



OSTBAYERISCHE  
TECHNISCHE HOCHSCHULE  
REGENSBURG

Fakultät Informatik und Mathematik

Simulation einer intelligenten  
Abfallsammlung anhand von  
Füllstandssensoren für die Stadt  
Regensburg

BACHELORARBEIT

Zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

Verfasser: David Burger

Matrikelnummer: 2815385

Studiengang: Informatik

Betreuer: Prof. Dr. Jan Dünnweber

Externer Betreuer: Konstantin Kirsch

13.04.2018

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>4</b>
1.1 Zielsetzung und Aufbau der Arbeit . . . . .	4
<b>2 Projekt Bio-Tonne Regensburg</b>	<b>6</b>
2.1 Tourenplanung . . . . .	6
2.2 Tourbegleitung . . . . .	8
<b>3 Simulation</b>	<b>11</b>
3.1 Definition eines Behälterobjekts . . . . .	11
3.2 Füllstandanalyse . . . . .	13
3.3 Einlesen der Behälterliste . . . . .	15
3.4 Generierung einer Tour . . . . .	16
3.4.1 Kalkulation der Füllstände . . . . .	17
3.4.2 Zuweisung der Füllstände . . . . .	20
3.4.3 Aufteilung der Behälter . . . . .	23
3.5 Simulationsvorgang . . . . .	24
<b>4 Routenoptimierung</b>	<b>26</b>
4.1 ACO (Ant Colony Optimization) . . . . .	26
4.1.1 TSPLIB-Bibliothek . . . . .	27
4.1.2 Konvertierung der Behälterliste in das .tsp Format . .	28
4.1.3 Testlauf der ACO-Software . . . . .	30
4.1.4 Fehlerbehandlung . . . . .	32
4.1.5 Optimierung der Distanzberechnung . . . . .	33
4.2 arcGIS . . . . .	37
4.2.1 Visualisierung der Karte . . . . .	38
4.2.2 Planungsanalyse . . . . .	39
<b>5 Testphase</b>	<b>42</b>
5.1 Testszenario zur Untersuchung der Sensorverteilung . . . . .	42
5.2 Jahresanalyse . . . . .	45
<b>6 Santander als Vorbild</b>	<b>46</b>

<b>7 Fazit und Ausblick</b>	<b>48</b>
7.1 Verwendung der ACO-Software zur Routenoptimierung . . . . .	48
7.2 Allgemeines Fazit und Danksagung . . . . . . . . . . . . . . . . .	48
<b>Literaturverzeichnis</b>	<b>50</b>

# Abbildungsverzeichnis

1	Standortverteilung der Biotonnen in Regensburg unterteilt in Tagestouren . . . . .	7
2	Tourplan - Montag, 19.02.2018 . . . . .	8
3	Mögliche Füllstände . . . . .	14
4	CanList.xlsx - Behälterliste . . . . .	15
5	Unterteilung der Standorte in drei Bereiche ausgehend vom Stadtzentrum . . . . .	19
6	Funktionscode - Bestimmung des Fülllevels für jedes Behäl- terobjekt . . . . .	21
7	Funktionscode - Berechnung der Distanz zwischen zwei geo- grafischen Orten anhand der GPS-Koordinaten . . . . .	21
8	Skizzierte Versuchsbeschreibung [1, S. 47] - Aufteilung der Wegstrecke über beide Enden des Objekts . . . . .	27
9	.tsp Datei - Spezifikationsbereich . . . . .	29
10	.tsp Datei - Datencontainer mit vier hinterlegten Behälterob- jekten . . . . .	30
11	Methodenauszug der Funktion zur Berechnung der Distanz zwischen zwei Knoten der ACO-Software . . . . .	31
12	Tabellenauszug zum Vergleich zwischen den berechneten Di- stanzen . . . . .	33
13	Distanzbeispiel anhand von zwei Tonnen . . . . .	34
14	Erweiterung der Methode zum Lesen des .tsp Files . . . . .	35
15	Auszug aus der tabellarischen Gegenüberstellung der Distanz- werte (Längenangaben in Meter) . . . . .	36
16	arcGis Map - Zuweisung der GPS-Layer . . . . .	38
17	arcGis Map - Zuweisung des Attribute-Layers . . . . .	39
18	arcGis Analysis - Menüstruktur Routenplanung . . . . .	40
19	arcGis Map - Freitagstour optimiert . . . . .	41
20	Diagramm 1 - Einsparungspotential abhängig von der Sensor- verteilung . . . . .	43

21	Diagramm 2 - Streckendistanz abhangig von der Sensorverteilung	44
22	Diagramm 3 - Leerungen pro Jahr nach Wochentagen aufgelistet	45
23	Infrastruktur der Sensorkommunikation in Santander [5]	47

# **1 Einleitung**

## **1.1 Zielsetzung und Aufbau der Arbeit**

Zielsetzung dieser Arbeit ist es, den Prozess der Müllentsorgung der Stadt Regensburg beim Einsatz von Füllstandssensoren zu simulieren und dahingehend zu erweitern, dass anhand des Füllgrades der einzelnen Tonnen entschieden werden kann, ob eine Tonne geleert werden muss oder nicht. Durch das gezielte Auslassen leerer Tonnen entsteht eine Optimierungsmöglichkeit der aktuell noch statisch festgelegten Routen, wodurch ein erheblicher Faktor an Zeit und Kosten eingespart werden kann. Mit Hilfe einer in Java programmierten Simulation sollen Füllstände und Sensorverteilungen simuliert und somit Zeit- und Streckeneinsparungsmöglichkeiten geschätzt werden, um zu zeigen, welchen Vorteil der Einsatz von Füllstandssensoren bieten kann und ab welcher Menge an Sensoren eine Rentabilität entsteht. Anschließend werden mit Hilfe einer Software zur Lösung von Traveling-Salesman ähnlichen Problemstellungen, basierend auf dem Prinzip der Ant-Colony-Optimization, die Möglichkeiten der Routenoptimierung untersucht und mit der Geoinformationssystemsoftware arcGis verglichen.

Die Projektidee wurde beim Leiter des Amts für Abfallentsorgung in Regensburg vorgestellt und ist bereits in der Planungsphase für den Einsatz von Testsensoren. Während der Vorstellung wurde zusammen mit der Amtsleitung entschieden, das Szenario auf die Entsorgung des biologischen Abfalls von Regensburg anzuwenden.

Bei dem im Oktober 2017 eingeführten Projekt der “Bio-Tonne Regensburg“ handelt es sich um die Entsorgung von biologischen Abfällen der Bürger durch die Stadt Regensburg. Dieses Projekt wird von der Stadt Regensburg geplant und finanziert. Es wurden bereits an öffentlichen sowie privaten Standorten über 800 Tonnen aufgestellt, welche aktuell einmal pro Woche von einem Fahrzeug geleert werden. Die Biotonne bietet im Vergleich zur Restmüllentsorgung einige Vorteile für dieses Projekt. Zum Einen entsteht durch das noch anfängliche Planungs- und Entwicklungsstadium eine Möglichkeit, bereits frühzeitig in die organisatorischen Vorgänge einzutreten, diese zu ver-

ändern und zu optimieren. Des Weiteren ist die geringe Anzahl an aktuell eingesetzten Müllbehältern eine gute Voraussetzung für die Routenoptimierung, da aufgrund der Verteilung über die gesamte Stadtfläche große Abstände zwischen den einzelnen Tonnen entstehen. Das Auslassen einzelner Behälter bietet somit ein größeres Einsparungspotenzial an unnötig gefahrenen Strecken durch eine geringere Menge an eingesetzten Sensoren. Für die Restmülltonnen bildet der Umstand, dass die Entsorgung vom Bürger selbst bezahlt werden muss, ein wesentliches Problem für diese Thematik. Ein Auslassen der Leerung hätte zur Folge, dass der finanzielle Sektor dieses Verfahrens zusätzlich umgestaltet werden müsste.

Anhand dieser Gegenüberstellung wurde entschieden, das Szenario anhand der Biomüllentsorgung der Stadt Regensburg durchzuführen.

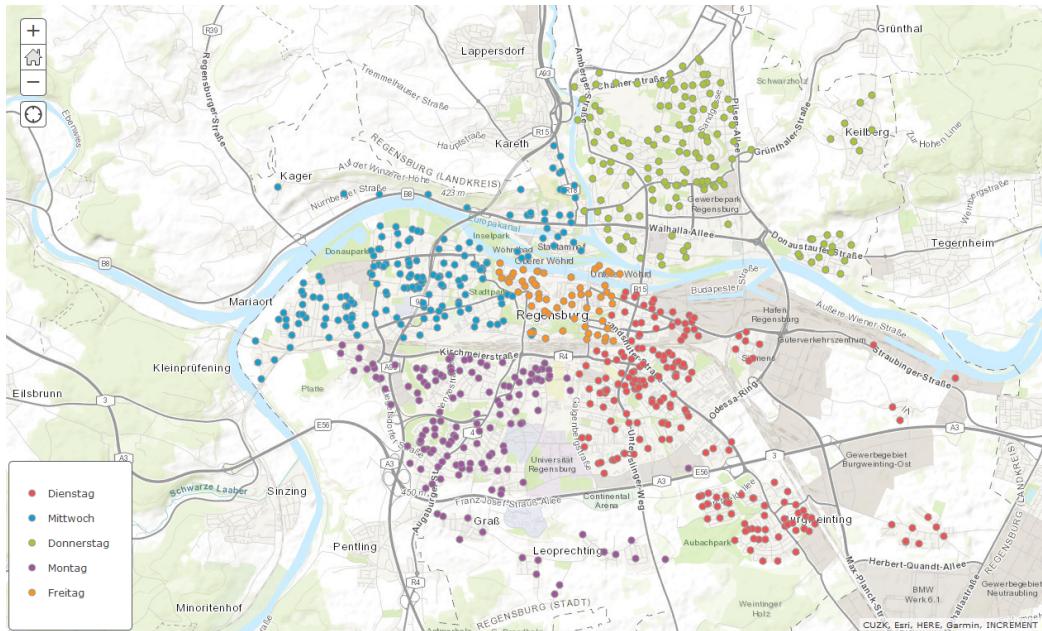
## 2 Projekt Bio-Tonne Regensburg

Das Projekt der Bio-Tonne wurde im Oktober 2017 im Abfallamt Regensburg etabliert und umfasst aktuell ca. 870 Müllbehälter im 240 Liter Format. Bei den Behältern handelt es sich ausschließlich um 240 Liter Tonnen, welche sich über das ganze Stadtgebiet verteilen. Die Standorte teilen sich auf in öffentliche und private Standorte. Öffentlich bedeutet, die Tonne steht an öffentlichen Plätzen, zum Beispiel direkt an einer Straße. Ein privater Standort bedeutet, dass die Tonne in einem Wohnkomplex mit eigener Hausverwaltung steht. Diese Unterscheidung dient lediglich organisatorischen Zwecken, auf die Abholung und Leerung der Tonnen hat das keine direkte Auswirkung. Die Stadt kümmert sich bei diesem Entsorgungssystem um alle Faktoren. Das bedeutet, die Stadt ist zuständig für die Organisation, die Routenplanung, die Abholung und Leerung der Tonnen sowie für die Entsorgung des biologischen Abfalls. Außerdem wird das ganze System von der Stadt finanziert. Zuständig für die Leerung der Tonnen sind aktuell zwei Müllwerker sowie ein Fahrer in Kombination mit einem Fahrzeug. Die Anzahl der Tonnen soll in den nächsten Monaten auf ungefähr 2100 Stück steigen. Der Prozess der Vergrößerung der Anzahl der Tonnen erfordert einen großen Planungsaufwand, da für jede neue Tonne ein Standort bestimmt werden muss. Außerdem muss neues Personal zur Verfügung gestellt werden, da die dreifache Menge an Tonnen von der bisher zur Verfügung stehenden Anzahl an Mitarbeitern nicht gestemmt werden kann. Diese Problematik würde durch den Einsatz der Sensoren verringert werden, da durch die entstehenden Möglichkeiten zur Routenoptimierung das Personal entlastet werden kann. Außerdem lässt sich durch die Sensoren bestimmen, in welchen Bereichen der Einsatz neuer Tonnen besonders effizient wäre, da anhand der gemessenen Füllstände Gebiete ausgemacht werden können, in welchen eine Überfüllung der Tonnen häufig auftritt.

### 2.1 Tourenplanung

Das Entsorgungssystem unterteilt das Stadtgebiet in seine vier Himmelsrichtungen und das Stadtzentrum. Dazu wird jeweils eine Tour an einem Werktag

von Montag bis Freitag für ein Gebiet gefahren.



**Abbildung 1:** Standortverteilung der Biotonnen in Regensburg unterteilt in Tagesstouren

Abbildung 1 zeigt die Aufteilung der Routen anhand der Wochentage. Für jeden Tag ist eine individuelle, aber statische Route vorgesehen. Diese wird nur an einzelnen Posten geändert, sollten Standorte wechseln oder neue Behälter hinzukommen. Solch eine Änderung kann entweder von den Müllwerkern, die diese Tonne leeren, oder von Anwohnern in der Nähe vorgeschlagen werden. Ein Grund hierfür könnte zum Beispiel sein, dass der Standort den Anwohner in irgendeiner Form behindert. Der Änderungsvorschlag kann im Amt für Abfallentsorgung eingebracht werden. Dieser wird dort von dem zuständigen Personal geprüft und gegebenenfalls durchgesetzt. Die Tourpläne liegen in Tabellenstruktur vor und werden in ausgedruckter Form, wie die folgende Abbildung 2 zeigt, an die Arbeiter verteilt.

