On Overcoming HPC Challenges of Trillion-Scale Real-World Graph Datasets

Mohsen Koohi Esfahani[1,3], Paolo Boldi[2], Hans Vandierendonck[1], Peter Kilpatrick[1], and Sebastiano Vigna[2]

[1]Queen's University Belfast, United Kingdom
[2]Università degli Studi di Milano, Italy
[3]University of Sistan, Iran

https://blogs.qub.ac.uk/DIPSA/MS-BioGraphs

***Abstract*—Progress in High-Performance Computing in general, and High-Performance Graph Processing in particular, is highly dependent on the availability of publicly-accessible, relevant, and realistic data sets.**

To ensure continuation of this progress, we (i) investigate and optimize the process of generating large sequence similarity graphs as an HPC challenge and (ii) demonstrate this process in creating MS-BioGraphs, a new family of *publicly available real-world edge-weighted graph datasets with up to* 2.5 *trillion edges*, that is, 6.6 times greater than the largest graph published recently. The largest graph is created by matching (i.e., all-to-all similarity aligning) 1.7 billion protein sequences. The MS-BioGraphs family includes also seven subgraphs with different sizes and direction types.

We describe two main challenges we faced in generating large graph datasets and our solutions, that are, (i) optimizing data structures and algorithms for this multi-step process and (ii) WebGraph parallel compression technique.

The datasets are available online on https://blogs.qub.ac.uk/DIPSA/MS-BioGraphs.

***Index Terms*—Big Data Management and Processing, Graph Datasets, High-Performance Computing, Biological Networks, Sequence Similarity Graph, Graph Algorithms**

I. Introduction

Because of the fast increase in the data production rate, and the existence of unstructured connections in these data, High-Performance Graph Processing (HPGP) has to date been widely applied in various fields of science, humanities, and technology. This fact has two main implications for the efficiency of public research and academia that aim to consider the real-world challenges and to design practically-applicable solutions to those challenges. The first effect is the necessity of having **realistic and up-to-date graph datasets** and the second implication is the necessity of considering the effects of new contributions (such as algorithms, processing models, parallelization, and data structures) on **a wide range of input datasets** to cover different application domains.

However, as we detail in Section II, the public graph datasets are small, domain-restricted, and not suitable indicators of real-world data which makes them not ideal for progressing HPGP. To confront this problem, we investigate and optimize the HPC process of generating sequence similarity graphs and demonstrate this process in creating and introducing **MS-BioGraphs**, a new family of real-world graphs with up to 2.5 trillion edges that makes them **the largest real-world public graphs** [1]. This family contains different graph sizes and direction types with similar structures that make them suitable for a range of applications with different input size requirements. Moreover, this graph family shows a very different graph structure in comparison to other real-world graphs (such as social networks and web graphs) and so, complements the current graph collection.

We faced two major challenges in optimizing (i) creation and (ii) compression of these large graphs. The creation of these large datasets is a multi-step process in which (a) the dependency between steps and (b) the processing requirements (i.e., availability of processing resources, memory, and storage) should be considered in the selection and creation of data structures and algorithms of each step. The flow of data between different steps of a multi-step process have important effects on the processing efficiency of the steps. As such, the whole process and processing requirements should be considered and be optimized by **process-wide engineering and design of data structures and algorithms**.

The processing model is one of the main choices in this optimization. The distributed-memory processing model [2], [3] implies two restrictions: (i) fixing the degree of parallelism (i.e., the number of machines/processors involved in the processing) and (ii) limiting the size of processed data to the total memory of the cluster. On the other hand, the storage-based processing model [4], [5] does not practically limit the size of data but deploys only one machine and increases the processing time. Therefore, we designed the processes as multi-step tasks where each step is performed as a distributed parallel computation but without communication between machines. Machines process the partitions independently from each other and use the cluster's shared storage for loading and storing the (intermediary) data.

The second major problem is efficient compression of graph datasets to facilitate fast transfer of the created datasets. The WebGraph Framework [6] provides graph compression at high scale, but the compression process is sequential and we **extend the WebGraph framework by parallelizing compression**.

