Orderings of Data - More Than a Tripping Hazard

Visionary

Anna Beer
LMU Munich
Munich, Germany
beer@dbs.ifi.lmu.de

Valentin Hartmann
EPFL
Lausanne, Switzerland
valentin.hartmann@epfl.ch

Thomas Seidl
LMU Munich
Munich, Germany
seidl@dbs.ifi.lmu.de

ABSTRACT

As data processing techniques get more and more sophisticated every day, many of us researchers often get lost in the details and subtleties of the algorithms we are developing and far too easily seem to forget to look also at the very first steps of every algorithm: the input of the data. Since there are plenty of library functions for this task, we indeed do not have to think about this part of the pipeline anymore. But maybe we should. All data is stored and loaded into a program in *some* order. In this vision paper we study how ignoring this order can (1) lead to performance issues and (2) make research results unreproducible. We furthermore examine desirable properties of a data ordering and why current approaches are often not suited to tackle the two mentioned problems.

CCS CONCEPTS

• **Computing methodologies** → *Ontology engineering*; • **General and reference** → *Measurement*;

KEYWORDS

Visionary, Ordering, Data Input, Uniqueness, Reproducibility

1 INTRODUCTION

Data is everywhere. We live in the age of data and most readers of this vision paper work with data every day. Data comes in all shapes and sizes, and humans try to gain knowledge from it. We apply classification, clustering, outlier detection, and many others algorithms to it and develop new tools every day. The questions we try to answer and the approaches we come up with are as diverse as the data itself. But all algorithms we have developed so far have one thing in common that rarely gets the attention it should. Discussion about algorithms is so centered around results and computational performance that the very beginning of every algorithm is quite often simply forgotten: The fundamental step of storing the data and presenting it to the algorithm, which is often not made by the researcher using the data, but by the one who collected the data, who wrote the preprocessing function or the function to load the data into main memory.

SSDBM 2020, July 7–9, 2020, Vienna, Austria

ACM ISBN 978-1-4503-8814-6/20/07.
https://doi.org/10.1145/3400903.3400911

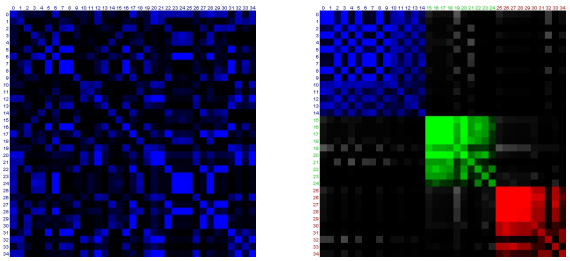

Figure 1: Adjacency matrices of a synthetic graph with differently ordered nodes.

However, this step is crucial not only for the performance, but also for the result of an algorithm, as Figure 3 illustrates. That phenomenon already occurs with simple algorithms and algorithm primitives like, e.g., k-NN-based algorithms. Here it can happen that several points have the same distance to a query point P and that the set of the k nearest neighbors depends on the tie-break rule. In this case the scikit implementation [4] still returns exactly k points, and bases the decision which points to discard on the ordering of the data points. Similar problems can occur in many other popular algorithms, too. In the field of clustering alone it encompasses algorithms like DBSCAN [13], k-Means [22], or agglomerative hierarchical clustering [17].

Randomly instead of deterministically deciding how to organize the data will also most likely not result in an ordering with optimal performance for the task at hand, since, based on this decision, datasets can have vastly different structural properties; see Figures 1 and 2. Thus, we want to encourage research on orderings of data, in more detail, we want to encourage finding solutions to several different aspects connected to orderings of data:

(1) A unique ordering for tabular data, which is especially interesting for high-dimensional data, as well as a unique ordering for the nodes of a graph
(2) A measure for the orderliness of data. That can either depend on how well suited an ordering is for the subsequent task, or how "close" it is to a unique ordering.

Section 2 discusses these issues in more detail. In Section 3 we develop approaches how to overcome them. We extract the need of an internal quality measure for orders and define the essential conditions of such a measure. In Section 4 we give an overview over the first steps that have already been made in this direction. Section 5 puts data orderings in context to several other research areas. Section 6 concludes this vision paper with a summary of the questions we rate as important and promising for further investigation.

