DHBW Mosbach
Lohrtalweg 10
74821 Mosbach
Deutschland

Realisierung einer Bibliothek mit modernen Algorithmen aus dem Kontext der Schwarmintelligenz

Studienarbeit EMIT an der Dualen Hochschule Baden-Württemberg Mosbach

Studiengang/-richtung:	B.Sc. - Angewandte Informatik
Kurs:	INF20B
Name, Vorname:	Robin Wollenschläger
Name, Vorname des wiss. Prüfenden/Betreuenden:	Prof. Dr. Carsten Müller

Inhaltsverzeichnis

1 Einleitung

1.1 Thema

Diese Arbeit befasst sich mit der Realisierung einer generalisierten Bibliothek von modernen Algorithmen aus dem Kontext der Schwarmintelligenz in Java, was insbesondere die Untersuchung der drei Algorithmen auf Ähnlichkeiten in der Basis in Hinblick auf mögliche Abstraktionen beinhaltet.

1.2 Zielsetzung

Als Ziel dieser Arbeit sollen die drei Optimierungsalgorithmen 'Grey Wolf Optimization', 'Rat Swarm Optimization' und 'Elephant Herding Optimization' in einer Java-Bibliothek realisiert werden, um zukünftig in Lehre und Forschung eingesetzt werden zu können.
Die Wirksamkeit der Implementierung soll anhand von geeigneten UnitTests anhand der Ackley-Funktion sichergestellt werden.

1.3 Abgrenzung

Diese Arbeit soll keine mathematischen Untersuchungen über das Verständnis der korrespondierenden Paper zu den geforderten Algorithmen hinaus beinhalten und auch keine weiteren Experimente als die Qualitätssicherung anhand der Ackley-Funktion abbilden.

1.4 Optimierungsalgorithmen aus dem Bereich der Schwarmintelligenz

Algorithmen aus dem Bereich der Schwarmintelligenz werden zur Optimierung von Problemen verwendet, in dem Verhaltensstrukturen aus der Natur mathematisch abgebildet und nutzbar gemacht werden.
Dabei wird versucht im Verhalten von Lebewesen Muster zu finden, mit denen ein Ziel erreicht werden kann, um somit die Zielfindung mathematischer Probleme zu optimieren. Gesucht wird dabei ein Optimum, also ein globales Minimum oder Maximum einer mehrdimensionalen mathematischen Funktion.

1 Einleitung

Der Algorithmus 'Grey Wolf Optimization' arbeitet rundenbasiert in Iterationen, wobei eine Obergrenze definiert werden kann.

2 Grey Wolf Optimization

2.1 Grauwölfe

2.1.1 Sozialverhalten

Grauwölfe (Canis lupus) leben in Rudeln mit einer festen Hierarchie, die sich in Alpha-, Beta-, Delta- und Omega-Wölfe gliedert (siehe Abbildung 2.1.1).
Pro Rudel gibt es jeweils Alpha-, Beta- und Deltawölfe. Der Rest des Rudels wird als Omegawolf klassifiziert. Diese Hierarchie ist sehr strikt und das Rudel folgt meist den Entscheidungen des Anführers (Alpha), auch wenn es selten zu demokratischen Entscheidungen kommen kann und der Alpha den anderen Mitgliedern des Rudels folgt. Der Alpha muss dabei nicht zwingend das stärkste Tier des Rudels sein, sondern der beste Anführer, der die besten Entscheidungen trifft.
Die zweite Hierarchieebene innerhalb des Rudels wird vom Betawolf eingenommen. Er hilft dem Alpha bei der Entscheidungsfindung und hat im Falle eines Dahinscheidens des Alpha die besten Chancen seinen Platz als Nachfolger und neuer Anführer des Rudels einzunehmen. Innerhalb des Rudels setzt der Beta die Anweisungen des Alpha durch und sorgt für Disziplin und Ordnung unter den niederrangigen Wölfen.
Diese niederrangigen Mitglieder sind aufgeteilt in Delta- und Omegawölfe. Der Delta muss zwar den Anweisungen von Alpha und Beta Folge leisten, dominiert aber über die Omegawölfe und fungiert in Rollen wie Scout, Wächter oder Jäger und tragen damit Sorge über die Sicherheit des Rudels und des zugehörigen Territoriums oder es handelt sich um Ältere, die zuvor den Rang eines Alpha oder Beta bekleidet haben.
Omegawölfe sind die Rangniedersten innerhalb des Rudels, aber sie sind für die soziale Struktur dennoch sehr wichtig, da es beim Verlust eines Omegas zu Kämpfen und Problemen innerhalb des Rudels kommt und die Hierarchie wieder hergestellt werden muss. Die Dominanz über die Omegas dient den anderen Wölfen zum Teil als Ventil und sorgt so dafür, dass sich innerhalb des Rudels keine Aggressionen anstauen, [3, vgl. Mirjalili 2014, S.4f].