The extended version of this paper [7] studies some **features of the MS-BioGraphs** showing that (i) while these large biographs follow a skewed degree distribution (similar to other

979-8-3503-2445-7/23©2023 IEEE
https://doi.org/10.1109/BigData59044.2023.10386309

real-world graphs), they expose a different arrangement of edges in comparison to previous graph types by having tight connections between the frequently-occurring high-degree vertices that make their graph structure distinct from other real-world graph types, (ii) weights have a skewed distribution with a tail close to power-law distribution, (iii) the main graph and its large subgraphs exhibit a high-degree of connectivity, and (iv) the asymmetric MS-BioGraphs have a close Push and Pull Locality which is different from social networks and web graphs.

The contributions of this paper are introduction of:

- the HPC-optimized multi-step process of creating large sequence similarity graphs,
- the MS-BioGraphs family as the largest real-world public graphs and publishing them as open datasets,
- parallel compression in the WebGraph framework, and

This paper is structured as: Section II motivates the discussion by exploring the needs for large real-world graphs and considering their effects on progressing HPGP. Section III introduces the processing model and parallel graph compression as our solutions for the major challenges in processing large graphs. Section IV explains the creation process of large graphs and demonstrates it for creating MS-BioGraphs.

II. Motivation

In this section, we consider (i) the necessity of creating updated and cross-domain datasets, (ii) the impacts of these datasets on the progress of HPGP, and (iii) the features of an ideal graph dataset.

A. Why Do We Need Updated and Real-World Graphs?

(1) While synthetic graph generators [8], [9] can create large graphs, *the structural features of synthetic graphs do not match the real-world ones*. E.g., they may expose several gaps in the degree distribution [10] and randomly selected vertices have a large percentage of similar neighbors. As such, the severity of challenges relating to partitioning, locality and load balance in synthetic graphs is often much lower than in real-world datasets. Therefore, the techniques that are sufficient for synthetic graphs may not be applicable for real-world datasets.

(2) Some graph optimizations are dependent on the architecture of machines and it is the tension between data size and the architecture capacities that forms the challenge context and presents the opportunity to design novel data structures, algorithms and processing models. E.g., the design of locality-optimizing algorithms [11], [12], [13], [14], [15] depends on the fact that CPU's cache contains a small portion of the data. By the advent of CPUs with cache sizes of multiple GigaBytes, the locality optimizing algorithms play no role for small datasets as accesses to a large portion of data is covered by cache. Similarly, the progress of distributed graph processing [2], [3] may be slowed down by increase in per-machine memory capacity that is enough to host available datasets. This shows that *without large real-world datasets, it is not possible to progress these architecture-competing HPGP activities*.

(3) Several HPC research fields (such as architecture design, distributed and disk-based processing, and high-performance IO) have tight connections and dependencies on graph algorithms and datasets. *The effectiveness and realness of graph datasets guarantees the efforts on the dependent fields to have real-world impacts*.

(4) Creating a real-world graph dataset provides a representation of the data that *acts as a new source for extracting domain-specific information and knowledge* by deploying graph algorithms. As an example, sequence similarity graphs have several usages in biology including sequence clustering [16], predicting pseudogene functions, effective selection of conotoxins [17], predicting evolution [18] and gene transfer [19].

A comprehensive graph representation of the data is also beneficial (i) to validate previous hypotheses (that have been verified on a small portion of data) in a wider perspective and (ii) to provide new opportunities to make new contributions by considering the new patterns and connections revealed in graph representation.

B. Why Do We Need Different Types of Real-World Graphs?

(1) Previous studies have shown that different real-world graph types exhibit contrasting behaviors with graph analytic algorithms and optimizations [20]. Examples include the long execution time of small road networks in Label Propagation Connected Components [21], [22] and the different impact of similarity and locality in web graphs and social networks [23]. This implies that a wider range of graph types will be necessary *to better study and comprehend the structure of graphs and to compare them*. This better understanding of different graph types and their structures will also be helpful to *design synthetic graph generators with greater similarity to real-world graphs* (Section II-A).