Abfuhrplan Biotonnen Montag 19.02.18					
Nr.	Ö / P	Standort	Anzahl	Bemerkungen	Nr.
25	Ö	Markomannenstr. / Recyclinghof	1		
139	Ö	Anfang Islinger Weg	1		
16	Ö	Stadlerstr. 5 gg.	1		
15	Ö	Bushaltestelle bei Rauberstr. 69	1		

**Abbildung 2:** Tourplan - Montag, 19.02.2018

## 2.2 Tourbegleitung

Für die Umsetzung eines möglichst realistischen Testszenarios wurde für die Schätzung der in der Simulation zu beachtenden Parameter eine Tour der Bio-Müllwerker Regensburg mit Hilfe einer Smartphone Applikation aufgezeichnet. Die Applikation verfolgt über das GPS-Signal des Smartphones die Fortbewegung und erstellt davon ein Bewegungsprofil, welches nach der Aufzeichnung analysiert werden kann. Dabei kann zunächst die zurückgelegte Strecke in Abhängigkeit der Zeit anhand einer Straßenkarte untersucht werden, außerdem sind allgemeine Informationen über jede Aufzeichnung abrufbar. Beispiele hierfür sind die Gesamtdauer der Tour und die Fortbewegungsgeschwindigkeit zu einem bestimmten Zeitpunkt.

Für die Tourbegleitung wurde die Freitagstour gewählt, wie sie jeden Freitag einmal in der Woche gefahren wird. Die Tour enthält 60 Biomülltonnen und verläuft im Stadtzentrum von Regensburg (Abbildung 1). Diese Tour enthält mit Abstand am wenigsten zu leerende Tonnen, da die Lage im Stadtzentrum die Anfahrt der Tonnen aufgrund der Verkehrslage sowie durch schwierig zu befahrende Stadtteile wesentlich verlängert. Vorlage für die Fahrt ist ein vorgegebener Fahrplan, wie er in Abbildung 2 gezeigt wird. Dieser Plan wird vor Beginn der Tour an den Fahrer ausgehändigt. Der Fahrer fährt die Strecke ohne technische Hilfe durch Navigationssysteme oder Ähnlichem nach

Gedächtnis und anhand der Liste. Dieses Wissen bedeutet für die Routenoptimierung, dass der Fahrer neu konzipierte Tourpläne unabhängig von den zu leerenden Tonnen genauso gut abfahren kann, wie die aktuell gegebenen Abfuhrpläne, da der Standort auf dem Fahrplan dem Fahrer genügt, um die Tonne anzufahren.

Start einer jeden Fahrt ist das Abfallamt Regensburg in der Markomannenstraße. Die Gesamtdauer der Tour beträgt zwei Stunden und 20 Minuten. Aufgrund der Pausenzeit, die die Müllwerker nach der Fahrt einlegen, kann bestimmt werden, dass diese Dauer der Gesamtfahrt ein durchschnittlicher Normalwert ist. Durch Festhalten der einzelnen Leerzeiten jeder Tonne ergibt sich eine durchschnittliche Leerzeit von 36 Sekunden. Die Leerzeit einer Tonne ist durch die Zeitspanne definiert, in der das Fahrzeug für die Leerung dieser Tonne steht. Die einzelnen Leerzeiten der Tonnen weichen teilweise stark voneinander ab. Die kürzeste gemessene Leerzeit beträgt 13 Sekunden. Diese kurze Zeitdauer entsteht, wenn die Tonne direkt an der Straße positioniert ist. Dadurch muss der Müllwerker lediglich vom Fahrzeug absteigen, die Tonne in die Leerungsvorrichtung einhängen, die Leerung abwarten und dann die Tonne wieder an den Platz zurückstellen. Das Kippen durch die Leerungsvorrichtung am Fahrzeug dauert circa fünf Sekunden. Die längste gemessene Leerzeit beträgt 133 Sekunden. Diese lange Zeitdauer entsteht dadurch, dass die Tonne in einer Wohnanlage positioniert und zusätzlich auch noch überfüllt ist. Der hohe Flüssigkeitsgehalt des biologischen Abfalls hebt das Gewicht einer Tonne sehr stark an, was bei einem Fassungsvermögen von 240 Liter ein sehr hohes Gewicht bedeuten kann. Dadurch verzögert sich die Beförderung der Tonne zum Fahrzeug um eine beträchtliche Zeitdauer. Die Gesamtdistanz der Tour beträgt 25,7 Kilometer. Das Ende einer Tour wird durch die Leerung der letzten Tonne definiert, da nach jeder Tour direkt im Anschluss zur Müllentsorgungsstation gefahren wird. Dieser Teil der Strecke ist statisch und kann nicht optimiert werden, da das Müllfahrzeug nach jeder Leerungsfahrt unabhängig vom Inhalt geleert werden muss. Durch Kontrollieren und Festhalten der Füllgrade der Tonnen lässt sich die Menge an Behältern, welche aufgrund einer Füllhöhe von weniger als 50% nicht hätten geleert werden müssen, auf ungefähr 30% abrunden. Dieser Wert steht

repräsentativ für diese Tour, da die Befragung der Müllwerker ergibt, dass die Leerung der Tonnen meistens ähnlich ausfällt und bei dieser Fahrt keine ausschlaggebende Abweichung zu den anderen Freitagsfahrten erkannt wurde.

Dieser Wert bestätigt die Vermutung, dass Einsparungspotential für das System der Abholung durch den Einsatz von Sensoren besteht. Die Aufzeichnungen dieser Tourbegleitung dienen als Grundlage für die folgende Simulation.

## 3 Simulation

Zielsetzung der Simulation ist es, eine Tour anhand einer ausgewählten Liste an Behältern simulieren zu können. Das bedeutet, die Behälterliste wird eingelesen und verarbeitet. Für jedes darin enthaltene Behälterobjekt wird ein Füllstand aus einer Kombination an Zufall und einer individuell berechneten Wahrscheinlichkeit bestimmt. Das Verfahren zur Ermittlung der Füllstände wird im weiteren Verlauf noch genau erklärt. Die zu simulierende Tour erzeugt zwei Ausgabedateien, die jeweils ein bestimmtes Format besitzen, um für weitere Untersuchungen der optimalen Route mit verschiedenen Softwaretypen verwendet werden zu können. Diese Softwaretypen werden im Kapitel 4 Routenoptimierung genau erläutert. Des Weiteren erzeugt die Simulation zusätzlich eine Schätzung an möglichen Einsparungen im Bereich der Leerungen und Leerzeiten. Diese Schätzung wird in einer dritten Datei als Excel-Tabelle erzeugt.

Diese Ergebnisse sollen zeigen, welches Einsparungspotential durch den Einsatz von Füllstandssensoren entstehen kann. Außerdem soll anhand der Ergebnisse untersucht werden können, ab welcher Anzahl an Sensoren eine Rentabilität entsteht und wie die Sensoren bestenfalls verteilt werden sollten.

Implementiert ist die Simulation in der Programmiersprache Java, um im weiteren Verlauf des Projekts für die Simulation ein eigenes Web-Portal zu entwickeln, wofür Java eine einfache Portierungsmöglichkeit bietet.

### 3.1 Definition eines Behälterobjekts

Grundlage einer Tour sind die zu leerenden Behälterobjekte. Ein Behälterobjekt entspricht einer klassischen Mülltonne im 240-Liter Format. Diese werden in der Simulation durch folgende Attribute genauer beschrieben:

- **Can Nr. - Behälternummer**

Diese bezeichnet eine Biomülltonne (Behälter) eindeutig. Um das Szenario der Stadt Regensburg realistisch darzustellen, werden hierfür die Behälternummern der Stadt verwendet. Problem hierbei ist, dass die Stadt Regensburg die Biomülltonnen in privat und öffentlich aufteilt

und das bei der Behälternummer unterscheidet. Das bedeutet, dass zwei gleiche Behälternummern existieren können, wenn die Tonnen jeweils privat und öffentlich sind. Um dieses Problem in der Simulation zu umgehen, wird die Behälternummer als String definiert und bei jeder privaten Tonne ein “PT“ als Kennzeichnung an den String angefügt. Somit ist jede Mülltonne anhand der Behälternummer eindeutig definiert.

- **Location - Standort**

Der Standort der Mülltonne besteht aus Straße und Hausnummer und wird von der Stadt Regensburg festgelegt. Die Standorte wurden von der Abfallamtsleitung in Kombination mit Zuständigen der Verwaltung der Stadt Regensburg festgelegt. Dieser Standort ist fix und darf nicht verändert werden. Sollte sich eine Tonne zum Leerzeitpunkt nicht an diesem Ort befinden, wird diese ausgelassen. Dieser Fall tritt auf, wenn Bürger die Tonnen an einen anderen Platz schieben. Dieser Fall tritt aber nur sehr selten auf. In der Simulation muss keine genaue Aufteilung in Straße und Hausnummer vorgenommen werden, da alle Berechnungen und Kalkulationen des Standorts über die folgenden GPS-Daten vorgenommen werden. Der Standort dient nur zur Orientierung und als Beschreibung der Tonne.

- **GPS-Daten**

Die GPS-Daten wurden für jede Tonne aus der Standortkarte der Stadt Regensburg ausgelesen und sind aufgeteilt in Latitude °N und Longitudo °E und werden im Standardformat dd.ddddd° gespeichert. Um dieses Format in der Behältertabelle in Excel richtig darzustellen, wird ein benutzerdefiniertes Format mit der Struktur ###““###### erzeugt.

- **Emptying time in seconds - Leerzeit in Sekunden**

Für jede Mülltonne wird ein individueller Wert für die Leerzeit in Sekunden hinterlegt. Dieser Wert entspricht der in der Tourbegleitung gemessenen Leerzeit der einzelnen Tonnen. Dadurch kann bei der Berechnung der Einsparung an Zeit für jede Tour eine Zeiteinschätzung

abhängig von den zu leerenden Tonnen errechnet werden. Für Tonnen, bei denen keine individuelle Leerzeit bekannt ist, wird die durchschnittliche Leerzeit aller Tonnen der Tourbegleitung verwendet.

- **Empty filllevel probability - Wahrscheinlichkeit für einen leeren Behälter**

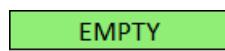
Dieser Wert bestimmt die Wahrscheinlichkeit, ob ein Behälter zur Leerzeit leer ist. Diese Wahrscheinlichkeit wird in der Simulation zur Bestimmung des Füllgrades verwendet. Bestimmt wurde der Wert durch Inspektion der einzelnen Füllgrade der Tonnen während der Tourbegleitung. Die Berechnung und weitere Verwendung dieser Wahrscheinlichkeit wird im Kapitel 4.5.1 Generierung der Füllstände erklärt.

- **Sensor**

Eine bewusste Verteilung der Sensoren auf ausgewählte Tonnen ist gegenüber einer zufälligen Verteilung sinnvoll, da auch der Standort der Tonne Einfluss auf den Füllgrad haben kann. Hierfür gibt es mehrere Einflussfaktoren welche im Kapitel 3.4.1 Kalkulation der Füllstände beschrieben werden. Deswegen wird die Sensorverteilung nicht zufällig gewählt, sondern vom Benutzer in der Behälterliste der Tonnen mit angegeben.

## 3.2 Füllstandanalyse

Die wichtigste Entscheidung in diesem Szenario ist der Leerungsbedarf der Tonne, welcher durch den Füllstand entschieden wird. Dieser wird von den meisten Sensoren als prozentualer Anteil gemessen. Ein Beispiel hierfür ist der Ultraschallsensor UMF303U035 der Firma Wenglor [2]. Da der genaue Füllgrad der Tonnen in diesem Szenario allerdings keine Rolle spielt, wird in der Simulation lediglich zwischen zwei Zuständen unterschieden:



weniger als zur Hälfte gefüllt  
-> kein Leerungsbedarf



mindestens bis zur Hälfte gefüllt  
-> Leerungsbedarf besteht

**Abbildung 3:**  
Mögliche Füllstände

Der Füllstand “EMPTY” (leer), in Abbildung 3 zu sehen, in Kombination mit Sensorausstattung der Tonne definiert das Szenario einer nicht zu leeren Tonnen. Ist die Tonne voll, muss diese zwingend geleert werden. Alle Tonnen ohne Sensor müssen sowieso geleert werden, unabhängig vom Füllstand, da dieser bis zur Leerung nicht bekannt ist.

