A Peer-to-Peer Based Cloud Storage Supporting Orthogonal Range Queries of Arbitrary Dimension

Markus Benter[1](✉), Till Knollmann[2](✉), Friedhelm Meyer auf der Heide[2](✉), Alexander Setzer[3](✉), and Jannik Sundermeier[2](✉)

[1] Jobware GmbH, Technologiepark 32, 33100 Paderborn, Germany
m.benter@jobware.de
[2] Computer Science Department and Heinz Nixdorf Institute, Paderborn University, Fürstenallee 11, 33102 Paderborn, Germany
{tillk,fmadh,janniksu}@mail.uni-paderborn.de
[3] Computer Science Department, Paderborn University, Paderborn, Germany
asetzer@mail.uni-paderborn.de
https://www.hni.uni-paderborn.de/alg/, https://cs.uni-paderborn.de/ti/

Abstract. We present a peer-to-peer network that supports the efficient processing of orthogonal range queries $R = \bigtimes_{i=1}^{d}[a_i, b_i]$ in a d-dimensional point space. The network is the same for each dimension, namely a distance halving network like the one introduced by Naor and Wieder (ACM TALG'07). We show how to execute such range queries using $\mathcal{O}\left(2^{d'} d \log m + d\,|R|\right)$ hops (and the same number of messages) in total. Here $[m]^d$ is the ground set, $|R|$ is the size and d' the dimension of the queried range. Furthermore, if the peers form a distributed network, the query can be answered in $\mathcal{O}\left(d \log m + d \sum_{i=1}^{d}(b_i - a_i + 1)\right)$ communication rounds. Our algorithms are based on a mapping of the Hilbert Curve through $[m]^d$ to the peers.

Keywords: Distributed storage · Multi-dimensional range queries · Peer-to-Peer · Hilbert Curve

1 Introduction

Consider a scenario in which the content of a music sharing platform is distributed among the participants of a peer-to-peer network (P2P network). Classical P2P networks only consider search queries for a specific attribute of the data contained in the network. For a music sharing platform, however, a crucial requirement is to allow more complex search queries. Users typically want

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center 'On-The-Fly Computing' (SFB 901).

to filter the data by different attributes and, most important, filter by *multiple* attributes. A typical request would be to find all jazz, funk & soul songs published in the last two years. Further queries could involve the language, the audio quality of the tracks or the duration. In the last years, several publications investigate the design of P2P networks allowing for multi-dimensional range queries. For more details, we refer the reader to Sect. 1.1. Most of these solutions, however, involve very complex network designs and the design depends heavily on the dimensionality of the data stored in the network. In this work, we aim at a more lightweight solution which uses the same network structure independent of the dimensionality of the data. Additionally, we are interested in an analysis of the message complexity and the delay for answering multi-dimensional range queries. More formally, the topic of this work can be described as follows:

Consider a distributed system storing a set of data items associated with keys. The keys are given by multiple attributes of the data items such that they can be seen as points in a d-dimensional mesh. We consider the d-dimensional Range Query Problem defined as follows.

Definition 1 (d-dimensional Range Query Problem) *Let $M(m, d)$ be a d-dimensional mesh with side-length m where $m = 2^k$ for a $k \in \mathbb{N}$. Let $R = \times_{i=1}^{d}[a_i, b_i]$ be an orthogonal range (or range for short) in $M(m, d)$. The d-*dimensional Range Query Problem *is the task of reporting all points in R. The* dimension of a range*, d', is the number of all dimensions i with $a_i \neq b_i$.*

Our goal is to distribute the m^d points of $M(m, d)$ over m^d peers of a P2P network such that range queries can be answered using few hops and few communication rounds. In a communication round, each peer can receive and send a set of messages. Messages sent in communication round i are all received in round $i + 1$. The solution can be scaled to fewer peers in a simple way, as mentioned in the final section. The main focus of this work is to realize range queries of arbitrary dimension on one fixed network, rather than using a specific one for each dimension. In addition, we aim at a topology with constant degree.

Let $|R| = \prod_{i=1}^{d}(b_i - a_i + 1)$ be the size of a range R in our mesh. A trivial topology for the d-dimensional Range Query Problem would be the d-dimensional grid. This topology allows to answer a query for R in $\mathcal{O}(c + |R|)$ hops where c is the distance from the entry peer to the closest point in R. However, the grid topology depends on the dimension of the keys. Another trivial solution when assuming a constant degree network would be to query each point in R separately. While the network is independent of the mesh, the total number of hops can only be bounded to $\mathcal{O}(|R| \cdot d \cdot \log m)$ due to each constant degree network with m^d nodes having a diameter of $\Omega(d \cdot \log m)$. The main question we deal with in this paper is how close we can come to the bound of $\mathcal{O}(c + |R|)$ while still having a constant degree network topology.

