Theoretical Computer Science 412 (2011) 3530–3536

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

An exact algorithm for minimum distortion embedding☆

Fedor V. Fomin[a,*], Daniel Lokshtanov[b], Saket Saurabh[c]

[a] Department of Informatics, University of Bergen, Norway
[b] Department of Computer Science and Engineering, University of California, San Diego, USA
[c] The Institute of Mathematical Sciences, Chennai, India

ARTICLE INFO

Article history:
Received 18 March 2010
Received in revised form 10 February 2011
Accepted 22 February 2011
Communicated by J. Kratochvil

Keywords:
Exact algorithm
Metric embedding
Distortion

ABSTRACT

Let G be an unweighted connected graph on n vertices. We show that an embedding of the shortest path metric of G into the line with minimum distortion can be found in time $5^{n+o(n)}$. This is the first algorithm breaking the trivial $n!$-barrier.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Given an undirected connected graph G with the vertex set $V(G)$ and the edge set $E(G)$, the *graph metric* of G is $M(G) = (V(G), D_G)$, where the distance function D_G is the shortest path distance between u and v for every pair of vertices $u, v \in V(G)$. Given a graph metric M and another metric space M' with distance functions D and D', a mapping $f : M \to M'$ is called an *embedding* of M into M'. The mapping f has *contraction* c_f and *expansion* e_f if for every pair of points p, q in M,

$$D(p, q) \leq D'(f(p), f(q)) \cdot c_f,$$

and

$$D(p, q) \cdot e_f \geq D'(f(p), f(q))$$

respectively. We say that f is *non-contracting* if c_f is at most 1. A non-contracting mapping f has *distortion* d if e_f is at most d.

In this paper we provide an exact algorithm for the following fundamental problem: for a given graph G, find a minimum distortion embedding of the graph metric of G into the line. In this case the metric space M' is $\mathbb{R}^1$ and D' is the Euclidean distance. A simple algorithm is to try all possible permutations of the vertex set. Each permutation corresponds to an embedding where the distance between two consecutive vertices on the line is equal to the shortest path distance between them. The running time of this algorithm is $O(n!n)$ and to the best of our knowledge, no faster exact algorithm for any kind of embedding problem was known prior to our work.

The problem of finding an embedding with low distortion between metric spaces is a fundamental mathematical problem [12,14] that has been studied intensively. Embedding a graph metric into a simple low-dimensional metric space like the real line has proved to be a useful algorithmic tool in various fields. A long list of applications given in [11] includes approximation algorithms for graph and network problems, such as sparsest cut, minimum bandwidth, low-diameter decomposition and optimal group Steiner trees, and on-line algorithms for metrical task systems and file migration problems. The algorithmic issues of metric embeddings have recently begun to develop [1–3,13]. For example, Bădoiu et al. [1] describe approximation algorithms and hardness results for embedding general metrics into the line. In particular, they show that the minimum

☆ Preliminary version of this paper was presented at WG'09 [10].

* Corresponding author. Tel.: +47 555818181.
E-mail addresses: fedor.fomin@ii.uib.no (F.V. Fomin), dlokshtanov@cs.ucsd.edu (D. Lokshtanov), saket@imsc.res.in (S. Saurabh).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.02.043

distortion for a line embedding is hard to approximate up to a factor polynomial in n, even for weighted trees where the ratio of maximum/minimum weights is bounded by a polynomial in n. For the case of unweighted graphs, it was shown by Bădoiu et al. [2] that there is a constant $a > 1$, such that a-approximation of the minimum distortion of an embedding into the line is *NP*-hard. Bădoiu et al. also provided an exact algorithm for computing an embedding with distortion at most d in time $n^{O(d)}$. For $d = \Omega(n)$ the running time of such an algorithm is $n^{O(n)}$. Fellows et al. [8] studied the parameterized complexity of metric embeddings and proved that embedding into the line and more generally, into trees with bounded vertex degrees, is fixed parameter tractable when parameterized by the distortion. For embedding a graph metric into the line the running time of the algorithm described in [8] is $O(nd^4(2d+1)^{2d})$, which also does not break the barrier of $n!$ when $d = \Omega(n)$.

It is worth to mention the resemblance between the problem of embedding into the line and the Bandwidth Minimization problem. In the Bandwidth Minimization problem the objective is for a given graph G to find a bijective mapping $f : V(G) \to \{1, \ldots, n\}$, for which the *bandwidth*, that is $b = \max_{(u,v)\in E(G)} |f(u) - f(v)|$, is minimized. Observe that the only difference between the two problems is that in the Bandwidth Minimization problem we demand $1 \leq |f(p) - f(q)|$ for every pair of vertices while the non-contraction constraint in our embedding problem is $D(p, q) \leq |f(p) - f(q)|$.

The Bandwidth Minimization problem is one of the test-bed problems in the area of moderately exponential time algorithms and has been studied intensively. Trying all possible permutations of the vertex set yields a simple $O(n!n)$ time algorithm while the known algorithms for the problem with running time $O(c^n)$ are far from straightforward. The $O(n!)$-barrier was broken by Feige and Kilian [7] who gave an algorithm with running time $10^n n^{O(1)}$. This result was subsequently improved by Cygan and Pilipczuk down to $5^n n^{O(1)}$ in [4] and then to $4.383^n n^{O(1)}$ in [5].

