Natural Computing 2022

Exam* — July 26th, 2022

Please mind the following:

– Fill in your **personal information** in the fields below.

– You may use an unmarked dictionary during the exam. **No** additional tools (like calculators, e.g.) might be used.

– Fill in all your answers **directly into this exam sheet**. You may use the backside of the individual sheets or ask for additional paper. In any case, make sure to mark **clearly** which question you are answering. Do not use pens with green or red color or pencils for writing your answers. You may give your answers in both **German and English**.

– Points annotated for the individual tasks only serve as preliminary guideline.

– At the end of the exam hand in your **whole exam sheet**. Please make sure you do not undo the binder clip!

– To forfeit your exam ("entwerten"), please cross out clearly this **cover page** as well as **all pages** of the exam sheet. This way the exam will not be evaluated and will not be counted as an exam attempt.

– Please place your LMU-Card or student ID as well as photo identification ("Lichtbildausweis") clearly visible on the table next to you. Note that we need to check your data with the data you enter on this cover sheet.

Time Limit: 90 minutes

First Name:	
Last Name:	
Matriculation Number:	

Topic 1	max. 10 pts.	pts.
Topic 2	max. 12 pts.	pts.
Topic 3	max. 10 pts.	pts.
Topic 4	max. 20 pts.	pts.
Topic 5	max. 15 pts.	pts.
Topic 6	max. 10 pts.	pts.
Topic 7	max. 13 pts.	pts.
Total max. 90 pts.		pts.

---

*Corrections to the original handout are marked in red in this file.

We will reiterate definitions known from the lecture and introduce some new ones in this exam. Please mind the following notational conventions:

Notation. $\wp(X)$ denotes the power set of X. A vector $\langle x_0, ..., x_{n-1} \rangle$ with length $n \in \mathbb{N}$ can also be written as $\langle x_i \rangle_{0 \leq i \leq n-1}$ for a new iteration variable i. $\_$ denotes unspecified function arguments ($f(\_) = 0$ is the constant function that always returns zero, e.g.). We commonly write set operators ($\in, \subseteq, \wp$ etc.) for multisets where they can be used trivially and transparently use multisets instead of sets where necessary.

1 General Knowledge / Multiple Choice   10pts

For each of the following questions select **one** correct answer ('1 of n'). Every correct answer is awarded one point. Multiple answers or incorrect answers will be marked with zero points.

(a) **Which domain of science does natural computing usually *not* draw inspiration from?** (1pt)

i Physics	ii Biology	iii History	iv Chemistry

(b) Any computer program can be encoded into an instance of Conway's game of life that performs an equivalent computation. **Thus, Conway's game of life is...?** (1pt)

i Turing-complete	ii von-Neumann-complete	iii Zorn-complete	iv Gabor-complete

(c) As measured by sample efficiency, all optimization algorithms perform the same when averaged over all possible target functions. **What is this theorem called?** (1pt)

i Cheap Breakfast Theorem	ii No Free Lunch Theorem	iii Pay For Dinner Theorem	iv Secret Midnight Snack Theorem

(d) **How many functions in the λ-calculus do have a fixpoint?** (1pt)

i none	ii only $(\lambda x.(x\ x))$	iii exactly 42	iv all of them

(e) Assume a soup made of λ-expressions that react by being applied to each other. We observe that certain expressions occur way more often than others and that these expressions are able to copy themselves in a stable manner. **Thus, we call them...?** (1pt)

i bridges	ii warp engines	iii replicators	iv deflectors

(f) **Why are measures to increase the diversity within the population of an evolutionary algorithm usually employed?** (1pt)

i ensure algorithmic greediness	ii ensure quick exploitation	iii avoid early exploration	iv avoid premature convergence

(g) **What is a *necessary* part of the definition of a neural network?** (1pt)

i colors	ii weights	iii biases	iv feelings

(h) **Sufficiently large neural networks can...?** Pick the strongest true statement! (1pt)

i approximate only linear functions to a fixed degree at best	ii approximate only linear functions to an arbitrary degree	iii approximate any function to a fixed degree at best	iv approximate any function to an arbitrary degree

(i) A quantum bit (qubit) can assume a state that when measured collapses to the classical states 0 or 1, but before measuring acts to some extent like a mixture of the states 0 and 1. **This state is called...?** (1pt)

i plusinformation	ii superposition	iii hyperrelaxation	iv ubertunneling

(j) Assume that we entangled two qubits so that they can assume the combined states $(0, 0)$ and $(1, 1)$. We prepare the exact same experiment multiple times and each time measure the first qubit and notice that we receive the state 0 in 25% of experiments and 1 in 75% of the experiments. Each time, we then measure the second qubit after having measured the first. **What results do we observe for the second qubit?** (1pt)

i a state 1 in 0% of the experiments	ii a state 1 in 25% of the experiments	iii a state 1 in 75% of the experiments	iv a state 1 in 100% of the experiments

2 Game of Life   **12pts**

The following definition was given in the lecture.