2.1.2 Jagdverhalten

Neben dem Sozialverhalten ist das Jagdverhalten der Grauwölfe von besonderem Interesse. Dieses teilt sich nach [3, vgl. Mirjalili 2014, S.5] in folgende Phasen:

- Suchen, Jagen und Erreichen der Beute

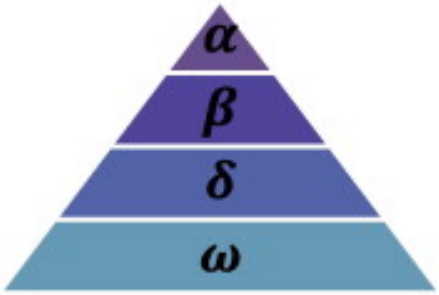

Abbildung 2.1.1: Soziale Hierarchie bei Grauwölfen [3]

- Verfolgen, Einkreisen und Beunruhigen der Beute, bis sie stehen bleibt
- Angreifen der Beute

Der Algorithmus 'Grey Wolf Optimization' stützt sich auf die mathematische Umsetzung des Jagdverhaltens und des Sozialverhaltens von Grauwölfen.

2.2 Optimierung

Im folgenden Abschnitt soll die mathematische Modellierung der Sozialen Hierarchie, der Beutesuche, deren Einkreisen und der Angriff vorgestellt werden.

2.2.1 Soziale Hierarchie

Für die Optimierung wird davon ausgegangen, dass Alpha-, Beta- und Deltawolf besten Kenntnisse über den Standort der Beute haben und sich das Rudel anhand deren Positionen ausrichtet. Dazu werden pro Runde die besten drei Lösungen ermittelt und entsprechend als Alpha, Beta und Delta klassifiziert. Alle übrigen Lösungen werden zu den Omegas gezählt und damit nicht weiter untergliedert.
Alpha, Beta und Delta führen die Jagd (Optimierung) an und alle übrigen Omegas folgen diesen. Pro Runde werden für jeden Wolf neue Positionen anhand der Positionen von Alpha, Beta und Delta berechnet und damit die Position Wölfe immer in eine bestimmte Richtung konvergiert. Durch die Anwendung von Limits kann das Rudel wieder zusammen gezogen werden, wenn sich die einzelnen Mitglieder zu weit verstreuen, [3, vgl. Mirjalili 2014, S.5]

2.2.2 Beutetier einkreisen

Das Umkreisen der Beute wird mit folgenden Formeln beschrieben:

$$\vec{D} = |\vec{C} \cdot \vec{X_p}(t) - \vec{X}(t)| \tag{2.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2.2}$$

Wobei t die momentane Iteration angibt und $\vec{A}$ und $\vec{C}$ Koeffizientenvekoren sind, $\vec{X}_p$ die Position der Beute (Prey) und $\vec{X}$ die Position eines Wolfes.
Der Vektor $\vec{D}$ (Gleichung 2.1) gibt die Richtung an, in die der Rest des Rudels konvergieren soll und wird zur Bestimmung des nächsten Positionsvektors eines Wolfes ($\vec{X}(t+1)$, Gleichung 2.2) gebraucht.
Die Vektoren $\vec{A}$ und $\vec{C}$ werden mit folgenden Formeln bestimmt:

$$\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a} \tag{2.3}$$

$$\vec{C} = 2\vec{r_2} \tag{2.4}$$

Die Komponenten von $\vec{a}$ werden mit jeder Iteration linear von 2 bis 0 verringert und $\vec{r_1}$ und $\vec{r_2}$ sind Zufallsvektoren in $[0, 1]$.

