Theoretical Computer Science 985 (2024) 114352

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Routing schemes for hybrid communication networks [☆,☆☆]

Sam Coy [a,1], Artur Czumaj [a,2], Christian Scheideler [b,3], Philipp Schneider [c], Julian Werthmann [b,*,4]

[a] University of Warwick, United Kingdom of Great Britain and Northern Ireland
[b] Paderborn University, Germany
[c] University of Freiburg, Germany

ARTICLE INFO

Keywords:
Hybrid networks
Unit-disk graphs
Routing schemes

ABSTRACT

We consider the problem of computing routing schemes in the HYBRID model of distributed computing where nodes have access to two fundamentally different communication modes. In this problem nodes have to compute small labels and routing tables that allow for efficient routing of messages in the local network, which typically offers the majority of the throughput. Recent work has shown that using the HYBRID model admits a significant speed-up compared to what would be possible if either communication mode were used in isolation. Nonetheless, if general graphs are used as the input graph the computation of routing schemes still takes polynomial rounds in the HYBRID model.

We bypass this lower bound by restricting the local graph to unit-disc-graphs and solve the problem deterministically with running time $O(|\mathcal{H}|^2 + \log n)$, label size $O(\log n)$, and size of routing tables $O(|\mathcal{H}|^2 \cdot \log n)$ where $|\mathcal{H}|$ is the number of "radio holes" in the network. Our work builds on recent work by Coy et al., who obtain this result in the much simpler setting where the input graph has no radio holes. We develop new techniques to achieve this, including a decomposition of the local graph into path-convex regions, where each region contains a shortest path for any pair of nodes in it.

1. Introduction

The HYBRID model was introduced as a means to study distributed systems which leverage multiple communication modes of different characteristics [4]. Of particular interest are networks that combine a *local* communication mode, which has a large

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

☆☆ We would like to thank Martijn Struijs for valuable discussions concerning the geometric insights of the paper. Any errors remain our own.

* Corresponding author.

E-mail addresses: S.Coy@warwick.ac.uk (S. Coy), A.Czumaj@warwick.ac.uk (A. Czumaj), scheideler@upb.de (C. Scheideler), philipp.schneider2@unibe.ch (P. Schneider), jwerth@mail.upb.de (J. Werthmann).

[1] Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP) and an EPSRC studentship.

[2] Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), EPSRC award EP/V01305X/1, and the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021–2023.

[3] This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre *On-The-Fly Computing (GZ: SFB 901/3)* under the project number 160364472.

[4] This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre *On-The-Fly Computing (GZ: SFB 901/3)* under the project number 160364472.

https://doi.org/10.1016/j.tcs.2023.114352

Received 17 July 2023; Received in revised form 8 December 2023; Accepted 15 December 2023

Available online 20 December 2023

bandwidth but is restricted to edges of a graph on the nodes of the network, with a *global* communication mode where any two nodes may communicate in principle, but the bandwidth is heavily restricted. This concept captures various real distributed systems, notably networks of cellphones that combine high bandwidth but locally restricted wireless communication on a *unit-disc-graph* with data transmission over the cellular network.

Routing Schemes are one of the most fundamental distributed data structures, most prominently employed in the Internet, and are used to forward packets among connected nodes in a network in order to facilitate data exchange between any pairs of nodes. In the distributed variant of the problem the nodes initially only know their incident neighbors in the network and need to communicate as efficiently as possible using their available means of communication such that subsequently each node knows its label and a routing table with the following properties. Given a packet with the label of the receiver node in the header, any node must be able to forward this packet in the network using the label and its routing table such that the packet eventually reaches the intended receiver. Algorithms for routing schemes in hybrid networks are of increasing importance, as contemporary communication standards support such settings, one prominent example being the 5G standard [2]. Formally we define routing schemes as follows.

Definition 1 *(Routing schemes)*. A routing scheme on a connected graph $G = (V, E)$ consists of labels $\lambda(v)$ and routing functions (aka routing table) ρ_v for each $v \in V$. ρ_v maps labels to neighbors of v in G, such that the following holds. Let $s, t \in V$. Let $v_0 = s$ and $v_{i+1} = \rho_{v_i}(\lambda(t))$ for $i \geq 1$. Then there is an $\ell \in \mathbb{N}$, such that $v_\ell = t$. A routing scheme is an approximation with *stretch* α if $\ell_{st} \leq \alpha \cdot \mathrm{hop}(s,t)$ for all $s, t \in V$, where $\mathrm{hop}(s,t)$ is smallest number of edges of any st-path, and ℓ_{st} is the length of the induced routing path from s to t.[5]

Since typically large amounts of packets are exchanged between senders and receivers as part of simultaneously ongoing sessions we concentrate on routing schemes for the local network graph, which offers much larger throughput that than what is possible on the global network, since the latter either involves higher costs or is more restricted as infrastructure is shared, like communicating via the cellular network (however, we need very little local communication to actually compute the routing scheme).

Our first goal is to optimize the round complexity of computing such a routing scheme, which is important since frequent changes in the topology of a local network among mobile devices necessitate its fast re-computation. The second goal is to minimize the size of the labels and local routing tables as these must be shared in advance (e.g. via the global network) to initiate a session between two nodes. The third goal is to minimize the *stretch* of the routing path between sender and receiver, minimizing latency and alleviating congestion.

We consider the above problem in the HYBRID model that has received increasing attention during the last few years [4,13,11,7,8,1,14,9]. Formally, the HYBRID model builds on the classic principle of synchronous message passing:

Definition 2 *(Synchronous message passing, cf. [20])*. We have n computational nodes with some initial state and unique identifiers (IDs) in $[n] := \{1, \ldots, n\}$. Time is slotted into discrete rounds. In each round, nodes receive messages from the previous round; they perform (unlimited) computation based on their internal states and the messages they received so far; and finally, based on those computations, they send messages to other nodes in the network.

Note that the synchronous message passing model focuses on the analysis of round complexity of a distributed problem (the number of rounds required to solve it). The HYBRID model restricts which nodes may communicate in a given round and to what extent.

Definition 3 *(*HYBRID *model, cf. [4])*. The *local* communication mode is modeled as a connected graph, in which each node is initially aware of its neighbors and is allowed to send a message of size λ bits to each neighbor in each round. In the *global* communication mode, each round each node may send or receive γ bits to/from every other node that can be addressed with its ID in $[n]$ in case it is known. If any restrictions are violated in a given round, an arbitrary subset of messages is dropped.

In this paper, we consider a weak form of the HYBRID model, which sets $\lambda \in O(\log n)$ and $\gamma \in O(\log^2 n)$, which corresponds to the combination of the classic distributed models CONGEST[6] as local mode, and NODE CAPACITATED CLIQUE (NCC)[7] as global mode. Note that while it might appear that more global than local communication is allowed in our model, the local network allows each node to exchange messages with each neighbor (of which there could be $\Theta(n)$), which is not possible in the global network.

The distributed problem of computing routing schemes on the local communication graph is an excellent fit for the HYBRID model, since the problem is known to require $\widetilde{\Omega}(n)$ rounds of communication (where $\widetilde{O}, \widetilde{\Omega}$ hides $\mathrm{polylog}\, n$ factors) if only *either* communication via the local mode *or* the global mode is permitted, see [14]. The lower bound for the local communication mode holds even if the input graph is a path and even for unbounded local communication.

It is natural to wonder if adding a modest amount of global communication on top of a local network significantly improves the required number of communication rounds to establish a routing scheme in the network. This question was recently answered

5 Minimizing hop-distance in a unit-disc-graph essentially minimizes the Euclidean distance that the path covers, thus graph weights are not required.

6 Some previous papers that consider hybrid models use $\lambda = \infty$, i.e., the LOCAL model as local mode.

7 Our methods also work for to the stricter NCC_0 model as the global network, where only incident nodes in the local network and those that have been introduced, can communicate globally.

positively by [14], where it was shown that routing schemes with small labels can be computed in $\widetilde{O}(n^{1/3})$ rounds for *arbitrary* local graphs. However, [14] also shows that a polynomial number of rounds is required to solve the problem even approximately (in particular, an exact solution with labels up to size $O(n^{2/3})$ requires $\widetilde{\Omega}(n^{1/3})$ rounds).

To mitigate this lower bound, [9] considers local communication networks that are restricted to certain interesting classes of graphs for which they can compute routing schemes in just $O(\log n)$ rounds. In this article we will continue this line of work and consider local communication graphs that are unit-disk graphs (UDGs). Such a UDG $G = (V, E)$ satisfies the property that nodes are embedded in the Euclidean plane and are connected iff they are at distance at most 1. UDGs appear as a natural consequence of range limitations in wireless networks, for instance they are induced by the so called SINR model (see Section 1.3, last paragraph) and have been extensively studied as a model capturing wireless ad-hoc connections (see, [5,6,9,15–17], all of which handle routing schemes in UDGs).

In [9] it was shown that the nodes of a UDG can together simulate a much simpler *grid graph* structure with constant overhead in round complexity, such that the connectivity, the hole-freeness, and the hop distance up to a constant factor are preserved. Furthermore, [9] shows that a routing scheme on a grid graph can efficiently be transformed into a routing scheme for the underlying UDG, which introduces only a constant overhead on label size and local routing information and takes only a constant number of additional rounds.

This allows them to consider the much simpler grid graphs for the computation of routing schemes to generate good routing schemes for UDGs. However, their actual algorithm for computing a routing scheme for a grid graph comes with a caveat: it works only for grid graphs *without holes*, which, loosely speaking, are points in the grid without nodes on them that are enclosed by a cycle in the grid graph (more formally in Section 2), which implies routing schemes only for UDGs without "radio holes", which roughly correspond to areas enclosed by the UDG that are not covered by nodes (see [9] for the formal definition of radio holes in UDGs).

In this work we extend the solution to UDGs with such radio holes. Note that the transformations given in [9] from UDGs to grid graphs work even if there are holes, which essentially allows us to focus on the computation routing schemes for grid graphs with holes, which gives the same for arbitrary UDGs, which have a set $\mathcal{H}$ of radio holes (formally defined later). Our algorithm is asymptotically as efficient (in computation time, and label and table sizes of the resulting routing scheme) as the algorithm of [9] if the number of holes $|\mathcal{H}|$ is small.

We stress that it is *significantly* more challenging to compute routing schemes on grid graphs with such holes than on grid graphs without holes. In the paper of Coy et al. ([9]) the authors heavily exploit the property that any simple st-path can be deformed into any other simple st-path: this is *not true* in our setting. In particular, it is easy to come up with examples of grid graphs with $|\mathcal{H}|$ holes for which there are at least $2^{|\mathcal{H}|}$ "reasonable" classes of st-paths (intuitively: paths which do not completely encircle or spiral around holes) which cannot be deformed into each other. Worse still, we can make all-but-one of these classes of paths almost arbitrarily long, and so we cannot just consider one arbitrary class of st-paths and obtain an approximate shortest path. We must determine the class in which an exact shortest st-path lies, which seems to require a sparsifying structure that scales in complexity with the number of holes.

1.1. Contributions

Our main contribution is the extension of the result presented in Coy et al., [9], which assumes that no holes are present making the problem much simpler.

Theorem 4 *(Main result for grid graphs). Given a grid graph Γ with a set $\mathcal{H}$ of holes (defined in Section 2), we can compute an exact routing scheme for Γ in $O(|\mathcal{H}|^2 + \log n)$ rounds in the* HYBRID *model (notably, even* NCC_0 *suffices). The labels are of size $O(\log n)$; nodes need to locally store $O(|\mathcal{H}|^2 \log n)$ bits.*

Note that, since grid graphs have constant degree, the local network can be simulated using the NCC_0 model. Therefore, in the theorem above and in all our subsequent claims about grid graphs, the use of HYBRID can be replaced with the weaker NCC_0 model. Although this is interesting, we consider computation of routing schemes solely in the global network as an artificial problem: the fact that we are computing a routing scheme for a local network suggests that this network can be used to help construct it. Furthermore, the local network of the HYBRID model is required by [9] to efficiently transform a unit disc graph into a sparsifying grid graph structure that approximates it well (we briefly summarize this in Section 2). This result leads to the following corollary:

Corollary 5 *(Main result for UDGs). The routing scheme of Theorem 4 can be transformed into a routing scheme which yields constant-stretch shortest paths for unit-disk graphs in the* HYBRID *model. Round complexity, label size, and local storage are asymptotically the same.*

Beyond UDGs the corollary above extends to any graph class in which we can efficiently simulate a grid graph structure with small approximation error. We believe that several of our technical contributions are of independent interest. Our main technical contribution is a decomposition of a grid graph into *simple, path-convex* regions which have useful properties for routing. We also provide a small skeleton structure of the UDG called a *landmark graph* such that shortest paths in the landmark graph are topologically the same (i.e., circumnavigate holes in the same way) as in the original graph, which may be useful when solving shortest paths on grid graphs and UDGs in HYBRID or for similar problems in other models of computation. Furthermore we give an $O(\log n)$ round algorithm for solving SSSP exactly in simple grid graphs and an $O(\log n)$ round algorithm for finding the distance from every node in a simple grid graph to a portal.

1.2. Overview

For an end-to-end overview of our approach, following an example graph, see Appendix A.

In Section 3, we give an algorithm that solves SSSP in hole-free grid graphs (see Definition 7 and Definition 15) in $O(\log n)$ rounds deterministically. We create two helper graphs, one representing its horizontal connections and one for the vertical connections, and we prove that SSSP solutions for these two graphs suffice to solve SSSP in the original grid graph. Further, we present a scheme that allows us to compute shortest paths of nodes to *portal* sets (i.e., a connected straight path in the grid graph).

In Section 4, we show that a grid graph can be decomposed into regions such that each region is simple (contains no holes) and path-convex (for any nodes s, t in a region, a shortest st-path lies entirely in their region), and with overlap only in portals (specifically called gates). We do this in stages: we decompose the graph into simple regions; we decompose these regions further so that they are "tunnel" shaped (i.e., bounded by at most 2 gates); and finally we decompose them further still to ensure they are path-convex. We show that we can perform this decomposition quickly and the number of resulting regions is low.

In Section 5, we give a skeleton structure on this decomposition to facilitate routing. We mark vertices which lie on many shortest paths as *landmarks*. We connect some landmarks in the same region to each other to form a *landmark graph*, and show it can be computed quickly. Finally, we prove that for any grid nodes s and t, a shortest st-path goes through the same regions as the shortest path from a landmark near s to a landmark near t.

In Section 6, we combine our results to construct the routing scheme, first showing that each node can efficiently learn the whole landmark graph. We then show that the region which a packet should enter next can be determined by combining information about the landmark graph, the target node's label, and local information about the region the message is currently in. Nodes forward received packets to their neighbor which is closest to the packet's next region. We repeat this until the packet arrives at the target region, switching then to the routing scheme of [9].

1.3. Related work

Shortest Paths in Hybrid Networks. Previous work in the HYBRID model has mostly focused on shortest path problems [4,13,11,7,1]. In the k-sources shortest path (k-SSP) problem, all nodes must learn their distance in the (weighted) local network to a set of k sources. Particular focus has been given to the all-pairs (APSP, $k = n$) and single-sources (SSSP, $k = 1$) shortest-path problems. Note that solving the APSP problem gives a solution to the routing scheme problem. The complexity of APSP is essentially settled: [4,13] give an algorithm taking $\widetilde{O}(\sqrt{n})$ rounds, and this matches a lower bound of $\widetilde{\Omega}(\sqrt{k})$ to solve k-SSP even for polynomial approximations. This lower bound even matches a deterministic algorithm due to [1], although only with an approximation factor of $O(\frac{\log n}{\log\log n})$.[8] For k-SSP the $\widetilde{\Omega}(\sqrt{k})$ lower bound has been matched by [7] with a constant stretch algorithm, given sufficiently large k (roughly $k \in \Omega(n^{2/3})$).[8] Whether there are any $\widetilde{O}(\sqrt{k})$ round k-SSP algorithms on general graphs for $1 < k < n^{2/3}$ remains open. The state-of-the-art algorithm for exact SSSP is provided by [7] and takes $O(n^{1/3})$ rounds.[8] A recent result by [21], which solves SSSP by $\widetilde{O}(1)$ applications of an instruction set called "minor-aggregation" when given access to an oracle that solves the so called Eulerian Orientation problem, can be adapted for a $(1+\varepsilon)$ approximation of SSSP in $\widetilde{O}(1)$ rounds, as was shown in [23].[8] An *exact, deterministic* solution for SSSP in $O(\log n)$ rounds has been achieved on specific classes of graphs (e.g. cactus graphs, which includes trees) by [11]. Another exact SSSP algorithm that takes $\widetilde{O}(\sqrt{SPD})$ rounds (where SPD is the shortest-path-diameter of the local graph) is provided by [4].

Routing Schemes in Distributed Networks. Our work builds on [9], in which they show how to compute a routing scheme for a UDG, by computing a routing scheme for a corresponding grid graph (see Section 2). Their approach requires a simplifying assumption: the grid graph needs to be free of holes (formally defined in the next section), and this imposes a similar restriction on the underlying UDG. We remove that assumption in this work. In a recent article, [14] considers computing routing schemes in the HYBRID model on general graphs: they show that in $\widetilde{O}(n^{1/3})$ rounds one can compute exact routing schemes with labels of size $\widetilde{O}(n^{2/3})$ bits, or constant stretch approximations with smaller labels of $O(\log n)$ bits. Interestingly, [14] also gives lower bounds: they show that it takes $\widetilde{\Omega}(n^{1/3})$ rounds to compute exact routing schemes that hold for relabelings of size $O(n^{2/3})$ and on unweighted graphs. They also give polynomial lower bounds for constant approximations on weighted graphs, implying that in order to overcome this lower bound in round complexity and label size, the restriction to a class of graphs is necessary! The distributed round complexity of computing routing schemes was also considered in the CONGEST model. In general, it takes $\widetilde{\Omega}(\sqrt{n}+D)$ rounds to solve the problem [22] (where D is the graph diameter). This was nearly matched in a series of algorithmic results [18,19,10], for example, [10] gives a solution with stretch $O(k)$, routing tables of size $\widetilde{O}(n^{1/k})$, labels of size $\widetilde{O}(k)$ in $O\left(n^{1/2+1/k}+D_G\right)\cdot n^{o(1)}$ rounds.

SINR Model as Local Network. A well-accepted model for communication in wireless networks is the **S**ignal to **I**nterference and **N**oise **R**atio model, where a message is only received if the signal strength over the distance between sender and receiver is larger than the interference by other nodes plus some ambient noise. The latter implies a maximum range at which messages can be received, thus nodes that can communicate at zero interference induce a UDG. A result of Halldórsson and Tonoyan [12] connects the SINR model to the CONGEST model. They show that the SINR model allows to simulate the CONGEST model on a locally sparse backbone structure which inherits the UDG property from the SINR model. Thus, by using the SINR model as local mode with our

[8] These results are in the more powerful hybrid combination of LOCAL and NCC.

global mode (NCC_0), we obtain a routing scheme on the backbone. This solution can be extended to nodes outside the backbone, by leveraging the "download" routine provided in [12].

2. Unit disk graphs and grid graphs

In this section we introduce concepts from [9] with some additional concepts related to holes, which we require in many of the subsequent sections. We consider the class of *Unit Disk Graphs* (UDG) defined as follows:

Definition 6 *(Unit Disk Graphs).* $G = (V, E)$ is a UDG if each node $v \in V$ is associated with a unique point in $\mathbb{R}^2$ and $u, v \in V$: $\{u, v\} \in E$ iff $\|u-v\|_2 \leq 1$.

Grid graphs can be seen as a sparsifying structure for UDGs which can be easily simulated while preserving certain geometric properties and significantly simplifying the construction of algorithms for the original UDG (cf., [9], more on that further below).

Definition 7 *(Grid graphs).* A *grid graph* $\Gamma = (V_\Gamma, E_\Gamma)$ is a graph such that the vertices V uniquely correspond to points on a square grid $\mathbb{Z}^2$. Two vertices are connected by an edge in E_Γ iff their corresponding points on the grid are horizontally or vertically adjacent.

2.1. Grid graph abstractions of unit disk graphs

We further explain the correspondence between UDGs and grid graphs. As was shown in [9], one can compute and simulate a grid-graph abstraction $\Gamma = (V_\Gamma, E_\Gamma)$ of any input UDG using only local communication in $O(1)$ rounds. In this simulated grid graph Γ, each grid node is represented with a close by node in the UDG. Any UDG node has to represent only a constant number of grid nodes (and can thus simulate all grid nodes it represents).

Theorem 8 *(cf. [9]). Let $G = (V, E)$ be a UDG. We can compute a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ in $O(1)$ rounds, such that each grid node $g \in V_\Gamma$ is represented by some node $v \in V$ with $\|v-g\|_2 \leq 1$.* Every *node $v \in V$ has a representative $r \in V$ of some grid node $g \in V_\Gamma$ with* $\mathrm{hop}_G(v, r) \leq 1$. *A round of the* HYBRID *model on Γ can be simulated by the set of representatives in V in $O(1)$ rounds, and for any $s, t \in V$ with $\{s, t\} \notin E$ we have close-by representatives $g_s, g_r \in V_\Gamma$ such that from a shortest path $\Pi_\Gamma(g_s, g_t)$ from g_s to g_t in Γ we can (locally and in $O(1)$ rounds) construct a path $\Pi_G(s, t)$ in G with $|\Pi_G(s, t)| \leq 36 \cdot \mathrm{dist}_G(s, t)$.*

The theorem above means that approximate paths with decent stretch on the UDG can be constructed from shortest paths on the grid graph abstraction Γ. In particular, this implies that any constant stretch routing scheme on Γ also gives a constant stretch routing scheme on the underlying UDG. Therefore, in order to obtain constant stretch routing schemes on UDGs it is sufficient to consider the problem on the (easier) class of grid graphs.

Theorem 9 *(cf. [9]). Any algorithm that computes routing schemes with stretch s, labels of at most x and local routing information at most y bits on* any grid graph *in z rounds (where x, y, z depend on n) implies an algorithm to compute a routing scheme with stretch $36 \cdot s$ on any UDG with labels of $O(x)$ bits, local routing information of $O(y)$ bits in $O(z)$ rounds.*

2.2. Geometric structure of grid graphs

In order to describe the geometric structure of UDGs $G = (V, E)$ and grid graphs $\Gamma = (V_\Gamma, E_\Gamma)$ (which are also UDGs) we require some basic notation. We start by defining paths and distances in grid graphs.

Definition 10 *(Paths).* Let $G = (V, E)$ be a graph. We define a path $\Pi \subseteq E$ as a subset of edges that form an sequence of incident edges in G. A uv-path Π is a path where the first and last node in the sequence of edges are u, v respectively. For a uv-path Π and a vw-path Π' we denote the *composite path* from u to w obtained by concatenating Π and Π' as $\Pi \circ \Pi'$. By $|\Pi|$ we denote the number of edges (or *hops*) of a path Π. We will occasionally identify simple paths by the set of endpoints of its edges.

