Assignment 4
COMP6741: Parameterized and Exact Computation
Serge Gaspers
Semester 2, 2018

Assignment 4 is a group assignment. The lecturer-in-charge will announce the groups on 03 September 2018. Each group consists of 3–4 members.

For the solutions to this assignment, you may rely on all theorems, lemmas, and results from the lecture notes. If any other works (articles, Wikipedia entries, lecture notes from other courses, etc.) inspired your solutions, please cite them and give a list of references at the end. You may use any result you find in the literature, without re-proving it. Existing implementations and libraries may also be used, as long as their licenses allow unrestricted academic use. General-purpose solvers (such as SAT solvers, Constraint solvers, Integer Linear Programming solvers) are prohibited though.

If you have questions about this assignment, please post them to the Forum.

Due date. This assignment is due on Sunday, 16 September 2018, at 23.59 AEST. Submitting $x$ days after the deadline, with $x > 0$, reduces the grade by $20 \cdot x$ per cent.

How to submit. There will be one Bitbucket GIT repository per group. The Readme.md file in this repository describes where to put various files, including the report answering the questions below, by the submission deadline.

We will consider various parameterizations of Vertex Cover and its optimization version, Minimum Vertex Cover. It is recommended to read the whole assignment sheet before staring to solve individual exercises.

**Vertex Cover**

Input: A graph $G = (V, E)$ and an integer $k$

Question: Does $G$ have a vertex cover of size at most $k$?

**Minimum Vertex Cover**

Input: A graph $G = (V, E)$

Output: A smallest vertex cover of $G$

We use the notation from \cite{H} to compare graph parameters. A graph parameter $p$ is a function that assigns a real number to each graph. If $p, q$ are graph parameters, then the parameter $p$ upper bounds the parameter $q$ if there is a function $f$ such that for every graph $G$ we have that $q(G) \leq f(p(G))$. If $p$ does not upper bound $q$ and $q$ does not upper bound $p$, then $p$ and $q$ are said to be incomparable.

Choose two incomparable parameters $r$ and $s$ from the following list:

- the feedback vertex set number of $G$;
- its distance to graphs with maximum degree at most 2;
- its distance to bipartite graphs;
- $\tau(G) - \nu(G)$, where $\tau(G)$ denotes the number of vertices in a smallest vertex cover of $G$ and $\nu(G)$ denotes the number of edges in a largest matching of $G$;
- the twin cover number of $G$; and
- the neighborhood diversity of $G$.
It is expected that you will need to check the literature for the appropriate definitions of some of these parameters. The distance of a graph $G$ to a graph class $C$ is the smallest number of vertices we need to delete from $G$ in order to obtain a graph from $C$.

**Exercise 1.** Prove that your chosen parameters $r$ and $s$ are incomparable. [15 points]

**Exercise 2.** Prove that Vertex Cover is FPT for parameter $r$. [15 points]

**Exercise 3.** Prove that Vertex Cover is FPT for parameter $s$. [15 points]

In Assignment 5, you will be asked to implement algorithms for Minimum Vertex Cover that exploit the parameters $r$ and $s$. In particular, for an input graph $G$, it is expected that your implementation is fast whenever $r(G)$ or $s(G)$ is small. You will run your implementation on benchmark instances.

**Exercise 4.** Design and implement a method for generating instances for Minimum Vertex Cover. Ideally, it generates instances where you believe your implementation of Assignment 5 will solve many such instances in under 10 minutes, whereas the implementations of other groups will not find a solution by the cut-off time of 10 minutes.

Give a high-level description of your methods and explain your choices.

Place 20 instances into the sub-folder with your benchmark instances (GIT repository). [20 points]

**Exercise 5.** Discuss how you propose to implement and combine the two algorithms for Minimum Vertex Cover exploiting the parameters $r$ and $s$. They do not necessarily need to closely follow your solution of Exercises 2 and 3. For example, give details on

- proposed programming language
- if you plan to compute, approximate, or heuristically estimate $r(G)$ and/or $s(G)$, what methods and algorithms do you use?
- what data structure will you use for graphs
- if you use a branching algorithm, how will you pass the instance to the recursive calls; will you need to copy the instance each time?
- if you use a branching algorithm, will you use any heuristics to select the vertex to branch on?
- etc.

You should describe implementation details that are relevant to your proposed implementation. [25 points]

**Exercise 6.** Describe how you divided the work of Assignment 4, what milestones you set, and what was your timeline. Did any of these evolve over time? If any issues or setbacks arose, how did you handle them? [10 points]

**References**