**Particle Swarm Optimization** (**PSO**)

Vergleichbar mit dem Tierreich arbeitet die Particle Swarm Optimization wie Bienen die sich gegenseitig über gute Futterplätze informieren und somit in einem großen Gebiet die besten Plätze finden können. Die Bienen haben somit eine Optimierungsaufgabe gelöst. Genau dieses Verhalten versucht der PSO zu implementieren um somit Optimierungsprobleme möglichst effizient und schnell zu lösen.

Nachteile vom PSO liegen zum einen darin, das der Algorithmus durchaus in einem lokalen Optimum terminieren kann. Zudem führt der Algorithmus nicht für alle Probleme zum optimalsten Ziel. Der Aufwand sich auf eine Problemstellung einzurichten kann daher stark schwanken.

**Implementierung**

Insgesamt besteht der PSO aus 7 Klassen und einem Interface.

|  |  |
| --- | --- |
| PSOConstants | Beinhalltet einen Großteil der Stellschrauben des Algorithmus: Swarm size, Max Iterationen, Problem Dimension, C1, C2, W Upperbound, W Lowerbound. |
| PSODriver | Die Main Klasse mit der, der Algorithmus gestartet werden kann. |
| PSOProcess | In der PSOProcess steckt zum einen die Antworten von rabbitMQ des Weiteren wird hier der Swarm mit neuen Werten initialisiert, sowie die Fitness der letzten Iteration überprüft. Zudem beinhaltet die Klasse eine Terminierung des Algorithmus, die sich nach der Anzahl der Max Interationen und der Ergebniss Tolerance richtet. |
| PSOUtility | Hier wird bewertet wie nützlich der Wert eines Partikels ist im Vergleich zum Gesamtergebnis. |
| Velocity | Hier wird bestimmt wie schnell die Partikel sich im Ergebnisraum bewegen. |
| Location | Hier wird die Position eines Partikels im Ergebnisraum bestimmt. |
| Particle | Initialisieren von Velocity und Location der Partikel. |
| ProblemSet | In der Klasse ProblemSet werden die Wertebereiche der 17 Variablen definiert zudem wird die maximale Geschwindigkeit der Partikel in positiver sowie negativer Richtung bestimmt und für die Abbruchbedingung des Algorithmus die Toleranz (ERR\_TOLERANCE) festgesetzt.  Die Methode evaluate() weist die neuen Positionen der Partikel den 17 Variablen zu. |

**Ergebnisse**

Die Ergebnisse der einzelnen Testfunktionen mit den zugehörigen Einstellungen können der Datei PSO\_Auswertung entnommen werden.

Am schwierigsten war es dabei die Rastrigin function zu optimieren, da der Algorithmus bei einer Fehlerfunktion schnell in ein lokales Optimum läuft. Um dieses Problem zu lösen ist zum einen die Geschwindigkeit der Partikel entscheiden. Zum andern kann man mit mehreren Iterationen bei tendenziell stätig absteigenden Funktionen den Wertebereich immer weiter eingrenzen um dem Optimum so möglich nah zu kommen.

Ein großer Vorteil des PSO ist die Geschwindigkeit mit der er ein möglichst optimales Ergebnis findet. In den meisten Fällen reichten weniger als 100 Iterationen um ein gutes Ergebnis zu finden. Aufbauend auf diesem Ergebnis kann der Wertebereich sowie die Geschwindigkeit der Partikel eingrenzt werden, um dem absoluten Optimum noch näher zu kommen.

Die Implementierung des Algorithmus ist für alle vier Testfunktion, mit den entsprechend optimalsten Einstellungen, in den Ordnern Projekt1 bis Projekt4 untergebracht.

<https://gandhim.wordpress.com/2010/04/04/particle-swarm-optimization-pso-sample-code-using-java/>