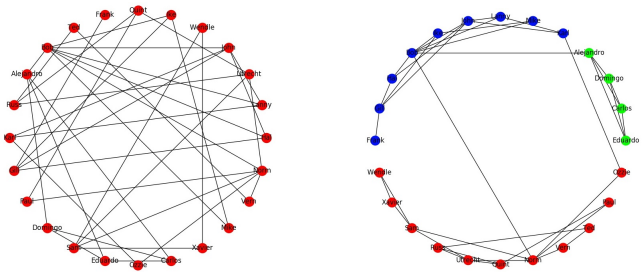

Figure 2: Graph of the social network "Strike" [11, 23]: Randomly ordered in a circle on the left, ordered by their similarity according to [8] with colors matching the ground truth on the right.

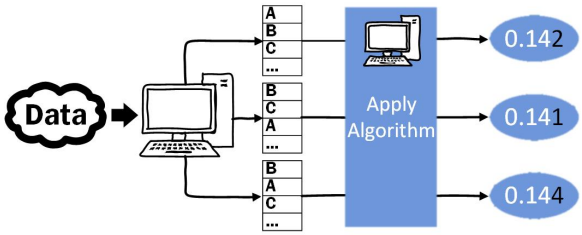

Figure 3: Different orderings of the same data can lead to different results when the same algorithm is applied to it.

2 THE PROBLEM

Not considering the ordering of the data objects one feeds to an algorithm has the major drawback of potentially making the output as well as the runtime of the algorithm non-deterministic, as Figure 3 illustrates. In the first part of this section we focus on the reproducibility in regards of research results and in the second part we address the variance of runtimes, as consistent runtimes are of high importance in various domains as, e.g., autonomous driving and other real-time applications.

Reproducibility. In the past years the problem of reproducibility of research results in computer science has received increasing attention with top-tier conferences awarding prices for reproducibility [3] or hosting reproducibility workshops [6], and universities trying to reproduce results at a large scale [5]. One of the reasons for these efforts is that especially in machine learning, theory is lacking behind the state-of-the-art, so validity and performance of many methods is only proven via experiments.

Randomness in algorithms can have a huge influence not only on the running time, but also on the output of an algorithm [14, Fig. 5] (consider, e.g., the case of training a non-convex machine learning model with several local minima). Addressing this problem by fixing a random seed alone is not sufficient; one also needs to fix the data ordering.

Performance. Physically organizing the data in a way that best fits the access patterns is a problem with a long history in database research [19, 27, 29]. The common idea is to exploit the locality of reference present in many applications [12]. In the case of data ordering, the spatial locality of reference is of particular interest. A simple example is breadth first search, where the nodes of a graph that are connected to each other are accessed shortly after each other, as well as nodes having the same depth with respect to the root node. To achieve optimal performance, such elements that are frequently accessed together or right after each other should also be close to each other on the physical storage. More concretely, it should be more efficient to access them sequentially — potentially reading a small amount of unwanted data in addition — than via random access. Sequential access is not only far more efficient for hard disks, but also for SSDs [9], eMMCs [20] and DRAM [16, Page 62]. Especially for column-store DBS, ordering of data plays a central role for the performance, like, e.g., when using holistic indexing [26].

Mathematical Formulation. Let D be the set of objects of a certain class, e.g., graphs on n unlabeled vertices. Let R be the set of representation of these objects of a certain type, e.g., the set of all $n \times n$ adjacency matrices. Then there exists an injective mapping from R to D (each adjacency matrix corresponds to exactly one graph). The inverse mapping, however, in many cases is not well-defined, since there can be many different representations of the same mathematical object (consider an adjacency matrix that only differs from another one in the order of the vertices).

3 HOW TO APPROACH THE PROBLEM

To remove the non-determinism inherent in all algorithms that work with non-unique representations of data, one has to define an injective mapping from the set of objects D to the set of representations R. Since the representations that algorithms work with are typically bit strings that list the elements that D consists of in a certain order, this means defining a unique ordering.

There are "better" or "worse" orderings. Depending on the application, different properties may be desirable. Two examples are:

- Points or nodes that are similar to each other should occur close together. Orderings of this type can, for example, efficiently support classic clustering.
- Points or nodes with similar roles, like, e.g., points in the middle of a cluster or hubs in a graph, should occur close together. Orders like these could significantly improve speed and support pruning.

