NAT-DL: Self-study questions with answers **Set 1 (week 3)**

1. The knapsack (or: rucksack) 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: $W = \{5, 8, 10, 23, 27, 31, 37, 41\}$, $T = 82$

   a) Solve the instance of the knapsack problem given above.

   Answer: 82 = 41+31+10 is one solution. Another is 41+23+10+8. Any other solutions?

   b) 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.

   c) 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, $f(D) = 1/(1+|T\text{-}D|)$. Think of possible alternative functions if possible, e.g. linear tent-shaped function, $T\text{-}|T\text{-}D|$. See also question e)

   d) Given your answer to question b, what selection methods would be appropriate?

   Answer: Rank or Tournament. Fitness-proportionate may give too much emphasis to strong solutions, which may not be important in the present example, but the population may be dominated by individuals that contain the largest item(s), although the optimal solution may consist of many small ones.

   e) Consider also the case that the total weight of the subset must not exceed *T*. Would you need to change your approach?

   Answer: Whether solutions with D>T are acceptable depends on the context, but it would often be reasonable to say that D>T is not admissible. In this case the fitness should be $(T\text{-}D)\Theta(T\text{-}D)$, i.e. $f(D)=T\text{-}D$ if $T \geq D$ and $f(D)=0$ otherwise (Θ is the Heaviside step function). Note that this fitness function is not deceptive, although it may help to allow initially violations of admissibility in the sense that a symmetric fitness is used, while the non-exceedance enforce only in later generations.

2. **Computer exercise**: Implement a simple GA with fitness-proportionate selection, roulette-wheel sampling, N=100, p_c=0.7, p_m=0.001. As fitness, use the integer value that is obtained when considering the genome of an individual as a binary number.

   a) How does the number of generations needed to find the optimum depend on the size of the genome? This is just an exercise, not an assignment, so just try a few sizes and record the result.

   b) Compare your result with a hill-climbing algorithm on the same problem.

   c) For a fixed size of the genome (D=20) the time to find the solution depends on the parameters p_c and p_m as shown in the figure 1 below (green (x) is for random, red (+) for zero initialisation). Discuss the figure.

   d) The second figure show analogous results for a “deceptive” fitness function over a discrete search space {0,1,….,31}, with $F(x) = x^2$ for $x = 1, \ldots, 30$, and $F(0) = 961$ and $F(31) = 0$. Why are the results here independent of p_c? Why is the curve monotonous?

Answer: Check how your program, tests, parameters, problems etc. differ from solutions by other students your group. Check if you understand the results that you have obtained (this is not always possible, but for the simple problem considered here not too difficult.

There is no program for hill-climbing included: The idea is that because hill climbing finds the optimal solution in *D* steps, where *D* is the dimension of the search space, no computing is necessary. Note that hill-climbing also approaches the optimum optimally fast, because it checks by which mutation it achieves the largest progress. In other problems this may not work, though.

c) and d) are based on the figures (see below). For c), consider first the case with a zero initialisaton (red "+"s): Every "1" needed for the solution needs to be produced by a mutation in at least one individual. If there are very few mutations (one the left), a long time is needed before this the last 1 is finally found. The spread of the "1"s that are already there has happened by this time, simply be selection, even for a small recombination rates (crossover probabilities are from 0 to 1 in steps of 0.1, from bottom to top). If there are too many mutations (on the right), then it is likely that "1"s that are already there are lost from individual that have already collected a large number of "1"s. For an intermediate rate of mutations, the selection can counterbalance the loss of "1"s among the good individuals. Recombination (the top curve has the lowest recombination rate) helps to reduce the time further, as "1"s that are found by some individuals can spread quickly, and the lucky individuals that got many in a crossover, get a high fitness and will soon dominate the population.

For random initialisation (green x), a low mutation rate is not much of a problem, because the good bits are already somewhere in the population, but still need to be combined by recombination. Therefore the crossover rate has a much bigger impact here.

d) For the fully deceptive problem, it becomes clear that a fully random search is the best that can be done in this case, because any information that can be "harvested" along the way in here only misleading. Also, the more noise the better, it is not even possible to add too much noise, because there is no information (w.r.t. the optimum) that could be destroyed. Likewise, the initialisation does not matter here, as it is quickly forgotten by the accumulating noise.

3. **The travelling salesperson problem** asks to find the shortest path through a set of *N* cities given the pairwise distances. Create randomly the (x,y) positions of 20 - 50 cities, determine their distances, and then mutate (how?) a string representing the tour and evolve a tour that leads back to the starting city with the shortest distance.

