Combinatorial Optimization

MSc Jarosław Synak

1. The goal of this project is to prepare two heuristic algorithms, which solve a given NP-hard problem.
2. Projects are prepared in pairs.
3. Students are supposed to implement their solutions in C++ (it has to compile under g++ 6.0 compiler on *Linux*) or Python (any version), *make* can be used. Programs have to read everything from *stdin* and print to *stdout* (exact input and output formats will be given). **Any suspicious operations like accessing the filesystem (reading or writing to any files), printing unnecessary things to *stdout*, using network connection can result in failing grade.**
4. Algorithms have 5 min to print the solution, so it is advised to implement a timer. If after 5 min no solution is returned – project will be rejected.
5. There are two versions to prepare: simple (for example using greedy algorithm) and complex (with a more sophisticated solution). Both programs have to be sent to me before the deadline (**2021-01-16**) with an instruction how to compile and run them (e.g. which Python version with which plugins was used).
6. Together with the algorithms, a complete description of the methods used should be sent. Necessary parts:
   - Names of the authors and their student IDs, name of the subject, student group
   - Theoretical part, covering the methods with a short explanation why they were chosen (what are the advantages).
   - Implementation part – a description how the algorithms were implemented
   - Conclusion and sources (literature and webpages used)
7. After sending your project via email, you also have to come personally, so I can ask you about your solution and confirm that you understand how your code works.
8. Your grade will be based on how complex is your approach, quality of your description and how well you can answer my questions.
9. At the end of the semester, your solutions will be compared and the best one will be chosen.
10. Example approaches (can be combined):
    - Branch and bound
    - Genetic algorithm
    - Simulated annealing
    - Tabu search
    - Deep learning