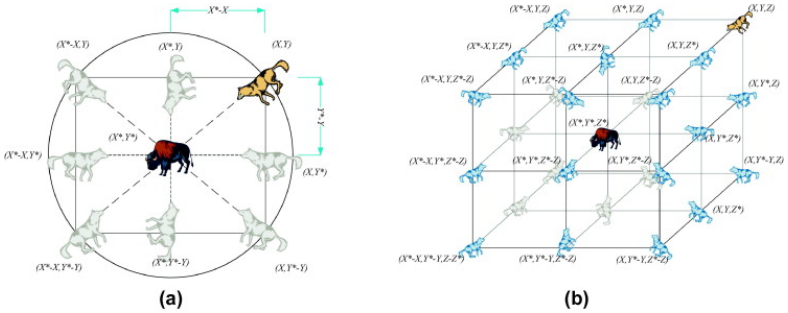

Abbildung 2.2.1: Positionsneuberechnung im GWO [3]

In Abbildung 2.2.1 wird dargestellt, welchen Effekt Gleichung 2.1 und Gleichung 2.2 haben. In Abbildung 2.2.1 (a) wird dargestellt, wie ein Wolf mit Position (X, Y) seine Position hin zur Beute auf Position (X*, Y*) verändern kann. Durch Variieren der Vektoren $\vec{A}$ und $\vec{C}$ können verschiedene Positionen rund um den Alpha eingenommen werden.
Durch die Vektoren $\vec{r_1}$ und $\vec{r_2}$ kann ein Wolf jede zufällige Position rund um die Beute, wie in Abbildung 2.2.1 dargestellt, erreichen, ([3, vgl. Mirjalili 2014, S.6]).

2.2.3 Jagd

Um das Jagdverhalten zu simulieren, wird angenommen, dass Alpha, Beta und Delta das beste Wissen über die Beute haben. Dazu werden bei jeder Iteration die Wölfe neu klassifiziert und die drei besten Lösungen als Alpha, Beta und Delta klassifiziert. Die übrigen Rudelmitglieder konvergieren bei der Neuberechnung ihrer Positionen in Richtung dieser besten Lösungen.

Somit wird der Schwarm mit jeder Iteration hin zu besten bisherigen Lösung gezogen. Dies wird mit folgenden Formeln bestimmt:

$$\vec{D_\alpha} = |\vec{C_1} \cdot \vec{X_\alpha} - \vec{X}|; \vec{D_\beta} = |\vec{C_2} \cdot \vec{X_\beta} - \vec{X}|; \vec{D_\delta} = |\vec{C_3} \cdot \vec{X_\delta} - \vec{X}| \tag{2.5}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1} \cdot (\vec{D_\alpha}); \vec{X_2} = \vec{X_\beta} - \vec{A_2} \cdot (\vec{D_\beta}); \vec{X_3} = \vec{X_\delta} - \vec{A_3} \cdot (\vec{D_\delta}); \tag{2.6}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{2.7}$$

Hier ist deutlich der Einfluss von Alpha, Beta und Delta auf die neue Position des einzelnen Wolfes zu sehen, der den Schwarm in Richtung der Beute konvergieren lässt, siehe Abbildung 2.2.2, ([3, vgl. Mirjalili 2014, S.7]).

