2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 979-8-3503-6460-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/IPDPSW63119.2024.00105

# ECG: Expressing Locality and Prefetching for Optimal Caching in Graph Structures

Abdullah T. Mughrabi*, Morteza Baradaran*, Ahmed Samara†, and Kevin Skadron*
*Department of Computer Science, University of Virginia, Charlottesville, Virginia
Email: {atmughra, rgq5aw, skadron}@virginia.edu
†Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina
Email: asamara@ncsu.edu

***Abstract*—Despite state-of-the-art caching strategies, graph analytics pose a significant challenge for prefetching and replacement policies, as their access patterns are often random with low locality. Prior attempts at addressing these issues involved architectural solutions such as specialized graph prefetchers, compressed graph representation stored in a reserved cache section, or graph reordering for custom replacement policies. However, combining these solutions required extensive modifications to the processor architecture. In this paper, we propose Expressive Caching for Graphs (ECG), which masks the graph clustering knowledge with previous cache optimization techniques into the unused bits (ECG bits) of the graph vertex ID. The processor pipeline's specialized graph addressing mode can extract the vertex ID from the graph edge list to request property data while utilizing the masked ECG bits to determine the replacement policy for better caching and prefetching the next related vertex.**

***Index Terms*—Caching, Reordering, Graph Processing, hardware/software co-design**

## I. Introduction

In the ever-evolving field of data science, graph analytics is widely used to interpret complex relationships in large datasets across various domains, such as social networks, recommender systems, and bioinformatics [1]–[3]. A graph, composed of vertices and edges, represents the relationships between various entities in the dataset. Fig. 1 illustrates how a graph processing technique, i.e., vertex-centric model, uses a Compressed Sparse Row (CSR) matrix data structure to represent the graph in memory. As explained in section II, the CSR model makes it easy to access neighboring vertices, random neighbor IDs for each visited vertex results in irregular memory access patterns, leading to poor spatial and temporal cache locality [4]–[7], which causes delays in retrieving data from lower memory levels. Additionally, stride prefetching [8]–[11] techniques are less effective due to unpredictable access patterns in graph processing.

Several techniques, including Practical Optimal Cache Replacement (P-OPT) [12] and GRAph SPecialized (GRASP) Last-Level Cache (LLC) management [13]–[15], have focused on developing prefetch and cache replacement policies specifically designed for graph analytics. However, these solutions may necessitate significant modifications to the processor architecture, which can be advantageous but challenging to implement for most existing systems that rely heavily on CPU architecture and design features.

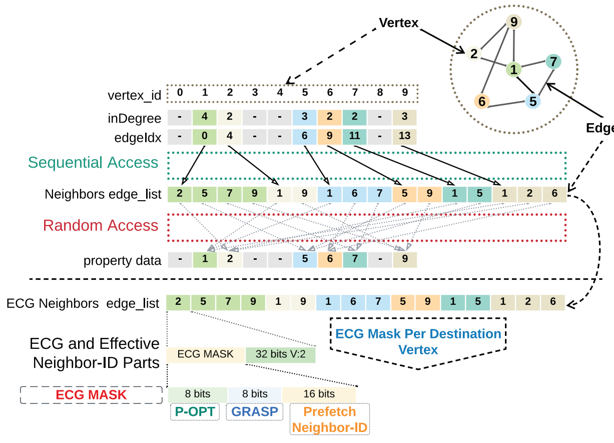


Fig. 1: Graph representation in CSR format, accessing vertex data is random and fine-grained, with a high cache miss rate.

In contrast, our novel solution, Expressive Caching for Graphs (ECG,) utilizes the power-law distribution present in real-world graphs [16]–[21], where most vertices have few connections, and a small number of vertices (the hubs) have many connections. ECG encodes the clustering information derived from prior graph-specialized caching techniques into the unused bits of the graph vertex ID, called ECG bits. The processor pipeline, equipped with a specialized graph-addressing mode, can extract the vertex ID from the graph edge list to request property data while using the masked ECG bits to dictate the replacement policy and predict the next related vertex to prefetch. Consequently, ECG bits increase cache utilization and reduce miss rate, thus improving the overall performance of graph processing without requiring any major architectural modifications.

