2018 IEEE 34th International Conference on Data Engineering

Onion Curve: A Space Filling Curve with Near-Optimal Clustering

Pan Xu [#], Cuong Nguyen [*], Srikanta Tirthapura [*]
[#] *University of Maryland, College Park, MD, USA,* [*] *Iowa State University, Ames, IA, USA*

***Abstract*—Space filling curves (SFCs) are widely used in the design of indexes for spatial and temporal data. Clustering is a key metric for an SFC, that measures how well the curve preserves locality in mapping from higher dimensions to a single dimension. We present the *onion curve*, an SFC whose clustering performance is provably close to the optimal for cube and near-cube shaped query sets. We show that in contrast, the clustering performance of the widely used Hilbert curve can be far from optimal, even for cube-shaped queries. Since clustering performance is critical to the efficiency of multi-dimensional indexes based on the SFC, the onion curve can deliver improved performance for data structures for multi-dimensional data.**

I. Introduction

A space filling curve (SFC) is a mapping from a multi-dimensional universe to a single dimensional universe, and is a widely used tool in the design of data structures for multi-dimensional data. As described in [1], [2], such a mapping allows one to apply indexing techniques developed for single dimensional data to index multi-dimensional data, such as in a spatial database. SFCs have been applied to several scenarios in managing spatial and temporal data, such as in distributed partitioning of large spatial data [3], [4], multi-dimensional similarity searching [5], load balancing in parallel simulations [6], cryptographic transformations on spatial data [7], to name a few. A search for applications of space filling curves yields more than a thousand citations on Google Scholar, from areas such as databases and parallel computing.

In an application of an SFC to an indexing data structure, it is usually important to have good clustering performance and locality preservation. A query point set, such as a rectangle in the multi-dimensional universe, should be mapped to a point set that forms a small number of contiguous regions in a single dimension, to the extent possible. Many SFCs have been considered in the literature. Orenstein and Merrett [1] suggested the use of the Z curve, which is based on interleaving bits from different coordinates, to support range queries. Faloutsos [8], [9] suggested the use of the Gray-code curve for partial match and range queries. The SFC that seems to have attracted the most attention is the Hilbert curve [10]. Jagadish [2] considered different space filling curves, including the column-major and row-major curves, the Z curve, the Gray-code curve, and the Hilbert curve. His experiments showed that among those considered, an ordering based on the Hilbert curve performed the best in preserving locality. The locality preserving property of an SFC is captured by the *clustering number*, as defined further.

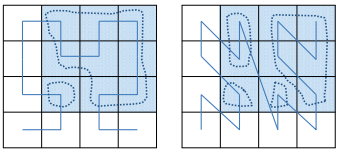


Fig. 1: For the same query region shown, the Hilbert curve has a clustering number of 2, and the Z curve has a clustering number of 4. Each cluster from the space filling curve is shown within a dotted region.

Model: Let U be a discrete d-dimensional universe of n cells, of dimensions $\sqrt[d]{n} \times \sqrt[d]{n} \ldots \times \sqrt[d]{n}$. In this paper, we focus our analysis on the cases $d = 2, 3$*. A space filling curve (SFC) π on U is a bijective mapping $\pi : U \rightarrow \{0, 1, \ldots, n-1\}$. A *query* q is any subset of U; we use $|q|$ to denote the number of cells in it. In this work, we are concerned with rectangular queries, which are formed by the intersection of halfspaces. A set of cells $C \subseteq U$ is said to be a *cluster* of SFC π if the cells in C are numbered consecutively by π.