The solution we present in this work uses a Hilbert Curve that maps the m^d points of $M(m, d)$ to a one-dimensional line. We assume a very dense set of keys, i.e., all points in $M(m, d)$ refer to existing keys stored in the system. As we consider m^d points, we assume a bijective mapping from $[m]^d$ to the peers such that the Hilbert Curve defines an ordering of the peers. In Sect. 2, we present the topology of our P2P network, namely the Distance Halving Graph, the mapping of the nodes of $M(m, d)$ on it, and properties of this mapping. We present an algorithm for answering a range query R in Sect. 3 and prove that the total number of hops needed to answer a range query R is $\mathcal{O}\left(2^{d'} d \log m + d\,|R|\right)$. Moreover, we show that the algorithm can be parallelized and then answers a range query in $\mathcal{O}\left(d \log m + d \sum_{i=1}^{d}(b_i - a_i + 1)\right)$ communication rounds.

1.1 Related Work

Recently, the design of P2P networks for the purpose of answering multi-dimensional range queries has attracted much attention in the research community. For one-dimensional range queries, CAN, P-Grid, Baton, Armada, Saturn, and PHT have been proposed [3,6,9,11,14,15]. Throughout this section, let n be the number of peers in the network. Among these, Baton and Armada can answer one-dimensional range queries in time $\mathcal{O}(\log n + |R|)$, where $|R|$ denotes the number of data items in the range R. This is asymptotically optimal [11]. Baton, however, has a logarithmic degree at each node, whereas Armada requires only a constant degree.

For multi-dimensional ranges, there is no approach known which comes close to the optimal bound while having only a constant degree. In the literature, most work focuses on an experimental evaluation of P2P networks allowing for multi-dimensional range queries. In this work, we aim at a rigorous analysis of the message complexity and the communication rounds needed for answering a range.

P2P networks designed for multi-dimensional range queries can be subdivided into classes based on their approaches of mapping the data space and connect the peers to each other. MURK uses a k-d tree to partition the data space such that each node of the network is responsible for exactly one hypercuboid of the data space [7]. Moreover, MURK supports efficient routing by adding random links or a space-filling Skip Graph. Both approaches, however, require $\mathcal{O}(\log n)$ references stored at each node such that the resulting degree of the network is $\mathcal{O}(\log n)$.

Other approaches using Skip Graphs, are SkipIndex [20], ZNet [18] and SCRAP [7]. The difference in these approaches is the way of partitioning the data (for instance a space-filling Z-curve in Znet) and the way of choosing additional overlay edges for efficient routing. As all these approaches are based on Skip Graphs, the degree of each network is $\mathcal{O}(\log n)$.

MatchTree [10] uses an interesting approach which builds an individual tree for each query. The underlying P2P network is a variant of Kleinberg's small world network [12]. In these networks, each node has $\mathcal{O}(\log n)$ shortcut neighbors chosen randomly. Since the tree for each query is built at the time of processing the query, each node has to store a lot of structural information. In addition, lots of nodes are involved in answering a query.

Another approach working with trees is DRAGON [5]. In DRAGON, the identifiers of peers are distributed in $[0, 1)$ and an aggregation tree is built upon the interval $[0, 1)$. The structure of the aggregation tree depends on the space filling Z-curve such that each node is responsible for a certain region of the curve. To ensure efficient query processing, each node stores a reference to each level of the aggregation tree in its local routing table which also results in a degree of $\mathcal{O}(\log n)$.

SWORD [2] and MAAN [4] are architectures which use multiple hash functions, one for each dimension of the data, and map each of these hash functions onto the same network. MAAN uses a locality preserving hash function for each of the d dimensions per resource. A triple of <*dimension, value, resource-info*> is stored at each node that stores a dimension of a resource due to a hash function. The authors present two routing algorithms for resolving ranges. The first one does a range query for each dimension and returns the cut of all results. While the cost for routing to the peer at the beginning of each range is in a total of $\mathcal{O}\left(d \log n\right)$ hops, the cost for gathering all values in a range concerning only one dimension results in $\mathcal{O}(m^{d-1} \sum_{i=1}^{d}(b_i - a_i + 1))$ hops in our scenario. Further, due to the splitting of the dimension set of each resource, an additional total memory of $\mathcal{O}\left(d\, m^d\right)$ is needed. The second routing algorithm improves the routing time at the cost of memory requirement. Each resource is stored d times in total. The routing then only processes a range query along one specific dimension and removes all false-positives. This induces a routing time in $\mathcal{O}\left(\log n + \min_i\{(b_i - a_i + 1\}\right)$ while the memory demand increases by a factor of d. In contrast to MAAN, we focus on querying exactly those points which are in the range, i.e., we do not have an (intermediate) over-approximation of the results. Further, our solution does not increase the memory requirement for storing keys.