## II. Background: Graph Reordering and Cache Optimizations

Graph Reordering [7], [22]–[33] involves assigning new vertex labels that consolidate neighbor vertex data accesses based on their temporal or spatial reuse, as illustrated in Fig. 2. **Degree Based Grouping (DBG)** [14] utilizes coarse grain degree-based sorting on a graph by dividing the vertices into different buckets based on their degree range. **Rabbit Order**

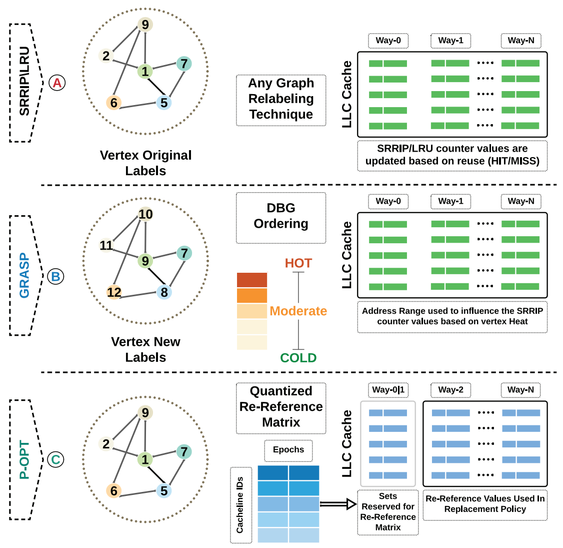

SRRIP/LRU
Ⓐ
Vertex Original Labels
Any Graph Relabeling Technique
LLC Cache
Way-0
Way-1
Way-N
SRRIP/LRU counter values are updated based on reuse (HIT/MISS)
GRASP
Ⓑ
Vertex New Labels
DBG Ordering
HOT
Moderate
COLD
Address Range used to influence the SRRIP counter values based on vertex Heat
P-OPT
Ⓒ
Quantized Re-Reference Matrix
Epochs
Cacheline IDs
Way-0|1
Way-2
Sets Reserved for Re-Reference Matrix
Re-Reference Values Used In Replacement Policy

Fig. 2: How different hardware graph cache optimization techniques interact with LLC.

[18], [33] achieves hierarchical community-based ordering by using a parallel incremental aggregation function and modifying it to serve as a new criterion of vertex ID mapping. **Gorder** [34] analyzes the graph through a sliding window technique to determine the optimal neighbor permutation that provides the best caching.

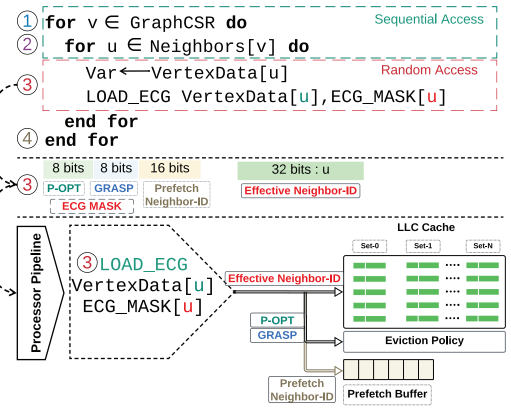
The LLC cache tends to be the most impacted in graph algorithms, making it the focus of optimization efforts for graph cache hardware, some of which are depicted in Fig. 2. **Re-reference Interval Prediction (RRIP)** Ⓐ is a cache management scheme that uses a probabilistic approach to classify cache blocks as low- or high-reuse when new blocks are inserted. **GRAph SPecialized (GRASP) LLC management** Ⓑ improves the RRIP for graph structures and introduces specialized cache management policies prioritizing hot vertices in the LLC by utilizing DBG with a lightweight interface that helps hardware pinpoint hot vertices among irregular access patterns. **Practical Optimal Cache Replacement (P-OPT)** Ⓒ employs Transpose-based Cache Replacement (T-OPT,) a policy that utilizes next-reference data for all graph information.

## III. ECG Architecture

### A. Creating ECG bits

As mentioned before, large-scale power-law graphs exhibit a lower number of unique vertex IDs in a specific range compared to the total number of edges. Hence, using 64-bit or 32-bit representation for a vertex ID, as benchmark tools often do [20], may be excessive considering the range of the graph's unique vertex IDs. Expressive Caching for Graphs (ECG) offers a solution to utilize the unused bits in vertex ID more effectively since a 32-bit or less representation is sufficient for most graphs.

