#5 Assignment - CMPT 405

Luiz Fernando Peres de Oliveira - 301288301 - lperesde@sfu.ca

November 29, 2018

#1

Because an edge covering of G is a set $A \subset E$ such that for every node $v \in V$, v is one of the vertices in at least one of the edges in A, we know that there is a polynomial algorithm that describes finding an edge cover of G, for example by simply running *BFS*. However, because finding the smallest possible set of edges $A^* \subset E$ that satisfy an edge covering is an optimization problem, we need to prove that finding such set can be done in polynomial time.

The intuition of our algorithm to minimize the number of edges will be that an edge covering cannot have subpaths of more than two edges. A simple proof would be: let A' be an edge covering of G and let p be a subpath formed by some of the edges of A', where $|p| > 2$. If p is $a \to b \to c \to d$, then $|A'| > |A^*|$, as there is an extra edge, (b, c) in this case, that could be removed, thus breaking p into two different paths $p_1 = a \to b$ and $p_2 = c \to d$ would minimize the edge covering and therefore $|A'|$ cannot equal $|A^*|$.

Because the property above holds true, where for every subpath $p \subset A^*$, $|p| = 1$ or $|p| = 2$, we can solve this problem by first computing a maximum matching on G (therefore covering every p, where $|p| = 1$), and then, for each vertex v left uncovered, covering v by adding an edge connecting to one of the already covered vertices (therefore augmenting the size of some p). Because the second part of the algorithm is trivial ($O(m + n), n = |V|, m = |E|$), the total running time of the algorithm is the same as Edmonds algorithm ($O(n^2 m)$ or $O(\sqrt{m}n$ with improvements), thus, it can be implemented in polynomial time.

#2

The intuition here is that the *half-3-CNF-SAT* is NP-Complete, as the problem is a subset of the *3-SAT* problem. We then want to prove that it is possible to reduce the *3-SAT* problem to the *half-3-CNF-SAT* in polynomial time and finally, show that finding a truth assignment ϕ that satisfies exactly $k/2$ of the k clauses is NP-hard.

Proof. **Reducing 3-SAT to half-3CNF-SAT:** Because half-3-CNF-SAT problem is the problem of checking if exactly half of the clauses evaluate to true, then half of the clauses must evaluate to false. Imagine ϕ contains m clauses, then, in our reduction we will add three variables x, y and z and create a ϕ' with $4m$ clauses, such that ϕ' is an extension of ϕ, meaning that ϕ' contain all m clauses of ϕ and:

- m clauses of the form: $(x \vee \neg x \vee y)$ *// always true*
- $2m$ clauses of the form: $(x \vee y \vee z)$ *// either all true or all false*

Reduction in polynomial time: The reduction can be done in polynomial time because we only added three new variables, copied the m clauses from ϕ and created $3m$ clauses.

3-SAT has a solution iff half-3-CNF-SAT has a solution:

$\Longrightarrow$ Assume that there is a truth assignment ϕ that satisfies the m clauses. Then there are $2m$ clauses in ϕ' that are satisfied (the clauses in $phi+$ the m clauses that are always true and we let x, y and z be false so that $2m$ clauses are false. Therefore, half of the clauses are true and half o the clauses are false.

$\Longleftarrow$ Assume that there is a truth assignment that satisfies *half-3-CNF-SAT* or a truth assignment that makes half the clauses in ϕ' true and half false. As m clauses are always true in ϕ', $2m$ clauses cannot be true if *half-3-CNF-SAT* is satisfied because that would comprise $3m$ clauses, therefore breaking it. Thus, the $2m$ clauses must be false so that ϕ is true and meaning that *3-SAT* is also satisfied.

NP-hard problem: Because *half-3-CNF-SAT* is NP-complete, its optimization version (finding a set of exactly $k/2$ clauses that are satisfied by ϕ) is NP-hard.

$\square$

#3

#4

#5