Assignment 5
COMP6741: Parameterized and Exact Computation
Serge Gaspers
Semester 2, 2018

Assignment 5 is a group assignment. The groups are the same as for Assignment 4.

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 of your report. 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, 07 October 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. Use the existing Bitbucket GIT repository that was assigned to your group. The Readme file in this repository describes where to put various files, including the report answering the questions below, by the submission deadline.

In Assignment 4 you have chosen two parameters $r$ and $s$ for Vertex Cover and proposed FPT algorithms for these parameterizations. It was also stated in Assignment 4 that

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.

In addition, you have already implemented a method for generating instances for Minimum Vertex Cover.

Exercise 1. This should be done in the beginning of your work on Assignment 5, but can evolve over time. Give a plan for the work done in Assignment 5, including milestones, a timeline with deadlines, assignment of tasks/subtasks to group members. Be conscious that several tasks need to be performed after the implementation is completely finalized, including running it on benchmark instances (estimate how long this takes), evaluating the results, and finalizing your report. Ideally, you would also leave some additional time in case there are unforeseen issues. Did your timeline evolve over time? If any issues or setbacks arose, how did you handle them? [20 points]

Exercise 2. Implement an algorithm, which, given a description of an instance for Minimum Vertex Cover on the standard input, prints “ok” to the standard output if the description has a valid format, and it prints error messages otherwise. [10 points]

Exercise 3. Implement an algorithm, which, given a valid description of an instance for Minimum Vertex Cover and a description of a solution, prints “ok” to the standard output if the solution describes an actual vertex cover for the instance (not necessarily a smallest vertex cover), and it prints error messages otherwise. [10 points]

Exercise 4. Implement three algorithms

- an algorithm based on the parameterization by $r$,
- an algorithm based on the parameterization by $s$, and
- an algorithm combining the two previous algorithms.
Describe if, and why, any implementation details have changed, as compared to your plan in Assignment 4. Recall that your implementations do not necessarily need to closely follow your solutions of Exercises 2 and 3 in Assignment 4. [40 points]

**Exercise 5.** Describe your test environment (details on CPU, operating system, system memory, etc.). [5 points]

**Exercise 6.** Run the three implementations of Exercise 4 on the final set of benchmark instances in your test environment, with a 10-minute time-out. Depict the results in a cactus plot. In case you use randomization, how many times do you perform each experiment? Interpret the results: which implementation performed best; among the instances that you generated, did your solvers have the expected running times? Are your results reproducible (i.e., is it easy for anyone with access to your report, your code, a similar test environment, and an Internet connection to reproduce your results)? [15 points]

**Bonus Question.** What can be improved in your implementations, and what are the obstacles that need to be overcome for these improvements? [5 points]