International Journal of Foundations of Computer Science
© World Scientific Publishing Company

Fast Algorithms for Diameter-Optimally Augmenting Paths and Trees*

Ulrike Große and Christian Knauer and Fabian Stehn
Institut für Angewandte Informatik, Universität Bayreuth,
Universitätsstrasse 30, 95447 Bayreuth, Germany
ulrike.grosse|christian.knauer|fabian.stehn@uni-bayreuth.de

Joachim Gudmundsson
School of Information Technologies, J12, The University of Sydney,
NSW 2006, Sydney, Australia
joachim.gudmundsson@sydney.edu.au

Michiel Smid
School of Computer Science, Carleton University
1125 Colonel By Drive, Ottawa, Canada K1S 5B6
michiel@scs.carleton.ca

We consider the problem of augmenting an n-vertex graph embedded in a metric space, by inserting one additional edge in order to minimize the diameter of the resulting graph. We present exact algorithms for the cases when (i) the input graph is a path, running in $O(n \log^3 n)$ time, and (ii) the input graph is a tree, running in $O(n^2 \log n)$ time. We also present an algorithm for paths that computes a $(1+\varepsilon)$-approximation in $O(n + 1/\varepsilon^3)$ time.

Keywords: *Computational Geometry, Geometric Graphs, Diameter, Approximation, Parametric Search, Optimization*

1. Introduction

Let $G = (V, E)$ be a graph in which each edge has a positive *weight*. The *weight* (or *length*) of a path is the sum of the weights of the edges on this path. For any two vertices x and y in V, we denote by $\delta_G(x, y)$ their shortest-path distance, i.e., the minimum weight of any path in G between x and y. The *diameter* of G is defined as $\max\{\delta_G(x, y) \mid x, y \in V\}$.

Assume that we are also given weights for the non-edges of the graph G. In the *Diameter-Optimal r-Augmentation Problem*, $\text{DOAP}(r)$, we have to compute a set F of r edges in $(V \times V) \setminus E$ for which the diameter of the graph $(V, E \cup F)$ is minimum.

*A preliminary version of this paper appeared in the Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP), Part I, Lecture Notes in Computer Science, Vol. 9134, Springer-Verlag, Berlin, 2015, pp. 678–688. M.S. was supported by NSERC. J.G. was supported by the ARCs Discovery Projects funding scheme (DP150101134).

In this paper, we assume that the given graph is a path or a tree on n vertices that is embedded in a metric space, and the weight of any edge and non-edge is equal to the distance between its vertices. We consider the case when $r = 1$; thus, we want to compute one non-edge which, when added to the graph, results in an augmented graph of minimum diameter. Surprisingly, no non-trivial results were known even for the restricted cases of paths and trees.

Throughout the rest of the paper, we assume that $(V, |\cdot|)$ is a metric space, consisting of a set V of n elements (called *points* or *vertices*). The distance between any two points x and y is denoted by $|xy|$. Our contributions are as follows:

(1) If G is a path, we solve the problem DOAP(1) in $O(n \log^3 n)$ time.
(2) If G is a path, we compute a $(1+\varepsilon)$-approximation for DOAP(1) in $O(n + 1/\varepsilon^3)$ time.
(3) If G is a tree, we solve the problem DOAP(1) in $O(n^2 \log n)$ time.

Note that the distance between any two vertices of G can be reported in constant time after spending linear pre-processing time in case that G is a path or after spending quadratic pre-processing time if G is a tree.

1.1. *Related Work*

The Diameter-Optimal r-Augmentation Problem for edge-weighted graphs and many of its variants have been shown to be NP-hard [23], or even $W[2]$-hard [12, 13]. Because of this, several special classes of graphs have been considered. Chung and Gary [6] and Alon et al. [1] considered paths and cycles with unit edge weights and gave upper and lower bounds on the diameter that can be achieved. Ishii [15] gave a constant factor approximation algorithm (approximating both r and the diameter) for the case when the input graph is outerplanar. Erdős et al. [10] investigated upper and lower bounds for the case when the augmented graph must be triangle-free.

The general problem: The Diameter-Optimal Augmentation Problem can be seen as a bicriteria optimization problem: In addition to the weight, each edge and non-edge has a cost associated with it. Then the two optimization criteria are (1) the total cost of the edges (denoted by B) added to the graph and (2) the diameter of the augmented graph. We say that an algorithm is (α, β)-*approximation algorithm* for the DOAP problem, with $\alpha, \beta \geq 1$, if it computes a set F of non-edges of total cost at most $\alpha \cdot B$ such that the diameter of $G' = (V, E \cup F)$ is at most $\beta \cdot D^B_{\text{opt}}$, where D^B_{opt} is the diameter of an optimal solution that augments the graph with edges of total cost at most B.

For the restricted version when all costs and all weights are identical [3, 5, 9, 16, 17], Bilò et al. [3] showed that, unless P=NP, there does not exist a $(c \log n, \delta < 1 + 1/D^B_{\text{opt}})$-approximation algorithm for DOAP if $D^B_{\text{opt}} \geq 2$. For the case in which $D^B_{\text{opt}} \geq 6$, they proved that, again unless P=NP, there does not exist a $(c \log n, \delta < \frac{5}{3} - \frac{7-(D^B_{\text{opt}}+1) \bmod 3}{3D^B_{\text{opt}}})$-approximation algorithm.

The algorithm by Li et al. [17] constituted a $(1, 4+2/D^B_{\text{opt}})$ approximation. Bilò et al. [3] improved the analysis of that algorithm and showed that it actually gives a $(1, 2+2/D^B_{\text{opt}})$-approximation and additinally gave an $(O(\log n), 1)$-approximation algorithm.

For general costs and weights, Dodis and Khanna [9] gave an $O(n \log D^B_{\text{opt}}, 1)$-approximation algorithm. Their result is based on a multi-commodity flow formulation of the problem. Frati et al. [12] recently considered the DOAP problem with arbitrary integer costs and weights. Their main result is a $(1, 4)$-approximation algorithm with running time $O((3^B B^3 + n + \log(Bn))Bn^2)$.

Oh et al. [21] present an $O(n^2 \log^3 n)$ algorithm to compute the shortcut that minimizes the diameter of an arbitrarily weighted tree (where the weights have not necessarily been induced by a metric). They also present an algorithm of the same runtime to solve the continuous version of the problem.

Geometric graphs: In the geometric setting, when the input is a geometric graph embedded in the Euclidean plane, there are only few results on graph augmentation in general. Rutter and Wolff [22] proved that the t-vertex-connectivity and t-edge-connectivity augmentation problems (that is adding the smallest number of edges, so that the resulting graph remains planar and is t-connected) are NP-hard on plane geometric graphs, for $t = 2, 3, 4$, and 5; the problem is infeasible for $t \geq 6$ because every planar graph has a vertex of degree at most 5. A connected graph is called *t-edge-connected* (*t-vertex-connected*), if it remains connected after removing any set of up to $t - 1$ edges (vertices). Currently, there are no known approximation algorithms for this problem.

The problem of adding one edge to a geometric graph while minimizing the dilation had been studied Farshi et al. [11] who gave approximation algorithms for this problem that are faster than the trivial algorithm to compute an exact solution.

The *dilation* of an embedded weighted graph G is defined as the largest ratio of the shortest path length between two any vertices of G and their distance in the embedded space. There were several follow-up papers [18, 26], but there is still no non-trivial result known for the case when $r > 1$.

