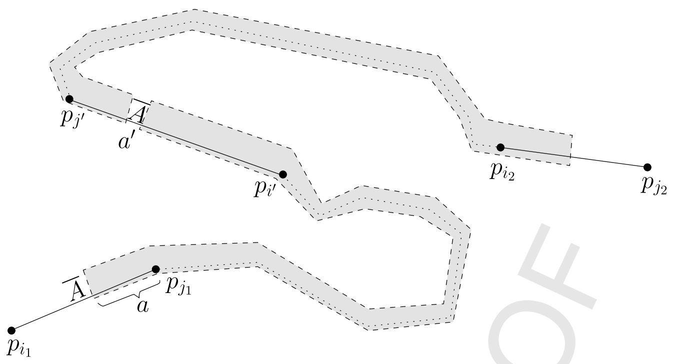

Fig. 9. If $\mathcal{Q}_{i_1 j_2}$ specifies OptNHS$[i_1, j_1, i_2, j_2, a, c]$, there exists a link $p_{i'} p_{j'}$ in $\mathcal{Q}_{i_1 j_2}$ such that $\mathrm{CS}(p_{i'} p_{j'})[a'] = \mathrm{CS}(p_{i_1} p_{j_1})[|p_{i_1} p_{j_1}| - a + 1]$ for some a'— where a' is the index of $\overline{A}$ in $\mathrm{CS}(p_{i'} p_{j'})$. Hence, the CS of both gray areas should be empty.

Theorem 8. *Suppose that $\mathcal{P}$ is a polygonal path with size n in a plane containing the m point obstacles and that $T_F(n)$ is the time needed to compute G_ε under the distance function F in the absence of the obstacles. The optimal homotopic simplification can be computed in $O(n^6 m^2) + T_F(n)$ time and $O(n^4 m^2)$ space.*

5. Conclusions

We have proposed the first polynomial-time algorithm to compute the optimal simplification $\mathcal{Q}$ of a polygonal path $\mathcal{P}$ homotopic to $\mathcal{P}$ with respect to some point obstacles in a plane. We have also presented algorithms to compute the optimal strongly homotopic simplification of an x-monotone and a simple polygonal path $\mathcal{P}$ in $T_F(n) + O(m \log(n+m) + n \log n \log(n+m) + k)$ and $T_F(n) + O(n(m+n)\log(n+m))$ time, respectively, where n is the size of $\mathcal{P}$, m is the number of point obstacles and $T_F(n)$ is the time needed to compute G_ε under the distance function F.

Our algorithms, except for the one proposed for x-monotone paths, are not guaranteed to produce a simple simplification, which can be an important requirement in many areas. We leave this problem for future research.

Appendix A. Computing OptHS and OptNHS

We describe the precise method of computing matrices OptHS and OptNHS. Note that the matrices OptHS and OptNHS maintain the number of links of the minimum subsequence.

OptHS. To compute OptHS$[i, j, b, l]$, where $1 \leqslant i < j \leqslant n$, $0 \leqslant b \leqslant |\mathrm{CS}(p_i p_j)|$, and $0 \leqslant l \leqslant |\mathrm{CS}(\mathcal{P})|$, we distinguish three cases:

- $b = 0$:

$$\mathrm{OptHS}[i, j, 0, l] = \min_{i'} \mathrm{OptHS}\big[i', i, \big|\mathrm{CS}(p_{i'} p_i)\big|, l\big] + 1 \quad \text{where } \big(i' < i\big).$$

- $l = 0$:

$$\mathrm{OptHS}[i, j, b, 0] = \min_{i'} \mathrm{OptNHS}\big[1, i', i, j, \big|\mathrm{CS}(p_1 p_{i'})\big|, b\big] \quad \text{where } \big(i' \leqslant i\big).$$

- $b, l > 0$:

$$\begin{aligned}\mathrm{OptHS}[i, j, b, l] = \min_{i', j', b'} & \mathrm{OptHS}\big[i', j', b' - 1, l - 1\big] \\ & + \mathrm{OptNHS}\big[i', j', i, j, \big|\mathrm{CS}(p_{i'} p_{j'})\big| - b' + 1, b\big] - 1 \\ & \text{where } \big(\big(i' = i, j' = j, b' = b - 1\big) \text{ or } \big(i' < j' \leqslant i\big)\big) \\ & \text{and } \mathrm{CS}(\mathcal{P})[l] = \mathrm{CS}(p_{i'} p_{j'})\big[b'\big].\end{aligned}$$