Um ein realistisches und nicht auf zu vielen Zufällen basierendes Szenario herzustellen, müssen die Füllstände der Tonnen möglichst realgetreu simuliert werden. Durch Befragung der Müllwerker bei der Tourbegleitung, die tagtäglich die einzelnen Tonnen leeren, stellt sich heraus, dass der Füllgrad einiger Tonnen bereits vor der Leerung relativ gut eingeschätzt werden kann. Das hat mehrere Gründe. Die meisten Tonnen der Freitagstour, die bei der Begleitung inspiziert wurden und überfüllt oder bis zum Rand gefüllt waren, sind das regelmäßig. Oftmals wissen die Müllwerker bereits vorher, dass die Tonne höchstwahrscheinlich wieder ähnlich voll ist. Das bedeutet, dass man davon ausgehen kann, dass die Füllstände der Tonnen, die bei der Begleitung aufgezeichnet wurden, keine wirklichen Zufallsfüllungen sind, sondern diese Werte entsprechen oft dem Regelfall. Das stellt den ersten Faktor bei der Kalkulation der Füllstände in der Simulation dar, der berücksichtigt werden muss. Einen weiteren Faktor für die Höhe des Füllgrades bestimmt der Standort der Tonne. Tonnen, die im Stadtzentrum positioniert sind, sind im Regelfall besser gefüllt als Tonnen, die eher außerhalb vom Stadtzentrum stehen, sagen die Müllwerker. Dies liegt daran, dass im Stadtzentrum eine höhere Bevölkerungsdichte als außerhalb herrscht, somit fallen auf eine Tonne rechnerisch mehr Verbraucher an. Des Weiteren ist in den Gegenden außerhalb vom Stadtzentrum eine eigene biologische Müllentsorgung häufi-

ger der Fall, da Einwohner hier ihren biologischen Müll oftmals noch selbst kompostieren, was im Stadtzentrum aufgrund von Platzmangel und Geruchsbelästigung in deutlich geringerem Umfang auftritt. Diese Faktoren werden in der Simulation berücksichtigt und im Kapitel 3.4.2 Zuweisung der Füllstände berechnet.

### 3.3 Einlesen der Behälterliste

Die Simulation kann als ausführbare Java Datei über die Kommandozeile mit folgendem Befehl gestartet werden:

```
java -jar TrashCanSimulation.jar -cl CanList.xlsx -c 60 -tn Test-Tour
```

Über den Parameter `-cl` wird definiert, dass die folgende Datei im Input-Ordner “CanData“ der Anwendung hinterlegt ist. Dies erspart ein unnötiges Navigieren durch den Projektexplorer über die Kommandozeile, um den genauen Dateipfad der Behälterliste zu spezifizieren. Der Parameter bietet durch Weglassen aber auch die Möglichkeit, die Datei aus einem anderen Verzeichnis zu laden. Bei der Datei “CanList.xlsx“ (Abbildung 4) handelt es sich um eine Excel-Tabelle, die alle für die Simulation verwendbaren Behälterobjekte mit den in Kapitel 3.1 Definition eines Behälterobjekts definierten Attributen enthält.

	A	B	C	D	E	F	G
1	Can Nr.	Location	GPS lat	GPS long	Emptying time in seconds	Empty filllevel probability	Sensor
2	<b>181</b>	Luitpoldstraße 15b	49.01392	12.10602	40	0,2	true
3	<b>471</b>	Hemauerstraße 20a	49.01222	12.10715	57	0,2	true
4	<b>432</b>	Sternbergstraße 14b	49.01221	12.10556	48	0,3	true
5	<b>244PT</b>	Hemauerstraße 19	49.01272	12.10569	17	0,1	true
6	<b>422</b>	Landshuterstraße 10	49.01493	12.10497	45	0,2	true
7	<b>460</b>	Von-der-Tann-Straße 17	49.01632	12.10438	36	0,2	true

**Abbildung 4:** CanList.xlsx - Behälterliste

Der Parameter `-c` bestimmt die Anzahl an Tonnen, die für die Simulation berücksichtigt werden soll. In diesem Beispiel werden die ersten 60 Tonnen der Behälterliste simuliert. Der letzte Parameter definiert die Tourbezeichnung und kann frei gewählt werden. Existiert bereits eine Tour mit

dieser Bezeichnung im Ausgabeverzeichnis der Simulation, so wird die Tourbezeichnung um einen Index erweitert, um das Überschreiben von bereits vorhandenen Dateien zu vermeiden. Sollte das der Fall sein, erfolgt eine Benachrichtigung über die Konsole.

Sobald die Simulation gestartet wird, erzeugt die `main`-Methode der Simulationsklasse ein InputManager-Objekt. Dieses Objekt ruft über seinen Konstruktor eine Methode zum Einlesen der Behälterobjekte aus der Exceltabelle auf. In der Methode wird mithilfe der ApachePoi-Programmbibliothek über einen `FileInputStream` ein XSSFWorkbook-Objekt aus der “CanList.xlsx”-Datei erzeugt und über dessen Zeilen iteriert. Die Anzahl der Iterationen ist abhängig vom Parameter `-c` in der Kommandozeile. Dieser definiert, wie bereits beschrieben, die Anzahl der zu verwendenden Tonnen und bestimmt somit die Menge an zu lesenden Zeilen der Tabelle. Dabei wird für jede Zeile eine neue `TrashCan`-Instanz generiert und mit den hinterlegten Attributen gefüllt. Dieses Objekt wird in einer `ArrayList` gespeichert, die bei Verlassen der Methode zurückgegeben und im InputManager-Objekt hinterlegt wird.

### 3.4 Generierung einer Tour

Nachdem die Behälterliste im InputManager-Objekt gespeichert wurde, erfolgt eine Prüfung, ob die Liste einen Inhalt besitzt. Mindestanforderung um eine Tour erzeugen zu können, sind zehn Tonnen, da sich für eine niedrigere Anzahl eine Fahrt nicht lohnt. Ist diese Prüfung erfolgreich, wird aus der im InputManager gespeicherten Behälterliste eine Tour erzeugt. Dies erfolgt über einen Methodenaufruf im Konstruktor des InputManager-Objekts.

Hier wird eine neue Instanz der Klasse `Tour` erzeugt. Dieser wird die Behälterliste vom InputManager, der das Tourobject erzeugt, übergeben. Sobald die Liste übergeben wurde, springt das Programm in die Methode zur Kalkulation und Zuweisung der einzelnen Füllstände der Tonnen im Konstruktor des erzeugten Tourobjekts.

### 3.4.1 Kalkulation der Füllstände

Das wichtigste Kriterium für einen möglichst realistischen Simulationsvorgang ist die Bestimmung der Füllstände der Tonnen, da von ihnen das gesamte Ergebnis der Simulation abhängig ist.

Im Kapitel 3.2 Füllstandanalyse werden die drei wichtigsten Faktoren bereits erklärt, die Einfluss auf den Füllgrad der einzelnen Tonnen haben. Um diese Faktoren bei der Fülllevelbestimmung zu berücksichtigen, muss ein Konstrukt entworfen werden, in dem diese Faktoren zur Geltung kommen. Hierfür wurde nach einigen Testentwürfen folgendes Szenario kreiert. Für jede Tonne wird eine Wahrscheinlichkeit berechnet, die abhängig der im Kapitel der Füllstandanalyse genannten Faktoren ist. Um diese Wahrscheinlichkeit zu berechnen wurde folgende Formel entworfen:

$$P(x_i) = q_i * \left(\frac{c_i}{d_i}\right) \quad (1)$$

Dabei beschreibt  $P(x_i)$  die Wahrscheinlichkeit eines Füllgrades von weniger als 50% für die Tonne  $i$ , wodurch die Tonne als leer gilt. Diese Wahrscheinlichkeit errechnet sich aus den folgenden Faktoren:

#### - $q_i$ - Füllstandswahrscheinlichkeit anhand von Beobachtungen

Laut den Müllwerkern ist die Regelmäßigkeit der Füllungen so typisch für die einzelnen Behälter, dass dieser Faktor ausschlaggebend ist. Das bedeutet, für alle inspizierten Tonnen der Freitagstour wird für den Füllstand folgender Wahrscheinlichkeitswert hinterlegt:

- **0,5** - Für Tonnen, die bei der Inspektion als leer galten (deutlich weniger als halbvoll)
- **0,3** - Für Tonnen, die bei der Inspektion circa halbvoll waren
- **0,2** - Für Tonnen, die bei der Inspektion als voll galten (wesentlich mehr Inhalt als halbvoll)
- **0,1** - Für Tonnen, die bei der Inspektion überfüllt waren (Tonne lässt sich aufgrund der Überfüllung nicht mehr richtig schließen)

Für die restlichen Tonnen, die nicht in der Freitagstour mit einbezogen sind, wird der Wahrscheinlichkeitswert von 0,3 verwendet. Dieser entspricht dem errechneten Durchschnittswert der inspizierten Tonnen der Tourbegleitung und sollte somit einen guten Vergleichswert darstellen. Die Wahrscheinlichkeit  $q_i$  muss für jede Tonne in der Eingabedatei "CanList.xlsx" hinterlegt werden, da sie nicht berechnet werden kann. Sollte kein Wert hierfür hinterlegt werden, wird dem Behälter-Objekt beim Einlesen automatisch der Standardwert 0,3 zugewiesen.

**-  $c_i$  - Faktor zur Bestimmung der Anzahl an Tonnen in unmittelbarer Nähe**

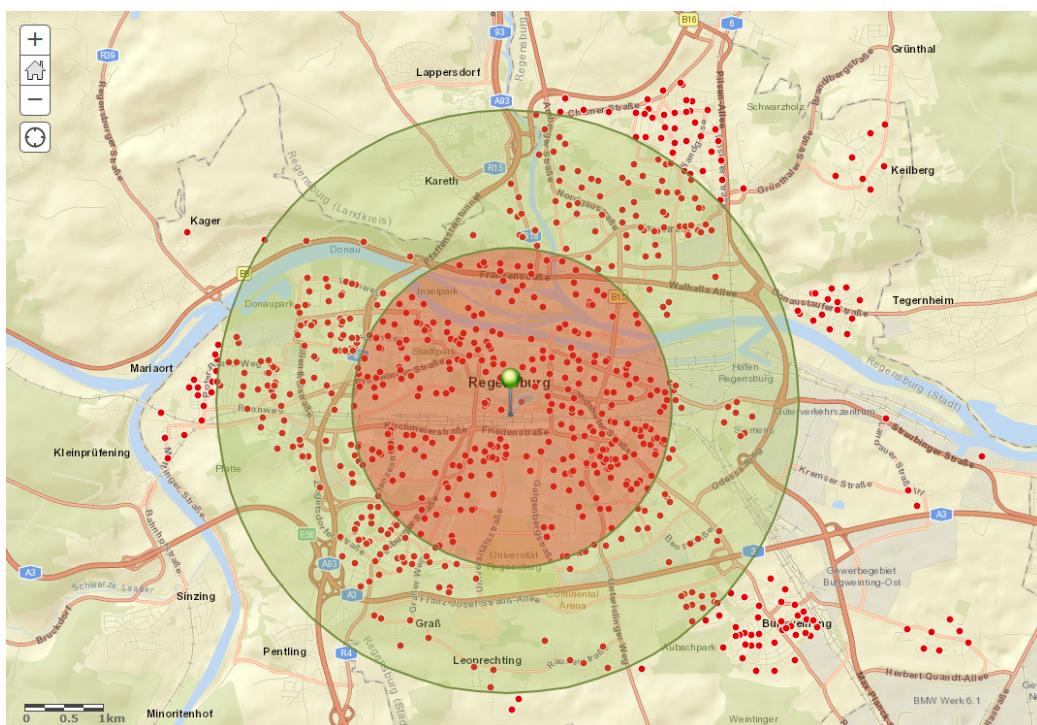
Um diesen Faktor zu bestimmen, wird ermittelt, wie viele Tonnen sich in unmittelbarer Nähe der Tonne  $i$  befinden. Umso mehr Tonnen in einem kleinen Gebiet stehen, desto größer wird der Verteilungsraum für den anfallenden biologischen Abfall in diesem Gebiet. Und umso größer die Verteilungsmöglichkeit wird, desto weniger Müll fällt in den einzelnen Tonnen an, wodurch die errechnete Wahrscheinlichkeit steigt. Daraus lässt sich bestimmen, das mit wachsendem Faktor  $c_i$  auch die Wahrscheinlichkeit steigt. Dabei werden die folgenden drei Fälle unterschieden:

- **0,9** - Im Umkreis von 200 Metern steht maximal eine weitere Tonne
- **1,1** - Im Umkreis von 200 Metern stehen maximal fünf weitere Tonnen
- **1,3** - Im Umkreis von 200 Metern stehen mehr als fünf weitere Tonnen

**-  $d_i$  - Entfernung zum Stadtzentrum**

Die Gründe für den Umstand, dass Tonnen, die sich im Stadtzentrum oder in der Nähe befinden, statistisch gesehen voller sind als Tonnen außerhalb, wurden bereits bei der Füllstandanalyse erklärt. Um dies bei der Generierung der Füllstände zu berücksichtigen, wird der Faktor  $d_i$  in der Formel eingefügt. Dieser definiert den Entfernungsfaktor der Tonne vom Stadtzentrum. Dafür wird das Stadtgebiet Regensburg, in Abbildung 5 zu sehen, in drei Bereiche

eingeteilt:



**Abbildung 5:** Unterteilung der Standorte in drei Bereiche ausgehend vom Stadtzentrum

Für die Faktorbestimmung gilt folgende Regel:

Der Wert der Wahrscheinlichkeit für einen leeren Behälter vergrößert sich mit zunehmender Entfernung zum Stadtzentrum. Der Faktor kann aufgrund seiner Aufteilung in drei Bereiche folgende Werte annehmen:

- **1,2** - Roter Bereich mit Abstand zum Zentrum kleiner als zwei Kilometer
- **1,1** - Grüner Bereich mit Abstand zum Zentrum kleiner als vier Kilometer
- **0,9** - Außerhalb der markierten Bereiche mit Abstand zum Zentrum von vier Kilometern oder mehr

Die Bestimmung dieser Parameter entstand durch eine Reihe an Berechnungen mit Testvergleichen unterschiedlicher Tonnen. Um die Formel an einem Beispiel zu erläutern, wird folgendes Testszenario erstellt:

Nehmen wir an, diese Beispieltonne wäre bei der Inspektion der Tourbegleitung voll gewesen, so würde ein Wahrscheinlichkeitswert für eine Füllung weniger als halb voll bei der Simulation von 0,2 hinterlegt werden. Des Weiteren nehmen wir an, die Tonne befindet sich im Stadtzentrum und hat lediglich eine weitere Tonne in Ihrer Umgebung (näher als 200 Meter). So sprechen alle Parameter für eine Abnahme der Wahrscheinlichkeit für eine leeren Tonne. Dies bestätigt die Berechnung:

$$P(x_i) = 0,2 * \left(\frac{0,9}{1,2}\right) = 0,15$$

Die Wahrscheinlichkeit für die Tonne, leer zu sein, verringert sich somit um 5% durch die berücksichtigten Einflussfaktoren.