In the *continuous* version of the diameter-optimal augmentation problem, the input graph G is embedded in the plane and the edges to be added to G can have their endpoints anywhere *on* G, i.e., the endpoints can be in the interior of edges of G. Moreover, the diameter is considered as the maximum of the shortest-path distances over all points *on* G. Yang [27] considered the continuous version of the problem of adding one edge to a path so as to minimize the continuous diameter. He presented sufficient and necessary conditions for an augmenting edge to be optimal. He also presented an approximation algorithm, having an *additive* error of ϵ, that runs in $O((n + |P|/\epsilon)^2 n)$ time, where $|P|$ denotes the length of the input path P and ϵ is at most half of the length of a shortest edge in P. De Carufel et al. [4] improved the running time to $O(n)$ and also considered the continuous version of the problem for cycles that are embedded in the plane. They showed that it is not possible to decrease the continuous diameter of a cycle by adding one edge to it. On the other hand, two edges can always be added that decrease the continuous diameter. De Carufel et al. gave a full characterization of the optimal two edges. If the input cycle is convex, they find the optimal pairs of edges in $O(n)$ time.

De Carufel et al. [8] extended their result for paths, and showed that, in the continuous version, an optimal shortcut for a tree with n vertices can be computed in $O(n \log n)$ time.

Recently, our result of the DOAP(1) for paths from the conference version of this article [14] was improved by Wang [25] to run in $O(n \log n)$ by replacing the parametric search technique we use. His algorithm first computes and then filters a candidate set of optimal values for the diameter, and then applies sorted-matrix searching techniques to solve the optimization problem.

2. Augmenting a Path with One Edge

We are given a path $P = (p_1, \dots, p_n)$ on n vertices in a metric space and assume that it is stored in an array $P[1, \dots, n]$. To simplify notation, we associate a vertex with its index, that is $p_k = P[k]$ is also referred to as k for $1 \le k \le n$. This allows us to extend the total order of the indices to the vertex set of P. We denote the start vertex of P by s and the end vertex of P by e.

For $1 \le k < l \le n$, we denote the subpath $(p_k, \dots, p_l)$ of P by $P[k, l]$, the cycle we get by adding the edge from p_k to p_l, denoted by $\overline{p_k p_l}$, to $P[k, l]$ by $C[k, l]$, and the (unicyclic) graph we get by adding the edge $\overline{p_k p_l}$ as a *shortcut* to P by $\overline{P}[k, l]$; the length of $X \in \{P, P[k, l], C[k, l]\}$ is denoted by $|X|$. We will consider the functions $\overline{p}_{k,l} := \delta_{\overline{P}[k,l]}$ and $c_{k,l} := \delta_{C[k,l]}$, where $\delta_G : V^2 \to \mathbb{R}^+$ is the length of the shortest path between two vertices in G. For $1 \le k < l \le n$, we let

$$M_P(k, l) := \max_{1 \le x < y \le n} \overline{p}_{k,l}(x, y)$$

denote the *diameter* of the graph $\overline{P}[k, l]$. From now on, we will omit the subscript P as all functions implicitly will refer to the input path P.

Our goal is to compute a shortcut $\overline{p_k p_l}$ for P that minimizes the diameter of the resulting unicyclic graph. That is, we want to compute a shortcut such that the resulting unicyclic graph has a diameter of

$$m(P) := \min\nolimits_{1 \le k < l \le n} M(k, l).$$

2.1. *Algorithm*

In the following, we will describe an algorithm that adds an optimal shortcut to a path to minimize its diameter.

The algorithm consists of two parts. We first describe a sequential algorithm for the *decision problem.* Given P and a threshold parameter $\lambda > 0$, decide if $m(P) \le \lambda$ (see Lemma 1 a) below). In a second step, we argue that the sequential algorithm can be implemented in a parallel fashion (see Lemma 1 b) below), thus enabling us to use the parametric search paradigm of Megiddo [19, 20].

Lemma 1. *Given a path P on n vertices in a metric space and a real parameter $\lambda > 0$, we can decide in*

a) *$O(n \log n)$ time, or in*
b) *$O(\log n)$ parallel time using n processors*

whether $m(P) \le \lambda$; the algorithms also produce a feasible shortcut if it exists.

Proof. To prove this lemma, observe that

$$m(P) \le \lambda \text{ if and only if } \bigvee_{1 \le k < l \le n} (M(k,l) \le \lambda)\,.$$

The algorithm checks, for each $1 \le k < n$, whether there is some $k < l \le n$ such that $M(k,l) \le \lambda$. If one such index k is found, we know that $m(P) \le \lambda$; otherwise $m(P) > \lambda$. Clearly this approach also produces a feasible shortcut if it exists.

We decompose the function $M(k,l)$ into four monotone (in the second parameter) parts, see Figure 1. This will facilitate our search for a feasible shortcut and enable us to do (essentially) binary search: For $1 \le k < l \le n$, we let

$$S(k,l) := \max_{k \le x \le l} \overline{p}_{k,l}(s,x), \qquad E(k,l) := \max_{k \le x \le l} \overline{p}_{k,l}(x,e),$$
$$U(k,l) := \overline{p}_{k,l}(s,e), \qquad O(k,l) := \max_{k \le x < y \le l} c_{k,l}(x,y).$$

Then we have $M(k,l) = \max\{S(k,l), E(k,l), U(k,l), O(k,l)\}$. The triangle inequality implies that

$$S(k,l) \le S(k,l+1), \qquad E(k,l) \ge E(k,l+1),$$
$$U(k,l) \ge U(k,l+1), \qquad O(k,l) \le O(k,l+1).$$

The function U is easy to evaluate once we have the array $D[1,\ldots,n]$ of the prefix-sums of the edge lengths: $D[i] := \sum_{1 \le j < i} |p_j p_{j+1}|$. These sums can be computed in $O(n)$ time sequentially or in $O(\log n)$ time using n processors (see Chapter 30.1.2 in [7]), which in turn allows us to report the path length of any path connecting two vertices of P in constant time. If in addition to D, the vertices $s' = \max\{u \,|\, \delta_P(s,u) \le \lambda\}$ and $e' = \min\{v \,|\, \delta_P(v,e) \le \lambda\}$ are computed for a fixed λ in $O(\log n)$ time (via binary search on D), the following decision problems can be answered in constant time (for fixed k and l):

$$S(k,l) \le \lambda, \quad E(k,l) \le \lambda, \quad U(k,l) \le \lambda.$$

$U(k,l)$ can be computed directly using D and $S(k,l) \le \lambda$ ($E(k,l) \le \lambda$) exactly if $\overline{p}_{k,l}(s, s'+1) \le \lambda$ ($\overline{p}_{k,l}(e, e'-1) \le \lambda$).

We denote the maximum of these three functions by

$$N(k,l) = \max(S(k,l), E(k,l), U(k,l)).$$

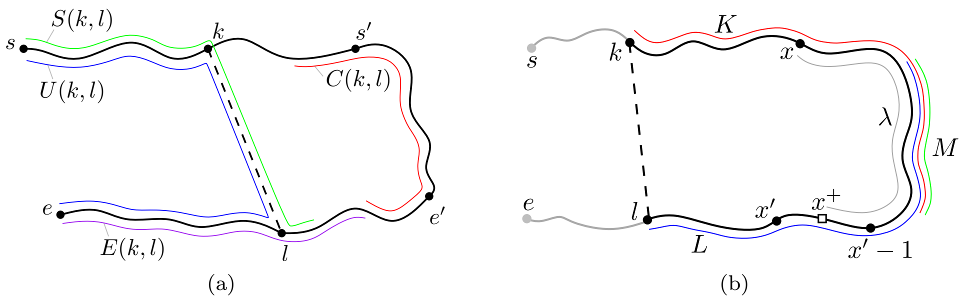

Fig. 1: **(a)** Illustration of the four distances that define the diameter of a shortcut $\overline{p_k p_l}$: $U(k,l)$ is the length of the shortest path connecting s and e; $O(k,l)$ is the length of the longest shortest path between any two points in $C[k,l]$; $S(k,l)$ ($E(k,l)$) is the length of the longest shortest path from s (e) to any vertex in $C(k,l)$. **(b)** Illustration of the computation of $O(k,l)$.

