Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Geometry 30 (2005) 59–77
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

# Optimal spanners for axis-aligned rectangles ☆

Tetsuo Asano [a], Mark de Berg [b], Otfried Cheong [c,*], Hazel Everett [d], Herman Haverkort [e], Naoki Katoh [f], Alexander Wolff [g]

[a] *JAIST, Japan*
[b] *Technische Universiteit Eindhoven, the Netherlands*
[c] *KAIST, South Korea*
[d] *LORIA, Nancy, France*
[e] *BRICS, Department of Computer Science, University of Aarhus, Denmark*
[f] *Kyoto University, Japan*
[g] *Fakultät für Informatik, Universität Karlsruhe, Germany*

Received 23 December 2003; received in revised form 6 September 2004; accepted 8 September 2004

Available online 7 October 2004

Communicated by S. Suri

**Abstract**

The dilation of a geometric graph is the maximum, over all pairs of points in the graph, of the ratio of the Euclidean length of the shortest path between them in the graph and their Euclidean distance. We consider a generalized version of this notion, where the nodes of the graph are not points but axis-parallel rectangles in the plane. The arcs in the graph are horizontal or vertical segments connecting a pair of rectangles, and the distance measure we use is the $L_1$-distance. The dilation of a pair of points is then defined as the length of the shortest rectilinear path between them that stays within the union of the rectangles and the connecting segments, divided by their $L_1$-distance. The dilation of the graph is the maximum dilation over all pairs of points in the union of the rectangles.

We study the following problem: given $n$ non-intersecting rectangles and a graph describing which pairs of rectangles are to be connected, we wish to place the connecting segments such that the dilation is minimized. We obtain four results on this problem: (i) for arbitrary graphs, the problem is NP-hard; (ii) for trees, we can solve the problem by linear programming on $O(n^2)$ variables and constraints; (iii) for paths, we can solve the problem in

☆ Part of this research was done during the First Utrecht-Carleton Workshop on Computational Geometry. H.H. acknowledges support by the Netherlands' Organization for Scientific Research (NWO).

* Corresponding author.
*E-mail addresses:* t-asano@jaist.ac.jp (T. Asano), m.t.d.berg@tue.nl (M. de Berg), otfried@tclab.kaist.ac.kr (O. Cheong), everett@loria.fr (H. Everett), herman@haverkort.net (H. Haverkort), naoki@archi.kyoto-u.ac.jp (N. Katoh), awolff@ira.uka.de (A. Wolff).

0925-7721/$ – see front matter 
doi:10.1016/j.comgeo.2004.09.001

time $O(n^3 \log n)$; (iv) for rectangles sorted vertically along a path, the problem can be solved in $O(n^2)$ time, and a $(1+\varepsilon)$-approximation can be computed in linear time.


*Keywords:* Geometric spanners; Dilation optimization; Isothetic rectangles; Manhattan distance

## 1. Introduction

Geometric networks arise frequently in our everyday life: road networks, telephone networks, and computer networks are all examples of geometric networks that we use daily. They also play a role in disciplines such as VLSI design and motion planning. Almost invariably, the purpose of the network is to provide a connection between the nodes in the network. Often it is desirable that the connection through the network between any pair of nodes be relatively short. From this viewpoint, one would ideally have a direct connection between any pair of nodes. This is usually infeasible due to the costs involved, so one has to compromise between the quality and the cost of the connections.

For two given nodes in a graph, the ratio of their distance in the graph and their 'direct' distance is called the *dilation* or *stretch factor* for that pair of nodes, and the dilation of a graph is the maximum dilation over all pairs of nodes. For geometric networks, this is more precisely defined as follows. Let $S$ be a set of $n$ points (in the plane, say), and let $\mathcal{G}$ be a graph with node set $S$. Now the dilation for a pair of points $p, q$ is defined as the ratio of the length of the shortest path in $\mathcal{G}$ between $p$ and $q$, and the length of the segment $pq$. (The length of a path is the sum of the lengths of its edges.) Again, the dilation of $\mathcal{G}$ is the maximum dilation over all pairs of points in $S$. A graph with dilation $t$ is called a *$t$-spanner*. Ideal networks are $t$-spanners for small $t$ with small cost.

Spanners were introduced by Peleg and Schäffer [8] in the context of distributed computing, and by Chew [3] in the context of computational geometry. They have attracted much attention since—see for instance the survey by Eppstein [4]. The cost of spanners can be measured according to various criteria. For example, it is sometimes defined as the number of edges (here the goal is to find a spanner with $O(n)$ edges), or as the total weight of the edges (here the goal is to find a spanner whose total weight is a constant times the weight of a minimum spanning tree). Additional properties, such as bounding the maximum degree or the diameter, have been considered as well.

We generalize the notion of spanners to geometric networks whose nodes are rectangles rather than points. Let $S$ be a set of $n$ non-intersecting, axis-parallel rectangles and let $E$ be a set of axis-parallel segments connecting pairs of rectangles. For any two points $p, q$ in the union of the rectangles, the dilation is now the ratio of the length of the shortest rectilinear path in the network between $p$ and $q$ and their $L_1$-distance. Here a path in the network is a path that stays within the union of the rectangles and the connecting segments. The dilation of the network is the maximum dilation over all pairs $p, q$. Again, our aim is to construct a network whose dilation is small. To illustrate the concept, imagine one is given a number of rectangular buildings, which have to be connected by footbridges. It is quite frustrating if, to walk to a room opposite ones own room in an adjacent building, one has to walk all the way to the end of a long corridor, then along the footbridge, and then back again along the corridor in the other building. Hence, one would usually place the footbridge in the middle between buildings. Following this analogy, we will call the rectangles in the input *buildings* from now on, and the connecting segments *bridges*. We call the underlying graph of the network the *bridge graph*.

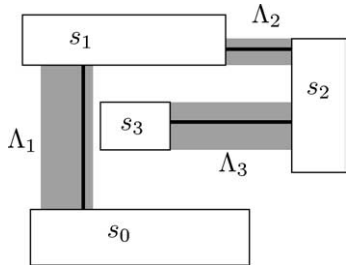


Fig. 1. A bridge graph and a bridge configuration.

The generalization we study introduces one important additional difficulty in the construction of a spanner: for points one only has to decide *which* edges to choose in the spanner, but for buildings, one also has to decide *where* to place the bridge between a given pair of buildings. It is the latter problem we focus on in this paper: we assume the topology of the network (the bridge graph) is given, and our only task is to place the bridges so as to minimize the dilation.

Formally, our problem can be stated as follows: we are given a set $S$ of axis-parallel disjoint rectangles (buildings) in the plane, a graph $\mathcal{G}$ with node set $S$, and for each arc $e$ of $\mathcal{G}$ a *bridge region* $\Lambda_e$, an axis-aligned rectangle connecting the two buildings. Buildings may degenerate to segments or points. The bridge graph $\mathcal{G}$ must only have arcs between buildings that can be connected by a horizontal or vertical segment, and may not have multiple edges or loops. The bridge regions must be disjoint from each other and the buildings. Our goal is to find a set of horizontal or vertical bridges lying in the bridge regions that has minimum dilation.

Fig. 1 shows a bridge graph (the bridge regions are shaded) and a set of possible bridges. Note that the bridge regions $\Lambda_2$ and $\Lambda_3$ simply allow any bridge between the two buildings, but bridge region $\Lambda_1$ has been chosen so as to avoid intersecting $s_3$ or the bridge between $s_2$ and $s_3$.

Our results are as follows.

- In general, the problem is NP-hard.
- If the bridge graph is a tree, then the problem can be solved by a linear program with $O(n^2)$ variables and constraints.
- If the bridge graph is a path, then the problem can be solved in $O(n^3 \log n)$ time.
- If the bridge graph is a path and the buildings are sorted vertically along this path, the problem can be solved in time $O(n^2)$. A $(1+\varepsilon)$-approximation can be computed in linear time.

## 2. The bridge graph is arbitrary

In this section we show that the bridge-placement problem is NP-hard if the bridge graph is allowed to be arbitrary. We prove this by a reduction from PARTITION. The input to PARTITION is a set $B$ of $n$ positive integers, and the task is to decide whether $B$ can be partitioned into two subsets of equal sum. PARTITION is NP-hard [5, Problem SP12].

**Theorem 1.** *It is NP-hard to decide whether the bridges in a given bridge graph on $n$ rectangular buildings can be placed such that the dilation is at most* 2.

