Proceedings of the 4th International Conference on Applied Physics and Mathematical Modeling
DOI: 10.54254/2753-8818/2026.37501

Sparse Matrix Representations for Large-Scale Web Link Analysis

Siyuan Huang

Woodbridge High School, Irvine, USA
DavidHuangSiyuan@outlook.com

Abstract. The World Wide Web can be viewed as a very large directed graph. Web pages are nodes, and hyperlinks are edges. If all possible links are stored in a dense adjacency matrix, most entries will be zero. This form will be relatively memory-intensive and computationally expensive. Sparse matrix methods only store the non-zero links. They can store web graphs efficiently and provide the required functions of link analysis and ranking. This paper presents the general ideas of web graphs and sparse matrices. It introduces some general sparse formats, such as Coordinate Format (COO), Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC). It also compares the storage, matrix-vector multiplication and update performance of these formats. A few studies in recent years have been conducted. The paper presents the main strengths and deficiencies of the current methods. Finally, some possible future directions are introduced. This review explains how linear algebra tools are used to support large-scale web link analysis.

Keywords: sparse matrix, web graph, web link analysis, PageRank

1. Introduction

There are billions of Web pages in the world. These pages are linked by hyperlinks. The World Wide Web can be represented as a directed graph. Each Web page is a node. Each hyperlink is a directed edge connecting one page to another. The above is called a Web graph. Web graphs are very large and continue to grow every day. Many important applications require the analysis of these graphs. Web page ranking, finding communities of related pages, and building recommendation systems are examples. Link analysis is a typical type of search engine technology mentioned in [1, 2].

A graph can be expressed as a matrix in a certain way. The adjacency matrix has a value of 1 if there is a link from a source page to a destination page, and 0 otherwise. For a selected edge direction, the outgoing-link weights are normalised to create a transition matrix, so each non-dangling row defines a probability distribution. This transition matrix is used in ranking algorithms such as PageRank. A web graph with millions or billions of pages would have a very large dense matrix in terms of memory. Almost all entries in the matrix are zero because most pages only link to a few other pages. This situation is called sparsity. Sparse matrix representation only stores the non-zero values and their indices. Matching the storage format to the access pattern can reduce memory consumption and improve the speed of sparse operations, such as matrix-vector multiplication [3].

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/2026.37501

Over the past twenty years, researchers have developed many ways to store sparse matrices. Early methods used simple lists of non-zero entries. Later, Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC) formats were also added. These formats are space-efficient and support fast matrix-vector multiplication. Blocked formats and special graph compression techniques have also been studied for web-scale data. A spy plot can be used to show the pattern of non-zero entries in a matrix. However, most of the research to date has focused on storage formats and high-efficiency computing rather than on visual plots [4, 5].

This paper reviews sparse matrix representations for large-scale web link analysis. The aim is to introduce the general ideas, typical sparse formats, recent research results and outstanding problems. The paper does not attempt to build a new crawler or a full-fledged search engine. It only collects and organises existing knowledge. The paper is organised as follows. Section 2 provides the general ideas of Web graphs and sparse matrices. Section 3 shows some common formats of sparse matrices and their computational characteristics. Section 4 reports on the new studies in the past three years. Section 5 discusses the limitations of current methods and future directions. Section 6 is the Conclusion.

2. Basic concepts of web graphs and sparse matrix representation

A Web graph is a directed graph of nodes and edges. Each node is a web page, and each directed edge is a hyperlink from one page to another. The number of web pages is the number of nodes, and the number of hyperlinks is the number of directed edges. The out-degree of a node is the number of links leaving that page. The in-degree is the number of links that point to that page. Most pages in a real web graph have a small out-degree, and only a few pages have a very high degree. The degree distribution is generally heavy-tailed and can be approximated by a simple power-law model [6].

The adjacency matrix of a web graph is a grid, and both its rows and columns are web pages. An entry in this grid is 1 if there is a hyperlink from the row page to the column page, and otherwise it is 0. Since the total number of edges is much smaller than the square of the total page count, the matrix is sparse. Many large web graphs are extremely sparse, but the exact sparsity depends on graph size and collection method. For the row corresponding to a page with one or more outgoing links, a transition matrix is obtained by normalising the outgoing-link weights. After normalisation, the sum of each row is 1. This matrix is used in a random-walk model. If a page has no out-links, it is called a dangling node. Dangling nodes must be handled separately, and generally, an empty transition row is replaced by a selected probability distribution [1, 7].