Now clearly

$$M(k,l) = \max(N(k,l), O(k,l))$$

and, consequently

$$M(k,l) \leq \lambda \text{ if and only if } N(k,l) \leq \lambda \text{ and } O(k,l) \leq \lambda.$$

For fixed $1 \leq k < n$, the algorithm will first check whether there is some $k < l \leq n$ with $N(k,l) \leq \lambda$. If no such l exists, we can conclude that $M(k,l) > \lambda$ for all $k < l \leq n$. The monotonicity of S, E, and U implies that, for fixed $1 \leq k < n$, the set

$$N_k := \{k < l \leq n \mid N(k,l) \leq \lambda\}$$

is an *interval*. This interval can be computed (using binary search in P and in D as described above) in $O(\log n)$ time. If $N_k = \emptyset$ we can conclude that for the $1 \leq k < n$ under consideration and for all $k < l \leq n$, we have that $M(k,l) > \lambda$.

If N_k is non-empty, the monotonicity of O implies that it is sufficient to check for $l_k = \min N_k$ (i.e. the starting point of the interval) whether $O(k, l_k) \leq \lambda$:

$$\exists k < l \leq n : O(k,l) \leq \lambda \text{ if and only if } O(k,l_k) \leq \lambda.$$

Note that in this case we know that $N(k,l_k) \leq \lambda$.

Deciding the diameter of small cycles: We now describe how to decide for a given shortcut $1 \leq k < l \leq n$ if $O(k,l) \leq \lambda$, given that *we already know that* $N(k,l) \leq \lambda$. To this end, consider the following sets of vertices from $C[k,l]$: $K := \{k \leq x \leq l \mid \delta_P(k,x) \leq \lambda\}$, $L := \{k \leq x \leq l \mid \delta_P(x,l) \leq \lambda\}$, $T := K \cap L$, $K' := K \setminus L$, $L' := L \setminus K$.

These sets are intervals and can be computed in $O(\log n)$ time by binary search. Since $N(k,l) \leq \lambda$, we can conclude the following:

- the set of vertices of $C[k,l]$ is $K \cup L$
- $c_{k,l}(x,y) \le \lambda$ for all $x,y \in K$
- $c_{k,l}(x,y) \le \lambda$ for all $x,y \in L$
- $c_{k,l}(x,y) \le \lambda$ for all $x \in T$, $y \in C[k,l]$

Consequently, if $c_{k,l}(x,y) > \lambda$ for $x,y \in C[k,l]$, we can conclude that $x \in K'$ and $y \in L'$. In order to establish that $O(k,l) \le \lambda$, it therefore suffices to verify that

$$\bigwedge_{x \in K', y \in L'} c_{k,l}(x,y) \le \lambda.$$

Note that on P any vertex x of K' is at more than λ away from the vertex l, i.e., $\delta_P(x,l) > \lambda$. For a point $x \in K'$, let x^+ be the point on (a vertex or an edge of) P that is at distance exactly λ to x (along P) and that is on the subpath from x to l (see Figure 1b), i.e., x^+ is the unique point on P such that

$$\delta_P(x^+,l) < \delta_P(x,l) \text{ and } \delta_P(x,x^+) = \lambda.$$

The next (in the direction of l) *vertex* of P will be denoted by x', i.e., $x < x' \le l$ is the unique vertex of P such that

$$\delta_P(x,x'-1) \le \lambda \text{ and } \delta_P(x,x') > \lambda.$$

Since x is a vertex of K', x' is a vertex of L'. For the following discussion we denote the distance achieved in $C[k,l]$ by using the shortcut by $c^+_{k,l}$ and the distance achieved by travelling along P only by $c^-_{k,l}$, i.e.,

$$c^-_{k,l}(x,y) := \delta_P(x,y) \text{ and } c^+_{k,l}(x,y) := \delta_P(x,k) + |\overline{p_k p_l}| + \delta_P(l,y).$$

Clearly

$$c_{k,l}(x,y) = \min(c^+_{k,l}(x,y), c^-_{k,l}(x,y)), \text{ and } |C[k,l]| = c^+_{k,l}(x,y) + c^-_{k,l}(x,y).$$

For every vertex $y < x'$ on L' we have that $c_{k,l}(x,y) \le c^-_{k,l}(x,y) \le \lambda$, so if there is some vertex $y \in L'$ with $y \ne x'$ such that $c_{k,l}(x,y) > \lambda$, we know that $x' < y \le l$; in that case we have that $c^+_{k,l}(x,y) \le c^+_{k,l}(x,x')$. Since we assume that $c_{k,l}(x,y) > \lambda$, we also know that $c^+_{k,l}(x,y) > \lambda$ and we can conclude that $c^+_{k,l}(x,x') > \lambda$, and consequently that $c_{k,l}(x,x') > \lambda$, i.e., for all $x \in K'$ we have that

$$\bigwedge_{y \in L'} c_{k,l}(x,y) \le \lambda \text{ if and only if } c_{k,l}(x,x') \le \lambda.$$

The distance between (the point) x^+ and (the vertex) x' on P is called the *defect* of x and is denoted by $\Delta(x)$, i.e., $\Delta(x) = \delta_P(x^+,x')$.

Next, we have to show that

$$c_{k,l}(x,x') \le \lambda \text{ if and only if } |C[k,l]| \le \Delta(x) + 2\lambda. \qquad (1)$$

Observe that

$$\begin{aligned}|C[k,l]| &= \delta_P(x,k) + |\overline{p_k p_l}| + \delta_P(l,x') + \delta_P(x',x^+) + \delta_P(x^+,x)\\ &= \delta_P(x,k) + |\overline{p_k p_l}| + \delta_P(l,x') + \Delta(x) + \lambda\\ &= c^+_{k,l}(x,x') + \Delta(x) + \lambda.\end{aligned}$$

Since $c^-_{k,l}(x,x') > \lambda$, we have that $c_{k,l}(x,x') \leq \lambda$ if and only if $c^+_{k,l}(x,x') \leq \lambda$; the claim stated in (1) follows.

To summarize the above discussion, we have the following chain of equivalences (here $\Delta_{k,l} := |C[k,l]| - 2\lambda$):

$$O(k,l) \leq \lambda \Leftrightarrow \bigwedge_{x \in K'} c_{k,l}(x,x') \leq \lambda \Leftrightarrow \bigwedge_{x \in K'} \Delta_{k,l} \leq \Delta(x) \Leftrightarrow \min_{x \in K'} \Delta(x) \geq \Delta_{k,l}.$$

Since K' is an interval, the last condition can be tested easily after some preprocessing: To this end we compute a $1d$-range tree on D (see Chapter 5.3 in [2]) and associate with each vertex in the tree the minimum Δ-value of the corresponding canonical subset. For every *vertex* x of P that is at least λ away from the end vertex of P we can compute $\Delta(x)$ in $O(\log n)$ time by binary search in D. With these values the range tree can be built in $O(n \log n)$ time. A query for an interval K' then gives us $\mu := \min_{x \in K'} \Delta(x)$ in $O(\log n)$ time and we can check the above condition in $O(1)$ time.

