1 Introduction

For a sparse linear system **Ax=B** where **A** is the coefficient matrix, **x** is the vector of unknowns and **B** the vector of constants, matrix factorization methods, like LU [14], QR [4], Cholesky [19], will decompose **A** into a lower triangular **L** matrix and an upper triangular **U** matrix. During the decomposition, new nonzeros called **fill-ins** are added in the decomposed matrix. Since **L** and **U** often contain more nonzeros, it is desirable to minimize the new fill-ins.

This proposal attempts to reduce the fill-ins from two aspects. First, our algorithm innovation will both reduce the matrix bandwidth, as well as try to assign smaller row indices to the rows of smaller degrees (so that smaller degree rows are eliminated first in matrix factorization). Particularly, *minimum degree* approach prioritizes the elimination of the vertices with the smaller degree, while both *graph traversal* and *graph partitioning* based algorithms aim to reduce the bandwidth of the matrix. As we will show in Figure 1, all these designs are effective in pursuit of reducing fill-ins. Whereas, very few prior arts have exploited these methods collaboratively to further reduce the fill-ins. Our proposed approach will depart from nested dissection, and further integrate the minimum degree concept as follows: after each step of nested dissection, we propose to further compact the separator vertices rows as well as the denser diagonal region for fill-in reduction. Adding this heuristic, which is analogous to minimum degree, to nested dissection, we can achieve better fill-in reduction ordering. As a preliminary result, **Figure 1 shows that the number of fill-ins is reduced to two in our new ordering, whereas the state of the art approaches introduce four or more fill-ins.**

Second, our system optimization will deploy growing graph component of ParMETIS and Fieldler vector extraction component of spectral partition on GPUs. The nested partitioning based tools like ParMETIS [11] use the growing graph method for partitioning. The approach of partition detection involves Breadth-First Search (BFS) traversal, which can be implemented in GPUs. The most expensive phase for getting the Fiedler vector in spectral partitioning is also well suitable for GPUs given it mainly involves the matrix-vector operations. We will design and implement this most expensive phase of spectral ordering algorithm for fill-in reduction in a distributed GPU based system. The rest of this paper is organized as follows: Section 2 provides the background and the related works to the problem and Section 3 provides the proposed research.

2 Background and Related Work

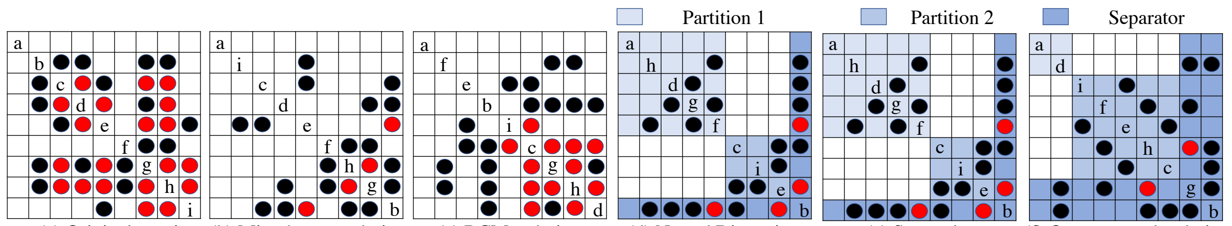
Vertex ranking based strategy. Minimum degree [1] reorders the matrix such that the elimination of the vertex with the minimum degree is done earlier. Minimum degree is a greedy approach where during each step of elimination, a vertex with the minimum degree is chosen to be eliminated first. The minimum degree method places the rows/columns with the most of non-zeros towards the end of elimination.

Graph traversal based strategy. Cuthill-Mckee (CM) and Reverse CM (RCM) [2, 16] methods reorder the vertices based upon the BFS traversal order of the graph. This approach achieves fill reduction by reducing the *bandwidth* (a function of farthest non-zeros of rows from diagonal) of the matrix. The CM method involves reducing the bandwidth of the matrix by starting a BFS from a vertex in the graph, and assigning a new index to the vertices based on the level on which the vertex is encountered during the traversal. The vertices within the same level in the traversal can be prioritized for the indices based on the degree of the vertex by selecting the vertex with the smallest degree. RCM orderings are obtained by reversing the orderings obtained from CM. The goal of the CM/RCM ordering is to bring the non-zeros towards the diagonal and make the diagonal band as thin and dense as possible.