(2) A wide range of real-world datasets facilitates *cross-domain evaluation of the new contributions* and provides broad and correct assessment across a variety of use cases (i.e., better pruning of the falsifiable insights [24]). Also, we will have the opportunity to improve several graph algorithms and optimizations that exploit the structure of graphs [3], [25], [12], [26], [27], [28], [29].

C. Creating Real-World Graphs: An HPC Problem

(1) Creating real-world graphs is a time-consuming process [30], [31], [32] and is periodically repeated. As the size of input dataset (connections in web graphs, links in social networks, or similarities in sequences) grows, greater amounts of computations and processing resources are required.

(2) Some tasks in creating graphs are widely used in deploying graph algorithms, such as format conversion, transposition, and symmetrization, are time-consuming. Optimizing these steps is directly transferred in graph algorithms.

D. The Current Largest Graph Datasets

At present, the last largest public graph dataset we are aware of is the Software Heritage 2022-04 version-control-history graph[1] [31] with 376 billion edges that was published in 2022.

The largest web graph is Web Data Commons 2012 hyperlink graph[2] [32], with 128 billion edges that was published about 9 years ago. The largest social network graph is a snapshot of Twitter on 2010 [33] with 1.5 billion edges.

These graphs are outdated and/or not indicative of the growth in size of data that is happening in the real world.

E. What Is An Ideal Graph Dataset?

The discussions in this section show that a new family of graphs should ideally (i) be backed by a real-world phenomenon, (ii) cover a wide range of graph sizes to make it suitable for different applications, (iii) exhibit new structural features that are not seen in other real-world graphs, (iv) contain graphs much larger than existing ones and in line with the exponential growth rate of the worldwide datasets [3] , and (v) be available as open datasets to research communities.

III. HPC Challenges and Our Solutions

In this section, we present two major challenges we faced in creating large datasets. Section III-A explores how to efficiently utilize a small cluster for processing large datasets. Section III-B explores how to parallelize the compression process of the large weighted graph datasets. We demonstrate our solutions for these two challenges in Section IV where we detail creation of MS-BioGraphs.

A. The Processing Model

We search for a processing model that *(i) dynamically adjusts the degree of parallelism (i.e., the number of machines/processors involved in the processing) and (ii) does not restrict the size of processed data to the total memory of the cluster* while machines have access to a shared storage that hosts the datasets and the intermediary data.

The distributed-memory processing model [2] sets an upper bound for the size of dataset based on the total memory of the cluster. This model also makes the waiting time of jobs dependent on the size of the requested resources. If we need a greater number of machines, we may need to wait for a longer time before scheduling the job. Therefore, to optimize cluster utilization it is necessary *to minimize the waiting time*.

The storage-based processing model [4], [5], on the other hand, does not practically limit the size of data, but deploys one machine and increases the processing time.

To satisfy the mentioned requirements, we deploy a distributed model in which algorithms are designed as a number of sequential steps with parallel workloads per step. In each step, machines contribute to the total processing independently of each other and the input and output data for each processing slot is loaded from and stored to the shared storage. So, machines only communicate (a) to the shared storage to retrieve/store data and (b) to the scheduler to receive a partition of a task or to inform completion of a partition.

In this way, each machine requires a memory size that is enough to complete a partition. This facilitates processing the datasets whose sizes are greater than the available memory.

Moreover, as the machines do not communicate with each other, each step can be started as soon as at least one machine becomes available and new machines can join/leave a running step. This (i) relaxes the assumption of permanent availability of a fixed number of resources during the whole execution time, (ii) minimizes the waiting time, and (iii) optimizes cluster utilization.

B. Parallelizing Graph Compression

As MS-BioGraphs have binary sizes of up to 20 TeraBytes, it is necessary to compress them to make their storage, transfer over the network, and processing more efficient.