We describe the algorithm in pseudocode; see Algorithm 1.

The correctness of the algorithm follows from the previous discussion. ComputePrefixSums runs in $O(n)$ time, ComputeRangeTree runs in $O(n \log n)$ time, ComputeFeasibleIntervalForN runs in $O(\log n)$ time, a call to CheckOForShortcut requires $O(\log n)$ time. The total runtime is therefore $O(n \log n)$. It is easy to see that with n processors, the steps ComputePrefixSums and ComputeRangeTree can be realized in $O(\log n)$ parallel time and that with this number of processors, all calls to CheckOForShortcut can be handled in parallel. Therefore, the entire algorithm can be parallelized and has a parallel runtime of $O(\log n)$. □

Megiddo developed in [19, 20] the parametric search technique, whose essence is stated in the following theorem (see also [24]).

Theorem 2 (Megiddo) *Given a decision algorithm $\mathcal{A}_S$ with runtime T_S that decides problem P and a parallel version $\mathcal{A}_P$ that uses X processors with runtime T_P, the smallest value of a* YES*-instance of P can be computed in $O(XT_P + T_S T_P \log X)$ time.*

Plugging the result of Lemma 1 into this machinery directly gives our main result.

Theorem 3. *Given a path P on n vertices in a metric space, we can compute $m(P)$, and a shortcut realizing that diameter, in $O(n \log^3 n)$ time.*

Algorithm 1: Algorithm for deciding if $m(P) \leq \lambda$

```
   DecisionAlgorithm($P, \lambda$) ;                      // Decide if $m(P) \leq \lambda$
1  begin
       global $D \leftarrow$ ComputePrefixSums($P$);
       global $s' \leftarrow \max\{v \mid \delta_P(s,v) \leq \lambda\}$;
       global $e' \leftarrow \min\{v \mid \delta_P(v,e) \leq \lambda\}$;
       global $T \leftarrow$ ComputeRangeTree($P, \lambda$);
       for $1 \leq k < n$ do
           $N_k \leftarrow$ ComputeFeasibleIntervalForN($k, \lambda$);
           if $N_k \neq \emptyset$ and CheckOForShortcut($k, \min(N_k), \lambda$) then
               return True
       return False
   end
   CheckOForShortcut($k, l, \lambda$) ;                  // Decide if $O(k,l) \leq \lambda$
2  begin
       $K' \leftarrow \{k \leq x \leq l \mid \delta_P(k,x) \leq \lambda \wedge \delta_P(x,l) > \lambda\}$;     // Compute the
       interval by binary search
       $\mu \leftarrow \min_{x \in K'} \Delta(x)$ ;                  // Query the range tree $T$
       return  $(\mu \geq |C[k,l]| - 2\lambda)$
   end
```

From the above discussion, we note that, since there are only four possible distances to compute in order to determine the diameter of a path augmented with one shortcut edge, we can derive the following corollary (as all four distances that determine the diameter can be computed in linear time).

Corollary 4. *Given a path P on n vertices in a metric space and a shortcut (u,v), the diameter of $P \cup (u,v)$ can be computed in $O(n)$ time.*

3. An Approximation Algorithm

As before, we are given a path $P = (p_1, p_2, \ldots, p_n)$ on n vertices in a metric space. Any pair p, q of distinct vertices of P is called a *shortcut* for P. The graph that results by adding pq as an edge to P will be denoted by $P + pq$. For any two vertices x and y of P, we denote the shortest-path distances between x and y in P and $P + pq$ by $\delta_P(x,y)$ and $\delta_{P+pq}(x,y)$, respectively. The diameter of $P + pq$, i.e., $\max_{x,y} \delta_{P+pq}(x,y)$, where x and y range over all vertices of P, will be denoted by $Diam(P + pq)$.

In this section, we will present a linear-time approximation algorithm for computing a shortcut for which the diameter of the augmented network is minimized.

Throughout the rest of this section, we assume, without loss of generality, that the total length of the path P, i.e., $\sum_{i=1}^{n-1} |p_i p_{i+1}|$, is equal to 1.

Let α be a real number with $0 < \alpha < 1$. We define a subsequence $r_1, r_2, \dots$ of the vertices of P:

(1) Let $r_1 := p_1$.
(2) For any $i \geq 1$, assume that $r_1, r_2, \dots, r_i$ have been defined. Let k be the index such that $r_i = p_k$. Let

$$j := \min\{\ell : \ell \geq k+1, \delta_P(r_i, p_\ell) > \alpha\}.$$

If the index j exists, then we define $r_{i+1} := p_j$. Otherwise, the construction of the sequence terminates and $r_{i+1} = p_n$ is added to the sequence unless R already ended on p_n (that is unless $r_i = p_n$).

Let $r_1, r_2, \dots, r_m$ be the sequence obtained in this way. Since the total length of the path P is equal to 1, we have $m \leq \lceil 1/\alpha \rceil + 1$.

We define the path

$$R := (r_1, r_2, \dots, r_m)$$

and give each edge $r_i r_{i+1}$ a weight defined by

$$\omega(r_i, r_{i+1}) := \delta_P(r_i, r_{i+1}).$$

Observation 1. *For each vertex p of P, there is a vertex r of R such that $\delta_P(p, r) \leq \alpha$.*

Our approximation algorithm will compute a shortcut rs for the path R. The weight of this shortcut will be equal to $|rs|$. For any two vertices x and y of R, we denote the shortest-path distances between x and y in R and $R+rs$ by $\delta^\omega_R(x,y)$ and $\delta^\omega_{R+rs}(x,y)$, respectively. Observe that $\delta^\omega_R(x,y) = \delta_P(x,y)$. The diameter of $R+rs$ will be denoted by $Diam^\omega(R+rs)$.

3.1. *The Algorithm*

Let ϵ be a real number with $0 < \epsilon < 1$. On input P, the algorithm does the following:

Step 1: Compute the path R as defined above, where $\alpha = \epsilon/22$.

Step 2: For each pair r, s of distinct vertices of R, compute $Diam^\omega(R+rs)$.

Step 3: Return the pair a, b found in Step 2 for which $Diam^\omega(R+ab)$ is minimized.

Step 1 can obviously be performed in $O(n)$ time. Since the path R has $O(1/\epsilon)$ vertices, Step 2 computes the value of $Diam^\omega(R+rs)$ for $O(1/\epsilon^2)$ pairs r, s. Since, for any given pair r, s, the value of $Diam^\omega(R+rs)$ can be computed in $O(1/\epsilon)$ time, the total time for Step 2 is $O(1/\epsilon^3)$. Finally, Step 3 takes $O(1)$ time. Thus, the total time of the algorithm is $O(n + 1/\epsilon^3)$.

In the following subsection, we will analyze the approximation factor of our algorithm.

3.2. *The Approximation Factor*

The analysis of the approximation factor is based on a sequence of lemmas.

Lemma 5. *The following claims hold:*

(1) For any shortcut pq of P, $Diam(P + pq) \geq 1/3$.
(2) For any shortcut rs of R, $Diam^{\omega}(R + rs) \geq 1/3$.

Proof. We start by proving the second claim. Recall that $\delta^{\omega}_R(r, s) = \delta_P(r, s)$. By the triangle inequality, we have $|rs| \leq \delta_P(r, s)$. If $|rs| = \delta_P(r, s)$, then

$$Diam^{\omega}(R + rs) = Diam^{\omega}(R) = Diam(P) = 1 \geq 1/3.$$

Thus, we may assume that $|rs| < \delta_P(r, s)$.