Definition 11 *(Distances).* The *hop-distance* between two nodes $u, v \in V$ is defined as $d_G(u, v) := \min_{u\text{-}v\text{-path } \Pi} |\Pi|$. We generalize distances for subsets of nodes $S \subseteq V$ as follows $d_G(u, S) := \min_{v \in S} d_G(u, v)$. Note that the hop distance is equal to the distance as all our edges have unit weight, hence we use the two interchangeably. Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a grid graph. Let Π_x be the set of horizontal edges on some path Π. Then $d_{\Gamma,x}(u, v) := \min_{u\text{-}v\text{-path } \Pi} |\Pi_x|$ is the horizontal distance between u and v. Analogously we define the vertical distance $d_{\Gamma,y}$. We will drop the subscript G or Γ when it is clear from the context.

To analyze grid graphs we define some geometric structures, starting with *portals* [9].

Definition 12 *(Portals).* Let E_v be the set of vertical edges of some grid graph $\Gamma = (V_\Gamma, E_\Gamma)$. Then the vertical portals are the connected components of the sub graph $(V_\Gamma, E_{\Gamma,v})$. We say that portals p_1 and p_2 are *adjacent* if p_1 is connected by a (horizontal) edge in E to a node in p_2. Horizontal portals are based on the set of horizontal edges $E_{\Gamma,h}$ and the corresponding terms are defined analogously.

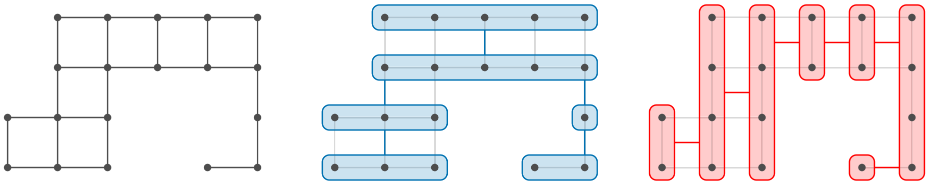


Fig. 1. A simple grid graph (left) and the corresponding horizontal (center) and vertical (right) portal graphs.

Next, we introduce the concept of *holes* in a grid graph. Intuitively, the area that is covered by a grid graph can be described by "filling in" the grid cells (those are unit squares $[a, a+1] \times [b, b+1]$ with $a, b \in \mathbb{Z}$) where all four corner nodes represent nodes in Γ as well as all edges of Γ. The holes are the connected areas in the Euclidean plane which are not "filled".

Definition 13. Given a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$. We define $S_\Gamma \subseteq \mathbb{R}^2$ as

$$S_\Gamma^{\text{cells}} := \{[a, a+1] \times [b, b+1] \mid \text{nodes in } V_\Gamma \text{ at } (a,b), (a+1,b), (a,b+1), (a+1,b+1)\}$$
$$S_\Gamma^{\text{edges}} := \{x \cdot u + (1-x) \cdot v \mid \{u, v\} \in E_\Gamma, x \in [0,1]\}$$
$$S_\Gamma := S_\Gamma^{\text{cells}} \cup S_\Gamma^{\text{edges}}$$

Then all *maximal connected components* in the complement $\mathbb{R}^2 \setminus S_\Gamma$ constitute the set of holes induced by Γ. We define the *set of inner holes* $\mathcal{H}$ as all *bounded*, maximal connected components in the complement $\mathbb{R}^2 \setminus S_\Gamma$. As the number of nodes in Γ is finite, there is exactly one maximal connected component in $\mathbb{R}^2 \setminus S_\Gamma$ that is unbounded, which we call the *outer hole*.

We now give some additional definitions pertaining to holes.

Definition 14. Given a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ with inner holes $\mathcal{H}$. Then the boundary of some $H \in \mathcal{H}$ is the set of nodes whose corresponding points in the plane are on the geometric boundary ∂H of H. We say that a node is incident to H if it is part of its boundary. Similarly a node set (like a portal) is incident to H if it contains an incident node.

Definition 15. A grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ with $\mathcal{H} = \emptyset$ is called simple.

Finally, [9] gives an efficient algorithm for the special case of computing routing schemes in *simple* grid graphs. This implies a constant stretch routing scheme for UDGs without holes by Theorem 9.

Theorem 16 *(cf. [9]). An exact routing scheme for a simple grid graph Γ using node labels and local space of $O(\log n)$ bits can be computed in $O(\log n)$ rounds in the* HYBRID *model.*

3. Closest point computations in simple grid graphs

In this section, we present two important subroutines for our main results. Firstly, an $\mathcal{O}(\log n)$ Single Source Shortest Paths problem (SSSP) algorithm for simple grid graphs. The problem input is given by the grid graph in a distributed way (each node initially knows only its incident edges) and it is solved as soon all nodes know their distance to a dedicated source node. Secondly, we use this algorithm to solve a problem we call the *Single Portal Shortest Path* problem (SPSP). Here, we want to compute the distance of each node to its corresponding closest point some dedicated portal (Definition 12). We start with the following theorem.

Theorem 17. *Given a simple grid graph Γ, we can solve SSSP on Γ in $\mathcal{O}(\log n)$ rounds.*

To achieve the logarithmic runtime, we split the SSSP problem into two subproblems: Horizontal SSSP, which only considers the horizontal steps a path makes and Vertical SSSP, which only considers the vertical steps a path makes. To be able to efficently solve these problems, we introduce horizontal and vertical *portal graphs*, which are depicted in Fig. 1 as well.

Definition 18. Given a grid graph Γ, we define the *vertical portal graph* $\mathcal{P}_v$ to be the graph with vertices corresponding to the vertical portals of Γ. Two vertices in $\mathcal{P}_v$ are connected by an edge iff their corresponding portals are adjacent (i.e., connected with a horizontal edge) in Γ. The horizontal portal graph $\mathcal{P}_h$ is defined analogously.

As we will see later, the distances in $\mathcal{P}_v$ correspond to horizontal distances taken by paths and those in $\mathcal{P}_h$ correspond to vertical distances, respectively. This motivates us to run SSSP on these graphs and combine the solutions.

Lemma 19. *If a grid graph Γ is simple, both its vertical portal graph $\mathcal{P}_v$ and its horizontal portal graph $\mathcal{P}_h$ are trees.*

Proof. For simplicity, we state the proof for the vertical portal graph $\mathcal{P}_v$, the claim for $\mathcal{P}_h$ follows by symmetry. We prove the claim by contradiction. Assume there was a cycle in $\mathcal{P}_v$. This cycle would correspond to a set of portals in Γ, such that two adjacent nodes in the cycle correspond to two adjacent portals in Γ. We pick two portals from this set that have the same x coordinate arbitrarily. Such a pair must exist due to the existence of a cycle in $\mathcal{P}_v$. As the portals are maximal vertically connected components by definition, there is some space enclosed between them. Further, as the portals are part of a cycle in $\mathcal{P}_v$, they are connected by two paths in P_v. Starting from the northern portal of the pair we picked, one of those paths surround the area between the pair of portals in a clockwise direction, while the other surrounds it in a counter-clockwise direction. Together, they enclose it and it must correspond to the hole, which contradicts simplicity. □

This allows us to employ a slightly modified version of the hybrid SSSP algorithm for trees presented in [11, Theorem 6].

Lemma 20. *Given a simple grid graph G, we can solve SSSP on its vertical and horizontal portal graph in time $\mathcal{O}(\log n)$.*

Proof. The algorithm of [11] can be used to compute a SSSP solution in trees in time $\mathcal{O}(\log n)$ in the HYBRID model. Therefore, it suffices to construct a subgraph of Γ that maintains the tree structure and the distances of $\mathcal{P}_v$. It consists of all nodes of Γ and all vertical edges. For each pair of adjacent portals in $\mathcal{P}_v$ we add a *single* arbitrary horizontal edge connecting the two portals. Note that identifying adjacent pairs of portals and one edge that connects those can be done in $O(\log n)$ rounds in the HYBRID model. We achieve this, by first employing *pointer jumping* on the portal to aggregate and broadcast (Lemma 100) the minimum identifier, acting as a portal identifier. Each node informs its horizontal neighbors about its own portal's identifier. Next the nodes communicate with their vertical neighbors, which identifiers they have received this way. This allows the bottommost node neighboring a specific portal to mark its edge in the corresponding direction as the edge connecting the two portals. As vertical portals are maximal vertically connected subgraphs, this selection is unique. To preserve the distances in $\mathcal{P}_v$ we simply assign a weight of 1 to all horizontal 0 to all vertical edges. The procedure for $\mathcal{P}_h$ works analogously (exchange the words horizontal and vertical). □

Next, we combine the SSSP solutions for $\mathcal{P}_v$ and $\mathcal{P}_h$ into a SSSP solution for Γ.

Lemma 21. *Given a simple grid graph Γ, a starting node s and SSSP solutions for its vertical and horizontal portal graphs $\mathcal{P}_v$ and $\mathcal{P}_h$ starting from the corresponding portals containing s, we can compute an SSSP solution for Γ starting from s in time $\mathcal{O}(\log n)$.*

Proof. Each node computes its distance to s in Γ by adding the two distances obtained by the SSSP computations in $\mathcal{P}_v$ and $\mathcal{P}_h$. To show the correctness of these distances, we perform an inductive argument along the shortest paths. Let $w \in V$ be an arbitrary node and denote its computed vertical distance with $d_v(w)$ and its computed horizontal distance with $d_h(w)$. Consider a shortest path $(s = w_1, \ldots, w_\ell = w)$ from s to w. Since both portals containing s correspond to the starting nodes of the SSSP executions in $\mathcal{P}_v$ and $\mathcal{P}_h$, both distances s obtained by the procedure described above are 0, i.e., we have $d(s) = d_v(s) + d_h(s) = 0$, which is our base case. Next, we argue that $d(w_n) = d_v(w_n) + d_h(w_n)$ implies $d(w_{n+1}) = d_v(w_{n+1}) + d_h(w_{n+1})$, for $n + 1 \leq \ell$. Note that node w_{n+1} is node w_n's successor in the shortest path and further away from s than it by construction. If it is vertically adjacent to w_n, we have $d_v(w_{n+1}) = d_v(w_n) + 1$ and $d_h(w_{n+1}) = d_h(w_n)$. Naturally, $d(w_{n+1}) = d(w_n) + 1$ also holds. Therefore, we have $d(w_{n+1}) = d(w_n) + 1 = \left(d_v(w_n) + 1\right) + d_h(w_n) = d_v(w_{n+1}) + d_h(w_{n+1})$. The case where the shortest path makes a horizontal step is analogous.

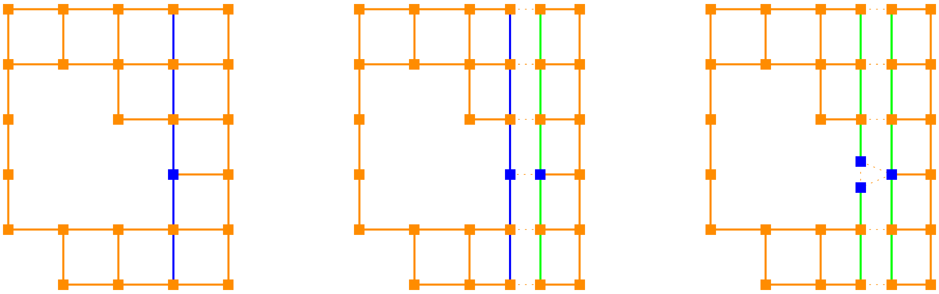
Since the distance values are correct and the predecessor pointers are picked to point to any neighbor closer to s, their correctness immediately follows. □

Using Lemma 20, we combine the SSSP solutions for the portal graphs $\mathcal{P}_v$ and $\mathcal{P}_h$ to compute an SSSP solution for the grid graph Γ, which allows us to conclude Theorem 17. We can extend Theorem 17 to compute the distance of each node in a grid graph to a dedicated portal P. We refer to this problem as the *Single Portal Shortest Path* (SPSP) problem. To solve the SPSP problem, we pick an arbitrary node of P as the starting node of an SSSP execution and consider the weights of all edges of P to be zero during that execution.

Corollary 22. *Given a simple grid graph Γ and a portal P in Γ. We can compute the distances of every vertex in Γ to P and its neighbor that is closest to P in time $\mathcal{O}(\log n)$.*

We conclude this section with an observation on how to compute a shortest path tree using the distance values of an SSSP computation or an SPSP computation.

Observation 23 *(Shortest path tree).* After computing SSSP (or SPSP), by having each node pick a predecessor pointer arbitrarily from its neighbors with minimal distance to s we can establish the corresponding shortest path tree in a distributed sense (this step takes just $\mathcal{O}(1)$ time using local edges).



(a) Before split at blue portal P node v

(b) After split at P; before split at v

(c) After split at blue node v

Fig. 2. Illustration of the process by which portals are split at a boundary node on a portal.

4. Path-convex region decomposition

In the following section we establish that grid graphs can be partitioned into comparatively few sets of nodes (with some overlap at the borders) called *regions* that are simple, i.e., they have no holes, and path-convex, which we define as follows.

Definition 24 *(Path convexity).* Let $\Gamma = (V, E)$ be a grid graph and let $R \subseteq V$. Then R is called *path-convex* if for any pair $u, v \in R$ there is a shortest uv-path contained completely within that region.[9]

Definition 25 *(Region decomposition).* Let $\Gamma = (V, E)$ be a grid graph. A *region decomposition* of Γ is a family of node sets $\mathcal{R} = \{R_1, \ldots, R_\ell\}$ (regions) with $V = R_1 \cup \cdots \cup R_\ell$, where each region $R \in \mathcal{R}$ induces a connected grid graph (cf., Definition 7, we mostly use the term region in that sense of a grid graph). With this in mind the definitions of simple (cf., Definition 13) and path-convex apply for regions $R \in \mathcal{R}$ (cf., Definition 24). We call the decomposition $\mathcal{R}$ simple and path-convex if all regions in $\mathcal{R}$ are simple and path-convex (cf., Definition 24).

The construction breaks down into three main steps. In the first step, we decompose Γ into simple regions. Second, we break those regions up further into "tunnels", which are defined by the property that they overlap in at most two portals (called gates) with all of their neighboring regions. In the final step we show that such tunnels have crucial properties that allow us to make them convex by subdividing them a constant number of times. Ultimately, we prove the following theorem.

Theorem 26. *For any grid graph Γ, a simple, path-convex region decomposition of size $O(|\mathcal{H}|)$ can be computed in $O(|\mathcal{H}| + \log n)$ rounds in the* HYBRID *model.*

Splitting Operations. For the computation of a region decomposition with the desired properties we will frequently split Γ at suitable portals and, in case this disconnects parts of Γ consider each connected component as separate region. For the construction we require that nodes on the portals are contained in both regions bordering it. Thus regions may intersect at portals.

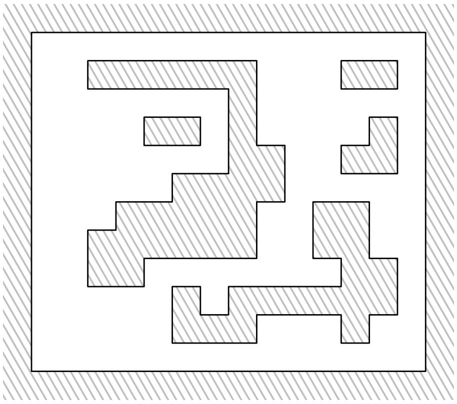
The most basic such operation is to split a grid graph (or a region) at a given portal that intersects holes only at its endpoints, see Fig. 2. In this case, each node will simulate two copies of itself, a "right copy" which has no left neighbor and a "left copy" which has no right neighbor. This establishes a new grid graph where nodes that have been split will act in the role of the nodes they simulate only, which blocks paths through the splitting portal and might disconnect Γ.

If such a splitting portal touches the boundary of a hole not only at its endpoints, we usually further split at a boundary node. In particular, we split the simulated boundary node that is on the "same side" as the hole (say the left copy if the hole is left of the portal) into a "top" and "bottom" copy, which do not have a bottom or top neighbor, respectively. This can be used to break up cycles around a hole, thereby making the resulting regions simple. We describe such splitting operations in detail and prove that they can be conducted efficiently in Appendix C.3, Definition 105 and Lemma 106.

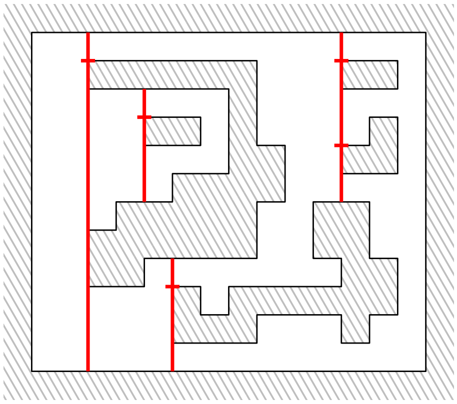
The region decomposition in the sense of Definition 25 is given implicitly by the connected components in the grid graph formed by the simulated nodes after splitting at a portal or a node according to Definition 105. As a consequence of these splits, during intermediate steps, we occasionally end up with two (unconnected) copies of a grid node within the region that occupy the same point in the plane. Thus, for some intermediate steps we have to slightly generalize our notion of grid graphs to accommodate for such node copies, but after the complete decomposition, all will end up in different regions.

Since we split at portals, we obtain the property that regions overlap only in node sets that form portals (i.e., maximal vertically connected components). To distinguish those from ordinary portals (Definition 12), we call these *gates*, which we formally define as follows.

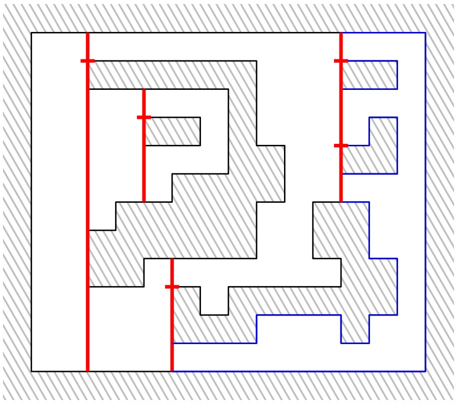
[9] This leans on the standard definition of convex node sets in the context of graphs. A node set is called convex if *all* (instead of just one) shortest paths between any pairs of nodes are within the set.



(a) A stylised grid graph with holes, where white areas are occupied by grid nodes and shaded areas correspond to holes (see Definition 13).



(b) Red lines mark portals through a leftmost node of some hole at which a split occurs. Further splits take place at a dedicated leftmost node of each hole (details in Definition 105).



(c) Blue lines in the bottom right region mark the walls of the region, see Definition 28.

Fig. 3. Decomposition of grid graph into simple regions by splitting at certain portals & nodes.

Definition 27 *(Gate)*. Let Γ be a (simulated) grid graph after splitting at some vertical portal (according to Appendix C.3 Definition 105) has been conducted (the case with a horizontal portal is symmetric). Then a node on that portal is called a *gate node*. A maximal vertically or horizontally connected component of gate nodes within a given region of the resulting decomposition after the split is called a *gate*.

In the subsequent sections, gates will be vertical until the last step. Furthermore, we classify connected segments of the boundary nodes of some region, which are not on gates. We will refer to these as *walls*, formally defined as follows.

Definition 28 *(Wall)*. Let $\mathcal{R}$ be a region decomposition of Γ established by conducting splitting operations as defined in Appendix C.3 Definition 105. For $R \in \mathcal{R}$ we denote *connected* segments of hole boundary nodes in R (see Definition 14) that are *not* gate nodes (cf. Definition 27) as *walls*, see Fig. 3. Note that walls and gates alternate on the boundary of R. Also note that different wall segments might share nodes in case a region is connected by a single grid edge.

4.1. Decomposition into simple regions

The first step is to split our grid graph into *simple* regions, i.e., regions without holes. The following procedure can achieve this goal as described in the following definition on an abstract level along the lines of the previous section. Implementation details of the construction are given separately in Appendix C.3.

Definition 29 *(Splitting Γ into simple regions)*. For each $H \in \mathcal{H}$ we do the following (in parallel, see Appendix C.2 on how to identify holes boundaries). Determine the leftmost node v_H on the boundary of H (make v_H unique by choosing the northernmost among leftmost boundary-nodes, see Appendix C.1 on how to identify v_H). Let P_H be the unique vertical portal with $v_H \in P_H$. We conduct splits at P_H and v_H (see Definition 105 case two and three). In general, P_H might contain leftmost nodes of boundaries of different inner holes. In that case we handle these holes together by conducting a vertical split at the northernmost node of each, as described in the third case of Definition 105. Fig. 3 gives an example.

The connected components that result from the above construction form regions that are simple. The idea is that each such split will make the grid graph at the portal P_H horizontally impassable and at v_H vertically impassable. Roughly speaking, the split at a portal and a node on the portal and hole boundary create a “thin” hole such that in the resulting graph two holes “merge”, and become a single hole. This decreases the overall number of *inner* holes by at least one.

However, splitting portals this way leads to “infinitely thin hole sections” which does not fit with our definition of holes (Definition 13). Therefore, we use as an auxiliary construction a grid graph with finer granularity $\frac{1}{3}$. This construction has the same topological properties in particular pertaining the number and shape of holes per Definition 13. We modify this finer grid graph such that it becomes simple and prove this by essentially showing that all inner holes “merge” with the outer hole.

We then “round” the simple grid graph of granularity $\frac{1}{3}$ back to integer granularity and show that the result equals *exactly* the one from the construction above. Since the rounding can not create holes, the result from the construction can not have any holes either. In the following, we give the detailed work on this, in particular we show the following two lemmas:

Lemma 30 *(Correctness of simple decomposition). The construction in Definition 29 decomposes Γ into at most $|\mathcal{H}|+1$ simple regions.*

Lemma 31 *(Computing the simple decomposition). A* simple *region decomposition can be computed for any grid graph Γ in $O(\log n)$ rounds in the* HYBRID *model.*

For the proof, we start with the Definition of the grid graph of granularity $\frac{1}{3}$.

Definition 32 *($\frac{1}{3}$-Grid transformation).* We transform a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ into a grid graph $\Gamma' = (V_{\Gamma'}, E_{\Gamma'})$ that has granularity $\frac{1}{3}$ as follows. For each node $v \in V_\Gamma$ at position $(a,b) \in \mathbb{Z}^2$, we create nine nodes in V'_Γ at positions $(a+x, b+y)$ with $x, y \in \{-\frac{1}{3}, 0, \frac{1}{3}\}$. We say that $u', v' \in V_{\Gamma'}$ belong to the group of $v \in V_\Gamma$ if they were created from v (i.e., rounding their coordinates to the closest integer gives the position of v). For each pair $u', v' \in V_\Gamma$ we add an edge $\{u', v'\}$ to $E_{\Gamma'}$ between two nodes $u', v' \in V_{\Gamma'}$ if $\|u' - v'\|_2 = \frac{1}{3}$. An edge $\{u', v'\} \in E_{\Gamma'}$ is assigned weight 0 if u', v' belong to the same group, else weight 1.

