Hello. My name is Vyacheslav. Topic of my presentation is “Сalculation and optimization of flows in networks”.

First of all I would like to tell about the task of my master’s work. We are given undirected graph, bandwidth and probability of existence for all channels, and we need to find the mathematical expectation of the values of the maximum flows between all pairs of vertexes.

Theme of my course work is find the values of the maximum flows between all pairs of vertexes in undirected constant graph.

I would like to introduce some definitions. This will denote edge between vertexes i and j. The channel capacity will be denoted by letter c with two indexes: initial and final vertexes. The value of maximum flow will be denoted the same way, but with letter f.

The maximum flow problem between pair of vertexes is a standard problem in graph theory. There are many algorithms, which can solve this problem. In my work, I used three algorithms: Edmonds-Karp algorithm, Dinic’s algorithm and push-relabel algorithm. Simplest solution of my course work is to apply any of this algorithm to all pair of vertexes, but this can be very slow and not using an additional information. The standart approach is to use reduction methods.

First of reviewed reduction methods is split graph into disconnected components. Obviously, maximum flow between vertexes, lying in different components, is equals to zero, so we can eject components and calculate maximum flow in them.

The second reduction is chain contractions. When we find a chain, we can replace it to one edge, which bandwidth equal to minimum of all bandwidths in chain, and calculate flows in resulting graph. To restore maximum flows in the original graph, we need to consider all pairs in initial graph. There are three cases: when both vertexes are not in the chain, when one vertex are in chain and another is not, and when both vertexes are in chain. All of this cases are considered in text of the work.

Next reduction is bridge handling. When we find a bridge, we can split graph into two components and calculate flows in them. To restore flows in initial graph, we need to consider all pairs. There are two cases: when both vertexes in one component and when vertexes lie in different components. When both vertexes lie in one component, second component doesn’t affect to flows between him. When vertexes lie in different components, the maximum flow is minimum from maximum flow from initial vertex to bridge, bandwidth of bridge and maximum flow from bridge to final vertexes.

Last reviewed reduction is removing hanged vertexes. When we find hanged vertex, we can delete in from graph and calculate flows in resulting graph. To restore flows, we need to consider two cases: when both vertexes are not hanged, maximum flow is equal to maximum flow in transformed graph. When one vertex are hanged, maximum flow between him equals to minimum of maximum flow from initial vertex to supporting vertexes and bandwidth of supporting edge.

Six algorithms were implemented: three of them are brute-force of pairs and three of them use reduction methods.

Conclusions:

The Edmonds-Karp algorithm showed itself better both in the case of brute-force pairs, and in the case of brute-force pairs using reductions

Reductions give an increase in speed, even at small dimensions

The increase in speed from reductions increases with increasing dimension

Further directions:

Solving problems set in the master's work

Search and analysis of new methods of reduction

Improvement of the developed algorithms

Implementation of new basic algorithms