Die Hausaufgaben müssen einzeln bearbeitet und abgegeben werden. Geben Sie die schriftlich zu bearbeitenden Aufgaben als pdf-Dateien ab.

**Abgabefrist ist der 19.11.2023 – 23:59 Uhr**

## Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studenten selbst sowie für die Betreuer und Korrektoren sichtbar.

Für die Hausaufgabe benötigst du **ein neues** Repository.

Dieses kann über folgenden Link erstellt werden, falls nicht bereits geschehen:

https://classroom.github.com/a/sOfjPsdO

**Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet!**

## Aufgabe 1 – Verständnis evolutionärer Algorithmen (10P)

Finden Sie ein Optimierungsproblem, bei dem Evolutionäre Algorithmen im Vergleich zu alternativen Ansätzen sicherlich sehr schlecht abschneiden würden. Erläutern Sie, warum dies Ihrer Meinung nach der Fall sein sollte.

## Aufgabe 2 – Implementierung SGA (30P)

Implementieren Sie den *Simple Genetic Algorithm* aus der Vorlesung in Python. Setzen Sie dabei die einzelnen Funktionen und evolutionären Operatoren um, wie in der Vorlesung besprochen. Also:

**Initialisierung** Gleichverteilt zufälliges Setzen jeden Bits auf 0 oder 1

**Terminationsbedingung** Termination nach einer konfigurierbaren Anzahl von Iterationen; der Algorithmus erhält dadurch einen zusätzlichen Parameter (Anzahl der gewünschten Iterationen)

**parent selection** Roulette Wheel Selection

**Crossover** 1-Punkt Crossover mit randomisierter Wahl des Crossoverpunkts, durchgeführt gemäß einer gegebenen Wahrscheinlichkeit für Crossover

**Mutation** Bitswitch gemäß einer gegebenen Mutationswahrscheinlichkeit

In ihrer Implementierung dürfen Sie Standardbibliotheken wie `NumPy` oder `random` verwenden, jedoch keine der vorhandenen Implementierungen evolutionärer Algorithmen in Python kopieren!

## Aufgabe 3 – Verwendung SGA (20P)

In dieser Aufgabe sollen Sie Ihre Implementierung des SGA nutzen, um zu versuchen, das Damenproblem zu lösen. Gehen Sie dazu in den folgenden Schritten vor.

1. Erklären Sie, wie Sie eine Positionierung von Damen auf einem $n \times n$-Schachbrett als Bitstring kodieren, und geben Sie eine Fitnessfunktion an, die Sie maximieren oder minimieren wollen (schriftliche Abgabe!). Diese Fitnessfunktion soll ihr Maximum (oder Minimum) genau für Lösungen des Damenproblems annehmen (also für Positionierungen von $n$ Damen, von denen sich keine zwei gegenseitig schlagen können). Implementieren Sie die Dekodierung eines Bitstrings in eine Positionierung von Damen und die Berechnung der Fitness. Sie dürfen die Fitness auch direkt aus dem Bitstring berechnen.
2. Überlegen Sie sich **drei** verschiedene Einstellungen für die Parameter des Simple Genetic Algorithm (also Crossoverwahrscheinlichkeit, Mutationswahrscheinlichkeit, Populationsgröße und Anzahl der Iterationen) und lassen Sie Ihren SGA pro Wahl **zehnmal** laufen, um das Damenproblem für ein $5 \times 5$-Schachbrett zu lösen. Die Anzahl der Iterationen soll jeweils mindestens 500 betragen. Dokumentieren Sie Ihre drei Wahlen von Parametern und dann für jeden Durchlauf für die letzte Population
   *a*) die beste/eine der besten Lösung/en und deren Fitness und
   *b*) die durchschnittliche Fitness der gesamten Population.

   **Bemerkung:** Wenn Sie möchten, können Sie für diese Aufgabe Ihre Implementierung des SGA auch so anpassen, dass er terminiert, sobald er eine Lösung gefunden hat. Wenn Sie das umsetzen, dokumentieren Sie in Ihrer Angabe der gewählten Parameterwerte nicht die (dann obsolete) vorher festgelegte Anzahl durchzuführender Iterationen, sondern für jeden der 30 Durchläufe die Anzahl der benötigten Iterationen bis zur Termination. Vorsicht: Die Berechnung muss nicht terminieren!
3. Bewerten Sie Ihr oben durchgeführtes Experiment:
   *a*) Beschreiben Sie (ohne Wertung!) Ihre Beobachtungen: Für welche Wahl von Parametern sehen Sie welches Verhalten des Algorithmus?

   **Beispiel:** "*Für die erste Wahl von Parametern wurde überhaupt nur in einem Durchlauf eine Lösung gefunden. Für das zweite und das dritte Setup gab es jeweils in neun von zehn Durchläufen eine Lösung. Allerdings war die durchschnittliche Fitness der Schlusspopulation im dritten Setup immer deutlich höher als die der zweiten.*"
   *b*) Ziehen Sie Schlussfolgerungen aus Ihren Beobachtungen: Welche Einstellungen waren vorteilhaft/nachteilhaft?

   **Beispiel:** "*Im zweiten und dritten Setup unterschied sich lediglich die Anzahl der Iterationen. Da dies bereits in neun von zehn Fällen zum Finden einer Lösung gereicht hat, ist die niedrigere Anzahl an Iterationen ausreichend, sofern man nur am Finden* einer *Lösung interessiert ist. Im dritten Setup wird die übrige Rechenzeit damit verbracht, die Population weiter zu optimieren (wie man an der erhöhten durchschnittlichen Fitness erkennt). Dies lohnt sich jedoch nur, wenn man daran interessiert ist, verschiedene mögliche Lösungen zu finden.*"