A dense matrix stores all the elements, including the zeros, explicitly. The required space is proportional to the square of the total number of pages. A sparse matrix stores only the non-zero entries. Common measures of sparsity compare the number of active links with the total grid size. Web graphs generally have a much smaller number of active link entries compared to the square of the total page count, so sparse storage is significantly more economical than dense storage. Spy plots can be used to show the locations of the non-zero entries. Spy plots can show block or clustered patterns in the non-zero structure, but their appearance is highly dependent on node ordering and graph organisation.

One should know the general ideas of all kinds of storage before learning about them. The matrix size, the number of non-zero elements, the degree distribution and the presence of dangling nodes all affect which sparse format is more suitable. The following sections will use the above ideas to present the strengths and weaknesses of each method.

DOI: 10.54254/2753-8818/2026.37501

3. Sparse matrix formats and computational methods for web graphs

Several typical ways of storing a sparse matrix are shown below. The first kind is a coordinate form, or COO. It maintains three arrays of the same size. The first array is the row index of a non-zero element. The second array contains the column indices. The third array is the actual data. COO is relatively simple to build and convenient to modify when adding a new link. COO generally has more index storage than compressed formats such as CSR or CSC because both the row and column indices of non-zero elements are stored. It is also not the fastest form for repeated matrix-vector multiplication [3, 8].

Compressed Sparse Row (CSR) is a typical representative of the above formats. CSR has three arrays. The values array contains all the non-zero elements in row-major order. The column indices array stores the columns of the non-zero elements. The row pointers array stores the starting index of each row in the values array. CSR is suitable for row-oriented access because the non-zero elements of each row are stored contiguously. Sparse matrix-vector multiplication (SpMV) is relatively fast with CSR when multiplying the vector from the right. Compressed Sparse Column (CSC) is the column-oriented form of CSR. CSC is more suitable for algorithms that need to access columns frequently [8, 9].

Blocked sparse formats attempt to group non-zeros into small dense blocks. They can enhance the cache performance of contemporary central processing units and graphics processing units. Graph compression methods, such as the WebGraph framework, aim to use the special structure of web graphs. Web pages with similar addresses often have similar link patterns. These methods can achieve a higher compression ratio than plain CSR. However, they generally require an additional decoding step for the data upon reading. This can slow down random access [4, 5].

The essential operations of a web graph include matrix-vector multiplication (the core of PageRank), calculating the degree of every node, finding the neighbours of a node, and updating the graph after adding or deleting links. CSR is generally fast for SpMV on static graphs but slow for updates. COO is convenient to update but slow for repeated multiplication of the same matrix. Hybrid forms aim to meet the above demands. Typical evaluation indicators are the size of the memory required for the stored matrix, the time taken by a single SpMV operation, compression ratio and scalability to larger graphs [9].

4. Recent studies of sparse matrix methods for large-scale web link analysis

In the last few years, studies of sparse representations for large graphs have focused on several related goals, especially reducing storage overhead and improving processing efficiency. Gan et al. introduced GraphCSR in 2025, for example. The method groups vertices that have the same degree and only stores the starting index of each group. Experiments on large web and social network graphs showed good memory savings and faster processing in a distributed computing environment [10].

Beyond storage reduction, another line of work focuses on computational efficiency, especially hardware acceleration for SpMV in PageRank. A 2023 study designed a multi-mode SpMV accelerator that supports both half-precision and single-precision floating-point numbers. The design uses the CSR format. Tests on four real-world graphs showed speedups of 1.3 to 1.9 times and maintained a low numerical error under the tested conditions [11]. These results are consistent with the broader analysis of Abdelaal and Veras, who found that the structure of sparse data significantly affects performance [12].