LORM [17] also uses a distributed hash table. In LORM, the peers are organized in clusters and each cluster of peers is responsible for a single dimension. To answer queries, the original query is split into subqueries for each dimension which are then answered by the affected clusters. The worst case bounds for identifying nodes which are involved in a range query of LORM are similar to the bound mentioned for MAAN. To achieve the bounds, LORM also relies on a logarithmic network degree.

The approach closest to ours, called Squid, combines the Chord-architecture [19] with the space-filling Hilbert Curve [16]. The multi-dimensional space is mapped onto the one-dimensional space by using a Hilbert Curve. Queries are answered by recursively refining a query and map subqueries to peers of the network. Although the authors prove that only a small number of nodes is contacted while query processing, it is not clear how many messages are sent in the process and which delay is caused by answering a query.

2 Notations

The ground set of elements to be queried in the d-dimensional range query problem is the node set $[m]^d$ of the d-dimensional mesh $M(m, d)$ with side-length m. For ease of description, we assume $m = 2^k$ for $k \in \mathbb{N}_0$. Fix some i, $0 \leq i \leq k$, and let $m' = \frac{m}{2^i}$. $M(m,d)$ can be divided into 2^{di} meshes with side-lengths m'. These meshes are denoted as *canonical submeshes* with side-lengths m' (c.f. Fig. 1).

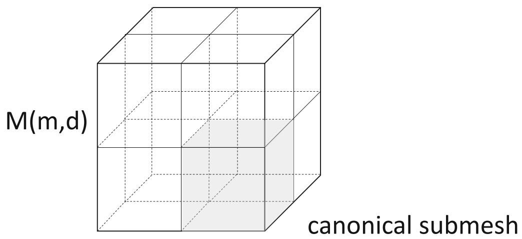

Fig. 1. The subdivision of a 3-dimensional mesh into $2^3 = 8$ canonical submeshes with side-lengths $m' = \frac{m}{2}$.

2.1 Embedding a Hilbert Curve in a Mesh

The *M(m,d)-Hilbert Curve* is a curve that connects all points of $M(m,d)$. It is a discrete version of the space-filling Hilbert Curve [8]. The $M(2,2)$-Hilbert Curve is defined as in the left picture of Fig. 2. The $M(m,2)$-Hilbert Curve can be subdivided into four rotated $M(\frac{m}{2},2)$-Hilbert Curves. The upper left and upper right $M(\frac{m}{2},2)$-Hilbert Curves are not rotated. The lower left one is rotated by 90° clockwise and the lower right one is rotated by 90° counterclockwise.

The $M(m,d)$-Hilbert Curve is an extension of the $M(m,2)$-Hilbert Curve. For d dimensions, a $M(m,d)$-Hilbert Curve can be subdivided into 2^d $M(\frac{m}{2},d)$-Hilbert Curves. For the formal definition, we refer the reader to [1]. The definition of the $M(m,d)$-Hilbert Curve ensures that all points of a canonical submesh are connected by a Hilbert Curve of smaller size. Further, we define an ordering of all points in $M(m,d)$ by the order in which they are visited via the $M(m,d)$-Hilbert Curve starting at the origin. Then the *Hilbert Coordinate* $p(v)$ of $v \in M(m,d)$ is the position of v in this ordering.

2.2 Network Properties

The P2P network we consider is the Distance Halving Graph as defined in [13]. We use the version with perfect smoothness. In this case, we can simply assume that the ids of the n peers $\{w_0,\ldots,w_{n-1}\}$ are $\{0,\ldots,n-1\}$ instead of $\{\frac{0}{n},\ldots,\frac{n-1}{n}\}$. The peer w_i has undirected edges to its direct neighbors w_{i-1} and w_{i+1} (for $i > 0$ and $i > n-1$, resp.). In addition, it is connected to the peers with ids $\lfloor\frac{i}{2}\rfloor$ and $\lfloor\frac{i}{2}\rfloor + \lfloor\frac{n}{2}\rfloor$ (if $\lfloor\frac{i}{2}\rfloor + \lfloor\frac{n}{2}\rfloor \leq n-1$). We consider each edge to be undirected. It has been shown in [13] that the Distance Halving Graph with perfect smoothness has a constant degree. Lemma 1 is a direct consequence of the above construction.

