Edge connectivity / Vertex connectivity

Table of Contents

Definition

Given an undirected graph G with n vertices and m edges. Both the edge connectivity and the vertex connectivity are characteristics describing the graph.

Edge connectivity

The **edge connectivity** λ of the graph G is the minimum number of edges that need to be deleted, such that the graph G gets disconnected.

For example an already disconnected graph has an edge connectivity of 0, a connected graph with at least one bridge has an edge connectivity of 1, and a connected graph with no bridges has an edge connectivity of at least 2.

We say that a set S of edges **separates** the vertices s and t, if, after removing all edges in S from the graph G, the vertices s and t end up in different connected components.

It is clear, that the edge connectivity of a graph is equal to the minimum size of such a set separating two vertices s and t, taken among all possible pairs (s, t).

Vertex connectivity

The **vertex connectivity** κ of the graph G is the minimum number of vertices that need to be deleted, such that the graph G gets disconnected.

For example an already disconnected graph has the vertex connectivity 0, and a connected graph with an articulation point has the vertex connectivity 1. We define that a complete graph has the vertex connectivity $n-1$. For all other graphs the vertex connectivity doesn't exceed $n-2$, because you can find a pair of vertices which are not connected by an edge, and remove all other $n-2$ vertices.

We say that a set T of vertices **separates** the vertices s and t, if, after removing all vertices in T from the graph

G, the vertices end up in different connected components.

It is clear, that the vertex connectivity of a graph is equal to the minimal size of such a set separating two vertices s and t, taken among all possible pairs (s, t).

Properties

The Whitney inequalities

The **Whitney inequalities** (1932) gives a relation between the edge connectivity λ, the vertex connectivity κ and the smallest degree of the vertices δ:

$$\kappa \le \lambda \le \delta$$

Intuitively if we have a set of edges of size λ, which make the graph disconnected, we can choose one of each end point, and create a set of vertices, that also disconnect the graph. And this set has size $\le \lambda$.

And if we pick the vertex and the minimal degree δ, and remove all edges connected to it, then we also end up with a disconnected graph. Therefore the second inequality $\lambda \le \delta$.

It is interesting to note, that the Whitney inequalities cannot be improved: i.e. for any triple of numbers satisfying this inequality there exists at least one corresponding graph. One such graph can be constructed in the following way: The graph will consists of $2(\delta+1)$ vertices, the first $\delta+1$ vertices form a clique (all pairs of vertices are connected via an edge), and the second $\delta+1$ vertices form a second clique. In addition we connect the two cliques with λ edges, such that it uses λ different vertices in the first clique, and only κ vertices in the second clique. The resulting graph will have the three characteristics.

The Ford-Fulkerson theorem

The **Ford-Fulkerson theorem** implies, that the biggest number of edge-disjoint paths connecting two vertices, is equal to the smallest number of edges separating these vertices.

Computing the values

Edge connectivity using maximum flow

This method is based on the Ford-Fulkerson theorem.

We iterate over all pairs of vertices (s,t) and between each pair we find the largest number of disjoint paths between them. This value can be found using a maximum flow algorithm: we use s as the source, t as the sink, and assign each edge a capacity of 1. Then the maximum flow is the number of disjoint paths.

The complexity for the algorithm using Edmonds-Karp is $O(V^2 V E^2) = O(V^3 E^2)$. But we should note, that this includes a hidden factor, since it is practically impossible to create a graph such that the maximum flow algorithm will be slow for all sources and sinks. Especially the algorithm will run pretty fast for random graphs.

Special algorithm for edge connectivity

The task of finding the edge connectivity if equal to the task of finding the **global minimum cut**.

Special algorithms have been developed for this task. One of them is the Stoer-Wagner algorithm, which works in $O(V^3)$ or $O(VE)$ time.

Vertex connectivity

Again we iterate over all pairs of vertices s and t, and for each pair we find the minimum number of vertices that separates s and t.

By doing this, we can apply the same maximum flow approach as described in the previous sections.

We split each vertex x with $x \neq s$ and $x \neq t$ into two vertices x_1 and x_2. We connect these to vertices with a directed edge (x_1, x_2) with the capacity 1, and replace all edges (u, v) by the two directed edges (u_2, v_1) and (v_2, u_1), both with the capacity of 1. The by the construction the value of the maximum flow will be equal to the minimum number of vertices that are needed to separate s and t.

This approach has the same complexity as the flow approach for finding the edge connectivity.