Die Umsetzung der Füllgradbestimmung im Quellcode wird im folgenden Kapitel erläutert.

### 3.4.2 Zuweisung der Füllstände

Nachdem das Tourobject erzeugt wurde, wird über einen Methodenaufruf im Konstruktor des Objekts die Füllstandssimulation gestartet. Hierbei wird über die Behälterliste iteriert und für jede Tonne der folgende Berechnungsablauf durchgeführt:

```

for(TrashCan can : canList) {
    pCenterDistance = calculateDistanceProbability(can.getGpsData(), cityCenter);
    pCansInNeighbourhood = calculateCansInNeighbourhoodP(can.getGpsData());

    // Calculation of the probability for an empty filllevel
    pCalculated = (can.getEmptyFillLevelProbability() * (pCansInNeighbourhood / pCenterDistance));
    fillLevel = chooseFillLevel(pCalculated);

    if(fillLevel == 0)
        can.setFillLevel(EFillLevel.FULL);
    else
        can.setFillLevel(EFillLevel.EMPTY);
}

```

**Abbildung 6:** Funktionscode - Bestimmung des Fülllevels für jedes Behälterobjekt

Zuerst wird der Abstand zum Stadtzentrum berechnet, um den Parameter  $d_i$  für die Formel (1) zu ermitteln.

Hierfür wird der Funktion `calculateDistanceProbability` das GPS-Daten-Objekt der Tonne sowie das GPS-Daten-Objekt des Stadtzentrums übergeben (Abbildung 6, Zeile 2). In der Funktion erfolgt ein weiterer Funktionsaufruf der Methode `calcucalteDistace`, welcher beide GPS-Daten-Objekte übergeben werden. Diese Methode (Abbildung 7) errechnet die Distanz der beiden Punkte. Die Formel zur Berechnung der Distanz basiert auf der Formel zur Berechnung des geografischen Abstands zweier GPS-Koordinaten und ist in der Methode folgendermaßen hinterlegt:

```

private int calculateDistance(GpsData location1, GpsData location2) {
    int distance = 0;
    double lat, dx, dy;

    lat = (location1.getLatitude() + location2.getLatitude()) / 2 * 0.01745;
    dx = 111.3 * Math.cos(lat) * (location1.getLatitude() - location1.getLongitude());
    dy = 111.3 * (location1.getLatitude() - location2.getLatitude());
    distance = (int) (Math.sqrt(dx * dx + dy * dy) * 1000);

    return distance;
}

```

**Abbildung 7:** Funktionscode - Berechnung der Distanz zwischen zwei geografischen Orten anhand der GPS-Koordinaten

Der direkte Abstand wird in Metern berechnet und vor der Rückgabe als double-Wert ins Integer-Format gecastet. Anschließend wird in der Funktion

`calculateDistanceProbability` für diese Distanz der Faktor  $d_i$  nach den oben genannten Kriterien für die Tonne bestimmt und der Variable **pCenterDistance** zugewiesen.

Danach wird der Faktor  $c_i$  ermittelt. Um zu bestimmen, wie viele Tonnen sich in einer bestimmten Umgebung befinden, muss für jede Tonne der Abstand zu allen anderen Tonnen der Tour errechnet und überprüft werden. Das passiert in der Methode `calculateCansInNeighbourhoodP`. Dazu wird über eine Iteration der Behälterliste der geografische Abstand der zu untersuchenden Tonne zu allen anderen Tonnen der Behälterliste anhand der Funktion `calculateDistance` berechnet, die bereits für die Bestimmung von  $d_i$  verwendet wurde. Nach jeder Distanzberechnung erfolgt eine Prüfung, ob die Distanz weniger als 200 Meter beträgt (Kriterium für die Umkreisbestimmung). Ist das der Fall, so wird eine Zählervariable im Funktionsrumpf erhöht. Nachdem alle Abstände berechnet und überprüft wurden, beschreibt die Zählervariable die Anzahl der Tonnen in der gesuchten Nachbarschaft. Anhand dieser Anzahl wird der Faktor  $c_i$  nach den oben genannten Kriterien ermittelt. Bei dieser Vorgehensweise ist darauf zu achten, dass beim Iterieren über die Behälterliste eine Prüfung erfolgt, ob die Tonne am aktuell verwendeten Index nicht die selbe Tonne ist, für welche die Distanz eigentlich berechnet werden soll, sonst würde der Zähler eine Tonne zu viel zählen. Da diese Prüfung allerdings für jeden einzelnen Durchlauf zusätzliche Rechenzeit bedeutet, wird die Prüfung weggelassen. Dafür wird nach der letzten Iteration die Zählervariable wieder um den Faktor 1 dekrementiert.

Der Faktor  $q_i$  wird wie bereits beschrieben in der eingelesenen Datei hinterlegt und ist an diesem Punkt der Simulation bereits in jedem Behälterobjekt gespeichert.

Nun wird mithilfe der Formel(1) die Wahrscheinlichkeit eines leeren Fülllevels für das Behälterobjekt bestimmt. Der errechnete Wert wird auf zwei Stellen nach dem Komma gerundet.

Um nun anhand dieser Wahrscheinlichkeit für das Fülllevel zu entscheiden, ob dieses auf voll oder leer gesetzt werden soll, wird ein davon abhängiges Zufallsprinzip konstruiert und in der Methode `chooseFillLevel` implementiert. In dieser Funktion wird ein Integer-Array mit 100 Einträgen initialisiert. Zu

Beginn werden alle Einträge auf null gesetzt. Nun wird die errechnete Wahrscheinlichkeit mit dem Faktor 100 multipliziert. Man erhält die Anzahl an Stellen, die im Array auf 1 gesetzt werden. Um den Zufall zu bewahren, wird über die Funktion Math.random() jeder einzelne Index des Arrays zufällig bestimmt. Sind alle nötigen Stellen im Array auf 1 gesetzt, sind in der Liste nun genauso viele 1en enthalten, dass bei einem zufälligen Auswählen einer Stelle im Array genau die Wahrscheinlichkeit eine 1 zu ziehen besteht, die vorher für den leeren Füllstand der Tonne berechnet wurde. Nun wird ein letztes Mal ein zufälliger Index erzeugt. Ist der Wert an der Indexstelle im Array eine 0, so wird das Fülllevel als Enumeration als “FULL“ deklariert. Ist an der Stelle eine 1, so wird das Fülllevel als “EMPTY“ deklariert.

### 3.4.3 Aufteilung der Behälter

Die Tour-Klasse besitzt neben der allgemeinen Behälterliste, die in den letzten Kapiteln bearbeitet wurde, noch eine Reihe an weiteren Listen zur Speicherung von Behälterobjekten. Diese Listen sind bis zum Zeitpunkt der Fülllevelbestimmung noch leer. Sobald alle Füllgrade bestimmt wurden, wird noch in der Methode zur Fülllevelkalkulation eine Funktion zur Verteilung der Behälter aufgerufen.

Diese Methode bekommt als Parameter die nun mit den simulierten Füllständen ausgestatteten Behälterobjekte in Form der ArrayList vom Typ TrashCan übergeben. In der Funktion wird über die Behälterliste iteriert und jedes TrashCan-Objekt kopiert. Anhand verschiedener Kriterien werden die kopierten Objekte in die verschiedenen Listen eingefügt. Vor der ersten Zuweisung eines Objekts muss für jede Liste erst eine Instanz als ArrayList vom Typ TrashCan erzeugt werden. Danach kann die Zuweisung erfolgen. Dabei wird folgende Aufteilung beachtet:

- **Komplette Tour**

In diese Liste wird jedes Behälterobjekt unabhängig von Füllstand und Sensorsausstattung kopiert.

- **Optimierte Tour**

Die Optimierte Tour enthält alle Tonnen, die geleert werden müssen.

Das sind alle Tonnen, die keinen Sensor besitzen, und alle Tonnen, die einen Sensor besitzen aber voll sind.

- **Gesparte Tonnen**

Hier werden alle eingesparten Behälterobjekte gespeichert. Das sind alle Tonnen, die einen Sensor besitzen und leer sind.

- **Unnötig geleerte Tonnen**

Hier werden alle Tonnen gelistet, die nicht geleert hätten werden müssen. Das sind alle Tonnen, die keinen Sensor besitzen und leer sind.

Diese Listen dienen als Grundlage für den folgenden Simulationsvorgang.

### 3.5 Simulationsvorgang

Sobald die Belegung der Listen im Tourobject abgeschlossen ist, wird über die Methode `simulateTour` die Tour simuliert. Hierbei wird ein neues Objekt vom Typ `ResultSet` erzeugt, dem das zu simulierende Tourobject übergeben wird. In diesem `ResultSet`-Objekt werden über im Konstruktor definierte Methoden folgende Parameter bestimmt:

- **Anzahl Leerungen - gesamt**

Dieser Parameter wird anhand der Anzahl an Elementen in der Liste der optimierten Tour (Kapitel 3.4.3 Aufteilung der Behälter) bestimmt.

- **Anzahl eingesparte Leerungen**

Entspricht der Anzahl an Elementen in der Liste der gesparten Tonnen.

- **Anzahl unnötige Leerungen**

Entspricht der Anzahl an Elementen in der Liste der unnötig geleerten Tonnen.

- **eingesparte Leerzeit**

Entspricht der Summe aller individuellen Leerzeiten der Tonnen aus der Liste der gesparten Tonnen.

- **unnötige Leerzeit**

Dieser Parameter wird aus der Summe der individuellen Leerzeiten der Liste der unnötig geleerten Tonnen errechnet.

Diese Daten werden für jede simulierte Tour in eine Excel-Tabelle in den output-Ordner “SimulationResults“ ausgelagert. Neben dieser Datei werden noch zwei weitere Dateien erzeugt.

Das passiert, nachdem alle Attribute des ResultSet-Objekts berechnet wurden. Nach der Berechnung werden zwei Methoden aufgerufen, welche die anzufahrenden Behälterlisten jeweils in zwei verschiedene Ausgabeformate für die Routenoptimierung konvertieren. Der genaue Aufbau dieser Dateien wird im folgenden Kapitel genau beschrieben.

## 4 Routenoptimierung

Das Szenario der Mülltonnenleerung im Bereich der Routenplanung ist ein komplexes Problem mit vielen verschiedenen Einflussfaktoren. Die allgemeine Verkehrslage, Baustellen oder falsch geparkte Autos sind nur ein kleiner Auszug an Faktoren, die eine hundertprozentig genaue Kalkulation der Fahrtzeiten und -strecken unmöglich macht. Allerdings lässt sich durch Beachten einzelner Parameter die Planung näherungsweise an das optimale Ergebnis heranführen.

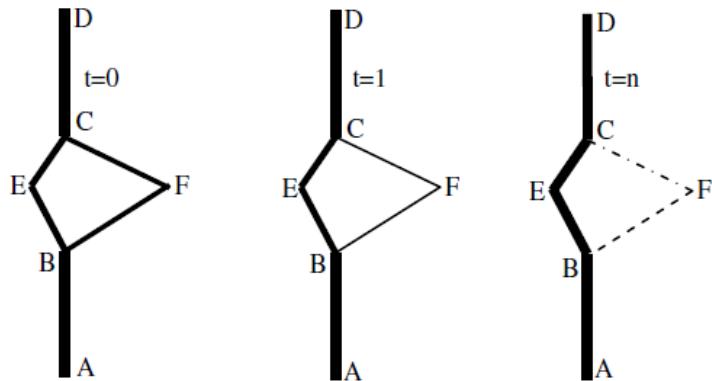
Das Grundproblem der Routenplanung für dieses Szenario lässt sich folgendermaßen definieren. Gegeben ist eine Menge  $n$  an Müllbehältern, die geleert werden müssen und somit vom Fahrzeug angefahren werden. Die Standorte der Tonnen stellen den zu besuchenden Ort für das Fahrzeug dar. Zielsetzung ist es, die kürzeste Strecke zu ermitteln, in der alle Standorte vom Müllwagen angefahren werden. Dieses Problem lässt sich als “Travelling Salesman Problem” oder auch als “Vehicle Routing Problem” klassifizieren [1, S. 46]. Für diese Problemklassen gibt es bereits eine Vielzahl an Lösungsoperationen in Form von Algorithmen. Einer der Bekanntesten ist der sogenannte Ant-Colony-Optimization Algorithmus.