We will decompose the transformed graph $\Gamma' = (V_{\Gamma'}, E_{\Gamma'})$ with a procedure similar to the one proposed for Γ and make an argument that it is simple, i.e., no region has any holes. Afterwards we “round” the nodes of Γ' back to the integer grid and argue a) that the region decomposition is still simple and b) that the final result is exactly the same as doing the procedure in Definition 29 for Γ. The reverse Transformation works as follows.

Definition 33 *(Reverse transformation).* We transform a grid graph $\Gamma' = (V_{\Gamma'}, E_{\Gamma'})$ that has granularity $\frac{1}{3}$ back into a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ with granularity 1. Note that the resulting graph may have nodes sharing the same coordinate. For $p \in \mathbb{Z}^2$ let V_p be the set of nodes in $V_{\Gamma'}$ whose coordinates are p when rounded to the closest integer. Let $\tilde{u}, \tilde{v}$ be two connected components in V_p. We create a new node V_Γ at p, for each connected component (this is where the nodes occupying the same coordinate can occur). Finally we define the edges of Γ. Let $p_1, p_2 \in \mathbb{Z}^2$ and let $\tilde{u}$, $\tilde{v}$ be connected components in V_{p_1} and V_{p_2}, respectively, i.e., $\tilde{u}, \tilde{v} \in V_\Gamma$. We add an edge $\{\tilde{u}, \tilde{v}\}$ between nodes $\tilde{u}, \tilde{v} \in V_\Gamma$, if there are two nodes $u \in \tilde{u}$ and $v \in \tilde{v}$ with $\{u, v\} \in E_{\Gamma'}$, i.e., u, v share an edge in Γ'.

The next definition gives an iterative construction on how to make Γ' simple.

Definition 34 *(Construction to remove H in Γ').* Let Γ' be the $\frac{1}{3}$ Transformation of a grid graph Γ. For each inner hole $H \in \mathcal{H}$ of Γ we repeat the following. Let v_H be the splitting node, i.e., the chosen leftmost node on the boundary of H. Let $v'_H \in V_{\Gamma'}$ be the middle node (the one with coordinates in $\mathbb{Z}$) of the 9 nodes created from v_H in Γ'. Let P'_H be the unique portal in Γ' with $v'_H \in P'_H$. We remove all nodes on P'_H from Γ' as well as the node right of v'_H (which is at the boundary of some hole in Γ'). If there is a splitting node $v_{H'}$ of another hole H' with P_y we repeat the same procedure for $v_{H'}$ in Γ'.

Proof of Lemma 30. When splitting at some hole $H \in \mathcal{H}$ (according to the procedure in Definition 105) we introduce at most one additional connected component, i.e. one additional region, in case P_H is a node separator[10] in its current region. Note that, even though we also split at v_H, the region to the right of P_H will remain connected via the boundary of H (we end up with two copied nodes at the position of v_H in the same region, though). Hence the number of regions is at most $|\mathcal{H}|+1$.

[10] A node separator w.r.t. two nodes u, v is a set of nodes, the removal of which decomposes the graph into (at least) two connected components, where u, v are in different components.

The idea to prove the simple property is that each split of the grid graph as described in the procedure above (Definition 29), will make the portal P_H impassable in the horizontal direction and the node v_H impassable in the vertical direction. Loosely speaking, in the resulting grid graph at least two holes will "merge", and become one hole. After repeating this for all holes, we obtain a grid graph with a single hole, namely the outer hole, i.e., it is simple (Definition 15).

However, the idea to "merge" holes via an impassable portal is not reconcilable with our definition of holes 13. Thus we use a $\frac{1}{3}$-transformation to Γ', where we can show that at least two holes merge using our formal definition with an analogous construction in Γ' (Definition 34). We then argue that the reverse transformation of the resulting simple graph with granularity $\frac{1}{3}$ corresponds to the same grid graph that we obtain with the original procedure has the same topology, that is, the same number of holes per region, i.e., none.

Let H be said hole with splitting node v_H. Let Γ' be the $\frac{1}{3}$-transformation of Γ *after* removing H as per the construction explained above. After the construction, the transformed nodes from the Portal P_y form a "channel", which connects H with at least one other hole, as otherwise v_H could not have been the leftmost node. Hence the number holes in Γ' decreases by at least one.

After repeating the above for all nodes, the remaining nodes in Γ' only have a single hole left, the outer hole. This is equivalent to the fact that for any pair of nodes, two paths in Γ' between those nodes (if any exist) can be continuously transformed into one another within $S_{\Gamma'}$ (see Definition 13), i.e., without "transforming" paths through holes. The same follows for any pair of nodes in Γ, as Γ does not admit the passing of the same portals which we "channeled" in Γ'. This is equivalent to the fact that Γ has only one hole, the outer hole. □

Lemma 35. *After the construction to remove holes in Γ (Definition 29), the only nodes that share coordinates are two copies of splitting nodes v_H.*

Proof. For a contradiction assume that after executing the construction (Definition 29) in Γ, the left and right node copies, say v_ℓ and v_r of P_H are in the same connected component (region). Then there is a v_ℓ, v_r-path within that region. Since P_H can not be crossed, that path must enclose at least one other inner hole (which touches on of the end nodes of P_H). But then this path would also have to cross the splitting portal $P_{H'}$ of at least one inner hole H', which is a contradiction. □

We show that the procedure in Definition 29 can be conducted in $O(\log n)$ rounds.

Proof of Lemma 31. In Appendix C.2 and Appendix C.3 we describe a few basic primitives that allows us to conduct broadcasts and aggregations on paths and cycles in $O(\log n)$ rounds. Note that these procedures can be done in parallel in case such structures intersect only $O(1)$ times in each node. Also note that portals and boundaries of holes form paths and cycles with only $O(1)$ intersections per node.

Using these primitives, it is easy to establish splitting nodes v_H and splitting Portals P_H (see Definition 29) by broadcasting the minimum x-coordinate on the boundary of H (breaking ties by maximum y coordinate) and subsequently informing the nodes on P_H of their role as nodes on a splitting portal. We do this in parallel for each H and P_H (note that only one broadcast has be conducted on P_H in case P_H is splitting portal for multiple holes). Determining the roles of nodes is the main part, the subsequent splitting procedure and simulation overhead takes just $O(1)$ rounds by Lemma 106. □

4.2. Decomposition into tunnel regions

Our next step is to ensure that each region is a *tunnel*, which we define as a region that has at most two gates (see Definition 27). Recall that as a consequence of the previous Section 4.1, we start out with a grid graph Γ that is decomposed into simple regions (cf. Lemma 30).

Remark 36. Recall that the boundary of each region is composed of alternating segments of walls and vertical gates (cf. Definitions 27, 28). Being a member of a wall or a gate of some region is a condition that each node can determine locally. Furthermore, walls and gates can both compute unique identifiers in $O(\log n)$ rounds, using our procedures for aggregation and broadcast in Appendix C.1 (Lemma 100).

In the following, we say that two vertical portals (cf. Definition 12) are adjacent to each other, if some horizontal grid edge consists of one node from each portal. The notion of walls and gates allows us to define *junction portals* that we require for the next stage of our decomposition. Informally, a junction portal is a portal at which a simple region "diverges" into at least three "tunnels" (although there are some degenerate cases for such junction portals, where a gate cuts away one of these "tunnels"). Formally we define (see also Figs. 4a and 4c):

Definition 37 *(Junction portals).* Let R be a simple region. A vertical portal P in R is a *junction* portal if:

i. P has at least 3 adjacent portals each intersecting at least 2 distinct walls; or
ii. P is a gate, and has at least 2 adjacent portals each intersecting at least 2 distinct walls.

The idea is to perform portal splitting operations (as described in Appendix C.3 Definition 105) on each junction portal and on specific nodes on the junction portal in order to separate these divergent tunnels. More specifically the construction works as follows.

Definition 38 *(Splitting at junction portals)*. Let P be a junction portal. We note that Definition 37 implies that there are at least 2 adjacent portals which each intersect multiple distinct walls to the left of P or to the right of P, or both. Suppose that portals $P_1, P_2, \dots, P_k$ are such portals with the property of intersecting distinct walls *to the left* of P and let these portals be ordered from north to south (the procedure for those to the right is analogous).
We first conduct a splitting operation at P (Definition 105 case 1).

Then, for each portal $P_i, 1 \le i < k-1$ we choose the bottommost node on P which is adjacent to some node on P_i and split at this node (see Definition 105 case 2+3). Note that this node must coincide with a hole boundary, since one of the two portals does not extend further south. Finally, note that this procedure effectively splits off each region that is bordered by one of the portals P_i since R is simple.

The goal of this section is the following three lemmas pertaining to the number of regions created, and the correctness and computational complexity of the construction:

Lemma 39 *(Computation of tunnel decomposition). Given a decomposition into simple regions, finding and splitting all junction portals can be done in $O(\log n)$ rounds.*

Lemma 40 *(Correctness of tunnel decomposition). All of the regions resulting from the procedure of finding splitting portals and splitting them are bounded by at most two walls and two gates.*

Lemma 41 *(Regions after splitting junctions). After decomposing all junctions into tunnels with the procedure above (Definition 38), the resulting number of regions is $O(|\mathcal{H}|)$.*

We first give a proof of Lemma 39, which states that junction portals can be found and split in parallel in $O(\log n)$ rounds of the HYBRID model:

Proof of Lemma 39. Nodes can determine whether their portal is incident to multiple distinct walls in $O(\log n)$ rounds as follows. Nodes broadcast any incident wall IDs to all other nodes on their portal. As soon as a node receives two wall IDs it instead broadcasts message "yes", indicating that the portal is incident to multiple distinct walls. If a node receives the message "yes", it stops what it was doing and starts broadcasting the message "yes" for $O(\log n)$ rounds. If no node receives two wall IDs or the message "yes" in $O(\log n)$ rounds then nodes can conclude that their portal is *not* incident to multiple walls. Each node can then inform their neighbors in one round whether its portal is incident to multiple distinct walls.

In the same way, each portal can determine whether it satisfies the criteria to be a junction portal (all nodes on a portal know whether or not the portal is a gate, and therefore know which case of Definition 37 is required). Finally, each node v can locally determine whether it is split if its portal is a junction portal. If the node to the right of v (resp. to the left of v) is on a portal incident to multiple distinct walls, and the node below and to the right to v is not, then v is to be split. Finally, the splitting can be performed according to Definition 105 in $O(\log n)$ rounds, per Lemma 106. □

Next, we introduce a tree structure which will be useful for the proofs of Lemmas 40 and 41. This structure is used for the purpose of proving claims only, we need not actually construct it.

Definition 42 *(Portal tree without cavities)*. Given a simple region R, consider the vertical portal graph $\mathcal{P}$ of the region R as in Definition 18. As R is simple, by Lemma 19 $\mathcal{P}$ is a tree. Remove all leaves of $\mathcal{P}$ which correspond to portals which only intersect one wall. Repeat this until no such leaves remain in $\mathcal{P}$. We call this the *portal tree without cavities* of R.

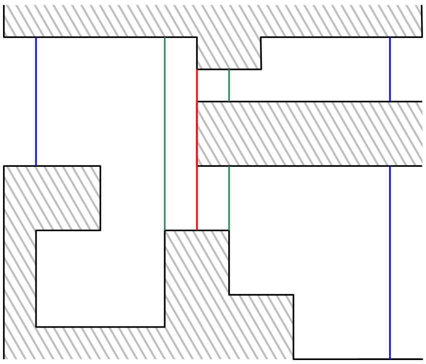
A visual example of a portal tree without cavities is given in Fig. 4b.

Lemma 43. *Consider a simple region R and an associated portal tree without cavities $\mathcal{P}$ (as in Definition 42). The following statements all hold:*

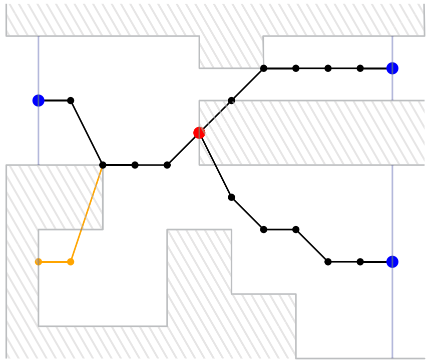
1. *Since we have only removed leaves from the portal tree, $\mathcal{P}$ remains connected (or empty).*
2. *All portals corresponding to nodes in $\mathcal{P}$ are incident to multiple distinct walls.*
3. *All remaining leaves in $\mathcal{P}$ correspond to gates.*
4. *The portal tree $\mathcal{P}$ contains all junction portals in R.*
5. *Iff a node in $\mathcal{P}$ has degree at least 3, or corresponds to a gate and has degree at least 2, then it corresponds to a junction portal.*

Proof. We take the statements in order:

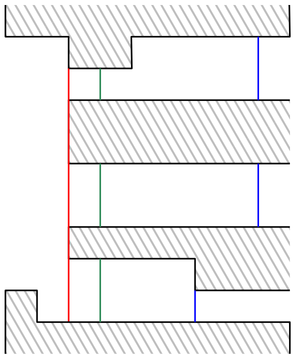
1. Removing leaves does not disconnect a tree.



(a) An example of case (i) of Definition 37. Blue portals are pre-existing gates. The red portal is a junction portal, because it has three adjacent portals (shown in green) which are incident to multiple distinct walls.



(b) An example of a vertical portal tree (corresponding to the example in Figure 4a). The orange nodes and edges would be removed, giving us a portal tree without cavities. In the remaining graph, blue nodes correspond to gates, and the red node corresponds to a junction portal.



(c) An example of case (ii) of Definition 37. The blue portals are pre-existing gates. The red portal is also a pre-existing gate, and it is adjacent to three portals which intersect multiple distinct walls (in green), so it is a junction portal.

Fig. 4. An explanation of junction portals and the portal tree without cavities. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

2. We will prove this by contradiction. Fix a portal P which corresponds to a node in $\mathcal{P}$, and suppose it is only incident to one wall, W. Let $p_\uparrow$ and $p_\downarrow$ be the top and bottom points of P; both are incident to W. Since region boundaries are alternating sequences of walls and gates, we must have that $p_\uparrow$ and $p_\downarrow$ are connected by some sub-path of W. Call this sub-path W' and suppose wlog that W' meets $p_\uparrow$ and $p_\downarrow$ from the left. Consider R': the subregion of R bounded by P and W'. Clearly R' must be simple, and therefore no other hole can appear in R'.
   Note that P cannot have degree 1 in $\mathcal{P}$, otherwise we would have removed it from $\mathcal{P}$. So P must have at least two adjacent portals which are incident to multiple walls: let P_1 and P_2 be two such portals. Furthermore P_1, P_2 must be to the right of P since the cavity R' is to its left. If either P_1 or P_2 of P would start and end with nodes in W this would be the beginning of another cavity R'', and by the same argument in the previous paragraph would have already been removed from $\mathcal{P}$. Therefore, P_1 and P_2 must have at least one wall other than W as one of their endpoints.
   For P to be incident to *two* portals to its right, P must have a hole immediately to the right of it that "separates" these. By our assumption, since the boundary of this hole intersects P it must belong to the wall W. And so P_1 and P_2 must both have one endpoint in W to the right of P as well. Combined with the above statement this implies that P_1, P_2 each have exactly one endpoint in W (which is to the right of P) and another endpoint in another wall. Since W must be contiguous, the section of W to the right of P must connect to either $p_\uparrow$ or $p_\downarrow$ and thus intersect both endpoints of at least one of the two portals P_1, P_2. This is a contradiction to the fact that both P_1 and P_2 must have an endpoint in a wall which is not W.
3. All remaining leaves must be incident to multiple distinct walls, by the previous statement. For a contradiction, fix a leaf which corresponds to a portal P but is not a gate. Note that P is only incident to one portal which remains in $\mathcal{P}$, suppose wlog that it is to the right of P.

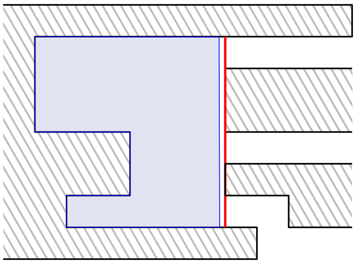


Fig. 5. An example of a dead-end region, in blue. The junction portal (in red) splits to the right, because there are at least three neighboring portals which connect distinct walls. But it does not split to the left, and so a dead-end region is formed.

If there is no portal to the left of P in R, then P is a section of W_1 (by our assumption that P is not a gate). Some node along P must be part of a different wall W_2, otherwise P would have been removed. Let $p_\uparrow$ be the node to the right of the topmost node on P which intersects W_2, and let $p_\downarrow$ be the node to the right of the bottom-most node on P which intersects W_2. $p_\uparrow$ and $p_\downarrow$ are both on different vertical portals to the right of P, and both of their portals intersect W_1 and W_2. This contradicts that P is a leaf in $\mathcal{P}$.

If there is a portal to the left of P, then it must be incident to only one wall, W (otherwise P would not be a leaf). If both endpoints of P are incident to W, then the logic from the previous paragraph applies: some node on P must be incident to a different wall, implying that P has two portals to its right which intersect distinct portals. So suppose this is not the case, and that one of the endpoints of P (wlog the top-most endpoint) is incident to a different wall W'. Let v be the topmost point on P which intersects W. The point immediately to the left of v must lie on a vertical portal which intersects W (since v intersects W) and W'. This is a contradiction, since by assumption no portal to the left of P is incident to multiple distinct walls.

4. We argue that nodes which we removed could only be adjacent to at most one portal which is incident to multiple walls, and therefore fail to satisfy the definition in Definition 37. This is the case because nodes only have degree 1 when we remove them: they may previously have been connected to nodes which we later removed, but the nodes which we later removed must only have been incident to one wall.
5. This claim immediately follows from statement 2, since this mirrors the definition of junction portals given in Definition 37. □

We now prove that the splitting procedure is correct (Lemma 40).

Proof of Lemma 40. Consider, for the sake of contradiction, a region R which has more than three gates after all junction portals have been split. Consider the portal tree described in Definition 42.

If this tree has at least 3 leaves, then it must (graph-theoretically) have a node with degree at least 3. But this node is a junction portal by Lemma 43. If this tree has only 2 leaves, it must have an internal node (with degree at least 2) which corresponds to a gate, due to our assumption that R has three gates. But by Lemma 43, this must also have been a junction portal. Therefore both cases lead to a contradiction and no such region R can exist. □

Next, we observe that splitting a junction portal can lead to a "dead-end" region, bounded by one gate and one wall. Note that a junction portal which creates a dead-end region must have both endpoints incident to the same wall. An example of this case is given in Fig. 5. Note that each junction portal can only create one dead-end region, since if there were a dead-end region on each side, then there would be no adjacent portals which connect two distinct walls (since each region is simple). Note also that if there is a dead-end region on one side, then, since the region is simple and therefore consists of alternating series of walls and gates, there can be no portal on that side which connects two distinct walls. Therefore the junction portal is not split on that side, and at most one dead-end region can be created on that side.

Observation 44. When splitting at a junction portal, this can result in at most one "dead-end" region being created: that is, a region which is bounded by one gate and one wall.

This happens when the junction portal only has one portal to its left (say) which only intersects one wall. This cannot happen on both sides because then it would not be a junction portal.

Proof. Let P be a junction portal, W be a wall, and let there be a dead-end region R to the left (w.l.o.g.) of P which is bounded by P and W. Since both ends of P are on a wall, they must be on W.

First, notice that R must be the only region to the left of P, since, if there was a region to the left of P which intersected two distinct walls, then it would not be a dead end, and we would not cut P between this region and R, so R would be a part of that region (we would call it a "cavity").

If there was a dead-end region also to the right of P, then since the endpoints of P are on W, this region must also be bounded solely by P and W. But we reach a contradiction, since this means that there could not be any walls aside from W in the region

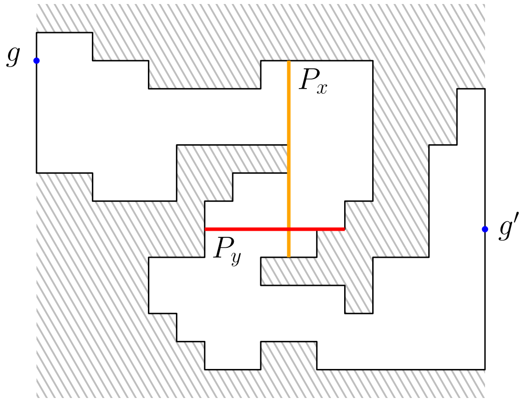


Fig. 6. Splitting a tunnel with gates consisting of single nodes.

to the right of P, since the regions are simple and region is only bounded by P and W, and therefore P could not be a junction portal. □

Next, we prove Lemma 41: that the number of regions resulting from splitting junction portals is not too large.

Proof of Lemma 41. We argue that a k-junction introduces at most $O(k)$ new regions and splitting portals. The claim then follows since the sum of gates of all junctions is in $O(|\mathcal{H}|)$.

Consider a region R which was a k-junction (i.e. was bounded by k gates) after the decomposition into simple regions, and consider its portal tree without cavities $\mathcal{P}$ as in Definition 42.

By Lemma 43 (3.) leaves of $\mathcal{P}$ must be gates of R thus $\mathcal{P}$ has $\ell \leq k$ leaves. Note that the remaining $k - \ell$ gates of R correspond to internal nodes of $\mathcal{P}$ (recall that these correspond to case (ii) of Definition 37). The ℓ leaves imply that there are, in the worst case, $\ell - 2$ nodes of degree at least 3 in $\mathcal{P}$. Each of these is incident to one dead end by Observation 44. Then, each of these junction portals is incident to 4 regions created by splitting at junction portals, giving $4\ell - 8$ regions. Suppose (again in the worst case), that the $k - \ell$ internal nodes which are gates only have degree 2. These gates are incident to 2 regions each created by splitting at these junction portals, giving $2k - 2\ell$. Summing these, and recalling that $\ell \leq k$, gives that at most $O(k)$ regions are created from a k-junction, as required. □

Finally, we conclude with a lemma which will be useful for next section: during the tunnel decomposition we get rid of almost all node copies within a tunnel, except for the two copies of a splitting node.

Lemma 45. *In the tunnel decomposition obtained above, there can be at most one pair of nodes that are copies of each other in each region. If that is the case, this pair of nodes forms the two point shaped gates in their tunnel region.*

Proof. As the region decomposition is simple, we can not have any pair of horizontally copied nodes in the same region as these would need to be connected by a path encircling the hole north- or south-adjacent to their corresponding portal. Hence we only need to consider the vertically copied nodes introduced by a horizontal node split. As all splits added in this section divide regions that are already simple, none of the resulting vertically copied nodes can be in the same region. Hence all of them must be created during the splitting into simple regions described in Definition 29. At that point we can still have more than one pair of vertically copied nodes, if the leftmost node of two different hole boundaries aligns. As those two holes induce two different walls and there must be at least one more wall, which may be both above and below those holes, we must have a junction portal splitting the two pairs of vertically copied nodes. □

4.3. Path-convex decomposition

Finally, we make the tunnel regions that we obtained in the previous section path-convex (cf. Definition 24), by splitting them at appropriate portals. Intuitively, for any two nodes in such a tunnel with a shortest path in Γ that travels outside the tunnel, the additional portals will separate that pair of nodes. We will also see that it is sufficient to further separate tunnels at a constant number of vertical or horizontal portals. The main question that we answer in this section is *where* the tunnels should be split to make them path-convex. While the construction of the splits is not too complicated, the main challenge is the proof of correctness, i.e., all "offending" node pairs whose shortest path runs outside of the region must be separated. We answer this question in stages.

