# INF3490/INF4490 Exercise Solutions - Evolutionary Computing

Ole Herman S. Elgesem*

$\mathbb{P}$ marks the programming exercises, we strongly recommend using the python programming language for these. Exercises may be added/changed after publishing.

## 1 Evolution strategy(ES)

### 1.a

A common variant of evolution strategies used for (local) search is the $(1+4)$ ES. How would this differ from the $(1+1)$ ES in how the search space is explored? How does this, and $(1+\lambda)$ in general, compared to hill climbing and greedy search?

*Answer:*

The $(1 + 4)$ ES would generate four candidate solutions from the same origin before potentially changing parent instead of one. This would make for a more informed choice in which direction to move in the search space. The higher $\lambda$ is, the more information is gathered about the neighborhood of the current solution. Recall that greedy search checks out all neighbors before making a move, while hill climbing on makes one. So we would expect the $(1 + \lambda)$ ES to behave increasingly like a greedy search as we increase $\lambda$.

### 1.b

What effect does an adaptive search strategy have on optimization performance?

*Answer:*

An adaptive search strategy will, in most cases, increase the convergence rate of the search especially in the late stages. However, it does not by itself help avoid getting stuck in local optima.

### 1.c

How would it affect the search if the strategy parameters were mutated after the solution parameters instead of before?

*Answer:*

When strategy parameters are mutated first, the change in strategy has immediate effect on the new solution that is created. Thus the fitness of this solution also indirectly rates the strategy to some degree. If the strategy parameters are mutated after the solution parameters this link is weaker, and we would expect the strategies to adapt slower, if at all.

## 2 ES Implementation

### $\mathbb{P}$ 2.a

Ignoring mutation, and starting with the population $\{1, 2, 3, 4\}$, implement and run 3 generations of a $(4+8)$ ES maximizing $g(x) = x$, and observe what the end population looks like (use intermediary recombination).

*Code:* *(`inf3490-sol4-p2a.py`)*

*See **README** for complete list of authors/contributors.

### 2.b

If a $(4, 8)$ ES had been used in Problem 2.a, what would the probability of the optimal solution ($x = 4$) surviving the first generation have been?

*Answer:*

The best solution will always survive as an offspring, so it just needs to be created at least once in the first place. If parents are drawn without replacement the probability is zero - 4 can only be one of the parents. Otherwise the probability of any one offspring having 4 as both parents is $\frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$. The chance of none of the eight offspring having 4 as both parents is $(\frac{15}{16})^8 \approx 0.597$ and so the probability of the solution surviving is $1 - 0.597 = 0.403$.

### $\mathbb{P}$ 2.c

Repeat Problem 2.a with an EP with $q = 2$. How do the two algorithms compare?

*Code:* (`inf3490-sol4-p2c.py`)

## 3 Knapsack problem

In a 0-1 **knapsack problem**, how could you implement a repair mutation to transform infeasible solutions into feasible ones (i.e. make the sum of costs of the selected items go below the budget)?

*Answer:*

One way would be to randomly deselect items until the solution is under budget. One might also consider a greedy approach that iteratively deselects the items of least worth until the budget is met. One could even consider this as a smaller 0-1 knapsack problem from only the selected items and optimize that using some potentially completely different algorithm.

## Contact and Github

Corrections of grammar, language, notation or suggestions for improving this material are appreciated. E-mail me at **tjbersta@uio.no** or use **GitHub** to submit an issue or create a pull request. The **GitHub repository** contains all source code for assignments, exercises, solutions, examples etc. As many people have been involved with writing and updating the course material, they are not all listed as authors here. For a more complete list of authors and contributors see the **README**.