Despite the similarities between low distortion embedding into the line and bandwidth, the non-contraction constraint makes the algorithmic complexity of the two problems significantly different. A striking example is that the parameterized version of the Bandwidth Minimization problem is one of the hardest problems in Parameterized Complexity, while low distortion embedding into the line is fixed parameter tractable [8]. Thus, it is not surprising that a direct transmission of the ideas for the Bandwidth Minimization problem to low distortion embeddings does not work. Nevertheless, our approach is still based on the approaches from [4,7], especially the initial and final parts of our algorithm. However, to handle non-contraction we need a non-trivial additional link connecting these parts.

2. Preliminaries

Let G be an undirected graph with vertex set $V(G)$ and edge set $E(G)$. We denote the number of vertices by n. For u and $v \in V(G)$, $D_G(u, v)$ is the shortest path distance between u and v in G. For a subset $V' \subseteq V(G)$, by $G[V']$ we mean the subgraph of G induced by V'. An *integer interval* is a set $\{x, x+1, \ldots, y-1, y\}$ of integers appearing consecutively. An embedding of a graph G into the line is a function $f : V(G) \to \mathbb{R}$. The *distortion* of an embedding f is $\max_{u,v\in V(G)} \frac{|f(u)-f(v)|}{D_G(u,v)}$. An embedding is called *non-contracting* if $|f(u) - f(v)| \geq D_G(u, v)$ for every pair u, v of vertices. If f is non-contracting we say that a vertex u *pushes* vertex v if $D_G(u, v) = |f(u) - f(v)|$. For an embedding f, let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices such that $f(v_1) < f(v_2) < \cdots < f(v_n)$. We say that f is *pushing* if v_i pushes v_{i+1}, for each $1 \leq i \leq n-1$.

A *partial embedding* of G into the line is a function $f' : V' \to \mathbb{R}$ for some subset V' of V. For a partial embedding f' with domain V', let $v'_1, v'_2, \ldots, v'_{n'}$ be an ordering of V' such that $f'(v'_1) < f'(v'_2) < \cdots < f(v'_{n'})$. We say that f' is *pushing* if v'_i pushes v'_{i+1}, for each $1 \leq i \leq n'-1$. The distortion of a pushing partial embedding f' is $\max_{uv\in E(G[V'])} |f'(u) - f'(v)|$.

3. Exact algorithm for distortion

In this section we give an exact algorithm for the following problem.

> Given an input graph G with the vertex set $V(G)$ and the edge set $E(G)$, find a mapping f from $V(G) \to \mathbb{R}^+$ such that for all $u, v \in V(G)$, $|f(u) - f(v)| \geq D_G(u, v)$ and the function
>
> $$dist(G) = \max_{u,v\in V(G)} \frac{|f(u) - f(v)|}{D_G(u, v)}$$
>
> is minimized.

In order to reduce the search space we apply a simple lemma proved in [8] on minimum distortion embedding of graphs into the line.

Lemma 1 ([8]). • *If G can be embedded into the line with distortion d, then there is a pushing embedding of G into the line with distortion d. Furthermore, every pushing embedding of G into the line is non-contracting.*
• *Let f be a pushing embedding of a connected graph G into the line with distortion at most d. Then $D(v_{i-1}, v_i) \leq d$ for every $1 \leq i \leq n$.*

Lemma 1 implies that it is sufficient to look for an optimal pushing embedding. Notice that a pushing embedding with $f(v_1) = 0$ maps every vertex to an integer coordinate. Therefore we can without loss of generality restrict ourself to functions $f : V(G) \to \{0, \ldots, dn\}$. We also assume that our input graph G is *connected*, because otherwise some pair of vertices have infinite distance between them and hence there is no non-contracting embedding of G into the line.

EXACT-DIST(G, d, h, J, g)
(Here d is the distortion, h is the fixed bucket assignment, $J = \{x, \ldots, y\}$ is the set of indices of buckets and g is a partial embedding of some of the vertices in the graph.)

1. If $|J| > \frac{n}{\log^2 n}$, then find a bucket V_j of the kind described in Lemma 2 else go to Step 3.
2. Enumerate all possible pushing partial embeddings $g_j : V_j \to \mathcal{B}_j$ of distortion at most d. For every such g_j:
   - Assign $g'(v) = g_j(v)$ if $v \in V_j$ and $g'(v) = g(v)$ if v is in the domain of g. Let $J_1 = \{x, \ldots, j-1, j\}$ and $J_2 = \{j, j+1, \ldots, y\}$. Recursively solve the subproblems EXACT-DIST(G, d, h, J_1, g') and EXACT-DIST(G, d, h, J_2, g'). Return "YES" if both recursive calls return "YES".
3. In this case solve the problem using Lemma 5 of Section 3.3.

Fig. 1. Description of the Algorithm.