Let $B := \{\beta_0, \ldots, \beta_{n-1}\}$ be an instance of PARTITION. For $0 \leqslant i < n$, we define

$$\alpha_i := \beta_i \Big/ \left(2 \sum_{0 \leqslant j < n} \beta_j\right).$$

Note that $\sum_{0 \leqslant j < n} \alpha_j = 1/2$, and that $B$ can be partitioned equally if and only if $\{\alpha_0, \ldots, \alpha_{n-1}\}$ can be partitioned into two subsets of sum $1/4$. We create a bridge graph $\mathcal{G}(B)$ with $8n+2$ buildings, as follows:

- for each $0 \leqslant i < n$, we have two point-shaped buildings, namely $P_i := (4i, 0)$ and $Q_i := (4i + 2 - 2\alpha_i, 0)$;
- for each $0 \leqslant i < n$, we have four segment-shaped buildings, namely $R_i := \{4i\} \times [1 - \alpha_i, 1]$ and $S_i := \{4i + 2 - 2\alpha_i\} \times [1 - \alpha_i, 1]$, and their mirrored images $R'_i := \{4i\} \times [-1, \alpha_i - 1]$ and $S'_i := \{4i + 2 - 2\alpha_i\} \times [-1, \alpha_i - 1]$;
- for each $0 < i < n$, we have two point-shaped buildings, namely $T_i := (4i - 1, 1)$ and $T'_i := (4i - 1, -1)$;
- we have two more point-shaped buildings $S_{-1} := (0, 2n + 3/4)$ and $S'_{-1} := (0, -2n - 3/4)$, and two more segment buildings $R_n := \{4n\} \times [1, 2n + 3/4]$ and $R'_n := \{4n\} \times [-2n - 3/4, -1]$.

The arcs in $\mathcal{G}(B)$ are as follows:

- for each $0 \leqslant i < n$, we have arcs $(P_i, R_i)$, $(P_i, R'_i)$, $(Q_i, S_i)$, $(Q_i, S'_i)$, $(R_i, S_i)$ and $(R'_i, S'_i)$;
- for each $0 < i < n$, we have arcs $(S_{i-1}, T_i)$, $(T_i, R_i)$, $(S'_{i-1}, T'_i)$, $(T'_i, R'_i)$ and $(T_i, T'_i)$;
- we have arcs $(S_{-1}, R_0)$, $(S'_{-1}, R'_0)$, $(S_{n-1}, R_n)$, $(S'_{n-1}, R'_n)$, $(R_n, R'_n)$.

Observe that $(R_i, S_i)$ and $(R'_i, S'_i)$ are the only bridges that can still be moved; all other bridges are fixed by the geometry. The construction is illustrated in Fig. 2; the bridges to be placed are indicated as gray segments or rectangles. For the sake of clarity, we chose different scales on the $x$- and $y$-axis.

The reduction can clearly be done in polynomial time. The following lemma now implies the theorem.

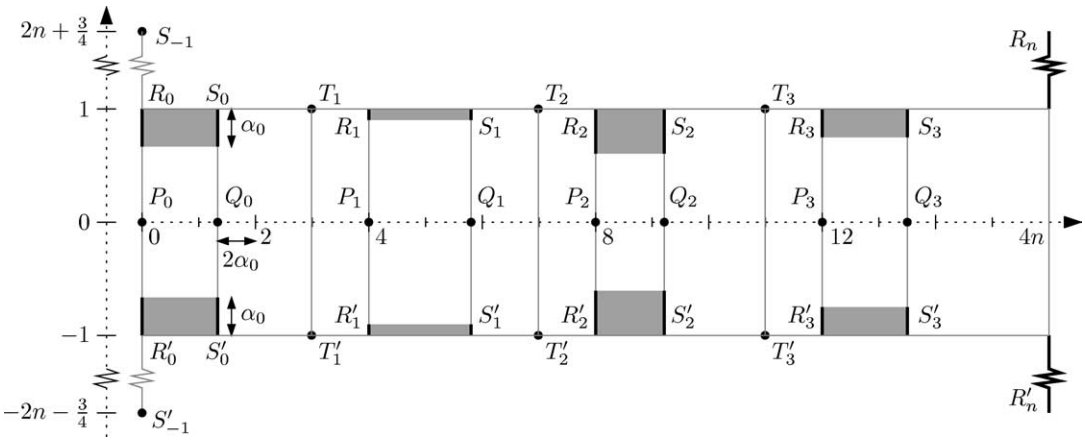


Fig. 2. An instance of the bridge decision problem.

**Lemma 2.** *The set $B$ can be partitioned into two subsets of equal sum if and only if the bridges in $\mathcal{G}(B)$ can be placed such that the dilation is at most* 2.

**Proof.** "If" Suppose we can place the bridges in $\mathcal{G}(B)$ such that the dilation is at most 2. Then the dilation must be at most 2 for any pair $(P_i, Q_i)$, which implies that either the bridge $(R_i, S_i)$ must be placed in its bottommost position or $(R'_i, S'_i)$ must be placed in its topmost position. Let $I$ denote the set of indices for which the former holds, and $I'$ the set of indices for which the latter holds.

Now consider $S_{-1}$ and the top vertex of $R_n$. The $L_1$-distance between them is $4n$. The shortest path between them in $\mathcal{G}(B)$ cannot visit any point on the $x$-axis, because the length of such a path would be at least $4n + 2(2n + 3/4)$ so its dilation would be larger than 2. Hence, the shortest path must visit $R_0, S_0, T_1, \ldots, R_{n-1}, S_{n-1}$ in order from left to right. Any $i \in I$ induces an extra vertical distance $2\alpha_i$. Adding the vertical distance between $S_{-1}$ and $R_0$ and along $R_n$, and the horizontal distance traversed, we get a total length of at least $\sum_{i\in I}(2\alpha_i) + 2(2n - 1/4) + 4n$. Hence, $\sum_{i\in I}\alpha_i \leqslant 1/4$. A similar argument for $S'_{-1}$ and the bottom vertex of $R'_n$ shows that $\sum_{i\in I'}\alpha_i \leqslant 1/4$. It follows that $I$ and $I'$ induce an equal partition of $B$.

"Only if" Suppose there is an equal partition of $B$. Then there are disjoint sets of indices $I$ and $I'$ with $I \cup I' = \{0, \ldots, n-1\}$ such that $\sum_{i\in I}\alpha_i = \sum_{i\in I'}\alpha_i = 1/4$. For $i \in I$ place the bridges $(R_i, S_i)$ and $(R'_i, S'_i)$ in their bottommost position, and for $i \in I'$ place the bridges $(R_i, S_i)$ and $(R'_i, S'_i)$ in their topmost position.

Consider two points $p, q$, each lying on a building, with $p_x \leqslant q_x$. If $p_x = q_x$, then $q$ can be reached without any detour. Otherwise, we distinguish two cases.

- The first case is that $p$ or $q$ (or both) have non-zero $y$-coordinate. Assume without loss of generality that $p_y > 0$ or that $p_y = 0$ and $q_y > 0$. Consider the path that goes up or down from $p$ until reaching $y = 1$, then goes to the right while staying above the $x$-axis until the $x$-coordinate of $q$ is reached, and then goes straight down or up to $q$.
  If $p = S_{-1}$ and $q \in R_n$, then the length of the path is bounded by
  $$4n + \sum_{i\in I}(2\alpha_i) + 2(2n - 1/4) = 8n.$$
  Since $|p_x - q_x| = 4n$, the dilation is at most 2.
  If $p \neq S_{-1}$ or $q \notin R_n$, the length of the path is bounded by
  $$|p_x - q_x| + 2\sum_{i\in I}\alpha_i + |1 - p_y| + |1 - q_y| = |p_x - q_x| + 1/2 + |1 - p_y| + |1 - q_y|.$$
  If $p_y$ and $q_y$ are not both $\leqslant 1$, then $|1 - p_y| + |1 - q_y| = |p_y - q_y|$, otherwise, $|1 - p_y| + |1 - q_y| = |p_y - q_y| + 2|1 - \max(p_y, q_y)| \leqslant |p_y - q_y| + 1/2$. In both cases the length of the path is at most $|p_x - q_x| + |p_y - q_y| + 1$, and from $|p_x - q_x| \geqslant 1$ it follows that the dilation is at most 2.
- The second case is that $p_y = q_y = 0$. Now the vertical distance traversed by the shortest path is at most $2 + \sum_{i\in I}(2\alpha_i) = 5/2$. Hence, if $|p_x - q_x| \geqslant 5/2$, the dilation is at most 2. But $|p_x - q_x| < 5/2$ implies that $p = P_i$ and $q = Q_i$ for some $0 \leqslant i < n$ or that $p = Q_i$ and $q = P_{i+1}$ for some $0 \leqslant i < n$. In the former case the dilation is 2 because either $(R_i, S_i)$ is bottommost or $(R'_i, S'_i)$ is topmost. In the latter case the dilation is less than 2 because the vertical distance traversed is exactly 2 and $|p_x - q_x| > 2$. □

## 3. The bridge graph is a tree

In this section we will show that the bridge-placement problem can be solved by a linear program if the bridge graph is a tree. We start by introducing some terminology and notation, and by proving some basic lemmas.

