Assignment 3: Evolutionary Submodular Optimisation

Core Body of Knowledge classification: Abstraction, Design, Teamwork concepts & issues, Data, Programming, Systems development, Analysis

Due date: 11:59pm, 2nd November 2025, **Weight:** 10% of course

1 Assignment structure and practical requirements

The assignment has to be done in **groups of 5-6 students**. Each student has to take major responsibility for parts of the assignment and collaborate with the team members. The group has to make sure that the workload is evenly distributed. Report progress and task distribution to your tutor during the workshops and obtain feedback.

Do not use generative AI for coding or text answers unless explicitly allowed (in Exercise 3). Do not copy code or text from anywhere. We are automatically running plagiarism checks and generative AI detectors. Instead, make your submission original and unique by writing everything yourself, choosing function and variable names well and documenting your code with your own comments.

You have to use the programming language Python and upload all source code, data, and documentation in one zip file by the due date.

2 Team roles and contributions

Include in a file `doc/team_contributions.txt` the name and student id of each group member and write a short paragraph for each group member about the role and contributions to the assignment. **Submission of the file `doc/team_contributions.txt` detailing the contribution of each student is mandatory to receive marks for this assignment.**

3 Assignment

In this assignment, you will work with IOHprofiler https://iohprofiler.github.io/ which is a tool to run and analyse iterative search algorithms. A general tutorial is available at https://github.com/IOHprofiler/IOHexperimenter/blob/master/example/tutorial.ipynb You will design and implement different algorithms for Submodular Optimisation and examine their performance with the use of IOHanalyzer. To get started, read the article "Benchmarking Algorithms for Submodular Optimization Problems Using IOHProfiler" available at https://ieeexplore.ieee.org/document/10254181 which describes the set up the different problems in IOHprofiler.

You can get a list of all submodular problem instances integrated via *ioh.ProblemClass.GRAPH.problems*.

We consider the following set of instances during this assignment:
2100: 'MaxCoverage2100', 2101: 'MaxCoverage2101', 2102: 'MaxCoverage2102', 2103: 'MaxCoverage2103'
2200: 'MaxInfluence2200', 2201: 'MaxInfluence2201', 2202: 'MaxInfluence2202', 2203: 'MaxInfluence2203',
2300: 'PackWhileTravel2300', 2301: 'PackWhileTravel2301', 2302: 'PackWhileTravel2302'

As an example, you can get access to the first MaxCoverage instance listed (id 2100) via f = ioh.get_problem(2100, problem_class=ioh.ProblemClass.GRAPH).

Fixed budget plots should be done using IOHAnalyzer and you are free to choice your own way of plotting trade-off results for multi-objective problems.

Exercise 1 *Single-Objective Approaches (10+10 marks)*

Each algorithm should be run on each of the instances 30 times with the budget of 10,000 fitness evaluations.

- Run RLS, (1+1) EA and your own GA (from Assignment 2, Exercise 3) on the instances. Provide for each of the problem instances a fixed budget plot showing the mean and standard deviation for the three algorithms.
- Provide an analysis of the results and a comparison of the different algorithms as part of your submission.

Exercise 2 *Multi-Objective Evolutionary Submodular Optimisation (5+10+5+5 marks)*

Each algorithm should be run on each of the instances 30 times with the budget of 10,000 fitness evaluations.

- In the given MaxCoverage and MaxInfluence problem instances each node has a cost of 1. Use this to establish the multi-objective formulation for monotone submodular problems as discussed in the lecture and implement the multi-objective fitness function.
- Implement the GSEMO algorithm using your multi-objective formulation and test it on the MaxCoverage and MaxInfluence problem instances. Add the results to the fixed budget plots of Exercise 1. Provide the plots and an analysis of the results as part of your submission.
- Test the GSEMO algorithm using your multi-objective formulation on the PackWhileTravel problem instances. Add the results to the fixed budget plots of Exercise 1. Provide the plots and an analysis of the results as part of your submission.
- Provide for each of the given problem instances a plot showing the trade-offs of the first of your 30 runs.

Exercise 3 *New Approaches for Monotone Submodular Optimisation (15+15+10 marks)*

You only need to consider the MaxCoverage and MaxInfluence instances for this exercise and can assume that fitness should be maximized subject to a uniform constraint. You need to design and implement a single-objective and a multi-objective approach You are allowed to incorporate problem specific components that make use of problem characteristics and/or properties of encountered solutions. As an example, you could remove items if you think that a current solution is infeasible as indicated by a negative fitness value in the case of MaxCoverage and MaxInfluence.

You are also allowed to use generative AI to come up with the design and implementation as long as you provide a description of the process and design as part of your submission. Each algorithm should be run on each of the instances 30 times with the budget of 10,000 fitness evaluations.

- Design and implement a population-based single-objective evolutionary algorithm for monotone submodular problems with a uniform constraint. Experiment with population size 10, 20, 50 and examine different set ups for maintaining diversity. Your algorithm should perform as best as possible. Describe your design choices and the process of coming up with your final algorithm.
- Design and implement a population-based multi-objective evolutionary algorithm for monotone submodular problems with a uniform constraint. Experiment with population size 10, 20, 50 and examine different set ups for maintaining diversity. Your algorithm should perform as best as possible. Describe your design choices and the process of coming up with your final algorithm.
- Add the results of your algorithms to the fixed budget plots from Exercise 2. Provide these plots and an analysis of the results as part of your submission.

Exercise 4 *Large Budgets (5+5+5 marks)*

You only need to consider the MaxCoverage and MaxInfluence instances for this exercise. Run all six approaches now for 100,000 fitness evaluations on each problem instance of MaxCoverage and MaxInfluence 30 times.

- Provide for each of the problem instances a fixed budget plot showing the mean and standard deviation for the six algorithms.
- Provide for each of the instances a plot showing the trade-offs of the first of your 30 runs for GSEMO.
- Provide a statement on observations in terms of performance of the algorithms and improvements achieved compared to the results for 10,000 fitness evaluations.

Exercise 5 *Sliding-Window GSEMO (Postgraduate students only) (30 marks)*

This exercise is for COMP SCI 7316 students only. PG Students have to submit for this exercise individually through MyUni (in addition to the group submission for Exercise 1-4.)

You have to provide in your own words an explanation of the results obtained in Section 3.1 and 3.2 of the article ”Fast Pareto Optimization Using Sliding Window Selection for Problems with Determinstic and Stochastic Constraints”. A copy of the article has been provided together with the assignment. As part of your submission you should:

- Discuss how SW-GSEMO (Algorithm 2) differs from the GSEMO algorithm (Algorithm 1) and how it tries to obtain better results.
- Compare the upper bounds GSEMO for the optimisation of monotone submodular functions with a uniform constraint to the one given for SW-GSEMO in Theorem 1 and discuss the reason for their differences. Summarize the main technical ideas used in the proof of Theorem 1.
- Summaries the results given in Section 3.2 and outline the main technical ideas used in the proof of Theorem 2. Explain the difference between Theorem 1 and 2.

Word limit for your submission of this exercise is 500 words.

4 Marking

Marking will be done according to the following guidelines:

- correct overall implementation – 20%
- quality of the results – 30%
- description and analysis of approaches and results - 50%

5 Submission

Make sure you have addressed every single instruction in the assignment and that your code runs correctly. Construct a file `doc/readme.txt` which outlines how to run your code. Place all implementations placed into a directory `final/code/` and all descriptions, figures and analyses into a directory `final/doc/`. Create subfolders for the different exercises. Submit a zip file including all files on MyUni (one per group).