We now present an algorithm that decides whether there is an embedding of distortion at most d for the input graph G. It is well known that any graph G with n vertices can be embedded into the line with distortion at most $2n - 1$ [2]. Thus, if we want to find the minimum d such that there is an embedding of G into the line with distortion at most d it is sufficient to try all values between 1 and $2n - 1$ for d. Next we describe the three main components of our algorithm and show how to combine them in order to obtain an algorithm running in time $5^{n+o(n)}$ and using $2^{n+o(n)}$ space. The first and third parts of our algorithm go along the lines of the known algorithms for BANDWIDTH [7,4]. While these two parts are sufficient to compute bandwidth, in order to solve our problem we need an intermediate divide-and-conquer step to bridge the first and last parts.

3.1. Fixing an assignment into buckets

The algorithm loops over all possible distributions of the vertices into "buckets" on the integer line. The remaining two steps of the algorithm deal with finding an optimal embedding that agrees with the distribution made in the first step. Formally, we are looking for a pushing embedding $f : V(G) \to \{0, \ldots, dn\}$. A *bucket assignment* is a function $h : V(G) \to \{0, \ldots, n\}$ and an embedding $f : V(G) \to \{0, \ldots, dn\}$ of G *agrees* with h if for every vertex v of G we have $h(v) = \lfloor \frac{f(v)}{d+1} \rfloor$. For $i \geq 0$, the ith bucket of h (or the ith bucket for short) is $\mathcal{B}_i = \{(d+1)i, \ldots, (d+1)(i+1) - 1\}$ and the content of the ith bucket is $V_i = \{v : h(v) = i\}$.

The outer loop of the algorithm goes over a set of bucket assignments such that if there is a pushing embedding $f : V(G) \to \{0, \ldots, dn\}$ with distortion at most d then some h we have looped over agrees with f. We guess a vertex v such that $h(v) = 0$ and fix a spanning tree T of G with r_T as root. Once $h(p)$ has been determined for the parent p of a node u in T, we loop over all possible values of $h(u)$. If h is to agree with some pushing embedding $f : V(G) \to \{0, \ldots, dn\}$ with distortion at most d we have that $h(u) = h(p) - 1$, $h(u) = h(p)$ or $h(u) = h(p) + 1$ and that $h(u) \geq 0$. Since we have at most 3 possibilities for the placement of each vertex the outer loop needs only to go over at most $n \cdot 3^n$ different bucket assignments h.

3.2. Dealing with many buckets

In this section and Section 3.3, we provide an algorithm which given an initial bucket assignment h, decides whether there is a pushing embedding f of the input graph into the line with distortion at most d that agrees with h.

Our algorithm EXACT-DIST solves a slight modification of the problem. Input to this problem is a graph G, an integer d, a bucket assignment h, an interval $J = \{x, x+1, \ldots, y\}$ of integers and a function $g : V' \to \{0, \ldots, dn\}$ for some subset V' of $V(G)$. Let $\mathcal{B}_J = \bigcup_{j \in J} \mathcal{B}_j$ and $V_J = \bigcup_{j \in J} V_j$. The algorithm determines whether there is a partial pushing embedding $f : V_J \to \mathcal{B}_J$ with distortion at most d such that f agrees with h and $f(v) = g(v)$ for all vertices in $V' \cap V_J$. To solve the original problem we make a call to EXACT-DIST (G, d, h, J, g) where $J = \{0, \ldots, n\}$ and the domain V' of g is empty. Before commencing with the algorithm, we perform a "sanity check". That is, given h check whether it is even remotely feasible that f can exist. We verify that h satisfies the following properties.

- For every i, $|V_i| \leq d + 1$.
- Similarly, for every edge uv, $|h(u) - h(v)| \leq 1$.

Indeed, if some of these cases do not hold, there is no embedding f with distortion d that agrees with h and we can immediately answer "NO". At all later stages of the algorithm we assume that h satisfies these properties. An outline of the algorithm without these preliminary steps is given in Fig. 1. In Section 3.3 we will give an algorithm which implements Step 3 in time $2^n \cdot n^{O(b)}$ time, where $b = |J|$ is the number of buckets considered.

The idea behind the algorithm is as follows. When the number of buckets $|J|$ is large, our algorithm follows a divide-and-conquer approach and if the number of buckets is "small", that is roughly $n/\log^2 n$, we do dynamic programming. To deal with the large number of buckets we look for a "small balanced separator" to branch on. The first step of algorithm EXACT-DIST is based on the following lemma.