To that end, we used the WebGraph framework[4] [6] which is an open-source graph compression framework that has been continuously maintained and updated during the last 20 years. This framework provides graph compression and includes a rich set of graph operations and analytics. Moreover, the users of languages and frameworks with WebGraph support, such as Hadoop, C++, Python, and Matlab, benefit from direct access to MS-BioGraphs.

WebGraph provides facilities for storing edge-labelled graphs. Labels are stored contiguously in a bitstream in edge order (i.e., lexicographical source/destination order), and an offset file containing pointers to the start of the sequence of labels associated with the neighbors of a vertex. The bitstream can be loaded into memory or memory-mapped to support graphs with a larger size than core memory. Moreover, offsets are loaded using the Elias–Fano representation, a quasi-succinct data structure that brings the required storage space for each offset to a few bits [34].

Historically, the design of the labelled facilities in WebGraph decoupled the compression of the underlying graph and the storage of the labels. This approach has the advantage of implementing a clear separation of concerns and makes it possible to pair compression schemes and label storage schemes arbitrarily.

However, in processing MS-BioGraphs, it became clear that the approach is very inefficient in a number of situations, and in particular when transposing, symmetrizing or permuting very large labelled graphs. In all of these operations, graph edges are first divided into batches that are sorted in core memory using a parallel sorting algorithm and compressed on disk; then, one can traverse the resulting transposed (or symmetrized, or permuted) graph sequentially. However, this traversal is quite expensive as the compressed representation is optimized for space and ease of storage, but not for speed of traversal; ideally, the transformed temporary graph should be traversed exactly once.

[1] https://docs.softwareheritage.org/devel/swh-dataset/graph/dataset.html

[2] http://webdatacommons.org/hyperlinkgraph/2012-08/download.html

[3] https://www.idc.com/getdoc.jsp?containerId=US49018922 and https://www.statista.com/statistics/871513/worldwide-data-created

[4] https://webgraph.di.unimi.it/

The previous design was thus at odds with this approach, as two passes were necessary to compress the graph and to store the labels. Moreover, the current implementation of labelled graphs did not allow for parallel storage—a fundamental requirement in processing large-scale graphs.

We extended the WebGraph framework in two directions: in the first phase, we extended labelled graphs to support parallel compression of the underlying graph. This first extension decreased significantly the compression time (scaling is linear in the number of cores) but did not solve the problem of multiple passes over the temporary representation.

In the second phase, we partially violated the decoupled design of labelled graphs in WebGraph, adding to the compression phase of the main storage format class of WebGraph, `BVGraph` (that compresses and stores the underlying graph), an option to store the labels at the same time. This created a dependency of `BVGraph` on a *specific* labelled graph implementation; that is, the parallel and simultaneous compression of graph and labels can only happen with a specific, bitstream-based label representation. However, since recompressing the underlying graph in a different format can be performed with very low cost, and the bitstream-based label implementation is the only presently-available option, the implementation remains, in practice, highly (albeit not completely) decoupled.

IV. Generating MS-BioGraphs

In Section III, we introduced solutions for major challenges in processing large graphs. In this section, we demonstrate those solutions to design and implement the algorithms required in the different steps of creating the MS-BioGraphs.

A. Terminology

A directed graph $G = (V, E)$ is defined by a set of vertices V and a set of edges (a.k.a. arcs) $E \subseteq V \times V$; an edge is an ordered pair (u, v) that indicates an edge from vertex u to v. In a directed weighted (a.k.a., labelled) graph $G_w = (V, E)$, the set of edges is a subset of $V \times V \times \mathbb{N}$, where $(u, v, w) \in E$ represents an edge from u to v with weight w. The *undirected weighted graph* $G_u = (V, E)$ is defined as a directed weighted graph where for each $(u, v, w) \in E$, there is an edge $(v, u, w) \in E$.