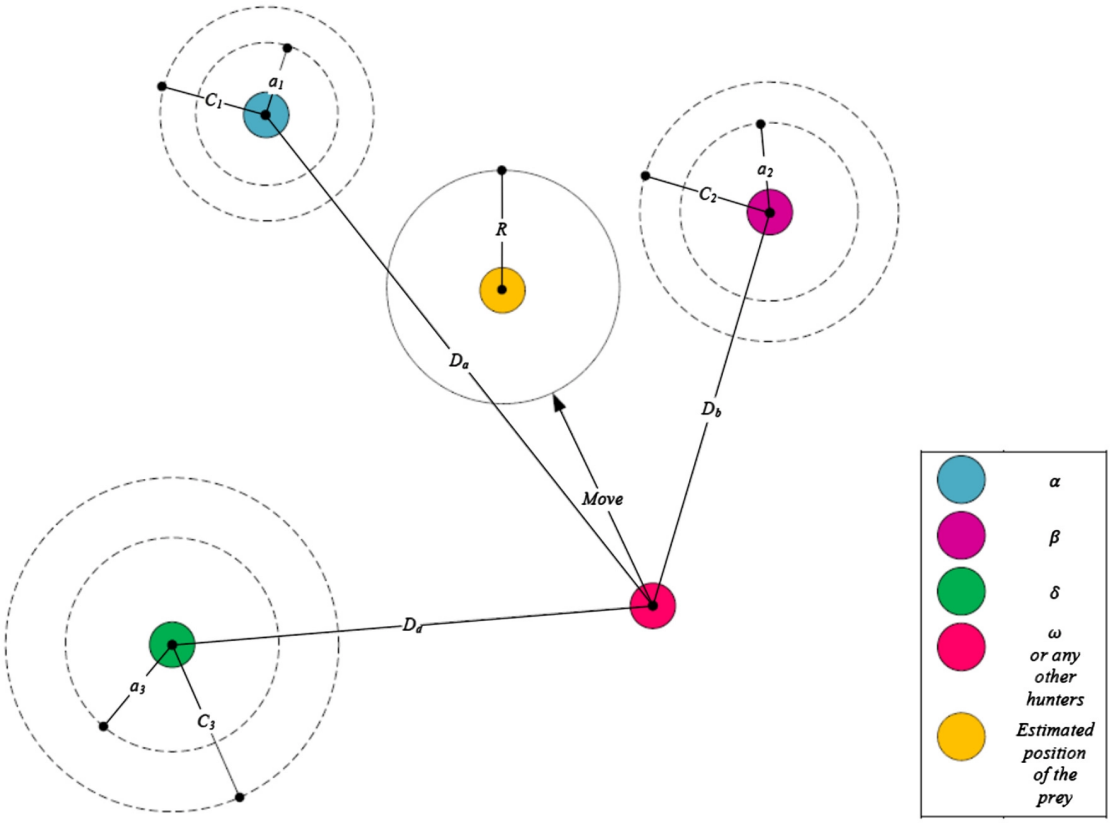

Abbildung 2.2.2: Positionsneuberechnung im GWO [3]

2.2.4 Beute angreifen

Für das Angreifen der Beute wird kein extra Schritt und auch keine weitere Formel benötigt, sondern es geschieht mittels der Verringerung des Vektors $\vec{a}$ und damit auch des Vektors $\vec{A}$, der im Interval $[-a, a]$ liegt. Mit $\vec{A}$ in $[-1, 1]$ kann die nächste Position eines Wolfes immer auf jeder Position zwischen seiner jetzigen Position und der Position der Beute sein.
Somit attackiert gezwungenermaßen jeder Wolf für $|A| < 1$ die Beute und die einzelnen Wölfe konzentrieren sich auf einen Punkt, dem gesuchten Optimum, ([3, vgl. Mirjalili 2014, S.8]).

2.2.5 Pseudocode

Folgend der Pseudocode des GWO:

Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
***while** (t < Max number of iterations)*
    ***for** each search agent*
        Update the position of the current search agent
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    t=t+1
end while
return X_α

Abbildung 2.2.3: Pseudocode GWO [3]

3 Elephant Herding Optimization

3.1 Elefanten

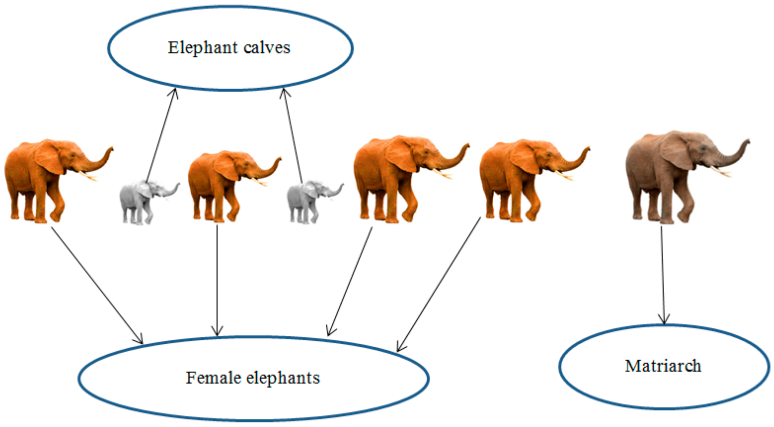

Abbildung 3.1.1: [2, Li et al, S.3]

Elefanten sind soziale Tiere mit komplexen Sozialstrukturen. Eine Elefantenherde unterteilt sich in mehrere Clans, die aus weiblichen Tieren und ihren Jungtieren bestehen. Jeder Clan wird von einem Matriarch angeführt, der oft durch die älteste zugehörige Elefantenkuh repräsentiert wird, (siehe Abbildung 3.1.1). Männliche Elefanten leben in Isolation und scheiden mit im Laufe ihres Heranwachsens aus dem Clan aus, [4, vgl. Wang et al. 2015, S.1].
Für den Algorithmus wird die Fortbewegung der Elefanten in Abhängigkeit von ihrem Clan und dem zugehörigen Matriarchen abgebildet und das Ausscheiden der männlichen Tiere aus einem Clan, [4, vgl. Wang et al, S.1].

3.2 Optimierung

3.2.1 Initialisierung

Zu Beginn muss die Herde aller Elefanten in Clans aufgeteilt werden, wobei davon ausgegangen wird, dass jeder Clan eine feste Nummer an Tieren beinhaltet, genau einen Matriarchen hat

und, dass mit jeder Generation eine feste Anzahl an männlichen Elefanten ihren Clan verlässt, [4, vgl. Wang et al, S.2].

