Algorithm Engineering
Winter Semester 22/23

Exam Task

The implementation and the report have to be submitted by **15.03.2023**. For the exam task TEAMS ARE NOT ALLOWED! However, you can use your team GIT to submit your work. For that make subfolders named ExamSurname. If you have your own GIT repository anyway, it suffices to call the folder Exam.

Task

Choose an NP-complete graph problem. RESTRICTIONS: Do not choose problems that were explicitly discussed in the lecture or already used for exercises (as Vertex Cover or TSP), or something very similar to those. And if you happen to do your project or thesis on a suitable topic, it is also not allowed to choose the respective problem here. If you are in doubt whether your favourite graph problem is a good choice, don't hesitate to ask in the forum or via email.

For the chosen problem, implement

- an **exact algorithm**: The exact algorithm should either be one that allows for parametrized analysis (as a *bounded search tree*) or it should be an *ILP-formulation* followed by a call to a suitable ILP-solver.
- an **approximation algorithm**: Any algorithm with a *constant or logarithmic approximation factor* is fine here.
- a **heuristic**: This should be a sensible algorithm for which nevertheless instances exist on which the solution quality is (arbitrarily) bad compared to the optimum solution.

Furthermore, **parallelize** at least one of these three algorithms. Make sure that you implement the parallelization part on your own (calling an ILP-solver that allows for parallel computation does not count), and also that the parallelization happens at the core of the algorithm.

Input

You can also freely choose your input data. Here are some sources of nice graph data:

- PACE challenges (`https://pacechallenge.org/`). Of course, the graph data provided for a particular challenge can also be used for problems that were not the focus of that challenge.
- DIMACS challenges (`http://dimacs.rutgers.edu/programs/challenge/`).
- Stanford Large Network Dataset Collection (`https://snap.stanford.edu/data/`).
- (OSM) road networks (`https://i11www.iti.kit.edu/resources/roadgraphs.php`).
- TSP instances (`http://www.math.uwaterloo.ca/tsp/index.html`). Those are useful for graph problems defined on complete graphs.

You can use other graph sources as well. However, make sure that you document in your report precisely what data was used in the experiments. Of course, you don't have to evaluate your algorithms on all graph data you can find. Choose an interesting benchmark set.

Implementation

Implement the algorithms in a sensible fashion, that means, do not introduce unnecessary complexity. Enclose in your report the implementation details that are critical for the overall running time (e.g. the usage of certain data structures if applicable). Any Algorithm Engineering ideas used to accelerate the computation or improve the solution quality are very welcome and should be described in the report as well.

Experiments

Evaluate your algorithms using suitable experimental setups. Some guiding questions are listed below.

- How well do the three algorithms scale? What is the largest/most complex input graph they can deal with? What are the average/maximum running times? Do theoretical and practical running times concur?
- How good is the solution quality of the approximation algorithm and the heuristic on the tested instances? (This could be answered by a comparison to the exact solutions on instances where those could be obtained or by considering suitable lower/upper bounds on the optimum solution value.)
- What graph/input aspects, if any, have a significant influence on the running time/solution quality (for example, graph density, maximum node degree, weights, ...)?
- What speed-up can be achieved with the parallelized algorithm?
- How successful are the AE ideas in practice?

Report and discuss the results in any way you deem sensible (e.g. tables, plots, visualizations).

Evaluation

The final grade will depend on the following four quality criteria:

- Quality of Implementation: The code should be running and suitable test cases should be provided. The implementation should be sensible and not unnecessarily convoluted or complex. Use abstraction whenever possible (e.g. to avoid repeated code or very long methods) and comment your code.
- Algorithm Engineering Ideas: There should be AE aspects included in the implementation and the report (e.g. providing an analysis in PRAM or a parametrized running time analysis or coming up with pruning rules or other methods to accelerate an algorithm). Note that it is not demanded that those are your own ideas; it is fine to transfer those from papers. But there should be a discussion which involves the theoretical aspects and the practical outcomes.
- Expressiveness of Evaluation: The experimental setup should be suitable to investigate the questions listed above. The results should be presented in a clean fashion and it should be discussed whether those results comply with theoretical expectations; and if not what the reason for that could be.
- Quality of the Report: The report should document all important details. Use the same template as for the exercises. This time, make sure to include references to papers from which you got your problem description or algorithms. Also make sure to to include links or references to all benchmark data sets you use. Apart from that, follow the same guidelines as for the exercises. The length of the report should roughly be 10 pages (including images).

Bonus: Draw a funny/entertaining (!!) comic that either describes your graph problem or some related algorithmic aspect in an accessible fashion (ideally even for people who don't study computer science).