Lemma 2. *Let h be a bucket assignment and let $J = \{x, x+1, \ldots, y\}$ be an integer interval such that $\frac{n}{\log^2 n} < |J|$. Then there exists $j \in I = \{\frac{3x+y}{4} + 1, \ldots, \frac{x+3y}{4}\}$ such that $|V_j| \leq 2\log^2 n$.*

Proof. The proof follows from an averaging principle. For the sake of contradiction, let us assume that for every $j \in I$, $|V_j| > 2\log^2 n$. Then the total number of elements in the buckets V_j with $j \in I$ is at least

$$\sum_{j \in I} |V_j| > 2\log^2 n \cdot \frac{|J|}{2} > 2\log^2 n \cdot \frac{n}{2\log^2 n} = n.$$

But the sets V_j are disjoint, and thus the sum does not exceed $|V(G)| = n$, which is a contradiction. □

If $|J|$ is at least $n/\log^2 n$, the algorithm picks a bucket $\mathcal{B}_j$ and branches on all possible ways to lay out V_j in $\mathcal{B}_j$. After this the problem breaks up into two independent subproblems (G, d, h, J_1, g') and (G, d, h, J_2, g'); see Fig. 1. We argue that the two subproblems are indeed independent. Let f be a pushing partial embedding of V_J into $\mathcal{B}_J$ with distortion at most d such that f agrees with h and coincides with g. This means that f restricted to V_j is a pushing partial embedding of V_j into $\mathcal{B}_j$. We choose g_j to coincide with f on V_j and define $g'(v) = g_j(v)$ if $v \in V_j$ and $g'(v) = g(v)$ if v is in the domain of g, just as in Step 2 of algorithm Exact-Dist. If $J = \{x, \ldots, y\}$ then $J_1 = \{x, \ldots, j\}$ and $J_2 = \{j, \ldots, y\}$. Now f restricted to J_1 is a pushing partial embedding from V_{J_1} to $\mathcal{B}_{J_1}$ while f restricted to J_2 is a pushing partial embedding from V_{J_2} to $\mathcal{B}_{J_2}$.

In the other direction, let f_1 and f_2 be pushing partial embeddings from V_{J_1} to $\mathcal{B}_{J_1}$ and from V_{J_2} to $\mathcal{B}_{J_2}$ respectively, agreeing with h and coinciding with g'. Since $J = J_1 \cup J_2$ and $J_1 \cap J_2 = \{j\}$ we can choose f to be the partial embedding from V_J to $\mathcal{B}_J$ that coincides with both f_1 and f_2. Since both f_1 and f_2 are pushing partial embeddings, so is f. Since every edge with both endpoints in V_J has both endpoints either in V_{J_1} or in V_{J_2} and both f_1 and f_2 have distortion at most d, so does f.

Let $T(n, b)$ be the time required by algorithm Exact-Dist on an n-vertex graph G with $|J| = b$. Let $T^*(n)$ be the time required by algorithm Exact-Dist on an n-vertex graph G and with $|J| < n/\log^2 n$. An analysis of Steps 1 and 2 of algorithm Exact-Dist yields the following recurrence.

$$T(n, b) = \begin{cases} \binom{d+1}{2\log^2 n}(2\log^2 n)! \cdot 2T\left(n, \frac{3b}{4}\right) & \text{if } b > \frac{n}{\log^2 n} \\ T^*(n) & \text{otherwise.} \end{cases}$$

Thus

$$T(n, b) \leq \left[2\binom{d+1}{2\log^2 n}(2\log^2 n)!\right]^{O(\log b)} T^*(n).$$

Since $b \leq n$, we have that $T(n, b) \leq 2^{O(\log \frac{n}{n/\log^2 n})} \cdot T^*(n) = 2^{o(n)} \cdot T^*(n)$. In Section 3.3, we show how to implement the last step of algorithm Exact-Dist to run in time $2^n n^{O(b)}$ which is at most $2^n \cdot 2^{o(n)}$ since $b \leq n/\log^2 n$. This yields a $2^{n+o(n)}$ runtime bound for algorithm Exact-Dist and a $6^{n+o(n)}$ bound for deciding whether G can be embedded into the line with distortion at most d. In Section 3.4 we will show that the running time of our algorithm in fact is bounded by $5^{n+o(n)}$.

3.3. Dealing with few buckets

In this section we give an algorithm which given an initial bucket assignment h, a partial assignment g, and an integer interval $J = \{x, \ldots, y\}$ with $|J| = b < n/\log^2 n$ decides whether there is a pushing partial embedding $f : V_J \to \mathcal{B}_J$ with distortion at most d, agreeing with h and coinciding with partial assignment g. Our algorithm runs in time and space $2^n n^{O(b)}$.

The number of slots in J, that is positions in the line to where vertices can be mapped, is at most $b \cdot (d+1)$. Thus there could be many slots with no vertex mapped to them. We start our algorithm by guessing for every $j \in J$ the leftmost non-empty slot in each bucket $\mathcal{B}_j$ and a vertex from V_j to be placed there. Naturally, if the layout of a bucket $\mathcal{B}_j$ with $j \in J$ has already been determined by g our guesses must be consistent with this. For every $j \in J$, let t_j denote the vertex guessed to be placed leftmost in bucket j. Also let l_j denote the position guessed for t_j. After having made the guess we modify the problem at hand—we now look for a pushing partial embedding $f : V_J \to \mathcal{B}_J$ with distortion at most d, agreeing with h, coinciding with g such that for every bucket $\mathcal{B}_j$ with $j \in J$, the leftmost vertex mapped to $\mathcal{B}_j$ is t_j, which is mapped to l_j. The number of possible guesses is bounded by $(d+1)^b n^b$.