3.2.2 Clan-Update-Operator

Jeder Clan hat einen Matriarchen, dem die Tiere folgen. Daher ist die neue Position $x_{new,ci,j}$ eines Elefanten j in Abhängigkeit von seinem Clan ci und der Position des zugehörigen Matriarchen $x_{best,ci}$ bestimmt (siehe Gleichung 3.1).

$$x_{new,ci,j} + \alpha \cdot (x_{best,ci} - x_{ci,j}) \cdot r \tag{3.1}$$

$x_{ci,j}$ stellt dabei die alte Position des Elefanten und $\alpha \in [0, 1]$ einen Skalierungsfaktor, der den Einfluss des Matriarchen ausdrückt. Für r gilt $r \in [0, 1]$.

Die Position des Matriarchen kann mit Gleichung 3.1 nicht berechnet werden und es muss Gleichung 3.2 genutzt werden.

$$x_{new,ci,j} = \beta \cdot x_{center,ci} \tag{3.2}$$

Die Position des Matriarchen wird mittels der Position des zentralen Tieres $x_{center,ci}$ innerhalb des Clans ci und $\beta \in [0, 1]$ repräsentiert einen Skalierungsfaktor. $x_{center,ci}$ kann mittels Gleichung 3.3 berechnet werden.

$$x_{center,ci,d} = \frac{1}{n_{ci}} \cdot \sum_{j=1}^{n_{ci}} x_{ci,j,d} \tag{3.3}$$

Die Position des Tieres $x_{center,ci}$ ist zusätzlich abhängig von der Dimension d mit $1 \leq d \leq D$ und der Anzahl der Tiere in einem Clan n_{ci}, [4, vgl. Wang et al, S.2].

3.2.3 Separierungs-Operator

Der Separierungsprozess beschreibt das Verlassen der männlichen Elefanten des Clans beim Heranwachsen. Die Zahl der Mitglieder bleibt dabei jedoch gleich, die Elefanten werden lediglich ersetzt. Zur Optimierung der Annäherung an das Ziel wird dabei angenommen, dass der Elefant mit der schlechtesten Position ($x_{worst,ci}$) zum Ziel den Clan verlässt (siehe Gleichung 3.4).

$$x_{worst,ci} = x_{min} + (x_{max} - x_{min} + 1) \cdot rand \tag{3.4}$$

x_{max} und x_{min} stellen die oberen bzw. unteren Grenzen der jeweiligen Dimension dar und $rand \in [0, 1]$ eine Zufallszahl, [4, vgl. Wang et al, S.2].

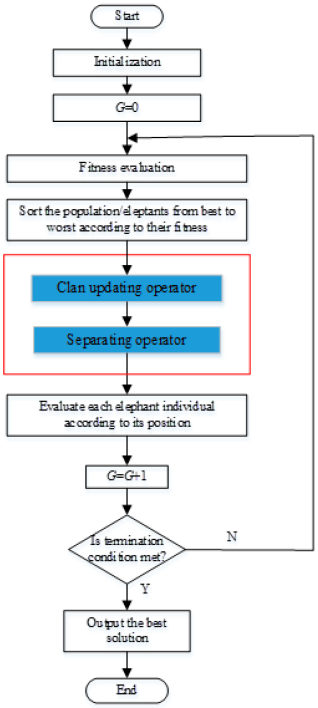

Abbildung 3.2.1: [2, Li et al, S.4]

3.2.4 Algorithmus

Aus Unterabschnitt 3.2.2 und Unterabschnitt 3.2.3 ergibt sich der Ablauf Abbildung 3.2.1, der aufzeigt, dass über die Generationen die einzelnen Elefanten immer näher hin zum gesuchten Optimum konvergieren, bis das Endkriterium erreicht ist. Dieses wird durch die maximale Anzahl an Generationen definiert.
In Abbildung 3.2.2 ist der zugehörige Pseudocode aufgeführt.
Die Elephant Herding Optimization setzt auf eine weitere Unterteilung der Sucher, durch die Aufteilung der Herde in Clans, die als Subgruppen agieren. Dadurch ergibt sich eine höhere Diversität bei der Suche, was zu einer geringeren benötigten Anzahl an Generationen führen kann.
Durch die Separierung kann die Konvergenz zum Optimum gezielt erhöht werden, allerdings sollte die Anzahl separierter Tiere nicht höher, als $\frac{n_{ci}}{2}$ liegen, da sonst auch Tiere separiert werden, die zur Zielstrebigkeit des Clans positiv beitragen.