First, we impose the assumption that we have a tunnel region T with gates that are single nodes g and g', see Fig. 6. We show that we can split T into path-convex regions using only a horizontal and a vertical portal P_x and P_y. Roughly, if $d_x = d_{x,T}(g, g')$ is the horizontal distance from g to g' in T then all nodes which are at distance $\frac{d_x}{2}$ to g and g' will be part of P_x (details in Definition 49),

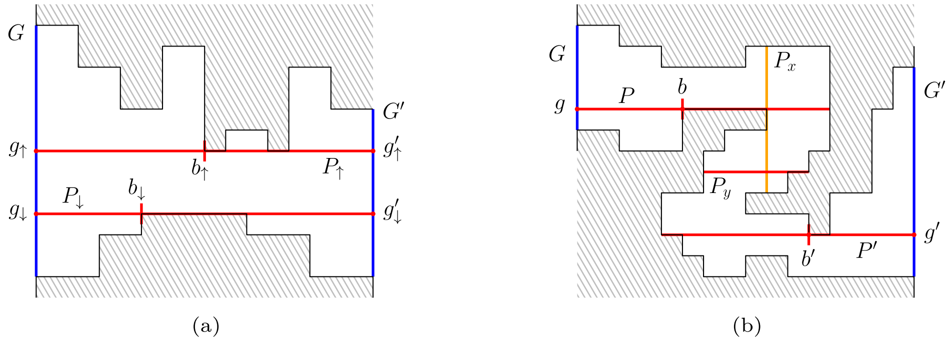


Fig. 7. Splitting tunnels into path-convex regions in two cases according to Definition 58.

which we show forms a vertical portal (cf., Lemma 50). The horizontal portal P_y is defined symmetrically. We split T at P_x and P_y (cf., Fig. 2).

As the first part of our proof we show that two nodes u, v that end up in the same region R after splitting at P_x and P_y that both lie on (possibly different) shortest paths from g to g' have a shortest path within R (see Lemma 55). Loosely speaking, this covers the well behaved case where u, v are not located in "cavities" of T. We then move on to this harder case and show the same claim for any pair of nodes $u, v \in R$ (see Lemma 56) by using fundamental property of grid graphs (see Lemma 97) that lets us fall back to the easier case.

In the last stage of the construction, we remove the assumption that gates are point-shaped. We distinguish two cases, depending on whether there is a horizontal portal connecting the two gates (see Definition 58 and Fig. 7). We then give a decomposition procedure for each case, whereas the first case (a) is well behaved and the second case (b) has the property that for any pair of nodes in a region in the "middle part" a hypothetical shortest path that goes outside T can always leave and enter T through two nodes g, g' in the gates G, G', which allows us to fall back to the proof of correctness for this case and split the "middle part" at P_x, P_y.

We conclude with lemmas pertaining to the number of regions created, the correctness, and the computational complexity.

Lemma 46 *(Regions after convex decomposition). The construction in Definition 58 produces at most 10 regions for every tunnel which we split.*

Lemma 47 *(Correctness of path-convex decomposition). The construction in Definition 58 (illustrated in Fig. 7) decomposes T into path-convex regions.*

Lemma 48 *(Computation of path-convex decomposition). The construction of the portals in Definition 58 and 49 and the described splitting of the tunnel region T takes $O(\log n)$ rounds.*

The lemmas from the three sections of the region decomposition can then be combined to obtain the proof for Theorem 26: that our construction is efficient and results in a path-convex region decomposition that is linear in the number of holes.

Proof of Theorem 26. We obtain a simple region decomposition in the claimed number of rounds by Lemma 30. These simple regions can be transformed into tunnel regions, i.e. regions with at most two gates, due to Lemma 40. Finally, we have shown that we can further break up tunnels into convex regions by Lemma 47. The computation of this decomposition requires $O(\log n)$ rounds in the HYBRID model, by Lemmas 31, 39 and 48. The number of regions is $O(|\mathcal{H}|)$ by Lemmas 30, 41 and 46. □

It remains to prove Lemmas 47 and 48. We will subdivide the approach into several stages, where we, step by step, remove some of the simplifying assumptions and show that the more general case can be reduced to the simpler one.

In the first stage we assume the special case that the two gates of a tunnel are single nodes (points). We use the following construction to make T path-convex. We start with the following definition. A visual depiction of the splitting procedure described in Definition 49 is given in Fig. 6.

Definition 49 *(Splitting portals for tunnels with point shaped gates).* Let T be a tunnel region with two gates $g, g' \in T$ each consisting of a single node. Let $d_x := d_x(g, g')$ and $d_y := d_y(g, g')$, which are well defined since a tunnel T is also simple. Let $P_x = \{v \in T \mid d_x(g, v) = \lceil \frac{d_x}{2} \rceil, d_x(g', v) = \lfloor \frac{d_x}{2} \rfloor\}$ and analogously $P_y = \{v \in T \mid d_y(g, v) = \lceil \frac{d_y}{2} \rceil, d_y(g', v) = \lfloor \frac{d_y}{2} \rfloor\}$.

Lemma 50. *The set P_x, respective P_y (see Definition 49 and Fig. 6) forms a vertical, respective a horizontal portal in the tunnel region T (the simplicity property of tunnels suffices here). Each portal is a node separator w.r.t. g and g'.*

Proof. By Definition 12, we have to show that P_x, P_y, are maximal connected components in the graph induced by horizontal or vertical edges, respectively. Due to symmetry, it suffices to consider P_x. Let $a \in P_x$ and let P_a be the maximally vertically connected component of a. Any vertically adjacent point to a must be also in P_x since the horizontal distance to g and g' does not change, thus $P_a \subseteq P_x$.

Assume there is a node $b \in P_x \setminus P_a$. Note that since the horizontal distances of a, b to g and g' add up to exactly $d_x(g, g')$, a and b must lie on gg' paths Π_a, Π_b that are shortest w.r.t. the horizontal distance. The path Π_b can not visit any point in P_a since otherwise either $b \in P_a$ or $b \notin P_x$ (for the latter, consider the definition of P_x). Due to vertical maximality, P_a meets the boundary of T at both ends. Thus Π_a and Π_b must together enclose some boundary nodes of T, which means T has a hole, a contradiction.

It remains to show that P_x is a node separator. Assuming that there would be a gg'-path that does not intersect P_x would again induce a hole in T, since P_x is maximal vertically connected. □

Remark 51. The portals P_x and P_y characterized in Definition 49 result in regions consisting of nodes categorized by their location with respect to P_x and P_y. We obtain at most four regions. Each region is formed by the set of nodes that are on the same side of both node separators P_x and P_y. Formally, we assume that nodes on the portals P_x and P_y are part of all regions which they are adjacent to. On the algorithmic level, we split T at P_x and P_y, see Fig. 6 where nodes on P_x and P_y are "copied" with one copy in each resulting region, as in Definition 105.

We identify the following easy case.

Fact 52. Given a region R of some grid graph Γ that is bordered by a vertical and horizontal gate G, G' which intersect in a node. Then there is *no* shortest path in Γ between two points in R that leaves and then reenters through any of those gates as such a path can always be made shorter. In particular, this implies that a region that has just one or at most two gates that intersect, must be path-convex.

Another important property is the following.

Lemma 53. *Let R be a path-convex region and let P be a portal through R. If we split R at P the resulting regions remain path-convex.*

Proof. Let $u, v \in R$ both be in one of the resulting regions after splitting at P. Since R was path-convex there is a shortest uv-path Π inside R. Assume Π would cross P, i.e., leave the new region of u, v and reenter it through nods on P, then Π could clearly be shortened along P, which contradicts the fact that Π is shortest. □

We continue with the first step of the overall proof, namely grid nodes $u, v \in T$ that lie on a shortest path from g to g'. In particular, this means that u, v are not located in "cavities" of the tunnel region, which is a more complicated case, which we will carefully break down into this simpler case afterwards. We start with the following technical lemma that shows that two points that are on shortest gg'-paths and on P_x and P_y, are relatively close. Recall that $d_x := d_x(g, g')$ and $d_y := d_y(g, g')$.

Lemma 54. *Let T be a tunnel region with two gates g, g' that are single nodes. Let Π, Π' be two shortest gg'-paths. Let $u \in P_x \cap \Pi$ and $v \in P_y \cap \Pi'$ (interpreting Π, Π' as set of nodes on the path; for P_x, P_y see Definition 49). Then $d_x(u, v) \le \lceil \frac{d_x}{2} \rceil$ and $d_y(u, v) \le \lceil \frac{d_y}{2} \rceil$.*

Proof. By symmetry, it suffices to prove $d_x(u, v) \le \lceil \frac{d_x}{2} \rceil$. Let $u' \in P_x \cap \Pi'$. Since T is simple, we have $|\Pi|_x = |\Pi'|_x = d_x$ by Lemma 92. In particular, $\Pi'_{gu'} = \lceil \frac{d_x}{2} \rceil$ for the sub path from g to u' (by the definition of P_x and since $\Pi'_{gu'}$ is shortest too). Furthermore, P_x separates g, g' by Lemma 50. Assume that v is on the same side as g, otherwise the subsequent estimations admit swapping g and g'. Then

$$d_x(u, v) \le \underbrace{d_x(u, u')}_{=0,\text{ as } u,u' \in P_x} + d_x(u', v) = \underbrace{d_x(u', v)}_{\text{both on } \Pi'} \le d_x(u', g) = |\Pi'_{gu'}|_x = \lceil \tfrac{d_x}{2} \rceil.$$

Note that the requirement $v \in P_y$ is only needed for the proof for the vertical distance. □

The lemma above already shows that leaving T is *not* required to get from u to v on some shortest path, given that both points are on shortest gg'-paths and on P_x and P_y. We will now generalize the latter condition for two such points within the same region of T after splitting at P_x, P_y.

Lemma 55. *Let T be a tunnel region with two gates g, g' that are single nodes. Let Π, Π' be two shortest gg'-paths. Let u on Π and v on Π' and $u, v \in R$ for some region R. Then $d_{R,x}(u, v) \le \lceil \frac{d_x}{2} \rceil$ and $d_{R,y}(u, v) \le \lceil \frac{d_y}{2} \rceil$.*

Proof. We show $d_x(u,v) \le \lceil \frac{d_x}{2} \rceil$, the vertical distance case is analogous. Note that Π and Π' must intersect P_x and P_y in points $x_1 \in P_x$, $y_1 \in P_y$, $x_2 \in P_x$ and $y_2 \in P_y$, respectively.

First we show that w.l.o.g., we can assume that $y_1 = y_2$. If $y_1 \neq y_2$, then, by Lemma 95 one of the two paths $\Pi_{gy_1} \circ \overline{y_1 y_2} \circ \Pi'_{y_2 g'}$ or $\Pi'_{gy_2} \circ \overline{y_2 y_1} \circ \Pi_{y_1 g'}$ is a shortest uv-path as well; let this path be $\widetilde{\Pi}$. Since $u, v \in R$ are on the same side w.r.t. the separator P_y, we have that u or v is on $\widetilde{\Pi}$. We replace $\Pi := \widetilde{\Pi}$, if u is on $\widetilde{\Pi}$, or $\Pi' := \widetilde{\Pi}$, if v is on $\widetilde{\Pi}$. Note that the preconditions of this Lemma still hold. *Additionally* we obtain the property that Π and Π' both intersect in some point $R \cap P_y$.

Note that, Π and Π' do not enter, exit and subsequently reenter R (see Fact 52). This means that there are points u_1, u_2, v_1, v_2, such that $\Pi_{u_1 u_2}$ and $\Pi'_{v_1 v_2}$ stay within R. The points of entry and exit into R can only be on g, g', P_x, P_y. The only case where the two entry (or exit points) of Π, Π' into R may differ, is if they are located on P_x, due to our previous statement and due to the fact that g, g' are points. We will assume that $u_1 \neq u_2$ is possible but $v_1 = v_2$ (for instance $u_1, u_2 \in P_x$ and $v_1, v_2 \in P_y$ or $v_1, v_2 = g'$) since the following proof allows to analogously switch roles of the point pairs. More importantly, we observe that $d_x(u_1, u_2) = 0$, meaning they are "the same" in terms of vertical distance to other points.

We split the sub paths Π_{x_1, u_2} and Π'_{v_1, v_2} into the following segments:

$$|\Pi_{u_1, u_2}|_x = \alpha + \beta, \text{ where } \alpha := d_x(u_1, u), \beta := d_x(u, u_2)$$

$$|\Pi'_{v_1, v_2}|_x = \gamma + \delta, \text{ where } \gamma := d_x(v_1, v), \delta := d_x(v, v_2)$$

Since $u_1, v_1 \in P_x$ and $v_2, u_2 \in P_y$, we can apply Lemma 54 thus:

$$\alpha + \beta \le \lceil \tfrac{d_x}{2} \rceil \text{ and } \gamma + \delta \le \lceil \tfrac{d_x}{2} \rceil$$

This implies

$$\alpha + \gamma \le \lceil \tfrac{d_x}{2} \rceil \text{ or } \beta + \delta \le \lceil \tfrac{d_x}{2} \rceil \tag{1}$$

as otherwise one of the inequalities of (1) would be false. If the former of the two inequalities (1) is true then applying the triangle inequality gives

$$d_{x,R}(u,v) \le |\Pi_{u,u_1}|_x + \underbrace{d_x(u_1, v_1)}_{=0} + |\Pi_{v_1, v}|_x = \alpha + \gamma \le \lceil \tfrac{d_x}{2} \rceil.$$

If the latter of inequalities (1) is true then we get

$$d_{x,R}(u,v) \overset{u_2 = v_2}{\le} |\Pi_{u,u_2}|_x + |\Pi_{v_2, v}|_x = \beta + \delta \le \lceil \tfrac{d_x}{2} \rceil. \quad \square$$

In the next stage, we show that we have path convexity for arbitrary points u, v in a given region R (but still gates consist of a single node). We will show that a shortest uv-path that leaves R will necessarily also have to leave T, and we will see that it can not be shorter than a path that stays inside T, partly relying on previous lemmas. One caveat is that after splitting the tunnel T the boundaries to the outer hole can be of arbitrary shape, in particular, we could have highly complex cavities attached to such regions, like, e.g., spirals.

To show that such structures do not matter in terms of path convexity, we exploit a fundamental property of grid graphs, namely that for any three of grid nodes a, b, v, there is a shortest ab-path Π and a point v' on Π such that any shortest $av'v$- or $bv'v$-path is a shortest av- or bv-path, respectively (see Appendix B Lemma 97). This allows us to fall back to our previous result in Lemma 55 where u, v lie on shortest gg'-paths.

Lemma 56. *Let T be a tunnel region with two gates g, g' that are single nodes. Let $u, v \in R$ for some region R of T after splitting at P_x, P_y. Let $\widetilde{\Pi}$ be a uv-path that is partially outside of R. Then $|\widetilde{\Pi}| \ge d_R(u,v)$ (which implies path convexity).*

Proof. We will prove that for any such path $\widetilde{\Pi}$ there is one that stays within R and is at most as long as $\widetilde{\Pi}$. First assume that $\widetilde{\Pi}$ leaves R but not T, i.e., $\widetilde{\Pi}$ has a grid node that is not in R but none that is not in T. This means $\widetilde{\Pi}$ does not cross through g or g'. Then, by Fact 52, there must be a shorter path that stays within R. Furthermore, $\widetilde{\Pi}$ must be a simple path (no node visited twice) since otherwise $\widetilde{\Pi}$ could be made shorter by short-cutting the loop.

This implies that if $\widetilde{\Pi}$ leaves T at g it can only reenter at g' or vice versa, which is the case left to consider. Let $\widetilde{\Pi}_T$ be the subpaths of $\widetilde{\Pi}$ inside T. W.l.o.g., we will assume that $\widetilde{\Pi}_T$ connects g with u and g' with v (otherwise we switch the roles of g, g'), we refer to this as property (A). We show that for such $\widetilde{\Pi}$ we can fall back on Lemma 56 by employing Lemma 97 in order to prove $|\widetilde{\Pi}| \ge d_R(u,v)$. Let Π, Π' be two shortest gg'-paths that are *closest* to u and v, respectively, and let u' and v' be two nodes on Π, Π' that are closest to u and v, respectively. By Lemma 97 we have $d_T(g,u) = |\Pi_{gu'}| + d_R(u', u)$ and $d_T(g', v) = |\Pi'_{g'v'}| + d_R(v', v)$.

Note that there is always some shortest gg'-path that intersects R. This is clear if P_x, P_y do not intersect, since each region of T has to be crossed. If they do intersect, then by definition of P_x, P_y there is a shortest gg'-path that crosses that intersection and thus R (note that we consider the nodes in the portals P_x and P_y that delimit a region R as part of that region, see Remark 51). Therefore, the definition of u', v' implies that both are in R. This, in turn, implies that P_x and P_y are each crossed by *one* of the two path segments $\Pi_{gu'}, \Pi'_{g'v'}$ (could be the same path segment for both), which by Definition 49 of P_x and P_y means that we have property (B): $|\Pi_{gu'}| + |\Pi'_{g'v'}| \geq \lfloor \frac{d_x}{2} \rfloor + \lfloor \frac{d_y}{2} \rfloor$.

$$
\begin{aligned}
|\widetilde{\Pi}_T| &\overset{(A)}{\geq} d_T(g,u) + d_T(g',v) \\
&\overset{\text{Lem.97}}{=} |\Pi_{gu'}| + d_R(u',u) + |\Pi'_{g'v'}| + d_R(v',v) \\
&\overset{(B)}{=} \lfloor \tfrac{d_x}{2} \rfloor + \lfloor \tfrac{d_y}{2} \rfloor + d_R(u',u) + d_R(v',v) \\
&= \lceil \tfrac{d_x}{2} \rceil + \lceil \tfrac{d_y}{2} \rceil - \frac{\mathbb{1}_{[d_x \text{ odd}]} + \mathbb{1}_{[d_y \text{ odd}]}}{2} + d_R(u',u) + d_R(v',v) \\
&\overset{\text{Lem.55}}{\geq} d_{R,x}(u',v') + d_{R,y}(u',v') + d_R(u',u) + d_R(v',v) - \frac{\mathbb{1}_{[d_x \text{ odd}]} + \mathbb{1}_{[d_y \text{ odd}]}}{2} \\
&\overset{\text{Lem.92}}{=} d_R(u',v') + d_R(u',u) + d_R(v',v) - \frac{\mathbb{1}_{[d_x \text{ odd}]} + \mathbb{1}_{[d_y \text{ odd}]}}{2} \\
&\overset{\Delta\text{-ineq.}}{\geq} d_R(u,v) - \frac{\mathbb{1}_{[d_x \text{ odd}]} + \mathbb{1}_{[d_y \text{ odd}]}}{2}
\end{aligned}
$$

Note that we lower bounded the part of $\widetilde{\Pi}$ inside T. If g and g' are distinct points then $|\widetilde{\Pi}| \geq |\widetilde{\Pi}_T| + 1 \geq d_R(u,v)$. If g and g' happen to be unconnected copies of a node (see Lemma 45) then any $|\widetilde{\Pi}|_x, |\widetilde{\Pi}|_y$ must both be even, thus $|\widetilde{\Pi}_T| \geq d_R(u,v)$. □

Finally, we generalize our approach to handle tunnels T with gates G, G' that are vertical portals. Note that when we created the tunnels in Section 4.2 we only used vertical gates to delimit those. Also, note that our construction to make tunnels with single node gates path-convex can not be used out of the box, in particular since P_y from Definition 49 is no longer well defined.

We describe here two fundamental cases of the shape of T and give a construction to make convex in either case, see Fig. 7.

Fact 57. Let T be a tunnel with gates G, G'. Let $g_1, \dots, g_\ell$ be the set of nodes on G from bottom to top (i.e., sorted by ascending y-coordinate). Let $d(i) := d_{T,y}(g_i, G')$ be the vertical distance in T from g_i to the point in G' that minimizes it (cf. Definition 11). Consider $d(i)$ as a function over $i \in [\ell]$. We have intervals $[1, \dots, i_1 - 1]$, $[i_1, \dots, i_2]$ and $[i_2 + 1, \dots, \ell]$ where $d(i)$ is monotonously decreasing, assuming its minimum and monotonously increasing, respectively. Note that the first and third interval can be empty ($i_1 = 1$ or $i_2 = \ell$) and $i_1 = i_2$ is possible, in which case we define $i' := i_1$. This property of $d(i)$ is due to the fact that T is simple and G and G' are vertical portals (our claims are illustrated by Fig. 7). We distinguish two cases

(a) $d(i)$ strictly decreases until $d(i_1) = 0$ for some $i_1 \in [\ell]$ then stays constant at 0 until some $i_2 \in [\ell]$ after which $d(i)$ strictly increases (includes $i_1 = i_2$ and $d(i_1) = 0$).
(b) $d(i)$ strictly decreases until assuming an absolute minimum $d(i') > 0$ for $i' \in [\ell]$ and then strictly increases (excludes $d(i') = 0$).

We split the tunnels differently depending on which case of Fact 57 our tunnel falls into (see Fig. 7 for a visualization).

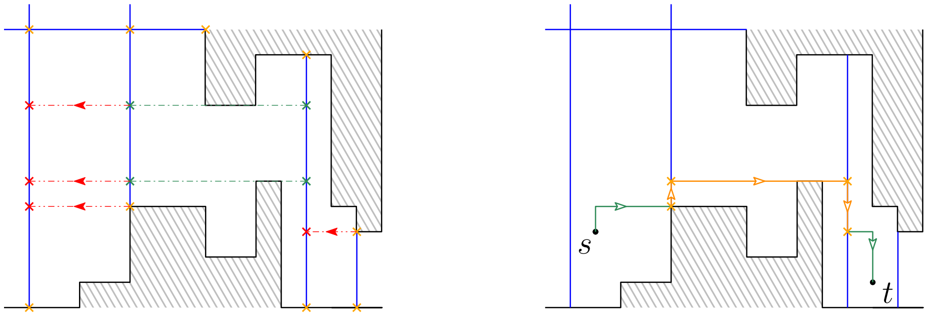
Definition 58. Based on this case distinction in Fact 57 we make the following construction to decompose T into path-convex regions (consult Fig. 7 for an illustration).

