Simulation einer individuenbasierten Population anhand eines Rattenmodells

Diplomarbeit der Fakultät für Biologie
der Eberhard Karls Universität Tübingen

vorgelegt von
Hurlebaus, Christine

Tübingen, April 2010

Simulation einer individuenbasierten Population anhand eines Rattenmodells

Diplomarbeit der Fakultät für Biologie
der Eberhard Karls Universität Tübingen
vorgelegt von

Hurlebaus, Christine

Tübingen, April 2010

Erstgutachter:
Prof. Dr. Hanspeter A. Mallot
(Biologie)
LS für Kognitive Neurowissenschaften
Universität Tübingen

Zweitgutachter:
Prof. Dr. N. Michiels
(Biologie)
LS für Evolutionsökologie der Tiere
Universität Tübingen

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum Unterschrift

Zusammenfassung

Die hier vorgestellte Arbeit beschreibt die Programmierung eines von simulierten Ratten ('Agenten') bewohnten Baus. Es wurde eine modulare und erweiterbare Simulation entwickelt, mit deren Hilfe Tiermodelle von unterirdisch lebenden Tieren implementiert werden können und verschiedene Einflussfaktoren untersucht werden können, denen eine Tierpopulation unterliegt. Die Agenten selbst sind als begrenzte Individuen konzipiert, die auf der Grundlage ihrer eigenen Bedürfnisse agieren, wobei Zufallsprozesse mit eingebunden werden. Die Umwelt der Simulation besteht aus dem Bau und einer nicht näher ausgeführten Umgebung des Baus. Für die Simulation wurde in einem ersten Schritt eine Bewegungsstrategie entwickelt, die ein Zusammenleben mehrerer Agenten in einem Bau ermöglicht. Hierzu musste eine zielgerichtete Wegfindung und eine sinnvolle Ausweichstrategie der Agenten bewerkstelligt werden, da ein Tunnelsystem nur begrenzt Raum bietet und sich daher nur eine begrenzte Anzahl von Tieren gleichzeitig darin aufhalten kann. Anschließend wurden weitere einfache Verhaltensweisen implementiert und unterschiedlichen Verhaltensstrategien getestet. Den Agenten wurden verhaltensbiologische Daten von Ratten zu Grunde gelegt.

Es wurde versucht Zusammenhänge zwischen dem Bau, dem Rattenverhalten und der gesamten Population bei unterschiedlichen Verhaltensstrategien aufzudecken. Diese Zusammenhänge sollten mit Hilfe zweier verschiedener Futtereintragestrategien getestet werden. Bei der ersten Strategie sollten sich die Agenten möglichst viel innerhalb des Baus aufhalten das heißt die simulierten Ratten sollten den Bau nur zur reinen Futtersuche verlassen. In der zweiten Futtereintragestrategie sollte der Bau möglichst lang verlassen werden. Die Fortpflanzung stellt eine weitere unabdingbare Voraussetzung des Überlebens einer Population dar. Für die einzelne Ratte bedeutet eine Trächtigkeit eine Belastung. Dem wurde in der Simulation durch einen höheren Energieverbrauch der trächtigen Agenten Rechnung getragen. Es wurde untersucht, welchen Einfluss dies für die gesamte Population hatte. Im nächsten Schritt wurde ein Prädator eingeführt. Hierzu wurde ein Prädatorenangriff außerhalb des Baus simuliert. Das führt dazu, dass Agenten, die sich länger außerhalb vom Bau aufhalten, häufiger einem Fressfeind zum Opfer fallen. Der Einfluss des Prädators wurde unter den beiden genannten Futtereintragsstrategien untersucht.

Danksagung

Zuallererst möchte ich mich bei Prof. Mallot vom Lehrstuhl Kognitive Neurowissenschaften für die Möglichkeit zur Erstellung dieser Diplomarbeit und seine oftmals sehr hilfreichen und kritischen Bemerkungen und Ideen bedanken. Außerdem möchte ich mich bei Prof. Michiels vom Lehrstuhl für Evolutionsökologie der Tiere bedanken, der sich zur Korrektur dieser Arbeit bereiterklärt hat. Ein besonderer Dank geht an meine Eltern und meine Familie, die mich bei meinem Studium immer unterstützt haben, immer ein offenes Ohr für meine Ideen in diesem Projekt hatten und mir Zuversicht bei mancher frustrierenden Bugsuche gaben. Zum Schluss ein Dank an alle Mitarbeiter am Lehrstuhl für Kognitive Neurowissenschaften, die durch die Schaffung einer angenehmen Arbeitsatmosphäre die Arbeit angenehm gemacht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die hier vorgestellte Arbeit beschreibt die Programmierung eines von simulierten Ratten ('Agenten') bewohnten Baus. Mit Hilfe des implementierten Programms können Untersuchungen darüber angestellt werden, in welcher Form die Größe bestimmter, zu definierender Parameter (Anzahl der Agenten im Bau, die benötigte oder zur Verfügung stehende Futtermenge, die Vermehrungsrate, die Bewegungsstrategie etc.) die Entwicklung einer Population dieser simulierten Tiere beeinflusst.

Hintergrund der Arbeit ist eine Kooperation des Lehrstuhls für Kognitive Neurowissenschaften an der Universität Tübingen und der Abteilung Distributed Systems der Rheinisch-Westfälischen Technischen Hochschule Aachen, die die Entwicklung eines Systems zum Ziel hat, mit dem Wanderratten in ihrer natürlichen Umgebung beobachtet werden können. (Projekt 'ratpack') Im Rahmen des Projektes sollen die Bewegungen und das Verhalten von Ratten auch unterirdisch verfolgt werden. Hierzu sollen Ratten mit Sensoren ausgestattet werden, die Daten bezüglich des Aufenthaltes, der Bewegungen und der Kommunikation der Ratten sammeln. Mit diesen Daten soll via 'Dead Reckoning', also einer Ortsbestimmung durch Messung der Richtung einer Bewegung, der Geschwindigkeit und der Zeit, das Tunnelsystem der Ratten rekonstruiert und dessen genaue Nutzung aufgezeichnet werden. Die dazu entwickelten Sensorknoten werden am Rücken einer Ratte befestigt. Die darin enthaltenen Messsysteme (Beschleunigungssensoren, Gyroskope und Magnetometer) liefern Daten, die bis zur Weiterleitung vorverarbeitet und zwischengespeichert werden. Außerdem können von den Sensorknoten, die zusammen ein 'drahtloses, dynamisches Netz' bilden, Daten an Basisstationen weitergeleitet werden, die an den Ausgängen des Rattenbaus positioniert werden sollen. Dies reduziert die zur Zwischenspeicherung anfallende Datenmenge. Da die Sensorknoten immer nur Funkkontakt haben wenn sich zwei Ratten in direkter Nähe befinden, gibt die zeitliche Variation der Verbindungen im Netzwerk Aufschluss über die Häufigkeit und Art der Begegnungen zwischen den Tieren und lässt damit später Schlüsse auf ihr Sozialverhalten zu. Hierzu wird zusätzlich ein System entwickelt, das Rattenlaute klassifiziert und die Klassifikationsergebnisse ebenfalls auf dem Sensorknoten speichert und weiterleitet [P.08]. Für die Rekonstruktion des Baus aus den Daten von

