Which Graph Representations Does Giraph Support?

3/5/2013 UPDATE: Lots of new API changes and new Graph representation options, especially around Edges, Edge weights/values, and Multigraph support. More docs to follow!

Due to the distributed BSP nature of Giraph, all graphs are represented by application code as a Vertex implementation. By subclassing existing Vertex implementations (strongly suggested at this stage of Giraph development) the user can decide on trade offs in memory and speed vs. feature set of that Vertex. This is also important because common, provided base classes in Giraph handle the underlying mechanisms for messaging vertices that are not local to the worker node hosting the sender vertex, etc. Only very experienced users are likely to have good luck re-implementing low-level Vertex bases such as those in graph/ – New users are encouraged to extend or emulate user-ready templates in the example/ directory instead.

Directed Graphs: All Giraph input data is considered directed by default. All edges represented in input data are assumed to be out-edges extending from the vertex with which they are associated in said data.

Undirected Graphs: When supplying a Giraph job with input data, the user must ensure that this data includes an out-edge from both adjacent neighbors in an undirected relationship. Example: if vertex 1 has an out-edge to vertex 33 in your data, when vertex 33 is read it had better contain an out-edge back to 1. If the data does not supply this information, Giraph will not detect this discrepancy for you!

Unweighted Graphs: In Giraph, an unweighted graph is selected by (1) Choosing or supplying a VertexInputFormat/VertexReader combo that expects out-edges without attached value data, and (2) setting the generic E parameter in the Vertex<I,V,E,M> as a NullWritable. This includes supplying NullWritable for the edge values where required (use NullWritable.get() to acquire a NullWritable object in Giraph, use "new" for other Writables!)

Weighted Graphs: A weighted graph can be supplied as input to Giraph when the user (1) Chooses or supplies a VertexInputFormat/VertexReader combo that expects out-edge destination data to also include out-edge value data, and (2) chooses the Writable subclass that the user wishes to represent the edge values and to applies it to the generic E parameter in application code.

Multigraphs: Giraph now features native support for Multigraphs! More to follow...