1 Common Approach

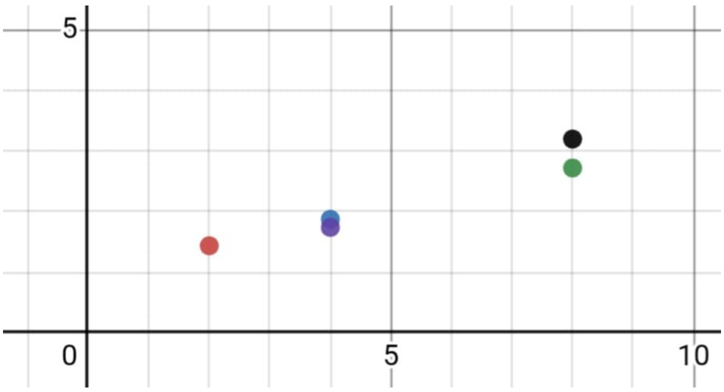
The algorithm used to find the truss value of each edge is the same for both task 1 and task 2. The algorithm implemented is the hybrid truss algorithm. A directed version of the graph is stored at each rank, and the triangles are enumerated for a subset of edges.
Through inter-process communication, the values of each edge reduce monotonically and eventually converge from the support of the edge to the final truss value.

2 Task 1

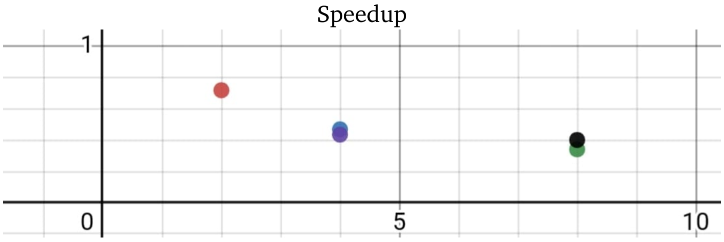
2.1 Approach

Not Verbose:
Check if a truss of the given size exists.
Verbose:
If a truss exists, get all existing edges and run BFS to find connected components.

2.2 Speedup & Efficiency



Speedup



Efficiency

2.3 Iso-Efficiency

After observing the results, the best function for this algorithm's iso efficiency is $\Omega(p^{1.5})$.

2.4 Sequential Fraction Estimation

The sequential fraction of the algorithm is strictly decreasing as the number of processes and threads increases. Therefore I estimate that the sequential fraction of the algorithm will be approximately 0.15,0.1,0.08 for 16,32,64 cores, respectively.

2.5 Scalability

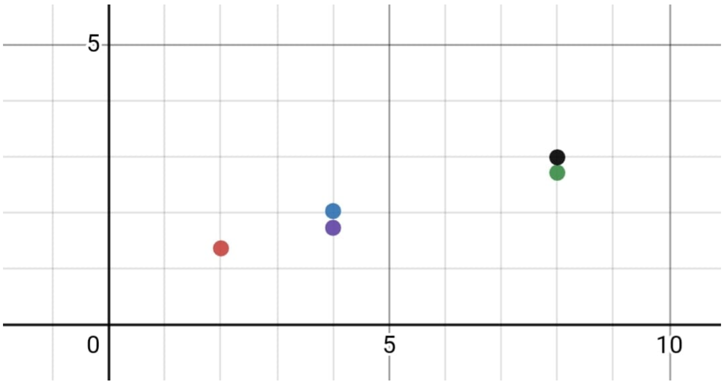
The algorithm depends on the graph structure, so drawing a concrete conclusion about the scalability is difficult. However, from the figures, it is visible that the problem is scaling decently as cores increase.

3 Task 2

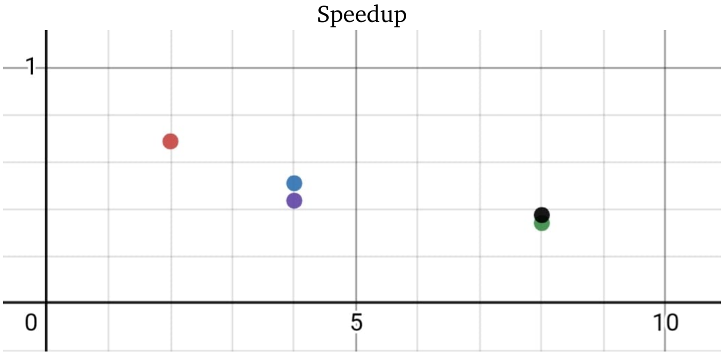
3.1 Approach

Take the connected components of the k-truss - found using BFS.
Not Verbose
Check which vertices are connected to a sufficient number of components.
Verbose
Find the vertices connected to a sufficient number of components and output those components.

3.2 Speedup & Efficiency



Speedup



Efficiency

3.3 Iso-Efficiency

After observing the results, the best function for this algorithm's iso efficiency is $\Omega(p^{1.5})$.

3.4 Sequential Fraction Estimation

The sequential fraction of the algorithm is strictly decreasing as the number of processes and threads increases. Therefore I estimate that the sequential fraction of the algorithm will be approximately 0.15,0.1,0.08 for 16,32,64 cores, respectively.

3.5 Scalability

The algorithm depends on the graph structure, so drawing a concrete conclusion about the scalability is difficult. However, from the figures, it is visible that the problem is scaling decently as cores increase.