Partitioning based strategy. Similar to graph traversal based strategy, partitioning based ones also try keeping the non-zeros close to the diagonal (i.e., reducing the bandwidth of the matrix). For that, the strategy follows two steps. First, it partitions the (sub-)matrix into two approximately equal parts via a separator. Second, it reduces the number of vertices in the separator. Both steps aim to keep the non-zeros close to the diagonal. Nested dissection [6, 18, 12] from ParMETIS [11] is a representative example of this genre. The method reorders the matrix such that a set of separator vertices **S**, which separates two partitions, are moved towards the bottom of the matrix. The set of separator vertices are minimized so that the number of non-zeros remains close to the diagonal. We discuss two relevant methods used for the partitioning of a graph. First, *spectral partitioning* [22, 24] partitions the matrix based on the spectrum i.e. eigenvalues and eigenvectors of the Laplacian matrix **L** [17]. This method involves the expensive computation of the eigenvectors of **L**. The Fiedler pair (λ_2, X_2) of the Laplacian **L** is computed so that the selection of vertices for partition is done based on the sign of the components of Fiedler vector X_2. Optimization like spectral regularization [23, 9] can be done to get better partitioning results. Second, *growing graph partitioning* (GGP) [3, 10] partitions the graph by running multi-source BFS [20, 15] to get the separator vertices such that the traversals have the coverage of similar vertex range. ParMETIS uses a greedy approach of GGP for fill-in reduction ordering [10].

3 Proposed Research

Our algorithmic optimization will populate the diagonal region of the matrix with non-zeros (RCM and partitioning based method) as well as pushing the non-zeros towards the bottom of the matrix (heuristic used by minimum degree ordering). As a system optimization, we will use GPUs for the detection of partitions. Spectral based partitioning and growing graph partitioning approach have good potential for GPU implementation. Our proposed tasks are two folds.



(a) Original matrix (#fill-ins = 16) (b) Min. degree ordering (#fill-ins = 4) (c) RCM ordering (#fill-ins = 12) (d) Nested Dissection ParMETIS (#fill-ins = 4) (e) Spectral (#fill-ins = 4) (f) Our proposed ordering (#fill-ins = 2)

Figure 1: Comparison of different fill-in reducing ordering methods.

Task 1. ***Algorithmic optimization*** *will use nested dissection and minimum degree together to reduce fill-ins.*

Rule #1. We should migrate a vertex from one partition to the other such that non-zeros of separator vertices are closer to diagonal without negatively impacting the bandwidth of the matrix.

Rule #2. We allow minimum degree ordering method to run across the partitions in partition-based strategy.

Figure 1 presents the comparison of different fill-in reduction orderings for a toy matrix in Figure 1(a). Figures 1(b), (c), (d), (e) and (f) are, respectively, the filled matrix (red dots as fill-ins) obtained after the factorization of the reordered matrix obtained via minimum degree ordering, RCM ordering, nested dissection (ParMETIS), spectral partitioning method of reordering with Kernighnan-Lin (KL) refinement [21, 13], and proposed new ordering approach. The original matrix without any reordering optimizations ends with a fill-in count of 16. The minimum degree ordering is done as a pre-processing to allow for parallel implementation. The fill-ins are reduced to four with this method. The RCM ordering, though performs better than the case without any reordering methods, has poor results in comparison to other strategies. The nested dissection, from ParMETIS, has the same resultant ordering as that of spectral approach, in part because both of them use the KL refinement. **Our proposed reordering** applies multiple reordering methods according to Rules #1 and #2. Comparing Figures 1(d) and 1(f), Rule #1 and Rule #2 allow the vertex *g* to be moved into the separator set of vertices. Likewise shifting of the vertices *h* from partition 1 to 2 and *c* within the partition 2 makes the separator vertex *b* closer to the diagonal. We can also observe that in spite of going against the rules of attempting for a smaller separator set, as well as making the partitions imbalanced, can lead to fewer fill-ins, as long as our rules are obeyed. In the coming summer research program, we will investigate and implement more of such rules with Dr. Xiaoye S. Li to i) minimize bandwidth, and ii) prioritize minimum degree rows for fill-in reduction.