Walk along the path R, from r to s, and let y be the first vertex for which the shortest path, in $R + rs$, from r to y uses the shortcut rs. Let x be the vertex of R that is immediately before y. Observe that x and y exist. Also, the shortest path, in $R + rs$, from s to y does not use the shortcut rs, and the shortest path, in $R + rs$, from x to y does not use the shortcut rs.

Let $D := Diam^{\omega}(R + rs)$. We have

$$\omega(x, y) = \delta^{\omega}_{R+rs}(x, y) \leq D$$

and

$$\begin{aligned} \delta^{\omega}_{R+rs}(r_1, x) + \omega(x, y) + \delta^{\omega}_{R+rs}(y, r_m) &= \delta^{\omega}_R(r_1, x) + \omega(x, y) + \delta^{\omega}_R(y, r_m) \\ &= \delta_P(r_1, x) + \delta_P(x, y) + \delta_P(y, r_m) \\ &= 1, \end{aligned}$$

implying that

$$\delta^{\omega}_{R+rs}(r_1, x) + \delta^{\omega}_{R+rs}(y, r_m) = 1 - \omega(x, y) \geq 1 - D.$$

Since both $\delta^{\omega}_{R+rs}(r_1, x)$ and $\delta^{\omega}_{R+rs}(y, r_m)$ are at most equal to D, we get

$$1 - D \leq 2D,$$

implying that $D \geq 1/3$, proving the second claim.

Observe that the second claim holds for any value of α. If we choose α to be less than the minimum edge weight of the path P, then R is equal to P and, thus, the first claim follows as well. □

Lemma 6. *Let pq be an arbitrary shortcut for P, let r be a vertex of R such that $\delta_P(p, r) \leq \alpha$, and let s be a vertex of R such that $\delta_P(q, s) \leq \alpha$. Then, for any two vertices x and y of R, we have*

$$\delta^{\omega}_{R+rs}(x, y) \leq \delta_{P+pq}(x, y) + 4\alpha.$$

Proof. First assume that the shortest path between x and y in $P + pq$ does not use the shortcut pq. Then we have

$$\delta^{\omega}_{R+rs}(x,y) \leq \delta^{\omega}_{R}(x,y) = \delta_P(x,y) = \delta_{P+pq}(x,y) \leq \delta_{P+pq}(x,y) + 4\alpha.$$

In the rest of the proof, we assume that the shortest path between x and y in $P+pq$ uses the shortcut pq. We may assume, without loss of generality, that this shortest path starts by going from x to p, then takes the shortcut from p to q, and finally goes from q to y. Since

$$\begin{aligned} |rs| &\leq |rp| + |pq| + |qs| \\ &\leq \delta_P(r,p) + |pq| + \delta_P(q,s), \end{aligned}$$

we have

$$\begin{aligned} \delta^{\omega}_{R+rs}(x,y) &\leq \delta^{\omega}_{R}(x,r) + |rs| + \delta^{\omega}_{R}(s,y) \\ &= \delta_P(x,r) + |rs| + \delta_P(s,y) \\ &\leq \delta_P(x,p) + \delta_P(p,r) + |rs| + \delta_P(s,q) + \delta_P(q,y) \\ &\leq \delta_P(x,p) + 2\cdot\delta_P(p,r) + |pq| + 2\cdot\delta_P(s,q) + \delta_P(q,y) \\ &\leq \delta_P(x,p) + 2\alpha + |pq| + 2\alpha + \delta_P(q,y) \\ &= \delta_P(x,p) + |pq| + \delta_P(q,y) + 4\alpha \\ &= \delta_{P+pq}(x,y) + 4\alpha. \end{aligned}$$ □

Lemma 7. *Let pq be an arbitrary shortcut for P, let r be a vertex of R such that $\delta_P(p,r) \leq \alpha$, and let s be a vertex of R such that $\delta_P(q,s) \leq \alpha$. Then,*

$$\mathit{Diam}^{\omega}(R+rs) \leq (1+12\alpha)\cdot \mathit{Diam}(P+pq).$$

Proof. Let x and y be arbitrary vertices of R. By applying Lemmas 5 and 6, we have

$$\begin{aligned} \delta^{\omega}_{R+rs}(x,y) &\leq \delta_{P+pq}(x,y) + 4\alpha \\ &\leq \mathit{Diam}(P+pq) + 12\alpha\cdot\mathit{Diam}(P+pq). \end{aligned}$$ □

Lemma 8. *Let rs be an arbitrary shortcut for R. Then,*

$$\mathit{Diam}(P+rs) \leq (1+6\alpha)\cdot\mathit{Diam}^{\omega}(R+rs).$$

Proof. Let x and y be arbitrary vertices of P, let x' be a vertex of R such that $\delta_P(x,x') \leq \alpha$, and let y' be a vertex of R such that $\delta_P(y,y') \leq \alpha$. We have

$$\begin{aligned} \delta_{P+rs}(x,y) &\leq \delta_P(x,x') + \delta_{P+rs}(x',y') + \delta_P(y',y) \\ &\leq \alpha + \delta_{P+rs}(x',y') + \alpha \\ &= \delta^{\omega}_{R+rs}(x',y') + 2\alpha \\ &\leq \mathit{Diam}^{\omega}(R+rs) + 2\alpha. \end{aligned}$$

Using Lemma 5, we obtain

$$\delta_{P+rs}(x,y) \leq \mathit{Diam}^{\omega}(R+rs) + 6\alpha \cdot \mathit{Diam}^{\omega}(R+rs). \qquad \square$$

We are now ready to prove the approximation factor of our algorithm:

Lemma 9. *Let pq be an optimal shortcut for the path P. Let ab be the shortcut for R that is computed by our algorithm. Then*

$$\mathit{Diam}(P+ab) \leq (1+\epsilon) \cdot \mathit{Diam}(P+pq).$$

Proof. Let r be a vertex of R such that $\delta_P(p,r) \leq \alpha$ and let s be a vertex of R such that $\delta_P(q,s) \leq \alpha$. First observe that

$$\mathit{Diam}^{\omega}(R+ab) \leq \mathit{Diam}^{\omega}(R+rs).$$

By Lemma 8, we have

$$\begin{aligned}\mathit{Diam}(P+ab) &\leq (1+6\alpha) \cdot \mathit{Diam}^{\omega}(R+ab)\\ &\leq (1+6\alpha) \cdot \mathit{Diam}^{\omega}(R+rs).\end{aligned}$$

Applying Lemma 7, we obtain

$$\mathit{Diam}(P+ab) \leq (1+6\alpha)(1+12\alpha) \cdot \mathit{Diam}(P+pq).$$

We conclude the proof by observing that, since $0 < \epsilon < 1$ and $\alpha = \epsilon/22$,

$$(1+6\alpha)(1+12\alpha) \leq 1+\epsilon. \qquad \square$$

The following theorem summarizes the results of this section:

Theorem 10. *Given a path P on n vertices in a metric space and given a real number ϵ with $0 < \epsilon < 1$, we can compute, in $O(n + 1/\epsilon^3)$ time, a shortcut ab for P such that the diameter of $P+ab$ is at most $(1+\epsilon) \cdot D_{opt}$, where D_{opt} is the diameter of an optimal solution.*

4. Augmenting a Tree with One Edge

Next we consider the case when the input graph is a tree $T = (V, E)$, where V is a set of n vertices in a metric space. The aim is to compute an edge f in $(V \times V) \setminus E$ such that the diameter of the resulting unicyclic graph $(V, E \cup \{f\})$ is minimized. The naive approach of trying all possible shortcuts and computing the diameter of the resulting unicyclic graphs results in a runtime of $O(n^3 \log n)$ using the $O(n \log n)$ algorithm to compute the diameter of a unicyclic graph by Oh et al. [21].