Sensorknoten musste unter anderem ein neuartiger Algorithmus zur Rekonstruktion dreidimensionaler Karten aus Bewegungsdaten entwickelt werden [Fab09]. Damit ist auch eine Lokalisierung und Verfolgung einzelner Ratten möglich. Zur Prüfung des entwickelten Verfahrens wurde in der o.g. Arbeit von Gregor Fabritius [Fab09] eine einfache Simulation entwickelt, in der sich Agenten in einem Bau zufällig bewegen. Die Agenten wurden jeweils mit einem 'virtuellen' Sensorknoten versehen, der ähnliche Daten generiert, wie sie der entwickelte Sensor aufnehmen wird. Diese aus der Simulation gewonnenen Daten sollen der Überprüfung der Algorithmen zur Rekonstruktion von Rattenbauen dienen. Gemäß der Zielsetzung des Programms ist die Bewegungsstrategie der Agenten sehr einfach implementiert, so benutzen sie immer den kürzesten Weg zu ihrem Ziel. Ist der Weg versperrt, beispielsweise durch einen anderen Agenten, gehen sie zu einem anderen zufälligen Ort im Bau. Außerdem können die simulierten Tiere fressen, den Bau betreten und verlassen und beim Betreten eventuell Futter mitführen. Weitere Eigenschaften haben die Agenten nicht. Es besteht kein Selektionsdruck, die simulierten Tiere können weder verhungern noch sich vermehren. Während der Programmierung dieses Testprogramms kam die Idee auf, eine komplexere, erweiterbare Simulation zu entwickeln, mit deren Hilfe genauere Tiermodelle implementiert werden können und mit deren Hilfe verschiedene Einflussfaktoren untersucht werden können, denen eine Tierpopulation unterliegt.

Im Rahmen der hier vorgestellten Diplomarbeit wurde eine solche Simulation programmiert. Ausgehend von einem Rattenmodell wurden anschließend mit Hilfe dieses Programms erste Zusammenhänge zwischen dem Bau, dem Verhalten der einzelnen simulierten Tiere und der Entwicklung der Population untersucht.

In der Simulation sollten alle Bedingungen einer 'individuenbasierten Simulation' erfüllt werden. Darunter versteht man ein Computermodell, bei dem die Aktionen und Interaktionen jedes autonomen Agenten simuliert werden und dann die Konsequenzen auf das Gesamtsystem, in unserem Fall die Population, betrachtet werden. Das Modell simuliert hierbei die gleichzeitigen Operationen und Interaktionen vieler Agenten, die Aktionen und Konsequenzen werden protokolliert. Dies erlaubt das Auftreten bestimmter Phänomene anschließend zu erklären. Dabei versucht man, von einem niedrigen Level auf ein höheres Level zu schließen. So stellt die Simulation einer einfachen Bewegungsstrategie den niedrigen Level dar, die Betrachtung der Auswirkungen auf die Population repräsentiert den höheren Level. So wäre denkbar, dass unter einer bestimmten Bewegungsstrategie die simulierte Population ausstirbt. Es wird versucht, durch einfache Verhaltensweisen der einzelnen Agenten ein komplexes System aufzubauen, das nach und nach alle für das Überleben der Gesamtpopulation erforderlichen Aspekte berücksichtigt. Die Agenten selber sind hierbei als begrenzte Individuen konzipiert, die auf der Grundlage ihrer eigenen Bedürfnisse agieren, wie etwa sozialem Status, ökonomischem Vorteil oder Reproduktionsstatus. Die hier implementierte Simulation hat folgende Eigenschaften: Es existiert eine Umwelt, in der sich die Agenten bewegen können. Diese Umwelt besteht einerseits aus dem Bau, andererseits aus einer nicht näher ausgeführ-

ten Umgebung des Baus. Das Modell verfügt über mehrere autonome Individuen, die auf Grundlage ihrer Bedürfnisse handeln. Zudem werden Zufallsprozesse mit eingebunden. So kann etwa das Geschlecht einer simulierten Ratte bei der Geburt zufällig gewählt werden oder der Angriff eines Prädators zufallsgesteuert sein, wobei die Wahrscheinlichkeit eines Angriffs mit zunehmender Aufenthaltsdauer eines Agenten außerhalb seines Baus zunimmt. Eine Reihe von Eigenschaften ist im Modell bereits vorgesehen, das Programm ist aber darauf ausgelegt, dass ohne größeren Programmieraufwand weitere Parameter eingeführt werden können. Desgleichen ist es möglich, andere Baue oder Erweiterungen zum bestehenden Bau einzuführen.

Mit dieser Simulation wurden anschließend Aspekte aus der Verhaltensbiologie und vor allem der Verhaltensökologie untersucht. In der Verhaltensbiologie werden die Verhaltensweisen von Tieren und Menschen beobachtet und beschrieben. Verhaltensökologen versuchen zu erklären, wie durch Entstehung bestimmter Verhaltensweisen im Verlauf der Stammesgeschichte die Überlebenschancen und der Vermehrungserfolg der Individuen und ihrer Art gestärkt werden. Hierzu werden auch Vergleiche zwischen den Individuen verschiedener Arten gezogen [Kre93].

In der Ethoökologie wird untersucht, welche Rolle ein bestimmtes Verhalten in einer bestimmten Umwelt für das Überleben der Individuen und für deren Vermehrung beziehungsweise für die Ausbreitung ihrer Art spielt. Als Basis dieser Forschungen gilt die Evolutionstheorie, die besagt, dass die Angepasstheit eines Merkmals an die Umwelt des Individuums letztendlich das Ergebnis einer Selektion ist, die zur Erhöhung der Überlebenswahrscheinlichkeit einer Art führt. Ein grundlegendes Verhalten, das das Überleben einer Art hierbei sichert, ist eine erfolgreiche Futtersuche. MacArthur und Pianka [MP66] entwickelten hierzu erstmals ein theoretisches und empirisches Konstrukt, die *optimal foraging theory* ('Theorie der optimalen Nahrungssuche'). Diese Theorie geht davon aus, dass die Individuen diejenigen Nahrungsquellen bevorzugen, welche bei einem geringstmöglichen Energieaufwand die größtmögliche Energieausbeute bieten.

In der hier vorgestellten Diplomarbeit sollen einige Aspekte unterschiedlicher Verhaltensstrategien untersucht werden. Für die Simulation wurde in einem ersten Schritt eine Bewegungsstrategie entwickelt, die ein Zusammenleben mehrerer Agenten in einem Bau ermöglicht. Hierzu musste eine zielgerichtete Wegfindung und eine sinnvolle Ausweichstrategie der Agenten bewerkstelligt werden, da ein Tunnelsystem nur begrenzt Raum bietet und sich daher nur eine begrenzte Anzahl von Tieren gleichzeitig darin aufhalten kann. Anschließend wurden weitere einfache Verhaltensweisen implementiert und die unterschiedlichen Verhaltensstrategien getestet. Bei der Entwicklung des Programms wurde darauf geachtet, das Programm möglichst modular und erweiterbar zu gestalten, so dass verschiedene Tierarten, Verhaltensweisen und Umgebungen leicht in das Programm implementiert werden können. Für die in dieser Arbeit durchgeführten Versuche wurde ein Rattenbau benutzt. Der implementierte Bau entspricht dem Grundriss eines tatsächlichen, ausgemessenen Rattenbaus [Cal63]und den Agenten wurden verhaltensbiologische Daten

von Ratten zu Grunde gelegt; so floss Wissen hinsichtlich der von Ratten benötigten Futtermengen, ihrem Schlafbedürfnis und bezüglich ihrer Fortpflanzungsraten mit ein. Die simulierten Ratten können sich selbstständig im Bau bewegen, außerhalb des Baus Futter suchen, dieses fressen oder im Bau einlagern und sie suchen gesteuert von ihrem jeweiligen Ruhebedürfnis einen Schlafplatz auf. Programmtechnisch sind diese Verhaltensweisen durch die den Agenten möglichen Aktionen abgebildet. Mit diesen Voraussetzungen wurden folgende Simulationsstudien durchgeführt: Es wurde untersucht, welche Auswirkungen verschiedene Verhaltensweisen der Agenten unter der implementierten Bewegungsstrategie haben. Betrachtet wurde beispielsweise die Baunutzung, d.h. welche Teile des Baus besonders häufig und welche selten aufgesucht werden. Es sollte versucht werden Zusammenhänge zwischen dem Bau, dem Rattenverhalten und der gesamten Population bei unterschiedlichen Verhaltensstrategien aufzudecken.