Algorithm 1. Elephant herding optimization

(1) **Begin**
(2) **Initialization.** Set the initialize iterations $G = 1$; initialize the population P randomly; set maximum generation $MaxGen$.
(3) **While** stopping criterion is not met **do**
(4) Sort the population according to fitness of individuals.
(5) **For** all clans ci do
(6) **For** elephant j in the clan ci **do**
(7) Generate $x_{new,\,ci,j}$ and update $x_{ci,j}$ by Equation (1).
(8) **If** $x_{ci,j} = x_{best,ci}$ **then**
(9) Generate $x_{new,\,ci,j}$ and update $x_{ci,j}$ by Equation (2).
(10) **End if**
(11) **End for**
(12) **End for**
(13) **For** all clans ci **do**
(14) Replace the worst individual ci by Equation (4).
(15) **End for**
(16) Evaluate each elephant individual according to its position.
(17) $T = T + 1$.
(18) **End while**
(19) **End.**

Abbildung 3.2.2: [2, Li et al, S.5]

4 Rat Swarm Optimization

4.1 Ratten

Ratten leben in territorialen Gruppen und besitzen von Natur aus eine soziale Intelligenz. Sie verhalten sich sowohl in der Gruppe, als auch bei der Jagd sehr aggressiv, was die Motivation hinter diesem Algorithmus ausmacht.
Für den Algorithmus werden die Verhaltensweisen von Jagen und Kämpfen mit der Beute modelliert, [1, vgl. Gaurav Dhiman, S.2f].

4.2 Optimierung

4.2.1 Jagen der Beute

Ratten jagen in der Gruppe und es kommt ihr Sozialverhalten zum Tragen. Mathematisch wird dies dadurch abgebildet, dass eine Ratte als Schwarmführer fungiert und alle anderen Ratten dieser folgen. Als Schwarmführer nimmt man die beste bisherige Lösung, zu der alle anderen Mitglieder des Schwarmes hin konvergieren.
Hierzu werden folgende Formeln angenommen:

$$\vec{P} = A \cdot \vec{P_i}(x) + C \cdot (\vec{P_r}(x) - \vec{P_i}(x)) \tag{4.1}$$

$$A = R - x \cdot (\frac{R}{Max_{Iteration}}) \tag{4.2}$$

$$C = Random \in [0, 2] \tag{4.3}$$

$$R = Random \in [1, 5] \tag{4.4}$$

Der Vektor $\vec{P_i}(x)$ steht dabei für die Position einer Ratte im Schwarm und der Vektor $\vec{P_r}(x)$ bildet die Ratte mit der besten Lösung als Schwarmführer ab. x steht für den Zähler der momentanen Iteration, der bis zur Grenze $Max_Iteration$ hochgezählt wird.
R und C bilden Zufallszahlen ab, die für eine bessere Verteilung der einzelnen Ratten im Schwarm über die Dauer der Iterationen sorgen, [1, vgl. Gaurav Dhiman, S.3]

4.2.2 Kampf mit der Beute

Für den Kampf mit der Beute müsse sich die Ratten auf die Beute zu bewegen, was durch die folgende Formel ausgedrückt wird:

$$\vec{P}_i(x+1) = |\vec{P}_r(x) - \vec{P}| \tag{4.5}$$

$\vec{P}_i(x+1)$ steht dabei für die nächste Position einer einzelnen Ratte im Schwarm und richtet sich anhand der Position des Schwarmführers, der die beste bisherige Lösung abbildet, aus. Abbildung 4.2.1 zeigt die Verteilung der Ratten um die Beute. Durch Veränderung der Parameter A und C (siehe Gleichung 4.2 und Gleichung 4.3) kann die Position einer Ratte auf alle abgebildeten Positionen verändert werden.
Die optimale Lösung wird im Algorithmus der 'Rat Swarm Optimization' mit den möglichst wenigen Operatoren gespeichert, was diesen Algorithmusmus sehr schlank und leichtgewichtig bei gleichzeitig sehr großer Effizienz und Konvergenz hin zum Optimum macht, [1, vgl. Gaurav Dhiman, S.4]
Der zu diesem Algorithmus zugehörige Pseudocode ist unter Abbildung 4.2.2 zu finden.