(a) Let $g_\downarrow := g_{i_1}$ and $g_\uparrow := g_{i_2}$. Note that we have two nodes $g'_\downarrow, g'_\uparrow$ on G', that fulfill exactly the same conditions on G' as $g_\downarrow, g_\uparrow$ on G. The pairs are "facing" each other horizontally, i.e., we have $d_y(g_\downarrow, g'_\downarrow), d_y(g_\uparrow, g'_\uparrow) = 0$. We split T at the two horizontal portals $P_\downarrow, P_\uparrow$ in T between the pairs $g_\downarrow, g'_\downarrow$ and $g_\uparrow, g'_\uparrow$ (possibly $P_\downarrow = P_\uparrow$). Furthermore, in case $P_\uparrow$ touches a boundary node $b_\uparrow \neq g_\uparrow, g'_\uparrow$ of a hole *above* $P_\uparrow$, then we split the region above $P_\uparrow$ at the leftmost such node in horizontal direction. We do this symmetrically for the region below $P_\downarrow$.
(b) Let $g := g_{i'}$ the node on G where $d(i')$ is minimal and let g' be the node with the same property on G'. We split T at horizontal portals P and P' in T through g and g'. Let b, b' be the leftmost node where P, P' touch the boundary with the hole located above P or below P', respectively. We conduct a horizontal split at b and b', respectively. Then we split T at P_x, P_y that are defined with respect to the two nodes g, g' as in Definition 49.

We conclude with the proofs of Lemmas 46 to 48.

Proof of Lemma 46. Case (b) potentially produces the most regions. P_x intersects at most 3 horizontal portals which gives at most 8 regions. Splitting at b, b' adds at most 2 more regions. □

Proof of Lemma 47. *Case (a):* In non-degenerate cases we obtain the 5 regions shown in the example in Fig. 7a. The middle one forms a rectangle, which is clearly path-convex. The others are bordered by at most 2 portals each, thus are path-convex by Fact 52.



(a) An example of the placement of landmarks. Portal-end landmarks are in orange; overhang-induced landmarks are in green; and projected landmarks are in red.

(b) An example of a landmark path from s to t. The Orange edges are edges in E_Λ, and the green edges are composed of multiple edges in E_Γ.

Fig. 8. An overview of our landmark construction.

Case (b): If we disregard the split at P_x, then the regions above and below P and P' are path-convex by Fact 52. This property is retained for the resulting regions after potentially splitting these two regions again at P_x, by Lemma 53. We end up with two regions that are bordered, only by P, G and P', G', respectively, which are also path-convex by Fact 52.

Let T' be the connected area that is enclosed by P, P' (such that P_y is *inside* T') with the nodes on the portals P, P' part of T'. The remaining area that is unaccounted for resembles our tunnel with single node gates. In particular, for any node pair u, v inside T' any uv-path Π that leaves the *tunnel* T has to intersect P *and* G when leaving T and P' *and* G' when reentering. Since P, G and P', G' intersect in g and g', respectively, there is also a uv-path $\widetilde{\Pi}$ that goes through g and g' and is at most as long as Π (see Fig. 7b).

We can therefore consider T' as tunnel region with single node gates g, g', since these points of entry and exit suffice to obtain shortest paths that travel outside of T'. By Lemma 56, after splitting at P_x and P_y, the regions that T' decomposes into are path-convex. □

Proof of Lemma 48. We first invoke Corollary 22 for the two gates G, G', after which every node in T will know their distance to these gates. This takes $O(\log n)$ rounds. In particular, this allows the nodes on G to determine their role, in particular, whether they are the nodes $g_\downarrow, g_\uparrow$ or g described in Fact 57, just by looking at the distance to G' of their neighbors and using the properties of the distance function d defined in Fact 57. We proceed analogously for the nodes on G'.

In case (a) we construct the two horizontal portals $P_\downarrow, P_\uparrow$ in T with endpoints at $g_\downarrow, g_\uparrow$ using our procedures from Sections C.1 and C.2. Using these subroutines, we can also identify the nodes $b_\downarrow, b_\uparrow$ described in case (a). We conduct a split at $b_\uparrow$ on $P_\uparrow$ with respect to the hole above $b_\uparrow$, as described in Section C.3.

Let us consider case (b). Constructing P, P' and b, b' is analogous to case (a). Since T is simple we can use Corollary 102 and broadcast the distances $d_x = d_{x,T}(g, g')$ and $d_y = d_{y,T}(g, g')$ to all nodes in T. Since these also know their horizontal and vertical distances to G and G', nodes can locally decide whether they are part of the portals P_x, P_y (see Definition 49). We have thus constructed all portals and nodes at which we want to split our regions (meaning that all nodes know about their roles in the splitting process). We apply the construction described in Definition 105, where the round complexity of $O(\log n)$ is guaranteed by Lemma 106. □

5. Landmark graph

Our next task is to provide a skeleton of the regionalization which facilitates optimal routing. To do this, we construct what we call a *landmark graph*. The intuition is that we mark certain gate nodes as *landmarks* if they are likely to appear on a shortest path between two regions, and so we can (at least approximately) reduce the problem of finding shortest paths to the problem of finding shortest paths between landmarks.

We connect pairs of landmarks which lie in the same region with virtual edges. These edges have weights corresponding to the distance between the landmarks (calculated using the SSSP subroutine developed in Section 3). We call the resulting graph the *landmark graph*. Our goal is to distribute the landmark graph to every node so that routing decisions can be made locally: this is made possible by its relatively small size.

For this section, we assume that we have a grid graph Γ which has been regionalized by Theorem 26. We call this a *regionalized grid graph*; we refer to the set of regions as $\mathcal{R}$. This section introduces landmarks and presents various useful properties of them, shows that the landmark graph is of small size, shows that shortest paths in the landmark graph pass through the same regions as shortest paths in the underlying grid graph, and finally demonstrates that the landmark graph can be computed quickly.

First, we define which nodes are marked as landmarks. An example of the placement of landmarks, colored by type, is given in Fig. 8a.

Definition 59. Given a regionalized grid graph Γ, consider a vertex $v \in V_\Gamma$ which lies on a *gate* G. Let P be the portal perpendicular to G passing through v (note that if v lies at the intersection of four gates, it is a landmark of the first type), and let R be one of the regions incident to G. Note that v is one of the endpoints of $P \cap R$ (the portal P restricted to the region R).

Then v is a landmark if one of the following holds:

i. v is an **endpoint** of G.
ii. v is an **overhang-induced landmark**. Let u be the other endpoint of $P \cap R$. Then v is an overhang-induced landmark if for some $p \in P \cap R$:
   - p lies on a wall W; and
   - u lies either on W or a gate which is not G

   We describe p as an "overhang" (though note: p is not marked as a landmark).
iii. v is a **projection landmark**: that is, if any node on P is a landmark of either of the first two types.

Next, we define the landmark graph as follows:

Definition 60. Suppose we have a regionalized grid graph $\Gamma = (V_\Gamma, E_\Gamma)$ with a set of landmarks $V_\Lambda \subseteq V_\Gamma$. We define the *landmark graph* as $\Lambda = (V_\Lambda, E_\Lambda)$, where $\{u, v\} \in E_\Lambda$ if $u, v \in V_\Lambda$ and either: (i) u and v are on the same portal and there is no landmark between them; or (ii) u and v are on different gates incident to the same region, and v is the closest landmark on its gate to u.

Note that we do not connect *all* pairs of landmarks which lie in the same region: to do so would give $|E_\Lambda| = \omega(|\mathcal{H}|^2)$, and this would adversely affect the running time. We show in Section 5.1 (Lemma 70) that the edges that we select are sufficient to preserve the properties which we need. We present the following observation, which gives us that $|E_\Lambda| = O(|V_\Lambda|)$:

Observation 61. Definition 60 has two rules adding edges to the landmark graph. The first adds edges between adjacent landmarks on a gate. The second adds edges to the closest point on gates bordering the same region. As each region has a constant number of gates according to Observation 75, we can conclude that each landmark has constant degree.

Next, we define the notion of a landmark path (see also Fig. 8b). Note that the following definition does not require that s and t are landmarks, merely vertices in the grid graph.

Definition 62. Suppose we have a regionalized grid graph Γ with a set $V_\Lambda \subseteq V_\Gamma$ of landmarks and two vertices s, t which lie in different regions in $\mathcal{R}$. We define a *landmark path* from s to t as any path which goes from s to some landmark ℓ_s via edges in E_Γ, then goes from ℓ_s to some other landmark ℓ_t via edges in E_Λ (Definition 60), and then goes from ℓ_t to t via edges in E_Γ. We say that a landmark path induces a sequence of regions $(R_0, R_1 \dots R_m)$: these are the regions which the underlying grid edges of the landmark path pass through, in order.[11]

We now give an overview of the rest of the section. In Section 5.1, we give some miscellaneous properties of the landmark graph. In Section 5.2, we give several lemmas relating to the routing-related properties of landmark graphs, and we conclude with a proof of the following theorem:

Theorem 63. *Let Γ be a regionalized grid graph with a set $V_\Lambda \subseteq V_\Gamma$ of landmarks, and two vertices s, t in different regions in $\mathcal{R}$. Some shortest path in the grid graph from s to t crosses exactly the same sequence of regions as those induced by the shortest landmark path from s to t.*

Finally, we claim that the landmark graph is also easy to compute. In particular, we have the following lemmas regarding its computation, which we prove in Section 5.3.

Lemma 64. *Given a regionalized grid graph Γ, nodes can be informed whether they are landmarks (as in Definition 59) in $O(\log n)$ rounds.*

Lemma 65. *Given a regionalized grid graph Γ with a set of landmarks V_Λ (computed as in Lemma 64), the landmark graph $\Lambda = (V_\Lambda, E_\Lambda)$ can be computed, such that every landmark node knows its adjacent landmarks in Λ in $O(\log n)$ rounds.*

5.1. Properties of the landmark graph

In this section give some miscellaneous properties of landmarks, the landmark graph, and landmark paths which will be useful in the two subsequent subsections.

[11] In the special case that two consecutive regions R_i, R_{i+1} meet at a corner, we include one of the regions which is adjacent to both R_i and R_{i+1} (chosen arbitrarily) in the sequence, between them.

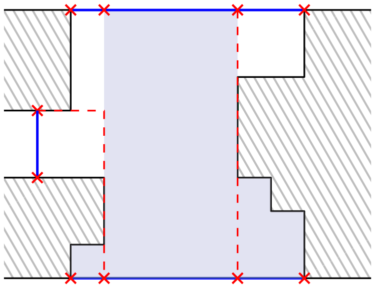


Fig. 9. A visual explanation of Lemma 66. Blue lines are gates, red crosses are landmarks, and two example paths between landmarks are depicted with red dotted lines. The bottom gate corresponds to G, and R_V is denoted by a blue shaded area. Notice that if landmarks on other gates lie inside R_V then there must be a corresponding landmark on G; if landmarks on other gates lie outside R_V then they must enter from the left or right, and then the shortest path is direct to G.

We begin with a lemma which says that, if we start at a landmark in a region R, the closest point in another region (equivalently, the closest point on another gate bounding R) is also a landmark. See also Fig. 9 for some visual intuition for the lemma.

Lemma 66. *Let Γ be a regionalized grid graph with a set of landmarks $V_\Lambda \subseteq V_\Gamma$. Let u be a landmark incident to a region R, and let G be a gate incident to R, on which u does not lie.*

The closest vertex on G to u is a landmark.

Proof. Suppose wlog that G is horizontal and R is to the north of G. Let $(v_1, v_2 \dots v_k)$ be the set of nodes of G from left to right. For each v_i, let P_i denote the vertical portal on which v_i lies, restricted to R, and let R_V denote $\bigcup P_i$, i.e. the union of the vertical portals through the nodes on G. Let Π denote a shortest path from u to G.

Observe that (i) Π must enter R_V (since $G \in R_V$), and (ii) from the first point on Π which lies on some portal $P_i \subseteq R_V$, the shortest path to G is along P_i.

If $u \in P_i$ for some $P_i \in R_V$, then, since u lies on a gate (as it is a landmark), and there is a portal connecting u and G, either v_i is a landmark of the first two types, or v_i must be a projected landmark (case (iii) of Definition 59).

Otherwise $u \notin R_V$. Let q be the last node on Π which is not in R_V, and let $p \in P_i$ (for some P_i) be the first node which is. It suffices to argue that v_i is a landmark. Observe that the grid edge from q to p must be horizontal (if it were vertical then clearly $q \in P_i$, which is a contradiction); suppose wlog that q is to the left of p.

Since $q \in R_V$, there is some grid point which is not a grid node on the line segment between q and G. The grid point immediately to the right of this grid point must be a grid node, however, since it lies on P_i: call this node p'. It is clear that p' is incident to a wall W (since the grid point to its left is unoccupied). We argue that p' is an overhang, and that v_i is therefore an overhang-induced landmark (case (ii) of Definition 59).

Let v_i' be the endpoint of P_i which is not v_i. If v_i' lies on a gate, or v_i' is a wall of node of the wall $W' \neq w$ then v_i is an overhang-induced landmark and we are done. Suppose v_i' lies on W. Let R' be the region bounded by W and the segment of P_i between p' and v_i'. Since R' contains no landmarks, Π must have entered this region to reach q and then returned to the boundary of the region to reach p. So Π could be shortened by not entering R', which contradicts that Π is a shortest path. ☐

A useful corollary follows from the case distinction of this lemma:

Corollary 67. *Let Γ be a regionalized grid graph, let u be a point in a region R, and let G be a gate bounding R, on which u does not lie.*

If the closest vertex on G to u is not a landmark, the shortest path from u to G is a straight line.

Next, we argue that in a simple region R and given a point u in R, it must be the case that there's a unique closest point to u on any gate bounding R. We start by giving a slightly more general lemma which then immediately implies this.

Lemma 68. *Let Γ be a regionalized grid graph. Let u and v be vertices in the same region R, let v lie on a gate bounding R and let v_1 and v_2 be the neighbors of v on its portal. Then $d_\Gamma(u, v_1) > d_\Gamma(u, v)$, or $d_\Gamma(u, v_2) > d_\Gamma(u, v)$.*

Proof. Let Π_1^* be a shortest path in R from u to v_1, and let Π_2^* be a shortest path in R from u to v_2. Let p be the last point that Π_1^* and Π_2^* have in common, when traversing both paths from u to their respective destinations, and let the subpath of Π_1^* starting at p and ending at v_1 be Π_1 (resp. Π_2). Note that if v is a point on either of Π_1 or Π_2 we are done (as we just terminate the path in question at v—it must be shorter than the full path), and so assume it is not: this implies that the area enclosed by Π_1, (v_1, v), (v, v_2), and Π_2 is a simple cycle in the grid graph (as otherwise, Π_1 or Π_2 could be shortened).

Consider a line segment ℓ which starts at v and extends into R, perpendicular to P_v, until it intersects either Π_1 or Π_2. Note that all points on ℓ are within the simple cycle formed by the paths and vertices v_1, v, v_2, and therefore all vertices along ℓ exist (i.e. none lie within holes). Suppose without loss of generality that ℓ intersects Π_1 at point v^*, and the length of ℓ from v to v^* is x.

Now consider the path formed by concatenating the path Π_1^* from u up to the point v^*, with the line segment ℓ (from v^* to v). Since $d_\Gamma(v^*, v_1) \geq x+1$ but ℓ is of length x, this path is shorter than Π_1^* and goes from u to v, proving the lemma. □

Corollary 69. *Let Γ be a regionalized grid graph. Let u be a vertex in a region R, and let G be a gate incident to R.*

There is a unique *closest point to u on P. Let this point be v. Given another point w on the same portal as v, $d_\Gamma(u,w) = d_\Gamma(u,v) + d_\Gamma(v,w)$.*

Next, we show that the edges that we have selected to be part of our landmark graph are sufficient to represent shortest paths between all pairs of landmarks which border a region.

Lemma 70. *Let u and v be landmarks in the same region R. Some shortest path in Γ from u to v can be expressed as the concatenation of the underlying paths of edges in E_Λ.*

Proof. This follows from Lemmas 66 and 68. First, recall that since R is path-convex there exists some shortest path between u and v in Γ which lies entirely in R.

Let P_u and P_v be the portals on which u and v lie. If $P_u = P_v$, then the lemma statement is trivially true, since there are edges in E_Λ between adjacent landmarks on the same portal, and the shortest path between u and v is along the portal on which they both lie.

If $P_u \neq P_v$, then let v^* be the closest point on P_v to u, which, by Lemma 66, is a landmark (and hence (u, v^*) is an edge in E_Λ). We claim that some shortest path between u and v first goes from u to v^*, and then goes from v^* to v along P_v; equivalently, $d_\Gamma(u,v^*) + d_\Gamma(v^*,v) = d_\Gamma(u,v)$. This follows easily from Corollary 69, and this path can be constructed using edges in E_Λ. □

Finally for this section, we compute the size of the landmark graph. This will be particularly useful when we consider the computational complexity of distributing our landmark graph and setting up our routing scheme in Section 6.

Lemma 71. *Given a regionalized grid graph Γ with $|\mathcal{H}|$ holes and a set of landmarks V_Λ as in Definition 59, the following properties all hold:*

- *Each landmark is adjacent to $O(1)$ regions.*
- *Each region has $O(|\mathcal{H}|)$ adjacent landmarks.*
- *The landmark graph Λ has $|V_\Lambda| = O(|\mathcal{H}|^2)$ nodes.*
- *The landmark graph has $|E_\Lambda| = O(|\mathcal{H}|^2)$ edges.*

Proof. We begin by observing that since landmarks are always placed on portals (cf. Definition 59), and each portal bounds at most a constant number of regions (follows from the splitting procedure), the first property holds.

For the second and third properties, we count the three different origins of landmarks separately:

- **Endpoints of a gate:** From Theorem 26 we know that $|\mathcal{R}| = O(|\mathcal{H}|)$. Consider the graph where vertices are regions and there is an edge between regions for each portal that connects them. Clearly this graph is planar and therefore the number of portals bounding regions is $O(|\mathcal{H}|)$ as well. Trivially, since each region is bounded by a constant number of portals, each region has a constant number of adjacent landmarks of this type.
- **Overhang-induced regions:** By the definition of overhangs there can be at most two overhang-induced landmarks for each pair of horizontal (resp. vertical) gates bounding a region. Since the number of gates bounding a region is constant, there are a constant number of overhang-induced portals per region, and the total over the graph is $O(|\mathcal{H}|)$.
- **Projected landmarks:** Clearly we must have $O(|\mathcal{H}|^2)$ of these, since there are $O(|\mathcal{H}|)$ portals of the other two types in total and each could be "projected" onto any gate bounding any other region. There are $O(|\mathcal{H}|)$ regions and each is bounded by $O(1)$ gates, giving the claimed bound.

The number of landmark edges in each region is $O(|\mathcal{H}|)$, as by Observation 61 each landmark has at most a constant number of neighbors in each region. Since there are $O(|\mathcal{H}|)$ regions, the bound in the fourth property follows immediately. □

5.2. Routing properties of the landmark graph

Next, we show that our choice of where to place landmarks give us useful properties for routing. We conclude the section with the proof of Theorem 63.

First, we show that, if we fix a sequence of regions through which a path must travel, a shortest path can be obtained by repeatedly routing to the closest point in the next region in the sequence.

Lemma 72. *We say that an s-t path is* shortest-possible *relative to a sequence of pairwise-adjacent regions $(R_0, R_1, \ldots R_m)$, where $s \in R_0$ and $t \in R_m$, if no shorter path passes through the same sequence of regions.*

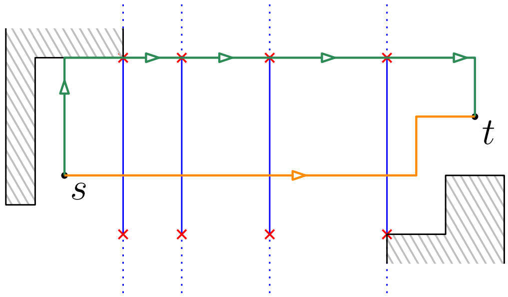


Fig. 10. An example of the case where there are no landmarks on a shortest path from s to t.

A shortest-possible s-t path can be obtained by routing from s to the closest point in R_1, then to the closest point in R_2, and so on: and when finally R_m is entered, taking a shortest path to t.

Proof. Let Π^* be a shortest-possible path from s to t, and let Π be a path which follows the strategy outlined in the lemma. We will show that $|\Pi| = |\Pi^*|$. We will notate the length of path Π between points x and y as $|\Pi(x, y)|$ (and analogously for the length of Π^*).

Suppose for each R_i, Π^* enters R_i at some point v_i^*, and Π enters R_i at v_i. Note that since pairs of adjacent regions are connected by a single portal, v_i and v_i^* are on the same portal. It suffices to show that $|\Pi^*(s, v_m^*)| = |\Pi(s, v_m)| + d_\Gamma(v_m, v_m^*)$, since when routing to t in the final step, clearly $d_\Gamma(v_m, t) \le d_\Gamma(v_m, v_m^*) + d_\Gamma(v_m^*, t)$ (as we can route via v_m^* if this is shortest).

We prove this by induction. Note that as a consequence of Corollary 69, we have that $|\Pi^*(s, v_1^*)| = |\Pi(s, v_1)| + d_\Gamma(v_1, v_1^*)$, hence the base case is satisfied. Now by two applications of Corollary 69 we have, for all $v_i, v_{i+1}, v_i^*, v_{i+1}^*$, a kind of rectangle equality:

$$d_\Gamma(v_i, v_{i+1}) + d_\Gamma(v_{i+1}, v_{i+1}^*) = d_\Gamma(v_i, v_i^*) + d_\Gamma(v_i^*, v_{i+1}^*)$$

And this gives us:

$$\begin{aligned}
|\Pi^*(s, v_{i+1}^*)| &= |\Pi^*(s, v_i^*)| + d_\Gamma(v_i^*, v_{i+1}^*) \\
&= |\Pi(s, v_i)| + d_\Gamma(v_i, v_i^*) + d_\Gamma(v_i^*, v_{i+1}^*) \\
&= |\Pi(s, v_i)| + d_\Gamma(v_i, v_{i+1}) + d_\Gamma(v_{i+1}, v_{i+1}^*) \\
&= |\Pi(s, v_{i+1})| + d_\Gamma(v_{i+1}, v_{i+1}^*)
\end{aligned}$$

as required. □

Next, we show that if *any* shortest path between two nodes contains a landmark, then there is a shortest path between those two nodes which is a landmark path.

Lemma 73. *If there is a landmark on some shortest path in the grid graph from s to t, then some shortest path in the grid graph from s to t is a landmark path.*

Proof. Firstly, note that if v is a point on a shortest st-path, then this shortest st-path is composed of a shortest sv-path and a shortest vt-path.