DOI: 10.54254/2753-8818/2026.37501

Compression-oriented methods reveal a related set of trade-offs. In 2025, three high-performance methods for large-scale web graph PageRank SpMV operations were compared: grammar-based compression, k2-trees, and Zuckerli. The results showed trade-offs among space, running time and energy consumption, with the preferred choice also depending on the number of available threads [13]. Meanwhile, dynamic formats, such as LSMGraph, combine the write-friendly characteristics of log-structured merge trees with multi-level CSR structures. This design supports fast updates and good read performance at the same time [14].

These reported gains should also be interpreted in light of differences in datasets and evaluation environments. Many studies use public web graph datasets provided by the Laboratory for Web Algorithmics or other projects, ranging from a few million edges to several billion. The mean out-degree is generally small, while the distribution of non-zero entries is very uneven. Node ordering, including the reverse ordering of web addresses in some web-graph compressors, can substantially affect compression efficiency. Different papers also use different machines and timing methods. Therefore, their absolute runtime results should not be directly compared [4, 5].

Taken together, current results show that the relative advantage of a sparse representation depends on the workload being optimized. CSR remains a relatively simple choice for static SpMV workloads, compression-oriented methods emphasize storage efficiency but can introduce decoding or access costs, and dynamic formats improve update support while retaining good read performance. Thus, no single format is suitable for all tasks and graphs. The preferred representation depends on graph size, degree distribution, update frequency, and target hardware.

5. Limitations and prospects

The trade-offs summarized above also reveal several limitations of current methods. First, real web graphs are huge and keep changing. New pages and new links are added daily. Static CSR is relatively costly to update because adding or deleting edges may require shifting or reallocating stored entries at a large scale. Rebuilding a matrix with billions of edges is therefore costly in both time and memory. Second, the degree distribution of web graphs is highly skewed. A few nodes have millions of connections, whereas most nodes have only a few. Therefore, a single fixed format may not be suitable for graph regions with significantly different degree distributions [6, 12].

Third, research groups have used different datasets, node orderings, machines and measurement methods, so reported results cannot always be compared directly. At the method level, high compression ratios can also lead to slower random access because of extra decoding overhead. Fourth, most studies either focus on static graphs or consider only one type of operation, such as SpMV. As a result, support for mixed workloads that combine reading, writing and several different algorithms remains limited.

These limitations, together with the reported results above, suggest several directions for future research. Because static CSR is costly to update and LSMGraph shows that dynamic structures can combine fast updates with good read performance, dynamic sparse formats should further improve insertion and deletion efficiency [14]. The highly skewed degree distribution of web graphs also motivates hybrid storage that uses different formats for high-degree and low-degree nodes [6, 12]. For real web-scale workloads, results from GraphCSR and hardware-accelerated SpMV indicate that parallel, distributed, and hardware-aware designs are promising for improving processing efficiency [10, 11]. Since node ordering can substantially affect compression efficiency, improved reordering techniques also deserve further study [4, 5]. Finally, because existing studies use different datasets, node orderings, hardware, and measurement methods, a general-purpose public benchmark with

DOI: 10.54254/2753-8818/2026.37501

standard datasets and evaluation scripts would support more reliable comparison and reproducibility [12].

6. Conclusion

This paper reviews sparse matrix representations for large-scale web link analysis. Web graphs are large directed graphs with very high sparsity, so dense storage is generally impractical at web scale. COO, CSR and CSC are common sparse formats, but they have different strengths for storage, access, multiplication and update workloads. COO is convenient for adding or modifying links, and compressed formats are more suitable for repeated row- or column-oriented operations. CSR is still a relatively common choice for static sparse matrix-vector multiplication, and graph-compression methods can further reduce storage by exploiting structural regularity in web graphs. Dynamic structures extend the above ideas by supporting updates without rebuilding the whole representation. The above studies have shown that storage efficiency cannot be assessed in isolation from computational cost. Node ordering, degree distribution, graph size, hardware, thread count, and the ratio of sequential to random access can all affect the measured performance. Therefore, the runtime or compression results of different studies should not be directly compared without considering their experimental environments. For dynamic web graphs, update cost can be as important as the throughput of repeated multiplications, and mixed workloads are even more difficult to optimise than single-operation static workloads. Therefore, future research will continue to develop dynamic and hybrid sparse formats, hardware-aware implementations, scalable parallel and distributed methods, and common public benchmarks with consistent evaluation procedures. Sparse matrix representation is a general-purpose basis for web link analysis, but the most suitable type of representation depends on both the structure of the graph and the computational demands of the system.