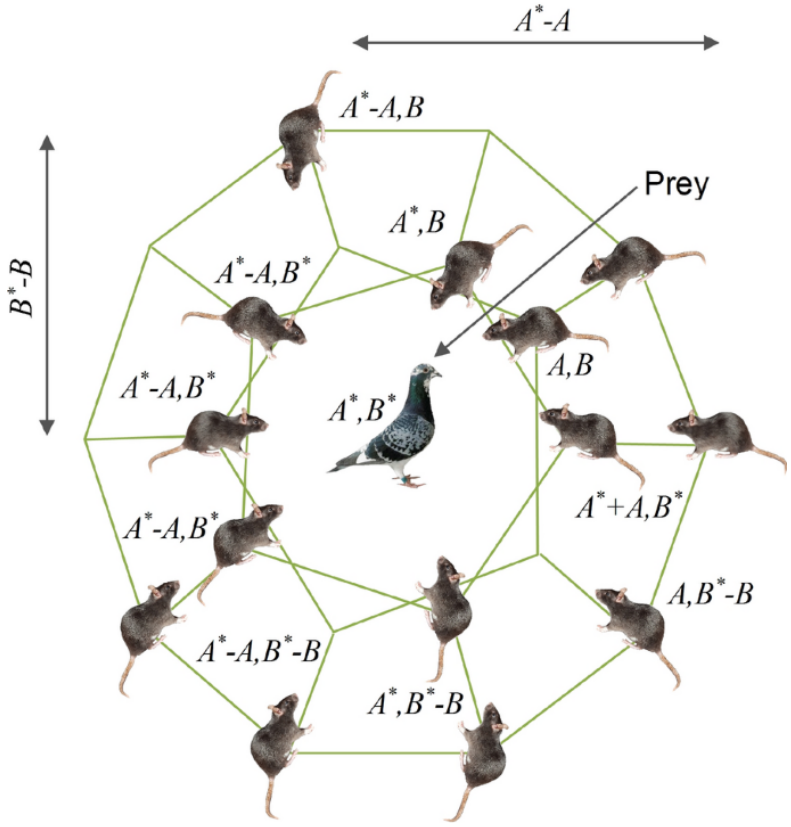

Abbildung 4.2.1: Positionsverteilung der Ratten im Schwarm um die Beute [1]

Algorithm : Rat Swarm Optimizer

Input: the rats population P_i $(i = 1, 2, \ldots, n)$
Output: the optimal search agent
procedure RSO
Initialize the parameters A, C, and R
Calculate the fitness value of each search agent
$P_r \leftarrow$ the best search agent
    while $(x < Max_{Iteration})$ **do**
        for each search agent **do**
            Update the position of current search agent by Eq. (4)
        end for
        Update parameters A, C, and R
        Check if there is any search agent which goes beyond the given search space and then adjust it
        Calculate the fitness of each search agent
        Update P_r if there is a better solution than previous optimal solution
        $x \leftarrow x + 1$
    end while
return P_r
end procedure

Abbildung 4.2.2: Pseudocode RSO [1]

5 Fazit

Durch die generalisierte Lösung auf Basis eines abstrakten Packages 'BaseSwarm', das allen realisierten Algorithmen zugrunde liegt konnte mit der Realisierung der Bibliothek zu modernen Algorithmen aus dem Kontext der Schwarmintelligenz eine sehr gute Basis geschaffen werden, um sowohl die drei gefoderten Algorithmen 'Grey Wolf Optimization', 'Rat Swarm Optimization' und 'Elephant Herding Optimization' in Lehre und Forschung einsetzen zu können, als auch auf der Basis des 'BaseSwarm' weitere Algorithmen auf diesem Fundament einfach implementieren zu können (vgl. Abbildung 5.0.1).
Auf Basis der Ackleyfunktion konnten die realisierten Algorithmen gegenübergestellt werden und im direkten Vergleich gegeneinander auf gleicher Hardware Funktion und Geschwindigkeit pro Algorithmus sichergestellt werden. Dazu sind die jeweiligen Testläufe, inkl. Zeitmessung in Abbildung 5.0.2 (Elephant Herding Optimization), Abbildung 5.0.3 (Rat Swarm Optimization) und Abbildung 5.0.4 (Grey Wolf Optimization) zu finden.