Die Art der Futtersuche beeinflusst das Überleben einer Population. Die Tiere benötigen eine ausreichende Futtermenge, das Futter kann dabei direkt vor Ort gefressen werden, oder es werden Vorräte innerhalb des Baus angelegt. Futtersuche beinhaltet für freilebende Ratten ein Risiko einem Fressfeind zum Opfer zu fallen. Verbleibt eine Ratte möglichst viel in ihrem Bau, kann sie hier von ihren Futtervorräten leben und minimiert das eigene Risiko. Andererseits erfordert eine größere Anzahl an Ratten innerhalb des Baus eine zusätzliche Beanspruchung des Baus, der dann eventuell unter der angenommenen Bewegungsstrategie nicht mehr ausreichend Platz für die gesamte Population bietet. Diese Zusammenhänge sollten mit Hilfe zweier verschiedener Futtereintragestrategien getestet werden. Bei der ersten Strategie sollten sich die Agenten möglichst viel innerhalb des Baus aufhalten das heißt die simulierten Ratten sollten den Bau nur zur reinen Futtersuche verlassen. In der zweiten Futtereintragestrategie sollte der Bau möglichst lang verlassen werden.

Die Fortpflanzung stellt eine weitere unabdingbare Voraussetzung des Überlebens einer Population dar. Für die einzelne Ratte bedeutet eine Trächtigkeit eine Belastung. Dem wurde in der Simulation durch einen höheren Energieverbrauch der trächtigen Agenten Rechnung getragen. Es wurde untersucht, welchen Einfluss dies für die gesamte Population hatte. Im nächsten Schritt wurde der angesprochene Prädatordruck betrachtet. Hierzu wurde ein Prädatorangriff außerhalb des Baus simuliert, die Wahrscheinlichkeit eines Angriffs stieg dabei mit der außerhalb vom Bau verbrachten Zeit an. Das führt dazu, dass Agenten, die sich länger außerhalb des Baus aufhalten, häufiger einem Fressfeind zum Opfer fallen. Der Einfluss des Prädators wurde unter den beiden genannten Futtereintragsstrategien untersucht. All die durch die beschriebenen Simulationen gewonnenen Daten können in einem zweiten Schritt mit den Daten verglichen werden, die durch das Ausrüsten von wildlebenden Ratten mit Sensorknoten gewonnen werden.

1.1 Gliederung

Die vorliegende Diplomarbeit gliedert sich in mehrere Abschnitte:

In Kapitel 2 werden die Verhaltensweisen und die Lebensbedingungen der real existierenden Art *Rattus norvegicus* beschrieben und einige Studien vorgestellt, auf deren Ergebnisse sich die vorgestellte Simulation stützt. In Kapitel 3 wird das Programm zur Simulation beschrieben. In 3.1 werden die verwendete Programmiersprache und die Klassenbibliotheken kurz vorgestellt. Der nächste Abschnitt beschäftigt sich mit der Implementierung eines Baus. Es wird erklärt, wie die Struktur eines Baus simuliert werden kann und welche Besonderheiten der hier verwendete Bau aufweist. Anschließend wird der Programmaufbau der Simulation erklärt mit allen wichtigen implementierten Objekten und den Aktionen der Agenten, die Bewegungsstrategie und die hierfür verwendeten Algorithmen werden beschrieben und die unterschiedlichen Futtereintragestrategien vorgestellt. Zum Schluss werden die Methoden zur Datengewinnung dargelegt. In Kapitel 4 werden die Ergebnisse der unterschiedlichen Simulationen beschrieben und diese werden in Kapitel 5 diskutiert. Außerdem wird ein Ausblick über weitere mögliche Erweiterungen des Programms gegeben.

Kapitel 2

Rattus norvegicus

Innerhalb der Simulation soll eine vereinfachte Darstellung einer Rattenpopulation ermöglicht werden. Hierbei wurde Bezug auf bereits existierende Studien über die Ratte *Rattus norvegicus* genommen. In diesem Abschnitt soll nun ein kurzer Überblick über die Lebensweise dieses Tieres aufgezeigt werden.

2.1 Ökologie

Die Wanderratte (*Rattus norvegicus*) gehört zur Ordnung der Nagetiere (*Rodentia*) und hier zur Familie der Altweltmäuse (*Murinae*). Das ursprüngliche Verbreitungsgebiet der Wanderratte umfasste den Südosten von Sibirien, den Nordosten Chinas und Teile von Japan [Rue10], wo die Tiere noch in Erdbauten leben. Die meisten Rattenarten leben zum überwiegenden Teil in Wäldern. Ihre Lebensräume können von tief gelegenen Regenwäldern bis Gebirgswäldern variieren, die meisten Arten meiden die Nähe des Menschen. Einige Rattenarten, zu der auch *Rattus norvegicus* zählt, haben sich dagegen als Kulturfolger an die Nähe des Menschen angepasst und finden sich in Häusern und Feldern und anderen landwirtschaftlich genutzten Flächen wieder. Wann die Wanderratten sich dem Menschen anschlossen, ist unsicher; es mag vor einigen tausend, vielleicht aber auch nur vor mehreren hundert Jahren der Fall gewesen sein. Vermutlich sind Wanderratten schon während des Mittelalters nach Europa eingedrungen. Geschichtlich belegte Erstbeobachtungen liegen allerdings erst aus dem achtzehnten Jahrhundert vor, so beispielsweise 1730 aus England, 1735 aus Frankreich, 1750 aus dem östlichen Deutschland und 1800 aus Spanien [Grz00]. Bereits im Jahr 1755 wurde die Wanderratte von Schiffen nach Nordamerika eingeschleppt. Der Schiffsverkehr ist wahrscheinlich auch für die weltweite Verbreitung der Ratten verantwortlich so dass sie heute fast in jeder Hafenstadt zu finden sind [Grz00]. In der Nähe des Menschen bildete sich ein Zusammenleben von Ratten und Menschen aus. Ratten ernähren sich omnivor und können eine große Vielzahl an pflanzlicher und tierischer Nahrung zu sich nehmen. Sie fressen Samen, Körner, Nüsse und Früchte, ergänzen den Speiseplan aber auch

mit Insekten und anderen Kleintieren wie Vögeln und deren Eiern, kleinen Säugetieren und anderen Wirbeltieren und Fischen. In der Nähe des Menschen finden die Tiere ihre Nahrung häufig in Vorratslagern, auf Feldern oder im Abfall. Der Mensch bietet den Tieren in seinen Häusern und Städten Nahrung, Wärme und Schutz vor Prädatoren. Dies führte zur Ausbildung eines Kommensalismus, dem Zusammenleben zweier artfremder Organismen, von dem die Ratte als Kommensale bis heute profitiert [Han10]. Ratten folgten dem Menschen hierbei über die gesamte Welt und kommen heute als eingeschleppte Rasse weltweit vor. Ihre Lebensform ist allerdings rein terrestrisch, im Gegensatz zu *Rattus rattus* einem anderen Kulturfolger, der auch auf Bäumen leben kann und auch als Schiffsratte bekannt ist. Das natürliche Vorkommen wilder Ratten ist daher heutzutage primär in der Nähe des Menschen auf Bauernhöfen, in Städten, auf Müllhalden und in Kanalisationen.