### 4.1 ACO (Ant Colony Optimization)

Die grundlegende Idee hinter diesem Algorithmus entstand durch die Beobachtung von Ameisenkolonien in den 90er Jahren. In einem Experiment von 1996 wurde gezeigt, dass der hohe Sozialisierungsgrad dieser Insektenart Auswirkungen auf das Finden kürzester Wege hat. Bei diesem Experiment wurde den Ameisen auf der Strecke zu ihrer Futterstelle ein längliches Objekt mit zwei Enden E und F in den Weg gelegt (Abbildung 8). Dabei wurde das Objekt so positioniert, dass sich das Ende E des Objekts wesentlich näher an der Standardroute der Ameisen befand, wodurch über diese Strecke der kürzere Weg entstand.

Anfangs teilten sich die Ameisen zu zwei gleichen Teilen auf beide Seiten am Objekt vorbei auf. Bei der Fortbewegung hinterlassen die Tiere eine pheromonhaltige Substanz, die von Artgenossen über den Geruchssinn wahr-

genommen werden kann und mit der Zeit verdampft. Da sich die Insekten mit ähnlichen Geschwindigkeiten fortbewegen, kehrten die Ameisen, die den Weg über das Ende E wählten früher zurück als die Artgenossen vom Weg F.



**Abbildung 8:** Skizzierte Versuchsbeschreibung [1, S. 47] - Aufteilung der Wegstrecke über beide Enden des Objekts

Durch das frühere Zurückkehren entsteht eine höhere Intensität an Pheromonen auf dem Teilweg E, wodurch nachrückende Ameisen sich mit höherer Wahrscheinlichkeit für diese Strecke entscheiden. Der Pheromongehalt der Teilstrecke F wird schwächer ( $t=1$ ) und verdampft nach einiger Zeit ( $t=n$ ) komplett, wodurch alle nachrückenden Ameisen die andere Seite wählen und somit der kürzeste Weg gefunden wurde. Dieses Prinzip wurde programmiertechnisch in vielen verschiedenen Varianten umgesetzt. Für dieses Szenario wird die in Java implementierte Software ACOTSP.V1.03 [3] verwendet. Dieser Quellcode darf frei verwendet und verändert werden.

#### 4.1.1 TSPLIB-Bibliothek

Um die ACOTSP-Software für dieses Szenario verwenden zu können, müssen die erzeugten Behälterlisten der Simulation in das richtige Format konvertiert werden. Grundlage hierfür ist die TSPLIB-Bibliothek. Diese enthält eine Vielzahl an Beispieldaten von TSP verwandten Problemen. Die Grund-

funktion dieser Bibliothek lässt sich allgemein so beschreiben, dass für ein ausgewähltes Problem eine Anzahl an zu besuchenden Knoten gegeben ist. Für diese Knoten wird eine Distanzmatrix berechnet, nach der der kürzeste Weg über den Algorithmus gesucht wird. Die Bibliothek bietet verschiedene Funktionen zur Distanzberechnung der Knoten abhängig von der eingesetzten Problemform. Für unser Szenario wird das symmetrische “Traveling Salesman Problem“ angewendet. Dieses beschreibt die in Kapitel 4 Routenoptimierung definierte Problemstellung der Leerungsfahrt mit der Eigenschaft, dass die Distanz von Standort i nach j die gleiche ist wie von Standort j nach i.

#### 4.1.2 Konvertierung der Behälterliste in das .tsp Format

Im ResultSet-Objekt, welches als Ergebnis der Simulation erzeugt wird, ist die Liste mit den ausschließlich zu leerenden Behältern enthalten. Diese beinhaltet alle Mülltonnen, die das Fahrzeug anfahren muss. Die Mülltonnen definieren mit ihrem Standort die Knotenpunkte des TSP-Problems und somit die Größe der darin errechneten Distanzmatrix. Um die Behälter als Knotenpunkte verwenden zu können, müssen diese in eine Datei mit folgendem Dateiformat ausgelagert werden. Dieses Dateiformat [4, S. 2] teilt sich in zwei Bereiche auf:

- **Spezifikation**

Die Spezifikation definiert den Kopfbereich der Datei (Abbildung 9) und dient zur Einstellung der Grunddefinitionen. Alle Einträge haben die folgende Form:

<keyword>: <value>

<keyword> beschreibt ein von der TSPLIB-Bibliothek vorgeschriebenes Keyword und ist fix. <value> beschreibt den für die Datei spezifischen Wert.

```

1 NAME: Tour Friday
2 TYPE: TSP
3 COMMENT: Regensburg biological waste collection
4 DIMENSION: 60
5 EDGE_WEIGHT_TYPE: GEO
6 DISPLAY_DATA_TYPE: COORD_DISPLAY
7 NODE_COORD_SECTION

```

Abbildung 9: .tsp Datei - Spezifikationsbereich

#### **TYPE:**

spezifiziert den Problemtyp. Wie oben bereits beschrieben wird das symmetrische Traveling-Salesman-Problem verwendet. Dieses wird über die feste Zuweisung **TSP** ausgewählt.

#### **DIMENSION:**

definiert je nach Problemtyp einen anderen Parameter. Ist der Problemtyp **TSP** gewählt, muss hier die Anzahl der Knoten hinterlegt werden. Dieser Wert entspricht der Behälterlistengröße (Anzahl der Tonnen).

#### **EDGE\_WEIGHT\_TYPE:**

definiert den Typ der Kantengewichtung (Distanzberechnung der einzelnen Knotenverbindungen). Auch hierfür gibt es unterschiedliche Auswahlmöglichkeiten. Die meisten TSP-Probleme basieren auf dem Euklidischen Abstand **EUC\_2D**. Die Behälterliste speichert die Behälterobjekte mit GPS-Daten. Zur Berechnung der Kantengewichte über den geografischen Abstand anhand der GPS-Daten bietet die TSPLIB-Bibliothek den Parameter **GEO**.

- **Datencontainer**

Direkt nach dem letzten Keyword beginnt in der neuen Zeile der Datencontainer (Abbildung 10). Der Datencontainer besteht aus einer Liste an Knoten in dem zum Problemtyp passenden Format [4, S. 4]. Für die geografische Berechnung wird ein Knoten folgendermaßen definiert:

<node> <latitude> <longitude>

```

7 NODE_COORD_SECTION
8 181PB 49.01392 12.10602
9 471PB 49.01222 12.10715
10 412PB 49.01336 12.09053
11 413PB 49.01348 12.09013
12 EOF

```

**Abbildung 10:** .tsp Datei - Datencontainer mit vier hinterlegten Behälterobjekten

Um die Tonnen nach Berechnung des Algorithmus zu identifizieren, wird für den Wert <node> die Behälternummer aus der Simulation eingefügt. **EOF** kennzeichnet das Ende der Datei.

Sobald ein ResultSet-Objekt in der Simulation erzeugt wurde, wird über einen Methodenaufruf im Konstruktor des Objekts im Output-Ordner “TspFiles“ der Simulation ein .tsp File generiert, welches das soeben beschriebene Format besitzt. Die Dimension wird über die Größe der Behälterliste festgelegt. Nachdem der ansonsten statische Kopfbereich der Datei erzeugt wurde, wird über die Behälterliste iteriert und für jedes Objekt eine neue Zeile im .tsp File erzeugt. Darin werden die Tonnenattribute Behälternummer, Latitude und Longitude der Gps-Daten, wie die Abbildung 10 zeigt, hinterlegt.

#### 4.1.3 Testlauf der ACO-Software

Für erste Testversuche wird im Simulationsprogramm die Freitagstour simuliert um anhand der Ergebnisse der Tourbegleitung eine gute Vergleichsmöglichkeit zu erhalten. Hierfür werden alle 60 Tonnen der Tour mit Fülllevel “FULL“ simuliert, damit diese auch in der zu leerenden Liste gespeichert werden. Die von der Simulation erzeugte .tsp Ausgabedatei enthält nun alle 60 Behälter und wird als Input-Datei für die ACO-Software verwendet. Der erste Testversuch zeigt, dass der Algorithmus der ACO-Software arbeitet und die von der Simulation erzeugte Datei kompatibel formatiert ist. Die damals gemessene Strecke der Tourbegleitung zeigte knapp 26 Kilome-

ter. Der vom Algorithmus erzeugte kürzeste Weg ist genau 50 Kilometer. Eine gewisse Abweichung der Werte wird erwartet, da die Berechnung über Kantengewichte niemals die gleiche Genauigkeit wie die einer GPS-Messung bietet. Dass die Strecke allerdings doppelt so lang kalkuliert ist, kann nicht korrekt sein.

Die erste Vermutung für diese fehlerhafte Berechnung der ACO-Software beruht auf der Berechnung der einzelnen Knotendistanzen. Um das nachzuprüfen wird die Funktion zur Berechnung der Distanzmatrix untersucht. Diese Funktion berechnet anhand zweier gegebener Knotenpunkte die geografische Distanz der Punkte anhand der GPS-Koordinaten. Bevor der Grund für die zu hohe Kalkulation gefunden wird, zeigt sich ein weiteres Problem in der vorletzten Zeile der Methode:

```

q1 = Math.cos(longi - longj);
q2 = Math.cos(lati - latj);
q3 = Math.cos(lati + latj);
dd = (int) ((6378.388 * Math.acos(0.5 * ((1.0 + q1) * q2 - (1.0 - q1) * q3)) + 1.0) * 1000);

return dd;
}

```

**Abbildung 11:** Methodenauszug der Funktion zur Berechnung der Distanz zwischen zwei Knoten der ACO-Software

Die Berechnung der Distanz, in der Methode (Abbildung 11) als Variable **dd** definiert, erfolgt in Kilometer und wird vor der Rückgabe in einen Integerwert gecastet. Das bedeutet, dass die Nachkommastellen für jede Distanz vernachlässigt werden. Da die Biotonnen im Schnitt aber weniger als einen Kilometer voneinander entfernt sind, würde der für unser Szenario entscheidende Teil wegfallen. In einem kurzen Beispiel erklärt würde das bedeuten: Wenn zwei Tonnen einen errechneten Abstand von 400 Meter haben, wird dieser vorerst in Kilometer als float-Wert errechnet. Das entspricht einem Wert von 0.40. Bei der Zuweisungsoperation muss der Wert in das Integerformat konvertiert werden, was bedeutet, dass der Teil hinter dem Komma wegfällt, wodurch die tatsächlich errechnete Distanz nun 0 Kilometer beträgt.

#### 4.1.4 Fehlerbehandlung

Um diesen Fehler zu beheben, war die erste Idee, die Distanz als Rückgabewert der Funktion nicht ins Integer-Format zu casten, sondern als Float-Wert zurückzugeben. Nach kurzer Formatierung im Code wird klar, dass diese einfache Änderung eine Kettenreaktion an Konvertierungsarbeit in der ACO-Software nach sich zieht, die nicht zu stemmen ist, da die Distanzmatrix Grundlage für alle weiteren Berechnungen im Algorithmusverlauf ist.

Folgender alternativer Lösungsansatz zeigt sich effektiv und wesentlich einfacher. Der errechnete Kilometerwert wird vor der Zuweisung an die Distanzvariable mit dem Faktor 1000 multipliziert. Dadurch werden die essentiellen Nachkommastellen, die die restliche Entfernung in Meter definieren, vors Komma geschoben. Danach wird der Wert wie in der Funktion beschrieben als Integer gecastet und die Nachkommastellen abgetrennt. Somit erhält man nun die Distanz in Meter. Der Rest der Berechnung bleibt identisch. Nun ist nur zu beachten, dass die errechnete Idealdistanz im Ergebnis nun in Meter ausgegeben wird.

Anhand von Testkoordinaten wird die Berechnung überprüft. Die Überprüfung zeigt, dass die Berechnung nun in Meter erfolgt und auch bei Distanzen von weniger als einem Kilometer der genaue Wert berechnet wird. Die Funktion wird im ACOTSP-Programmcode geändert. Ein neuer Testlauf liefert nun das Ergebnis 52756, welches auf den Meter genau berechnet wurde. Das Konvertierungsproblem von Kilometer in Meter wäre somit gelöst, nicht allerdings der Umstand, dass das Ergebnis immer noch ungefähr doppelt so hoch ist wie angenommen.

Bei genauem Überprüfen der Testkoordinaten fällt auf, dass Ergebnisse einzelner Distanzberechnungen von verschiedenen Tonnen viel zu hoch ausfallen. Folgender Tabellenauszug zeigt einen Beispielvergleich:

<b>Testkoordinaten</b>	<b>ACODistanz in Meter</b>	<b>Tatsächliche Entfernung in Meter</b>
<b>1</b>	<b>1344</b>	206
<b>2</b>	<b>1739</b>	443
<b>3</b>	<b>1968</b>	580
<b>4</b>	<b>1765</b>	459
<b>5</b>	<b>1228</b>	137

**Abbildung 12:** Tabellenauszug zum Vergleich zwischen den berechneten Distanzen

Die Berechnungen der Distanzfunktion weichen in allen Fällen maßgeblich von der wirklichen Entfernung der Tonnen ab, wie Abbildung 12 zeigt. Aus diesem Grund wird die Struktur der Methode zur Distanzberechnung grundlegend verändert. Die Methodendefinition bleibt gleich, da eine Konvertierung des Programms zu aufwändig ist. Lediglich der Rumpf der Funktion wird verändert. Hier wird eine neue Formel zur Berechnung der Distanz eingesetzt.

