**Analyse**

**Aufgabe 1.2**

1. Eine mögliche Abbruchbedingung wäre, dass weniger als 10% der Partikel kein anderes Partikel angrenzend hat. Partikel, die von Ameisen aufgenommen sind, werden nicht gezählt. Die Zeit bis dies erreicht ist hängt von Anzahl der Ameisen und Dichte der Partikel ab. Bei einer dichte von 0,07 und 100 Ameisen war dies nach 155 Steps der Fall.
2. Bei zentralisierter Initialisierung dauert es erst etwas bis sich die Ameisen verteilt haben und dadurch etwas länger bis alle effektiv arbeiten, da sofort alle Partikel in der Mitte aufgenommen werden und keine für die restlichen Ameisen übrigbleiben
3. Bei einer hohen Dichte (ab ca. 0,5) sind die Partikel schon in einem großen Cluster, daher kann nicht mehr viel verbessert werden. Bei einer geringen Dichte (ungefähr <0,15) funktioniert die Clusterung gut, jedoch sind die Ameisen relativ lange unterwegs ohne Partikel zu finden oder ohne andere Partikel zum Ablegen zu finden. Bei einer Mittleren Dichte haben die Ameisen kurze Wege aber können trotzdem Cluster bilden.
4. Wenn die Sprungweite klein oder sogar 1 ist, ist die Wahrscheinlichkeit recht groß das die Ameise, nachdem sie ein Partikel abgelegt hat, gleich ein Partikel aus dem gleichen Cluster wieder entfernt. Da dies nicht zielführend ist sollte die Sprungweite mindesten nicht 1 sein. Die Schrittweite hat wenig Einfluss auf das Ergebnis.
5. Das Clustering funktioniert auch mit nur einer Ameise, ist nur deutlich langsamer.

**Aufgabe 2.2**

**1.**

1. Bei dem vollständigen Clustering könnte man warten bis kein Partikel mehr ohne angrenzendes Partikel ist, außer es ist am Rand. Dies war bei 0,25 Partikelwahrscheinlichkeit und 200 Ameisen nach 1124 Steps der Fall.
2. Wieder ähnlich wie beim einfachen Clustering, die Ameisen nehmen zuerst alle Partikel aus der Mitte auf und dadurch entsteht ein Loch in der Mitte und die restlichen Ameisen brauchen erst etwas, bis sie etwas bewegen können. Das Loch in der Mitte bleibt bestehen.
3. Das Clustering funktioniert jetzt auch bei höheren Partikeldichten noch gut, da es verschiedene Typen von Partikeln gibt. Trotzdem funktioniert das Clustering ab ca. 0,85 nicht mehr so gut. Bei kleinen Dichten müssen die Ameisen immer noch lange umherlaufen, um Ziele zu finden.
4. Es gibt jetzt keine Sprungweiten mehr. Die Schrittweite hat wenig Einfluss auf das Ergebnis
5. Das Clustering sollte wieder auch nur mit einer Ameise funktionieren

**2.**

Wenn man die Parameter zur Berechnung der LF-Nachbarschaftsfunktion und die Skalierung der Distanzen auf das in der Simulation betrachtete Problem richtig einstellt, dann kann man mit einem einfachen Algorithmus in kurzer Zeit ein solides Clustering Ergebnis erhalten

