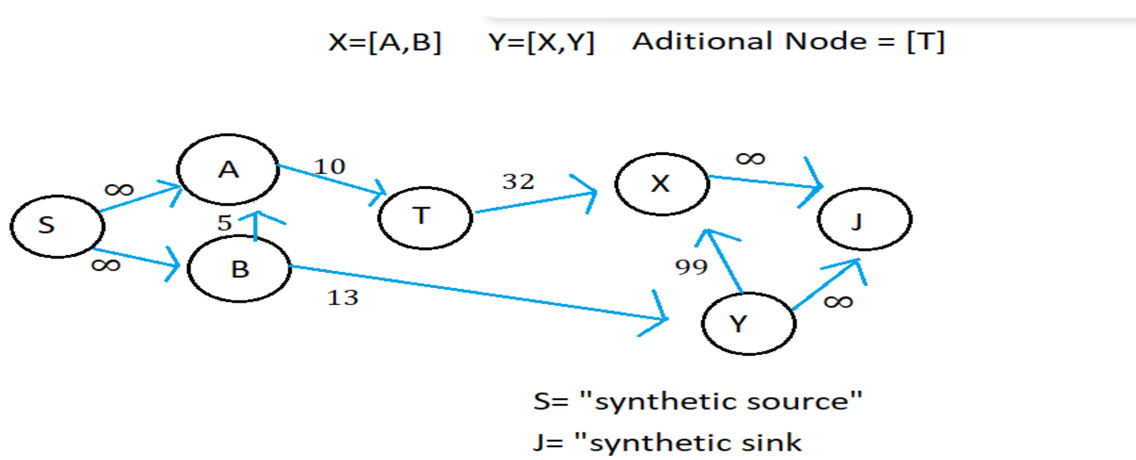
Elad Ohayon

1. This problem is mapped out to a network flow by representing the granaries as vertices and the rate at which grain can be transferred from one granary to another as an edge with a capacity. This problem is identical to the standard network flow model, with the addition that all granaries in X are source vertices and all granaries in Y are sink vertices; meaning there are multiple sources/sinks, and the source vertices may have incoming edges and the sink vertices may have outgoing edges. This is what is missing from a straightforward reduction, a way to account for multiple sources/sinks and these sources may be linked to each other. Any additional granaries "hitherto unknown nodes" are regular vertices and thus require no special attention.

2. I solved this issue by adding a "synthetic" source vertex which has an edge to all the vertices in X and adding an edge from all vertices in Y to a "synthetic" sink. This solves the issue as we now have a "source" with no incoming edges, and a "sink" with no outgoing edges which we can perform on Max-Flow Algorithm on. The capacity of the edges going from the synthetic source/sink to the real source/sinks are initialized to be ∞, so that they are not forming a bottleneck (will be proved in (3)).

Trivial Example:



In this example, edge weights represent capacities and display all the relevant reductions to the standard network flow model.

3. The goal of this problem is to move 10,000 bushels as fast as possible from one set of granaries to another. This directly translates to network flow as we substitute the granaries the bushels come from as source vertices, the destination granaries as sink vertices, and the network through which we can move bushels as edges with a capacity. This problem translates to a network flow directly. This model is feasible as we add a synthetic source and sink with edge capacity ∞, thus having one source and sink. This gives us the source and sink the algorithm needs to function and by making these capacities ∞ we ensure they don't create bottlenecks nor change the max flow. Proof: If we have an edge of capacity ∞ (synthetic vertex) to an edge of capacity 5 (original graph) the max flow of 5 will be preserved as the 5 limits the edge with ∞ flow.