Ihre unterirdische Lebensweise wurde besonders gründlich von John B. Calhoun untersucht [Cal63]. Er beobachtete mehrere Rattenpopulationen über zwei Jahre in einer semi-natürlichen Umgebung. Es wurden mehrere Tiere auf einem ca. 900 qm großen Areal ausgesetzt und ihr Verhalten analysiert. In der Mitte des Areals wurden den Tieren Futter und Wasser angeboten. Außerdem verhinderte ein Maschendrahtzaun einerseits die Abwanderung der Tiere, andererseits Angriffe von Prädatoren. Die Tunnelsysteme der Ratten wurden genau untersucht, ausgegraben und vermessen. Im Freien gegrabene Rattenbaue haben einen oder mehrere Öffnungen, zum Teil lange verzweigte Gänge, Schlafkammern und Vorratslager, sowie blind endende Gänge, die bei Gefahr als letzte unterirdische Zuflucht dienen können. Calhoun beobachtete und protokollierte die Entstehung eines Bausystems genau. Er stellte fest, dass der Grundstein eines Tunnelsystems oftmals von einem trächtigen Rattenweibchen gelegt wird. Der Bau beginnt mit dem Ausgraben einer Nisthöhle. Zunächst wird hierbei ein Tunnel gegraben der in einem weiteren Schritt dann zu einer Höhle erweitert wird. Diese Höhle wird dann mit Nistmaterial aus der Nähe des Baus ausgelegt. Außerdem kann eine komplexe Nistkugel aufgebaut werden, in die dann der Wurf von 4-12 Jungen gelegt wird. Nach der Geburt wird ein weiterer Tunnel zur Erdoberfläche gegraben. Dies kann den Tieren als zweiten Fluchtweg dienen und ermöglicht bei Erweiterungen des Baus einen leichteren Transport der Erde nach außen. Kurz vor Beendigung der Säugephase wird der Bau dann mit einem Tunnel und einer weiteren Kammer vergrößert, die vor allem als Vorratslager dient. Der so entstandene Bau wird dann von der Ratte und ihren Jungtieren erweitert und kann sich mit der Zeit zu einem großen und komplexen Rattenbau entwickeln. Calhoun untersuchte viele verschiedene Tunnelsysteme. Mittelwerte über 44 vermessene Tunnelsysteme zeigten im Mittel 5,5 erwachsene Bewohner 6,8 Ausgänge, 4,5 Kammern und 16,2 Tunnelsegmente, die die Ausgänge und Kammern in einem Netzwerk miteinander verbanden. Bei den Ausgrabung stellte er außerdem fest, dass die Tiefe der Kammern, die oft tiefer als andere Tunnelsegmente liegen, auf flachem Gelände über 30 cm betragen konnte, im Mittel allerdings eine Tiefe von rund 22 cm aufwiesen. Die Kammern boten genügend Platz für 3 - 11 Ratten.

Außerdem wurden die Tiere außerhalb des Baus genau beobachtet und festgestellt, dass sie sich häufig auf festen Routen bewegen. Diese Wege wurden von mehreren Ratten benutzt und dienten als Zugangswege zum Futter und Wasser. Hierbei zeigte sich, dass die Ratten ihre Nahrung in der Regel zum Bau trug um sie dort zu fressen.

2.2 Sozialverhalten

Wanderratten leben in Kolonien mit mehreren Tieren. Im Extremfall können sich Kolonien mit mehreren hundert Individuen entwickeln. Ist die Populationsdichte gering bilden sich kleine Kolonien mit einem Rattenbock und mehreren Weibchen und ihren Jungtieren aus. Das Revierverhalten ist ausgeprägt. In diesen Kolonien ist der Rattenbock der einzige Erzeuger der Jungtiere [Bar58]. Das Revier wird gegen andere männliche Ratten verteidigt [Lot84].

Steigt die Populationsdichte hingegen an, leben mehrere Rattenmännchen innerhalb eines Baus [Moo99]. Es kommt zur Entwicklung einer Hierarchie. Es bildet sich ein weibliches und ein männliches dominantes Tier heraus, dem sich die anderen Rattenweibchen und Rattenböcke unterordnen. Die Hierarchie der Männchen ist stärker ausgeprägt und Rangkämpfe werden aggressiver ausgetragen [Moo99]. Auch das Fortpflanzungsschema ändert sich: herrschte zuvor ein polygynes System, wandelt es sich unter diesen Bedingungen in ein polygynadrisches System. Die Räume in denen die Wanderratte tätig ist, werden in Form und Größe meist vom Nahrungsangebot bestimmt. Dieser Aktionsraum kann sich allerdings vom Revier der Ratten unterscheiden. Das Revier ist das unmittelbare Wohngebiet eines Tieres oder einer Kolonie und wird im Gegensatz zum Aktionsraum gegen fremde Artgenossen verteidigt. Der Aktionsraum der Tiere erscheint sehr unterschiedlich zu sein. Es zeigt sich, dass Kolonien innerhalb von Städten oftmals sehr viel kleinere Aktionsräume aufweisen als Kolonien in ländlicheren Gebieten. So zeigen Ratten in der Stadt oftmals einen Aktionsraum von lediglich 25 bis 150 m ([DDE48],[TTM$^+$06],[GSNF$^+$09]); Ratten in ländlichen Gegenden zeigen hingegen einen Aktionsraum von 260 bis über 2000 m ([KD78],[TK78],[MD95]).

2.3 Vermehrung und Tod

Für eine Populationsstudie ist die Vermehrungsrate der Population außerordentlich wichtig. Jungtiere sind nicht sofort nach der Geburt zeugungsfähig. Entsprechend ihrem Geschlecht ergeben sich Unterschiede hinsichtlich der Zeit bis zur Geschlechtsreife: Die erste Hitze der weiblichen Ratte tritt im Alter von 34 - 38 Tagen auf ([EHPSDvdW00]). Rattenböcke sind erst im Alter von 39-47 Tagen zeugungsfähig. Rattenweibchen haben einen Östruszyklus von 4 - 6 Tagen ([PD93], [Kri99]). Die Hitze hat eine Dauer von wenigen Stunden. In dieser Phase ist das

Rattenweibchen fruchtbar und sucht aktiv nach Partnern außerhalb des Baus. Lebt das Rattenweibchen in einer polygynen Kolonie kommt es zu mehrmaligen Verpaarungen mit einem Männchen. Lebt das Rattenweibchen hingegen in einer polygynadrischen Kolonie paart sich das Rattenweibchen mit möglichst vielen Rattenböcken. Es bilden sich größere Gruppen von Rattenböcken um ein Rattenweibchen, das sich in Hitze befindet. Es scheint dabei wenig Aggressivität zwischen den männlichen Bewerbern zu geben, vielmehr paart sich das Rattenweibchen mit allen Bewerbern [Moo99]. Die Wahrscheinlichkeit dass eine Ratte trächtig wird ändert sich mit ihrem Alter [Kin39]. Die Tragezeit von Ratten beträgt rund 21 Tage [Moh74]. In der Regel bekommen Ratten 3-11 Junge pro Wurf, die Wurfgröße ist dabei abhängig vom Alter der Mutter [PD93, Moh74]. Der Wurf wird in einer Kammer zur Welt gebracht, die zuvor mit Nistmaterial ausgestattet wird. Dazu wird oftmals eine komplexe Nisthöhle aus Blättern und Zweigen gebaut, die den Jungtieren einen zusätzlichen Schutz und Wärmeisolation bietet [Cal63]. Während der Trächtigkeit und des Säugens besitzen Ratten einen erhöhten Energieverbrauch [MW81], den sie während dieser Zeit stillen müssen. Um diesen erhöhten Energieverbrauch stillen zu können, legen einige weibliche Ratten einen Futtervorrat in einer benachbarten Kammer an.