> **Definition 1** (Conway's game of life). Let $G = (V, E)$ be a graph with vertices V and (undirected) edges $E \subseteq V \times V$. We define $neighborhood : V \to \wp(V)$ via $neighborhood(v) = \{w \mid (v, w) \in E\}$ as a topology for G. A state $x \in \mathcal{X}$ is a mapping of vertices to the labels $\{dead, alive\}$, i.e., the state space $\mathcal{X}$ is given via $\mathcal{X} = (V \to \{dead, alive\})$. Let x_t be a state that exists at time step $t \in \mathbb{N}$. We define $|v|_{x_t} = |\{w \mid w \in neighborhood(v) \wedge x_t(w) = alive\}|$. In the game of life, the evolution of a state x_t to its subsequent state x_{t+1} is given deterministically via
>
> $$x_{t+1}(v) = \begin{cases} dead & \text{if } |v|_{x_t} \leq 1, \\ x_t(v) & \text{if } |v|_{x_t} = 2, \\ alive & \text{if } |v|_{x_t} = 3, \\ dead & \text{if } 4 \leq |v|_{x_t}, \end{cases}$$
>
> for all $v \in V$. A tuple (G, x_S) is called an instance of the game of life for initial state $x_S \in \mathcal{X}$.
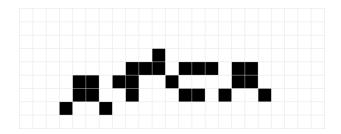
Let a *still life* be a state x_t in Conway's game of life (cf. Def. 1) so that $x_{t+1} = x_t$.

(a) **Give an example of a still life on the grid topology printed below.** (2pts)
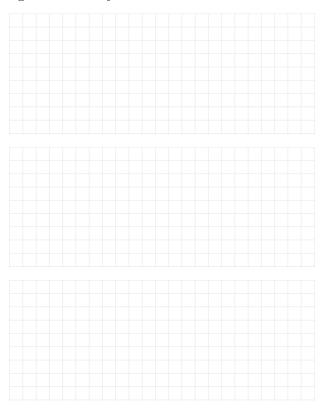
(b) **Use Def. 1 to evolve the following initial state for Conway's game of life on a grid for** 3 **steps. State if and why this pattern fits the description of an** *oscillator*, *space ship*, **or** *gun*. **What can this pattern be used for when trying to encode complex behavior like information processing in Conway's game of life?** (10pts)

You can use these grids to draw your next states.

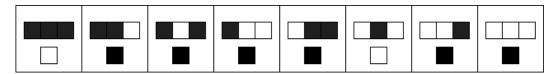# 3 Cellular Automata   10pts

An *exam cellular automaton* is defined by a function $f : \{dead, alive\}^3 \to \{dead, alive\}$ on an initial state $x \in \{dead, alive\}^n$ for some fixed $n \in \mathbb{N}$. Let $x(i)$ denote the ith value of x, i.e., $x = (x(0), ..., x(i), ..., x(n-1))$. Also let $x(-1) = x(n-1)$ and $x(n) = x(0)$. Given a state $x_t \in \{dead, alive\}^n$ at time step $t \in \mathbb{N}$ the next state $x_{t+1} \in \{dead, alive\}^n$ is given via

$$x_{t+1}(i) = f\big(x_t(i-1), x_t(i), x_t(i+1)\big).$$

(a) **What dimensionality does the *exam cellular automaton* have?** (Hint: Consider on what kind of grid it operates!) (1pt)

(b) **Define the neighborhood function *neighborhood* : $\mathbb{N} \to \wp(\mathbb{N})$ that returns all neighboring cells' indices given a cell index $i \in \mathbb{N}$.** (3pts)

(c) A cellular automaton's function f is often given as the template below, which shows the inputs to f on the top row and the respective output on the bottom row. **Give a (possibly concise) definition for f according to the template below.** (3pts)

(d) **With this function f you have just defined, evolve this initial configuration by hand for three generations. Use one row per generation.** (3pts)

4 Optimization   20pts

We give a shortened version of the definition for optimization processes from the lecture and then introduce a new optimization algorithm called *exam search*.

