Data movement has been the primary bottleneck for nearly all graph-theoretic applications. Irregular memory accesses that characterize most graph algorithms make preserving locality difficult. This is compounded by the scale-free characteristics of real-world graphs which generally follow the power-law degree distribution. The intuition behind vertex (re)ordering to improve locality is as follows: given an input (or *natural*) order of vertices, compute a permutation of the order such that proximity between pairs of vertices in the graph-space is reflected in the proximity of their labels or “ranks”. This increases the probability of finding the data contained in the neighbors of a vertex in the same cache line and hence leads to lower data movement thereby improving performance. To improve locality in computation, several vertex (re)ordering schemes have been proposed. However, a detailed characterization (both empirical and analytical) of various reordering schemes or their impacts on end applications has been largely missing in the literature.

Problem: In the study (published at the IISWC’20 conference) that this poster is based on, we present an extensive empirical evaluation of up to 11 reordering schemes, taken from different classes of approaches, on a set of 34 real-world graphs emerging from different application domains. Our study is presented in two parts: a) a thorough comparative evaluation of the different schemes on their effectiveness to optimize a set of linear arrangement gap measures, relevant to preserving locality; and b) an extensive evaluation of the impact of the ordering schemes on two real-world, parallel graph applications, namely, community detection and influence maximization.

Performance based on gap measures: Our studies show a significant divergence among the ordering schemes (up to 40x between the best and the poor) in their effectiveness to reduce the gap measures. Partition-based techniques like METIS, Grappolo, and Rabbit-Order outperform others in minimizing the average gap profile, with Reverse Cuthill-McKee showing competitive performance as well. In fact, we can observe four distinct performance tiers of schemes. The top-performing group is constituted by METIS-32, Grappolo, and Rabbit-Order; followed by RCM which generates an average gap profile that is between roughly 1x–8x more than the first group for at least 50% of the inputs. The third group consisting of a mixture of schemes from different categories generates an average gap profile that is roughly between 5x–25x larger, and the final group constituting of degree-/hub-based schemes is roughly between 10x–40x larger. We conclude that the partition-based schemes and RCM are superior to other schemes by this edge gap statistic.

Application impact: The performance of four different schemes was evaluated on a parallel community detection implementation. The key takeaway is that Grappolo usually outperforms Degree Sort, at times by factors 2x–4x when looking at Phase and Iteration times. It also usually has the highest Parallel Efficiency (Work%). It also has the lowest work per edge resulting in better load balancing. As for influence maximization, our evaluations showed little separation among the ordering schemes. This is owing to the application characteristic which performs many probabilistic BFS’s.