Die Jungtiere sind typische Nesthocker, blind und hilflos, mit nackter, rosaroter faltiger Haut. Im Alter von fünfzehn Tagen öffnen sie die Augen, sind voll behaart und beginnen die nähere Umgebung zu erkunden. Mit 22 Tagen verlassen sie den Bau. Die Männchen sind im Alter von drei Monaten fortpflanzungsfähig, die Weibchen meist etwas später [EHPSDvdW00].

Im Käfig lebende Wanderratten können bis zu siebenmal im Jahr Junge zur Welt bringen. Da sich die Fortpflanzungszeit im Freiland allgemein auf die warme Jahreszeit beschränkt, sind dort so viele Geburten allerhöchstens in sehr günstigen Fällen möglich. Ein im Käfig gehaltenes Paar von Zuchtratten kann theoretisch nach Ablauf eines Jahres mehr als achthundert Kinder und Kindeskinder hervorbringen. Im Freiland liegt diese Zahl selbst unter guten Bedingungen erheblich niedriger [Grz00]. Die Lebensspanne einer wild lebenden Ratte ist geringer als in Gefangenschaft. Die Mortalitätsspanne liegt nach 12 Monaten bei 95% [E.48]. Die Mehrzahl der Ratten scheint das erste Jahr nicht zu überleben. Hierbei fallen viele Tiere einem Prädator zum Opfer. Aber auch Krankheiten und Infektionen aus Rangkämpfen können zum Tod führen [Cal63]. Erwachsene Ratten werden in freier Wildbahn selten zu Tode gebissen. Meist flüchten diese Tiere oder verlassen die Kolonie bei zu starker Aggressivität ihnen gegenüber. Trotzdem kann es bei Ratten zu Kindstötung und eventuell anschließendem Kannibalismus kommen. Hierbei können geringe Futtervorkommen oder andersartige Stresssituationen Ratten zur Tötung eines nichtverwandten Wurfs [CB83] oder in extremen Stresssituationen auch des eigenen Wurfs veranlassen. Auch die Tötung schwacher oder kranker bzw. missgebildeter Jungtiere wurde beobachtet [TW84].

2.4 Futtereintragestrategien

Ratten verbringen viel Zeit in ihrem Bau, müssen diesen allerdings für die Futtersuche verlassen. Wenn sie Futter finden, müssen sie sich entscheiden, ob sie das Futter an Ort und Stelle verzehren oder ob sie es in den sicheren Bau bringen, um es dort zu fressen. Wird das Futter am Fundort gefressen, stillen sie sofort ihren Hunger, gehen aber ein größeres Risiko ein von Prädatoren wie Katzen, Steinmardern, Füchsen, Uhus und Waldkäuzen gefressen zu werden. Würden sie sich sofort mit dem Futter ins Nest begeben, wären sie zwar einerseits sicher, andererseits benötigt dieser Transport Zeit und Energie. Betrachtet man alleinig die Energiebilanz des Tieres, wäre es sinnvoll das Futter immer an Ort und Stelle zu verspeisen. Tatsächlich fressen Ratten im Allgemeinen kleineres Futter an der Futterstelle, während größere Futterstücke in den Bau transportiert werden [EO95]. Untersuchungen zeigen, dass die Entscheidung, ob das Futter sofort verzehrt wird oder in den Bau transportiert wird, von der Größe des Futterstücks, dessen Gewicht und von der Zeit abhängt, die dazu benötigt werden würde, das Futterstück zu fressen [EO95]. Allerdings wurden hier nur kurze Distanzen von 1,8 bis 7,2 m zu einem Unterschlupf getestet. Wie das Verhalten bei einer größeren Distanz zum Bau aussieht wurde nicht untersucht. Ratten sind auf freiem Feld angreifbarer für Prädatoren, da sie hier keine Möglichkeit haben, sich zu verstecken. Daher versucht die Ratte möglichst wenig Zeit außerhalb des Baus zu verbringen. Vor allem bei großen Futterstücken ist das Eintragen von Futter in den Bau somit von Vorteil, da hierdurch die Ratte weniger Zeit außerhalb des Nests verbringt und das Risiko gefressen zu werden, verringert wird. Die Ratte muss also zwischen sofortigem Hungerstillen und Risikovermeidung abwägen. Außerdem ist das Futtereintrageverhalten, also der Transport von Futter in ihren Bau, auch vom Hungerzustand der Tiere abhängig. Hungernde Tiere fressen mehr Futter außerhalb des Nests als gesättigte Tiere [YJ03].

Man kann daher Zusammenfassend feststellen, dass das Futtereintrageverhalten von Ratten nicht von einer einzigen Bedingung abhängt, sondern ein Zusammenspiel vieler verschiedener Parameter ist und somit situationsabhänig. Hierbei spielt unter anderem das Prädatorenvorkommen und der Hungerstatus der Ratten nachweislich eine entscheidende Rolle.

Kapitel 3

Material und Methoden

3.1 Programmiersprache und -umgebung

Zur Programmierung wurde die objektorientierte Sprache Java gewählt. Ein Vorteil von Java ist seine Plattformunabhängigkeit. Javaprogramme werden in Bytecode übersetzt und dann von der Java Laufzeitumgebung interpretiert und optimiert. Javaprogramme können somit ohne Anpassung auf unterschiedlichen Betriebssystemen, für die eine Java Virtual Maschine existiert, laufen. Sun bietet Java Virtual Maschines für die Betriebssystems Windows, Linux und Solaris an. Andere Hersteller lassen ihre Java Virtual Maschines für ihre Plattform zertifizieren, zum Beispiel die Firma Apple für Mac OS X.

Es sollen hier lediglich ein paar Vorteile dieser Programmiersprache für die Simulation genannt werden, aufgrund derer die Wahl auf diese Sprache fiel. Die wichtigsten Punkte waren hierbei: Leistungsfähigkeit, Architekturneutralität, Portabilität, Parallelisierbarkeit und Dynamik. Java ist aufgrund dynamischer Optimierungen der Virtuel Maschine eine der effizientesten Programmiersprachen und liefert Geschwindigkeiten die mit C++ oder C# Programmen vergleichbar sind. Wie bereits erwähnt, ist Java architekturneutral. Java wurde so entwickelt, dass dieselbe Version eines Programmes prinzipiell auf jeder beliebigen Computerhardware läuft, unabhängig vom verwendeten Prozessor oder anderen Hardwarebestandteilen. Zusätzlich ist Java portabel. Dies bedeutet, dass primitive Datentypen sowohl in ihrer Größe und internen Darstellung, als auch in ihrem arithmetischen Verhalten standardisiert sind. Das gilt auch für die Klassenbibliothek, die beispielsweise unabhängig vom Betriebssystem die gleiche GUI erzeugen kann. Java unterstützt zusätzlich Multithreading, also den parallelen Ablauf von eigenständigen Programmabschnitten. Hierzu enthält die Klassenbibliothek Unterstützungen für parallele Programmierung mit Threads. Java ist außerdem so aufgebaut, dass es sich an dynamisch ändernde Rahmenbedingungen anpassen lässt. Da die einzelnen Module erst zur Laufzeit gelinkt werden, können Teile der Software, wie Bibliotheken, neu ausgeliefert werden, ohne die restlichen Programmteile anpassen zu müssen. In-

terfaces können als Basis für Kommunikation zwischen zwei Modulen eingesetzt werden (näheres hierzu siehe 3.6.1), die eigentliche Implementierung kann aber dynamisch geändert werden.