Definition 2 (optimization (shortened)). Let $\mathcal{X}$ be an arbitrary set called state space. Let $\mathcal{T}$ be an arbitrary set called target space and $\leq$ be a total order on $\mathcal{T}$. A total function $\tau : \mathcal{X} \to \mathcal{T}$ is called target function. Optimization or minimization is the procedure of searching for a $x \in \mathcal{X}$ so that $\tau(x)$ is optimal or minimal.
An optimization run of length $g+1$ is a sequence of states $\langle x_t \rangle_{0 \leq t \leq g}$ with $x_t \in \mathcal{X}$ for all t. Let e be a possibly randomized or non-deterministic function so that the optimization run is produced by calling e repeatedly, i.e., $x_{t+1} = e(\langle x_u \rangle_{0 \leq u \leq t}, \tau)$, where x_0 is given externally (i.e., $x_0 =_{def} 42$) or chosen randomly (i.e., $x_0 \sim \mathcal{X}$). An optimization process is a tuple $(\mathcal{X}, \mathcal{T}, \tau, e, \langle x_t \rangle_{0 \leq t \leq g})$.

Algorithm 1 (exam search). Let $\mathcal{D} = (\mathcal{X}, \mathcal{T}, \tau, e, \langle x_u \rangle_{0 \leq u \leq t})$ be an optimization process. We assume $\mathcal{X} = \mathbb{B}^4$ and $\mathcal{T} = \mathbb{N}$. Let $neighbors : \mathbb{B}^4 \to \wp(\mathbb{B}^4)$ be given by

$$neighbors(x) = \{\big(\neg x(0), x(1), x(2), x(3)\big), \big(x(0), \neg x(1), x(2), x(3)\big),$$
$$\big(x(0), x(1), \neg x(2), x(3)\big), \big(x(0), x(1), x(2), \neg x(3)\big)\}$$

where $\neg$ is the logical NOT function, i.e., $\neg 0 = 1$ and $\neg 1 = 0$. The process $\mathcal{D}$ continues via exam search iff e is given by

$$e(\langle x_u \rangle_{0 \leq u \leq t}, \tau) = x_{t+1} = \underset{x' \in neighbors(x_t) \cup \{x_t\}}{\arg\min} \tau(x').$$

(a) Let target function

$$\tau\big((x(0), x(1), x(2), x(3))\big) = 15 - \big(8 \cdot x(0) + 4 \cdot x(1) + 2 \cdot x(2) + 1 \cdot x(3)\big).$$

Let initial state $x_0 = (0, 0, 0, 0)$. **Apply exam search to this setting for 5 time steps, i.e., compute $x_1, ..., x_5$. Does exam search reach the global optimum within within these time steps? What percentage of all possible states within $\mathbb{B}^4$ has been evaluated during this optimization run? How can this number be interpreted?** (11pts)

(b) **Give a target function $\tau : \mathbb{B}^4 \to \mathbb{N}$ for which exam search *never* reaches the global optimum when starting from $x_0 = (0, 0, 0, 0)$. Briefly explain why.** (5pts)

(c) **State if exam search is elitist. Briefly explain why.** (2pts)

(d) **State if exam search is greedy. Briefly explain why.** (2pts)

5 Lambda Calculus 15pts

Church encoding is a standard encoding for various common data types to λ-expressions.

Definition 3 (Church numerals)**.** A natural number $n \in \mathbb{N}$ can be used as a λ-term by defining
$$n = (\lambda f.(\lambda x.(f^n\ x)))$$
where $(f^0\ x) = x$ and $(f^n\ x) = (f\ (f^{n-1}\ x))$.