A protein sequence is a string of letters, each letter representing one of the 20 canonical amino acids. Each of these 20 amino acids is represented by a letter. Similarity is calculated by comparing aligned amino acids whose matches are not directional and match values are derived from a symmetrical matrix (e.g., PAM and BLOSUM). Two sequences may have multiple matches as the start point of match is not restricted. For a set of protein sequences, the *sequence similarity graph* is a weighted undirected graph whose vertices represent proteins and with an edge (u, v, w) expressing the fact that the similarity between proteins u and v is w.

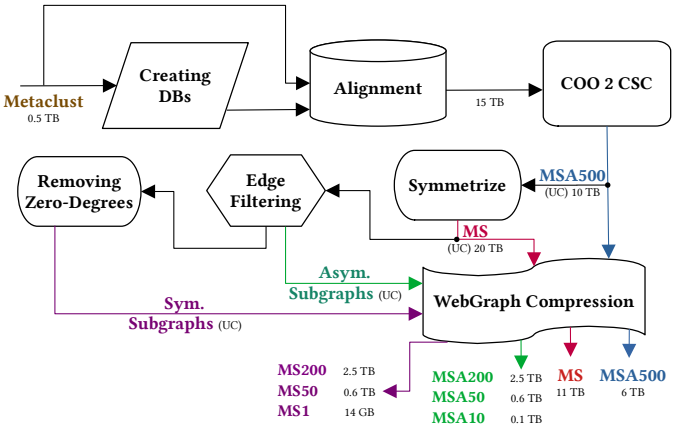

Fig. 1: Creation Steps (UC: uncompressed)

B. Input Dataset & Environment Setup

Inspired by HipMCL [35], we use the Metaclust dataset[5] [36] that contains 1.7 billion protein sequences in FASTA format. We collected all similarities produced by the LAST sequence alignment algorithm[6] [37] Version 1293. We selected LAST as aligner as it shows better single-machine performance and has been widely used and maintained since its publication in 2011.

Sequence matching by LAST is performed in two steps: (i) creating a database (DB) from sequences using a program called `lastdb` and (ii) aligning the sequences of a file against the created database using `lastal` (with `PAM30` scoring matrix and default values for other options) that outputs the matched sequences and their scores.

We used 9 machines that are set up in a job-sharing cluster and not all machines (and not all of their cores and memory capacity) were permanently available in all steps. The cluster is backed by a 2 PetaBytes Lustre file system that provided up to 8 Gbps bandwidth in our experiments.

We have implemented most of our algorithms as extensions to the LaganLighter framework[7] [29], in the `C` language with `OpenMP` parallelization.

C. Process-Wide Data Structures and Algorithms Engineering and Design

In this section, we design the general process of creating MS-BioGraphs as a multi-step process by considering the flow of data and dependencies of steps.

(1) To create MS-BioGraphs, we compute all-against-all matching of the sequences. We match each sequence only to sequences with lower IDs. This produces a directed weighted graph whose symmetric version represents all the matches and their scores. This imposes the cost of symmetrization but reduces the alignment computations by 50%.

We have the following steps as depicted also in Figure 1. First, we need to create LAST database(s) using `lastdb`

[5]https://metaclust.mmseqs.com/2018_06/metaclust_all.gz
[6]https://gitlab.com/mcfrith/last
[7]https://blogs.qub.ac.uk/DIPSA/LaganLighter/

and then call `lastal` to create the similarities, i.e., the asymmetric graph in the coordinate format (COO). The next step is converting the COO graph to the Compressed Sparse Columns (CSC) [38] format which is followed by symmetrizing and compression. We also create some subgraphs to support research studies with different graph size and direction requirements. Therefore an "Edge Filtering" step is required to create subgraphs and we need to remove zero-degree vertices.

(2) We need to consider whether to run the `lastal` in parallel mode on one single machine. Our preliminary evaluation showed that the `lastal` does not continuously engage all processors. The other problem is the long processing time (366 hours) as a result of deploying one machine.

