Exploring Algorithmic Solutions for the Maxflow/Mincut Problem

Team Members (Group 25)

Asad Ullah Chaudhry - ac07408
Shayan Shoaib Patel - sp07101
Ali Raza - ar07530

Problem Description:

The Max-Flow/Min-Cut Problem is one of the fundamental problems in graph theory and Computer Science. It involves calculating the maximum flow that can be sent from the starting node (Source Node) to the last node (Sink Node). Secondly, it also involves identifying the minimum capacity cut (Min cut) in the flow network by partitioning the network into two disjoint sets such that the Source node and the Sink node are in separate sets. The Max-Flow/Min-Cut Problem is extensively used to optimize network flows in any scenario where any resource is to be routed from a source to a destination such as traffic / transportation, water distribution networks, energy grids, telecommunication and computer networks.

Algorithms

- Dinic's Algorithm: Based on the concept of level graphs and blocking flow, it is more efficient than the Edmonds-Karp algorithm and has a time complexity of O(V^2*E) for integer capacities and O(min(V^(2/3), E^(1/2))*E) for real capacities.

- Push-Relabel Algorithms (e.g., Goldberg-Tarjan, Preflow-Push): These algorithms are based on the concept of "preflows" and "push-relabel" operations. They maintain a feasible flow throughout the process and iteratively improve it until reaching the maximum flow. They have a time complexity of O(V^2*E) in the general case, but with appropriate data structures and optimizations, they can achieve better performance in practice.

- Capacity Scaling Algorithm: This is a modification of the Ford-Fulkerson algorithm where the capacities of the edges are successively scaled down by powers of 2 until all capacities become integers. This allows for faster convergence in some cases.The scaling algorithm performs O(log D) scaling phases, O(m) shortest path augmentations in each scaling phase and, consequently, runs in O(m log D S(n, m)) time

- Network Simplex Algorithm: While traditionally used for solving linear programming problems, the network simplex algorithm can also be adapted to solve the maximum flow problem by transforming it into a minimum-cost circulation problem. It's efficient for dense graphs. Network simplex will run with a time complexity of O(VE) on average, and with a pretty good constant.

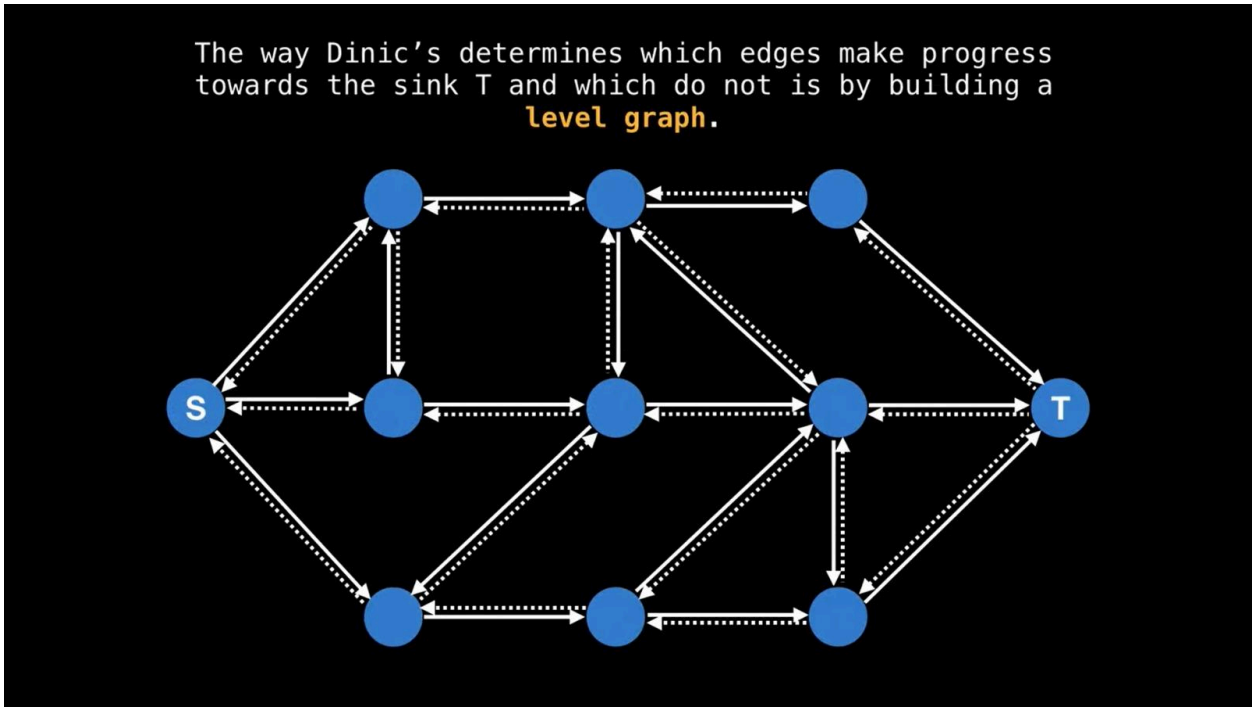
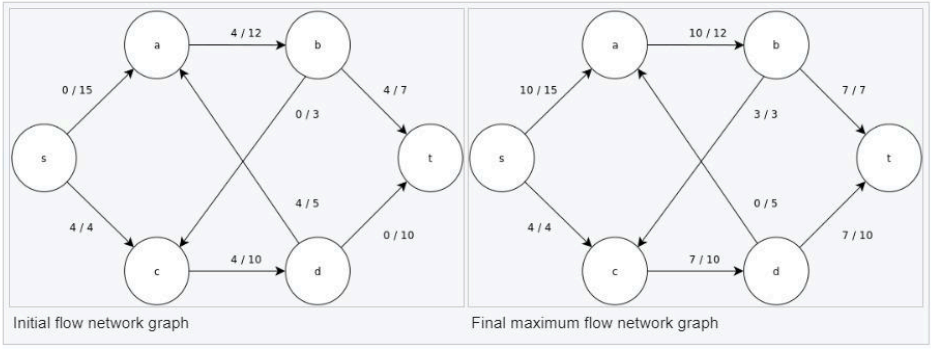


Fig 1.1 Dinic's Algorithm

Example [edit]

The following is a sample execution of the generic push-relabel algorithm, as defined above, on the following simple network flow graph diagram.



In the example, the *h* and *e* values denote the label ℓ and excess x_f, respectively, of the node during the execution of the algorithm. Each residual graph in the example only contains the residual arcs with a capacity larger than zero. Each residual graph may contain multiple iterations of the *perform operation* loop.

Fig 1.2 Push-Relabel Algorithms

References

-Wikipedia
-Introduction To Algorithms