The *clustering number* of an SFC π for query q, denoted $c(q, \pi)$, is defined as the minimum number of clusters of π that q can be partitioned into. See Figure 1 for an example. The importance of the clustering number is as follows. Suppose that multi-dimensional data was indexed on the disk according to the ordering induced by the SFC. If the query is to retrieve all points in a multi-dimensional region, then the clustering number measures the number of disk "seeks" that need to be performed in the retrieval. Since a disk seek is an expensive operation, a smaller clustering number means better performance. If the query is to compute some other function on a multi-dimensional region, then the clustering number measures the number of sub-queries on one-dimensional ranges, each of which can be handled efficiently using methods for indexing one-dimensional data. Hence the general goal is to design an SFC whose clustering number for a "typical" query is as small as possible. We consider the average clustering performance of an SFC for a class of queries, as in the following definition [11], [8], [2]. The clustering number of an SFC π for a query set Q is

*Extensions to higher dimensions are possible.

2375-026X/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDE.2018.00119

defined as:

$$c(Q,\pi) = \frac{\sum_{q \in Q} c(q,\pi)}{|Q|}$$

The query sets Q are constructed as follows. Consider a query shape q, which is a hyper-rectangle of length ℓ_i along dimension i. We say that q is a cube if $\ell_i = \ell_j$ for all $i \neq j$. We say that q is a near-cube if (1) There exists μ, $0 \leq \mu \leq 1$ such that for all $1 \leq i \leq d$, $\ell_i = \phi_i(\sqrt[d]{n})^\mu + \psi_i$ for some constants $\phi_i > 0$ and ψ_i; (2) $|\ell_j - \ell_i| = o(\sqrt[d]{n})$ for each $i \neq j$. The definition of "near-cube" captures the fact that the dependence of the side lengths of q on the grid size is the same, and also the gap between the largest and smallest sides is a lower order of the grid size. The second condition above is also necessary, see Lemma 1.

We primarily consider cases when q is a near-cube, or a cube. We consider query sets Q that are formed by all possible translations of such a rectangular query shape q. If q is a cube shaped query, we call Q as "cube query set" and if q is a near-cube shaped query, then we call Q as a "near-cube query set". For query set Q, let $\mathsf{OPT}(Q)$ be the smallest possible clustering number that could achieved by any SFC. For SFC π, let $\eta(Q,\pi) = c(Q,\pi)/\mathsf{OPT}(Q)$ be defined as the approximation ratio of π for Q. We focus on $\eta(Q,\pi)$ as our metric for evaluating the performance of π for Q. The lower the approximation ratio, the better is the clustering performance of π on Q.

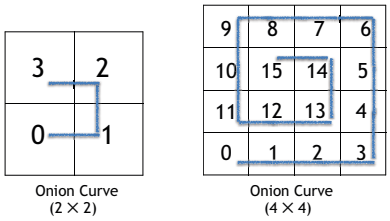


Fig. 2: The Two-dimensional onion Curve for the 2×2 and 4×4 universes. The number within each cell indicates the position of the cell in the onion curve.

A. Our Results

A. We present **a new SFC, the *Onion Curve***, with the following properties. (1) For cube query sets of any length in two dimensions, the approximation ratio of the onion curve is at most 2.32. For cube query sets in three dimensions, its approximation ratio is at most 3.4. Thus we call the onion curve *near-optimal* for cube queries. (2) For near-cube query sets in two dimensions, the approximation ratio of the onion curve is also a constant. *In contrast, we show that the Hilbert curve can have an approximation ratio of* $\Omega\left(\sqrt{n}\right)$ *for cube queries in two dimensions, and an approximation ratio of* $\Omega\left(n^{\frac{2}{3}}\right)$ *for cube queries in three dimensions*. Thus the approximation ratio of the Hilbert curve can be unbounded, even for the case of cube queries in two dimensions. All results are summarized in Table I. See Figure 2 for the onion curve in two dimensions.

[†]It can be expressed as a function of ϕ_1 and ϕ_2, omitted here.

B: General Rectangular Queries: We have shown an SFC (the onion curve) that has near-optimal clustering for cube queries of any length, and also near-cube queries. It is natural to ask if it is possible to have a single SFC that has near-optimal clustering for different rectangular query sets. We show that this is impossible to achieve.

Lemma 1: No SFC can have a constant clustering number over both of Q_R and Q_C, where Q_R and Q_C consist of all possible rows and columns in a two-dimensional universe respectively.