However, there is a more important implication of running one instance of `lastal` and that is its output. The output of the "Alignment" step is used as input to the "COO to CSC" step. The CSC format consists of two arrays: the `offsets` array and the `edges` array. The `offsets` array is indexed by a vertex ID to identify the index of the first edge of that vertex in the `edges` array. In creating the `edges` array, we need to read edges from the COO graph and to write each edge based on the offset identified by its destination endpoint. This requires random write accesses to the `edges` array which requires 8 Bytes per edge (4 bytes for the ID of the source endpoint and 4 Bytes for the weight), or about 10 TB memory.

As no machine has this size of memory, the other option is to convert the subgraphs of the COO format to the CSC subgraphs and then merge the CSC subgraphs to create the CSC graph. While this can be done in a distributed way (Section III-A), it implies one extra reading and one extra writing of all edges.

So, we face three problems: (i) load imbalance of `lastal` in parallel mode, (ii) long execution time in the "Alignment" step, and (iii) storage overhead in the "COO to CSC" step.

Our solution for this cross-step problem is to partition the input dataset that converts the adjacency matrix of the graph to a number of blocks. The graph construction is now performed by calling concurrent instances of `lastal` for different blocks, (i.e., pair of partitions) and each instance is run in sequential mode. This optimizes load balance, increases the cluster utilization, and significantly reduces the computation time by concurrently deploying multiple machines (Section III-A).

Each block of the adjacency matrix is stored in a separate file and allows us to efficiently create the CSC graph in the distributed model by partially creating the CSC graph for each partition where it is only needed to load the relevant blocks (for partition p_j, all edges exist in (p_i, p_j) blocks where $i \leq j$) and we do not need to keep the whole `edges` array in the memory. By having a sufficiently large number of partitions, we ensure the memory space required for a slice of the `edges` array is available on each machine.

(3) The output of "COO 2 CSC" (MSA-500) is symmetrized to create the main graph (MS graph). This is efficiently done in the distributed model by transposing and merging the transposed graph with the CSC graph. It is possible to merge the "COO 2 CSC" and "Symmetrize" steps into one step by transposing each partition while creating the CSC format and then merging the transposed subgraphs and CSC. However, this results in concurrency of two write and one read storage operations for all edges that may overload the storage bandwidth. Our evaluation shows that overloading storage bandwidth in our cluster (with per-user bandwidth limit) imposes longer delays. However, merging these steps is beneficial for clusters that provide greater storage bandwidth limit.

(4) "Edge Filtering" and "Removing Zero-Degrees" are efficiently done in the distributed model. The last step is creating the compressed version in WebGraph format which deploys a shared-memory model.

D. Created Graphs

The MS graph [39] has 2.5 trillion edges and is created by symmetrizing the MSA500 [40]. The MS graph is used to create the subgraphs. The undirected subgraphs MS200 [41], MS50 [42], and MS1 [43] are created by using the weight of edges as the filtering metric. For the directed subgraphs MSA200 [44], MSA50 [45], and MSA10 [46] the vertex-relative weight has been used as sampling metric.

V. Conclusion

To provide a more effective HPGP research environment by accessing realistic and updated datasets with better coverage of various application-domains, this paper presents solutions for the challenges in creation and compression of large graphs. We demonstrated the effectiveness of our solutions in generating the **MS-BioGraphs**, with up to 2.5 trillion edges which is 6.6 times greater than the previous largest real-world graph. The full version of this paper [7] presents a detailed explanation of generation of MS-BioGraphs and a comparative study of MS-BioGraphs to other real-world graphs.

Access to MS-BioGraphs Datasets

The datasets are open to public access on: https://blogs.qub.ac.uk/DIPSA/MS-BioGraphs.

Acknowledgements

This work was partially supported by (i) the High Performance Computing center of Queen's University Belfast and the Kelvin-2 supercomputer (UKRI EPSRC grant EP/T022175/1) and (ii) the SERICS project (PE00000014) under the NRRP MUR program funded by the EU - NGEU. First author was also supported by a scholarship from the Department for the Economy, Northern Ireland and Queen's University Belfast.