#### 4.1.5 Optimierung der Distanzberechnung

Die Grundlage für die neue Distanzberechnung stellt die Formel zur Berechnung des direkten geografischen Abstands zweier Koordinaten dar, die bereits im Kapitel 3.4.2 Zuweisung der Füllstände verwendet und in Abbildung 7 gezeigt wurde. Die Verwendung dieser Distanzberechnung ist für das Szenario der Mülltonnenabstände allerdings noch zu ungenau wie folgender Problemfall zeigt.



**Abbildung 13:** Distanzbeispiel anhand von zwei Tonnen

Es entsteht eine Abweichung von circa 20% zur errechneten Luftlinie. Der Vergleich mit weiteren Koordinaten zeigt logischerweise, dass diese Abweichung in fast allen Fällen auftritt. Ausnahme hierbei sind nur errechnete Abstände von Tonnen, die sich in der gleichen Straße befinden. Hier entspricht die errechnete Luftlinie auch der tatsächlichen Fahrtstrecke.

Die Abweichung der Werte entsteht durch den Richtungswechsel des Fahrzeugs. Dies geschieht im Allgemeinen in zwei Fällen. Entweder durch Abbiegen auf eine andere Straße (Abbildung 13), oder wenn der Straßenverlauf nicht gerade verläuft, so dass die Straße aufgrund der Richtungsänderung einen Bogen erzeugt.

Um dies im Algorithmus zu beachten, wird der Problemfall folgendermaßen analysiert.

Befinden sich zwei Tonnen in der selben Straße, so wird die Entfernung der Tonnen über die Luftlinie annähernd perfekt beschrieben, solange die Straße gerade verläuft. Untersucht man die Straßenkarte der Stadt Regensburg, so stellt man fest, dass annähernd alle Straßen einen geraden Straßenverlauf haben. Eine ausschlaggebende Krümmung von Straßen kommt so selten vor, dass Sie für dieses Szenario vernachlässigt werden kann. Um den Fall mit einzubeziehen, dass sich zwei Tonnen in der gleichen Straße befinden, muss das in der Distanzberechnung der Tonnen geprüft werden. Hierfür wird das Simulationsprogramm um einen Faktor erweitert. In der InputManager-

Abbildung 13 zeigt die ersten zwei Tonnen der Freitagstour (Testfall 1 in der Vergleichstabelle). Sie befinden sich lediglich zwei Straßen voneinander entfernt und haben einen errechneten direkten Abstand von 206 Meter (Luftlinie). Aufgrund der Verteilung über zwei Straßen entsteht allerdings eine tatsächliche Fahrtstrecke von 260 Metern.

Klasse wird als neues Attribut eine Straßenliste hinterlegt. Beim Einlesen der Tonnen wird nun zwischen Straße und Hausnummer unterschieden. Wird ein Behälterobjekt aus der Behälterliste im InputManager eingelesen, wird die Straße des Behälters in der Straßenliste gespeichert, insofern diese nicht schon vorhanden ist. Das Behälterobjekt bekommt dann den Listenindex der Straße zugewiesen.

Bei Erzeugung der .tsp Datei für den ACO Algorithmus wird im Datencontainer (Liste der Behälter) nun für jede Zeile der Straßenindex mit angegeben. Um diesen Faktor auch in der ACO-Software mit einzubeziehen, müssen zwei Änderungen am Quellcode vorgenommen werden. Zuerst wird die Struktur, die das Knotenelement definiert, erweitert. Diese besitzt aktuell nur zwei Attribute zur Speicherung der GPS-Koordinaten. In der Datenstruktur wird nun ein Attribut vom Typ Integer hinterlegt, um den Straßenindex beim Einlesen der Datei zu speichern. Die zweite Änderung wird in der Methode (Abbildung 14) vollzogen, die für das Einlesen der .tsp Datei zuständig ist.

```

} else {
    String[] city_info = line.split(" ");
    nodeptr[i] = new Tsp.point();
    nodeptr[i].x = Double.parseDouble(city_info[1]);
    nodeptr[i].y = Double.parseDouble(city_info[2]);

    // Straßenindex einlesen
    nodeptr[i].z = Integer.parseInt(city_info[3]);
}

i++;
}

```

**Abbildung 14:** Erweiterung der Methode zum Lesen des .tsp Files

Hier wird dem Knotenelement **nodeptr** der Straßenindex zugewiesen, der den vierten Teil der eingelesenen Zeichenkette einer jeden Zeile ausmacht. Nun ist für jeden Knotenpunkt ein Straßenindex hinterlegt. Um diesen Faktor auch in der Berechnung der Distanz zu berücksichtigen, wird in der Methode zur Distanzberechnung eine einfache Prüfung hinterlegt, ob die gegebenen zwei Knotenpunkte, für die die Distanz berechnet werden soll, den gleichen

Straßenindex besitzen. Ist dies der Fall, so wird die Distanz aus der Formel für die direkte Entfernung berechnet. Sollte der Straßenindex nicht identisch sein, so tritt der zweite Problemfall ein.

Bei dieser Problemstellung geht es darum, den tatsächlich fahrbaren kürzesten Weg zwischen zwei Tonnen zu bestimmen, die sich nicht in der gleichen Straße befinden. Für diesen Fall müssten Informationen übers Straßennetz für jeden Weg zwischen den Tonnen hinterlegt werden. Dies würde allein für die 61 Tonnen im Stadtzentrum eine Matrix mit Straßeninformationen von über 3600 Einträgen darstellen, die alle ermittelt werden müssen. Da das für die Simulation nicht realisierbar ist, muss ein Korrekturfaktor bestimmt werden, der die Distanz dahingehend optimiert, dass anhand der Entfernung der Standorte der Faktor näherungsweise an die tatsächliche Fahrtstrecke heranführt. Um diesen Faktor zu ermitteln, werden 50 Beispielkoordinaten auf ihren Zusammenhang zwischen der tatsächlichen Entfernung und der daraus resultierenden fahrbaren Strecke untersucht.

Bei der Untersuchung werden willkürliche Tonnen gewählt und deren Abstand untersucht. Hierbei wird überprüft, inwiefern sich der Wert von der Formel für den direkten Abstand von der fahrbaren Strecke unterscheidet. Die kürzeste fahrbare Strecke wird über Google Maps als kürzeste Route zwischen den zwei Tonnen bestimmt. Hierbei entsteht folgendes Ergebnis: Teilt man die Distanzen der Tonnen in zwei Bereiche auf, nämlich zwischen kleiner als 400 Meter und größer als 400 Meter, entsteht folgender Korrekturfaktor:

Vergleichspaar	direkte Distanz	kürzeste fahrbare Strecke laut Google Maps	Differenz	prozentualer Mehrwert
1	407	580	173	42,51
2	420	600	180	42,86
3	454	650	196	43,17
4	587	850	263	44,80
5	591	850	259	43,82
6	614	850	236	38,44
7	917	1300	383	41,77
8	1366	2000	634	46,41
9	1744	2100	356	20,41
10	1942	2700	758	39,03

**Abbildung 15:** Auszug aus der tabellarischen Gegenüberstellung der Distanzwerte (Längenangaben in Meter)

Die Tabelle (Abbildung 15) zeigt einen Auszug an zehn Beispielvergleichen, die alle im zweiten Bereich liegen, also eine größere direkte Distanz als 400 Meter haben. Bis auf das Vergleichspaar Nummer **9** bewegen sich alle prozentualen Abweichungen der Werte in einem ähnlichen Bereich. Der errechnete Durchschnitt aus allen 25 Vergleichspaaren ergibt einen prozentualen Mehrwert der tatsächlich fahrbaren Strecke laut Google Maps von 40,35%. Dieser Wert wird nun verwendet um eine Beispielberechnung einer kurzen Tour durchzuführen.

Hierfür werden die zehn Vergleichspaare der Tabelle verwendet. Erhöht man jede direkte Distanz mit dem Mehrwertprozentsatz von 40,35%, also multipliziert den Wert mit 1,4035, so entsteht eine Gesamtdistanz von 12690 Meter. Die Summe der aus Google Maps gemessenen Entfernung ergibt einen Wert von 12480 Meter. Das bedeutet, die anhand der direkten Abstände berechnete Tour weicht von der laut Google Maps möglichen Optimalstrecken um 210 Meter ab. Das entspricht einer Abweichung von 1,7%.

Dieser Wert reicht aus, um den Korrekturfaktor in die Funktion zur Berechnung der Distanzen zu integrieren.

Für die berechneten Abstände, die kürzer als 400 Meter sind, fällt die Diskrepanz zwischen den prozentualen Mehrwerten der einzelnen Berechnungen etwas größer aus. Hier lässt sich anhand der 25 Testkoordinaten ein durchschnittlicher Faktor von 36,17 bestimmen. Für die Berechnungen der Distanzen wird somit der Multiplikator 1,36 verwendet.

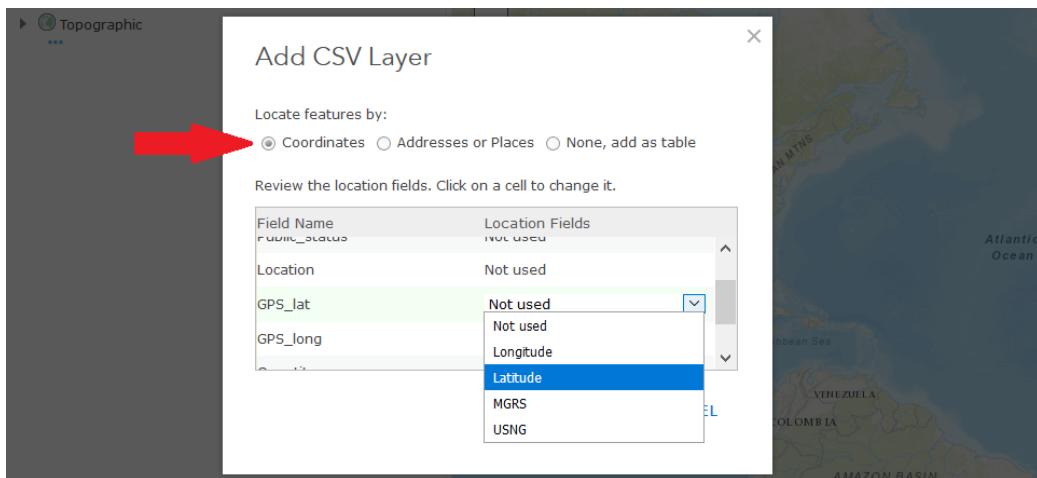
## 4.2 arcGIS

Für die Visualisierung der simulierten Tour wird die Geoinformationssystem-Software **arcMap** verwendet. Dabei handelt es sich um eine Softwarekomponente der Produktreihe **arcGis** der Firma ESRI Inc. Diese ist kostenfrei über einen Online-Account verfügbar. Der User kann eigene Maps erstellen und verschiedene Funktionen auf diesen ausführen, wie zum Beispiel Routenplanungen oder verschiedene Analysen.

Die von der Simulation erzeugte Ausgabedatei der Behälter wird automatisch in eine CSV-Datei konvertiert und kann über einen Web Browser per

Drag and Drop in die arcGis-Oberfläche gezogen werden. Aufgrund der richtigen Formatierung der Datei erkennt die Software jeden Behälter als eigenen Standort an. Über sogenannte “Layer“ können die verschiedenen Attribute realisiert werden.

Um die Standorte richtig zu definieren werden die Koordinaten über die GPS-Daten abgefragt. Dabei ist nach dem Hinzufügen der CSV-Datei die Konvertierung auf “Coordinates“ umzustellen (Abbildung 16). Anschließend werden die richtigen Spaltenattribute ausgewählt und den GPS-Standards Latitude und Longitude zugewiesen.

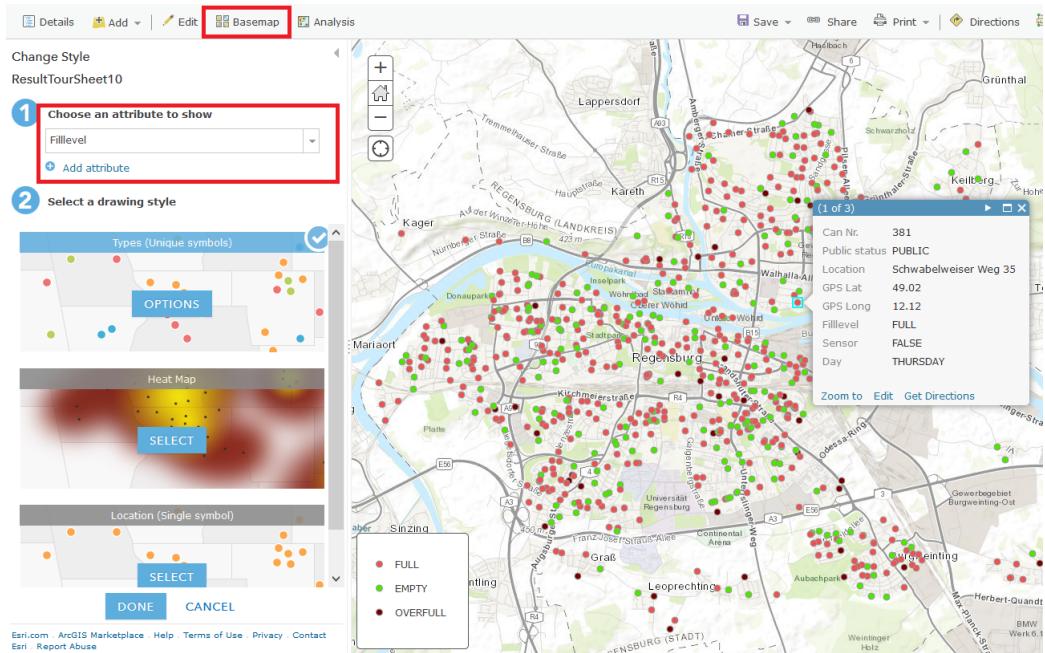


**Abbildung 16:** arcGis Map - Zuweisung der GPS-Layer

#### 4.2.1 Visualisierung der Karte

Nach Hinzufügen des GPS-Layers wird die Karte mit allen Standorten angezeigt (Abbildung 17). Hier kann im Menü über den Attribute-Layer eine verschiedene Sortierung nach Attributen vorgenommen werden. Hier kann eine farbige Visualisierung anhand des Füllgrades eingestellt werden.

