*Vertex (re)ordering* is a technique which has been used to accelerate graph applications by targeting the primary bottleneck in such applications, namely, data movement which stems from poor data locality. Reordering is nothing but a permutation of the original vertex ordering which aims to preserve locality-based properties of the input graph.

**Preliminary Background:**

Let be an input graph, with being the set of vertices and being the set of edges. The vertices are indexed continuously starting from till .

A *vertex ordering* of is a logical rearrangement of the vertex indices (bijection or permutation) of onto a linear order. In other words, . Since the input graph comes with already labeled vertices, the input ordering is treated as the *natural ordering* of . Logically, any other vertex (re)ordering can be thought of as a permutation of the natural order of the input graph. The term *rank* of vertex refers to , which is the mapping of a vertex .

Efficacy of an ordering: The ultimate objective of reordering is to preserve some sort of locality – proximity of a pair of vertices in the graph space should be reflected by the relative closeness of their ranks in the reordering . The *average linear arrangement gap* (or *average gap profile*) is a metric defined in {insert citation 1} to quantifying the efficacy of a reordering scheme in preserving locality. Given an ordering of , the *average linear arrangement gap* is defined as the absolute difference of ranks of all pairs of adjacent vertices averaged over all edges. In other words, for two vertices , connected by an edge, the gap in between vertices *i* and *j* is given by: .

The *average linear arrangement gap* is then defined as:

Vertex reordering schemes: Over the years, different reordering strategies have been used in practice that optimize for varying objectives. What follows is a review and classification of the same:

1. *Gap-based Strategies:* The **Minimum Linear Arrangement (MinLA)** problem, defined in {insert citation 10}, attempts to come up with a reordering that directly optimizes for the average linear arrangement gap. Multiple heuristics and/or approximations like simulated annealing {insert Simulated Annealing citation} have been used to solve this NP-Hard {insert Illya Safro’s citation} problem.
2. *Degree-* *and Hub-based Strategies:* Degree has been used as a computationally cheap heuristic to come up with vertex orderings. These schemes can be as simple as **Degree Sort**, where the vertices are labeled based on the sorted order of decreasing degree, or slightly more sophisticated versions like **Hub Sort** or **Hub Clustering** {insert citation 51, 52, 53} that prefers to focus more on “hub” vertices that have disproportionately large degrees.
3. *Window-based Strategies:* A prime example of such a strategy is **Gorder** {insert citation 12}, which is the current state-of-the-art w.r.t. minimizing cache misses. The methodology involves sliding a window of parameterized size over the vertex array in the input order and maximizing a score defined over every pair of vertices inside the window by looking at the number of shared neighbors and siblings.
4. *Partitioning-based Strategies:*  These kinds of schemes involve partitioning the input graph into pre-determined (e.g., using graph partitioner **METIS** {insert citation 13}) or an algorithmically determined (e.g., using community detection tool like **Grappolo** {insert citation 14}) number of partitions and using the partitions created to linearly order the vertices. Such schemes have been empirically shown {insert citation 1} to be better suited to optimize for average linear arrangement gap compared to schemes from other classes.
5. *Fill-reducing Strategies:* These techniques can be loosely defined as those that try to pack nonzeros close together in the sparse matrix representation of an input graph. **Reverse Cuthill-McKee (RCM)** is a representative example of such a scheme that works by doing an interleaved breadth-first search (BFS) and depth-first search (DFS)traversal of the graph. It starts at the vertex with the smallest degree and subsequently visits all unvisited neighbors in increasing order of their degrees. The traversal order of the vertices is then used to generate the ordering.