As before, we denote the bridge graph by $\mathcal{G}$. Any set of bridges realizing $\mathcal{G}$ will be called a *configuration*.

Given a configuration $B$ and two points $p$ and $q$ in the union of all buildings, we use $\pi(p, q, B)$ to denote the family of rectilinear shortest paths from $p$ to $q$ within the configuration (that is, paths whose links lie inside buildings or on bridges). The paths of this family are essentially the same, they differ only in how they connect two points inside the same building, and so we will simply speak about *the unique path* $\pi(p, q, B)$. The *dilation* of the path $\pi = \pi(p, q, B)$ is $\mathrm{dil}(\pi) := |\pi|/\|pq\|$, where $|\pi|$ is the total length of $\pi$ and $\|pq\|$ is the $L_1$-distance of $p$ and $q$. Fig. 3 shows a configuration and an example path.

The *dilation* $\mathrm{dil}(B)$ *of a configuration* $B$ is defined as the maximum dilation of any path with respect to $B$. Our aim is to find a configuration of minimum dilation. We first characterize pairs of points that are responsible for the dilation of a given configuration.

**Lemma 3.** *Let $\sigma$ be the dilation of a configuration $B$ whose underlying graph is a tree. Then there are points $p$ and $q$ with $\mathrm{dil}(\pi(p, q, B)) = \sigma$ such that the closed bounding box of $p$ and $q$ does not contain any point of a building other than $p$ and $q$, and at least one of the points $p$ and $q$ is a building corner.*

**Proof.** Among all pairs of points $(p, q)$ that have maximum dilation with respect to $B$, consider the subset of pairs where $\|pq\|$ is minimum. Choose a pair $(p, q)$ from this subset where $p$ is lexicographically smallest. Let $\beta$ be the closed bounding box of $p$ and $q$, and assume there is a point $r \in \beta$ distinct from $p$ and $q$ that belongs to a building. By our choice of $(p, q)$, we have $|\pi(p, r, B)| < \sigma\|pr\|$ and $|\pi(r, q, B)| < \sigma\|rq\|$. Since $r \in \beta$ we have $\|pq\| = \|pr\| + \|rq\|$. Combining with the triangle inequality we obtain

$$\big|\pi(p, q, B)\big| \leqslant \big|\pi(p, r, B)\big| + \big|\pi(r, q, B)\big| < \sigma\|pr\| + \sigma\|rq\| = \sigma\|pq\| = \big|\pi(p, q, B)\big|,$$

a contradiction, so no such point $r \in \beta$ exists.

It immediately follows that $p$ and $q$ are on the boundary of their buildings. It remains to prove that at least one of them is a building corner. Assume to the contrary that both are on the interior of a building edge. Then either $p$ and $q$ have the same $x$-coordinate and lie on the top and bottom edge of their buildings, or they have the same $y$-coordinate and lie on the left and right edge of their buildings. We discuss the first case, the second case is analogous. Clearly, moving both $p$ and $q$ the same distance to the left or right does not change $\|pq\|$. But what about $|\pi(p, q, B)|$? Let $\ell$ be the vertical line through $p$ and $q$, and let $e$ and $f$ be the points where $\pi(p, q, B)$ leaves the buildings containing $p$ and $q$, respectively. If $e$ and $f$ lie on opposite sides of $\ell$ as in Fig. 4, we can move $p$ and $q$ slightly to the left without changing $\mathrm{dil}(\pi(p, q, B))$, a contradiction to the assumption that $p$ is lexicographically smallest. It follows that $e$ and $f$ lie on the same side of $\ell$ (including $\ell$ itself), and so $|\pi(p, q, B)|$ increases if we move $p$ and $q$ into the opposite direction, a contradiction to the assumption that $\mathrm{dil}(\pi(p, q, B))$ is maximal. □

A point pair $(p, q)$ as in the lemma—its bounding box contains no other point of any building and at least one of $p$ and $q$ is a building corner—will be called a *visible pair*—see Fig. 5 for examples. We

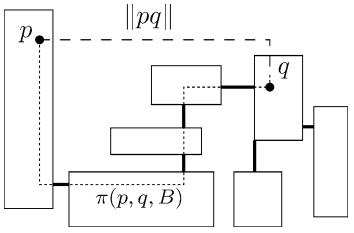


Fig. 3.

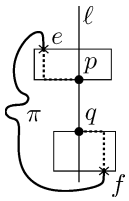


Fig. 4.

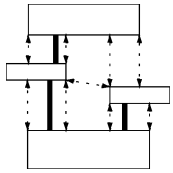

Fig. 5.

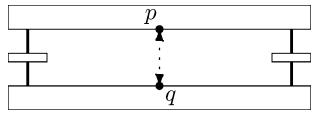


Fig. 6.

denote the set of all visible pairs by $\mathcal{V}$. Note that the second statement of the lemma does not hold if there are cycles in the bridge graph—the maximum dilation may occur between two points in the interior of building edges, as in Fig. 6.

**Lemma 4.** *For any set of $n$ buildings, there are at most $O(n^2)$ visible pairs and they involve at most $12n$ points. These points can be computed in $O(n \log n)$ time.*

**Proof.** Clearly there are at most $O(n^2)$ visible pairs where both points are building corners. These pairs involve only the at most $4n$ building corners. Consider a visible pair $(p, q)$ where only $p$ is a building corner. Then $q$ can be found by shooting a vertical or horizontal ray from $p$ until it hits another building. It follows that for each building corner $p$ there are at most two choices for $q$, so there are at most $8n$ such visible pairs, and at most $8n$ candidates for non-corner points that can be involved in a visible pair. They can be found in $O(n \log n)$ time by computing a vertical and a horizontal decomposition of the set of buildings [2]. □

Lemmas 3 and 4 allow us to compute the dilation of a given configuration efficiently. The quadratic bound is tight: even if the bridge graph is a path, there can be $\Omega(n^2)$ visible pairs.

Given a bridge graph $\mathcal{G}$, our goal is to minimize

$$\max_{(p,q)\in\mathcal{V}} \operatorname{dil}\bigl(\pi(p,q,B)\bigr)$$

over all configurations $B$ realizing $\mathcal{G}$. We will now reformulate this problem as a linear program.

**Theorem 5.** *If the bridge graph $\mathcal{G}$ is a tree, finding a minimum-dilation bridge placement can be reduced in $O(n^3)$ time to solving a linear program with $O(n^2)$ variables and constraints, where $n$ is the number of bridges in the bridge graph.*

**Proof.** For each edge $e$ of $\mathcal{G}$, we introduce a variable $X_e$ specifying the position of the corresponding bridge; $X_e$ is the $x$-coordinate of a vertical bridge or the $y$-coordinate of a horizontal bridge. We also introduce a variable $Z$. Our linear program will be such that a variable assignment is feasible if and only if the bridge assignment prescribed by the $X_e$ is a configuration realizing $\mathcal{G}$ with dilation $\leqslant Z$. Minimizing $Z$ will then solve the bridge-placement problem.

We will need a number of extra variables. We first define a set of points $U$ by taking all points that may be involved in a visible pair (in the sense of Lemma 4), as well as all bridge endpoints. By Lemma 4, the size of $U$ is $O(n)$. Some of the points in $U$, namely the points in a visible pair, are of the form $(\text{const}, \text{const})$, where "const" means: any constant. Other points in $U$ are of the form $(\text{const}, X_e)$ (the

endpoints of a horizontal bridge) or $(X_e, \text{const})$ (the endpoints of a vertical bridge). For each pair of points $u$ and $v$ from $U$ that lie in the same building, we introduce an extra variable $D_{uv}$.

