Assignment 2
COMP6741: Parameterized and Exact Computation

2017, Semester 2

Assignment 2 is based on group work. Each group consists of 3 students (group sizes of 2–4 are acceptable). Post on the Forum if you are looking for a group to join, or if your group is short of members.

If any works (articles, Wikipedia entries, lecture notes, etc.) inspired your solutions, please cite them and give a list of references at the end of your report.

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

Due date. This assignment is due on Wednesday, 20 September 2017, at 23.59 AEST. Submitting x days after the deadline, with $x > 0$, reduces the grade by $20 \cdot x$ per cent.

Submission. One group member submits a TAR(.GZ) or ZIP archive with the following files:

- a report in PDF format (the first page should contain the names and Student IDs of each group member), and
- all source files of the implementations,
- a statically-linked executable of the implementation, and
- two shell scripts `mycompile.sh` and `myrun.sh` such that `mycompile.sh` compiles the code (in case you use an interpreted language, this shell script might simply do nothing), and `myrun.sh` executes the algorithm implementation with default parameters on an instance that you provide.

Submit this archive using the command

give cs6741 a2 <myarchive>

from the CSE network, or use the WebCMS3 frontend for `give`.
Size limit: 50 Megabytes

Assignment

This assignment centers around the Boolean Satisfiability problem and focuses on practical aspects.

Satisfiability (Sat)	
Input:	A Boolean formula F in conjunctive normal form.
Question:	Is there an assignment $\alpha : \mathsf{var}(F) \rightarrow \{0,1\}$ to the variables of F for which F evaluates to 1?

Literature review

The first part of the assignment is a literature review. Your descriptions and explanations of procedures/algorithms should be detailed enough so that a typical COMP6741 student can implement them based only on your description. If there are multiple variants of a given concept, it is sufficient to describe only one variant.

1. Describe the unit clause simplification rule, also known as unit propagation (UP). Give an example of a SAT instance where UP can be applied. [5 points]

2. Describe the pure literal rule (PL). Give an example of a SAT instance where PL can be applied. [5 points]

3. Describe the concept of random restarts. [5 points]

4. Describe two variable selection heursistics, for choosing a variable to branch on (and maybe to decide which value we assign to the variable in the first branch). [10 points]

5. Describe Conflict-Driven Clause Learning (CDCL), including any other needed concepts, such as implication graphs. Give an example of a SAT instance and an execution of a CDCL algorithm where CDCL helps to decrease the size of the search tree. [25 points]

Implementation, experiments, and interpretation of results

Implement a CDCL algorithm for SAT using a common programming language such as C, C++, Python, or Java. It is recommended to base your implementation on an existing solver, such as the open-source solver MiniSAT.

Choose a set of benchmark instances. A good source of benchmark instances are the SAT competitions.

Describe your test environment, time-outs (if no solution is found after, e.g., 5 minutes, stop the algorithm), memory limit (should be around 1-4 Gigabytes), and how you simulate the generation of random choices.

- Quality of code and selection of benchmark instances. [20 points]
- Resproducibility of results: it should be easy for anyone with access to your report, your code, and an internet connection to reproduce your results. [10 points]
- Run variants of your implementation on your benchmark instances to try to answer the following questions:

  1. How much does conflict-driven clause learning help in SAT solving? Compare no CDCL, a small amount of CDCL, and a large amount of CDCL.
  2. How much do the simplification rules UP and PL help? Compare only UP, only PL, simplification rules only at every kth level of the search tree (for $k = 2$ or $k = 3$), or an exhaustive use of simplification rules.
  3. Which variable selection heuristic performs better?

  What conclusions do your experiments allow you to make for these questions? [20 points]

Bonus question. What structure or properties do instances have where the unsuccessful strategies above actually performed better? [5 points]