To work towards fulfilling such a property with an ordering, one needs to quantify it. So on the one hand we need unique representations, and on the other hand we need quality measures for orderings.

Even though many internal as well as external quality measurements for clusters have been developed (already in 1981, [24] performed a study of *thirty* internal measures for cluster analysis), so far there are vanishingly few for orderings. And yet quality measurements for ordering are not only closely related to those for clusterings, but could also support clustering. A good clustering implies a partial order on nodes, but the nodes inside a cluster stay orderless. Vice versa, an ordering of nodes can easily be used to cluster them, since similar nodes are already close to each other.

Thus, nodes which are ordered "well" regarding such a quality measurement for ordering build a better foundation for clustering than those which are ordered "poorly", as, e.g., shown in [8].

Requirements for an Internal Measure of Orderliness. Inspired by [28], we formulate requirements :

(1) Scale invariance. There shall be no bias against larger or smaller amounts of data.
(2) The measure should be absolute, i.e., it should be possible to compare the orderliness of different datasets.
(3) Invariance with respect to language (e.g., regarding the attribute description of a dataset).

When using an ordering as way of improving the performance of a downstream task, one also has to take into account the time the ordering itself takes, see Sec. 5. The importance of this, however, rapidly decreases when the dataset is not used only once, but either queried many times or distributed in its ordered form, so that multiple runs of downstream algorithms can profit from the ordering. In these cases, orderings may be automatically obtained by analyzing the access patterns of the downstream algorithms on the dataset at hand, and letting a machine learning model derive a good data organization strategy automatically. The advantage of such dataset-specific approaches has recently been demonstrated for index structures [21].

4 EXISTING SOLUTIONS

There are already some approaches to data ordering, which we will present here. They all have the drawback that they either do not maximize a quality measure at all or that this quality measure is very general and not aimed at a specific goal or adapted to a specific algorithm. We also note that there are algorithms which order data in a preprocessing step, thus a certain ordering is obviously important for some use-cases. On the other hand an ordering suitable for many different algorithms is desirable.

Order of Input. The most simple method is to use the order in which the data is given. Of course this order is typically not optimal performance-wise, but at least it is unique — if one ensures that always the same data file is used. But often there are many different versions of the same data set available, especially of popular ones. Take for example the graph Zachary's Karate Club [31], which can be obtained from [1], [2] and [7], which all use a different representation. Also, for the same file different representations in main memory can result depending on how it is read.

4.1 Ordering of Graphs

Order by degree. A straight-forward method many frameworks use is to order the nodes by their degree. Unfortunately, this is a non-deterministic order for most graphs, since it is very common that two or more nodes in a graph have the same degree.

Depth-first and Breadth-first. For depth-first and breadth-first graph traversal, a root node has to be determined, which already makes the resulting ordering non-deterministic. The order in which each node's neighbors are later visited is also non-deterministic.

Cuthill-McKee. The Cuthill-McKee algorithm [10] orders the nodes in a way such that the corresponding adjacency matrix is a band matrix with small bandwidth. For a given ordering π that assigns integers $1, \ldots, n$ to the n nodes v_i of a graph, the bandwidth is defined as $B = \max\{|\pi(v_i) - \pi(v_j)| : v_i v_j \in E\}$, i.e., the maximum distance of a non-zero entry to the diagonal in the adjacency matrix. However, the algorithm uses the node degree as one of the ordering criteria, which is not unique. Thus a tie-break rule must be used, which can either rely on randomness or on the order in which the data is presented to the algorithm.

Gscore. In [30] the Gscore, a measure for the closeness of nodes with many common predecessors, is introduced and an algorithm given to order the graph in such a way that the score is low. This is shown to decrease the number of cache misses of several graph algorithms. However, minimizing the Gscore in NP-hard, so one can only hope for coming close to the minimum. In addition, as the orderings introduced earlier, this ordering is agnostic of the algorithm processing the data further and will thus often not be able to reach the maximum possible performance improvement an ordering could give for a specific algorithm.