We can now describe the linear program. For each $X_e$, we need two simple constraints of the form $X_e \geqslant \text{const}$ and $X_e \leqslant \text{const}$, ensuring that the bridge indeed lies in the bridge region. For each $D_{uv}$, we add constraints enforcing $D_{uv} \geqslant \|uv\|$, as follows. Let $u = (x_u, y_u)$, $v = (x_v, y_v)$ (recall that each coordinate is either a constant, or one of the variables $X_e$, for some edge $e$). Then we add the constraints:

$$
\begin{aligned}
D_{uv} &\geqslant x_u - x_v + y_u - y_v,\\
D_{uv} &\geqslant x_v - x_u + y_u - y_v,\\
D_{uv} &\geqslant x_u - x_v + y_v - y_u,\\
D_{uv} &\geqslant x_v - x_u + y_v - y_u.
\end{aligned}
$$

Clearly, these four constraints together guarantee that $D_{uv} \geqslant \|uv\|$.

Finally, we introduce one constraint for each pair $p, q \in U$. Let $bl(p, q)$ be the total length of all bridges traversed by $\pi(p, q, B)$. Since $\mathcal{G}$ is a tree, the buildings and bridges traversed by $\pi(p, q, B)$ are independent of the configuration, and so $bl(p, q)$ is a constant. We can now write

$$\bigl|\pi(p, q, B)\bigr| = bl(p, q) + \sum_{uv} \|uv\|,$$

where the sum is over the entry and exit points $u$ and $v$ of $\pi(p, q, B)$ for each building traversed. Note that $u, v \in U$, and $u$ and $v$ lie in the same building. We introduce the constraint

$$bl(p, q) + \sum_{uv} D_{uv} \leqslant Z \cdot \|pq\|.$$

We now argue that if a variable assignment is feasible in this LP, then the bridge assignment prescribed by the $X_e$ is a configuration realizing $\mathcal{G}$ with dilation $\leqslant Z$. Indeed, consider a visible pair $(p, q)$. We have

$$\bigl|\pi(p, q, B)\bigr| = bl(p, q) + \sum_{uv} \|uv\| \leqslant bl(p, q) + \sum_{uv} D_{uv} \leqslant Z \cdot \|pq\|,$$

and so $\text{dil}(\pi(p, q, B)) \leqslant Z$.

On the other hand, assume there is a configuration $B$ realizing $\mathcal{G}$. Let $X_e$ be the placement of the bridge $e$ in $B$, let $D_{uv} = \|uv\|$, and let $Z$ be the dilation of $B$. It is now easy to see that this variable assignment is feasible.

It follows that the bridge-placement problem can be solved by minimizing $Z$ with respect to the LP described.

The LP can be constructed in $O(n^3)$ time: establishing $U$ costs $O(n \log n)$ time (Lemma 4), and $O(n^2)$ constraints are generated. Listing the $O(n)$ variables in the constraint for a pair $p, q \in U$ boils down to finding the path in the bridge tree from $p$ to $q$. This can be done in $O(n)$ time per constraint, for a total time of $O(n^3)$. □

## 4. The bridge graph is a path

In the previous section we have given a linear program for the bridge-placement problem for the case where the bridge graph is a tree. Linear programs can be solved in practice, and for integer coefficients,

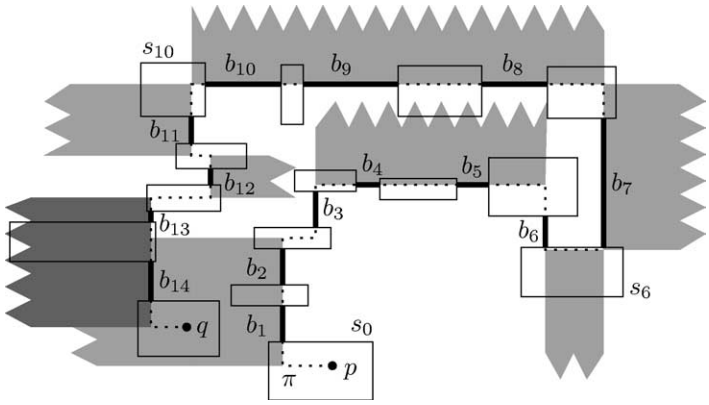


Fig. 7. U-turns and their outer sides.

interior-point methods can solve them in time polynomial in the bit-complexity of the input [6]. It is not known, however, if they can be solved in polynomial time on the real RAM, the standard model of computational geometry. In this section, we give polynomial time algorithms for the case where the bridge graph is a path.

Since the bridge graph $\mathcal{G}$ is a path, we can number the buildings and bridges so that bridge $b_i$ connects buildings $s_{i-1}$ and $s_i$, for $1 \leqslant i \leqslant n$ (so there are $n+1$ buildings and $n$ bridges). Before we continue, we need to introduce some more terminology. We consider a path $\pi = \pi(p, q, B)$ to be oriented from $p$ to $q$. After traversing a bridge $b$, the path can continue straight on to traverse the next bridge $b'$ if $b$ and $b'$ are collinear. In all other cases, it has to turn.

Given a path $\pi$, a *link* $\ell$ of $\pi$ is a maximal straight segment of the path. A link can contain more than one bridge if they are collinear. For example, in Fig. 7 there is a link containing $b_1$ and $b_2$, and another link containing $b_8$, $b_9$ and $b_{10}$.

The path $\pi$ turns at both ends of a link (except for the first and last link). The link is a *right U-turn* if $\pi$ turns right before and after the link. A *left U-turn* is defined symmetrically. In Fig. 7, the links containing bridges $(b_1, b_2)$, $(b_4, b_5)$ and $b_{12}$ are right U-turns, while the links containing $b_7$, $(b_8, b_9, b_{10})$, $b_{11}$ and $(b_{13}, b_{14})$ are left U-turns. Note that there can be U-turns that do not contain any bridges, such as the link of $\pi$ inside building $s_6$ in Fig. 7.

The *inner side* and *outer side* of a U-turn are rectangular regions infinite on one side, and bounded by the line supporting the link and the two lines orthogonal to it through the first and last points of the link. The outer side lies locally to the left of a right U-turn, or to the right of a left U-turn, the inner side lies locally to the right of a right U-turn or to the left of a left U-turn. In Fig. 7, the outer sides of all U-turns are shaded.

U-turns are the links of a path that determine its dilation, as the following lemma shows.

**Lemma 6.** *Let $B$ and $B'$ be configurations, $(p, q)$ a visible pair, and $\pi := \pi(p, q, B)$ and $\pi' := \pi(p, q, B')$ the paths between $p$ and $q$ with respect to the two configurations. If* $\mathrm{dil}(\pi') < \mathrm{dil}(\pi)$ *then there exists a U-turn $\ell$ containing $b_i \ldots b_j$ of $\pi$ such that the corresponding bridges $b'_i, \ldots, b'_j$ of $B'$ lie strictly on the inner side of $\ell$.*

**Proof.** For each U-turn $\ell$ of $\pi$, shade the outer side of $\ell$ as in Fig. 7. It is easy to see that $\pi$ is a shortest rectilinear path from $p$ to $q$ that visits all the shaded regions in order. Now suppose that the path $\pi'$ has smaller dilation than $\pi$ and that there is no U-turn $\ell$ of $\pi$ such that the corresponding bridges of $\pi'$ lie strictly on the inner side of $\ell$. Then $\pi'$ visits the shaded regions in the same order as $\pi$, and is therefore at least as long as $\pi$, a contradiction to $\mathrm{dil}(\pi') < \mathrm{dil}(\pi)$. □

### *4.1. The decision problem*

We will give an algorithm that takes as input the set of buildings $s_0, \ldots, s_n$ and a real number $\sigma > 1$, and computes a configuration $B$ with $\mathrm{dil}(B) \leqslant \sigma$, or determines that no such configuration exists.

The algorithm computes $n$ sets $I_1, I_2, \ldots, I_n$, where $I_i$ is a set of possible bridges between $s_{i-1}$ and $s_i$. The sets are defined recursively as follows. Assume that $I_1, \ldots, I_{i-1}$ have already been defined. For each visible pair $(p, q)$ with $p \in \bigcup_{j=0}^{i-1} s_j$ and $q \in s_i$ we define $I(p, q)$ as the set of bridges $b_i$ connecting $s_{i-1}$ and $s_i$ such that the following holds: there is a set of bridges $b_1 \in I_1, b_2 \in I_2, \ldots, b_{i-1} \in I_{i-1}$ such that $\mathrm{dil}(\pi(p, q, (b_1, \ldots, b_i))) \leqslant \sigma$. Finally, $I_i$ is the intersection of all $I(p, q)$.