As shown in Fig. 1, ECG incorporates P-OPT, GRASP, and prefetch information into the vertex ID. Out of the 64-bit representation for the vertex ID, ECG designates:



```
① for v ∈ GraphCSR do
②   for u ∈ Neighbors[v] do
③     Var ⟵ VertexData[u]
      LOAD_ECG VertexData[u],ECG_MASK[u]
    end for
④ end for
```

Fig. 3: How different hardware graph cache optimization are encoded into ECG bits, while ECG bits are handled within the processor pipeline to improve performance.

- 32 bits for the vertex ID (which is sufficient for most large-scale graphs.)
- 8 bits for GRASP metadata (affects cache eviction policy.) Mark the vertex as hot/warm/cold. To preserve the integrity of the original reorder, the GRASP eviction policy should not be applied with a DBG reorder. Instead, DBG (GRASP) data can be masked in ECG bits while maintaining the original graph labels.
- 8 bits for P-OPT re-reference matrix (affects cache eviction policy.) These 8 bits represent the re-reference matrix from the P-OPT quantization algorithm, which are typically stored in preserved sets of the LLC cache.
- 16 bits to encode the ”hot” prefetch vertices. These hot vertices have high connectivity or frequent access degrees.

In summary, the ECG strategy efficiently encodes P-OPT, GRASP, and clustering information in the unused bits of the vertex IDs to improve data management for caching and prefetching. In the past, P-OPT required maintaining one or more ways of the LLC for the re-reference matrix, but ECG eliminates this overhead through its proposed scheme.

### B. ECG Pipeline and Instruction Flow

In Fig. 3, the processor pipeline ③ uses a specialized graph instruction with ECG addressing mode to extract vertex IDs from the edge list and request related property data. At the same time, the masked ECG bits guide the cache replacement policy and facilitate the prefetching of the next predicted ”hot” vertices. Despite extending the edge list to a 64-bit representation, ECG's streaming behavior ensures minimal impact on performance. The streaming pattern, sequential and one-time processing, of the edge list in graph structure data ensures that the growth in its size only raises the memory footprint without affecting the time complexity of graph processing.

## IV. Evaluation and Results

The implementation used for evaluation is part of the OpenGraph framework [35], and instructions on how to run it are available on https://github.com/atmughrabi/ECG.

TABLE I: System environment for ECG evaluation

| | Host CPU |
|---|---|
| Model‖Cores‖Threads | Intel Xeon Silver 4216‖16‖16 |
| Clock Speed | 2.10 GHz |
| Memory | 256 GB |
| OS | Ubuntu 22.04 LTS |
| L1‖L2‖L3 | 512 kB (8-way)‖16 MB (16-way)‖22 MB (11-way) |

TABLE II: Graph workloads used in ECG evaluation

| Graph | #Vertices | #Edges | Avg. degree | References |
|---|---|---|---|---|
| amazon-2008(amazon) | 735324 | 5158388 | 7 | [23] |
| cnr-2000(cnr) | 325558 | 3128710 | 9 | [23] |
| dblp-2010(dblp) | 326184 | 807700 | 2 | [23] |
| enron(enron) | 69245 | 274608 | 3 | [23] |

## *A. Methodology*

**System Environment:** The specifications of the host system and its cache hierarchy are listed in Table. I. Additionally, Gorder, DBG, and Rabbit Order were tested on the same host system using reference implementations to accurately compare each technique.

**Cache Simulation:** To showcase ECG's cache performance, we have developed a basic trace-driven cache simulator. This tool extracts the access pattern of the designated graph algorithm during its execution and measures the miss rate. We test ECG with various cache sizes ($32\,\mathrm{kB}$-$1\,\mathrm{MB}$) for Gorder, DBG, and Rabbit orders. It is worth noting that due to the size of the graphs we choose smaller cache sizes, the full version of the paper will conduct larger simulations with larger graphs.

**Graph Datasets:** In this study, we focus on evaluating the graphs from the Laboratory for Web Algorithmics (LAW) [23]. These graphs are designed to improve cache locality by reordering clustered vertices. Table II in the graph collection includes real-world examples of graphs that test read/write scenarios to identify potential bottlenecks in the simulated cache performance. To demonstrate the effectiveness of the reordering technique for generating cache-optimized labels, the graphs use random (Rand) initialized labels.