References

[1] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems, 30*(1–7), 107–117. https://doi.org/10.1016/S0169-7552(98)00110-X

[2] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., & Wiener, J. (2000). Graph structure in the web. *Computer Networks, 33*(1–6), 309–320. https://doi.org/10.1016/S1389-1286(00)00083-9

[3] Gao, J., Ji, W., Chang, F., Han, S., Wei, B., Liu, Z., & Wang, Y. (2023). A systematic survey of general sparse matrix-matrix multiplication. *ACM Computing Surveys, 55*(12), Article 244, 1–36. https://doi.org/10.1145/3571157

[4] Francisco, A. P., Gagie, T., Köppl, D., Ladra, S., & Navarro, G. (2022). Graph compression for adjacency-matrix multiplication. *SN Computer Science, 3*, Article 193. https://doi.org/10.1007/s42979-022-01084-2

[5] Boldi, P., & Vigna, S. (2004). The WebGraph framework I: Compression techniques. In *Proceedings of the 13th International Conference on World Wide Web* (pp. 595–602). ACM. https://doi.org/10.1145/988672.988752

[6] Donato, D., Laura, L., Leonardi, S., & Millozzi, S. (2007). The web as a graph: How far we are. *ACM Transactions on Internet Technology, 7*(1), Article 4, 4–34. https://doi.org/10.1145/1189740.1189744

[7] Ipsen, I. C. F., & Selee, T. M. (2007). PageRank computation, with special attention to dangling nodes. *SIAM Journal on Matrix Analysis and Applications, 29*(4), 1281–1296. https://doi.org/10.1137/060664331

[8] Isaac–Chassande, V., Evans, A., Durand, Y., & Rousseau, F. (2024). Dedicated hardware accelerators for processing of sparse matrices and vectors: A survey. *ACM Transactions on Architecture and Code Optimization, 21*(2), Article 27, 1–26. https://doi.org/10.1145/3640542

[9] Filippone, S., Cardellini, V., Barbieri, D., & Fanfarillo, A. (2017). Sparse matrix-vector multiplication on GPGPUs. *ACM Transactions on Mathematical Software, 43*(4), Article 30, 1–49. https://doi.org/10.1145/3017994

[10] Gan, X., Li, T., Zhang, Q., Wu, G., Yang, B., Gong, C., Liu, J., & Lu, K. (2025). GraphCSR: A space and time-efficient sparse matrix representation for web-scale graph processing. In *Proceedings of the ACM on Web Conference 2025* (pp. 763–771). ACM. https://doi.org/10.1145/3696410.3714833

DOI: 10.54254/2753-8818/2026.37501

[11] Kim, W., Kim, H., Lee, J., Kim, H., & Kim, J. H. (2023). Multi-mode SpMV accelerator for transprecision PageRank with real-world graphs. *IEEE Access, 11*, 6261–6272. https://doi.org/10.1109/ACCESS.2023.3237079

[12] Abdelaal, K., & Veras, R. (2024). A framework for analyzing the performance of sparse matrix and graph operations. In *2024 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1–7). IEEE. https://doi.org/10.1109/HPEC62836.2024.10938464

[13] Tosoni, F., Bille, P., Brunacci, V., De Angelis, A., Ferragina, P., & Manzini, G. (2025). Toward greener matrix operations by lossless compressed formats. *IEEE Access, 13*, 56756–56779. https://doi.org/10.1109/ACCESS.2025.3555119

[14] Yu, S., Gong, S., Tao, Q., Shen, S., Zhang, Y., Yu, W., Liu, P., Zhang, Z., Li, H., Luo, X., Yu, G., & Zhou, J. (2024). LSMGraph: A high-performance dynamic graph storage system with multi-level CSR. *Proceedings of the ACM on Management of Data, 2*(6), Article 243, 243: 1–243: 28. https://doi.org/10.1145/3698818