(a) Let G be a free variable. **Reduce the λ-expression $(4\ G)$ step by step as far as possible.** Note that the shortcut f^n as used in Def. 3 is not a valid λ-expression and needs to be resolved before applying reduction! **State why Church encoding is chosen in such a way that numbers return this result when called with a function argument.** (4pts)

(b) Let $H = \lambda a.\lambda b.\lambda c.\big(b\ ((a\ b)\ c)\big)$. **Compute $(H\ 2)$ and reduce it step by step as far as possible. What mathematical function does H perform when applied to a Church numeral?** (6pts)

(c) **Reduce the following λ-expression step by step as far as possible:** (5pts)

$$\big(\big(\big((\lambda p.\lambda q.((p\ p)\ q)\ (\lambda a.\lambda b.a))\ (\lambda a.\lambda b.b)\big)\ B\big)\ A\big)$$

6 Artificial Chemistries / Soups 10pts

Recollect Def. 4, which was used in the lecture. We also introduce a special kind of artificial chemistry called *exam artificial chemistry* with Def. 5.

> **Definition 4** (artificial chemistry, soup)**.** Let $\mathcal{X}$ be a state space. Let $\mathcal{R}$ be a set of reaction rules $R \in \mathcal{R}$ where each R is a function $R : \wp(\mathcal{X}) \to \wp(\mathcal{X})$. Let $A : \wp(\mathcal{X}) \to \wp(\mathcal{X}) \times \mathcal{R}$ be a possibly randomized or non-deterministic function that returns a (multi-)set of reactants and a reaction rule to perform so that the reactants are suitable input to the reaction rule; the input to A is a (multi-)set of elements from the state space, i.e., a population. The tuple $(\mathcal{X}, \mathcal{R}, A)$ defines an artificial chemistry. A population $X \subseteq \mathcal{X}$ of an artificial chemistry is also called soup; each element $x \in X$ is also called particle. The evolution of a soup for g generations is a sequence $\langle X_0, .., X_g \rangle$ so that X_0 is given as the initial soup and
> $$X_{t+1} = (X_t \setminus X') \cup R(X')$$
> where $(X', R) = A(X_t)$. Reaction rules are usually written in the form $R = \sum_i x_i \rightharpoonup \sum_j x_j$ for $x_i, x_j \in \mathcal{X}, i, j \in \mathbb{N}$.

> **Definition 5** (exam artificial chemistry, exam soup)**.** Let $\mathcal{X}$ be a state space. Let $\mathcal{R}$ be a set of reaction rules $R \in \mathcal{R}$ so that each R is a function $R : \wp(\mathcal{X}) \to \wp(\mathcal{X})$ that is written in the form
> $$R = x_1 + ... + x_n \rightharpoonup y_1 + ... + y_m$$
> for some $m, n \in \mathbb{N}$ where $x_1, ..., x_n, y_1, ..., y_m \in \mathcal{X}$. The tuple $(\mathcal{X}, \mathcal{R})$ defines an exam artificial chemistry. A population $X \subseteq \mathcal{X}$ of an exam artificial chemistry is also called exam soup; each element $x \in X$ is also called exam particle.
> The evolution of an exam soup for g generations is a sequence $\langle X_0, .., X_g \rangle$ so that X_0 is given as the initial exam soup and
>
> $$X_{t+1} = (X_t \setminus X') \cup R(X')$$
>
> where R is a reaction rule chosen at random among those reaction rules whose required inputs exist within the exam soup and X' are said inputs (chosen at random if multiple such sets exist).

(a) **What is the main structural difference between Def. 4 and Def. 5? How does the difference affect the evolution of the respective soups?** (2pts)

(b) Consider the *exam artificial chemistry* given by $\mathcal{X} = \{\text{BEAVER}, \text{TREE}, \text{LOG}, \text{DAM}, \text{RIVER}, \text{LAKE}\}$ and $\mathcal{R} = \{R_1, .., R_5\}$ where

$$R_1 = \text{BEAVER} + \text{TREE} \rightharpoonup \text{BEAVER} + \text{LOG}$$
$$R_2 = \text{BEAVER} + \text{LOG} + \text{LOG} + \text{LOG} \rightharpoonup \text{BEAVER} + \text{DAM}$$
$$R_3 = \text{DAM} + \text{RIVER} \rightharpoonup \text{LAKE}$$
$$R_4 = \text{LAKE} + \text{BEAVER} \rightharpoonup \text{RIVER}$$
$$R_5 = \emptyset \rightharpoonup \text{TREE}$$

describes the cycle of beavers turning trees into logs (R_1), beavers building dams from multiple logs (R_2), dams turning rivers into lakes (R_3), lakes breaking the dam over time and killing a beaver in doing so (R_4), and trees regrowing over time (R_5). **Compute a possible evolution of the soup $X_0 = \{\text{BEAVER}, \text{BEAVER}, \text{LOG}, \text{LOG}, \text{RIVER}\}$ for as many time steps as necessary until a DAM appears.** (Hint: Instead of random choices, you can decide which rules to apply to make the process quicker. Use the table below.) (2pts)

time step	chosen rule	current soup
0	none	$\{\text{BEAVER}, \text{BEAVER}, \text{LOG}, \text{LOG}, \text{RIVER}\}$
1		
2		
3		
4		
5		