**Algorithm:** We investigate the concepts behind ECG utilizing PageRank [20], [36]–[38], an algorithm that operates iteratively on a graph. PageRank algorithm assigns a popularity score (rank) to each vertex during each iteration by traversing the graph vertices. The rank of a vertex is determined by the popularity of its incoming edges (in-neighbors.) The algorithm continues to iterate until the ranks converge within a specified error margin.

## *B. Cache Policy Performance and ECG prefetching*

In Fig. 4d, Fig. 4e, and Fig. 4f, the cache uses ECG prefetching for loading the hot neighboring vertex of the processed node. This results in lower cache miss rates as the next vertex from the same cluster is often already prefetched. However, when prefetching is disabled in the first row of Fig. 4, a higher cache miss rate is consistent regardless of the policy or the ordering technique. To illustrate, let us consider the "amazon" dataset using the "PLRU" replacement policy. When using the "Rand-Order" strategy, the cache miss rate increases from $72.76\,\%$ with prefetching enabled in Fig. 4e to $80.48\,\%$ with prefetching disabled in Fig. 4b. The "Rabbit" strategy also benefits from prefetching, although the miss rate improvement is smaller than the "Rand-Order" strategy. Based on the previous observations, ECG prefetching reduces the cache miss rate across all datasets in table II. This trend is consistent for all evaluated replacement policies, with higher cache miss rates observed in Fig. 4a, Fig. 4b, and Fig. 4c (prefetching disabled) compared to Fig. 4d, Fig. 4e, and Fig. 4f (prefetching enabled.)

## *C. GRASP-DBG Ordering vs. ECG-DBG Masking*

Fig. 5a and Fig. 5b showcase the impact of maintaining the original ordering of the graph while using ECG MASK as a standalone optimizations against GRASP cache eviction policy with DBG reordering.

**GRASP vs. ECG Rand+MASK:** Generally, GRASP seems to be outperforming ECG MASK for randomly relabeled graphs. This is a projected result since GRASP depends on DBG reordering for its eviction policy, unlike ECG MASK, which keeps its order intact. For example, for the 'amazon' dataset, the cache miss rate using the DBG strategy is lower for GRASP ($70.4\,\%$) compared to MASK ($72.04\,\%$.) This pattern repeats for 'cnr' and 'dblp' and 'enron'.

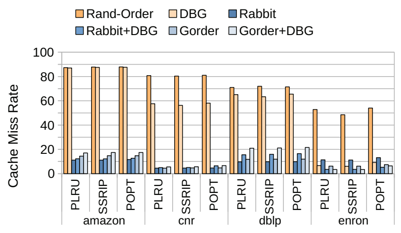
**Rabbit+GRASP vs. ECG Rabbit+MASK:** When paired with Rabbit, GRASP tends to have higher cache miss rates than ECG MASK due to GRASP depending on DBG reordering, which changes the original optimal Rabbit order. For instance, if we study Fig. 5b 'amazon' with ECG Mask there is a reduction of $63.44\,\%$ in cache miss rate.

**Gorder+GRASP vs. ECG Gorder+MASK:** Similarly to Rabbit order, GRASP has higher cache miss rates than MASK for 'amazon' a reduction of $29.03\,\%$ in cache miss rate. Although Gorder is known to be more efficient in traversal algorithms such as Breadth First Search (BFS), which were not evaluated in this study, its high preprocessing overhead is not ideal. Therefore, Rabbit order could be a more practical reordering technique to be used in conjunction with ECG.

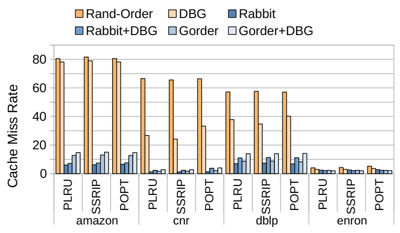
## *D. DBG Reordering Impact On Relabeled Graphs*

The GRASP strategy depends on reordering the graph structure using DBG, which can disrupt the original order of the data. This is evident in the increased cache miss rates when GRASP is combined with the Rabbit or Gorder strategies, which rely on particular data orderings. Disrupting this order can lead to more cache misses. On the other hand, the MASK strategy maintains the original order of the data. This can be beneficial when combined with strategies like Rabbit and Gorder that rely on a specific clustering order. In Fig. 5, we can see that ECG MASK often has lower cache miss rates than GRASP when combined with these strategies, which suggests that preserving the original order can lead to better performance in these cases. In conclusion, GRASP performs better when used alone, but there might be better choices when combined with ordering-dependent strategies like Rabbit and Gorder. On