Then, consider a landmark λ on a shortest st-path Π, and let $\Pi_{\lambda t}$ be the subpath of Π from λ to t. We can replace this subpath by a new path $\Pi_{\lambda t}^*$ obtained by the strategy given in Lemma 72, using the regions passed through by $\Pi_{\lambda t}$ to ensure the cost does not increase relative to $\Pi_{\lambda t}$. Note that our path starts at a landmark: by Lemma 66, each time we route to the closest point in the next region in the sequence, this point must also be a landmark. As Lemma 70 shows that we can replace shortest paths between landmarks by edges in E_Λ, we can see that our path $\Pi_{\lambda t}^*$ is a landmark path.

We can apply the same construction to replace $\Pi_{s\lambda}$ with a landmark path $\Pi_{s\lambda}^*$, and finally it is easy to see that concatenating $\Pi_{s\lambda}^*$ with $\Pi_{\lambda t}^*$ gives a landmark path of no greater length than Π: that is to say, this landmark path is a shortest st-path in the grid graph. □

We conclude the subsection with the proof of Theorem 63, that the shortest st-landmark path crosses the same sequence of regions as a shortest st-path. For a visual explanation of the second case of the following proof, see Fig. 10.

Proof of Theorem 63. First, note that if any shortest s-t path contains a landmark, then by Lemma 73 we are immediately done.

Suppose no shortest s-t path contains a landmark, and fix an arbitrary shortest path Π'. Let Π be a shortest-possible path relative to the same sequence of regions $(R_1, R_2, \dots R_m)$ induced by Π'. By Lemma 72, we can construct Π such that it is comprised of a shortest path from s to the region R_1, followed by a shortest path to the region R_2, etc., when the path enters R_m, it is then followed by a shortest path to t.

We now argue that, since Π does not pass through any landmarks, it must have a very specific structure. For each R_i, let v_i be the first node on Π in R_i. Since Π does not contain a landmark by assumption, by Corollary 67 the shortest path from s to v_1 must be a straight line. This logic also applies to the shortest path from v_1 to v_2, and from v_2 to v_3, all the way to v_m, and note further that all of the portals on which the v_i lie must be parallel to each other (if the path were to "turn" between some v_i and v_{i+1}, it would necessarily reach a landmark by contrapositive of Corollary 67). Finally, let the closest point to t on the gate separating R_{m-1} from R_m be t^*. Note that a shortest path from v_m to t may go via t^* first (by Lemma 69). To summarize: Π consists of a straight line from s to v_m, and then a path from v_m to t (via t^*).

It remains to show that the shortest landmark path induces the same sequence of regions. Suppose without loss of generality that the gates which Π crosses are all vertical, and let $\lambda_i^{\uparrow}$ and $\lambda_i^{\downarrow}$ be the closest landmarks above and below v_i on its gate, respectively. We first argue that all of the $\lambda_i^{\uparrow}$ lie in a horizontal line (and that all points along this line are vertices in the grid graph). We argue that the shortest path between $\lambda_1^{\uparrow}$ and $\lambda_2^{\uparrow}$ is a straight line: the argument applies analogously to all other $\lambda_i^{\uparrow}$ and all $\lambda_i^{\downarrow}$. Let v_1^i be the vertex i steps north of v_1. Iteratively consider the sequence of pairs of points $((v_1^1, v_2^1), (v_1^2, v_2^2) \dots)$. If any of the points on the line between v_1^i and v_2^i are incident to a hole boundary, then $v_1^i = \lambda_1^{\uparrow}$ and $v_2^i = \lambda_2^{\uparrow}$ (by case (ii) of Definition 59: the points are landmarks). Otherwise, suppose that (without loss of generality), $v_1^i = \lambda_1^{\uparrow}$. Then, by case (iii) of Definition 59, $v_2^i = \lambda_2^{\uparrow}$ (by projection). If neither of these holds, then proceed to (v_1^{i+1}, v_2^{i+1}). Eventually the end of one of the portals will be reached and both landmarks will be found at the same "height".

We claim that t is neither more north than $\lambda_m^{\uparrow}$ nor more south than $\lambda_m^{\downarrow}$. To see why, recall that t^* is the closest point to t on the gate containing v_m. If there was a landmark between t^* and v_m on their gate, it would lie on Π, which is a contradiction. Therefore the line segment (t^*, v_m) does not cross $\lambda_m^{\uparrow}$ or $\lambda_m^{\downarrow}$. Combining this with an application of Corollary 67 shows our claim to be true.

Finally, we show that the shortest landmark path induces the same sequence of regions as the shortest st-path. Consider the two landmark paths $L^{\uparrow}$, $L^{\downarrow}$, with $L^{\uparrow}$ going from s to $\lambda_1^{\uparrow}$ to $\lambda_2^{\uparrow} \dots$ to $\lambda_m^{\uparrow}$ to t, and $L^{\downarrow}$ defined analogously. Note that both of these paths induce the series of regions $(R_1, R_2, \dots R_{m-1}, R_m)$. Both of these paths will travel the same distance horizontally (the minimum possible), but one may travel less than the other vertically. We observe that, since there are no landmarks inside the box bounded by $\lambda_1^{\uparrow}$, $\lambda_m^{\uparrow}$, $\lambda_1^{\downarrow}$, and $\lambda_m^{\downarrow}$, any other landmark path must travel at least the same distance horizontally but a greater distance vertically. □

5.3. *Computation of the landmark graph*

We conclude the section on the landmark graph by showing that the landmark graph can be computed efficiently in the HYBRID model, proving the two lemmas to that effect which we gave at the beginning of the section.

Proof of Lemma 64. Consider the three cases where a node could be a landmark in Definition 59:

- **Endpoints of a gate:** Nodes already know whether they are the endpoints of a gate: they become aware of this during the region decomposition process.
- **Overhang-induced landmarks:** Nodes on gates know the ID of the hole boundary or gate which the portal perpendicular to the gate terminates at. Nodes know whether they are incident to a hole boundary and can broadcast this information to all nodes on their incident portals in $O(\log n)$ rounds. After this, nodes on gates can locally determine whether they are overhang-induced landmarks as they have all the necessary information.
- **Projected landmarks:** Nodes can broadcast whether they are a landmark of the first two types to the rest of the portals on which they lie in $O(\log n)$ rounds, per Lemma 100 (note that it suffices to know that *some* node on the same portal is a landmark). Nodes already know, from the region decomposition, whether they lie on a gate. □

Proof of Lemma 65. We claim that we can compute E_Λ with a constant number of applications of Corollary 22. This approach gives a running time of $O(\log n)$ rounds), so it remains to argue that this suffices. Note that we don't compute the underlying paths of edges in E_Λ (we address this in Section 6), it suffices to know *which* landmarks the edges are between.

First, note that nodes can easily identify edges in E_Λ which go between landmarks on the same portal in $O(\log n)$ rounds, using simple HYBRID primitives to discover the closest landmarks to them on their own gate. It remains to show that edges in E_Λ which go between landmarks on different gates can be computed in $O(\log n)$ rounds.

Fix an arbitrary region R, fix a gate G incident to R, fix a landmark λ which is not on G, and let λ_G be the closest point to λ on G, which by Lemma 66 is also a landmark. We argue that it suffices to run SSSP starting at each landmark of the first two types of Definition 59 on G, and that we do not need to run it for landmarks of the third type on G.

To see why, suppose that λ_G is a landmark of the third type (a projected landmark). Therefore, by reasoning in Lemma 66, there is a straight line path from λ to λ_G. Since the node at λ_G knows that it is a projected landmark, it can broadcast this information to all nodes on the portal on which it lies that is perpendicular to G. Since λ lies on this portal, and both nodes know that they are landmarks on gates incident to R, both nodes will know that there is an edge between them in E_Λ.

Finally, we observe that there are a constant number of landmarks of the first two types on each gate. This means that, for each gate, we need only run a constant number of SSSP computations to compute all edges incident to the gate. Since there are at most a constant number of gates per region, this gives a total running time of $O(\log n)$. □

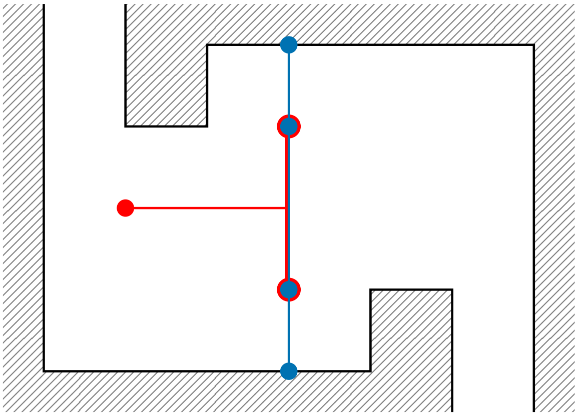


Fig. 11. A region boundary with landmarks (blue) with a node and the corresponding important landmarks (red). To obtain these, consider the closest point on the region boundary to the node and mark the closest landmark in each direction.

6. Routing

In this section we combine our previous results and use them to formulate our routing scheme. Recall that, according to Definition 1, for a routing scheme each node v needs to learn its label $\lambda(v)$ and routing function $\rho(v)$. After computing the region decomposition and the landmark graph as presented in the previous sections, the nodes use these structures to learn the information required for their own label and routing table. Specifically, the routing tables will be identical for each node and will consist of a version of the landmark graph that is labeled with additional information linking it to the actual grid graph. This means that each node has to learn the landmark graph and these additional labels. To use the landmark graph to make routing decisions, the nodes need to add themselves and the target node to the landmark graph. This requires them to know the distances to close landmarks for both themselves and the target node. Hence we make this information part of the node labels. To distinguish different cases of the algorithm, the node's label additionally contains its region identifier and its node identifier.

In the following we start by proving that the nodes do not need to learn *all* landmarks adjacent to their region in order to themselves and their nearby landmarks to their copy of the landmark graph. Instead, it suffices to learn two landmarks (see Definition 77) for each gate adjacent to the region. As each gate can have up to $O(|\mathcal{H}|)$ many landmarks, only have to learn two of them significantly reduces the size of the node labels required. Afterwards, we describe how the nodes compute and distribute the data required for the node labels and routing tables. Finally, we describe how a node can use the data it learned this way to forward a packet towards a target node.

To prepare for this, we first fix the following observations that are commonly used in the proofs in this section:

Observation 74. Each node is part of $O(1)$ regions. This follows from the constructions of Definition 29, Definition 38 and Definition 58.

Observation 75. Each region has $O(1)$ gates. This follows from the facts that tunnels have exactly 2 gates; that according to Lemma 46 we split tunnels into at most 10 regions; and that each of the splits is entirely horizontal or vertical.

Observation 76. The degree of each landmark is $O(1)$. This is because Definition 60 has two rules introducing edges to the landmark graph. The first connects landmarks closest to each other on the same gate and the second connects landmarks closest to each other on different gates. As there are $O(1)$ gates per region according to Observation 75, both rules add only a constant number of edges to the landmark graph for each landmark.

We first use the regionalization algorithm described in Section 4 to divide the graph into simple, path-convex regions. Then, we build a landmark graph as described in Section 5. Together, these constructions take $O(|\mathcal{H}| + \log n)$ time (cf. Theorem 26 and Lemma 65).

Afterwards, we describe which landmarks adjacent to their region the nodes need to include in their label, and how they can learn them. To enable nodes to make routing decisions, the label of the target node must contain the landmarks adjacent to the target node's region. As there could be $O(|\mathcal{H}|)$ landmarks that fit this description, we need to find a small subset of these landmarks that suffice without introducing any stretch. We formalize this notion of important landmarks with the following definition and show that they suffice for routing.

Definition 77. Given a node v, for each gate adjacent to v's region, consider the unique (Corollary 69) closest node to v. We call the closest landmark in each direction induced by the gate *important* (see Fig. 11).

Lemma 78. *For each landmark path starting from a node v and ending with a node t, such that v and t are in different regions, there is a landmark path of equal length that has important landmarks as its first and last landmarks, i.e., the landmark path starts with v and an important landmark of v and ends with an important landmark of t followed by t. For each node, there are $O(1)$ important landmarks in total.*

Proof. Let v' be the closest node to v on a given gate. Corollary 69 yields that for any landmark λ on that gate, $d(v,\lambda) = d(v,v') + d(v',\lambda)$. If λ is non-important, there must be an important landmark λ' closer to v'. As λ and λ' are on the same gate, we can conclude $d(v,\lambda) = d(v,v') + d(v',\lambda') + d(\lambda',\lambda)$. Hence, given a landmark path starting with v and λ, we obtain a landmark path of equal length starting with v and λ'. The argument for the end of the landmark path is analogous. The fact that there are $O(1)$ important landmarks follows directly from Observation 75, since there are at most 2 important landmarks per gate. □

Hence, it suffices to include these important landmarks in the node labels. To allow the nodes to make routing decisions using only locally stored information, we need to further augment the landmark graph with some labels. To this end, we start by computing a region identifier for each region and let each node learn its region's identifier.

Lemma 79. *Each node can learn the identifier of its region, i.e. the minimum identifier of any (virtual) node in the region in time $O(\log n)$. Each region identifier is unique and of size $O(\log n)$.*

Proof. Our goal is to pick the minimum identifier of any node in a region as its region identifier. Per Lemma 71, each node may be part of multiple regions. Hence, each occurrence of a node must have a unique identifier. To achieve this, any node that is part of multiple regions picks a unique suffix for each of them, and then appends this suffix to the identifier of the node. Since each node is part of a constant number of regions, the resulting identifiers are still of length $O(\log n)$.

Now, we can aggregate the resulting identifiers to find the minimum. To aggregate among a region, we perform two aggregations and a broadcast. In the first aggregation, each node learns the minimum identifier of its horizontal portal. In the next step, each node of the region boundary aggregates the results of the first step. This way, each boundary node learns the minimum identifier. In the third step, the rightmost node of each horizontal portal (that must be part of the region boundary) broadcasts that minimum to each node of that portal.

Since all these operations can be performed in $O(\log n)$ using pointer-jumping (Lemma 100), we have a total runtime of $O(\log n)$. □

Next, we have the nodes learn the distance to their adjacent gates and we use the SSSP tree resulting from this execution, to further obtain their distances to their important landmarks.

Lemma 80. *Each node can learn the distance to each gate adjacent to its region and its neighbor that is closest to that region in time $O(\log n)$.*

After learning about their adjacent gate, each node can infer the distance to each of its important landmarks in time $O(\log n)$. $O(\log n)$ bits suffice to store all distances learned this way.

Proof. As each region has $O(1)$ gates according to Observation 75, we can perform the algorithm from Corollary 22 a constant number of times to achieve the first claim.

For the second claim, using that we only have a constant number of gates per region again, we can describe the algorithm for a single gate.

After the execution of the algorithm of Corollary 22 each node learns its distance to the gate and its predecessor (Observation 23). Specifically, we obtain a shortest path tree structure connecting each node in an adjacent region to the gate. Next, we have all non-landmark gate nodes learn the distances to the closest landmark in each direction. This can be achieved in $O(\log n)$ time by performing pointer-jumping on the gates. Finally, each gate node performs the Euler tour technique of Lemma 101 on its subtree of the gate's shortest path tree. This allows it to distribute its distance values to the closest landmark in each direction in time $O(\log n)$. By adding the distances to the gate to the distances of gate nodes to their closest landmarks, the nodes can infer the distance to their important landmarks. □

To enable the nodes to make routing decisions, they need to learn which gate leads to which region and we require further labeling of the gates and the edges and landmarks of the landmark graph. The next three lemmas describe how to establish the required information.

Lemma 81. *After establishing region identifiers and performing a closest point computation, gate nodes can distribute their knowledge on what regions they are adjacent to all nodes closest to them in $O(\log n)$ rounds.*

Proof. Due to the establishment of region identifiers in Lemma 79, each gate node knows which regions it is adjacent to. Additionally, each gate node obtains an SSSP subtree during the closest point computation. They can execute the euler tour technique (Lemma 101) on these subtrees to transform them into a path graph and employ pointer-jumping (Lemma 99) to distribute the identifiers of the regions they are adjacent to. □

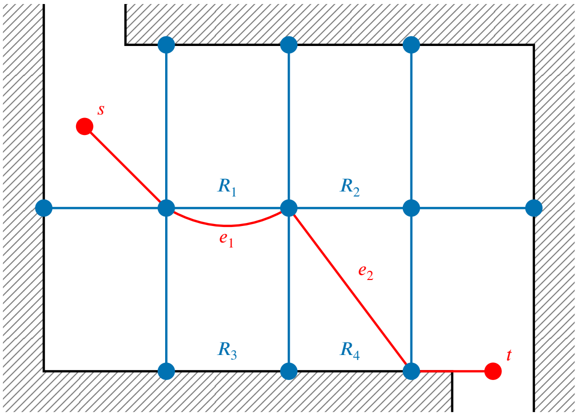


Fig. 12. A packet is to be sent from s to t. As the edge e_1 is aligned with the horizontal gate, it is considered to be in both region R_1 and region R_3. As R_1 is adjacent to the region containing s, it is the next region. Once the packet reaches R_1, e_2 will be considered to determine the next region. e_2 entirely lies within region R_4, which is not adjacent to R_1. Hence it is possible to either cross the gate towards region R_2 next or to cross the gate towards R_3 next.

Lemma 82. *Each node on a gate can learn the label of its gate (i.e. the minimum identifier of any node on it and a bit for tiebreaking) in time $O(\log n)$. Each gate identifier is unique and of size $O(\log n)$.*

Proof. The nodes of each gate perform pointer-jumping aggregating the minimum of their identifiers. As each node can be part of a horizontal and a vertical gate, we add a bit indicating the orientation for tiebreaking. The resulting identifiers are of size $O(\log n)$ and are obtained in $O(\log n)$ rounds. The gate identifiers can be distributed the same way the adjacent regions were distributed in Lemma 81. □

Lemma 83. *After establishing region identifiers and gate identifiers, we can label each edge of the landmark graph with the region identifiers of the regions the edge lies in and each landmark with the gate identifiers it lies on in $O(1)$ rounds. The total size of these labels is $O(|\mathcal{H}|^2 \cdot \log n)$ bits.*

Proof. To establish the edge labels, adjacent landmarks communicate which regions they are part of, keeping only the regions containing both. As each landmark is of constant degree according to Observation 76 this can be done in $O(1)$ rounds. Further, each edge is part of two regions yielding a label size of $O(\log n)$. During the establishment of the region identifiers, each landmark already learned which gates it lies on. Combining the identifiers of those gates yields a label of $O(\log n)$ bits. As there are a total of $O(|\mathcal{H}|^2)$ edges in the landmark graph and a total of $O(|\mathcal{H}|^2)$ landmarks, this yields a total size of $O(|\mathcal{H}|^2 \cdot \log n)$ bits. □

With these labels established, we can now show that they indeed suffice for the nodes to make routing decisions.

Lemma 84. *The labels established in Lemma 81, Lemma 82 and Lemma 83 suffice to allow a node to decide which gate should be crossed next.*

Proof. After computing the landmark path locally, v checks the edge label of its first edge e that does not lie within v's region. If there is a gate connecting v's region to one of e's regions (it has multiple regions, if both of its endpoints are on the same gate), that gate is the next gate. Otherwise the landmark path crosses the landmark before e diagonally (i.e. the landmark is at the intersection of two gates the next region is diagonally opposed to the current one) and v picks an arbitrary gate connected to that landmark. This process is depicted in Fig. 12. □

Next, we have every node learn the entire landmark graph including these labels. This gives each node enough knowledge about the grid graph to make local routing decisions.

Lemma 85. *Each node can learn the landmark graph and the labels from Lemma 83 in time $O(|\mathcal{H}|^2 + \log n)$.*

Proof. To achieve the runtime of the theorem, we employ the technique from Lemma 100. As it requires a path graph to work, we start by establishing one using the construction of Lemma 98. We can use that path graph to fix an ordering on the landmarks by performing pointer-jumping on the non-landmark nodes to reduce the distances between landmarks to $O(\log n)$ and sending the landmarks' identifiers over the established shortcuts afterwards. This takes $O(\log n)$ rounds. As each landmark now knows its successor, the first landmark can start broadcasting its neighbors over the butterfly network. Once it is done, it can inform that successor to do the same. By pipelining this process, and as $O(|\mathcal{H}|^2)$ messages of size $O(\log n)$ have to be sent according to Lemma 71 and Lemma 83, the entire broadcast takes $O(|\mathcal{H}|^2 + \log n)$ rounds. □

Applying the algorithm from [9], we obtain the following corollary.

Corollary 86. *For each simple region, we can compute exact routing tables in time $O(\log n)$.*

We conclude with a corollary combining the round complexity of each stage of the pre-processing (by Theorem 26, Lemma 65, and the results in this section):

Corollary 87. *The preprocessing takes $O(|\mathcal{H}|^2 + \log n)$ time.*

As well as the landmark graph, nodes require some more information about the target node to successfully route a packet. Specifically, each node needs to know the target's distance to its adjacent landmarks, and its region label. We include these in the node labels and assume it to be included in a routing request.[12] Each node label consists of the node's distance to important landmarks ($O(\log n)$ bits, Lemma 71) and its region label ($O(\log n)$ bits).

Additionally to the node labels, each node makes use of some additional information to forward a packet. Specifically, it uses its distance to important landmarks (already part of node label), its distance to region boundaries ($O(\log n)$ bits), its region routing tables ($O(\log n)$ bits) and the Landmark graph ($O(|\mathcal{H}|^2 \cdot \log n)$ bits, Lemma 71). All of this information has been collected as a part of the preprocessing and is now stored locally by the nodes. This directly implies the following corollary:

Corollary 88. *Node labels have $O(\log n)$ bits and routing tables have $O(|\mathcal{H}|^2 \cdot \log n)$ bits.*

To present our routing scheme, we describe how a node $v = (v.id, v.D, v.rid) \neq t$ forwards a packet with destination $t = (t.id, t.D, t.rid)$. $v.id$ and $t.id$ correspond to the nodes' identifiers, $v.D$ and $t.D$ are the distances to their important landmarks and $v.rid$ and $t.rid$ denote their region identifier. See Algorithm 1 for the pseudocode.

If the packet is already in the correct region, i.e. v and t have the same region identifier, we can use the routing algorithm for simple grid graphs given in [9]. Otherwise, v locally augments its landmark graph by adding itself and t to it, and adding connections to the adjacent landmarks according to the distance values stored in $v.D$ and $t.D$. Using the augmented landmark graph, v can locally compute a shortest vt-path and thereby infer the next gate (as in Lemma 84). Each node can learn which of its neighbors is closest to a specific gate of its region in $O(\log n)$ rounds (see Lemma 80), it can forward the packet along a shortest path towards that gate.

To conclude Theorem 4, we note that Theorem 63 yields that we pick the correct sequence of gates and Lemma 72 yields that repeatedly routing the package to the closest point of the next gate in that sequence results in a shortest path. The runtime, the label size and the additional memory requirement result from Corollary 87 and Corollary 88. Finally, [9, Theorem 3.8] yields the extension to UDGs described in Corollary 5.