We choose the ordering $\pi_1, \pi_2, \ldots, \pi_{|\mathcal{B}_J|}$ of the entries of $\mathcal{B}_J$ such that for every $i < j$ we have that $\pi_i \bmod (d+1) \leq \pi_j \bmod (d+1)$ and such that if $\pi_i \bmod (d+1) = \pi_j \bmod (d+1)$ then $\frac{\pi_i}{d+1} < \frac{\pi_j}{d+1}$. In other words, the number pi_k is in the position $k \bmod (d+1)$ in the bucket with number $\lfloor k/(d+1) \rfloor$.

For example, if $J = 3, 4, 5$ and $d = 4$, then

$$\pi_1, \ldots, \pi_{15} = 15, 20, 25, 16, 21, 26, 17, 22, 27, 18, 23, 28, 19, 24, 29.$$

We call the ordering $\pi_1, \ldots, \pi_{|\mathcal{B}_J|}$ the *bucket order* of $\mathcal{B}_J$. Next we define the notion of a state.

Definition 1. A state ζ is a quadruple (P, Q, R, p), where $P \subseteq V_J$, $Q \subseteq P$ is a set of vertices containing at most one vertex from each V_j such that if $V_j \cap P \neq \emptyset$ then $V_j \cap Q \neq \emptyset$ and $t_j \in P$, $R \subseteq \mathcal{B}_J$, is a set of integers containing at most one integer from each bucket $\mathcal{B}_j$ and $p \leq |J|$ is a non-negative integer.

Let us observe that the number of states is at most $2^n \times n^{|J|} \times (d+1)^{|J|} \times |\mathcal{B}_j|$. If $Q \cap V_j \neq \emptyset$, then define q_j to be the vertex in $Q \cap V_j$. If $R \cap \mathcal{B}_j \neq \emptyset$ let r_j be the integer in $R \cap \mathcal{B}_j$. Next we define what it means for a state to be feasible.

Definition 2. A state is called *feasible* if there exists a partial embedding f assigning the vertices of P to the first p positions in the bucket order such that the following conditions hold.

1. For any edge uv with $u \in P$ and $v \in P$, $|f(u) - f(v)| \leq d$, f agrees with h and coincides with g.
2. If $V_j \cap P \neq \emptyset$, then $f(t_j) = l_j$ and $f(q_j) = r_j$. There is no vertex $v \in V_j \cap P$ such that $f(v) < l_j$ or $f(v) > r_j$.
3. For any bucket V_j with $j \in J$, if $x, y \in V_j, f(x) < f(y)$ and no vertex is mapped by f to the interval $\{f(x)+1, f(y)-1\}$, then $f(y) - f(x) = D_G(x, y)$.
4. If $j \in J$ and j is not the largest element of J, $V_j \subseteq P$ and $V_{j+1} \cap P \neq \emptyset$, then $f(l_{j+1}) - f(r_j) = D_G(l_{j+1}, r_j)$.

The idea is to go through the slots in J one by one in the bucket order and for each of them determine which vertex (if any) gets mapped by f to this slot. The number p denotes the position in the bucket order that we have reached. The set P corresponds to the set of vertices that have already been placed. For every $j \in J$, t_j and q_j denote the vertices placed leftmost and rightmost in $\mathcal{B}_j$ respectively. Also l_j and r_j denote the positions of t_j and q_j in $\mathcal{B}_j$. Now we define the notion of a state succeeding another state.

Definition 3. Let $\zeta_1 = (P_1, Q_1, R_1, p)$ and $\zeta_2 = (P_2, Q_2, R_2, p+1)$ be two states. We say that ζ_2 *succeeds* ζ_1 if the following holds.

- Either $P_2 = P_1$, or $P_2 = P_1 \cup \{v\}$.
- If $P_1 = P_2$, then $Q_1 = Q_2$ and $R_1 = R_2$.
- If $P_2 = P_1 \cup \{v\}$ and $v \in V_j$, then $j = \lfloor \frac{\pi_{p+1}}{d+1} \rfloor$ and
  1. If $v \in t_j$, then $l_j = \pi_{p+1}$. If $g(v)$ is defined then $g(v) = \pi_{p+1}$.
  2. $Q_2 = (Q_1 \setminus \{q_j\}) \cup \{v\}$ and $R_2 = (R_1 \setminus \{r_j\}) \cup \{\pi_{p+1}\}$.
  3. If $V_j \cap P_1 \neq \emptyset$ then $\pi_{p+1} - r_j = D_G(v, q_j)$.
  4. If j is not the largest element of J then $l_{j+1} - \pi_{p+1} \geq D(v, t_{j+1})$.
  5. If $j \in J$ and j is not the largest element of J, $V_j \subseteq P_2$ and $V_{j+1} \cap P_2 \neq \emptyset$ then $f(l_{j+1}) - f(v) = D_G(l_{j+1}, v)$.
  6. If j is not the smallest element of J then $N(v) \cap V_{j-1} \cap P_2 = \emptyset$.

We now proceed to prove an observation that will be helpful for the correctness proof.