(c) When all rules are chosen at random in a fair manner, the soup from task (b) will eventually become unable to perform any reactions but R_5. **Briefly explain why. Give two new rules R_6 and R_6' so that any particle that occurs in R_6 (either as input or output of the rule) does not appear in R_6' (neither as input or output of the rule) and so that both $\mathcal{R} \cup \{R_6\}$ and $\mathcal{R} \cup \{R_6'\}$, i.e., the addition of R_6 *or* R_6' to $\mathcal{R}$ on their own, enable the soup from task (b) to keep on performing *at least two different reactions of* $\{R_1, ..., R_5\}$ indefinitely.** (3pts)

(d) We now assume $\mathcal{X}^\dagger = \{\text{BEAVER}, \text{TREE}, \text{LOG}, \text{DAM}, \text{RIVER}, \text{LAKE}\} \times \mathbb{R}^2$ so that the particle $(\text{BEAVER}, (1.3, 3.7))$ represents a beaver at the 2D coordinates $(1.3, 3.7)$, for example. We further assume $\mathcal{R}^\dagger = \{R^\dagger_{1,\_}, .., R^\dagger_{5,\_}\}$ so that

$$R^\dagger_{1,X,Y} = (\text{BEAVER}, X) + (\text{TREE}, Y) \rightharpoonup (\text{BEAVER}, Y) + (\text{LOG}, Y)$$

$$R^\dagger_{2,X,Y_1,Y_2,Y_3} = (\text{BEAVER}, X) + (\text{LOG}, Y_1) + (\text{LOG}, Y_2) + (\text{LOG}, Y_3) \rightharpoonup (\text{BEAVER}, X) + (\text{DAM}, X)$$

$$R^\dagger_{3,X,Y} = (\text{DAM}, X) + (\text{RIVER}, Y) \rightharpoonup (\text{LAKE}, X)$$

$$R^\dagger_{4,X,Y} = (\text{LAKE}, X) + (\text{BEAVER}, Y) \rightharpoonup (\text{RIVER}, X)$$

$$R^\dagger_{5,X} = \emptyset \rightharpoonup (\text{TREE}, X)$$

for any $X, Y, Y_1, Y_2, Y_3 \in \mathbb{R}^2$. **Give a function $A^\dagger$ so that the artificial chemistry $(\mathcal{X}^\dagger, \mathcal{R}^\dagger, A^\dagger)$ is a topological artificial chemistry.** You can (and might want to) assume a given distance function $\delta : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ in 2D space. You can also assume that the operation $X', R \sim^* X, \mathcal{R}$ randomly samples a valid pair (X', R) of reaction material $X' \subseteq X$ and reaction rule $R \in \mathcal{R}$ that can be applied to X' from a soup X and a reaction rule set $\mathcal{R}$. (3pts)

7 Evolutionary Computing   13pts

If necessary, you can recollect the definition for evolutionary algorithms as used in the lecture on this page. (Hint: Try the tasks first before reading this lengthy definition again.)

Algorithm 2 (typical evolutionary algorithm). Let $\mathcal{E} = (\mathcal{X}, \mathcal{T}, \phi, E, \langle X_u \rangle_{0 \leq u \leq t})$ be a population-based optimization process.

- Let $\sigma_N^{survivors}, \sigma_N^{parents} : \wp(\mathcal{X}) \rightarrow \wp(\mathcal{X})$ be (possibly randomized or non-deterministic) selection functions that return $N \in \mathbb{N}$ individuals, i.e., $|\sigma_N(X)| = N$ and $\sigma_N(X) \subseteq X$ for all X. They may possibly depend on $\mathcal{E}$ (most commonly ϕ). Let $\rho_N^{mutants} : \wp(\mathcal{X}) \rightarrow \wp(\mathcal{X})$ be a special selection function that, given a population X, returns a random subset $X' \subseteq X$ so that $|X'| = N$. Note that $\rho_{|X|}(X) = X$ for all X.