Note that for each visible pair $(p, q)$ we can choose the bridges in $I_1, \ldots, I_{i-1}$ independently. This makes it possible to compute $I_i$ efficiently, as we will see below. On the other hand, it implies that not every sequence of bridges chosen from the sets will be a configuration with dilation at most $\sigma$—our main lemma will be to show that such a sequence does indeed exist.

The opposite direction is nearly trivial: if a configuration with dilation at most $\sigma$ exists, it can be found in the sets we constructed, as we show now.

**Lemma 7.** *Let $B = (b_1, b_2, \ldots, b_n)$ be a configuration such that $b_i \notin I_i$ for some $i$. Then* $\mathrm{dil}(B) > \sigma$.

**Proof.** Let $i$ be the smallest index with $b_i \notin I_i$. Since $b_i \notin I_i$, there exists a visible pair $(p, q)$ with $p \in s_j$, $j < i$, and $q \in s_i$ such that for any set of bridges chosen from $I_1, \ldots, I_{i-1}$ the path between $p$ and $q$ that uses these $i - 1$ bridges and the bridge $b_i$ has dilation larger than $\sigma$. Since by our choice of $i$ we have $b_k \in I_k$ for $k < i$, we have indeed $\mathrm{dil}(\pi(p, q, B)) > \sigma$. □

We first argue that the sets $I_i$ can be represented and managed easily.

**Lemma 8.** *Let $I_1, I_2, \ldots, I_n$ be defined as above. Then the $x$-coordinates* ($y$-*coordinates*) *of the bridges in each set form an interval.*

**Proof.** It is sufficient to show that the sets $I(p, q)$ are intervals. Consider a visible pair $(p, q)$ with $p \in s_j$ and $q \in s_i$. Without loss of generality, assume the bridges in $I(p, q)$ to be vertical. Take three bridges $a, b, c$ with $x$-coordinates $a_x < b_x < c_x$ and $a, c \in I(p, q)$. We will show that $b \in I(p, q)$.

Due to symmetry, we can assume $q_x \geqslant b_x$. Since $a \in I(p, q)$, a path $\pi = \pi(p, q, (b_1, \ldots, b_{i-1}, a))$ exists (fat gray in Fig. 8) with $\mathrm{dil}(\pi) \leqslant \sigma$ that uses bridges $b_1 \in I_1, \ldots, b_{i-1} \in I_{i-1}$. Now we can exchange the part of $\pi$ from where $\pi$ enters $a$ to where $\pi$ reaches $q$ by a piece that uses $b$ instead of $a$ (dashed black in Fig. 8). This new path is at most as long as $\pi$, which shows that $b \in I(p, q)$. □

Once we know $I_1, \ldots, I_n$, we can recursively compute a configuration with dilation at most $\sigma$: choose an arbitrary bridge $b_n \in I_n$. If bridges $b_{n-1}, b_{n-2}, \ldots, b_{i+1}$ have been computed, choose a bridge $b_i \in I_i$

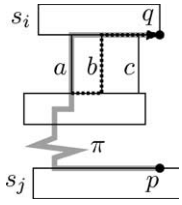


Fig. 8. Proof of Lemma 8.

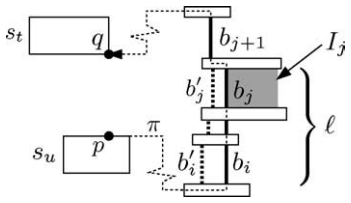


Fig. 9. Proof of Lemma 9.

whose distance from $b_{i+1}$ is minimal. Since $I_i$ is an "interval of bridges", this implies that either $b_i$ and $b_{i+1}$ are collinear, or $b_i$ is one of the extreme bridges in $I_i$. We now prove that this approach is correct.

**Lemma 9.** *Let $I_1, \ldots, I_n$ be given as defined above. A configuration $B$ with dilation $\mathrm{dil}(B) \leqslant \sigma$ exists if and only if $I_n \neq \emptyset$. If it exists, it can be computed in $O(n)$ time from the intervals.*

**Proof.** The "only if" part follows from Lemma 7. We show the "if" part by proving that the configuration $B = (b_1, \ldots, b_n)$ defined above has dilation $\leqslant \sigma$. Since this configuration can clearly be computed in linear time from the intervals, the last statement of the lemma will follow at the same time.

Assume that $\mathrm{dil}(B) > \sigma$. Then there is a visible pair $(p, q)$, such that $\mathrm{dil}(\pi(p, q, B)) > \sigma$. Let $\pi = \pi(p, q, B)$, and let $s_u$, $s_t$ be the buildings containing $p$ and $q$. Without loss of generality we can assume $u < t$. Since $b_t \in I_t$, there is a sequence of bridges $b'_1, \ldots, b'_{t-1}$ with $b'_k \in I_k$, such that the path $\pi' = \pi(p, q, (b'_1, \ldots, b'_{t-1}, b_t))$ has dilation at most $\sigma$.

We have $\mathrm{dil}(\pi') \leqslant \sigma < \mathrm{dil}(\pi)$. By Lemma 6 there is a U-turn $\ell = (b_i, \ldots, b_j)$ of $\pi$ (without loss of generality assumed to be a left U-turn) such that all the bridges $b'_i, \ldots, b'_j$ lie strictly to the left of $\ell$, see Fig. 9.

The last bridge of both $\pi$ and $\pi'$ is $b_t$, so $j < t$. It follows that $\pi$ passes through $b_{j+1}$. Since $\ell$ is a left U-turn, the bridge $b_{j+1}$ is strictly to the left of $b_j$. By definition of $b_j$, however, this implies that $b_j$ is the left endpoint of $I_j$, and $b'_j \notin I_j$, a contradiction. □

Given a point $p$ in a building $s_u$, we can define a configuration $B^p$ that is, in a sense, optimal for $p$ by choosing bridges $b^p_1, \ldots, b^p_n$ as follows. For $k \leqslant u$, choose an arbitrary bridge $b^p_k \in I_k$. Choose bridge $b^p_{u+1}$ as close as possible to $p$. The remaining bridges are chosen recursively, by choosing $b^p_k \in I_k$ to be as close to $b^p_{k-1}$ as possible. Let $m^p_i$ denote the endpoint of $b^p_i$ on the building $s_i$. The following lemma shows that $B^p$ is indeed optimal for $p$.

**Lemma 10.** *Let intervals $I_1, \ldots, I_n$ be as defined above, let $p \in s_u$ and $q \in s_t$, with $u < t$. Furthermore, let $B = (b_1, \ldots, b_n)$ be a configuration with $b_i \in I_i$ for $i < t$, and let $B'$ be the configuration $(b_1, \ldots, b_u, b^p_{u+1}, \ldots, b^p_{t-1}, b_t, \ldots, b_n)$. Then $\mathrm{dil}(p, q, B') \leqslant \mathrm{dil}(p, q, B)$.*

**Proof.** Let $\pi = \pi(p, q, B)$ and $\pi' = \pi(p, q, B')$. Assume that $\mathrm{dil}(\pi') > \mathrm{dil}(\pi)$. By Lemma 6 there is then a U-turn $\ell = (b^p_i, \ldots, b^p_j)$ of $\pi'$ (without loss of generality assumed to be a left U-turn) such that the corresponding bridges of $\pi$ lie strictly to the left of $\ell$. Since $\ell$ is a left U-turn, the bridge $b^p_{i-1}$ (or the point $p$, if $i - 1 = u$) lies to the left of $b^p_i$. The definition of $b^p_i$ implies that $b^p_i$ is then the leftmost bridge in $I_i$, a contradiction with $b_i \in I_i$. □

The following lemma shows that optimal paths are helpful in computing the intervals $I_i$.

**Lemma 11.** *Let $p \in s_u$, $q \in s_i$, with $u < i-1$. The interval $I(p,q)$ can be computed in constant time if $b_{i-1}^p$ and $|\pi(p, m_{i-1}^p, B^p)|$ are known.*

**Proof.** Recall that $I(p,q)$ is defined as the set of all bridges $b_i$ connecting $s_{i-1}$ and $s_i$, such that there is a set of bridges $b_1 \in I_1, b_2 \in I_2, \ldots, b_{i-1} \in I_{i-1}$ with $\mathrm{dil}(\pi(p,q,(b_1,\ldots,b_i))) \leqslant \sigma$. By Lemma 10 this is equivalent to $\mathrm{dil}(\pi(p,q,(b_1^p, b_2^p, \ldots, b_{i-1}^p, b_i))) \leqslant \sigma$. This path coincides with $\pi(p,q,B^p)$ up to and including bridge $b_{i-1}^p$, which is the path $\pi(p, m_{i-1}^p, B^p)$. Since the length of this path is known, we can compute $I(p,q)$ in constant time. □

**Lemma 12.** *The intervals $I_1, \ldots, I_n$ defined above can be computed in $O(n^2)$ time and $O(n)$ space.*

**Proof.** Let $P$ denote the set of all building corners and all points $p$ such that there is a visible pair $(p,q)$ with $p \in s_u$, $q \in s_t$ and $u < t$. By Lemma 4, $P$ contains at most $12n$ points and it can be computed in $O(n \log n)$ time.