Edge Length Minimization. One of the most recent orderings is described in [8], where all nodes of a graph are arranged in a circle and the optimal ordering is the one with the shortest average length of edges. Nevertheless, it is neither deterministic nor absolute.

4.2 Ordering of Tabular Data

Even though ordering of non-structured data seems to raise fewer questions at first, we will see that there are multiple aspects worth targeting when ordering multidimensional data.

Alphanumerical Sorting. Probably the easiest approach is to order the data alphanumerically, beginning with the first column, in case of ties proceeding to the second column and so on. For this we need to decide on an ordering of the columns. They can of course also be sorted alphanumerically by the attribute name, but that means that the ordering of the columns, and thus the ordering of the whole data, depends on the language in which the attributes are named. The same can occur when sorting within the attributes if the data is, e.g., categorical.

Locality Preserving Orderings. There are several orderings that try to preserve local structures in the data (e.g., Z-order [25] or Hilbert curve [15]), which can improve access times. When working in higher dimensionalities, this produces an ordering in which points that are close in Euclidean distance stay closer to each other in the ordering than they would using, e.g., alphanumerical sorting. But similar to alphanumerical sorting, the resulting orderings depend on the ordering of the dimensions, and the performance improvements are not optimal, since the methods are general and not specialized to specific algorithms.

Multidimensional scaling. Multidimensional scaling algorithms map higher dimensional data to a lower dimensional space while trying to preserve the distances between the points. The most famous result in this domain is the Johnson-Lindenstrauss Lemma [18]. When the target dimension is one, those algorithms return an ordering. However, the fewer dimensions one uses, the worse the distance preservation guarantees get, and in the extreme case of one dimension basically no guarantees can be given.

5 RELATION TO OTHER RESEARCH AREAS

We want to point out some connections of ordering to other research areas, since an exchange between those fields could be very fruitful:

- Anytime algorithms. These are algorithms that can be stopped at any execution step before having finished and still return a feasible — though generally not optimal — solution. For anytime algorithms the order in which data is processed is crucial for the results obtained in early and intermediate stages. On one hand, a deeper understanding of the influence of different properties of the orderings on the workings of algorithms might be obtained by looking at the performance of anytime algorithms. On the other hand, anytime algorithms could better predict the quality of intermediate results if properties of the data ordering were known beforehand.
- Stream processing. In data streams the order of the elements is fixed and algorithms have to adapt to this. The solutions in this field might help to determine the robustness of algorithms to different orderings of their inputs.
- Databases. Alphanumerical sorting is frequently applied in databases, in addition to other techniques such as index structures, to allow for faster execution of certain queries. Here the main challenge is to determine for which columns the overhead of sorting or the creation of an index structure is justified. This is an interesting, somewhat orthogonal direction of research, whose results could very well be combined with new methods for the sorting itself.

6 CONCLUSION

In this vision paper we brought attention to the problem of data orderings. We illustrated its importance by noting that every data user is also a data reader, and reading requires an implicit or explicit ordering of data. We highlighted the often underestimated impact the ordering can have: it can hinder attempts to reproduce research results, but also offer an opportunity to improve the performance of algorithms. Based on these observations, we deduced the necessity of a quality measure for orderings. Such a measure can not only be used as an objective function when ordering data, but also help to make a decision as to whether it is necessary to reorder the data before further usage or not. We described different goals that one can aim at with an ordering, as well as the challenges when trying to achieve such a goal. We furthermore gave an overview over existing approaches and their drawbacks. We want to encourage research regarding the ordering of data, in particular in the challenging area of graphs. We also want to encourage research on the impact different data orderings have on the behaviour of algorithms.