Let P_T be the common intersection of all heaviest paths in T. Observe that P_T is a non-empty path in T. We denote the endvertices of P_T by a and b. Let $F = T \setminus E(P_T)$ be the forest that results from deleting the edges of P_T from T. For any vertex u of T,

(1) let $\sigma(u)$ be the vertex on P_T that is in the same tree of F as u, and

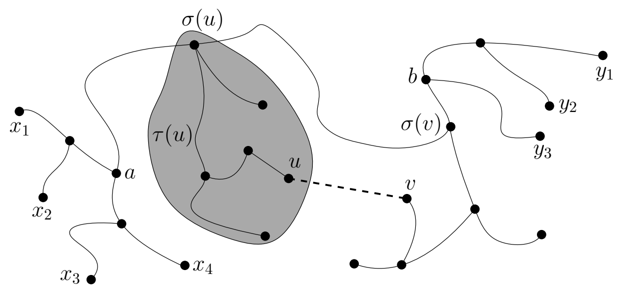

Fig. 2: Illustrating the input tree T with (u, v) as an optimal shortcut. The paths in T between x_i and y_j, for $1 \leq i \leq 4$ and $1 \leq j \leq 3$, represent all longest paths in T. These paths intersect in the path between a and b.

(2) let $\tau(u)$ be the tree of F that contains u.

Refer to Figure 2 for an illustration.

Consider any augmenting edge (u, v). In the following lemma, we will prove that the augmenting edge $(\sigma(u), \sigma(v))$ is at least as good as (u, v). That is, the diameter of $T \cup \{(\sigma(u), \sigma(v))\}$ is at most the diameter of $T \cup \{(u, v)\}$. In case $\sigma(u) = \sigma(v)$, $T \cup \{(\sigma(u), \sigma(v))\}$ is equal to T, and the diameter of $T \cup \{(u, v)\}$ is equal to the diameter of T.

Lemma 11. *There exists an optimal augmenting edge f for T such that both vertices of f are vertices of P_T.*

Proof. Consider an optimal augmenting edge (u, v). We may assume without loss of generality that $\sigma(u)$ is on the subpath of P_T between a and $\sigma(v)$, see Figure 2.

Let $T_{opt} = T \cup \{(u, v)\}$, let D_{opt} be the diameter of T_{opt}, and let $T' = T \cup \{(\sigma(u), \sigma(v))\}$. In order to prove the lemma, it suffices to show that the diameter of T' is at most D_{opt}. If D_{opt} is equal to the diameter of T, then this obviously holds, because the diameter of T' is at most the diameter of T. Thus, from now on, we assume that D_{opt} is less than the diameter of T.

We claim that there exist endvertices x and y of some longest path in T such that

(1) a is on the path in T between x and $\sigma(u)$,
(2) b is on the path in T between y and $\sigma(v)$,
(3) x and u are not in the same connected component of $\tau(u) \setminus \{\sigma(u)\}$,
(4) y and $v1$ are not in the same connected component of $\tau(v) \setminus \{\sigma(v)\}$.

To prove this, consider the leaves $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_\ell$ of T such that

(1) for each i with $1 \leq i \leq k$, a is on the path in T between x_i and b,
(2) for each j with $1 \leq j \leq \ell$, b is on the path in T between y_j and a,

(3) for each i and j with $1 \leq i \leq k$ and $1 \leq j \leq \ell$, the path in T between x_i and y_j is a longest path in T, and each longest path in T is between some x_i and some y_j.

Refer to Figure 2. Recall that P_T, the path from a to b in T, is the intersection of all heaviest paths in T. If $k = 1$, then $x_1 = a$ and we take $x = x_1$. Assume that $k \geq 2$. Consider the maximal subtree of T that contains a and all leaves $x_1, x_2, \ldots, x_k$, and imagine this subtree to be rooted at a. There is a child a' of a such that u is not in the subtree rooted at a'. We take x to be any x_i that is in the subtree rooted at a'. By a symmetric argument, we can prove the existence of the vertex y.

Recall that we assume that the diameter of T_{opt} (i.e., D_{opt}) is less than the diameter of T. This implies that the shortest path in T_{opt} from x to y contains the shortcut (u, v) and, therefore,

$$\delta_T(\sigma(u), u) + |uv| + \delta_T(v, \sigma(v)) < \delta_T(\sigma(u), \sigma(v)). \tag{2}$$

In particular, $\sigma(u) \neq \sigma(v)$.

Now let s and t be any pair of vertices. In the rest of the proof, we will show that $\delta_{T'}(s, t) \leq D_{opt}$. Up to symmetry, there are three main cases to consider with respect to the positions of s and t:

(1) *Both vertices are in trees of F that contain the shortcut vertices: $s, t \in \tau(u) \cup \tau(v)$*, see Figure 3.

(a) *The vertices are in different trees of F: $s \in \tau(u)$ and $t \in \tau(v)$.*
Since

$$\delta_{T'}(s, \sigma(u)) = \delta_T(s, \sigma(u)) \leq \delta_T(x, \sigma(u)) = \delta_{T'}(x, \sigma(u))$$

and

$$\delta_{T'}(\sigma(v), t) = \delta_T(\sigma(v), t) \leq \delta_T(\sigma(v), y) = \delta_{T'}(\sigma(v), y),$$

we have

$$\begin{aligned}
\delta_{T'}(s, t) &= \delta_{T'}(s, \sigma(u)) + |\sigma(u)\sigma(v)| + \delta_{T'}(\sigma(v), t) \\
&\leq \delta_{T'}(x, \sigma(u)) + |\sigma(u)\sigma(v)| + \delta_{T'}(\sigma(v), y) \\
&= \delta_{T'}(x, y) \\
&= \delta_T(x, \sigma(u)) + |\sigma(u)\sigma(v)| + \delta_T(\sigma(v), y) \\
&\leq \delta_T(x, \sigma(u)) + \delta_T(\sigma(u), u) + |uv| + \delta_T(\sigma(v), v) + \delta_T(\sigma(v), y) \\
&= \delta_{T_{opt}}(x, y) \\
&\leq D_{opt}.
\end{aligned}$$

(b) *The vertices are in the same tree of F: $s, t \in \tau(u)$.*
We will prove that in this case, the shortest paths between s and t in both T' and T_{opt} do not contain the shortcut, i.e., both of these shortest paths

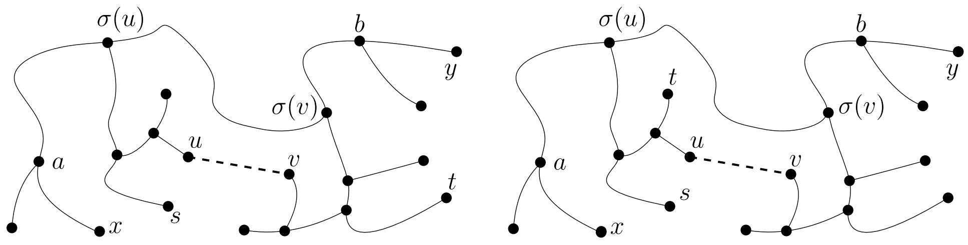

Fig. 3: Illustrating (left) case 1(a) and (right) case 1(b).

are equal to the path in $\tau(u)$ (and, thus, in T) between s and t. This will imply that

$$\delta_{T'}(s,t) = \delta_{T_{opt}}(s,t) \leq D_{opt}.$$

Consider the shortest path $P'(s,t)$ between s and t in T'. Observe that shortest paths do not contain repeated vertices. If $P'(s,t)$ contains the shortcut $(\sigma(u), \sigma(v))$, then this path visits the vertex $\sigma(u)$ twice. Thus, $P'(s,t)$ does not contain $(\sigma(u), \sigma(v))$.