   Answer: Check whether this was done at all by some students. For the GA solution, it seems from the numerics that there also a linear increase, although about 20 times slower than with hill-climbing, however already at $D>50$, the numerics becomes a problem, the fitness differences are so strong that the algorithm is more greedy and gets a bit faster. Note that this is with elitism, while without elitism the results will be qualitatively similar, but slower. Not only for the TSP it is important to realise that for large *N* the algorithm struggles to find the global optimum. Nevertheless, encoding and scaling are still interesting points.

4. **Termination**: The generational process in GA is repeated until a termination condition has been reached. Termination is needed to qualify as an algorithm (by definition). What termination criteria are suitable in GAs and similar algorithms?

   Answer:

   - A solution is found that has optimal fitness (or is sufficiently close to the optimum)

- Fitness indicates a sufficient improvement over alternative algorithms
- Fixed number of generations reached (only for safety!)
- Allocated budget (computation time/money) reached
- The diversity of the population has vanished (restart?)
- The fitness of the highest ranking solution is reaching or has reached a plateau such that successive iterations no longer produce better results (restart?
- Combinations of the above
- Note that after Termination it is reasonable to decide: Really finish or restart a variant of the GA on the same task.

5. **Natural evolution**: Recall what you know about natural evolution (DNA, genomes, natural selection etc.). What features of biological evolution are reflected in Gas, and how could GAs be improved by including additional features into the algorithm design?

   Answer: Genetic algorithms are only rough sketch of natural evolution. Most importantly, the idea of a fitness function that can be evaluated for a single individual is not meaningful in nature where the fitness of a species always depends on the fitness of other individual and other species. The logic is therefore different: In GA we use the fitness function to decide which individuals are selected for the next generation, whereas in biology fitness can be defined indirectly by the ability of an individual to produce offspring. There are many interesting observations some of which we will discuss later in the course such as

   - The possibility to improve fitness by learning and adaptation (*Baldwin effect*), which is in a sense what we try to achieve by including a hill-climbing stage into the process.
   - The occurrence of "junk"-DNA. Usually we make sure to use an efficient representation, but in GP we will mention also cases where the presence of unused parts of code in evolving programs is considered to be beneficial for the evolution.
   - The DNA coding principles: DNA codes directly amino acids, and amino acids that can have a similar function are represented by similar codes. Such considerations will usually be taken by the user of an MHO algorithm when encoding the problem to make sure that the fitness function is smooth or even monotonous over the hypercube that represents the genes. A similar principle is useful in encoding the problem in GAs: Related properties in the problem space should be encoded by nearby section of the genome.
   - Neutral mutations: Biological mutations have often no effect for the individual, which can be a result of stabilising mechanisms or error tolerant coding, but it is also possible that these mutations are occurring to produce a maximal width of the genome of the population filling thus the ecological niche as much as possible. In GA and other MHO algorithms this would correspond to methods the increase the diversity with minimal effects on fitness.

   Note that this list can be extended by many other points. This task was meant as an exercise in finding biological inspiration for design and further improvement of MHO algorithms, not in itself as learning material.

Fig. 1

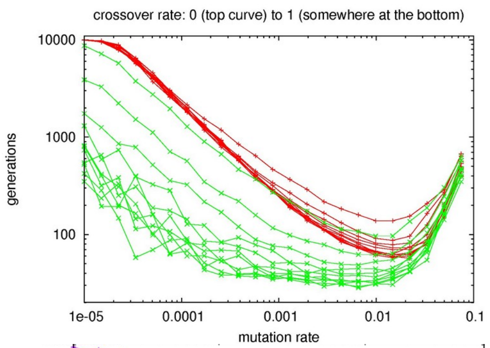
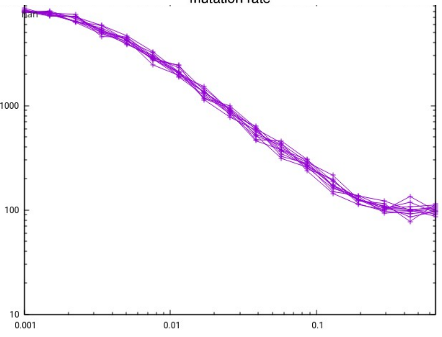


Fig. 2 (same axis titles)



1000
100
10
0.001
0.01
0.1