**Vorbemerkung:**
Da wir mehrere Datenstrukturen bzw. Modelle implementieren, die größtenteils unabhängig voneinander sind, haben wir schon einmal eine initiale voraussichtliche Aufteilung getroffen.

# 1 Geometrische Netzwerke (PubWeb)

PubWeb ist ein Algorithmus zur geometrischen Generierung von Netzwerken. Knoten sind dabei Punkte mit Koordinaten. Knoten, deren Abstand kleiner als eine vorgegebene Länge ist, werden miteinander verbunden. Der triviale Algorithmus (alle Paare testen) braucht quadratische Laufzeit. Zur Beschleunigung sollen nun geometrische Datenstrukturen implementiert werden. Voraussichtlich wird hierfür eine Oberklasse erstellt, um die Modularität zu erhöhen. Für die Datenstrukturen würde sich ein neuer Ordner (z.B. geometric) anbieten.

## 1.1 Gitter (Sarah)

## 1.2 kd-Baum (Simon)

Ein kd-Baum ist eine räumliche Datenstruktur. Der Raum wird hierfür rekursiv jeweils achsenparallel in zwei Teile geteilt, ein (i.A. unbalancierter) Binärbaum entsteht. Meist unterteilt man dabei die Achse mit der größten Ausdehnung. Zur Bestimmung, wo geteilt wird, gibt es mehrere Möglichkeiten. Implementiert werden der Aufbau der Datenstruktur nach dem Objekt-Median und dem räumliche Median, optional je nach Zeitreserven auch nach der Surface-Area-Heuristik (welche den durchschnittlichen Aufwand senken kann). Für die Anbindung an PubWeb wird eine Methode implementiert, die von einem gegebenen Knoten aus die Knoten unter einer bestimmten Entfernung zurückgibt.
Benutzt (und wenn nötig angepasst) werden voraussichtlich generators/PubWebGenerator.* und generators/DynamicPubWebGenerator.* für die Anbindung an PubWeb. graph/Graph.* wird für Graphoperationen verwendet. Für den kd-Baum soll eine eigene Klasse neu erstellt werden.

# 2 Generierung dynamischer Graphen

## 2.1 Watts-Strogatz (Sarah)

## 2.2 Dorogovtsev-Mendes (Sarah)

## 2.3 Forest Fire (Simon)

Das Forest-Fire-Modell ist ein randomisiertes Verfahren zur Generierung von Kleine-Welt-Netzwerken, welches also die Bedingungen des Potenzgesetzes und die der kurzen Distanz erfüllt. Die Kantenzahl wächst im Gegensatz zu manch anderen Modellen superlinear in der Knotenzahl. Immer, wenn ein Knoten neu hinzukommt, wird er mit einem zufälligen Knoten ("Botschafter") verbunden. Von diesem aus werden weitere Nachbarn

zufällig ausgewählt, die Anzahl wird in Abhängigkeit eines "Entflammbarkeitsparameter" $p$ durch eine geometrische Verteilung mit Mittelwert $\frac{p}{1-p}$ beschrieben (also wird mit Wahrscheinlichkeit $p$ ein zusätzlicher Nachbar gesucht; dies endet beim ersten Fehlschlag). Dies wird rekursiv wiederholt, wobei Knoten höchstens einmal besucht werden. Für den neu hinzugefügten Knoten werden Kanten zu allen so gefundenen Knoten hinzugefügt. Für ungerichtete Graphen ist die Wahrscheinlichkeit $r$ für Rückwärtssuchen irrelevant.
Implementiert wird der Forest-Fire-Algorithmus sowie eine Vergrößerung eines bestehenden Graphen mit dem Forest-Fire-Modell.
Der Code wird auf dem existierenden, unvollständigen Code in generators/ForestFireGenerator.* aufbauen. Für die allgemeinen Graphoperationen wird graph/Graph.* verwendet. Für den Zufall werden voraussichtlich auxiliary/Random.* sowie graph/Sampling.* benutzt. Da das Modell auch für dynamisch größer werdende Graphen verwendet werden soll, werden wohl auch die Dateien aus dem Ordner dynamics/* benötigt.