Lemma 3. *Let $\zeta_1 = (P_1, Q_1, R_1, p)$ be a feasible state and $\zeta_2 = (P_2, Q_2, R_2, p+1)$ be a state that succeeds ζ_1. Then ζ_2 is feasible.*

Proof. Since $\zeta_1 = (P_1, Q_1, R_1, p)$ is feasible there is a partial embedding f satisfying points 1–4 in Definition 2. If $P_1 = P_2$ then f satisfies the points 1–4 for ζ_2 as well. If $P_2 \neq P_1$ then $P_2 \setminus P_1$ contains a single vertex v. Let f' be a partial embedding assigning the vertices of P_2 to the first $p+1$ positions in the bucket order such that f' and f coincide and $f'(v) = \pi_{p+1}$. By point 1 of the definition of succession f' agrees with h and coincides with g. Since v has no neighbour in $P_1 \cap V_{j-1}$ it follows that for any edge uw with $u \in P_2$ and $w \in P_2$, $|f(u) - f(w)| \leq d$. Also, f' satisfies point 2 of Definition 2 because π_{p+1} is the rightmost position in $P_2 \cap \mathcal{B}_j$. Furthermore, f' satisfies point 3 of Definition 2 by point 3 of Definition 3. Finally f' satisfies point 4 of Definition 2 by point 5 of Definition 3. □

Now we are ready to prove the main lemma of the section which allows us to obtain the desired result.

Lemma 4. *There is a pushing partial embedding $f : V_J \to \mathcal{B}_J$ with distortion at most d such that f agrees with h, coincides with g and such that for every $j \in J$, $f(t_j) = l_j$ and no other vertex in V_j is mapped before t_j by f if and only if there exists sequence of states $\zeta_1, \zeta_1, \ldots, \zeta_{|\mathcal{B}_J|}$ such that*

(a) $\zeta_1 = (\emptyset, \emptyset, \emptyset, 0)$;
(b) ζ_{i+1} *succeeds* ζ_i *for all* $i \in \{1, \ldots, |\mathcal{B}_J| - 1\}$; *and*
(c) $\zeta_{|\mathcal{B}_J|} = (V_J, X, Y, |\mathcal{B}_J|)$ *for some vertex sets* X *and* Y.

Proof. Let $f : V_J \to \mathcal{B}_J$ be a pushing partial embedding with distortion at most d such that f agrees with h, coincides with g and such that for every $j \in J$, $f(t_j) = l_j$ and no other vertex in V_j is mapped before t_j by f. With the help of f we define the sequence of feasible states as follows. For every $p \leq |\mathcal{B}_J|$, P is the set of vertices f maps to $\pi_0, \ldots, \pi_p$, Q is the set of vertices in P such that for every j such that $P \cap V_j \neq \emptyset$, Q contains exactly one vertex q_j, f maps all vertices in $P \cap V_j$ to the left of q_j. Finally R is the set of positions that f maps the vertices of Q. The construction of the sequence of states implies that $\zeta_1 = (\emptyset, \emptyset, \emptyset, 0)$, ζ_{i+1} succeeds ζ_i for all $i \in \{1, \ldots, |\mathcal{B}_J| - 1\}$ and that $\zeta_{|\mathcal{B}_J|} = (V_J, X, Y, |\mathcal{B}_J|)$.

For the reverse direction suppose that we have a sequence of feasible states $\zeta_1, \zeta_1, \ldots, \zeta_{|\mathcal{B}_J|}$ such that $\zeta_1 = (\emptyset, \emptyset, \emptyset, 0)$; (b) ζ_{i+1} succeeds ζ_i for all $i \in \{1, \ldots, |\mathcal{B}_J| - 1\}$; and (c) $\zeta_{|\mathcal{B}_J|} = (V_J, X, Y, |\mathcal{B}_J|)$. Since $\zeta_1 = (\emptyset, \emptyset, \emptyset, 0)$ is feasible, we have that by Lemma 3, $\zeta_{|\mathcal{B}_J|} = (V_J, X, Y, |\mathcal{B}_J|)$ is feasible as well. The definition of feasibility guarantees the existence of the desired f, concluding the proof. □

Finally, we ready to proceed with the lemma used for the analysis of Step 3.

Lemma 5. *There is an algorithm that for given G, d, h, J, g and T decides whether there is a pushing partial embedding $f : V_J \to \mathcal{B}_J$ with distortion at most d such that f agrees with h, coincides with g and such that for every $j \in J$, $f(t_j) = l_j$ and no other vertex in V_j is mapped before t_j by f in time and space $2^n \cdot n^{O(|J|)}$.*