For each building $s_t$, we create a list of visible pairs $(p,q)$ with $q \in s_t$ and $p \in \bigcup_{u=0}^{t-1} s_u$ such that not both $p$ and $q$ are building corners. This can be done during the same computation.

The computation then proceeds in $n$ stages, with stage $i$ computing interval $I_i$. Throughout, we maintain for each point $p \in P$ the bridge $b_i^p$, as well as the length of the path $\pi(p, m_i^p, B^p)$.

Consider stage $i$. We compute the intervals $I(p,q)$, for all pairs $(p,q)$ with $p \in \bigcup_{u=0}^{i-1} s_u$ and $q \in s_i$ that are either visible pairs or where both $p$ and $q$ are building corners. (This avoids the need to precompute and store $O(n^2)$ visible pairs.) Note that all the points $p$ appearing in such pairs are in $P$, and so there are at most $12n$ such pairs.

By Lemma 11, it takes constant time to compute $I(p,q)$ using the information from the previous stage. We can determine $b_i^p$ and update the stored length for $\pi(p, m_i^p, B^p)$ in constant time as well.

It takes $O(n)$ time to compute the intersection interval $I_i$, so the total time spent per stage is $O(n)$. □

Lemmas 12 and 9 imply the following theorem.

**Theorem 13.** *Given a bridge graph $\mathcal{G}$ on a set of $n+1$ buildings that is a path and a real number $\sigma > 1$, we can in time $O(n^2)$ compute a configuration $B$ realizing $\mathcal{G}$ with $\mathrm{dil}(B) \leqslant \sigma$ or determine that no such configuration exists.*

It seems hard to improve this result when there are $\Theta(n^2)$ visible pairs that could determine the dilation. In fact, we do not even know how to decide in $o(n^2)$ time whether a *given* configuration has dilation $\leqslant \sigma$.

If the number of visible pairs of the given set of buildings is $o(n^2/\log n)$, it is possible to do better. The difficulty is that the size of the set $P$ is still linear, and we cannot maintain $b_i^p$ for all points $p \in P$ explicitly. Instead, we store $b_i^p$ and $|\pi(p, m_i^p, B^p)|$ in data structures that allow us to update them efficiently. We will need the simple data structure described in the following lemma.

**Lemma 14.** *There is a data structure that stores $m$ real numbers $a_1, \ldots, a_m$, can be built in time $O(m)$, and supports the following operations in time $O(\log m)$:*

- *given an index* $j \in \{1, \ldots, m\}$, *return* $a_j$;
- *given two indices* $j', j'' \in \{1, \ldots, m\}$ *and a real number* $b$, *replace the value of* $a_j$ *by* $a_j + b$ *for all* $j' \leqslant j \leqslant j''$.

**Proof.** The data structure is basically a segment tree [2]. It is a balanced binary tree, whose leaves correspond to the indices $1, \ldots, m$ in order. Each node $v$ of the tree contains a real number $b_v$, and the value of $a_j$ for a leaf $j$ is the sum of $b_v$ over the nodes on the path from the root to $j$. Clearly it can be returned in time $O(\log m)$. For the last operation, we find all the nodes $v$ of the tree such that the indices of all leaves in the subtree rooted at $v$ lie within the interval $[j', j'']$, while the subtree rooted at the parent of $v$ contains at least one leaf outside this interval. For all such nodes $v$, we add $b$ to $b_v$. □

Let again $\Lambda_i$ be the bridge region connecting $s_{i-1}$ and $s_i$. Let $b$ and $b'$ be two bridges in $\Lambda_i$, and consider them directed from $s_{i-1}$ to $s_i$. We let $b \prec b'$ if and only if $b$ lies left of $b'$. Now let $P$ be the set of points defined in Lemma 12, and let $P_i := P \cap \bigcup_{j=0}^{i} s_j$. Consider the union of all rectangles and all bridge regions. This is a single rectilinear polygon. We order the points of $P$ along the boundary of this polygon, in counter-clockwise order starting and ending on $s_n$ (note that there are no points of $P$ in $s_n$) and denote this order again by $\prec$.

**Lemma 15.** *Let* $p, p' \in P_{i-1}$. *If* $b_i^p \prec b_i^{p'}$ *then* $p \prec p'$.

**Proof.** If $p' \prec p$ while $b_i^p \prec b_i^{p'}$, then the paths $\pi(p, m_i^p, B^p)$ and $\pi(p', m_i^{p'}, B^{p'})$ have to cross, which is impossible. □

**Theorem 16.** *Given a bridge graph* $\mathcal{G}$ *on a set of* $n+1$ *buildings that is a path, and a real number* $\sigma > 1$, *we can in time* $O(k \log n)$ *compute a configuration* $B$ *realizing* $\mathcal{G}$ *with* $\mathrm{dil}(B) \leqslant \sigma$ *or determine that no such configuration exists, where* $k$ *is the number of visible pairs.*

**Proof.** It is sufficient to show how to compute the intervals $I_i$. We start by computing all visible pairs. This can be done in time $O(k \log n)$ (note that $k \geqslant n$), by computing both vertical and horizontal decompositions [2], and a modified version of the algorithm for reporting all direct visibility pairs by de Berg et al. [1]. For each building $s_t$ we build a list of visible pairs $(p, q)$ with $q \in s_t$ and $p \in P_{t-1}$.

The algorithm proceeds again in $n$ stages, computing $I_i$ in stage $i$. We maintain two data structures, $\mathcal{P}$ (paths) and $\mathcal{B}$ (bridges). $\mathcal{P}$ is the data structure of Lemma 14. It stores for each $p \in P$ a value $a_p$, with the points sorted by $\prec$. If $p \in s_u$, then $a_p = 0$ up to stage $u+1$, and $a_p = |\pi(p, m_{i-1}^p, B^p)|$ when stage $i \geqslant u+2$ is about to start. $\mathcal{B}$ is a dictionary. At the beginning of stage $i$, it stores all the bridges $b_{i-1}^p$, for $p \in P_{i-2}$, in the order $\prec$. A bridge shared by several points is only stored once. For each bridge $b$, we store the $x$- or $y$-coordinate, and two points $p', p'' \in P_{i-2}$ such that $b_{i-1}^p = b$ if and only if $p' \preceq p \preceq p''$. This is possible by Lemma 15.

In stage $i$, we retrieve the list of visible pairs $(p, q)$ with $q \in s_i$. For each pair, we compute $I(p, q)$. If $p \in s_{i-1}$, this is done directly, in constant time. Otherwise $p \in P_{i-2}$, and we compute $I(p, q)$ from $b_{i-1}^p$ and $|\pi(p, m_{i-1}^p, B^p)|$ in constant time by Lemma 11. We can find the bridge $b_{i-1}^p$ in $O(\log n)$ time in $\mathcal{B}$—by Lemma 15 $\mathcal{B}$ is sorted by points as well as by bridges. The value $|\pi(p, m_{i-1}^p, B^p)|$ is stored in $\mathcal{P}$. It follows that the total time, over all stages, for this computation is $O(k \log n)$.

It remains to discuss the updating of $\mathcal{P}$ and $\mathcal{B}$ to prepare them for the next stage. Let's first discuss $\mathcal{B}$. Consider the interval $I_{i-1}$. The part of $I_{i-1}$ that continues straight on into $I_i$ doesn't need to be touched. The bridges $b_{i-1}^p$ on the left or right of $I_{i-1}$ that cannot continue straight on (all bridges, if the orientation of $I_{i-1}$ and $I_i$ is different) are removed, and replaced by bridges on the edges of $I_i$. In addition, we insert new bridges for all $p \in P \cap s_{i-1}$. This can be done in time $O(d \log n)$, where $d$ is the number of bridges being removed and created. We charge the cost of removing a bridge to its creation. Since the number of bridges created during the course of the algorithm is $|P| + 2n = O(n)$, the total time for this is $O(n \log n)$.