References

[1] M. Koohi Esfahani, P. Boldi, H. Vandierendonck, P. Kilpatrick, and S. Vigna, "Dataset announcement: MS-BioGraphs, trillion-scale public real-world sequence similarity graphs," in *IISWC'23*. IEEE Computer Society, 2023.

[2] Harshvardhan, A. Fidel, N. M. Amato, and L. Rauchwerger, "KLA: A new algorithmic paradigm for parallel graph computations," ser. PACT '14. New York, NY, USA: ACM, 2014, p. 27–38.

[3] R. Chen, J. Shi, Y. Chen, and H. Chen, "PowerLyra: Differentiated graph computation and partitioning on skewed graphs," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15. New York, NY, USA: Association for Computing Machinery, 2015.

[4] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-Stream: Edge-centric graph processing using streaming partitions," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 472–488.

[5] Y.-Y. Jo, M.-H. Jang, S.-W. Kim, and S. Park, "Realgraph: A graph engine leveraging the power-law distribution of real-world graphs," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 807–817.

[6] P. Boldi and S. Vigna, "The webgraph framework i: Compression techniques," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 595–602.

[7] M. Koohi Esfahani, P. Boldi, H. Vandierendonck, P. Kilpatrick, and S. Vigna, "MS-BioGraphs: Sequence similarity graph datasets," *CoRR*, vol. abs/2308.16744, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.16744

[8] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining." in *SDM*. SIAM, 2004, pp. 442–446.

[9] H. Park and M.-S. Kim, "Trilliong: A trillion-scale synthetic graph generator using a recursive vector model," ser. SIGMOD '17. New York, NY, USA: ACM, 2017, p. 913–928.

[10] H. Cao, Y. Wang, H. Wang, H. Lin, Z. Ma, W. Yin, and W. Chen, "Scaling graph traversal to 281 trillion edges with 40 million cores," in *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 234–245.

[11] J. Arai, H. Shiokawa, T. Yamamuro, M. Onizuka, and S. Iwamura, "Rabbit order: Just-in-time parallel reordering for fast graph analysis," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. USA: IEEE, 2016, pp. 22–31.

[12] M. Koohi Esfahani, P. Kilpatrick, and H. Vandierendonck, "LOTUS: Locality optimizing triangle counting," in *27th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP 2022)*. ACM, 2022, p. 219–233.

[13] M. Drescher, M. A. Awad, S. D. Porumbescu, and J. D. Owens, "Boba: A parallel lightweight graph reordering algorithm with heavyweight implications," 2023.

[14] V. Balaji, N. C. Crago, A. Jaleel, and S. W. Keckler, "Community-based matrix reordering for sparse linear algebra optimization," in *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2023, pp. 214–223.

[15] A. Trostanovsky, "Vertex-and-edge ordering for faster parallel graph processing," Master's thesis, University of British Columbia, 2023. [Online]. Available: http://dx.doi.org/10.14288/1.0437140

[16] A. J. Enright, S. Van Dongen, and C. A. Ouzounis, "An efficient algorithm for large-scale detection of protein families," *Nucleic Acids Research*, vol. 30, no. 7, pp. 1575–1584, 04 2002.

[17] R. A. Mansbach, S. Chakraborty, T. Travers, and S. Gnanakaran, "Graph-directed approach for downselecting toxins for experimental structure determination," *Marine Drugs*, vol. 18, no. 5, 2020.

[18] B. L. Hie, K. K. Yang, and P. S. Kim, "Evolutionary velocity with protein language models predicts evolutionary dynamics of diverse proteins," *Cell Systems*, vol. 13, no. 4, pp. 274–285, 2022.

[19] E. Corel, P. Lopez, R. Méheust, and E. Bapteste, "Network-thinking: graphs to analyze microbial complexity and evolution," *Trends in Microbiology*, vol. 24, no. 3, pp. 224–237, 2016.

[20] M. Koohi Esfahani, P. Kilpatrick, and H. Vandierendonck, "Locality analysis of graph reordering algorithms," in *2021 IEEE International Symposium on Workload Characterization (IISWC'21)*. USA: IEEE Computer Society, 2021, pp. 101–112.