Proof. As we observed already, the number of states is at most $2^n \times n^{|J|} \times (d+1)^{|J|} \times |\mathcal{B}_j| \leq 2^n \cdot n^{O(|J|)}$. The algorithm decides the existence of f by applying Lemma 4. The algorithm starts in the state $(\emptyset, \emptyset, \emptyset, 0)$ and does breadth first search on the graph where vertices are the states and there is a directed edge from a state ζ_i to a state ζ_j if ζ_j succeeds ζ_i. We do not keep this graph explicitly and rather generate the vertices of this graph as and when required in our breadth first search. Whenever we are at state ζ we can find all possible successor states in polynomial time. By Lemma 4 there is a required embedding f if and only if there is a path from $(\emptyset, \emptyset, \emptyset, 0)$ to $(V_J, X, Y, |\mathcal{B}_J|)$. Our algorithm needs $2^n \cdot n^{O(|J|)}$ space to keep track of the set of states visited by the breadth first search algorithm. Since the number of states is bounded by $2^n \cdot n^{O(|J|)}$ and the number of successors of a state is at most $d+2$ the number of vertices and edges in the state graph is upper bounded by $2^n \cdot n^{O(|J|)}$. Hence the algorithm takes $2^n \cdot n^{O(|J|)}$ time and space. □

Observe that applying Lemma 5 together with the analysis presented for Algorithm Exact-Dist over the previous section yields a running time bound of $6^{n+o(n)}$. In fact, our algorithm runs in time $5^{n+o(n)}$. The next section is devoted to proving this.

3.4. A refined analysis

In this section we prove that the total number of states ever produced by our algorithm is $5^{n+o(n)}$. Since the running time of the algorithm is proportional to the number of states we generate up to a subexponential factor; this implies that algorithm Exact-Dist runs in time $5^{n+o(n)}$.

Lemma 6. *The algorithm described in the previous sections runs in time $5^{n+o(n)}$.*

Proof. Let a *super-state* be a two-tuple (h, ζ) where h is a bucket assignment and ζ is a state generated by algorithm Exact-Dist. We say that a super-state is visited by the algorithm to decide whether there is an embedding f of G into the line with distortion at most d if ζ is generated at a call to Exact-Dist with h as the required bucket assignment. The total running time of the algorithm is directly proportional to the total number of times each super-state is visited. First we argue that each super-state is visited at most $2^{o(n)}$ times. For a fixed $h, J = \{x, \ldots, y\}, T, l_x \ldots l_y$ and g the state (h, ζ) is visited at most one. However, the number of possible J's is $O(n^2)$, $|T| \leq \frac{n}{\log^2 n}$, $y - x \leq |T|$ so the number of possible sets T and $|J|$-tuples $l_x, \ldots, l_y$ is $2^{o(n)}$. Finally, in any recursive call the domain of g is at most $\log^2 n \cdot \log \frac{n}{n/\log^2 n} = 2\log^2 n \log\log n$. Hence the number of possible g's is $2^{o(n)}$. Thus each super-state is visited at most $2^{o(n)}$ times.

For every fixed h and $P \subseteq V(G)$ there are at most $2^{o(n)}$ triplets (Q, R, p) such that the super-state $(h, (P, Q, R, p))$ is visited. This is true because $|Q| \leq n/\log^2 n$, $|R| \leq n/\log^2 n$ and p is an integer in $\{1, \ldots, dn\}$.

Finally, we need to argue that there are at most $n^2 \cdot 5^n$ pairs (h, P) such that there is a triplet (Q, R, p) such that the super-state $(h, (P, Q, R, p))$ is visited. Let T be the spanning tree of G rooted at r_T that we used to list bucket assignments of G. We prove that for a fixed integer interval $J \subseteq \{1, \ldots, n\}$ with $|J| \leq n/\log^2 n$ the number of pairs (h, P) such that there is a triplet (Q, R, p) such that the super-state $(h, (P, Q, R, p))$ is visited by the algorithm during a call to algorithm Exact-Dist with J as parameter is at most 5^n. Every such pair corresponds to a labelling of the tree T. The vertices of T are labelled from the set $\{0, 1\}$ with a vertex v labelled 1 if $v \in P$. The edges of T are labelled from the set $\{-1, 0, 1\}$ such that for every $uv \in E(T)$ where u is a parent of v, the edge uv is labelled $h(v) - h(u)$. For a subtree T' of T rooted at r_T we say that a labelling L of T' is *good* if there is a super-state visited by the algorithm whose labelling restricted to T' is exactly L. We prove that if T' has n' vertices then the number of good labellings of T' is at most $5^{n'}$ by induction on n'. If $n' = 1$ this follows trivially. Suppose now that the assertion holds for some n' and consider a subtree T' on $n'+1$ vertices. Let l be a leaf of T'. Notice that any good labelling of T' restricted to $T' \setminus l$ is a good labelling of $T' \setminus l$. By the induction hypothesis there are at most $5^{n'}$ good labellings of $T' \setminus l$. We prove that there are at most 5 ways to extend a good labelling L of $T' \setminus l$ to a good labelling of T'.

Let l' be the parent of l in T and let $P_{l'}$ be the path from r_T to l' in T'. Let $z = \sum_{uv \in E(P_{l'})} L(uv)$. If $z - 1 \notin J, z \notin J$ or $z + 1 \notin J$ we prove that there are only 5 ways to extend L to a good labelling of T'. In order to extend L we need to specify $L(ll')$ and $L(l)$. Notice that if $z + L(ll') \notin J$ then $L(l)$ must be 0 in any good labelling. Thus in this case there are at most 5 ways to extend L. Now, consider the case that $\{z-1, z, z+1\} \subseteq J$ and $L(l') = 0$. Then if $L(ll') = -1$ then $L(l)$ cannot be 1 in a good labelling. Finally, consider the case that $\{z-1, z, z+1\} \subseteq J$ and $L(l') = 1$. Then if $L(ll') = 1$ then $L(l)$ cannot be 0 in a good labelling. In both these cases there are at most 5 ways to extend L, concluding the proof. □

We conclude with the following theorem.