ACKNOWLEDGMENTS

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

REFERENCES

[1] 2001. *UCINET IV DatasetsUCINET IV Datasets.* Retrieved Apr. 12, 2019 from http://vlado.fmf.uni-lj.si/pub/networks/data/Ucinet/zachary.dat
[2] 2013. *Network data.* Retrieved Apr. 12, 2019 from http://www-personal.umich.edu/~mejn/netdata/
[3] 2017. *SIGMOD Most Reproducible Paper Award.* Retrieved Apr. 12, 2019 from https://sigmod.org/sigmod-awards/sigmod-most-reproducible-paper-award/
[4] 2018. *1.6. Nearest Neighbors - scikit-learn 0.20.1 documentation.* Retrieved Apr. 12, 2019 from https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-nearest-neighbors
[5] 2018. *ICLR Reproducibility Challenge.* Retrieved Apr. 12, 2019 from https://www.cs.mcgill.ca/~jpineau/ICLR2019-ReproducibilityChallenge.html
[6] 2018. *Reproducibility in ML Workshop, ICML'18.* Retrieved Apr. 12, 2019 from https://sites.google.com/view/icml-reproducibility-workshop/home
[7] 2018. *ucidata-zachary | Miscellaneous Networks.* Retrieved Apr. 12, 2019 from http://networkrepository.com/ucidata_zachary.php
[8] Anna Beer and Thomas Seidl. 2019. Graph Ordering and Clustering: A Circular Approach. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management.* ACM, 185–188.
[9] Feng Chen, David A. Koufaty, and Xiaodong Zhang. 2009. Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems* (Seattle, WA, USA) *(SIGMETRICS '09).* ACM, New York, NY, USA, 181–192. https://doi.org/10.1145/1555349.1555371
[10] Elizabeth Cuthill and James McKee. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference.* ACM, 157–172.
[11] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. 2011. *Exploratory social network analysis with Pajek.* Vol. 27. Cambridge University Press.
[12] Peter J Denning. 2006. The locality principle. In *Communication Networks And Computer Systems: A Tribute to Professor Erol Gelenbe.* World Scientific, 43–67.
[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[14] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2017. Deep Reinforcement Learning That Matters. *arXiv preprint arXiv:1709.06560* (2017).
[15] David Hilbert. 1891. Ueber die reellen Züge algebraischer Curven. *Math. Ann.* 38, 1 (1891), 115–138.
[16] Intel. 2012. *Intel 64 and IA-32 Architectures Optimization Reference Manual.* https://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf
[17] Stephen C Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254.
[18] William B Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics* 26, 189-206 (1984), 1.
[19] J. P. Kearns and S. DeFazio. 1983. Locality of Reference in Hierarchical Database Systems. *IEEE Transactions on Software Engineering* SE-9, 2 (March 1983), 128–134. https://doi.org/10.1109/TSE.1983.236457
[20] Je-Min Kim and Jin-Soo Kim. 2012. *AndroBench: Benchmarking the Storage Performance of Android-Based Mobile Devices.* Springer Berlin Heidelberg, Berlin, Heidelberg, 667–674. https://doi.org/10.1007/978-3-642-27552-4_89
[21] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18).* ACM, New York, NY, USA, 489–504. https://doi.org/10.1145/3183713.3196909
[22] S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489
[23] Judd H Michael. 1997. Labor dispute reconciliation in a forest products manufacturing facility. *Forest products journal* 47, 11/12 (1997), 41.
[24] Glenn W Milligan. 1981. A Monte Carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika* 46, 2 (1981), 187–199.
[25] Guy M Morton. 1966. A computer oriented geodetic data base and a new technique in file sequencing. (1966).
[26] Eleni Petraki, Stratos Idreos, and Stefan Manegold. 2015. Holistic indexing in main-memory column-stores. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.* ACM, 1153–1166.
[27] W. Rödiger, T. Mühlbauer, P. Unterbrunner, A. Reiser, A. Kemper, and T. Neumann. 2014. Locality-sensitive operators for parallel main-memory database clusters. In *2014 IEEE 30th International Conference on Data Engineering.* 592–603. https://doi.org/10.1109/ICDE.2014.6816684
[28] Twan Van Laarhoven and Elena Marchiori. 2014. Axioms for graph clustering quality functions. *The Journal of Machine Learning Research* 15, 1 (2014), 193–215.
[29] Paul Watson. 2005. Databases in Grid Applications: Locality and Distribution. In *Database: Enterprise, Skills and Innovation*, Mike Jackson, David Nelson, and Sue Stirk (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.
[30] Hao Wei, Jeffrey Xu Yu, Can Lu, and Xuemin Lin. 2016. Speedup Graph Processing by Graph Ordering. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16).* ACM, New York, NY, USA, 1813–1828. https://doi.org/10.1145/2882903.2915220
[31] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.