Discrete Optimization: Homework #11, Ex. #4

Denis Steffen, Yann Eberhard & Gaëtan Bossy

May 21, 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
Output : M a maximum cardinality matching

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 a vertex v from the updated graph $G = (V, E)$. Let $F = \{(u, v) : (u, v) \in E\}$, we call $oracle((V, (E \backslash F) \cup \{f\}))$ for every $f \in F$ until there is a positive response of the oracle for a particular $f = (u, v) \in F$ (there exists a such f because G has a perfect matching). We put f in M and update $G = (V \backslash \{u, v\}, E \backslash (F \cup G))$ where $G = \{(u, w) \in E\}$.
We repeat this until G is an empty graph.

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