Als Programmierumgebung wurde Eclipse 3.5.1.Galileo gewählt. Hierbei handelt es sich um ein Open Source Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich für Java entwickelt wurde diese Programmierumgebung mittlerweile aber auch für andere Programmiersprachen erweitert. Zur Dokumentation des Codes wurde Javadoc verwendet mit dessen Hilfe aus Java Quellcode mit eingebetteten Kommentaren HTML Dokumentationsdateien erstellt werden können.

Die Darstellung der Graphik wird mittels Java3D und Swing ermöglicht. Java3D ist eine Klassenbibliothek von Java-Klassen zur Erzeugung und Darstellung dreidimensionaler Grafiken innerhalb von Java-Anwendungen und Applets. Mit Hilfe von Java3D können Objekte dreidimensional modelliert und gerendert werden. Außerdem kann das Verhalten und die Ansicht gesteuert und verändert werden. Dies dient vor allem der Darstellung des Baus als dreidimensionales Gebilde. Bei Swing handelt es sich um eine Programmierschnittstelle und Grafikbibliothek zum Programmieren graphischer Benutzeroberflächen. Da Swing sehr flexibel, modular und objektorientiert aufgebaut ist, eignet es sich für die Entwicklung komplexer Anwendungen. Auch hier gilt die Plattformunabhängigkeit, so dass viel Entwicklungs- und Testarbeit zur Anpassung an unterschiedliche Plattformen wegfällt.

3.2 Der Bau als Simulationswelt

Unterirdisch lebende Tiere bewohnen einen Bau. Um unterirdisch lebende Agenten zu simulieren, benötigt man also eine Baustruktur, in denen sich die Agenten während der Simulation aufhalten können. Existierende Tunnelsysteme von unterirdisch lebenden Tieren haben unterschiedliche Strukturen und Funktionen. Die meisten Tunnelsysteme bestehen aus zwei unterschiedlichen Strukturen: Tunnelsegmenten und Ausgängen. Viele Tunnelsysteme weisen auch Kammern bzw. Höhlen auf. Diese können als verbreiterte Tunnelsegmente angesehen werden, die von den Tieren z.B. als Vorratskammern oder Schlafplätze benutzt werden.

3.2.1 Repräsentierende Datenstruktur

Für die Simulation wurde ein einfaches System gesucht, dass die Realität widerspiegelt, aber andererseits nicht zu viel Rechenzeit benötigt, da ein Zeitraum von mehreren Monaten simuliert werden sollte. Ein naiver Ansatz die Struktur eines Baus zu repräsentieren, ist einen dreidimensionalen Raum mit Partikeln zu füllen. Hierbei kann ein Partikel Erdreich darstellen, fehlende Partikel können dann Zwischenräume im Erdreich darstellen, die als Tunnelsystem fungieren. Dies

ermöglicht eine sehr genaue Darstellung des Baus. Allerdings sind eine Hindernisvermeidung und eine zielgerichtete Wegfindung der Agenten dann sehr kompliziert und rechenintensiv. Die Darstellung des Baus ist sehr detailreich möglich und Strukturen innerhalb des Baus, wie Kammern oder darin befindliche Strukturen wie Nester oder Futterstücke, können exakt dargestellt werden, eine Simulation über Monate ist aber in sinnvoller Rechenzeit nicht möglich. Daher wurde eine weniger rechenintensive Struktur gesucht, mit deren Hilfe dennoch wichtige Strukturen und Eigenschaften des Baus repräsentiert werden können. Ein Graph als Repräsentation des Baus erwies sich dabei als sinnvoll, da sich jeder unterirdische Bau in miteinander verbundene Tunnelsegmente unterteilen lässt. Außerdem ermöglicht die Repräsentation als Graph eine einfache und rechensparsame Implementierung von Wegfindungsalgorithmen, die benötigt werden um zielgerichtete Bewegungen der Agenten im Bau zu ermöglichen.

Ein Graph besteht aus Knoten und Kanten. Jede Kante ist die direkte Verbindung zweier Knoten miteinander. Ein Knoten ist ein Punkt im dreidimensionalen Raum, der mindestens eine Kante besitzt. Knoten dienen als Verbindungspunkt zwischen den Kanten des Tunnelsystems. Somit repräsentiert eine Kante ein bestimmtes Tunnelsegment zwischen zwei Knoten. Um das Tunnelsystem realistischer darzustellen, werden jeder Kante zusätzliche Eigenschaften gegeben. So hat eine Kante eine bestimmte Kapazität, die angibt wie viele Agenten sich gleichzeitig auf der Kante befinden dürfen. Dies entspricht einer gewissen Ausdehnung des Tunnelabschnitts. Ist die Kapazität nicht explizit angegeben, wird eine Kapazität von eins angenommen, es darf sich also nur ein Agent auf ihr befinden. Außerdem kennt die Kante alle auf ihr befindlichen Agenten und Futterstücke. Kanten können besondere Eigenschaften haben, die sie als Kammern oder Höhlen auszeichnen. Kammern haben eine höhere Kapazität und dienen den Agenten zum Beispiel als Schlafplatz und Vorratskammer. Ausgänge werden in der Simulation als Knoten repräsentiert. Über sie können die Agenten den Bau verlassen bzw. wieder in den Bau eintreten. Die Simulation der äußeren Welt wurde in dieser Simulation unterlassen. Die Agenten verlassen den Bau und betreten ihn nach einer gewissen Zeit durch einen zufällig gewählten Eingang wieder.

3.2.2 Einlesen der Baustruktur

In das Programm können unterschiedliche Baustrukturen eingelesen werden, die dann als Grundlage für die Simulation dienen. Eine Datei namens worldgraph.txt enthält die Vorlage für den aufzubauenden Bau. Das Einleseschema orientiert sich an dem Schema, dass in Gregor Fabritius Arbeit [Fab09] verwendet wurde, so dass beide Programme dieselbe worldgraph-Datei als Grundlage benutzen können. Die verwendete Datei befindet sich im Anhang 6.1.

Knoten werden wie folgt repräsentiert:

V name x y z

Hierbei gibt V an, dass es sich um einen Knoten (engl. Vertex) handelt. Als Name kann ein beliebiger String angegeben werden. Die float Werte x, y und z repräsentieren die Koordinaten im dreidimensionalen Raum. Hierbei geben der x- und der y-Wert die Position in der Fläche und der z-Wert die Tiefe an. Der Nullpunkt des Koordinatensystems ist frei wählbar.

Soll es sich bei dem angegebenen Knoten um einen Ausgang handeln, kann dies durch den Zusatz exit=true angegeben werden. Zwei Beispiele hierfür:

V A01 19 5 0.8 exit=true
V A02 20 5 0.8

Es würde sich beim ersten Beispiel dann um einen Ausgang handeln, mit dem Namen A01 und den Koordinaten (19/5/0,8). Das zweite Beispiel wäre ein Knoten mit dem Namen A02 und den Koordinaten (20/5/0,8).