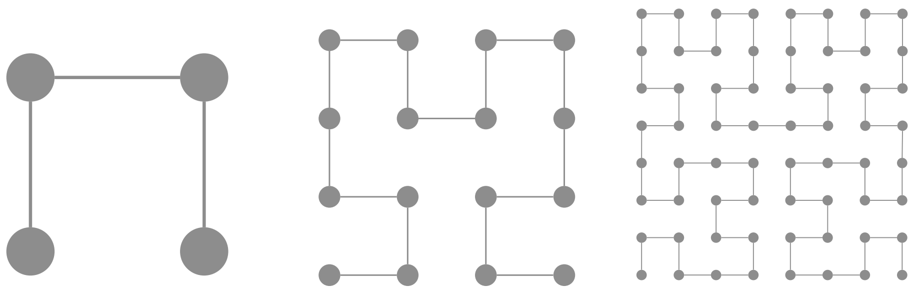

Fig. 2. M(m,d)-Hilbert Curve of dimension $d = 2$ and $m = 2, 4$ and 8.

Lemma 1. *The Distance Halving Graph supports a routing between w_i and w_j using at most $3 \cdot \log(|j - i| + 1)$ hops.*

Proof. Due to the definition of the Distance Halving Graph, w_i and w_j have edges to nodes $w_{i'}$ and $w_{j'}$ such that $|i' - j'| \leq \lfloor \frac{i}{2} \rfloor - \lfloor \frac{j}{2} \rfloor \leq \frac{1}{2}|i - j| + 1$. Applying an appropriate move of w_i to a direct neighbor reduces the distance to at most $\leq \frac{1}{2}|i - j|$. Iterating these three hops $\lfloor \log(|i - j| + 1) \rfloor$ times yields the lemma. (Note: We have only used one kind of the edges to non-direct neighbors. Combining both kinds of hops can be used to reduce the congestion of the routing, see [13].) $\square$

Now assume that n, the size of the P2P network, equals m^d, the size of our ground set. We define the one-to-one mapping from $M(m, d)$ to the peers by the ordering induced by the Hilbert Curve: Node v from $M(m, d)$ is mapped to the peer with id $p(v)$. As the nodes in a canonical submesh of $M(m, d)$ are ordered consecutively in the ordering induced by the Hilbert Curve, by Lemma 1 we can conclude:

Observation 1. *In a canonical submesh with side-length m', routing between any two points costs at most $\mathcal{O}(d \cdot \log(m') + 1)$ hops.*

3 Answering Range Queries

Whenever we talk about a range, we mean an orthogonal range defined as follows. An orthogonal range is given by $R = \times_{i=1}^{d} [a_i, b_i]$. We call the set of dimensions i in which $a_i < b_i$ the *extensions* of R, denoted by D. The number of extensions is the dimension of R denoted by $|D| = d'$.

A point $x = (x_1, \ldots, x_d)$ is a *corner point* of R if, for each i, $x_i = a_i$ or $x_i = b_i$ holds. Clearly, a range R with dimension d' has $2^{d'}$ corner points. Especially, a one-dimensional range, a *line* L, has two corner points. We say a line L *crosses a range* R if L only consists of points in R and has maximal length.

Let R be a d'-dimensional range. R is contained in a d'-dimensional Mesh M_R whose points are all points $(x_1, \ldots, x_d)$ of $M(m, d)$ with $x_i = a_i$ for all $i \notin D$.

We define canonical submeshes of M_R as intersections of canonical submeshes of $M(m, d)$ with M_R. From now on, we always consider M_R, i.e., if not stated otherwise, a canonical submesh is of M_R and dimensions which are not mentioned are assumed to match M_R. We further assume that dimensions 1 to d' are the dimensions in D.

Consider a line L with extension i such that $a_i < b_i$ and the smallest canonical submesh $S = \bigtimes_{j \in D}[x_j, y_j]$ surrounding L. We call L *touching*, if either $a = x_i$ or $b = y_i$ is a point at the border of S in dimension i. Note that, for a touching line L, the smallest enclosing canonical submesh has side-length at most $2|L|$. Consider a range R and its set of extensions D. R is *touching*, if every line L with an extension $i \in D$ that crosses R is touching (Figs. 3 and 4).

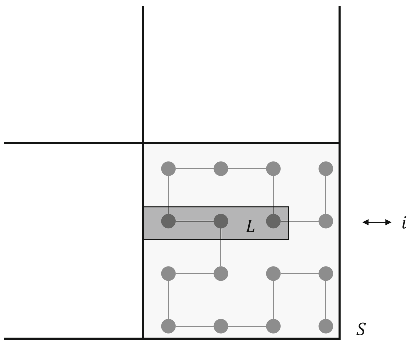

Fig. 3. A touching line query. L touches the canonical submesh S in dimension i.