Finally, we discuss the updating of $\mathcal{P}$. For all the bridges of $I_{i-1}$ that go straight on to $I_i$, we need to increase the path length by the same value. By Lemma 15, they correspond to a single interval of points of $P$, and so this can be done in time $O(\log n)$. For each bridge that has been removed, we increase the path length for its interval of points, in time $O(\log n)$ per bridge removed. Finally, for each point $p \in P \cap s_{i-1}$ inserted in this stage, we set its path length to the correct value. The total cost of updating is $O(n \log n)$ according to Lemma 14. □

### *4.2. The optimization problem*

We can now solve the original optimization problem using Megiddo's parametric search [7].

**Theorem 17.** *Given a bridge graph on a set of $n + 1$ buildings that is a path, we can compute a configuration with the optimal dilation in time $O(n^3 \log n)$, or in time $O(nk \log^2 n)$, where $k$ is the number of visible pairs.*

**Proof.** We run the algorithm of Lemma 12 with input $\sigma^*$, where $\sigma^*$ is the optimal dilation. Since $\sigma^*$ is not known, we parameterize all coordinates used by the decision algorithm in the form $a\sigma + b$. One can verify that all calculations performed by the algorithm are linear functions on the coordinates, and any linear combination of expressions of the form $a\sigma + b$ is again of this form.

Whenever the algorithm needs to compare two "numbers" $a\sigma + b$ and $a'\sigma + b'$, we compute the value $\sigma_0$ where $a\sigma_0 + b = a'\sigma_0 + b'$. We then run the decision algorithm of Theorem 13 using $\sigma_0$, which tells us whether $\sigma^* \leqslant \sigma_0$. The answer implies which of the two "numbers" is larger, and the parametrized algorithm can proceed. Note that if $\sigma^* = \sigma_0$, the outcome of the comparison is arbitrary—inspection of the algorithm shows that this is not a problem.[1]

When the parametrized algorithm finishes, it has computed a set of non-empty intervals $I_1, \ldots, I_n$, since a configuration with dilation $\leqslant \sigma^*$ exists. Since the outcome of the parametrized algorithm changes for $\sigma = \sigma^*$, the algorithm must have made a comparison against $\sigma^*$. It follows that $\sigma^*$ is the smallest $\sigma_0$ tested during the algorithm that resulted in a positive answer of the decision algorithm.

During the algorithm we maintain an interval of dilation values in which the optimal value is known to lie. Whenever a comparison requires answering $\sigma^* \leqslant \sigma_0$ for a $\sigma_0$ outside this interval, we can immediately return the correct answer without running the decision algorithm. At the end of the parametrized algorithm, we can report the upper end of the interval as $\sigma^*$.

[1] The reader may wonder why we do not simply augment the algorithm of Theorem 13 to report whether a configuration with dilation strictly less than $\sigma$ exists. This is indeed possible, for instance by allowing open and half-open intervals $I_i$, but seems to be more complex than the observation that tests for equality are not actually needed.

Following Megiddo [7], we organize the parametric algorithm as a "parallel" algorithm, using batches of independent computations. Recall that the algorithm of Lemma 12 proceeds in $n$ stages, with stage $i$ computing $I(p,q)$ for $O(n)$ pairs $(p,q)$ with $q \in s_i$. The computations for each pair are independent, and take time $O(1)$. It follows that we can implement them in total time $O(n \log n)$ plus $O(\log n)$ calls to the decision algorithm [7].

Forming the intersection $I_i$ is equivalent to the computation of a maximum and a minimum of $n$ "numbers" of the form $a\sigma + b$. Consider the "number" $a\sigma + b$ as the line $y = ax + b$. We compute the upper and lower envelope of all $n$ lines, in time $O(n \log n)$ [2]. We can now perform binary search on the vertices of the envelopes, using $O(\log n)$ calls to the decision algorithm, to determine between which two vertices $\sigma^*$ falls. This allows us to return the largest and smallest "number".

Each stage takes time $O(n \log n)$ plus $O(\log n)$ calls to the decision algorithm, so the total running time is $O(n^3 \log n)$. We can also use Theorem 16 to obtain total running time $O(nk \log^2 n)$. □

### 4.3. The case of vertically sorted buildings

There is one interesting case where we can prove that there are only $O(n)$ visible pairs, namely when the buildings are sorted vertically along the path, that is, all bridges are directed vertically upwards.

**Lemma 18.** *If the bridge graph is a path, and the $n+1$ buildings are sorted vertically along the path, then there are at most $O(n)$ visible pairs.*

**Proof.** A visible pair appears in the vertical decomposition of the set of buildings. □

Theorem 16 now leads to an $O(n \log n)$-time decision algorithm for this case. It is possible to do even better, as we will show in this section.

The improvement is based on a bracket structure formed by the visible pairs. Consider a visible pair $(p,q)$. The segment $pq$ is vertical. Without loss of generality, let $p$ be its bottom end. The path $\pi(p,q,B)$ is $y$-monotone, and since it cannot intersect $pq$, it lies either completely to the left or to the right of $pq$. We call a visible pair $(p,q)$ where the path lies completely to the right of $pq$ a *left-hand* visible pair, otherwise a *right-hand* visible pair.

**Lemma 19.** *Given a set of $n+1$ vertically sorted buildings as defined above, and two left-hand visible pairs $(p,q)$ and $(p',q')$, with $p \in s_u$, $q \in s_t$, $p' \in s_{u'}$, $q' \in s_{t'}$. Assume that $u \leqslant u'$. Then either the pairs are independent and $t \leqslant u'$, or $(p,q)$ is bracketed around $(p',q')$, that is, $p_x < p'_x$ and $u \leqslant u' < t' \leqslant t$.*

**Proof.** If $u' < t$, then the building $s_{u'}$ lies completely to the right of the segment $pq$, and so we have $p_x < p'_x$. The path $\pi(p',q',B)$ lies completely to the right of the segment $p'q'$, and so it cannot reach $s_t$ before reaching $s_{t'}$. This implies $u \leqslant u' < t' \leqslant t$. □

In a left-hand visible pair $(p,q)$, either $p$ is the top-left corner of a building and $q$ is on a bottom edge of a building, or $q$ is a bottom-left corner and $p$ is on the top edge of a building. Lemma 19 leads to a simple algorithm to compute all left-hand visible pairs in linear time. (The same procedure, with opposite orientation, can be used to find all right-hand visible pairs.) All we need is a stack. In stage $i$, we repeatedly check whether $p_x \geqslant q_x$, where $p$ is the top element of the stack and $q$ is the bottom-left

corner of $s_i$. While that is true, we report $(p, (p_x, q_y))$ as a visible pair and pop $p$ from the stack. Finally, either the stack is empty, or $p_x < q_x$. In the latter case, we report $((q_x, p_y), q)$ as a visible pair. Finally, we push the top-left corner of $s_i$ onto the stack, and proceed to the next stage.

**Theorem 20.** *Given a set of $n+1$ vertically sorted buildings as defined above and a real number $\sigma > 1$. We can compute in $O(n)$ time a configuration $B$ with dilation $\mathrm{dil}(B) \leqslant \sigma$, or determine that none exists.*

**Proof.** Again, we compute the intervals $I_1, \ldots, I_n$ in $n$ stages. The visible pairs are computed during the process, using a "left-side stack" for the top-left corners and a "right-side stack" for the top-right corners. During the course of computation, we again maintain two data structures $\mathcal{P}$ and $\mathcal{B}$ to store path lengths and optimal bridges. Define the index of the top-left corner of building $s_u$ to be $-(u+1)$, and the index of the top-right corner of $s_u$ to be $u+1$.

$\mathcal{P}$ is implemented as a doubly-linked list. In this list, we store the path lengths $|\pi(p, m_i^p, B^p)|$ for all points $p$ currently in the two stacks. The points are ordered by increasing index as defined above (which is the same as ordering them by the relation $\prec$ as defined before). The points on top of the stacks are thus found at the ends of the list. We store the path lengths by storing the *difference* between two adjacent values on the edges of the list. Only for the first and the last point in the list, we store $|\pi(p, m_i^p, B^p)|$ explicitly. Note that we do not explicitly store path lengths for points $p$ that are not the corner of a building. However, these path lengths can be derived in constant time from path lengths that are stored in $\mathcal{P}$: if $p$ on a building with top-left corner $l$ is part of a left-hand visible pair, then $|\pi(p, m_i^p, B^p)|$ is simply $|\pi(l, m_i^l, B^l)| - |pl|$; similarly, if $p$ is part of a right-hand visible pair, we can derive the path length from that of a top-right corner. Note that we can easily increase the path lengths for an interval of points in $\mathcal{P}$ in constant time by adjusting two difference or end values, provided we have pointers to the first and the last point of the interval.