C: We present results of experiments that empirically evaluate and compare the clustering performance of the onion and Hilbert curves. These results confirm our theoretical predictions and show that on cube, near-cube, and general rectangular queries, the clustering performance of the onion curve was often much better than the Hilbert curve, and always at least as good.

The significance of this result stems from the fact that the Hilbert curve has long been considered the "gold standard" of SFCs for purposes of clustering. So far, we did not know of an alternative that has a demonstrably better performance. Many past analyses have almost solely focused on understanding the performance of the Hilbert curve, for example, see [11], [12]. Our results show that the onion curve consistently outperforms the Hilbert curve in terms of clustering performance.

B. Related Work

Prior work [12], [11] considered an upper bound on the clustering of the Hilbert curve, and showed that if the size of the query remains constant, the clustering number of the Hilbert curve is asymptotically equal to the surface area of the query shape divided by twice the number of dimensions. Xu and Tirthapura [13] generalized this result by showing that when the size of the query remains constant, the clustering number of any SFC that has the property of being "continuous" is asymptotically equal to that of the Hilbert curve and further showed that for rectangular queries of constant size, the performance of the Hilbert curve is optimal. A significant difference between [13], [11], [12] and our work is that prior works assume that the query size is constant when compared with the size of the universe, where as we do not make this assumption.

While the clustering number is an important and well-analyzed metric for the performance of an SFC, we note that there are other metrics that shed light on other properties of an SFC. In particular Gotsman and Lindenbaum [14] consider the "stretch" of an SFC, defined as the ratio between the distance between two points in the d-dimensional grid and the one-dimensional numbering. Xu and Tirthapura [15] proved a lower bound on the "nearest-neighbor stretch" of any SFC. Asano *et al.* [16] considered the clustering performance in a model where: for a query q consisting of $|q|$ cells, the query processor is allowed to return a set of $C|q|$ cells which is a superset of q,

[‡]We assume $\ell \leq \sqrt[3]{n} - 20$. We did not optimize the constant $\eta(Q,\mathcal{O})$ when $\ell > \sqrt[3]{n} - 20$.

	$\eta(Q,\mathcal{O}),\ d=2$		$\eta(Q,\mathcal{O}), d=3$	$\eta(Q,\mathcal{H}), d\in\{2,3\}$
	$\ell_1 \le \ell_2$	$\ell_1 = \ell_2$	$\ell_1=\ell_2=\ell_3$	$\ell_i = \ell, i \in [d]$
$\mu = 0$	1	1	1	1
$0 < \mu < 1$	$(1+\frac{\phi_2}{\phi_1})$	2	2	unknown
$\mu = 1,\ 0 < \phi_1 \le \phi_2 \le 1/2$	$O(1)^{\dagger}$	≤ 2.32	≤ 3.4	unknown
$\mu = 1,\ 1/2 < \phi_1 \le \phi_2 < 1$	$\le 2+3\left(\frac{\phi_2-\phi_1}{1-\phi_2}\right)^2$	2	2	unknown
$\mu = 1,\ \phi_1 = \phi_2 = 1$	$\le 2+3\left(\frac{\psi_2-\psi_1}{1-\psi_2}\right)^2$	2	$\le 3^{\ddagger}$	$\Omega(n^{\frac{d-1}{d}})$

TABLE I: $\eta(Q,\mathcal{O})$ and $\eta(Q,\mathcal{H})$ for different near-cube query sets Q. $\mathcal{O}$ is the onion curve and $\mathcal{H}$ is the Hilbert curve.

and can be divided into a small number of clusters, where C is a constant greater than 1. A similar approach is considered by Haverkort [17]. In contrast, we require the query processor to return the set of exactly the cells present in the query q, and consider the number of clusters thus created.

II. Onion Curve

A. Two-dimensional Onion Curve

We first define the onion curve in two dimensions. For integral $j \ge 1$, let U_j denote the two dimensional $j \times j$ universe, whose coordinates along each dimension range from 0 to $(j-1)$. Let $\mathcal{O}_j$ denote the onion curve for U_j. $\mathcal{O}_j$ is defined using induction on $j, j = 2, 4, 6, \ldots, \sqrt{n}$.

