**Aufgabe 1.1 Landkartenfärben Problem**

Bei der Landkartenfärben Problem geht es darum, dass wir 6 verschiedene Regionen haben. Und diese Regionen sollten mit Hilfe von 3 Farben gefärbt werden. Dabei muss man drauf achten, dass die benachbarten Regionen unterschiedliche Farben haben.

Meine Vorgehensweiße war, ich habe mir überlegt ein List als Datenstruktur zu nehmen, dabei der Index waren die verschiedenen Regionen und die Werte an den Indexen waren die, dazugehörigen Farbe.

Zum Bsp. Region ist A somit in Array Position 0, B hat die Position 1 und so weiter.

Bei der Implementierung der Fitness Funktion wird überprüft, ob die benachbarten Länder auch denselben Farbe haben, wenn ja gab es Variable der die Anzahl der Einschränkungen ( violations ) auf 1 erhöht. Eine optimale Lösung ist, wenn ich bei der Fitness Funktion eine 0 als Wert erhält, weil 0 bedeutet keine Einschränkungen verletzt sind.

Bei der Implementierung der Crossover Funktion habe ich die Ein-Punkt Crossover Methode angewendet. Dabei wird ein zufälliger Punkt aus der List ausgewählt, an dem die Genome geändert werden. Dies wird bei den beiden Eltern Teilen gemacht und daraus werden 2 Kinder produziert.

Mit dem Implementieren der Mutation Funktion wollte ich die Farbe einer Lösung ändern. Die Stelle, da wo das Farbe geändert werden soll, habe ich mit Zufall ausgesucht. Die Mutation verhindert das lokale Optimum, so dass eine Population nicht irgendwo stecken bleibt.

**Aufgabe 1.2 8-Queens Problem**

Ich habe hier einen eindimensionalen Array mit der Länge N verwendet. Der Index gibt die Spalte an und der Wert gibt die Zeile an, an der eine Queen zu finden ist. Zuerst habe die beiden Element getauscht, aber dadurch gab es Problem, die ich nicht lösen könnte.

Mit der Fitness Funktion kann man herausfinden, wie viel paare von Queens sich nicht angreifen, also hierbei ein höherer Wert bedeutet eine gute Lösung.

Bei der Crossover Implementierung habe ich die Ein-Punkt Crossover Methode angewendet, wie bei der Landkartenfärben Problem. Hier geht es wieder darum, zwei verschiedene Eltern Gene miteinander kombinieren, um 2 Kinder ( Lösungen ) zu produzieren.

Mit der Mutation wollte ich hier die Zeile einer Queen zufällig ändern, um eine bessere Lösung zu finden.

**Aufgabe 1.3**

Um die beiden Probleme mit Simulated Annealing zu lösen, brauche ich einen Startzustand, eine Funktion, um die aktuellen Status zu ändern und wie bei der Metall Beispiel eine Abkühlplan, mit den schlechten Werten umzugehen.

**Aufgabe 2: Implementierung**

Ich habe für die beiden Probleme eine Lösung mit genetischen Algorithmen implementiert bekommen. Nach der Ausführung beider Algorithmen, könnte ich mir folgende Punkte feststellen.

Bei der 8 – Queens Problem hatte ich nicht immer eine perfekte Lösung bekommen und bei der Landkartenfärben Problem, habe ich jedes Mal eine richtige Lösung bekommen.

An diese Stelle muss ich noch sagen, ich weiß wirklich nicht, ob ich die beiden Algorithmen korrekt implementiert habe.

**Aufgabe 3: Anwendung**

Sein Ziel war, die Suchstrategie zu optimieren, damit man schnell und effizienter den Waldo finden kann.

Die verschiedenen Positionen von Waldo, hat er als Koordinaten dargestellt. Fitness wert wird berechnet, in dem er die zurückgelegte Strecke zwischen Waldo und die einzelnen Positionen minimiert.

In der Evolution Simulator werden verschiedene Suchstrategien als Chromosomen dargestellt, also verschiedene Lösungen.

American Fuzzy Lop ist eine Software, mit der man Softwarelücken testen kann. Er testet Eingabedaten -> bestimmt Mutation von Daten, um Lücken zu finden.

Genetische Algorithmen werden in mehreren Gebieten eingesetzt, wie z.B Finanzen, Biomedizinischen Gebieten, Maschinelles Learning…