$\mathcal{B}$ stores the optimal bridges $b_i^p$ for all points $p$ currently in the two stacks, and is implemented as a doubly-linked list as well. As before, a bridge shared by several points is stored only once. With each bridge, we store the index of the first and last point using it. For each point index, we store a pointer to the node of $\mathcal{P}$ that represents it.

A stage is now implemented as follows:

1. Using the two stacks, compute left-hand and right-hand visible pairs. Accessing the leftmost and rightmost nodes in $\mathcal{B}$ and $\mathcal{P}$, we can obtain path length values and bridge positions for these points. With these values, we compute the new interval $I_i$.
2. Remove from the ends of $\mathcal{P}$ all nodes for points popped from the two stacks. Remove from the ends of $\mathcal{B}$ all bridges that are not used by any point anymore (these bridges can be identified by comparing the index of the point on top of the stack with the indices of the points using the bridge). Adjust the interval of points used by the leftmost and rightmost bridge to end at the points on the top of the stacks.
3. For each bridge $b$ in $I_{i-1}$ that cannot go straight into $I_i$, update the path lengths for the corresponding interval of points in $\mathcal{P}$ (using the indices of the points for $b$ and the pointers for these indices into $\mathcal{P}$).
4. Finally, remove all these bridges, update in $\mathcal{P}$ the interval of all points that use the remaining bridges (the bridges that do continue straight into $I_i$), add the top-left and top-right corner of $s_{i-1}$ to $\mathcal{P}$ and add new bridges at the left and right margin of $I_i$, set the point interval of these bridges to the union

of what was just deleted and the new corner points, and push the top-left and top-right corner of $s_i$ on the two stacks.

Observe that all queries and updates of $\mathcal{B}$ and $\mathcal{P}$ are done at the ends of the lists and can be done in constant time each. Only updating path lengths in $\mathcal{P}$ requires access to an edge in the interior of the list, but this edge is found in constant time through the indices stored with the corresponding bridge at the end of $\mathcal{B}$. As before, the removal of bridges is charged to their creation. We thus spend constant time per stage, plus constant time per visible pair. □

Parametric search now leads directly to the following theorem. Unlike in Theorem 17, we make no attempt to parallelize the parametric algorithm.

**Theorem 21.** *Given a bridge graph on a set of $n+1$ buildings that is a path, we can compute a configuration with the optimal dilation in time* $O(n^2)$.

Finally, we can compute a $(1+\varepsilon)$-approximation in linear time. We first show a quality bound for an arbitrary placement of the bridges. For completeness, we cover the general case as well.

**Lemma 22.** *Given a bridge graph $\mathcal{G}$ on a set of $n+1$ buildings that is a path, and any configuration $B$ realizing $\mathcal{G}$. Then* $\mathrm{dil}(B) \leqslant (\sigma^*)^2$, *where $\sigma^*$ is the optimal dilation. If the buildings are sorted vertically along the path, then we have* $\mathrm{dil}(B) \leqslant 2\sigma^*$.

**Proof.** Let $B^* = (b_1^*, b_2^*, \ldots, b_n^*)$ be an optimal configuration, that is $\mathrm{dil}(B^*) = \sigma^*$. Consider the interval of possible bridges between $s_{i-1}$ and $s_i$, see Fig. 10. Let $d_i$ be the distance of $b_i^*$ to the farther endpoint of the interval, and let $h_i$ be the length of $b_i^*$. The pair of points $(p', q')$ indicated in the figure has dilation $(2d_i + h_i)/h_i \leqslant \sigma^*$, which implies $2d_i \leqslant (\sigma^* - 1)h_i$.

Now consider any visible pair $(p, q)$. If $\pi(p, q, B)$ uses bridges $b_u, \ldots, b_t$, we have

$$\begin{aligned}\big|\pi(p,q,B)\big| &\leqslant \big|\pi(p,q,B^*)\big| + \sum_{i=u}^{t} 2d_i \leqslant \big|\pi(p,q,B^*)\big| + (\sigma^*-1)\sum_{i=u}^{t} h_i \\ &\leqslant \big|\pi(p,q,B^*)\big| + (\sigma^*-1)\big|\pi(p,q,B^*)\big| \leqslant \sigma^*\big|\pi(p,q,B^*)\big| \leqslant (\sigma^*)^2\|pq\|.\end{aligned}$$

If the buildings are sorted vertically along the path, we can observe that $\|pq\| \geqslant \sum_{i=u}^{t} h_i$, and so we have

$$\big|\pi(p,q,B)\big| \leqslant \big|\pi(p,q,B^*)\big| + \sum_{i=u}^{t} 2d_i \leqslant \sigma^*\|pq\| + (\sigma^*-1)\sum_{i=u}^{t} h_i \leqslant 2\sigma^*\|pq\|. \qquad \square$$

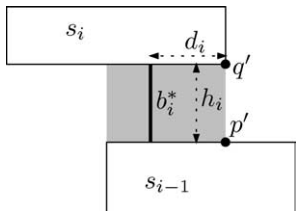


Fig. 10. Proof of Lemma 22.

The lemma leads directly to a PTAS for the vertically ordered case: start with an arbitrary configuration, compute its dilation $\sigma$, and approximate $\sigma^*$ by a binary search in the interval $\langle \sigma/2, \sigma ]$. This gives us a $(1+\varepsilon)$-approximation of $\sigma^*$ after $O(\log(1/\varepsilon))$ calls to the decision algorithm, leading to the following result.

**Theorem 23.** *Given a set of $n+1$ buildings sorted vertically along a path. We can compute a configuration with dilation at most $(1+\varepsilon)$ times the minimum dilation in time $O(n \log(1/\varepsilon))$.*

## 5. Concluding remarks

We posed the following question: given $n$ non-intersecting rectangles and a bridge graph—that is a graph describing which pairs of rectangles are to be connected—how fast can we find the connecting segments that minimize the dilation? We found that if the graph may contain cycles, this problem cannot generally be solved in polynomial time (unless P = NP), but if the graph is a path, the problem can be solved in $O(n^3 \log n)$ time. For the case of trees, the question is still open: so far, we can solve the problem by linear programming on $O(n^2)$ variables and constraints, but we have no strongly polynomial-time algorithm, that is, we have no polynomial-time algorithm for the real RAM model.

Concerning approximations, we have shown that any bridge placement has at most twice the minimum dilation in case the bridge graph is a path and buildings are stacked on top of each other. This observation gives rise to a fast $(1+\varepsilon)$-approximation algorithm. If the bridge graph is a path but buildings do not have a vertical order, we could only show that any bridge placement has dilation $(\sigma^*)^2$, where $\sigma^*$ is the minimum dilation of the instance. Is there a constant-factor-approximation algorithm for such instances? Or more important: is there a constant-factor-approximation algorithm for the case of general bridge graphs?

Having gained some insight in the bridge placement problem when the bridge graph is prescribed, it may now be interesting to study the problem with the bridge graph not given. For example: given a set of non-intersecting rectangles, find a set of connecting segments of given total length such that the dilation is minimized. Or: given a set of non-intersecting rectangles: find a set of connecting segments of minimum total length such that a given dilation is achieved. We might have to settle for approximation algorithms in this case.

When starting this research, we originally asked about how to connect convex polygonal objects by line segments unrestricted in orientation. It will be interesting to see to what extent the techniques for the axis-aligned case carry over to (approximation) algorithms for the unaligned case.

## References

[1] M.T. de Berg, S. Carlsson, M.H. Overmars, A general approach to dominance in the plane, J. Algorithms 13 (1992) 274–296.
[2] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer, Berlin, 1997.
[3] L.P. Chew, There are planar graphs almost as good as the complete graph, J. Comput. Syst. Sci. 39 (1989) 205–219.
[4] D. Eppstein, Spanning trees and spanners, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier, Amsterdam, 2000, pp. 425–461.

[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[6] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
[7] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, J. ACM 30 (4) (1983) 852–865.
[8] D. Peleg, A. Schäffer, Graph spanners, J. Graph Theory 13 (1989) 99–116.