We propose and implement three compressed out-edge representations for distributed graph processing. We focus on Pregel-like graph processing systems and examine the efficiency of our techniques by extending Apache Giraph. Unfortunately, vertex-centric partitioning hinders the task of compression, and all related research efforts have focused on a centralized machine environment.

**Our contributions:**

i) We build on empirically observed properties of real-world graphs to offer three space-efficient out-edge representations.

ii) Our representations retain the efficiency of uncompressed structures speed-wise.

iii) Our memory-optimized techniques are clearly superior to current approaches when memory is an issue, and successfully perform executions in settings where Apache Giraph fails due to memory requirements.

**BVEdges**

BVEdges adjusts methods of the WebGraph framework [BV04] in the context of vertex-centric partitioning. The focus is solely on compressing the out-edges of a vertex. We use the first out-edge as a reference and proceed with gap encoding: intervals: Elias γ-coding residuals: ζ-coding

We set the minimum interval length to 4. Unfortunately, bit-encoding involves significant overhead.

**IntervalResidualEdges**

IntervalResidualEdges also incorporates the idea of using intervals and residuals. Intervals comprise the unencoded value of the 1st out-edge, plus 4 bytes holding the size of the interval.

We set the minimum interval length to 2. Residuals are held unencoded. Due to the locality of reference, we can compress the adjacency list significantly, while also avoiding the use of expensive encodings and bit streams.

**ByteArrayEdges**

ByteArrayEdges is Apache Giraph’s default out-edge representation: A byte-array is used, in which target vertex ids are held consecutively.

We use the first out-edge as a reference and proceed with gap encoding: intervals: Elias γ-coding residuals: ζ-coding

We set the minimum interval length to 4. Unfortunately, bit-encoding involves significant overhead.

**IndexedBitArrayEdges**

IndexedBitArrayEdges exploits the concentration of edges in specific areas of the adjacency matrix, regardless of whether these edges are consecutive. We construct a data structure of 5-byte elements, one for each interval of out-edges with ids having the same quotient by 8.

The first 4 bytes of each element represent the quotient, while the last one serves as a set of 8 flags indicating whether each possible edge in this interval exists. As the neighbor ids of each node tend to concentrate within a few areas, the number of intervals is small and the compression achieved is exceptional.

**Future Directions - Contact**

We continue with a tree-based space-efficient representation that favors mutations. Our initial findings show space improvements above 40% over Apache Giraph’s HashMapEdges, without inducing additional computational costs.

Moreover, we will investigate how we can compress weighted graphs to further increase the space-efficiency of our approach.

Contact info: http://hive.diuoa.gr/network-analysis