Design Choices

Introduction

The goal was to build a population based evolutionary algorithm that could consistently maximise fitness for monotone submodular problems like MaxCoverage and MaxInfluence while staying within the uniform constraint.

I chose an evolutionary algorithm over greedy or deterministic methods because it explores the search space more flexibly and benefits from having multiple diverse solutions in the population.

Algorithm Overview

The implemented algorithm is a population-based EA that uses the following:

- Tournament selection to determine which individuals reproduce
- Crossover and bit-flip mutation to introduce variation
- Elitism to ensure the best individuals are preserved each generation
- Tracking of the best ever solution throughout the run

The algorithm continues to evolve the population until the evaluation budget of 10,000 fitness evaluations is reached.

Design Choices

1. Selection Strategy

Tournament selection was chosen for its simplicity and effective balance between exploration and exploitation.

A tournament size of 3 was used which provides moderate selection pressure. Good individuals are more likely to be chosen, but weaker ones still have a chance. This helps prevent premature convergence while steadily improving fitness over time.

2. Variation Operators

Two variation operators were used to generate new solutions.

Crossover: Combines two parent solutions to produce offspring, allowing promising parts of each solution to mix and form new combinations.

Mutation: A bit-flip mutation randomly flips bits within a solution's representation, introducing randomness and helping the population explore new areas of the search space.

These operators maintain a healthy balance between refining good solutions and exploring new ones.

3. Elitism

Elitism ensures that top performing individuals are preserved across generations, preventing the algorithm from losing the best solutions found so far.

In this implementation, the top two individuals were carried forward (elitism_size = 2). This small level of elitism provided stability and improved convergence while allowing enough population turnover for continued exploration.

4. Diversity and Exploration

Although the algorithm only optimises a single objective, maintaining diversity is still important for avoiding local optimums.

Diversity was encouraged through:

- Randomly generated initial populations
- Randomised mutation across all individuals
- Tournament selection that occasionally allows weaker individuals to reproduce

Together these factors keep the population from becoming too uniform and help sustain long-term exploration.

5. Termination and Evaluation

The algorithm runs until it reaches a total of 10,000 fitness evaluations. During evolution the best ever individual is tracked so that the final output always reflects the highest fitness achieved throughout the run.

Design Process

The development process followed an iterative approach:

- Began with a basic EA using random selection and mutation.
- Added tournament selection to improve selection pressure and consistency.
- Introduced elitism to preserve progress between generations.

- Tuned parameters such as population size, tournament size, and mutation frequency to achieve stable performance across test runs.

Conclusion

The final single-objective EA achieved consistent and stable results across both MaxCoverage and MaxInfluence instances. Overall, the design successfully balances diversity, convergence, and performance through a straightforward and reliable evolutionary framework.