Discrete Optimization: Homework #11, Ex. #4

Denis Steffen, Yann Eberhard & Gaëtan Bossy

May 20, 2018

We assume that $e_1 \neq e_2 \ \forall e_1, e_2 \in E$. If the graph isn't connected, we use the algorithm for every connected components.

Algorithm :

Input : $G = (V, E)$, a graph.

While $oracle(G) = false$: We delete the vertex v with minimum degree such that the graph without v remains connected and update $G = (V \backslash \{v\}, E')$ where $E' := \{(a, b) \in E : a \neq v \text{ and } b \neq v\}$.

Then we take the edge $e = (u, v)$ such that the sum of $deg(u)$ and $deg(v)$ is minimal and such that the graph without this edge remains connected. We put $M := \{e\}$ and update $G = (V \backslash \{u, v\}, E \backslash \{e\})$. We repeat this until $V = \varnothing$ and each time we add in M the edge that we delete.

At the end, M is a maximum cardinality matching of $G = (V, E)$ obtained with at most $|V| + |E|$ calls to the oracle.