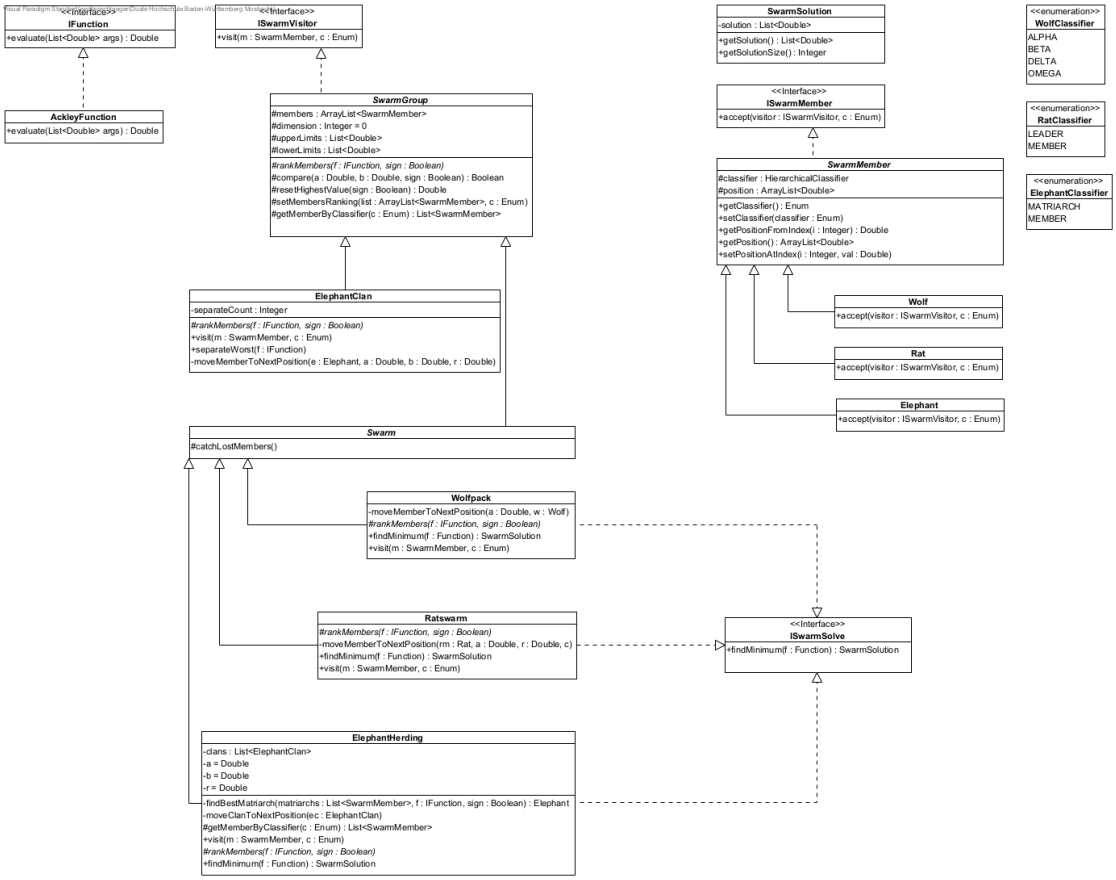

Abbildung 5.0.1: UML Bibliothek Schwarmintelligenz

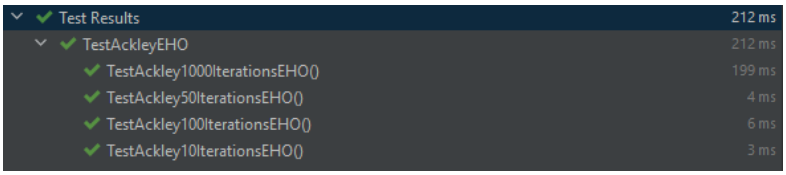

Abbildung 5.0.2: Testergebnisse Elephant Herding Optimization

Abbildung 5.0.3: Testergebnisse Rat Swarm Optimization

Abbildung 5.0.4: Testergebnisse Grey Wolf Optimization

Literatur

[1] Gaurav Dhiman u. a. „A novel algorithm for global optimization: Rat swarm optimizer“. In: *Journal of Ambient Intelligence and Humanized Computing* 12.8 (2020), S. 8457–8482. DOI: 10.1007/s12652-020-02580-0.

[2] Juan Li u. a. „Elephant herding optimization: Variants, hybrids, and applications“. In: *Mathematics* 8.9 (2020), S. 1415. DOI: 10.3390/math8091415.

[3] Seyedali Mirjalili, Seyed Mohammad Mirjalili und Andrew Lewis. „Grey Wolf Optimizer“. In: *Advances in Engineering Software* 69 (2014), S. 46–61. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2013.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0965997813001853.

[4] Gai-Ge Wang, Suash Deb und Leandro dos Coelho. „Elephant herding optimization“. In: *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)* (2015). DOI: 10.1109/iscbi.2015.8.