**Designentscheidungen:**

* Models:
  + Für jede Aufgabe eigene Model-Klasse
  + Ameisenagenten werden zufällig auf dem Raster verteilt
  + Bei zentraler Initialisierung werden alle Ameisen Agenten mit Koordinaten, die der Hälfte der übergebenen Rastergröße entsprechen initialisiert
  + Die Partikel Agenten werden, unter Berücksichtigung der übergeben Wahrscheinlichkeit, bei einer Iteration über alle Zellen des Rasters platziert
  + Beide Modelle verwenden eine scheduler, welcher die Agenten in einer zufälligen Reihenfolge nacheinander aktiviert und deren Step Funktion ausführt
  + Das Modell aus Aufgabe 2 implementiert außerdem noch eine Funktion zur Berechnung der LF-Nachbarschaftsfunktion unter der Verwendung des euklidischen Distanzmaßes, sowie die Berechnung der Wahrscheinlichkeit zum Aufheben und Ablegen von Partikeln. Da diese Berechnung für jeden Agenten gleich ist, wurde sie in das Model ausgelagert.
* Agenten/ Partikel
  + Sowohl für die Ameisenagenten, als auch die Partikelagenten wurde eine Überklasse erstellt, die von mesa.Agent erbt und jeweils alle Funktionen und Parameter enthält, welche die jeweiligen Unterklassen beide verwenden
  + Für Ameisen Agenten gibt es die Klassen SimpleAntAgent und AdvancedAntAgent, die beide die vom scheduler vorrausgesetzte step Funtkion, sowie die move Funktion individuell implementieren. Beide Klassen verwenden aber dieselben Parameter bzgl. Beladung und nutzen dieselben Funktion zum Aufheben bzw. Fallenlassen.
  + Bei den Partikeln erben die Klassen SimpleParticle (Aufgabe 1), StoneParticle, NutParticle und LeafParticle (alle Aufgabe 2) von oben genannter Überklasse für Partikel
  + Die Überklasse implementiert die leere step Funktion, damit jeder Partikel diese Funktion besitzt, auch wenn sie nichts tut, denn der scheduler ruft diese Funktion auch bei den Partikeln auf, da es sich ebenfalls um Agenten handelt
  + Die Partikelklassen aus Aufgabe 2 haben jeweils 2 numerische Attribute die den Unterschied in Form und Gewicht plakativ darstellen sollen und zur Berechnung der Distanz verwendet werden.
  + In einer ersten Implementierung konnten diese Werte für Partikel derselben Klasse unterschiedliche Werte in sinnvoll eingegrenztes Intervall annehmen, doch dies führte zu extrem niedrigen Drop Wahrscheinlichkeiten, weshalb die Werte einer Eigenschaft inzwischen für alle Partikel einer Klasse identisch sind.
  + Beide Modelle verwenden die gleiche Funktion zur Darstellung der Agenten auf dem Grid, denn diese Funktion kann alle Klassen unterscheiden
  + Step- und move- Funktion bei SimpleAntAgents
    - Die step Funktion gibt den Algorithmus aus der Angabe sehr gut wieder, denn falls der Agent auf eine Stelle mit einem Partikel trifft, wird abhängig von seiner Beladung eine Entscheidung getroffen was passiert
    - Falls er beladen ist, werden alle Agenten an dieser Position nach Partikeln gefiltert und daraus wird zufällig einer ausgewählt, auch wenn theoretisch in dieser Simulation nie 2 Partikel aufeinander liegen können. Dieser Partikel wird aufgehoben, in einem Attribut des Agenten gespeichert und vom Grid entfernt
    - Falls er nicht beladen ist wird mit immer größer werdendem Radius nach einer freien Stelle in der Nähe des Agenten gesucht und sobald bei einem genügen großen Radius freie Zellen gefunden werden, wird wieder eine zufällig ausgewählt und der Partikel dort abgelegt. Die Suche nach freien Positionen bricht ab, sobald der Radius die Größe einer Grid-Seite erreicht hat
    - Bei der move Funktion wird einfach zufällige eine benachbarte Zelle ausgewählt und der Agent begibt sich dort hin
  + Step- und move- Funktion AdvancedAntAgents
    - Die Step Funktion für AdvancedAntAgents spiegelt auch die if/else Logik des Algorithmus der Angabe wider
    - Ähnlich wie oben werden an einer Stelle mit Partikel und einem Agenten ohne Beladung die Partikel herausgefiltert und einer zufällig ausgewählt jedoch nur unter Berücksichtigung der pick-Wahrscheinlichkeit aufgehoben
    - Falls der Agent beladen an einer leeren Zelle ankommt, wird die drop Funktion ebenfalls nur unter Rücksichtnahme auf die drop-Wahrscheinlichkeit ausgeführt
    - In der move Funktion wird dieses Mal darauf Wert gelegt, dass sich der Agent nicht auf eine benachbarte Zelle bewegt, in welcher sich schon ein anderer Agent (Ameise) befindet. Dazu werden die Inhalte der benachbarten Zellen überprüft und die Zellen mit einem Agenten aus der Liste der möglichen Schritte entfernt. Wenn dann noch mögliche Schritte übrig sind, wird einer zufällig ausgewählt, falls nicht bewegt sich der Agent nicht.
* Main-Funktion:
  + In der main-Funktion wird der Nutzer zu einer Eingabe, welches Modell er verwende möchte, aufgefordert. Es kann zwischen ‚simple‘ und ‚advanced‘ gewählt werden
  + Abhängig von der Eingabe wird die Simulation mit dem entsprechenden Modell gestartet