Consider the shortest path $P_{opt}(s,t)$ from s to t in T_{opt}, and assume that this path contains (u,v). We may assume without loss of generality that, starting at s, this path traverses (u,v) from u to v. (Otherwise, we interchange s and t.) Since $P_{opt}(s,t)$ does not contain repeated vertices, this path contains the subpath in T from $\sigma(v)$ to $\sigma(u)$. This subpath must be the shortest path in T_{opt} between $\sigma(v)$ and $\sigma(u)$. However, as we have seen in (2), this is not the case. Thus, we conclude that $P_{opt}(s,t)$ does not contain (u,v).

(2) *Neither vertices are in trees of F that contain the shortcut vertices: $s,t \notin \tau(u) \cup \tau(v)$.*
If the shortest path in T_{opt} from s to t does not contain (u,v), then

$$\delta_{T'}(s,t) \leq \delta_T(s,t) = \delta_{T_{opt}}(s,t) \leq D_{opt}.$$

Assume that this shortest path contains (u,v). We may assume without loss of generality that this shortest path traverses the edge (u,v) from u to v. We have

$$\begin{aligned}\delta_{T'}(s,t) &\leq \delta_T(s,\sigma(u)) + |\sigma(u)\sigma(v)| + \delta_T(\sigma(v),t)\\ &\leq \delta_T(s,\sigma(u)) + \delta_T(\sigma(u),u) + |uv| + \delta_T(v,\sigma(v)) + \delta_T(\sigma(v),t)\\ &= \delta_{T_{opt}}(s,t)\\ &\leq D_{opt}.\end{aligned}$$

(3) *One vertex is in a tree of F that contains a shortcut vertex, the other is not:* $s \in \tau(u)$ and $t \notin \tau(u) \cup \tau(v)$.

(a) *t is a vertex in the maximal subtree of T having x and $\sigma(u)$ as leaves*, see Figure 4(left).
As in Case 1(b), it can be shown that the shortest paths between s and t

(as well as the shortest paths between s and x) in both T_{opt} and in T' do not contain the shortcut. Thus,

$$\delta_{T'}(s,t) \le \delta_{T'}(s,x) = \delta_T(s,x) = \delta_{T_{opt}}(s,x) \le D_{opt}.$$

(b) *t is a vertex in the maximal subtree of T having $\sigma(u)$ and $\sigma(v)$ as leaves*, see Figure 4(right).
We first observe that

$$\begin{aligned}\delta_{T'}(s,t) &= \delta_T(s,\sigma(u)) + \delta_{T'}(\sigma(u),t)\\ &\le \delta_T(x,\sigma(u)) + \delta_{T'}(\sigma(u),t)\\ &= \delta_{T'}(x,t).\end{aligned}$$

If the shortest path in T_{opt} from x to t does not contain (u,v), then

$$\delta_{T'}(x,t) \le \delta_T(x,t) = \delta_{T_{opt}}(x,t) \le D_{opt}.$$

Assume that the shortest path in T_{opt} from x to t contains (u,v). Then

$$\delta_{T_{opt}}(x,t) = \delta_T(x,\sigma(u)) + \delta_T(\sigma(u),u) + |uv| + \delta_T(v,\sigma(v)) + \delta_T(\sigma(v),t).$$

Observe that

$$\delta_{T'}(x,t) \le \delta_T(x,\sigma(u)) + |\sigma(u)\sigma(v)| + \delta_T(\sigma(v),t).$$

The triangle inequality implies that

$$\delta_{T'}(x,t) \le \delta_{T_{opt}}(x,t) \le D_{opt}.$$

(c) t is a vertex in the maximal subtree of T having $\sigma(v)$ and y as leaves.
In this case, we have

$$\begin{aligned}\delta_{T'}(s,t) &= \delta_T(s,\sigma(u)) + |\sigma(u)\sigma(v)| + \delta_T(\sigma(v),t)\\ &\le \delta_T(x,\sigma(u)) + |\sigma(u)\sigma(v)| + \delta_T(\sigma(v),y)\\ &\le \delta_T(x,\sigma(u)) + \delta_{T_{opt}}(\sigma(u),\sigma(v)) + \delta_T(\sigma(v),y)\\ &= \delta_{T_{opt}}(x,y)\\ &\le D_{opt}.\end{aligned}$$

This concludes the proof of the lemma. □

As a consequence of Lemma 11, the diameter of a tree cannot be improved by adding a single shortcut, if the intersection of all longest paths is a vertex or a single edge.

For a tree T with n vertices, let the intersection of all longest paths in T be the path P_T. In a preprocessing step, we convert T into a caterpillar tree T_{cp} by replacing every tree T' of $T \setminus E(P_T)$ by a single edge of length $\delta_T(t,v)$, where v is the common vertex of T' and P_T, and t is the furthest vertex in T' to v, see Figure 5. Note that T_{cp} has a unique longest path.

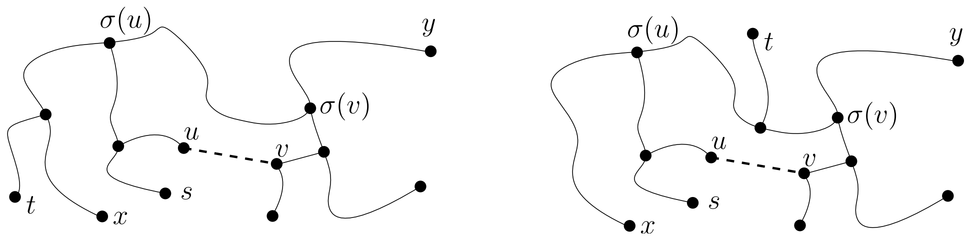

Fig. 4: Illustrating (left) case 3(a) and (right) case 3(b).

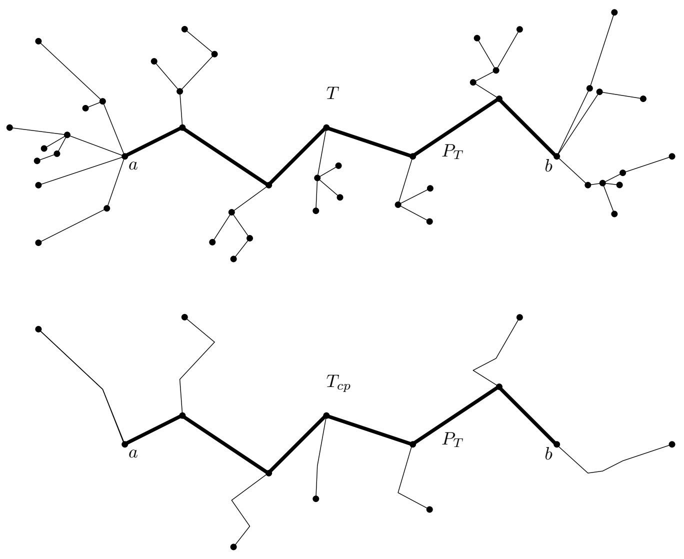

Fig. 5: Illustrating the conversion of the tree T embedded in the Euclidean plane to the caterpillar T_{cp}, where subtrees *dangling* from P_T (the path from a to b) are compressed to a single edge.