Durch einfaches Klicken des Objekts auf der Karte entsteht ein interaktives Informationsfenster, welches alle Informationen über den Behälter enthält.

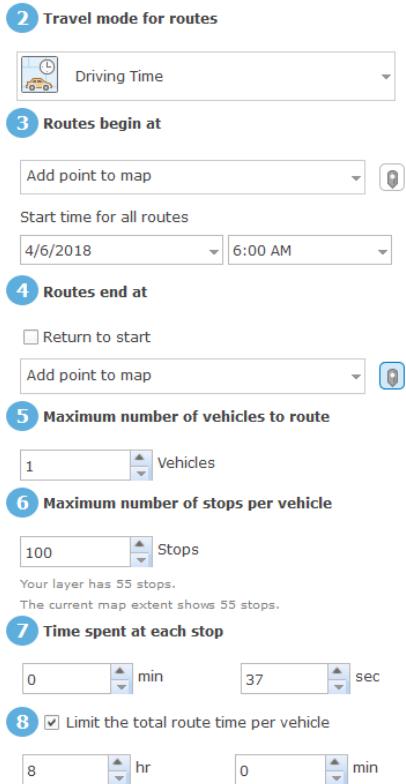


**Abbildung 17:** arcGis Map - Zuweisung des Attribute-Layers

Die Menüleiste bietet weitere Einstellungen, um die Map für das gegebene Themengebiet zu spezifizieren. Dabei kann die Ansicht der Karte über den Menübutton “Basemap” geändert werden. Hierfür gibt es mehrere Auswahlmöglichkeiten. Sinnvoll für die spätere Routenerstellung ist der Basemap-Layer Streets, welcher die Karte in eine Straßenansicht umwandelt. Im Reiter “Analysis“ können verschiedene Analysen und Kalkulationen anhand der eingefügten Standorte durchgeführt werden.

#### 4.2.2 Planungsanalyse

Für das Szenario der intelligenten Müllentsorgung ist der wichtigste Teil neben dem Erfassen der Füllstände die daraus resultierende Änderung bzw. Verbesserung der statisch gefahrenen Routen. Hierfür bietet arcGis den Bearbeitungspunkt “Plan Routes“ (Routenplanung). Über das folgende Kontextmenü (Abbildung 18) kann jede Fahrt individuell geplant und ausgearbeitet werden.



**Abbildung 18:** arcGis Analysis  
- Menüstruktur Routenplanung

## 2 Modus: Driving time

Es kann zwischen verschiedenen Fortbewegungsvarianten unterschieden werden. Für das Szenario der Müllleerung wird die normale Fahrtzeit eingestellt.

### 3 Beginn:

Einzustellen ist hierbei der Startpunkt der Tour im Abfallamt Regensburg, welcher per Mausklick auf der Karte lokalisiert werden kann. Dieser muss für jede Tour manuell bestimmt werden. Für das Zeitmanagement ist außerdem die Startzeit der Tour auswählbar.

### 4 Ende:

Falls die Tour nicht am Startpunkt wieder enden soll, kann per Mausklick ein alternativer Punkt hinzugefügt werden.

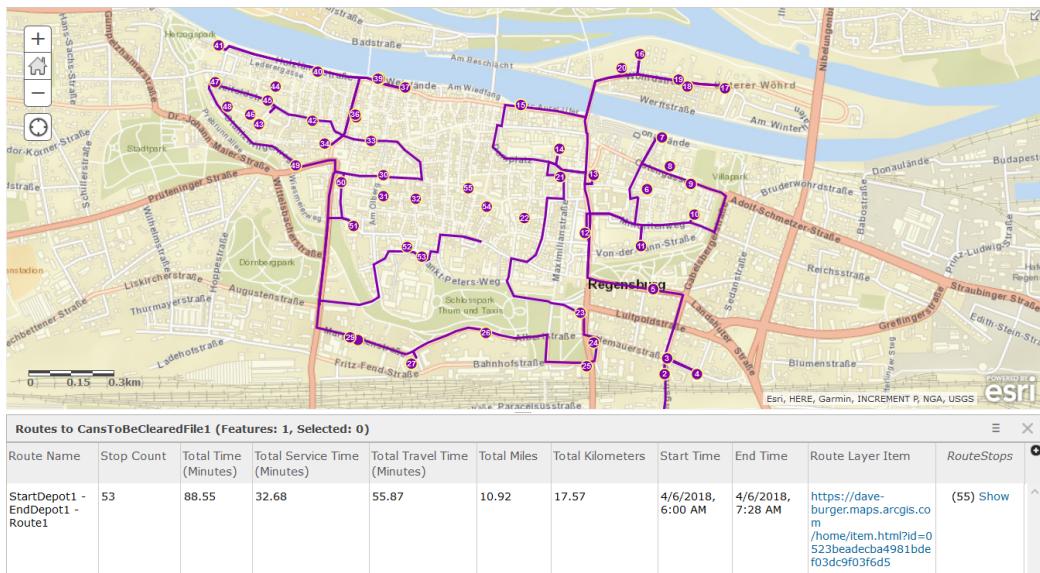
### 5 | 6 Fahrzeuge:

Auswahlmöglichkeit für die Anzahl der Fahrzeuge sowie eines maximalen Limits an Stopps pro Fahrzeug.

### 7 Zwischenhaltezeit:

Mit dieser Einstellung kann die Haltedauer definiert werden, für die jedes Fahrzeug an jedem Punkt der Route halten soll. Diese Zeitspanne entspricht genau der Leerzeit der einzelnen Tonnen. Hierfür wird der Durchschnittswert der Leerzeiten der zu leerenden Tonnen aus dem Simulationsergebnis errechnet und eingesetzt. Somit entspricht die Haltedauer an jedem Punkt in Summe genau der Summe der individuellen Leerzeiten für jede einzelne Tonne.

Nach Berechnung der Analyse wird die bestmögliche Route angezeigt (Abbildung 19). Diese wird anhand der einzelnen Haltestellen auf der Karte visualisiert. Darunter wird die Auswertung in Tabellenform angezeigt.



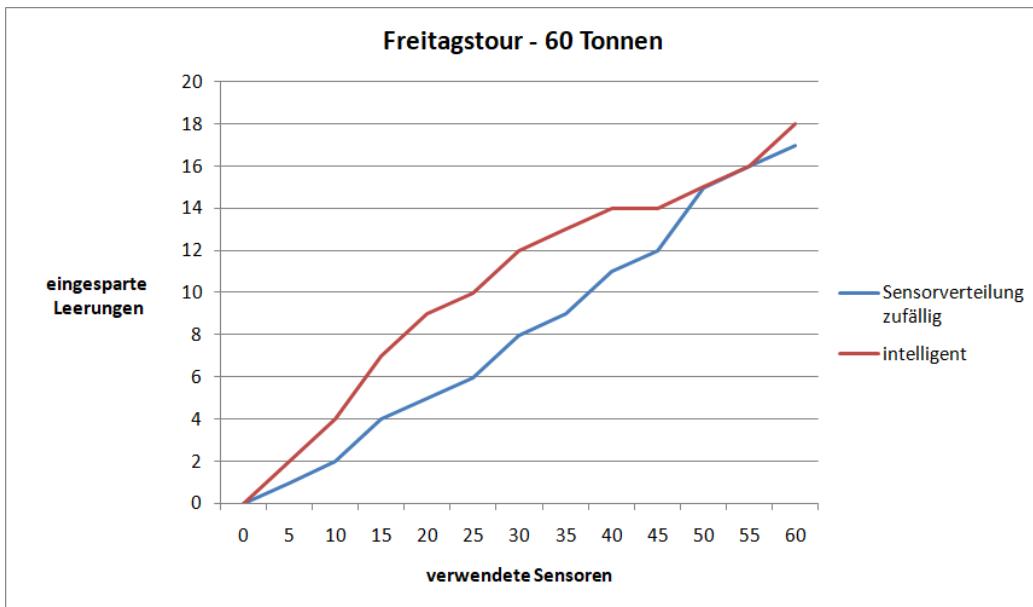
**Abbildung 19:** arcGis Map - Freitagstour optimiert

## **5 Testphase**

In der Testphase werden nun verschiedene Szenarien simuliert um die Auswirkung der Sensoren auf verschiedene Parameter zu zeigen. Dafür werden die Ergebnisse der Simulation, der Wegberechnung durch den ACO Algorithmus sowie die Ergebnisse der Routenberechnung in arcGis analysiert und verglichen. Hierfür werden drei Stichproben der Analysen beschrieben.

### **5.1 Testszenario zur Untersuchung der Sensorverteilung**

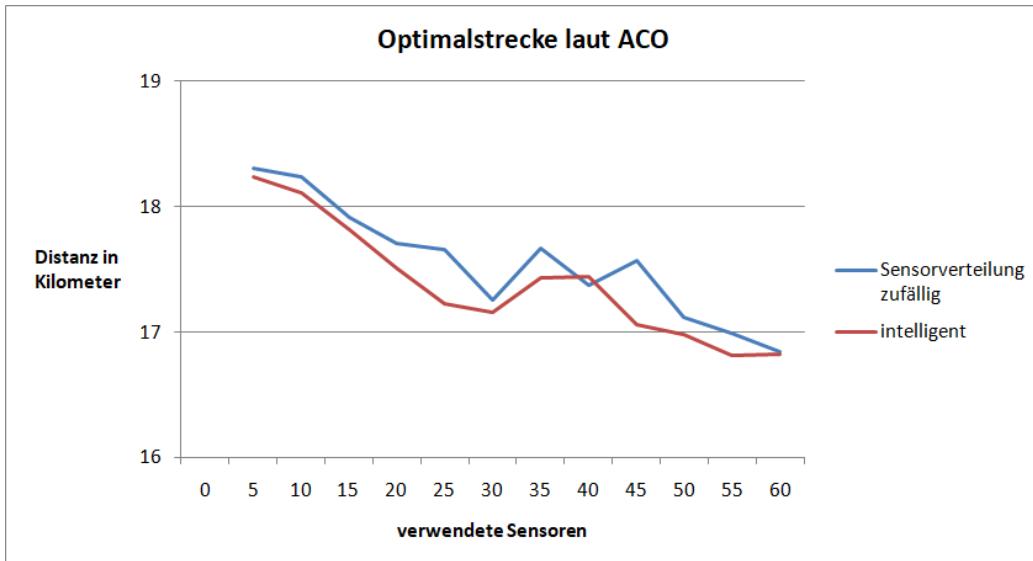
Dieser Test soll zeigen, wie sich das Verhalten zwischen willkürlicher und gezielter Sensorverteilung für mögliche Leerungseinsparungen verhält. Hierfür wird die Freitagstour mit 60 Tonnen simuliert. Dabei wird zwischen zwei Fällen unterschieden. Bei einer Testreihe wird die Sensorverteilung rein willkürlich gewählt. Bei der zweiten Testreihe erfolgt eine intelligente Sensorverteilung abhängig von der für jede Tonne hinterlegten Wahrscheinlichkeit eines leeren Füllgrades. Für jede Testreihe erfolgt eine Sensorzuteilung, bei der die Anzahl beginnend bei fünf Stück in fünf Schritten für jeden Durchgang erhöht wird. Jeder dieser Schritte wird 20 mal simuliert. Davon wird das jeweilige Durchschnittsergebnis ermittelt. Folgendes Diagramm (Abbildung 20) zeigt die Auswertung der Testreihe:



**Abbildung 20:** Diagramm 1 - Einsparungspotential abhängig von der Sensorverteilung

Dabei fällt auf, dass bereits ab fünf Sensoren ein messbares Resultat entsteht. Dieses Resultat steigt relativ stetig an und pendelt sich ab circa 20 verwendeten Sensoren bei einer Steigerung der Leerungseinsparung von ungefähr 30% gegenüber der willkürlichen Sensorverteilung ein. Gegen Ende ist das Resultat der Leerung wieder unabhängig von der Verteilung, da im letzten Drittel die Wahl der Tonnen auch bei gezielter Sensorverteilung ausschließlich auf Tonnen mit einer niedrigen Wahrscheinlichkeit fällt. Dieser Test zeigt, dass für das Stadtgebiet in dem die schlechtesten Wahrscheinlichkeitsbedingungen für einen leeren Füllgrad herrschen, eine intelligente Ausstattung jeder dritten Tonne eine Einsparung von circa 17% bedeutet. Die Einsparung durch bewusste Verteilung der Sensoren ist auf jeden Fall nicht zu unterschätzen. Hierfür würde es sich lohnen, Aufzeichnung der Füllstände für jede Tonne zu veranlassen um den Wahrscheinlichkeitswert dieser Tonnen festzulegen um somit Prognosen treffen zu können.

