Beyond 'Caveman Communities': Hubs and Spokes for Graph Compression and Mining

1. What is the problem that the paper wants to solve? Why is it difficult (related works)?
   - Problem definition: Given a real world graph, how can we compress it (and its edges)? This problem is difficult as previous work has shown that it is difficult to find 'good cuts' in real world graphs due to hub nodes.
   - Subproblem definition: Given a graph $G = (V, E)$ with adjacency matrix A find permutation $\pi : V \rightarrow [n]$ such that a cost function is minimized
2. What is the solution? What is the main idea?
   - Observation: Compressibility of adjacency matrix depends on node ordering
   - Find better node ordering by exploiting the existence of high-degree hub nodes to achieve a better compression ratio
   - The authors introduce an algorithm called SlashBurn to find such an ordering:
     1. remove top-k highest centrality scoring nodes, give these hubs the lowest id,
     2. give lowest-size connected components ("spokes") the lowest IDs
     3. repeat on GCC of G to get an ordering of nodes in the GCC
3. What is the result?
   - Higher rate of compression for real-world graphs that follow a power-law distribution
   - Speed-up for matrix-vector multiplications which play an important role in many graph mining algorithms as these depend on the node ordering
4. What is the main novelty that enabled the solution?
   - Observation that real-world graphs follow a power-law and that this graph structure can be used to "shatter" the graph and iteratively order the nodes to achieve a better compression of the adjacency matrix
5. What are the good aspects of the paper? Did you learn something from the paper?
   - Node reordering provides not only better compression but also speed-up for matrix-vector operations
   - Real-world graphs can be shattered quickly
6. What is the impact of the paper?
   - The authors provide a way to speed-up many graph mining algorithms that operate on large real-world graphs such as PageRank
   - Works on graphs with "no good cuts" where the cavemen graph compression approach will not work well
7. Are there weaknesses/missing parts in the paper? How can you improve it?
   - It is claimed that SlashBurn is better for real-world graphs than clique-based compression approaches as there is no good cuts, but there is no direct comparison between the two
8. How can you extend the paper?
   - Find a similar method which works well on graphs that do not follow a power-law as well (not all real-world graphs follow a power-law)
9. How can you apply the technique to other data/problems?
   - Exploit the graph structure (or structure of data in another problem) to achieve better performance