- For $j = 2$, $\mathcal{O}_2$ is defined as: $\mathcal{O}_2(0,0) = 0, \mathcal{O}_2(1,0) = 1, \mathcal{O}_2(1,1) = 2, \mathcal{O}_2(0,1) = 3$. See Figure 2.
- For $j > 2$, $\mathcal{O}_j(x_1, x_2)$ is defined as:
  1) x_1, if $x_2 = 0$
  2) $j - 1 + x_2$, if $x_1 = j - 1$
  3) $3j - 3 - x_1$, if $x_2 = j - 1$
  4) $4j - 4 - x_2$, if $x_1 = 0, x_2 \ge 1$
  5) $4j - 4 + \mathcal{O}_{j-2}(x_1 - 1, x_2 - 1)$, if $1 \le x_1, x_2 \le j - 2$

Another view of the onion curve is as follows. For cell $\alpha = (x, y)$, let $\nabla(\alpha) = \min(x + 1, \sqrt{n} - x, y + 1, \sqrt{n} - y)$ denote the distance of the cell to the boundary of $U_{\sqrt{n}}$. For integer t, $1 \le t \le m$, let $S(t) = \{\alpha | \nabla(\alpha) = t\}$ denote the set of cells whose distance to the boundary is t. $S(t)$ is also called the "t^{th} layer" of the grid. The onion curve orders all cells in $S(1)$ first, followed by cells in $S(2)$, and so on till $S(m)$. The orders induced by the onion curve for U_2 and U_4 are shown in Figure 2.

B. Three-dimensional Onion Curve

Consider the three-dimensional universe U with side length $\sqrt[3]{n} = 2m$, for integer m. For a cell $\alpha = (x, y, z)$, define $\nabla(\alpha) = \min(x + 1, 2m - x, y + 1, 2m - y, z + 1, 2m - z)$ as the distance of the cell to the boundary of U. For integer t, $1 \le t \le m$, let $S(t) = \{\alpha | \nabla(\alpha) = t\}$ which represents the set of cells whose distance to the boundary is t. $S(t)$ is called the "t^{th} layer" of the grid. *Similar to the two-dimensional case, the onion curve numbers cells in the order $S(1)$ first, then $S(2)$, and so on till $S(m)$.* For each layer $S(t), 1 \le t \le m$, we give a disjoint partition as $S(t) = \bigcup_{g=1}^{10} S_g(t)$ which are defined respectively as follows. Note that for $1 \le g \le 10$, $S_g(t)$ is either a line or a two dimensional square, whose sides are of even length.

$$
\begin{aligned}
S_1(t) &= \{i = t - 1\} \\
S_2(t) &= \{i = 2m - t\} \\
S_3(t) &= \{t \le i < 2m - t, j = t - 1, k = t - 1\} \\
S_4(t) &= \{t \le i < 2m - t, j = t - 1, t \le k < 2m - t\} \\
S_5(t) &= \{t \le i < 2m - t, j = t - 1, k = 2m - t\} \\
S_6(t) &= \{t \le i < 2m - t, j = 2m - t, k = t - 1\} \\
S_7(t) &= \{t \le i < 2m - t, j = 2m - t, t \le k < 2m - t\} \\
S_8(t) &= \{t \le i < 2m - t, j = 2m - t, k = 2m - t\} \\
S_9(t) &= \{t \le i < 2m - t, t \le j < 2m - t, k = t - 1\} \\
S_{10}(t) &= \{t \le i < 2m - t, t \le j < 2m - t, k = 2m - t\}
\end{aligned}
$$

Within each layer $S(t), 1 \le t \le m$, the onion curve will index cells in the order $S_1(t) \to S_2(t) \to \cdots \to S_{10}(t)$. Within each $S_g(t), 1 \le g \le 10$, the onion curve then will index each cell by the natural order induced by the line if $S_g(t)$ is a line or the order given by the two dimensional onion curve if $S_g(t)$ is a two-dimensional plane.