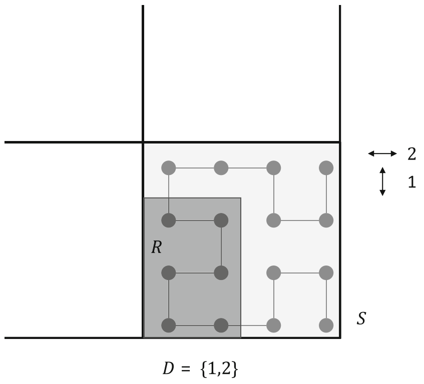

Fig. 4. A touching range query R in the canonical submesh S. The extension set of R is $D = \{1, 2\}$.

A major problem we have to deal with when answering range queries is the following: There are small ranges, whose smallest enclosing canonical submesh is large, may even be the entire mesh $M(m, d)$. An extreme range consists of the two nodes $(0, ..., 0, \frac{m}{2} - 1)$ and $(0, ..., 0, \frac{m}{2})$. See Fig. 5 for example. If we try to report such ranges following the Hilbert Curve, we have to follow many long routing paths between elements from the range, resulting in a large hop count. To deal with this problem, we will partition a range in subranges, each of which can be reported using only short such paths.

This partition is defined as follows. The *canonical box around* R is:

$$C(R) = \bigtimes_{i \in D} C_i, \text{ with } C_i = [c_i\, 2^j,\, (c_i + 1)\, 2^j - 1]$$

where $c_i \in \mathbb{N}_0$ and $j \geq 0$ minimal such that $c_i\, 2^j \leq a_i \leq b_i \leq (c_i + 1)\, 2^j - 1$.

The center z of $C(R)$ is

$$z = (z_1, \ldots, z'_d) \text{ with } z_i = \frac{c_i\, 2^j + (c_i + 1)\, 2^j - 1}{2}.$$

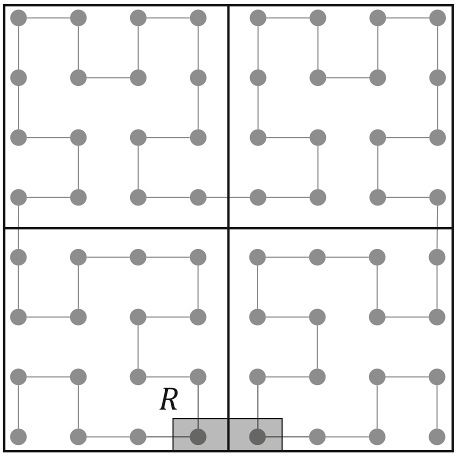

Fig. 5. Example of a small range R having a large smallest enclosing canonical submesh. The points of R are direct neighbors in $M(m, d)$ but far away from each other on the Hilbert Curve.

Consider the $2^{d'}$ orthants $O_1, \ldots, O_{2^{d'}}$ on M_R centered around z. For example, one of them is given by the points $\{x \in M_R \mid x_i > z_i \, \forall \, i \in D\}$. Now consider the subranges $R_i = R \cap O_i$. The following observation is crucial for our algorithm:

Observation 2. *Every subrange R_i is touching.*

Now consider the set $Z(R)$ of points of M_R centered around z:

$$Z(R) = \{p \mid p_i = z_i \pm (1/2) \text{ for } i \in D\}$$

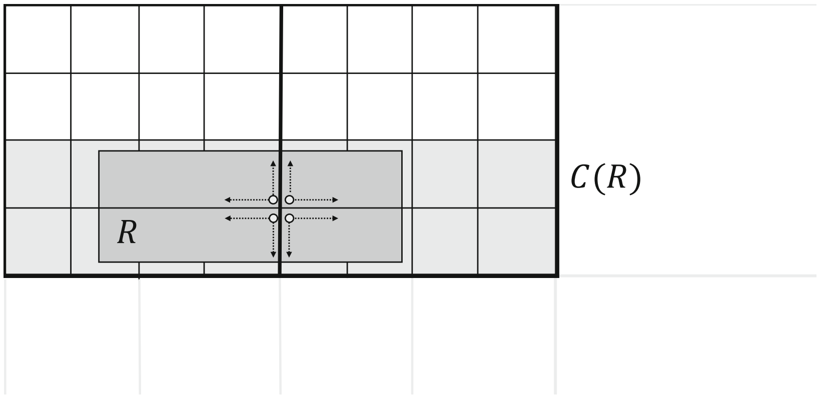

Fig. 6. The figure illustrates a range R together with its canonical box $C(R)$. The four points in the center of $C(R)$ are $Z(R)$. The midpoint between these points is z, the center of the canonical box $C(R)$.