- Let $mutate : \mathcal{X} \rightarrow \mathcal{X}$ be a (possibly randomized or non-deterministic) single-individual mutation function. We write $mutate[\_] : \wp(\mathcal{X}) \rightarrow \wp(\mathcal{X})$ for the per-individual application of $mutate$, i.e., $mutate[X] = \{mutate(x) : x \in X\}$.

- Let $recombine : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{X}$ be a (possibly randomized or non-deterministic) two-parent recombination function. We write $recombine[\_] : \wp(\mathcal{X}) \rightarrow \wp(\mathcal{X})$ for the random-pairing application of $recombine$, i.e., $recombine[X] = \{recombine(x_1, x_2)\} \cup recombine[X \setminus \{x_1, x_2\}]$ with $x_1, x_2 \sim X$ and $recombine[\{x\}] = \emptyset$ for all x as well as $recombine[\emptyset] = \emptyset$.

- Let $migrate : \emptyset \rightarrow \mathcal{X}$ be a (possibly randomized or non-deterministic) individual creation function. We write $migrate_N : \emptyset \rightarrow \wp(\mathcal{X})$ with $N \in \mathbb{N}$ for the iterated application of $migrate$, i.e., $migrate_N() = \{migrate()\} \cup migrate_{N-1}()$ with $migrate_0 = \emptyset$.

The process $\mathcal{E}$ continues via an evolutionary algorithm if E has the form

$$E(\langle X_u \rangle_{0 \leq u \leq t}, \phi) = X_{t+1} = \sigma_{|X_t|}^{survivors}\big(X_t$$
$$\cup\ mutate[\rho_M^{mutants}(X_t)]$$
$$\cup\ recombine[\sigma_{2C}^{parents}(X_t)]$$
$$\cup\ migrate_H()\big)$$

where $M \in \mathbb{M}$ is the number of mutants produced per generation, $C \in \mathbb{N}$ is the number of children produced per generation, and $H \in \mathbb{N}$ is the number of migrants (sometimes called hypermutants) generated per generation. Especially M is often expressed as a rate for random choice within the population, i.e., $M = \lceil m \cdot |X_t| \rceil$ where $m \in \mathbb{R}$ is called mutation rate.

(a) Assume you are using an evolutionary algorithm to come up with the answers for this exam. To this end, you are searching the space of all strings but you use a spell checker to convert all words occurring in your arbitrary strings to the closest valid word in the English dictionary. To grade your exam, we assign your solution a number of *missing points* between 90 and 0. While your algorithm surely wants to minimize the missing points, your overall goal is to minimize the grade you receive (between 1.0 and 4.0). **In this scenario, describe what your target function, fitness, genotype, and phenotype are. How is the relation between your target function and your fitness function and why is it good or bad to use two separate functions here?** (4pts)

(b) We assume an evolutionary algorithm implemented in Python. The algorithm is optimizing individuals that are simply lists of four integers. Note that the Python function `random.random()` returns a random float between `0.0` and `1.0`. Consider the following snippet of Python code.

```python
import random

def unknown_operator(individual_a, individual_b):
    new_individual = []
    for i, value_a in enumerate(individual_a):
        value_b = individual_b[i]
        if random.random() < 0.5:
            new_individual.append(value_a)
        else:
            new_individual.append(value_b)
    return new_individual

print(unknown_operator([0, 0, 0, 1], [1, 0, 0, 0]))
```

Give all possible unique print outputs of the above Python code snippet. State as precisely as possible which evolutionary operator is implemented here. (Hint: It is not mutation, cf. task (c)) (3pts)

(c) We still consider an evolutionary algorithm optimizing individuals x that are lists of four integers, i.e., $x \in \mathbb{N}^4$. **In a suitable formalism (Python, pseudo-code, mathematics, ...) give a viable mutation function for these kinds of individuals. Said mutation function must be able to reach any arbitrary individual in the state space — when given any arbitrary individual as a starting point and infinite time — but should also not change the mutated individual completely.** (Hint: You can use the function `random.random()` as described in task (b) and a function `round(r)`, which rounds float r to the nearest integer.) (4pts)

(d) **What is a *memetic algorithm* and how does it deviate from the standard evolutionary algorithm? When might it be advantageous to use one?** (2pts)