In our definition, the essential rule that the onion curve must follow is to organize different layers $S(t), 1 \le t \le m$ sequentially rather than intercross them. That is the key factor that makes the onion curve have a near-optimal clustering number, as we show later. In contrast, the order in which the onion curve organizes the different $S_g(t), 1 \le g \le 10$ for each $1 \le t \le m$ is not so important. We can actually adopt any permutation on that.

III. Experimental Study

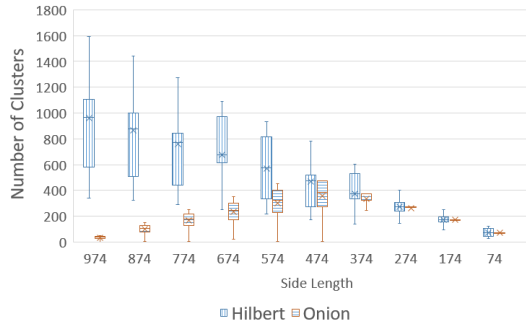
We present the results of experiments that compare the performance of the onion curve and Hilbert curve over a set of cube and rectangular queries in two and three dimensions.

A. Clustering Performance on Cube Queries

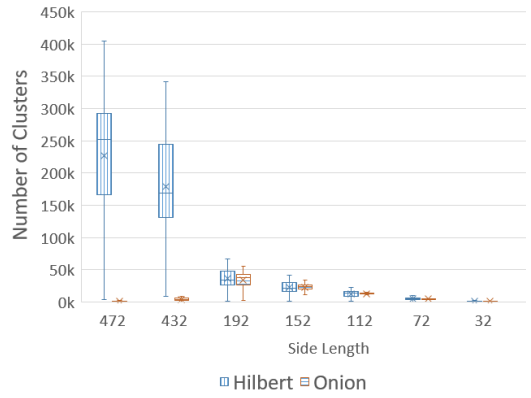
In this experiment, we consider cube queries of different lengths. For $d = 2$, we set the length of the universe to be $\sqrt{n} = 2^{10}$. For each given $\ell = 2^{10} - 50 \cdot k$, where $k \in \{1, 3, 5, \ldots, 19\}$, we generate a set $Q(\ell)$ of 1000 random squares of length ℓ. To generate a random square of a given side length, we choose the lower left endpoint of the square uniformly among all feasible positions for this point. For different values of ℓ, the distribution of clustering numbers for the two curves over $Q(\ell)$ is shown in Figure 3a.

Our setup for three dimensional cubes is similar. For $d = 3$, we set $\sqrt[3]{n} = 2^9 = 512$. For each given $\ell =$

$\{472, 432, 192, 152, 112, 72, 32\}$, we generate a set $Q(\ell)$ of 500 random three-dimensional cubes in the same way as the case of $d = 2$. The distribution of clustering numbers of the two curves over each $Q(\ell)$ is shown in Figure 3b.



(a) Two-dimensional Squares



(b) Three-dimensional Cubes

Fig. 3: The distribution of clustering numbers of the onion curve and Hilbert curve over a set of random squares and cubes with different side lengths. The box plot shows the 25 percentile and 75 percentile within the box, as well as the median, minimum, and maximum.

From Figure 3b, we observe that the onion curve performs much better than the Hilbert curve when the side length of the cube is large, greater than half the length of the axis. For instance, when $d = 3$ and the side length was more than 450, the clustering performance of the onion curve was more than 200 times better than that of the Hilbert curve. For each side length considered, whether large or small, the onion curve performed at least as well as the Hilbert curve, and was never worse than the Hilbert curve.