[21] D. Nguyen, A. Lenharth, and K. Pingali, "A lightweight infrastructure for graph analytics," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 456–471.

[22] M. Sutton, T. Ben-Nun, and A. Barak, "Optimizing parallel graph connectivity computation via subgraph sampling," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2018, pp. 12–21.

[23] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 587–596.

[24] K. R. Popper, "The logic of scientific discovery," *Central Works of Philosophy v4: Twentieth Century: Moore to Popper*, vol. 4, p. 262, 2015.

[25] M. Koohi Esfahani, P. Kilpatrick, and H. Vandierendonck, "Exploiting in-hub temporal locality in SpMV-based graph processing," in *50th International Conference on Parallel Processing*, ser. ICPP 2021. New York, NY, USA: Association for Computing Machinery, 2021.

[26] ——, "MASTIFF: Structure-aware minimum spanning tree/forest," in *36th ACM International Conference on Supercomputing*. New York, NY, USA: Association for Computing Machinery, 2022.

[27] ——, "SAPCo Sort: Optimizing degree-ordering for power-law graphs," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE Computer Society, 2022.

[28] ——, "Thrifty Label Propagation: Fast connected components for skewed-degree graphs," in *2021 IEEE CLUSTER*. USA: IEEE Computer Society, 2021, pp. 226–237.

[29] M. Koohi Esfahani, "On designing structure-aware high-performance graph algorithms," Ph.D. dissertation, Queen's University Belfast, 2022. [Online]. Available: https://blogs.qub.ac.uk/dipsa/ODSAHPGA

[30] P. Boldi, A. Marino, M. Santini, and S. Vigna, "Bubing: Massive crawling for the masses," *ACM Trans. Web*, vol. 12, no. 2, Jun. 2018.

[31] P. Boldi, A. Pietri, S. Vigna, and S. Zacchiroli, "Ultra-large-scale repository analysis via graph compression," in *2020 (SANER)*. London, ON, Canada: IEEE Computer Society, 2020, pp. 184–194.

[32] R. Meusel, S. Vigna, O. Lehmberg, and C. Bizer, "The graph structure in the web – analyzed on different aggregation levels," *The Journal of Web Science*, vol. 1, no. 1, pp. 33–47, 2015.

[33] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 591–600.

[34] S. Vigna, "Broadword implementation of rank/select queries," in *Experimental Algorithms*, C. C. McGeoch, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 154–168.

[35] A. Azad, G. A. Pavlopoulos, C. A. Ouzounis, N. C. Kyrpides, and A. Buluc, "Hipmcl: a high-performance parallel implementation of the markov clustering algorithm for large-scale networks," *Nucleic Acids Research*, vol. 46, no. 6, 1 2018.

[36] M. Steinegger and J. Söding, "Clustering huge protein sequence sets in linear time," *Nature Communications*, vol. 9, 06 2018.

[37] S. M. Kiełbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith, "Adaptive seeds tame genomic sequence comparison," *Genome research*, vol. 21, no. 3, pp. 487–493, 2011.

[38] Y. Saad, "Sparskit: a basic tool kit for sparse matrix computations - version 2," 1994.

[39] M. Koohi Esfahani, P. Boldi, H. Vandierendonck, P. Kilpatrick, and S. Vigna, "MS-BioGraphs - MS," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MS, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820808

[40] ——, "MS-BioGraphs - MSA500," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MSA500, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820810

[41] ——, "MS-BioGraphs - MS200," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MS200, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820812

[42] ——, "MS-BioGraphs - MS50," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MS50, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820819

[43] ——, "MS-BioGraphs - MS1," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MS1, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820827

[44] ——, "MS-BioGraphs - MSA200," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MSA200, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820815

[45] ——, "MS-BioGraphs - MSA50," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MSA50, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820821

[46] ——, "MS-BioGraphs - MSA10," http://blogs.qub.ac.uk/DIPSA/MS-BioGraphs-MSA10, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7820823