Kanten besitzen immer zwei Knoten und werden wie folgt dargestellt:

E v1 v2

Hierbei gibt E an, dass es sich um eine Kante handelt (engl. Edge). Jede Kante besitzt, wie bereits erwähnt, zwei Knoten, hier v1 und v2. Soll es sich bei der angegebene Kante um eine Kammer handeln, die den Agenten als Schlaf- und Vorratskammer dient, kann dies durch den Zusatz sleep=true angegeben werden. Außerdem kann, durch den Zusatz cap=Wert, eine festgelegte Kapazität übergeben werden, beispielsweise cap=4.0. Andernfalls wird die Kapazität der Kante auf 1,0 gesetzt. Ein Beispiel hierzu wäre somit:

E A01 A02 sleep=true cap=5.0

Es handelt sich dabei um eine Kante zwischen den beiden zuvor initiierten Knoten, die eine Kapazität von 5,0 hat und den Agenten als Schlafplatz und Vorratskammer für Futter dient. Diese Kante könnte somit von bis zu 5 Tieren gleichzeitig benutzt werden.

Kommentare können mit dem Sonderzeichen # eingeleitet werden. Sie werden beim Einlesen der Datei nicht beachtet.

3.2.3 Verwendete Baustruktur

In der Simulation wurde ein real vermessener Bau benutzt [Cal63] und von Hand in einen Graphen umgewandelt. Ein Koordinatensystem wurde hierfür über die Abbildung des Baus gelegt. Ein Abstand von 1 entspricht dabei 6 cm. Da es sich bei der Abbildung lediglich um eine zweidimensionale Angabe handelt, wurde die Höhe der Koordinaten in realistischen Daten angenommen. Die Tiefe des Baus beträgt in der Simulation zwischen 0 (an den Ausgängen) und 23 cm. Dies entspricht Daten von real vermessenen Rattenbauten von Calhoun [Cal63]. Diese weisen eine mittlere Tiefe von 25,1 cm auf mit einem tiefsten Punkt bei 32 cm. Die Länge des gesamten Tunnelsystems beträgt 30 m und erstreckt sich auf einer Fläche von 4,25 m x 5,5 m.

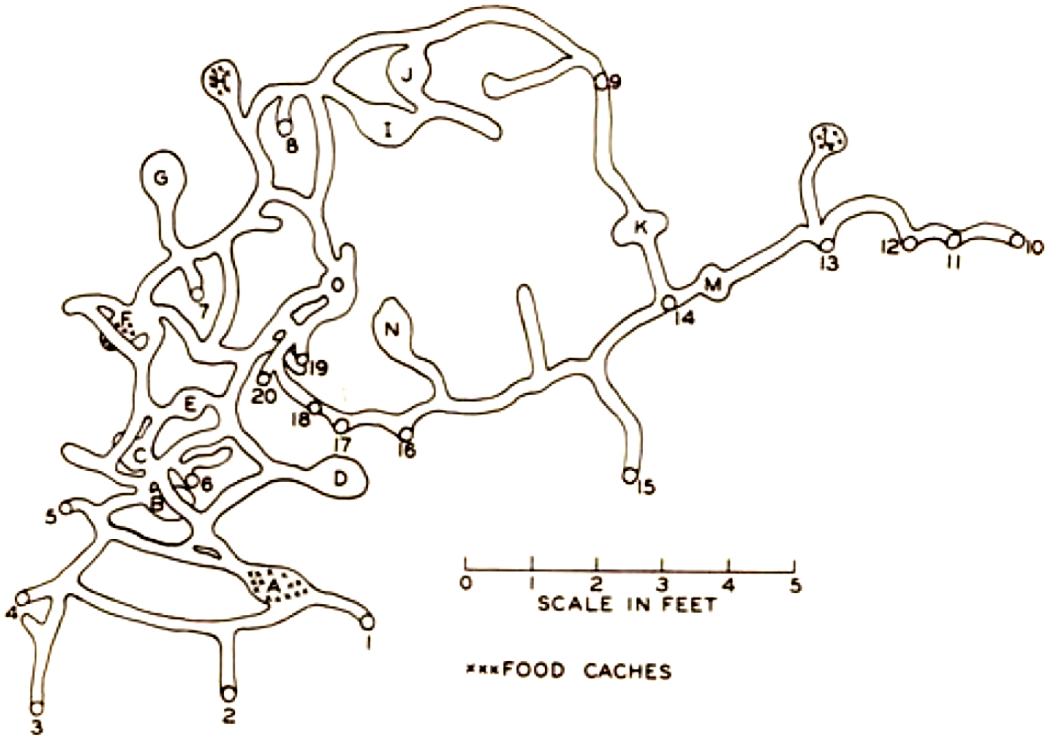

Abbildung 3.1: Karte des verwendeten Baus einer *Rattus norvegicus* Population nach Calhoun [Cal63].

Der Bau verfügt über Tunnel, Ausgänge und Kammern. Abbildung 3.1 zeigt den verwendeten Bau. Das Tunnelsystem wurde von Hand in Segmente unterteilt. Jedes dieser Tunnelsegmente repräsentiert eine Kante zwischen zwei Knoten. Um eine besonders gute Auflösung des Baus zu erreichen, wurde darauf geachtet, dass die Kanten nicht länger als 30 cm sind und im Mittel eine Länge von 18 cm aufweisen, was der Länge einer adulten Ratte in dieser Simulation entspricht. Die Tunnelabschnitte zwischen Kammern, oder Kammern und Ausgängen haben keine ausreichende Ausdehnung für zwei adulte Ratten. Somit können zwei entgegenkommende Ratten nicht aneinander vorbei. Ein normales Tunnelsegment hat eine Kapazität von eins, da im Normalfall keine zwei Ratten aneinander vorbei können. Ausnahmen hiervon sind Kanten die eine Kammer repräsentieren, deren Kapazität normalerweise höher ist. In der vorliegenden Simulation haben die verwendeten Kammern Kapazitäten zwischen 1 und 11, je nach Größe der Kammer. Dies entspricht vermessenen Kammern in realen Rattenbauten[Cal63].

3.3 Der Simulationsaufbau

Das Verhalten unterirdisch lebende Tiere kann sehr komplex und unterschiedlich sein. Trotzdem gibt es verschiedene Eigenschaften und Verhaltensweisen, die alle Tiere aufweisen. Diese sind in der Klasse Creature zusammengefasst. Zusätzlich weisen bestimmte Tierarten aber für sie spezifische Verhaltensweisen und Eigenschaften auf. Diese sollen in einer speziellen Klasse zusammengefasst werden. In der vorliegenden Simulation sollten bestimmte Eigenschaften von Ratten einfließen. Daher wurde eine Klasse Rat erstellt. Sie vereinigt alle rattenspezifischen Eigenschaften.

Die Zeit vergeht innerhalb der Simulation mit Hilfe von Ticks, die einer simulierten Sekunde entsprechen. Jeder Agent innerhalb der Simulation führt jeden Tick eine Aktion aus. Eine Aktion entspricht damit dem Verhalten des Agenten zu diesem Zeitpunkt.

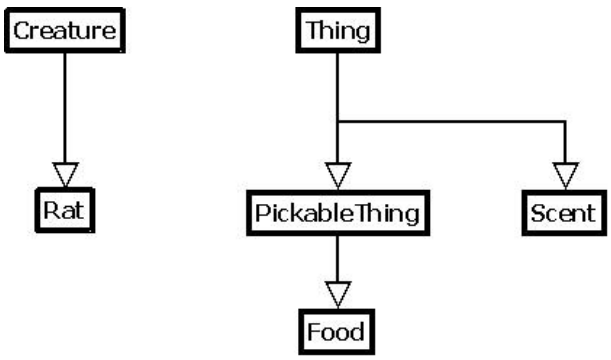

Abbildung 3.2: Vereinfachter Ausschnitt aus dem Klassendiagramm der Simulation.