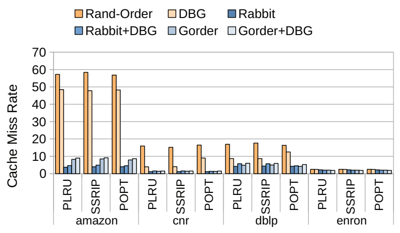
(a) 32KB cache with no ECG prefetching



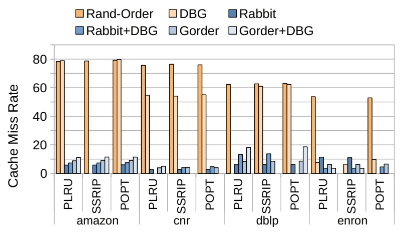
(b) 256KB cache with no ECG prefetching



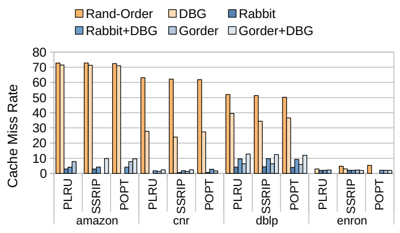
(c) 1MB cache with no ECG prefetching



(d) 32KB cache with ECG prefetching



(e) 256KB cache with ECG prefetching



(f) 1MB cache with ECG prefetching

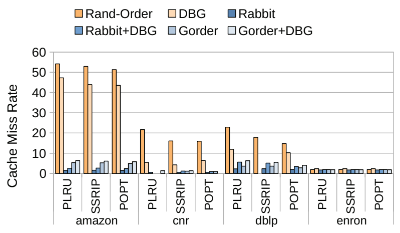
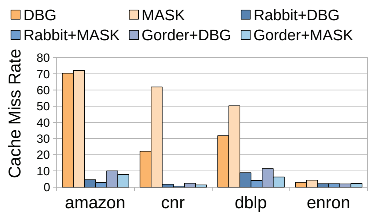
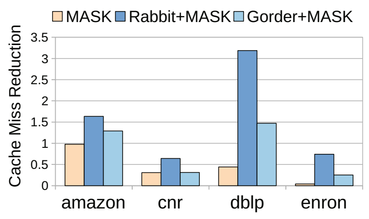


Fig. 4: Cache performance and miss rate. Results of simulating PageRank (PR) pull approach on a simple trace-driven cache simulator, combined with different reordering techniques



(a) 256KB cache GRASP vs. ECG



(b) 256KB cache GRASP with ECG

Fig. 5: Cache Policy miss rate against masking re-reference value into ECG bit and miss rate. Results of simulating PageRank (PR) pull approach on a simple trace-driven cache simulator, combined with different reordering techniques.

the other hand, ECG MASK might offer better performance in such cases due to its ability to maintain the original order of the data. Ultimately, the choice between GRASP and MASK should consider the specifics of the data, the importance of maintaining data order, and the cache miss costs associated with each strategy.

### *E. Processor Design and Architecture, Why ECG?*

**ECG & GRASP:** Encoding GRASP cache eviction hints (hot/cold) in ECG Mask via the edge list structures maintain the original order of the graph while allowing the GRASP policy to operate. **ECG & P-OPT:** Instead of preserving several ways in the cache for the re-reference matrix, ECG encodes them to the unused bits mask.

This work hypothesizes that while ECG combines multiple graph/eviction policies and prefetching techniques [15], [39], it achieves less overhead with the simple masking technique, thus maintaining a manageable architecture compared to fusing different graph optimization separately.

## V. Conclusion and Future Work

By proposing a pipeline-specialized graph addressing mode, we can extract the vertex ID from the graph edge list to request property data and use the masked ECG bits to determine the replacement policy for better caching and prefetching of the adjacent related hub vertex.