OptNHS. To fill OptNHS$[i_1, j_1, i_2, j_2, a, c]$, where $1 \leqslant i_1 < j_1 \leqslant i_2 < j_2 \leqslant n$, $0 \leqslant a \leqslant |\mathrm{CS}(p_{i_1} p_{j_1})|$, and $0 \leqslant c \leqslant |\mathrm{CS}(p_{i_2} p_{j_2})|$, we distinguish the following four cases:

- $c > 0$:

$$\begin{aligned}\mathrm{OptNHS}[i_1, j_1, i_2, j_2, a, c] = \min_{i', j', a'} & \mathrm{OptNHS}\bigl[i_1, j_1, i', j', a, a' - 1\bigr] \\ & + \mathrm{OptNHS}\bigl[i', j', i_2, j_2, \bigl|\mathrm{CS}(p_{i'} p_{j'})\bigr| - a', c - 1\bigr] - 1 \\ & \text{where } \bigl(j_1 \leqslant i' < j' \leqslant i_2\bigr) \text{ and } \bigl(0 < a' \leqslant \bigl|\mathrm{CS}(p_{i'} p_{j'})\bigr|\bigr) \\ & \text{and } \mathrm{CS}(p_{i'} p_{j'})\bigl[a'\bigr] = \mathrm{CS}(p_{i_2} p_{j_2})[c].\end{aligned}$$

- $a > 0$ and $c = 0$:

$$\begin{aligned}\mathrm{OptNHS}[i_1, j_1, i_2, j_2, a, 0] = \min_{i', j', a'} & \mathrm{OptNHS}\bigl[i_1, j_1, i', j', a - 1, a' - 1\bigr] \\ & + \mathrm{OptNHS}\bigl[i', j', i_2, j_2, \bigl|\mathrm{CS}(p_{i'} p_{j'})\bigr| - a', 0\bigr] - 1 \\ & \text{where } \bigl(j_1 \leqslant i' < j' \leqslant i_2\bigr) \text{ and } \bigl(0 < l' \leqslant \bigl|\mathrm{CS}(p_{i'} p_{j'})\bigr|\bigr) \\ & \text{and } \mathrm{CS}(p_{i'} p_{j'})\bigl[a'\bigr] = \mathrm{CS}(p_{i_1} p_{j_1})\bigl[\bigl|\mathrm{CS}(p_{i_1} p_{j_1})\bigr| - a + 1\bigr].\end{aligned}$$

- $a, c = 0$ and $j_1 \neq i_2$: $\mathrm{OptNHS}[i_1, j_1, i_2, j_2, 0, 0] = 3$ if $\mathrm{CS}(p_{j_1} p_{i_2}) = \mathrm{NULL}$ and otherwise $+\infty$.
- $a, c = 0$ and $j_1 = i_2$: $\mathrm{OptNHS}[i_1, j_1, i_2, j_2, 0, 0] = 2$ if $\mathrm{CS}(p_{j_1} p_{i_2}) = \mathrm{NULL}$ and otherwise $+\infty$.