We have also experimented with rectangular queries, with varying ratios of side lengths, for two and three dimensions. We observed that even for rectangular queries, the median performance of the onion curve is better than the Hilbert curve in all cases that were considered, though the difference between the onion curve and the Hilbert curve was not as pronounced as in the case of cube shaped queries. See detailed results in the full version of our paper [18]. Overall, this shows that the onion curve improves upon the Hilbert curve in clustering performance, over a range of query shapes.

IV. Conclusions

We presented a new SFC, the onion curve, in two and three dimensions. This is the first SFC we know of, whose clustering performance is within a constant factor of the optimal for all cube and near-cube query sets. In contrast, the clustering performance of the Hilbert curve can be far from optimal. Further, the clustering performance of the onion curve is no worse that the Hilbert curve in all cases considered. The definition of the onion curve is simple, making it easy to use.

The onion curve can be extended naturally to higher dimensions, using the idea of ordering points according to increasing distance from the edge of the universe. The analysis of such a higher dimensional onion curve is the subject of future work. It is also interesting to consider the performance of Hilbert curve for general near-cube query sets as we do for the onion curve. Another future work is to complete Table I by filling the missing results for the Hilbert curve for general near-cube query sets.

References

[1] J. A. Orenstein and T. H. Merrett, "A class of data structures for associative searching," in *ACM PODS*, 1984, pp. 181–190.

[2] H. V. Jagadish, "Linear clustering of objects with multiple attributes," in *ACM SIGMOD*, 1990, pp. 332–342.

[3] K. Aydin, M. Bateni, and V. Mirrokni, "Distributed balanced partitioning via linear embedding," in *ACM WSDM*, 2016, pp. 387–396.

[4] M. Warren and J. Salmon, "A Parallel Hashed-Octtree N-body Algorithm," in *Supercomputing*, 1993.

[5] T. Li, Y. Lin, and H. Shen, "A locality-aware similar information searching scheme," *Int. J. on Digital Libraries*, vol. 17, no. 2, pp. 79–93, 2016.

[6] H. Liu, K. Wang, B. Yang, M. Yang, R. He, L. Shen, H. Zhong, and Z. Chen, "Load balancing using hilbert space-filling curves for parallel reservoir simulations," *CoRR*, vol. abs/1708.01365, 2017.

[7] H. Kim, S. Hong, and J. Chang, "Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data," *Data Knowl. Eng.*, vol. 104, pp. 32–44, 2016.

[8] C. Faloutsos, "Multiattribute hashing using gray codes," *SIGMOD Record*, vol. 15, pp. 227–238, June 1986.

[9] ——, "Gray codes for partial match and range queries," *IEEE Trans. Software Engg.*, vol. 14, pp. 1381–1393, 1988.

[10] D. Hilbert, "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Math. Ann.*, vol. 38, pp. 459–460, 1891.

[11] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, "Analysis of the clustering properties of the Hilbert space-filling curve," *IEEE TKDE*, vol. 13, no. 1, pp. 124–141, 2001.

[12] H. V. Jagadish, "Analysis of the Hilbert curve for representing two-dimensional space," *Inf. Proc. Let.*, vol. 62, pp. 17–22, 1997.

[13] P. Xu and S. Tirthapura, "Optimality of clustering properties of space-filling curves," *ACM TODS*, vol. 39, no. 2, pp. 10:1–10:27, 2014.

[14] C. Gotsman and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Trans. Image Processing*, vol. 5, no. 5, pp. 794–797, 1996.

[15] P. Xu and S. Tirthapura, "A lower bound on proximity preservation by space filling curves," in *IEEE IPDPS*, 2012, pp. 1295–1305.

[16] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmayer, "Space-filling curves and their use in the design of geometric data structures," *Theor. Comput. Sci.*, vol. 181, no. 1, pp. 3–15, 1997.

[17] H. J. Haverkort, "Recursive tilings and space-filling curves with little fragmentation," *Journal of Computational Geometry*, vol. 2, no. 1, pp. 92–127, 2011.

[18] P. Xu, C. Nguyen, and S. Tirthapura, "Onion curve: A space filling curve with near-optimal clustering," *arXiv preprint arXiv:1801.07399*, 2018.