Every R_i contains exactly one point of $Z(R)$. The points in $Z(R)$ are corner points of the subranges R_i. See Fig. 6 for a visualization of R, $C(R)$, $Z(R)$ and z.

To show that we can answer the subranges efficiently, we need another property. Fix a value $q \in [m]$ and a dimension $i \in D$. Let $H(q,i) = \{x \in M(m,d) \mid x_i = q\}$. The following observation is crucial for the core of our algorithm:

Observation 3. *Let R be touching. Then $H(q,i) \cap R$ is also touching. Its extension set is $D \setminus \{i\}$.*

3.1 The Algorithm

Our algorithm for answering a range works in two main steps. The first step splits the range into the $2^{d'}$ subranges as defined above. Then each subrange is answered separately by a recursive algorithm. The base case of this recursion answers queries for touching lines L using $O(d \cdot |L|)$ hops. In our description and the analysis, we only consider the processing until all points of R have been visited by our algorithm. For answering the range, these points must be sent back to the querying peer. However, this can easily be achieved by reversing the steps our algorithm does implying only a constant factor of two on our bounds.

To answer a range query, Algorithm 1 is called. Initially, D, $C(R)$, $Z(R)$ and the $2^{d'}$ subranges of R are determined. Then, the algorithm routes towards the points of $Z(R)$ and answers the touching subranges independently of each other. Consider such a subrange. Algorithm 2 requires a touching range R with extension set D and a corner point $s = (s_1, \ldots, s_d)$ as input. Given this input, an arbitrary dimension $i \in D$ is selected. The algorithm queries a touching line L with extension i that crosses R. The reporting of the points on L is interrupted after the visit of each point $q = (q_1, \ldots, q_{d'})$ of L. Now a recursive call for the $H(q_i, i) \cap R$ with extension set $D \setminus \{i\}$ (c.f. Observation 3) and corner point q is triggered. The recursion stops when the extension set D is empty, i.e., all dimensions have been completed.

Algorithm 1. Algorithm for answering an orthogonal range query R

```
1: procedure AnswerQuery($R$)
       * $R$ is an orthogonal range query with extension set $D$. $C(R)$ and $Z(R)$
         are as defined above. *
2:     Route towards the $2^{d'}$ points in $Z(R)$
3:     for each subrange $R'$ of $R$ with corner point $p \in Z(R)$ do
           * Observation 2 ensures that $R'$ is touching *
4:         ProcessRange($R'$, $p$)
```

Algorithm 2. Algorithm for answering a touching R beginning at a corner s

```
1: procedure ProcessRange($R$, $s = (s_1, \ldots, s_d)$)
       * Let $D$ be the extension set of $R$ *
2:     if $D \neq \emptyset$ then
3:         $i \leftarrow$ arbitrary dimension in $D$
4:         $L \leftarrow$ line with extension $i$ and endpoint $s$ that crosses $R$
5:         Visit each point of $L$ consecutively
6:         for each visited point $q = (q_1, \ldots, q_{d'})$ on $L$ do
               * Due to Observation 3, $H(q_i, i)) \cap R$ is touching *
7:             ProcessRange($H(q_i, i) \cap R$, $q$)
```

3.2 Analysis

For the analysis, we are interested in the total number of hops as well as the number of communication rounds. The total number of hops reflects the message complexity of our solution. Our main result is stated in Theorem 1.

Theorem 1. *Algorithm 1 answers a range query R in $\mathcal{O}\left(2^{d'} d \log m + d\,|R|\right)$ hops within $\mathcal{O}\left(d \log m + d \sum_{i=1}^{d}(b_i - a_i + 1)\right)$ communication rounds.*

We already discussed that $|Z(R)| = 2^{d'}$. For the points in $Z(R)$, our algorithm has to do large routing steps that cost $\mathcal{O}(d \log m)$ hops each by Observation 1. Therefore, the first part of our algorithm requires $\mathcal{O}(2^{d'}\, d \log m)$ hops in total. The subranges can be answered in parallel such that the first part needs $\mathcal{O}(d \log m)$ communication rounds leading to the correctness of Lemma 2.

Lemma 2. *Algorithm 1 needs $\mathcal{O}\left(2^{d'}\, d \log m\right)$ hops and $\mathcal{O}(d \log m)$ communication rounds to route to all points in $Z(R)$.*

It is left to analyze the performance of Algorithm 2 for a touching subrange R' with extension set D, and the corner point $c \in Z(R)$ of R'. The recursive formulation reduces the problem of answering R' to the problem of answering touching lines crossing R'. Due to Observations 2 and 3, we know that all queried lines are touching. Thus, we show Lemma 3. Extending the result of Lemma 3 to the behavior of Algorithm 2 allows to show Lemma 4.