Algorithm 1 `forward-packet`.

```
if v.rid = t.rid then
    Forward packet using region routing table
else
    Augment landmark graph using v.D and t.D
    Compute shortest path from v to t in augmented landmark graph
    Send packet towards next gate according to shortest path
```

7. Conclusion

We believe that there are several interesting directions for interesting follow-up work. Efficiently computing compact routing schemes in more general classes of geometrically interesting graphs (for example planar graphs or visibility graphs) is a natural next step. We suspect that an extension to 3 and higher dimensions might be quite difficult (in particular the geometry required will certainly be more challenging), but the 3-dimensional case could have practical applicability in sensor networks and swarm robotics.

CRediT authorship contribution statement

Sam Coy: Conceptualization, Formal analysis, Writing – original draft. **Artur Czumaj:** Conceptualization, Formal analysis, Writing – review & editing. **Christian Scheideler:** Conceptualization, Formal analysis, Writing – review & editing. **Philipp Schneider:** Conceptualization, Formal analysis, Writing – original draft. **Julian Werthmann:** Conceptualization, Formal analysis, Writing – original draft.

[12] Note that each node knows all information required to construct its own label and any node knowing that node's identifier could then request this label via a global connection.

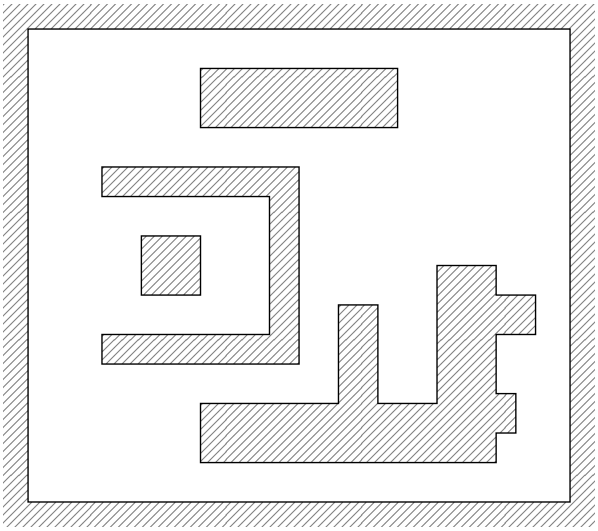


Fig. A.13. Abstract representation of the grid graph (white) and the holes (hatched). Both can be arbitrarily shaped.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Intuition for the main algorithms

In this section, we present a collection of figures that depict the different stages of the execution of our main algorithms, i.e., the preprocessing algorithm and the routing algorithm. Our goal is to aid the reader by giving intuitive presentations of what actions are performed during each of them. Hence, the section is written to be self contained and understandable without reading the definitions presented in the paper, and we do not worry about computability arguments or analytically well-defined wording here. For a rigorous presentation, we refer the reader to the main part of the paper.

A.1. Setting

As meaningful examples of grid graphs become too large to present really quickly, we opted for a more abstract representation, drawing a white area for grid spaces and a hatched area for inner holes and the outer hole. As suggested in Fig. A.13, we do not assume any structure for our grid graph (except connectedness), i.e., the induced polygon can have an arbitrary shape and contain any amount of arbitrarily shaped holes.

A.2. Regionalization

During the Regionalization, we divide the grid graph into regions that do not contain any holes and are path-convex, i.e., for any two nodes in a region, there is a shortest path connecting those nodes wholly within that region. To achieve this, we first mark the leftmost node of each hole (breaking ties by picking the northernmost one) and cut the grid graph vertically through that marked node and horizontally to its right. As each hole is connected to another hole this way and as we can not create cyclic connectedness due to the leftmost node being used, we obtain hole-free regions this way.

To satisfy path-convexity, we first split the resulting regions into tunnels, i.e. regions which have two dedicated exits that form vertical portals (a vertical portal is a maximal vertically connected component in the grid graph, see Fig. A.14). Tunnels are obtained by identifying portals where the region "diverges" into multiple directions, which we call junction portals, and splitting the region at those. Finally, to make the tunnels path-convex, we further subdivide them by making additional cuts to ensure that their length is effectively halved. Any shortest path between nodes of the resulting regions would have to go through at least one of the other regions, intuitively meaning that it would spend more than half of the regions diameter outside the region, thus we can find a shorter path inside the region. Each step is depicted in Fig. A.14.

A.3. Landmark graph

To enable the nodes to locally make routing decisions, we have them learn the so-called landmark graph, which essentially is a small size skeleton graph that captures important distances of the graph and the different ways to navigate around its holes. To this

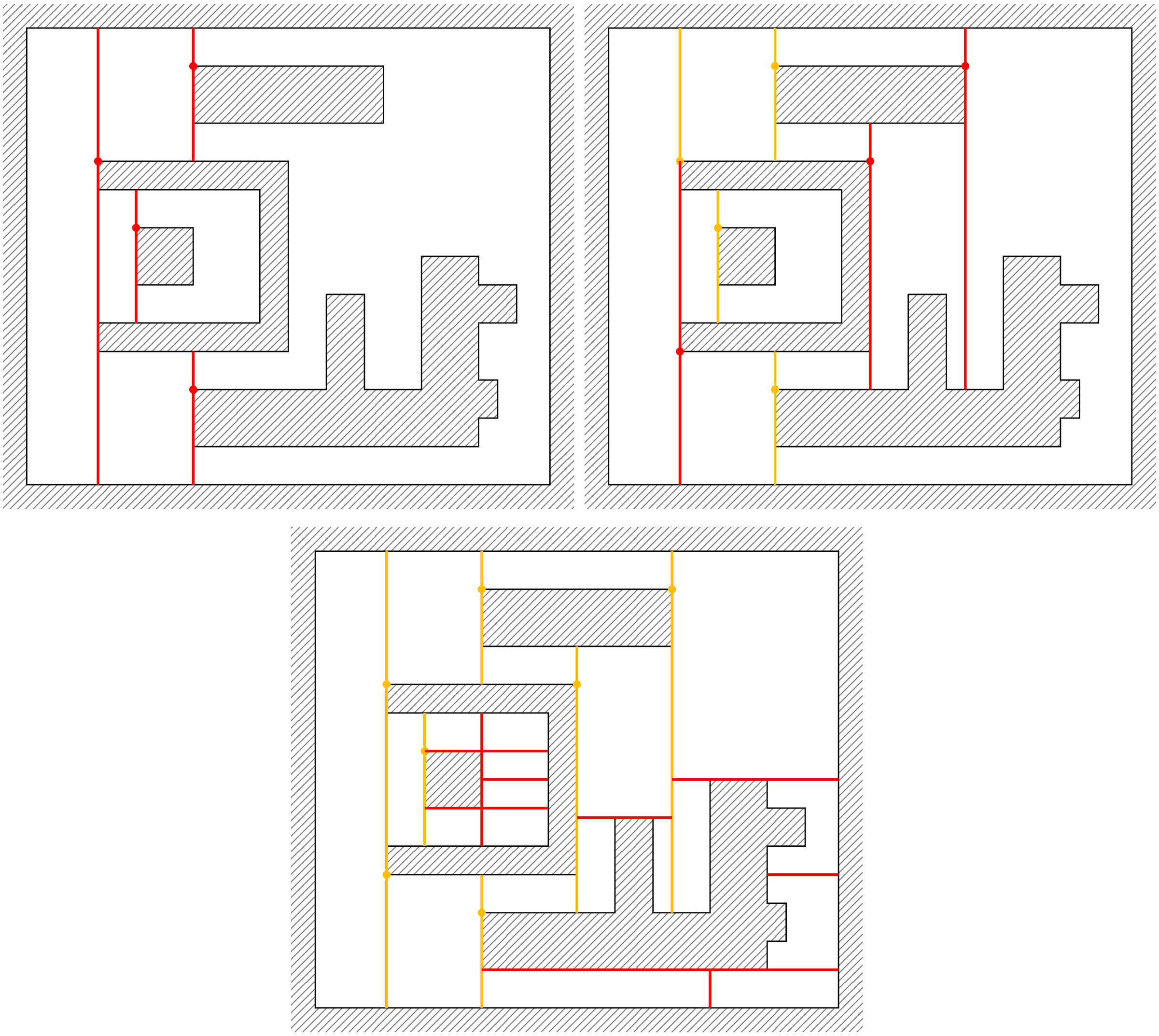


Fig. A.14. Regionalization of the grid graph. One after the other, we divide the grid graph into simple regions (top left), tunnels (top right) and path-convex regions (bottom).

end, we mark some nodes as *landmarks*, see Fig. A.15. First, we add the endpoint of each region boundary to the landmark graph. Next, for each region boundary that does not wholly see its adjacent region boundary, we consider its set of closest nodes to that other boundary and add the outermost of these nodes to the landmark graph as well.

Finally, for each landmark, we project it to all region boundaries that are orthogonally visible to it, i.e., consider the orthogonal line crossing the original node and add all nodes corresponding to crossings of that line with other region boundaries to the landmark graph. We connect two of these landmark nodes with an edge whenever they are on the same region boundary and no other landmark node is between them, or when they are on different region boundaries adjacent to the same region and they are closest to each other on those region boundaries. Fig. A.16 outlines how we connect two nodes of the landmark graphs (however, some edges are omitted to keep them easy to read).

A.4. Routing

Our routing algorithm distinguishes two cases. In the simple case, depicted in Fig. A.17, both source and target node of a routing request are in the same region. As our regions do not have holes, we can employ the routing scheme for hole free grid graphs from [9].

Fig. A.18 shows the general case, where source and target node of a routing request are in different regions. Here, we have the source node locally augment the landmark graph by adding itself and the target node to it. Then, the source node locally solves the shortest path problem on the landmark graph, to decide where the packet should exit its region. Subsequently, it forwards the packet one step towards the corresponding region boundary. After reevaluating the same computation, the receiving node forwards

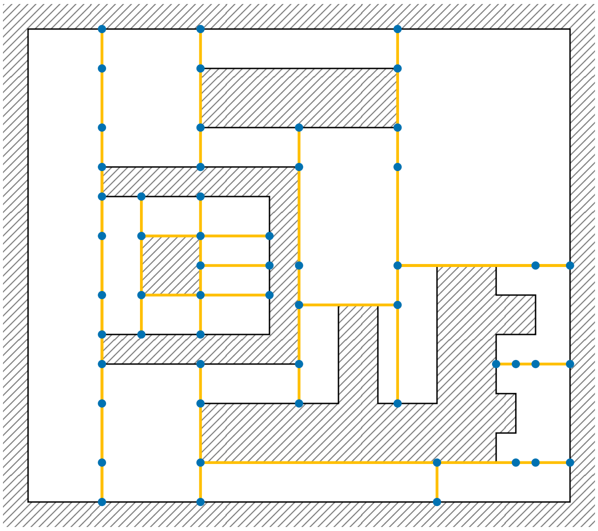


Fig. A.15. The nodeset of the landmark graph consists of the endpoints of the region boundaries, the points induced by obstructions between region boundaries and the points induced by projections.

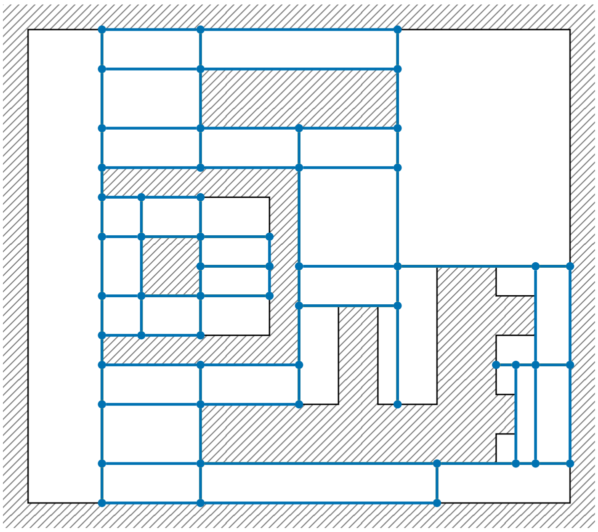


Fig. A.16. Two nodes of the landmark graph are connected, if they are on the same region boundary without another node of the landmark graph between them, or if they are on adjacent region boudaries and no other landmark on the corresponding other boundary is closer to them.

the packet again until it is passed to the next region, where it is handed over again. This is repeated until the packet arrives in the target node's region, where we fall back to the simple case from above.

Appendix B. Properties of grid graphs

In this section we prove some technical properties of grid graphs that we exploit throughout the paper. First, we provide some useful definitions and a lemma, relating to bottlenecks on paths through grid graphs.

Definition 89 *(Bottleneck).* A node in a graph is a *bottleneck* with respect to a set of paths in the graph if it appears on all paths in the set.

Definition 90 *(Monotonous path).* Given a grid graph $\Gamma = (V_\Gamma, E_\Gamma)$, a path $\Pi \subseteq E_\Gamma$ from a node u to v is monotonous if the sequence of edges, when oriented from u to v, point in at most two cardinal directions: at most one horizontal (east, west) and at most one vertical (north, south).

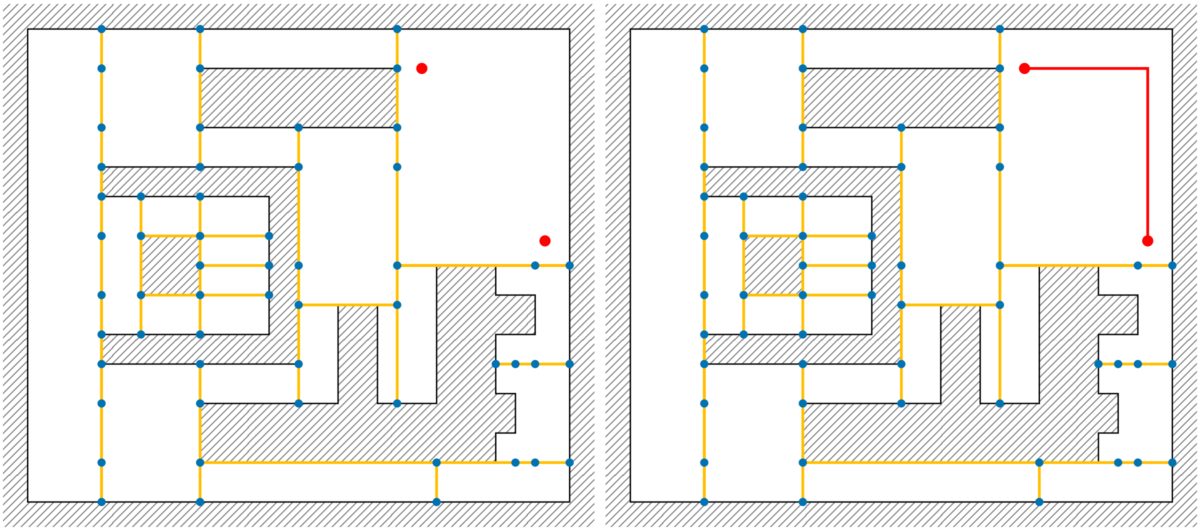


Fig. A.17. When source and target of a routing request are in the same region (left), we employ the routing scheme from [9] (right).

Lemma 91. *Let Γ be a simple grid graph and $u, v \in \Gamma$ be two nodes. If there exists a shortest uv-path that is not monotonous, then the set of shortest uv-paths have a bottleneck $w \neq u, v$ (see Definitions 90, 89).*

Proof. Let Π be a shortest uv-path that is not monotonous. Then Π contains a subpath $\tilde{\Pi} \subseteq \Pi$ that is a "turn", i.e., consists of incident edges going in three cardinal directions in an ordered fashion, w.l.o.g., first north, then east, then south. The turn $\tilde{\Pi}$ must intersect the boundary of Γ at some node $w \neq u, v$ at the most northern extent of $\tilde{\Pi}$ (and strictly more northern than u, v) with the outer hole to the south of w, because if there were no such node w the turn $\tilde{\Pi}$ could be cut short, which is a contradiction as Π is shortest.

Assume there is another shortest uv-path $\Pi' \neq \Pi$ that does not contain w. From the viewpoint of traveling from u to v, let s and t be the last intersection of Π with Π' before it reaches w, and the earliest after visiting w, respectively. Clearly s and t both exist as both Π and Π' start and end at u, v, respectively. Consider the region $R \subseteq \Gamma$ enclosed by the sub paths Π_{st} and Π'_{st} from s to t, excluding the nodes on the paths.

R cannot contain an inner hole since otherwise Γ would not be simple. Furthermore, R is located to the north of w since w has the outer hole to its south. Since Π'_{st} does not contain w it goes "strictly over" w, therefore Π'_{st} must have a turn, i.e., a consecutive sequence of edges that go north, then east, then south, with the east edges strictly to the north of w. This turn can be cut short through the simple region R which makes Π' strictly shorter, contradicting our assumption. $\square$

Lemma 92. *Let Γ be a simple grid graph and u, v two nodes. Then for any two* shortest *uv-paths Π, Π' we have that $|\Pi|_x = |\Pi'|_x = d_x(u, v)$ and $|\Pi|_y = |\Pi'|_y = d_y(u, v)$.*

Proof. It suffices to prove the claim for each sub path of Π and Π' between two nodes where both paths intersect with no other intersection node strictly between those two nodes. Therefore, we can assume that the nodes on Π and Π' intersect only in u and v. In particular, this implies that there can not be a bottleneck $w \neq u, v$ w.r.t. the set of uv-paths. Therefore the contrapositive of Lemma 91 implies that Π and Π' are monotonous, which immediately implies the lemma. $\square$

Lemma 93. *Given a grid graph Γ and nodes v, w, then for each neighbor v' of v either $d(v', w) = d(v, w) + 1$ or $d(v', w) = d(v, w) - 1$.*

Proof. It suffices to show $d(v, w) = d(v', w)$ never holds. Assume $d(v, w) = d(v', w)$. Consider any shortest vw-path and any shortest $v'w$-path. We denote the closest node to v which is part of both paths by x. By definition $d(v, x) + d(x, w) = d(v', x) + d(x, w)$ and hence $d(v, x) = d(v', x)$. Therefore the edge $\{v, v'\}$, the corresponding shortest vx-path and the corresponding shortest $v'x$-path induce a cycle of length $2d(v, x) + 1$. This yields a contradiction to the fact that any cycle in a grid graph has even length since grid graphs are bipartite. $\square$

Fact 94. Let Γ be a simple grid graph, let u, v be two nodes in Γ for which there exist two *monotonous* uv-paths Π, Π'. Presume there exist two nodes w and z on Π and Π', respectively, that have a wz-path $\tilde{\Pi}$ that goes in the same two cardinal directions (oriented from w to z) as Π (oriented from u to v). Then $\Pi_{uw} \circ \tilde{\Pi}_{wz} \circ \Pi'_{zv}$ is monotonous and thus also shortest.

Lemma 95. *Let Γ be a simple grid graph, let $u \neq v$ two nodes and let Π, Π' be two shortest uv-paths. Let w and z ($w \neq z$) be two nodes on Π and Π', respectively, with either $d_x(w, z) = 0$ or $d_y(w, z) = 0$. Let $\overline{wz}$ be the shortest wz-path (i.e., a horizontal or vertical path). Then either $\Pi_{uw} \circ \overline{wz} \circ \Pi'_{zv}$ or $\Pi'_{uz} \circ \overline{zw} \circ \Pi_{wv}$ is a shortest u, v-path.*

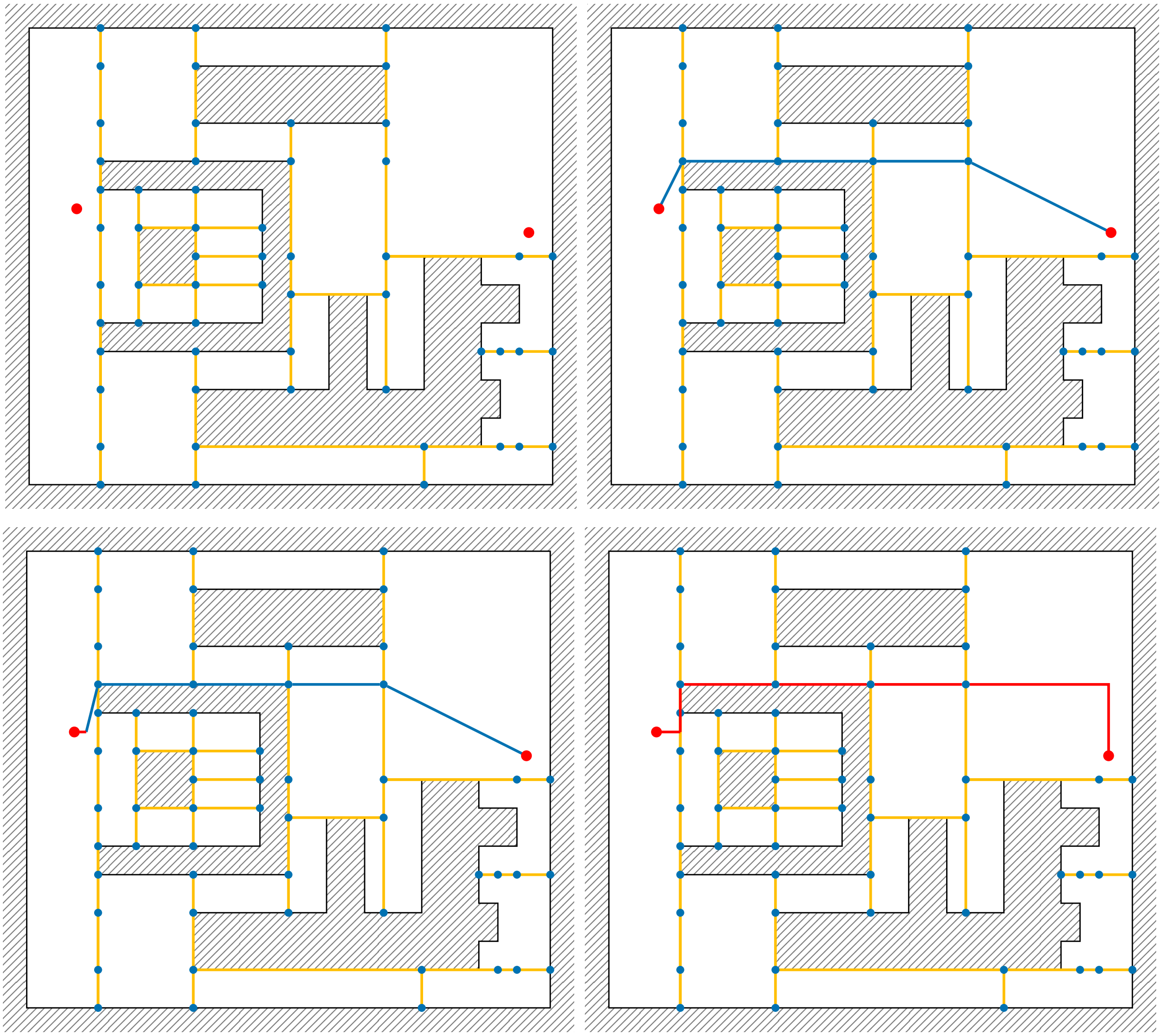


Fig. A.18. When source and target node are in different region (top left), they are added to the landmark graph, enabling each node to locally find a shortest path from the source node to the target node in the landmark graph (top right). After the source node performs this computation, it forwards the packet towards the suggested region boundary (bottom left). This is repeated until the packet the target region, where we it can be forwarded using the algorithm from [9] (bottom right).