Für die gleiche Testreihe wird nun die daraus resultierende kürzeste Fahrtstrecke mit Hilfe der ACO Software ermittelt. Dabei entsteht folgende Auswertung:

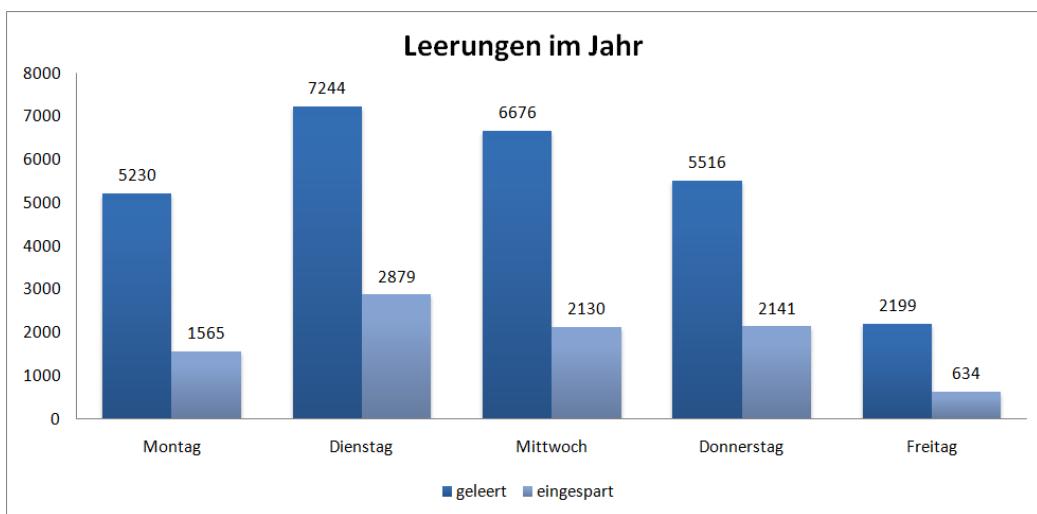


**Abbildung 21:** Diagramm 2 - Streckendistanz abhängig von der Sensorverteilung

Im Diagramm (Abbildung 21) ist zu erkennen, dass auch hier die intelligente Sensorverteilung eine bessere Optimierung für die Strecke bietet als die willkürliche Verteilung. Allerdings fällt das Ergebnis zur Einsparung nicht mehr so deutlich aus wie das Ergebnis der gesparten Leerungen. Das liegt daran, dass der Standort der leeren Tonnen in Bezug zu den nicht leeren Tonnen eine Rolle spielt. Der Faktor der Sensorverteilung, welche abhängig vom Fülllevel bestimmt wird, hat hier eine andere Auswirkung. Leere Tonnen die ausgelassen werden, haben in diesem Bezug eine andere Gewichtung. Hier zählen die Tonnen nicht mehr genauso viel wie beim Auslassen der Leerungen. Hier wird für jede ausgelassene Leerung nur der Wert summiert. Bei der Wegberechnung kommt es drauf an, was das Auslassen der Tonne an der Strecke wirklich verändert. Stehen zum Beispiel noch zwei weitere Tonnen in der gleichen Straße, so ist die weniger zu fahrende Distanz minimal. Dadurch ist die zu fahrende Strecke nur noch indirekt abhängig von der Verteilung der Sensoren. Aber auch hier zeigt sich, umso mehr Leerungen ausgelassen werden können, desto größer ist die Einsparung, in diesem Fall für die gefahrene Strecke.

## 5.2 Jahresanalyse

Um das mögliche Einsparungspotential für ein Kalenderjahr zu untersuchen, werden alle einzelnen Tagestouren jeweils 52 mal mit vollständiger Sensorsausstattung simuliert, um für jede Woche ein Ergebnis zu erhalten. Für jeden Wochentag werden alle durchgeführten Leerungen und alle ersparten Leerungen summiert.

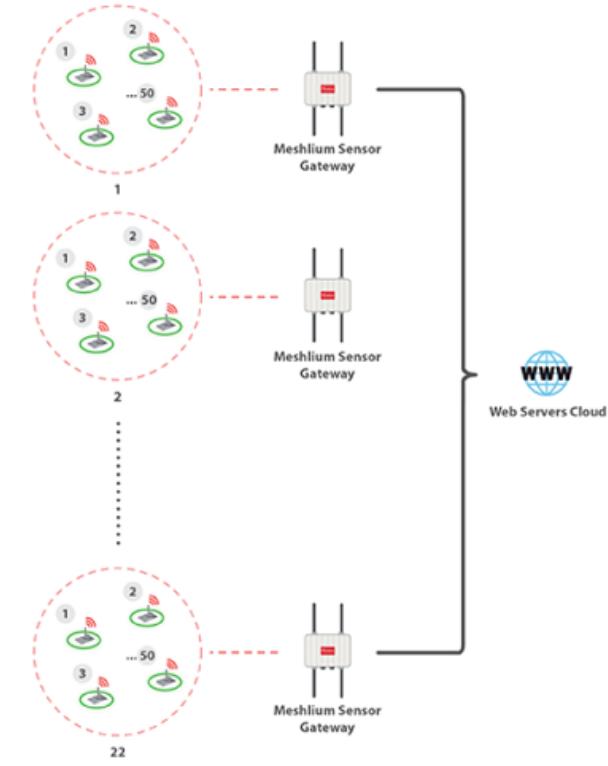


**Abbildung 22:** Diagramm 3 - Leerungen pro Jahr nach Wochentagen aufgelistet

Anhand der Grafik (Abbildung 22) ist zu erkennen, dass die Wochentage Dienstag und Donnerstag prozentual am besten ausfallen. Dies liegt daran, dass bei diesen Touren ein beträchtlicher Teil der Tonnen in ländlicher Gegend liegt, wodurch der Faktor der Entfernung zum Stadtzentrum zur Fülllevelsimulation stärker greift. Dieser Faktor spiegelt sich auch im Wert für den Freitag wieder, dieser fällt mit knapp 28,8% Leerungseinsparung am niedrigsten aus aufgrund der Lage der Tonnen im Stadtzentrum. Würde man also beginnen über eine Sensorsausstattung für einige Tonnen nachzudenken, würde es sich lohnen, in den Gebieten außerhalb der Stadt mit Aufzeichnungen zu beginnen.

## 6 Santander als Vorbild

Die wohl bekannteste “Smart-City“ ist die Stadt Santander in Spanien. Hier sind über 20000 Sensoren in allen Bereichen der städtischen Infrastruktur installiert. Neben der intelligenten Müllsammlung werden hier auch Parkplätze organisiert, Parkanlagen zur Wassereinsparung anhand des Feuchtigkeitsgrades überwacht und noch vieles mehr. Das Konzept dieser Anwendung besteht dabei aus einer sogenannten OTAP-Anwendung. Hierbei spricht man von “Over-The-Air-Programming“, was bedeutet, dass die Kommunikation dieser Sensoren ausschließlich drahtlos erfolgt. Dabei werden in regelmäßigen Abständen sogenannte “Sensor Gateways“ an öffentlichen Plätzen positioniert. Diese Gateways empfangen das Signal der Sensoren und schicken dieses über das GPRS-Netz an die Cloud des Web-Servers weiter, der die Daten dann verarbeitet (Abbildung 23).



**Abbildung 23:** Infrastruktur der Sensorkommunikation in Santander [5]

Die große Problematik bei diesen verwendeten Sensoren stellt die Stromversorgung da. Durch das ständige Senden der Daten wird die Batterie der Sensoren stark beansprucht. Die Batterien der Sensoren überstehen im Schnitt eine Dauer von zwei Jahren. Die Versorgung mit Energie der "Gateways", welche für die Vermittlung der gemessenen Sensordaten zuständig sind, erfolgt ebenso über eingesetzte Batterien. Um diese zu schonen, kann das Gateway so programmiert werden, dass nur in bestimmten Zeitabständen Informationen an den WebService geschickt werden. In der Zwischenzeit, in der keine Datenübermittlung erfolgt, fallen diese Geräte in einen "Sleep-Modus" um Energie zu sparen. Diese Technologie wäre auch für die Sensoren sinnvoll. Eine Messung zum Beispiel jede Stunde oder sogar einmal am Tag würde vollkommen ausreichen, da die Entscheidung der Leerung für jede Tonne spätestens am Tag vor der geplanten Leerung getroffen werden muss.

## **7 Fazit und Ausblick**

### **7.1 Verwendung der ACO-Software zur Routenoptimierung**

Die Verwendung der ACO-Software für das Problem zur Bestimmung der kürzesten Route in diesem Szenario erfordert eine gewisse Modifikation, da das Anwendungsgebiet von diesem Algorithmus eher im Bereich von weiteren Distanzen eingesetzt wird, zum Beispiel zur Berechnung von kürzesten Strecken für eine gegebene Anzahl an Städten.

Um die Software an das Müllentsorgungsszenario anzupassen, müssen einige Faktoren geändert werden um eine realistische Berechnung der kurzen Entfernungen zwischen den Tonnen zu ermöglichen.

Die Gegenüberstellung der Ergebnisse mit Hilfe der arcGis Software aber zeigt, dass der Algorithmus in seiner geänderten Form auch hier funktioniert und ein verwertbares Ergebnis erzeugt. Die schnelle Bearbeitung ermöglicht eine gute Grundlage für Schätzungen und Kalkulationen der Routen. Für genaue Planungen wird die Verwendung der Software als eher schwierig eingeschätzt, da eine Menge Faktoren hierbei nicht berücksichtigt werden können.

### **7.2 Allgemeines Fazit und Danksagung**

Dass für die Stadt Regensburg Einsparungspotential bei der Leerung der Biomülltonnen besteht, ist spätestens nach der ersten Begleitung des Müllwagens bei der Leerungsfahrt vom Freitag ersichtlich. Ein Anteil an leeren Tonnen von ungefähr 30% im Stadtzentrum, und somit im bevölkerungsdichten Teil der Stadt, lässt vermuten, dass die Füllungen der Tonnen auch außerhalb ähnlich leer sind. Dieser Umstand wird in den nächsten Monaten noch verstärkt, wenn die Anzahl der Biomülltonnen auf über 2000 Stück um das fast Dreifache steigen soll.

Die Ergebnisse der Analysen zeigen, dass bereits ein kleiner Anteil an verwendeten Sensoren in bedachten Gebieten ein hohes Einsparungspotential bietet. Auch die Sensorverteilung spielt hierbei eine Rolle. Sollte nur eine bestimmte Anzahl an Sensoren zum Einsatz kommen, wäre es wirtschaftlich

rentabel, für alle Tonnen eine Liste zu erstellen, in der das Fülllevel für ein paar Leerungen notiert wird, um eine Voraussage treffen zu können, ob die Tonnen regelmäßig leer sind oder nicht. Durch diese Wahrscheinlichkeit kann der Einsatz der Sensoren ein wesentlich besseres Ergebnis erzielen.

Mit der vorliegenden Arbeit konnten bereits erste Ansätze und wichtige Kriterien für die Umsetzung des Projekts ermittelt werden. Durch die Zusage von Testsensoren sollen demnächst einzelne Tonnen für Testzwecke ausgestattet werden. Zeigen sich diese Testläufe als Erfolg, so ist die Grundlage für eine intelligente Entsorgung des biologischen Abfalls der Stadt Regensburg geschaffen.

An dieser Stelle möchte ich mich besonders bei meinem betreuenden Professor Dr. Jan Dünnweber für die großartige Unterstützung während der Projektphase bedanken. Mein besonderer Dank gilt vor allem auch meinem Betreuer Konstantin Kirsch von der Firma KPIT Technologies GmbH, der die Grundidee für dieses Projekt schuf und mich während meiner Arbeitszeit rund um die Uhr betreute. Zuletzt auch ein Dank an die zuständigen Kontaktpersonen vom Abfallamt Regensburg sowie an die Truppe der Müllwerker, die ich bei der Tourbegleitung kennenlernen durfte.

## Literatur

- [1] N. V. Karadimas G. Kouzas I. Anagnostopoulos V. Loumos. Urban solid waste collection and routing: the ant colony strategic approach. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.563&rep=rep1&type=pdf>, 2005. (Zugriff: 02.04.2018).
- [2] <https://www.wenglor.com/de/produktwelt/produkte/ultraschallsensoren/>. (Zugriff: 08.04.2018).
- [3] <http://www.aco-metaheuristic.org/aco-code/public-software.html>. (Zugriff: 19.02.2018).
- [4] <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf>. (Zugriff: 20.02.2018).
- [5] <http://www.smartsantander.eu/index.php/testbeds/item/132-santander-summary>. (Zugriff: 01.04.2018).
- [6] Ekkehard Holzbecher. *Environmental Modeling*. Springer-Verlag Berlin Heidelberg, 2012.
- [7] <http://staff.washington.edu/paymana/swarm/stutzle99-eaecs.pdf>. (Zugriff: 13.03.2018).
- [8] <https://www.kompf.de/gps/distcalc.html>. (Zugriff: 01.04.2018).
- [9] <https://www.movable-type.co.uk/scripts/latlong.html>. (Zugriff: 28.03.2018).
- [10] Hugues Garnier Liuping Wang. *System Identification, Environmental Modeling, and Control System Design*. Springer, London, 2012.