NAT Tutorial 1

2. The knapsack problem is as follows: given a set of weights W, and a target weight T, find a subset of W whose sum is as close to T as possible.

Example: $W = \{5, 8, 10, 23, 27, 31, 37, 41\}$, $T = 82$

-- Solve the instance of the knapsack problem given above.

Answer: 82 = 41+31+10 is one solution. There may be others.

-- Consider solving the knapsack problem using the canonical GA. How can a solution be encoded as a chromosome?

Answer: If the weights set is of size W, then each bit in a chromosome encodes whether item $w[i]$ is present or not.

-- What fitness function can be used for the knapsack problem, so that better solutions have higher fitness?

Answer: If D is the total sum of the candidate solution, $f(D) = 1/(1+|T-D|)$. Think of possible alternative functions , e.g. linear tent-shaped function or asymmetric function that punish excess more than deficiency: Solutions that overfill the knapsack might not be admissible and should be be eliminated or dicouraged.

-- Given your answer to question 2, what selection methods would be appropriate?

Answer: Rank or Tournament. Fitness proportionate might give too much emphasis to strong solutions.

3. Assume you have a lot of data points that seem to fall into clusters, e.g. the 2D positions of mushrooms in a forest. Instead of applying a typical cluster algorithm such as the k-means algorithm directly, you decide to use GA to find the center positions of the clusters. How might you use a canonical GA to solve this, and what are the problems you might run into, particularly regarding the representation?

Answer: A straight forward GA solution would encode the cluster centers (k times two real numbers if we assume k cluster centers) into a binary string and to determine the fitness of each string by the average Euclidean distance of the data points to the position of the nearest cluster center. Mutation should respect the importance of the bits that encode the centroids, but is essentially uncritical. If the algorithm is already close to a good solution then cross-over might lead to strange effects because the order of the centroids might be different for each individual (see Miikkulainen's cart-pole example in the lecture).

Practically more useful would be a combination of a standard cluster algorithm and GA. The genom could e.g. encode (with low precision) the initial positions of the cluster centers. Then, the clustering algorithm is run (this could be either as part of the fitness evaluation or as local search) and after convergence the clustering error is used to determine the fitness. If for a particular application the standard k-means does not produce good results the combination might be useful, although takes much more time (k-means runs for each individual in every generation rather than just once).

A further question would be whether it makes sense to evolve not a population of assignments of k vectors to the data,

but a population of *k* individuals each of which represent a single centroid that competes with other centroids for data. Answer: This idea underlies in a sense the standard *k*-means, but is beyond the standard GA.

4. Run through a simple GA, applying fitness proportionate selection and single-point crossover. It could be the "maximize $f(x) = x$-squared" problem from the notes. Set up a population of individuals by tossing a coin to get the initial chromosomes and use coin-tossing wherever you need to generate random numbers. Note how the average fitness, sum of individual fitness values, and maximum fitness change over the generations. You will need a calculator for this so bring a laptop or a mobile phone (or even a calculator if they still exist!). Or brush up on long multiplication and division.

Answer: **"**Experience is simply the name we give our mistakes." (Oscar Wilde)

5. What are negative side effects of crossover (and mutation)? Assuming that crossover and mutations are important for good performance, what can be done to reduce the side effects?

Answer: What is a schema? A part of code that contributes in a specific way to the fitness function. This contribution may be invisible due to the presence of other code (i.e. schemata) that compensate the effect. In a large population this might average out (but often does not, because selection introduces biasses or dependencies)

Mutation and crossover may destroy schemata, but are necessary in order to generate improvements and innovations. Mutation rate should stay within the "evolutionary window". Crossover could be restricted to subpopulations in order to maintain a diverse population. Most important is a clever representation (before starting the algorithm): Move bits belonging to the same property to enarby regions on the string.

6. Why is crossover usually applied before mutation in GA?

Answer: Theory tells that the two operators are interchangeable without effects on the performance. For some implementation it may be more practical to use this order. Also is is biologically more plausible that life starts with crossover and mutations comes later. Mutations should be independent which may not be the case if crossover is combined with selection (i.e. if selected strings are immediately paired).

7. Discuss implications of the schema theorem for the following cases (recall the definition of the fitness of a schema):
    - a single instance of a high-fitness schema
    - two different non-overlapping schemas with the same fitness
    - two partially overlapping schemas with a fitness that are both high but not the same
    - a fitness function that depends on the presence of other individuals, such as in the evolution of an ecosystem consisting of rabbits and foxes.

Answer:

- a single instance of a high-fitness schema may not necessarily be selected or may get killed by mutation or recombination, so although the expectation is high it may not survive. Also it should be discussed that "high-fitness schema" refers to the average over all individuals with that schema, if there is only one then it is not clear whether it is "high-fitness" and in the next generation it may turn out to be not that great.
- this is an unstable solution; the average in the next generation will probably introduce an advantage for one of them such that it may take over later on (but possibly slowly)
- here the decision might be faster, the better one is likely to win, but in recombination they may be recombined into a new type of a scheme (if the particular place is cut)

- here also stable (or oscillating) solutions with several surviving species are possible, this is very important in natural evolution, but will not be of further interest here.