Lemma 3. *Let L be a touching line query. L can be answered in $\mathcal{O}(d\,|L|)$ hops if the routing starts at an endpoint of L.*

Proof. We observe that due to the touching property of L, the smallest surrounding canonical submesh S has a side-length t with $|L| \leq t \leq 2|L|$. The number of hops for visiting L is at most the number of hops for visiting L' which is L extended to cross S. L' has length t. Observe that L' lies completely in two canonical submeshes S' and S'', each of edge-length $t/2$. To analyze the number of hops $F(t)$ for visiting L', we consider the parts of L' in S' and S'' separately.

Then, $F(t)$ is composed of $2\,F(t/2)$ for answering the two parts of L', plus the number of hops for jumping from the endpoint of the line in S' to a neighboring endpoint of the line in S''. Since L' is completely contained in S, Observation 1 implies that the number of hops for the jump is in $\mathcal{O}\left(d \cdot \log t + 1\right)$. This yields the following recursion:

$$\begin{aligned} F(t) &\leq 2 \cdot F\left(\frac{t}{2}\right) + c \cdot d \cdot \log(t) + 1 && \text{for } t > 1 \\ F(1) &= 0 && \text{else} \end{aligned}$$

The solution for this recursion is $F(t) = (t-1)\,(2\,c\,d+1) - c\,d\,\log(t)$. As $t = |L'| \geq |L|$ this results in a number of hops of $2\,c\,d\,|L| \in O(d \cdot |L|)$ for answering the line query L. □

Lemma 4. *Let R' be a subrange of R with extensions D. Algorithm 2 answers R' in $\mathcal{O}\left(d\,|R'|\right)$ hops and $\mathcal{O}\left(d\,\sum_{i=1}^{d}(b_i - a_i + 1)\right)$ communication rounds when starting at a corner point of R'.*

Proof. Let p be the corner point of R' at which the routing starts. Due to Observation 2, we know that any line that crosses R' is touching and can be answered efficiently as captured in Lemma 3. Our algorithm consecutively fixes the dimensions in D. For each fixed dimension, a touching line L is answered. Each such line L can be answered due to Lemma 3 in $\mathcal{O}\left(d\,\log|L|\right)$ hops.

Let $(1,\ldots,d')$ be the sequence of dimensions of D which are fixed one by one by Algorithm 2. Let $r_i = b_i - a_i + 1$ be the side-length of R' along dimension i. Then $|R'| = \prod_{i=1}^{d'} r_i$. The number of hops $T(r_i,\ldots,r_{d'})$ needed to report the (touching) range R' is bounded by the following recursion:

$$\begin{aligned} T(r_1,\ldots,r_{d'}) &\leq\ c\,d\,r_1 + r_1\,T(r_2,\ldots,r_{d'}) && \text{for } d' > 1 \\ T(r_1) &\leq c\,d\,r_1 && \text{for } d' = 1 \end{aligned}$$

for a sufficiently large constant c. Thus, $T(r_1,\ldots,r_{d'}) \leq c\,d\,(r_1 + r_1\,r_2 + \cdots + r_1 \cdot r_2 \cdot \ldots \cdot r_{d'}) \leq 2\,c\,d\,r_1\,r_2\,\ldots\,r_{d'} = 2\,c\,d\,|R'|$. Therefore, all points can be visited in $\mathcal{O}\left(d\,|R'|\right)$ hops.

Note that the recursive steps for every point of a line can be processed in parallel. Therefore, the algorithm needs $\mathcal{O}\left(d\,\sum_{i=1}^{d}(b_i - a_i + 1)\right)$ communication rounds. □

Combining Lemmas 2 and 4, we obtain the bounds of Theorem 1.

4 Concluding Remarks

It is easy to scale down our construction to smaller P2P networks: for some $m' < m$ let each peer take care of a whole canonical submesh with edge length m'. Then only $\left(\frac{m}{m'}\right)^d$ peers are used.

A more interesting question is how to deal with sparse data sets: If only a few of the m^d points of $M(m, d)$ hold data records, we would like to achieve a hop count close to the number of records contained in the queried range. For this, it is interesting to investigate whether our combination of the Hilbert Curve and the Distance Halving Network can be extended to incorporate advantages of, for example, k-d trees.