References

[1] M.A. Abam, M. de Berg, P. Hachenberger, A. Zarei, Streaming algorithms for line simplification, Discrete Comput. Geom. 43 (3) (2010) 497–515.
[2] P.K. Agarwal, S. Har-Peled, N.H. Mustafa, Y. Wang, Near-linear time approximation algorithms for curve simplification, Algorithmica 42 (3–4) (2005) 203–219.
[3] B. Ben-Moshe, O. Hall-Holt, M.J. Katz, J.S.B. Mitchell, Computing the visibility graph of points within a polygon, in: Proc. Annual Symposium on Computational Geometry, 2004, pp. 27–35.
[4] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, third edition, Springer-Verlag, 2008.
[5] M. de Berg, M. van Kreveld, S. Schirra, A new approach to subdivision simplification, in: Proc. ACSM/ASPRS Annual Convention, Auto-Carto 12 4 (1995) 79–88.
[6] M. de Berg, M. van Kreveld, S. Schirra, Topologically correct subdivision simplification using the bandwidth criterion, Cartogr. Geogr. Inf. Sci. 25 (4) (1998) 243–257.
[7] S. Bespamyatnikh, An optimal morphing between polylines, Int. J. Comput. Geom. Appl. 12 (3) (2002) 217–228.
[8] S. Bespamyatnikh, Computing homotopic shortest paths in the plane, J. Algorithms 49 (2) (2003) 284–303.
[9] L. Buzer, Optimal simplification of polygonal chains for subpixel-accurate rendering, Comput. Geom. 42 (1) (2009) 45–59.
[10] S. Cabello, Y. Liu, A. Mantler, J. Snoeyink, Testing homotopy for paths in the plane, Discrete Comput. Geom. 31 (1) (2004) 61–68.
[11] W.S. Chan, F. Chin, Approximation of polygonal curves with minimum number of line segments, in: Proc. International Symposium on Algorithms and Computation, 650, 1992, pp. 378–387.
[12] B. Chazelle, An algorithm for segment-dragging and its implementation, Algorithmica 3 (1–4) (1988) 205–221.
[13] S. Daneshpajouh, M. Ghodsi, A. Zarei, Computing polygonal path simplification under area measures, Graph. Models 74 (5) (2012) 283–289.
[14] D. Eu, G.T. Toussaint, On approximating polygonal curves in two and three dimensions, CVGIP, Graph. Models Image Process. 56 (3) (1994) 231–246.
[15] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartogr. Int. J. Geogr. Inf. Geovis. 10 (2) (1973) 112–122.
[16] A. Driemel, S. Har-Peled, C. Wenk, Approximating the Fréchet distance for realistic curves in near linear time, Discrete Comput. Geom. 48 (1) (2012) 94–127.
[17] J.R. Driscoll, N. Sarnak, D.D. Sleator, R.E. Tarjan, Making data structures persistent, J. Comput. Syst. Sci. 38 (1) (1989) 86–124.
[18] A. Efrat, L.J. Guibas, S. Har-Peled, J.S.B. Mitchell, T.M. Murali, New similarity measures between polylines with applications to morphing and polygon sweeping, Discrete Comput. Geom. 28 (4) (2002) 535–569.
[19] R. Estkowski, J.S.B. Mitchell, Simplifying a polygonal subdivision while keeping it simple, in: Proc. Annual Symposium on Computational Geometry, 2001, pp. 40–49.
[20] M.T. Goodrich, Efficient piecewise-linear function approximation using the uniform metric, Discrete Comput. Geom. 14 (4) (1995) 445–462.
[21] L.J. Guibas, J. Hershberger, J.S.B. Mitchell, J. Snoeyink, Approximating polygons and subdivisions with minimum-link paths, Int. J. Comput. Geom. Appl. 3 (4) (1993) 383–415.
[22] J. Hershberger, J. Snoeyink, An $O(n \log n)$ implementation of the Douglas–Peucker algorithm for line simplification, in: Proc. Annual Symposium on Computational Geometry, 1994, pp. 383–384.
[23] H. Imai, M. Iri, An optimal algorithm for approximating a piecewise linear function, J. Inf. Process. 9 (3) (1986) 159–162.
[24] H. Imai, M. Iri, Polygonal approximations of a curve – formulations and algorithms, in: G.T. Toussaint (Ed.), Computational Morphology, North-Holland, Amsterdam, Netherlands, 1988, pp. 71–86.
[25] M.W. Krentel, The complexity of optimization problem, J. Comput. Syst. Sci. 36 (3) (1988) 490–509.
[26] A. Melkman, J. O'Rourke, On polygonal chain approximation, in: G.T. Toussaint (Ed.), Computational Morphology, North-Holland, Amsterdam, Netherlands, 1988, pp. 87–95.
[27] L. Palazzi, J. Snoeyink, Counting and reporting red/blue segment intersections, CVGIP, Graph. Models Image Process. 56 (4) (1994) 304–310.