Theorem 1. *There is an algorithm that given a graph G on n vertices constructs a non-contracting embedding of the shortest path metric generated by G into the line with minimum distortion in time $5^{n+o(n)}$ and space $2^{n+o(n)}$.*

4. Concluding remarks and open problems

In this paper we have provided the first single vertex exponential time algorithm for computing a minimum distortion embedding of a graph metric into the line. This result gives rise to many challenging questions.

How fast is it possible to compute a minimum distortion embedding of a graph G into the metric of another graph H? Is there a $2^{O(|V(G)|)}$ time algorithm for this problem, or can one show that this is impossible up to some complexity theoretic assumption? How does the problem behave if the host graph H is a tree? Even when H is a binary tree, this does not seem to be an easy problem. At a first glance it would seem that our algorithm should be directly extendable to find a minimum distortion embedding of a graph G into a given cycle C. However, this does not look to be easy and we leave it as an open problem whether finding a minimum distortion embedding of a graph G into a given cycle C can be done in $2^{O(|V(G)|)}$ time.

Another interesting question is on the embedding into line of weighted graphs. Even the first step of our algorithm, guessing the bags, does not work for the weighted case. If the maximum edge weight of a graph is W, then our algorithm can be adapted to run in time $O(W^{O(n)})$. We leave the possibility (or non-possibility) of embedding a weighted graph into line in time $2^{O(n)}$ as another interesting question.

We believe that the world of embeddings provides a lot of challenges to the area of exact exponential time algorithms [9] and is worth to be explored. We hope that our result will lead to further investigation of the combinatorially challenging field of embeddings within the framework of exact exponential time algorithms.

Very recently, Cygan and Pilipczuk in [6] succeed to improve the running time of our algorithm to $O(4.383^n)$ thus matching the running time of their algorithm for bandwidth from [5].

References

[1] M. Bădoiu, J. Chuzhoy, P. Indyk, A. Sidiropoulos, Low-distortion embeddings of general metrics into the line, in: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC, ACM, 2005, pp. 225–233.
[2] M. Bădoiu, K. Dhamdhere, A. Gupta, Y. Rabinovich, H. Räcke, R. Ravi, A. Sidiropoulos, Approximation algorithms for low-distortion embeddings into low-dimensional spaces, in: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, 2005, pp. 119–128.
[3] M. Badoiu, P. Indyk, A. Sidiropoulos, Approximation algorithms for embedding general metrics into trees, in: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, ACM, SIAM, 2007, pp. 512–521.
[4] M. Cygan, M. Pilipczuk, Faster exact bandwidth, in: Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2008, in: Lecture Notes in Computer Science, vol. 5344, 2008, pp. 101–109.
[5] M. Cygan, M. Pilipczuk, Exact and approximate bandwidth, Theoret. Comput. Sci. 411 (2010) 3701–3713.
[6] M. Cygan, M. Pilipczuk, Bandwidth and Distortion Revisited, arXiv:1004.5012v1.
[7] U. Feige, Coping with the NP-hardness of the graph bandwidth problem, in: Proceedings of the 7th Scandinavian Workshop on Algorithm Theory, SWAT, in: LNCS, vol. 1851, Springer, Berlin, 2000, pp. 10–19.
[8] M. Fellows, F.V. Fomin, D. Lokshtanov, E. Losievskaja, F. Rosamond, S. Saurabh, Distortion is fixed parameter tractable, in: Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP, in: LNCS, vol. 5555, Springer, Berlin, 2009, pp. 463–474.
[9] F.V. Fomin, D. Kratsch, Exact Exponential Algorithms, in: Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2010.
[10] F.V. Fomin, D. Lokshtanov, S. Saurabh, An exact algorithm for minimum distortion embedding, in: Proceedings of the 35th Workshop on Graph Theoretic Concepts in Computer Science, WG, in: LNCS, vol. 5911, Springer, Berlin, 2009, pp. 112–121.
[11] A. Gupta, I. Newman, Y. Rabinovich, A. Sinclair, Cuts, trees and l_1-embeddings of graphs, Combinatorica 24 (2004) 233–269.
[12] P. Indyk, Algorithmic applications of low-distortion geometric embeddings, in: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS, IEEE, 2001, pp. 10–33.
[13] C. Kenyon, Y. Rabani, A. Sinclair, Low distortion maps between point sets, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC, ACM, 2004, pp. 272–280.
[14] N. Linial, Finite metric-spaces—combinatorics, geometry and algorithms, in: Proceedings of the International Congress of Mathematicians, Vol. III, Higher Ed. Press, Beijing, 2002, pp. 573–586.