References

1. Alber, J., Niedermeier, R.: On multidimensional curves with hilbert property. Theory Comput. Syst. **33**(4), 295–312 (2000). https://doi.org/10.1007/s002240010003
2. Albrecht, J., Oppenheimer, D., Vahdat, A., Patterson, D.A.: Design and implementation trade-offs for wide-area resource discovery. ACM Trans. Internet Technol. **8**(4), 18:1–18:44 (2008). https://doi.org/10.1145/1391949.1391952
3. Andrzejak, A., Xu, Z.: Scalable, efficient range queries for grid information services. In: P2P 2002 Proceedings of the Second International Conference on Peer-to-Peer Computing, pp. 33–40 (2002). https://doi.org/10.1109/PTP.2002.1046310
4. Cai, M., Frank, M., Chen, J., Szekely, P.: MAAN: a multi-attribute addressable network for grid information services. J. Grid Comput. **2**(1), 3–14 (2003). https://doi.org/10.1007/s10723-004-1184-y
5. Carlini, E., Lulli, A., Ricci, L.: DRAGON: multidimensional range queries on distributed aggregation trees. Future Gener. Comput. Syst. **55**, 101–115 (2016). https://doi.org/10.1016/j.future.2015.07.020, http://www.sciencedirect.com/science/article/pii/S0167739X15002526
6. Datta, A., Hauswirth, M., John, R., Schmidt, R., Aberer, K.: Range queries in trie-structured overlays. In: P2P 2005 Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, pp. 57–66. IEEE (2005)
7. Ganesan, P., Yang, B., Garcia-Molina, H.: One torus to rule them all: multidimensional queries in P2P systems. In: WebDB 2004 Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004 WebDB 2004, pp. 19–24. ACM, New York (2004). https://doi.org/10.1145/1017074.1017081
8. Hilbert, D.: Über die stetige Abbildung einer Linie auf ein Flächenstück, pp. 1–2. Springer, Heidelberg (1935). https://doi.org/10.1007/978-3-662-38452-7-1
9. Jagadish, H.V., Ooi, B.C., Vu, Q.H.: Baton: a balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st International Conference on Very Large Data Bases VLDB Endowment, pp. 661–672 (2005)
10. Lee, K., Choi, T., Boykin, P.O., Figueiredo, R.J.: MatchTree: flexible, scalable, and fault-tolerant wide-area resource discovery with distributed matchmaking and aggregation. Future Gener. Comput. Syst. **29**(6), 1596–1610 (2013). https://doi.org/10.1016/j.future.2012.08.009, http://www.sciencedirect.com/science/article/pii/S0167739X12001653. including Special sections: High Performance Computing in the Cloud & Resource Discovery Mechanisms for P2P Systems
11. Li, D., Cao, J., Lu, X., Chen, K.C.C.: Efficient range query processing in peer-to-peer systems. IEEE Trans. Knowl. Data Eng. **21**(1), 78–91 (2009). https://doi.org/10.1109/TKDE.2008.99
12. Kleinberg, J.M.: Navigation in a small world. Nature **406**, 845 (2000)
13. Naor, M., Wieder, U.: Novel architectures for P2P applications: the continuous-discrete approach. ACM Trans. Algorithms **3**(3), 34 (2007). https://doi.org/10.1145/1273340.1273350

14. Pitoura, T., Ntarmos, N., Triantafillou, P.: Saturn: range queries, load balancing and fault tolerance in DHT data systems. IEEE Trans. Knowl. Data Eng. **24**(7), 1313–1327 (2012). https://doi.org/10.1109/TKDE.2010.266
15. Ramabhadran, S., Ratnasamy, S., Hellerstein, J.M., Shenker, S.: Prefix hash tree: an indexing data structure over distributed hash tables. In: Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing, January 2004
16. Schmidt, C., Parashar, M.: Squid: enabling search in DHT-based systems. J. Parallel Distrib. Comput. **68**, 962–975 (2008)
17. Shen, H., Xu, C.Z.: Leveraging a compound graph-based DHT for multi-attribute range queries with performance analysis. IEEE Trans. Comput. **61**(4), 433–447 (2012). https://doi.org/10.1109/TC.2011.30
18. Shu, Y., Ooi, B.C., Tan, K.L., Zhou, A.: Supporting multi-dimensional range queries in peer-to-peer systems. In: P2P 2005 Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, pp. 173–180, August 2005. https://doi.org/10.1109/P2P.2005.35
19. Stoica, I., et al.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Network. **11**(1), 17–32 (2003)
20. Zhang, C., Krishnamurthy, A., Wang, R.Y.: Skipindex: Towards a scalable peer-to-peer index service for high dimensional data. Technical report, Princeton University, May 2004