Recall that for adding an edge to a path, there are only four relevant distances to compute to determine the diameter; the same holds for a tree with a unique longest path. These distances can trivially be computed in $O(n)$ time, using a post-order traversal strategy. Now consider the case when one of the endpoints of the shortcut is fixed at a vertex v and the second endpoint is moving along P_T in T_{cp}. As for the path case, the four functions describing the distances are monotonically increasing or decreasing, hence, a simple binary search along P_T for the second endpoint can be used to determine the optimal placement of the shortcut. As a result, the optimal shortcut, given one fixed endpoint v of the shortcut, can be computed in $O(n \log n)$ time. We get:

Theorem 12. *Given a tree T on n vertices in a metric space, we can compute a* *shortcut that minimizes the diameter of the augmented graph in* $O(n^2 \log n)$ *time.*

Recall that Lemma 11 states that there exists an optimal shortcut with both its endpoints on P_T. However, our algorithm only exploits that one of the endpoints is on P_T. The obvious question is if one can modify the algorithm so that it takes full advantages of the lemma.

5. Concluding remarks

In this paper we studied the problem of augmenting an n-vertex path or tree embedded in a metric space, by inserting one additional edge in order to minimize the diameter of the resulting graph. We presented an $O(n \log^3 n)$ time algorithm for paths and an $O(n^2 \log n)$ time algorithm for trees. We also gave a $(1 + \varepsilon)$-approximation algorithm for paths with a running time of $O(n + 1/\varepsilon^3)$.

There are many interesting open problems remaining. In the non-metric case it is known that augmenting an edge-weighted graph with r edges is $W[2]$-hard. The existing hardness proofs can not be easily modified to the metric case, hence a hardness result for this case remains an open problem. As only special cases have been shown to have efficient algorithms, the most obvious open problem is to develop exact or approximation algorithms for more general settings. That is, are there efficient algorithms for augmenting a graph with r edges while minimizing the diameter? A brute force algorithm for the problem has a running time of $O(n^{2r+3})$. Can this be improved? Even algorithms for restricted inputs such as graphs with bounded treewidth or planar graphs would be interesting. Or is it possible to show conditional lower bounds for these restricted cases?

Acknowledgments

Part of this work was done at the *17th Korean Workshop on Computational Geometry*, held on Hiddensee Island in Germany, June 22–27, 2014. We thank the other workshop participants for their helpful comments. We also thank Carsten Grimm for his comments on the proof of Lemma 11.

References

[1] N. Alon, A. Gyárfás and M. Ruszinkó, Decreasing the diameter of bounded degree graphs, *Journal of Graph Theory* **35** (1999) 161–172.

[2] M. d. Berg, O. Cheong, M. v. Kreveld and M. Overmars, *Computational Geometry: Algorithms and Applications* (Springer-Verlag TELOS, Santa Clara, CA, USA, 2008).

[3] D. Bilò, L. Gualà and G. Proietti, Improved approximability and non-approximability results for graph diameter decreasing problems, *Theoretical Computer Science* **417** (2012) 12–22.

[4] J.-L. D. Carufel, C. Grimm, A. Maheshwari and M. Smid, Minimizing the continuous diameter when augmenting paths and cycles with shortcuts, *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*, **53** (2016), pp. 27:1–27:14.

[5] V. Chepoi and Y. Vaxès, Augmenting trees to meet biconnectivity and diameter constraints, *Algorithmica* **33**(2) (2002) 243–262.

[6] F. R. K. Chung and M. R. Garey, Diameter bounds for altered graphs, *Journal of Graph Theory* **8**(4) (1984) 511–534.

[7] T. H. Cormen, C. Stein, R. L. Rivest and C. E. Leiserson, *Introduction to Algorithms*, 1st edn. (MIT Press and McGraw-Hill, 1990).

[8] J.-L. De Carufel, C. Grimm, S. Schirra and M. Smid, Minimizing the continuous diameter when augmenting a tree with a shortcut, *Algorithms and Data Structures*, eds. F. Ellen, A. Kolokolova and J.-R. Sack (Springer International Publishing, Cham, 2017), pp. 301–312.

[9] Y. Dodis and S. Khanna, Designing networks with bounded pairwise distance, *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, (1999), pp. 750–759.

[10] P. Erdős, A. Gyárfás and M. Ruszinkó, How to decrease the diameter of triangle-free graphs, *Combinatorica* **18**(4) (1998) 493–501.

[11] M. Farshi, P. Giannopoulos and J. Gudmundsson, Improving the stretch factor of a geometric network by edge augmentation, *SIAM Journal on Computing* **38**(1) (2005) 226–240.

[12] F. Frati, S. Gaspers, J. Gudmundsson and L. Mathieson, Augmenting graphs to minimize the diameter, *Algorithmica* (2014) 1–16.

[13] Y. Gao, D. R. Hare and J. Nastos, The parametric complexity of graph diameter augmentation, *Discrete Applied Mathematics* **161**(10–11) (2013) 1626–1631.

[14] U. Große, J. Gudmundsson, C. Knauer, M. Smid and F. Stehn, Fast algorithms for diameter-optimally augmenting paths, *Automata, Languages, and Programming*, eds. M. M. Halldórsson, K. Iwama, N. Kobayashi and B. Speckmann (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015), pp. 678–688.

[15] T. Ishii, Augmenting outerplanar graphs to meet diameter requirements, *Journal of Graph Theory* **74** (2013) 392–416.

[16] S. Kapoor and M. Sarwat, Bounded-diameter minimum-cost graph problems, *Theory of Computing Systems* **41**(4) (2007) 779–794.

[17] C.-L. Li, S. T. McCormick and D. Simchi-Levi, On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems, *Operations Research Letters* **11**(5) (1992) 303–308.

[18] J. Luo and C. Wulff-Nilsen, Computing best and worst shortcuts of graphs embedded in metric spaces, *19th International Symposium on Algorithms and Computation*, *Lecture Notes in Computer Science*, (Springer, 2008).

[19] N. Megiddo, Combinatorial optimization with rational objective functions, *Math. Oper. Res.* **4** (1979) 414–424.

[20] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *Journal of the ACM* **30**(4) (1983) 852–865.

[21] E. Oh and H.-K. Ahn, A Near-Optimal Algorithm for Finding an Optimal Shortcut of a Tree, *27th International Symposium on Algorithms and Computation (ISAAC 2016)*, ed. S.-H. Hong *Leibniz International Proceedings in Informatics (LIPIcs)* **64**, (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2016), pp. 59:1–59:12.

[22] I. Rutter and A.Wolff, Augmenting the connectivity of planar and geometric graphs, *Journal of Graph Algorithms and Applications* **16**(2) (2012) 599–628.

[23] A. A. Schoone, H. L. Bodlaender and J. van Leeuwen, Diameter increase caused by edge deletion, *Journal of Graph Theory* **11** (1997) 409–427.

[24] R. van Oostrum and R. C. Veltkamp, Parametric search made practical, *Computa-*

tional Geometry **28**(2) (2004) 75 – 88.

[25] H. Wang, An improved algorithm for diameter-optimally augmenting paths in a metric space, *Algorithms and Data Structures*, eds. F. Ellen, A. Kolokolova and J.-R. Sack (Springer International Publishing, Cham, 2017), pp. 545–556.

[26] C. Wulff-Nilsen, Computing the dilation of edge-augmented graphs in metric spaces, *Computational Geometry - Theory and Applications* **43**(2) (2010) 68–72.

[27] B. Yang, Euclidean chains and their shortcuts, *Theoretical Computer Science* **497** (2013) 55–67.