Moving forward, we plan to implement an ECG-based tie-breaker eviction policy where cache lines with equal P-OPT RRPV (re-reference prediction value) consult the GRASP part of ECG bits to keep the cache lines with hottest vertices from eviction. For a more comprehensive end-to-end evaluation, we will explore other clustering masks with ECG while evaluating the results on more graph datasets and algorithms using cycle-accurate simulation for valid performance measurement.

## VI. Acknowledgement

This work was supported in part by Semiconductor Research Corporation (SRC.)

## References

[1] J. A. Ang, B. W. Barrett, K. Wheeler, and R. C. Murphy, "Introducing the Graph 500," 2010.

[2] J. Lee, H. Kim, S. Yoo, K. Choi, H. P. Hofstee, G.-J. Nam, M. R. Nutter, and D. Jamsek, "ExtraV: boosting graph processing near storage with a coherent accelerator," Aug. 2017. [Online]. Available: https://doi.org/10.14778/3137765.3137776

[3] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," *arXiv:1812.08434 [cs, stat]*, Jul. 2019, arXiv: 1812.08434. [Online]. Available: http://arxiv.org/abs/1812.08434

[4] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry, "Challenges in parallel graph processing," *Parallel Processing Letters*, vol. 17, no. 01, pp. 5–20, Mar. 2007, publisher: World Scientific Publishing Co. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0129626407002843

[5] S. Beamer, K. Asanovic, and D. Patterson, "Locality Exists in Graph Processing: Workload Characterization on an Ivy Bridge Server," in *2015 IEEE International Symposium on Workload Characterization*, Oct. 2015, pp. 56–65.

[6] S. Beamer, K. Asanović, and D. Patterson, "Reducing Pagerank Communication via Propagation Blocking," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2017, pp. 820–831, iSSN: 1530-2075.

[7] H. Wei, J. X. Yu, C. Lu, and X. Lin, "Speedup Graph Processing by Graph Ordering," New York, NY, USA, Jun. 2016, pp. 1813–1828. [Online]. Available: https://doi.org/10.1145/2882903.2915220

[8] M. Baradaran, A. Ansari, M. Sadrosadati, and H. Sarbazi-Azad, "Energy Consumption Analysis of Instruction Cache Prefetching Methods," in *2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, Oct. 2023, pp. 60–67. [Online]. Available: https://ieeexplore.ieee.org/document/10306038

[9] A. Ansari, F. Golshan, P. Lotfi-Kamran, and H. Sarbazi-Azad, "MANA: Microarchitecting an Instruction Prefetcher," Feb. 2021, arXiv:2102.01764 [cs]. [Online]. Available: http://arxiv.org/abs/2102.01764

[10] M. Ferdman, C. Kaynak, and B. Falsafi, "Proactive instruction fetch," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: Association for Computing Machinery, Dec. 2011, pp. 152–162. [Online]. Available: https://dl.acm.org/doi/10.1145/2155620.2155638

[11] A. Kolli, A. Saidi, and T. F. Wenisch, "RDIP: return-address-stack directed instruction prefetching," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-46. New York, NY, USA: Association for Computing Machinery, Dec. 2013, pp. 260–271. [Online]. Available: https://dl.acm.org/doi/10.1145/2540708.2540731

[12] V. Balaji, N. Crago, A. Jaleel, and B. Lucia, "P-OPT: Practical Optimal Cache Replacement for Graph Analytics," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021, pp. 668–681, iSSN: 2378-203X.

[13] P. Faldu, J. Diamond, and B. Grot, "Domain-specialized cache management for graph analytics," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 234–248.

[14] ——, "A Closer Look at Lightweight Graph Reordering," *arXiv:2001.08448 [cs]*, Jan. 2020.

[15] A. Manocha, J. L. Aragón, and M. Martonosi, "Graphfire: Synergizing Fetch, Insertion, and Replacement Policies for Graph Analytics," *IEEE Transactions on Computers*, vol. 72, no. 1, pp. 291–304, Jan. 2023, conference Name: IEEE Transactions on Computers.

[16] A.-L. Barabási, "Scale-Free Networks: A Decade and Beyond," *Science*, vol. 325, no. 5939, pp. 412–413, Jul. 2009, publisher: American Association for the Advancement of Science Section: Perspective. [Online]. Available: https://science.sciencemag.org/content/325/5939/412

[17] Y. Zhang, V. Kiriansky, C. Mendis, S. Amarasinghe, and M. Zaharia, "Making caches work for graph analytics," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec. 2017, pp. 293–302.