Task 2. ***Systemic optimization*** *will deploy growing graph component of ParMETIS and Fieldler vector extraction component of spectral partition on GPUs.*

The graph partitioning based methods of reordering for the fill-in reduction have been shown to have good fill-in reduction during the factorization phase in the sparse linear solver. The ParMETIS [10] uses multi-source BFS traversal in the Greedy Growing Graph Partitioning (GGGP) algorithm to detect the separator. GPUs are known to have state of the art implementation of BFS traversal [7]. However, the tool has not yet explored this portion in fill-in reduction ordering for GPU acceleration even though it carries a promising potential for improvement with GPU acceleration. Spectral methods of partitioning and fill-in reduction ordering can provide good quality ordering but are kept away from practical uses because of high computational cost. The most computationally expensive phase of spectral partitioning is the eigenvalue and eigenvector computation [5, 8] of graph Laplacian. The main kernels of eigensolver involve SpMV and dense matrix/vector operations. Such a kind of operation is well suited for the tensor core on GPUs. It is also important to note that as the rules in algorithmic optimizations may lead the partitions to become more unequal size depending upon the distance of non-zeros from the diagonal, they are likely to induce workload imbalance during the process of nested dissection. The challenge can be addressed by merge-path based workload dissection.

References can be found at **`http://personal.stevens.edu/~hliu77/docs/lbl_20.pdf`**

References

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] A. Azad, M. Jacquelin, A. Buluç, and E. G. Ng. The reverse cuthill-mckee algorithm in distributed-memory. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 22–31. IEEE, 2017.

[3] C.-E. Bichot and P. Siarry. *Graph partitioning*. Wiley Online Library, 2011.

[4] H. Bouwmeester, M. Jacquelin, J. Langou, and Y. Robert. Tiled qr factorization algorithms. In *SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2011.

[5] R. L. Dailey. Eigenvector derivatives with repeated eigenvalues. *AIAA journal*, 27(4):486–491, 1989.

[6] T. A. Davis. *Direct methods for sparse linear systems*, volume 2. Siam, 2006.

[7] A. Gaihre, Z. Wu, F. Yao, and H. Liu. Xbfs: exploring runtime optimizations for breadth-first search on gpus. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 121–131. HPDC, 2019.

[8] R. Ge, C. Jin, P. Netrapalli, A. Sidford, et al. Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis. In *International Conference on Machine Learning*, pages 2741–2750, 2016.

[9] A. Joseph, B. Yu, et al. Impact of regularization on spectral clustering. *The Annals of Statistics*, 44(4):1765–1791, 2016.

[10] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *ICPP (3)*, pages 113–122, 1995.

[11] G. Karypis and V. Kumar. Parmetis-parallel graph partitioning and fill-reducing matrix ordering. *URL http://glaros. dtc. umn. edu/gkhome/metis/parmetis/overview*, 2013.

[12] G. Karypis, K. Schloegel, and V. Kumar. Parmetis. *Parallel graph partitioning and sparse matrix ordering library. Version*, 2, 2003.

[13] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[14] X. S. Li and J. Demmel. A scalable sparse direct solver using static pivoting. In *PPSC*, 1999.

[15] H. Liu, H. H. Huang, and Y. Hu. ibfs: Concurrent breadth-first search on gpus. In *Proceedings of the 2016 International Conference on Management of Data*, pages 403–416. ACM, SIGMOD, 2016.

[16] W.-H. Liu and A. H. Sherman. Comparative analysis of the cuthill–mckee and the reverse cuthill–mckee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[17] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991.

[18] S. Rajamanickam and E. G. Boman. An evaluation of the zoltan parallel graph and hypergraph partitioners. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2011.

[19] R. B. Schnabel and E. Eskow. A new modified cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, 11(6):1136–1158, 1990.

[20] M. Then, M. Kaufmann, F. Chirigati, T.-A. Hoang-Vu, K. Pham, A. Kemper, T. Neumann, and H. T. Vo. The more the merrier: Efficient multi-source graph traversal. *Proceedings of the VLDB Endowment*, 8(4):449–460, 2014.

[21] J. L. Träff. Direct graph k-partitioning with a kernighan–lin like heuristic. *Operations Research Letters*, 34(6):621–629, 2006.

[22] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[23] Y. Zhang and K. Rohe. Understanding regularized spectral clustering via graph conductance. In *Advances in Neural Information Processing Systems*, pages 10631–10640, 2018.

[24] Z. Zhao, Y. Wang, and Z. Feng. Nearly-linear time spectral graph reduction for scalable graph partitioning and data visualization. *arXiv preprint arXiv:1812.08942*, 2018.