Proof. We assume that Π, Π' only intersect in u, v (w.l.o.g., since otherwise we can redefine u and v as the closest points before and after w where Π, Π' intersect, then it suffices to prove the claim for the two sections Π_{uv}, Π'_{uv}). In particular, this means that Π and Π' do not have any bottleneck node w.r.t. the set of uv-paths other than u and v and therefore must be monotonous (contrapositive of Lemma 91).

W.l.o.g. $d_y(w, z) = 0$, i.e., w, z are on the same horizontal portal. Orienting Π and Π' from u to v the paths must use at least two cardinal directions, w.l.o.g., south and west (one must be horizontal and one vertical) as otherwise $\Pi = \Pi'$. Say z is to the east of w. Then the path $\Pi_{uw} \circ \overline{wz} \circ \Pi'_{zv}$ is shortest by Fact 94. □

Lemma 96. *Given a simple grid graph Γ with $u, v \in \Gamma$, let $\mathcal{U}_{uv}$ be the set of nodes that are on a shortest uv-path. Then for each $x, y \in \mathcal{U}_{uv}$ every node on a shortest xy-path is in $\mathcal{U}_{uv}$.*

Proof. We will prove the lemma by contradiction. Assume a shortest xy-path Π leaves $\mathcal{U}_{vw}$ directly after some node x' and joins $\mathcal{U}_{vw}$ again directly before some node y', while no nodes between x' and y' on Π are in $\mathcal{U}_{vw}$. Let now Π' be the shortest $x'y'$-path along the border of $\mathcal{U}_{vw}$. Note that the border of $\mathcal{U}_{vw}$ consists of two shortest vw-paths. If x' and y' are on the same path Π', $\Pi_{xx'} \circ \Pi'_{x'y'} \circ \Pi_{y'y}$ is shorter than Π, which contradicts Π being a shortest path. If, however, x' is on the path Π', while y' is on the other path Π'', then Π eventually has to get around v or w depending on its direction. Lemma 92 yields that Π could be shortened. □

Lemma 97. *Given a simple region Γ, let $a, b, v \in \Gamma$, Π be the shortest ab-path closest to v, i.e.,*

$$\Pi \in \underset{\Pi' \text{ shortest } ab\text{-path}}{\arg\min} \; \min_{u \in \Pi'} d_R(u, v)$$

chosen arbitrarily and v' be the unique (Lemma 93) closest point to v on Π, i.e.,

$$v' := \arg\min_{u \in \Pi} d_R(u, v).$$

Then every shortest $vv'a$-path is a shortest va-path and every shortest $vv'b$-path is a shortest vb-path.

Proof. In the following, we refer to the nodes of Π with $p_1, \ldots, p_\ell$, where $\ell := |\Pi|$.

We will prove the statement for any shortest $vv'a$-path. The shortest $vv'b$-case is analogous.

Let $\mathcal{U}$ denote the set containing all nodes on any shortest av-path, i.e. $\mathcal{U} := \bigcup_{\tilde{\Pi} \text{ shortest } av\text{-path}} \tilde{\Pi}$.

We start by proving by contradiction that Π leaves $\mathcal{U}$ only once, i.e., there is a unique point $p_i \in \Pi$ s.t. $p_i \in \mathcal{U}$, but $p_j \notin \mathcal{U}$ for all $j > i$. Assume Π would leave $\mathcal{U}$ more than once and let p_i be the first leaving node. Then there would be a reentering node p_j, $j > i$ s.t. $p_{j-1} \notin \mathcal{U}$ but $p_j \in \mathcal{U}$. According to Lemma 96, all shortest $p_i p_j$-paths are inside $\mathcal{U}$. Therefore, Π could be shortened by replacing its segment $\Pi_{p_i p_j}$ with such a shortest path. This is a contradiction to Π being a shortest path.

We now call the unique point where Π leaves $\mathcal{U}$ $\overline{v'}$. Our next goal is to show $\overline{v'} = v'$, i.e., to show that Π leaves $\mathcal{U}$ at its closest point to v. To this end, we show that $\overline{v'}$ can not be closer to a than v' and can not be further away from a than v'. We start with the former.

Assume $d(\overline{v'}, a) < d(v', a)$. According to Lemma 93 and since Π is a shortest path, the next node in Π after $\overline{v'}$ must be closer to v' than $\overline{v'}$. As v' is the closest node to v on Π it must also be closer to v. Hence it is part of a shortest $\overline{v'}v$-path and, since $\overline{v'} \in \mathcal{U}$, it is part of $\mathcal{U}$. This is a contradiction to the definition of $\overline{v'}$.

Assume $d(\overline{v'}, a) > d(v', a)$. According to Lemma 93 and by definition of v' the node after v' along Π increases the distance to v. Therefore it can not be part of $\mathcal{U}$, which contradicts the definition of $\overline{v'}$.

Hence $d(\overline{v'}, a) = d(v', a)$. Since both $\overline{v'}$ and v' are on the shortest ab-path Π, this yields $\overline{v'} = v'$. By definition of $\overline{v'}$ we conclude $v' \in \mathcal{U}$, i.e., v' is contained in the set of shortest paths from a to v. This yields the lemma. □

Lemma 98. *It takes $O(\log n)$ rounds to establish an overlay path graph in a grid graph. After an additional $O(\log n)$ rounds, the overlay path graph can be transformed into an $\lfloor \log n \rfloor$-dimensional overlay butterfly network. Both transformations are deterministic.*

Proof. Similarly to the construction of [9], we start with the vertical Portals of the grid graph and have each portal's bottommost node adds an edge to the left if able and one to the right if able. As the resulting graph may have cycles, we remove horizontal edges again until it is a tree. Specifically, each node that is on the boundary of any inner hole and has no bottom neighbor marks itself as a node on top of the corresponding hole (Lemma 104 allows us to differentiate holes). Using pointer-jumping on the hole boundaries, we can find the leftmost of these nodes. We remove the left incident horizontal edge for that leftmost node. It remains to show that the resulting structure is indeed a tree. To this end, we can consider cutting an edge as connecting the hole to the next inner hole above it or to the outer hole. Note that the nodes of each inner hole's boundary still remain connected among themselves and the topmost node of each boundary is connected via the vertical portal it is part of to another inner hole or the outer boundary. Applying this argument inductively, each node on any hole boundary is connected to the outer hole boundary. As the same statement clearly holds for nodes not on hole boundaries, we can conclude that the graph remains connected. Therefore the result of this transformation is simple and connected and by the arguments in [9] it is a tree. The transformation took $\mathcal{O}(\log n)$ time in total.

To transform this tree into a path, we use the euler tour technique from Lemma 101. Note that this technique has each node introduce a constant number of virtual nodes in our setting. To get rid of them, we have each node mark its first virtual node and have the nodes perform pointer-jumping to reduce the distance between marked nodes to $O(\log n)$ allowing us to connect them after an additional $O(\log n)$ rounds.

Finally, we employ pointer-jumping on the resulting path graph again to establish hypercubic connections. Using the butterfly simulation technique from [3, Section 2.2], we can use the hypercubic connections to have the nodes simulate a $\lfloor \log n \rfloor$-dimensional butterfly network. □

Appendix C. Subroutines

In this section we will establish a few basic subroutines for the HYBRID model that we will frequently use for our constructions for setting up the routing scheme.

C.1. Pointer jumping, broadcast and aggregation

We start by explaining an important subroutine in the HYBRID model known as pointer-jumping, which works as follows (these procedures were extensively used, e.g., in [11]). The input is a path or cycle graph of length n. In step 1, each node introduces it left neighbor (sends the ID) to its right neighbor and vice versa. Given that in step $i - 1$ each node knows the node at distance 2^{i-1}

to its left and right, they can introduce these to one another using the global network, so that now each node knows the nodes with distance 2^i to their left and right. We can think of this as creating virtual shortcut edges via the global mode of communication. After $O(\log n)$ rounds the resulting structure has diameter and maximum degree of $O(\log n)$. A more formal algorithm is provided in the following

Algorithm 2 `pointer-jumping` ▷ *Executed by each v on a path or cycle graph G.*

$\ell_{v,1}, r_{v,1} \leftarrow$ identifier of left and right neighbor of v in G or $\perp$ if it does not exist
for $i \leftarrow 2$ to $\lceil \log n \rceil$ rounds **do**
  v sends $\ell_{v,i-1}$ to $r_{v,i-1}$ and vice versa via the global network
  $\ell_{v,i}, r_{v,i} \leftarrow$ identifiers (or $\perp$) received from nodes $\ell_{v,i-1}, r_{v,i-1}$ this round

Lemma 99 *(Pointer jumping structure). Let $G = (V, E)$ be a path graph or a cycle graph of length at most n. Algorithm 2 takes $O(\log n)$ rounds. Further, the pointer jumping structure $G' = (V, E')$ with $E' := E \cup \{\{v, \ell_{v,i}\}, \{v, r_{v,i}\} \mid i \in \{1, \dots, \lceil \log n \rceil\}\}$ has degree and diameter $O(\log n)$.*

Proof. The running time is clear, and since in each loop cycle at most two new connections are established for each node, the claim about the degree follows. Let $u, v \in V$ be at distance d in G. Consider the largest integer $i \geq 0$ such that $2^i \leq d$. Then there is an edge in G' from u to some node w with distance at most $d - 2^i < 2^{i-1}$ to u, i.e., the remaining distance to v in G' at least halves with each such a hop, thus the claim follows from $d \leq n$. □

This structure can be used to broadcast and aggregate messages.

Lemma 100. *On path and cycle graphs $G = (V, E)$ with length at most n, an $O(\log n)$ bit message can be broadcast from one node to all others in the structure in $O(\log n)$ rounds. Further, if each node in the structure has a $O(\log n)$ bit value, all nodes in the pointer jumping structure can agree on which value is the maximum or minimum in $O(\log n)$ rounds.*

Proof. We start with the broadcasting protocol. First we set up the pointer doubling structure $G' = (V, E')$ on G (see Lemma 99). Starting with the node initiating the broadcast, in round i, each node that already knows the message sends it to its neighbors in G' that are at distance 2^i in G. Let $s \in V$ be the source of the message and consider some $v \in V$. Then there is a path in G' from s to v that uses at most $\lceil \log n \rceil$ edges of strictly decreasing length (in G). As the procedure broadcasts the message on all such paths, v will receive the message. For computing the minimum (or maximum) value, we use the same procedure as above where each node only forwards the minimum (or maximum) value encountered so far. □

By constructing an Euler tour on a tree, we can replicate the pointer jumping structure for path graphs on trees.

Lemma 101 *(Euler tour technique for constant degree trees, cf., [11]). Let $T = (V, E)$ be a rooted, constant degree tree. We can construct an Euler tour on T, to obtain a path graph containing a constant number of virtual nodes for each original node in $O(1)$ rounds.*

Proof sketch. Each node of T starts by creating a virtual node for each of its neighbors. After ordering their neighbors locally, each node can decide in one round how its virtual nodes would be connected in a depth first search traversal and establishes these connections. As a result, we obtain a path graph and as each node has a constant degree by our assumption, the path graph has length $O(n)$ and each node is responsible for simulating $O(1)$ virtual nodes. □

The lemma above immediately implies the following corollary.

Corollary 102. *On trees $T = (V, E)$ with constant degree and at most n nodes, broadcasting an $O(\log n)$ bit message or aggregating the minimum or maximum value in T can be done in $O(\log n)$ rounds. The same is true for simple regions, since we can construct the portal tree (see Definition 18) of a simple region in $O(\log n)$ rounds.*

C.2. Identifying portals and holes

We can use the above techniques in the HYBRID model on grid graphs to assign a unique identifier to each hole (Definition 13) and portal (Definition 12) in the grid graph, which we often use implicitly throughout this work. To establish identifiers for all, say vertical, portals it is rather easy as the following argument shows.

Nodes first learn their neighbors on their portal P in a single round, which establishes a path (for each such portal in parallel). We then define the (x, y) coordinate of the lowest node on P as ID of P. We propagate this ID to all nodes of P by using an aggregation over the coordinates of all nodes on P, using the minimum over y coordinate as aggregation function (see previous section). It is clear that these IDs are unique among all vertical portals, since vertical portals do not intersect. For the same reason the aggregation on each portal can be conducted in parallel. Therefore the following lemma is a corollary of Lemma 100.

Lemma 103. *Given a grid graph Γ. For* all *portals P (Definition 12) in Γ, a unique ID of P can be propagated to each node on P in $O(\log n)$ rounds overall.*

The same claim is true for the set of hole boundaries, however the argument is slightly more involved as these can intersect.

Lemma 104. *Given a grid graph Γ with holes $\mathcal{H}$ (Definition 13). For* all *$H \in \mathcal{H}$ a unique ID of H can be propagated to each node on the boundary of H (Definition 14) in $O(\log n)$ rounds overall.*

Proof. From Definition 14, each node can locally identify whether it is incident to a hole in one round. Next, nodes identify their neighbors on a hole boundary in the following way.

- Consider the grid graph $\overline{\Gamma}_v$ formed by the at most 8 orthogonally and diagonally adjacent points in $\mathbb{Z}^2$ to a grid node v that are *not* occupied by a grid node.
- Each maximally connected component on $\overline{\Gamma}_v$ corresponds to a hole $H \in \mathcal{H}$ (not necessarily uniquely).
- For each maximally connected component in $\overline{\Gamma}_v$, consider the two existing nodes on either side and call them l and r ($l = r$ is possible in a special case where v borders a single hole).
- The first node on a shortest path from v to l and the first node on a shortest path from v to r are the neighbors of v with respect to the hole which h corresponds to (these might be the same nodes, however each node appears on a hole boundary of a given hole at most twice).

This procedure, can be performed locally by each node in $O(1)$ rounds and establishes a unique cycle of grid nodes for each hole in a distributed sense, i.e., each node knows for each incident hole its two (not necessarily distinct) neighbors on a cycle containing all hole boundary nodes (see Definition 14). Note that a node v may be incident to a constant number of holes and may appear at most a constant number of times on the cycle that we construct for each hole. In particular this implies that the length of any hole boundary is at most $n' \in O(n)$. Furthermore, a node may simulate a constant number of aggregations in parallel on the cycles we constructed for each hole boundary, within the confines of the HYBRID mode in $O(\log n') = O(\log n)$ rounds by Lemma 100.

It remains to define a unique ID for each hole boundary that we can determine by aggregation on the corresponding cycle. We define our identifier as the absolute minimum of a set of identifiers proposed by all nodes on a given hole boundary as follows. Each node proposes its coordinates $(x, y) \in \mathbb{Z}^2$ as an identifier for each incident hole. We define a total order on these coordinates as follows: $(x_1, y_1) < (x_2, y_2)$ iff $x_1 < x_2$ or $x_1 = x_2$ and $y_1 < y_2$. Then the ID of a hole $H \in \mathcal{H}$ corresponds to the minimum coordinate of nodes w.r.t. this order, that is, the ID of a left-most node on the boundary of H with y coordinate as tie breaking criterion (left-most, bottom-most node).

We argue that this ID is unique for each $H \in \mathcal{H}$. The node with minimum coordinate (x, y) on the boundary of H must have neighbors in Γ at positions $(x+1, y)$, $(x, y+1)$ and the point $(x+1, y+1)$ is in the interior of H (is not occupied by a grid node), as otherwise H would extend further left or down, contradicting the fact that (x, y) was minimum on the boundary. For a given node, this condition can only be true for a single hole $H \in \mathcal{H}$, thus each coordinate can be a minimum for at most one hole boundary. □

C.3. Splitting portals

In this section we describe the primitives necessary to split Γ along portals to obtain a decomposition of a grid graph Γ into regions (see Definition 25). This is used extensively in Section 4, where we aim to obtain a simple and path-convex region decomposition in $O(\log n)$ rounds. In this section we use three routines building on top of each other, that are described in the following.

Definition 105 *(Splitting procedure).*

1. *Splitting Γ at a portal P*: Assume P is a vertical portal, the other case works symmetrically. Each node on $v \in P$ creates a copy of itself, both of which it simulates; a left copy v_ℓ and a right copy v_r. We remove left neighbors of right copies and right neighbors of left copies.
2. *Splitting Γ at P and a boundary node $v \in P$ of some hole H*: Again, we assume that P is vertical, the other case is analogous. This procedure must specify the hole that v is boundary of in case v is at the boundary of more than one hole. We split P as in the procedure above. Then we create two further copies of the copy of v that is on the side of H: $v^\uparrow$ and $v^\downarrow$, which it henceforth both simulates. We define that the copy $v^\uparrow$ does not have a bottom neighbor and $v^\downarrow$ does not have a top neighbor.
3. *Splitting Γ at P and at boundary nodes $v_1, \ldots, v_\ell \in P$ of holes $\{H_1, \ldots, H_\ell\}$*: We split P as in the first procedure. In general P might have multiple nodes v_i that are boundaries of potentially different holes H_i. In that case we repeat the vertical split as described in the second step for each copy of v_i that is to the direction of H_i.

Lemma 106. *The procedures of splitting at portals P and boundary nodes as described above can be coordinated in $O(\log n)$ rounds. The subsequent simulation of node copies has at most constant overhead.*

Proof. In all of our usual applications of this lemma we assume that we have already identified the portal P and the boundary node v, and each node is aware of its role. Note that these preconditions are typically either local conditions, or conditions that we can easily establish, using our ability to identify and broadcast on hole boundaries and portals in $O(\log n)$ rounds (when we apply this lemma, we often implicitly use that these tasks are straightforward). The subsequent creation of node copies uses only local information, thus can be done $O(1)$ rounds. Note that after a splitting operation, each node has to simulate at most 3 copies, which is why we have only a constant overhead. □

References

[1] Ioannis Anagnostides, Themis Gouleakis, Deterministic distributed algorithms and lower bounds in the hybrid model, in: Seth Gilbert (Ed.), 35th International Symposium on Distributed Computing (DISC 2021), Dagstuhl, Germany, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 209, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 5:1–5:19, https://drops.dagstuhl.de/opus/volltexte/2021/14807.

[2] Arash Asadi, Vincenzo Mancuso, Rohit Gupta, An sdr-based experimental study of outband d2d communications, in: IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[3] John Augustine, Mohsen Ghaffari, Robert Gmyr, Kristian Hinnenthal, Fabian Kuhn, Jason Li, Christian Scheideler, Distributed computation in node-capacitated networks, in: Proc. of the 31st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2019), 2019, pp. 69–79.

[4] John Augustine, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, Philipp Schneider, Shortest paths in a hybrid network model, in: Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'20, Society for Industrial and Applied Mathematics, USA, 2020, pp. 1280–1299.

[5] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, Jorge Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, Wirel. Netw. 7 (6) (2001) 609–616, https://doi.org/10.1023/A:1012319418150.

[6] Jannik Castenow, Christina Kolb, Christian Scheideler, A bounding box overlay for competitive routing in hybrid communication networks, in: Proc. of the 21st International Conference on Distributed Computing and Networking (ICDCN 2020), 2020, pp. 14:1–14:10.

[7] Keren Censor-Hillel, Dean Leitersdorf, Volodymyr Polosukhin, Distance computations in the hybrid network model via oracle simulations, in: Markus Bläser, Benjamin Monmege (Eds.), 38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021), Dagstuhl, Germany, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 187, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 21:1–21:19, https://drops.dagstuhl.de/opus/volltexte/2021/13666.

[8] Keren Censor-Hillel, Dean Leitersdorf, Volodymyr Polosukhin, On sparsity awareness in distributed computations, in: Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'21, New York, NY, USA, Association for Computing Machinery, 2021, pp. 151–161.

[9] Sam Coy, Artur Czumaj, Michael Feldmann, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, Philipp Schneider, Martijn Struijs, Near-shortest path routing in hybrid communication networks, in: Quentin Bramas, Vincent Gramoli, Alessia Milani (Eds.), 25th International Conference on Principles of Distributed Systems (OPODIS 2021), Dagstuhl, Germany, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 217, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 11:1–11:23, https://drops.dagstuhl.de/opus/volltexte/2022/15786.

[10] Michael Elkin, Ofer Neiman, On efficient distributed construction of near optimal routing schemes, in: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, 2016, pp. 235–244.

[11] Michael Feldmann, Kristian Hinnenthal, Christian Scheideler, Fast hybrid network algorithms for shortest paths in sparse graphs, in: Proc. of the 24th International Conference on Principles of Distributed Systems (OPODIS 2020), 2020, pp. 31:1–31:16.

[12] Magnús M. Halldórsson, Tigran Tonoyan, Plain SINR is enough!, in: Proc. of the 38th ACM Symposium on Principles of Distributed Computing (PODC 2019), 2019, pp. 127–136.

[13] Fabian Kuhn, Philipp Schneider, Computing shortest paths and diameter in the hybrid network model, in: Proceedings of the 39th Symposium on Principles of Distributed Computing, PODC'20, New York, NY, USA, Association for Computing Machinery, 2020, pp. 109–118.

[14] Fabian Kuhn, Philipp Schneider, Routing schemes and distance oracles in the hybrid model, in: International Symposium on Distributed Computing (DISC), vol. 246, 2022, pp. 28:1–28:22.

[15] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, Aaron Zollinger, Geometric ad-hoc routing: of theory and practice, in: Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003), 2003, pp. 63–72.

[16] Fabian Kuhn, Roger Wattenhofer, Aaron Zollinger, Asymptotically optimal geometric mobile ad-hoc routing, in: Proc. of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2002), 2002, pp. 24–33.

[17] Fabian Kuhn, Roger Wattenhofer, Aaron Zollinger, Worst-case optimal and average-case efficient geometric ad-hoc routing, in: Proc. of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), 2003, pp. 267–278.

[18] Christoph Lenzen, Boaz Patt-Shamir, Fast routing table construction using small messages: extended abstract, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC'13, New York, NY, USA, Association for Computing Machinery, 2013, pp. 381–390.

[19] Christoph Lenzen, Boaz Patt-Shamir, Fast partial distance estimation and applications, in: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, 2015, pp. 153–162.

[20] Nancy A. Lynch, Distributed Algorithms, Elsevier, 1996.

[21] Václav Rozhoň, Christoph Grunau, Bernhard Haeupler, Goran Zuzic, Jason Li, Undirected $(1+\varepsilon)$-shortest paths via minor-aggregates: near-optimal deterministic parallel and distributed algorithms, in: Symposium on Theory of Computing, 2022, pp. 478–487.

[22] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, Roger Wattenhofer, Distributed verification and hardness of distributed approximation, SIAM J. Comput. 41 (5) (2012) 1235–1265.

[23] Philipp Schneider, Power and Limitations of Hybrid Communication Networks, PhD thesis, University of Freiburg, 2023.