[18] V. Balaji and B. Lucia, "When is Graph Reordering an Optimization? Studying the Effect of Lightweight Graph Reordering Across Applications and Input Graphs," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, Sep. 2018, pp. 203–214.

[19] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 251–262, Aug. 1999. [Online]. Available: https://doi.org/10.1145/316194.316229

[20] S. Beamer, K. Asanović, and D. Patterson, "The GAP Benchmark Suite," May 2017, arXiv:1508.03619 [cs]. [Online]. Available: http://arxiv.org/abs/1508.03619

[21] Y. Zhang, V. Kiriansky, C. Mendis, M. Zaharia, and S. Amarasinghe, "Optimizing cache performance for graph analytics," *arXiv preprint arXiv:1608.01362*, 2016, publisher: CoRR.

[22] J. Petit, "Experiments on the minimum linear arrangement problem," *ACM Journal of Experimental Algorithmics*, vol. 8, p. 2.3, Dec. 2004. [Online]. Available: https://doi.org/10.1145/996546.996554

[23] P. Boldi and S. Vigna, "The WebGraph Framework I: Compression Techniques." ACM Press, 2003, pp. 595–601.

[24] R. Barik, M. Minutoli, M. Halappanavar, N. R. Tallent, and A. Kalyanaraman, "Vertex Reordering for Real-World Graphs and Applications: An Empirical Evaluation," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2020, pp. 240–251.

[25] M. Frasca, K. Madduri, and P. Raghavan, "NUMA-aware graph mining techniques for performance and energy efficiency," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Washington, DC, USA: IEEE Computer Society Press, Nov. 2012, pp. 1–11.

[26] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998, publisher: SIAM.

[27] K. I. Karantasis, A. Lenharth, D. Nguyen, M. J. Garzarán, and K. Pingali, "Parallelization of Reordering Algorithms for Bandwidth and Wavefront Reduction: International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2014," *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, vol. 2015-January, no. January, pp. 921–932, Jan. 2014. [Online]. Available: http://www.scopus.com/inward/record.url?scp=84976618589&partnerID=8YFLogxK

[28] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 1969 24th national conference*, ser. ACM '69. New York, NY, USA: Association for Computing Machinery, Aug. 1969, pp. 157–172. [Online]. Available: https://doi.org/10.1145/800195.805928

[29] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 1997, pp. 21–29.

[30] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan, "On compressing social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 219–228.

[31] D. LaSalle and G. Karypis, "Multi-threaded Graph Partitioning," *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, 2013.

[32] A. George, "Nested Dissection of a Regular Finite Element Mesh," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 345–363, Apr. 1973, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/0710032

[33] J. Arai, H. Shiokawa, T. Yamamuro, M. Onizuka, and S. Iwamura, "Rabbit Order: Just-in-Time Parallel Reordering for Fast Graph Analysis," May 2016, pp. 22–31.

[34] J. Willcock and A. Lumsdaine, "Accelerating sparse matrix computations via data compression," in *Proceedings of the 20th annual international conference on Supercomputing*, ser. ICS '06. New York, NY, USA: Association for Computing Machinery, Jun. 2006, pp. 307–316. [Online]. Available: https://doi.org/10.1145/1183401.1183444

[35] A. T. Mughrabi, M. Ibrahim, and G. T. Byrd, "QPR: Quantizing PageRank with Coherent Shared Memory Accelerators," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2021, pp. 962–972, iSSN: 1530-2075.

[36] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web." Nov. 1999, library Catalog: ilpubs.stanford.edu:8090 Publisher: Stanford InfoLab. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[37] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," in *Proceedings of the seventh international conference on World Wide Web 7*, ser. WWW7. NLD: Elsevier Science Publishers B. V., Apr. 1998, pp. 107–117.

[38] M. Bianchini, M. Gori, and F. Scarselli, “Inside PageRank,” Feb. 2005. [Online]. Available: https://doi.org/10.1145/1052934.1052938

[39] A. Basak, S. Li, X. Hu, S. M. Oh, X. Xie, L. Zhao, X. Jiang, and Y. Xie, “Analysis and Optimization of the Memory Hierarchy for Graph Processing Workloads,” in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2019, pp. 373–386, iSSN: 2378-203X.