3.3.1 Creature

Die Klasse Creature vereinigt die wichtigen Eigenschaften und Methoden, die jeder Agent in der Simulation besitzen soll. Es werden Eigenschaften wie Geschlecht, Alter und Position des Agenten innerhalb der Simulation festgelegt. Außerdem besitzt jedes Lebewesen mehrere mögliche Zustände. So kann ein Lebewesen mehr oder weniger hungrig sein, wach oder müde. Jeder Zustand wird durch einen Wert repräsentiert, der sich durch zeitlichen Ablauf und Ereignisse ändert. Aus diesen Zustandswerten ergeben sich für die Tiere bestimmte Bedürfnisse: Überschreitet der Wert für Hunger ein bestimmtes Maß, wird der Agent hungrig. Entsprechend gilt der Agent als müde, wenn sein Müdigkeitswert eine bestimmte Grenze überschreitet. Die Werte für Hunger und Müdigkeit steigen bisher pro Zeiteinheit an. Diese Bedürfnisse führen zu bestimmten Verhaltensweisen. Je nach Art des Lebewesens können sich die Werte, ab der eine Aktion ausgeführt wird und die daraus

folgende Aktion unterscheiden. Daher sollten diese Aktionen artspezifisch in der jeweiligen Klasse der Tierart festgelegt werden.

Es wurden grundlegende Methoden implementiert. So hat jedes Lebewesen eine Methode, mit deren Hilfe der Agent geboren wird, außerdem eine Methode, mit deren Hilfe ein Agent stirbt und aus der Simulation entfernt wird. Mit Hilfe der Methoden MakeAStep und MakeAStepWithAStar kann sich ein Agent in der Welt bewegen, dies wird im späteren Abschnitt 3.4 erläutert. Hier ist nur wichtig, dass je nach Art des Agenten, eine unterschiedliche Schrittgröße angenommen werden kann, so dass sich unterschiedliche Tierarten auch unterschiedlich schnell bewegen können. Die Methoden setzen den Agenten, wenn möglich, um die Schrittgröße vorwärts. Ist dies nicht möglich werden, ja nach Art der Creature, unterschiedliche Bewegungsstrategien sinnvoll sein. Dies kann und muss in der entsprechenden Tier-Klasse implementiert werden.

Des Weiteren kann jeder Agent Dinge aufnehmen, herumtragen und wieder an einem anderen Ort ablegen. Dies ist für die Futtersuche und das Futtereintragever-halten von vielen unterirdisch lebenden Tieren besonders wichtig. Außerdem beinhaltet die Klasse Methoden zur Futteraufnahme. Ein Agent kann eine bestimmte Futtermenge pro Zeiteinheit aufnehmen, diese wird dann vom Hungerwert abgezogen. Außerdem hat jeder Agent die Fähigkeit zu schlafen. Hierbei wird der Müdigkeitswert um einen bestimmten Wert gesenkt. Zusätzlich sind für Populationsstudien Parameter wie eventuelle Trächtigkeit oder Paarungsbereitschaft wichtig. Auch diese sind vorhanden und können ja nach Tierart unterschiedlich eingestellt werden.

3.3.2 Rat

Die Klasse Rat erbt von Creature. Dies bedeutet, dass sie alle Eigenschaften und Methoden der Klasse Creature besitzt. Hinzu kommen allerdings noch spezifische Ratteneigenschaften, die das Rattenverhalten beeinflussen.

Energieverbrauch und Schlafzyklus

Alle Objekte der Klasse Rat benötigen etwa 15000 mg Futter am Tag um ihren Hungerwert niedrig zu halten. Untersuchungen an Ratten zeigten, dass Ratten rund 5 g/100 g Körpergewicht am Tag fressen ([PD93]). Trächtige Rattenweibchen benötigen hingegen mehr Futter pro Tag ([MW81]). In der Realität benötigt eine Ratte auch nach der Trächtigkeit, während der Säugephase, mehr Futter. Da das Säugen aber in der vorliegenden Simulation nicht simuliert wird, wurde ein höherer Energieverbrauch während der Trächtigkeit angenommen. Der gesamte Energieverbrauch der simulierten Ratte entspricht dann in etwa dem Energieverbrauch einer Ratte während Trächtigkeit und Säugephase. Sie benötigen in den vorliegenden Simulationen 35000 mg am Tag. Außerdem haben diese simulierten Ratten-Objekte ein Schlafbedürfnis von rund 10 h am Tag. Pro Sekunde steigt der Müdigkeitswert

der Tiere um 0,01. Schlafen die Tiere, sinkt der Wert pro Sekunde um 0,024. Beide Parameter besitzen einen kritischen Wert, ab dem sie ein bestimmtes Verhalten auslösen: So führt ein Hungerwert über 312,5 zur Futtersuche. Ein Müdigkeitswert von über 396 führt zum Aufsuchen einer Kammer, die als Schlafplatz dient. Hier schläft die Ratte dann bis ihr Müdigkeitswert wiederum 0 erreicht hat. Ein zu hoher Hungerwert führt zum Tod der Ratte - sie verhungert. Ist der Hungerwert über 15000 +/- 10 % stirbt die Ratte und wird aus der Simulation entfernt.

Vermehrung

Für eine Populationsstudie ist die Vermehrungsrate der Population außerordentlich wichtig. Jungtiere sind nicht sofort nach der Geburt zeugungsfähig. Entsprechend ihrem Geschlecht ergeben sich Unterschiede hinsichtlich der Zeit bis zur Geschlechtsreife: Die erste Hitze tritt im Alter von 34 - 38 Tagen auf [EHPSDvdW00]. Rattenböcke sind erst im Alter von 39-47 Tagen zeugungsfähig. In der verwendeten Simulation beginnt der Ovulationszyklus der Rattenweibchen zu Beginn des 36. Tages. Ein Rattenweibchen hat einen Östruszyklus von 4 - 6 Tagen [PD93, Kri99]. Die Rattenweibchen weisen in der Simulation eine unterschiedliche Dauer des Zyklus auf. Die Dauer zwischen zwei Zyklen schwankt allerdings nur zwischen verschiedenen Agenten. Dasselbe Rattenobjekt hingegen hat immer eine identische Dauer des Zyklus. Die Hitze hat eine Dauer von wenigen Stunden. In dieser Phase suchen Rattenweibchen nach Partnern. Da die Paarung allerdings außerhalb des Baus stattfindet, wurde auf die Simulation dieser Phase verzichtet. Stattdessen hat jedes Rattenweibchen eine bestimmte Wahrscheinlichkeit trächtig zu werden, die sich mit ihrem Alter ändert [Kin39]. Entsprechend der Angaben von Mohan [Moh74] wird von einer Tragzeit von 21 Tagen ausgegangen. Die Wurfgröße wird abhängig vom Alter der Mutter bestimmt[PD93, Moh74]. Zur Geburt sucht die Ratte eine Kammer auf. Während der Trächtigkeit wird für die weibliche Ratte ein erhöhter Energieverbrauch angenommen.

3.3.3 Thing

Die Klasse Thing beinhaltet alle Dinge, die keine Lebewesen sind, aber in der Simulation auftauchen können. Beispielsweise Futter gehört hierzu. Jede Sache in der Simulation verfügt über eine Position. Außerdem stehen Methoden zur Verfügung mit deren Hilfe das entsprechende Objekt in einem Fenster sichtbar gemacht werden kann.

3.3.4 PickableThing

PickableThing erweitert Thing. Alle Objekte der Klasse PickableThing können von Agenten aufgenommen werden und an einen anderen Ort getragen werden, um dort

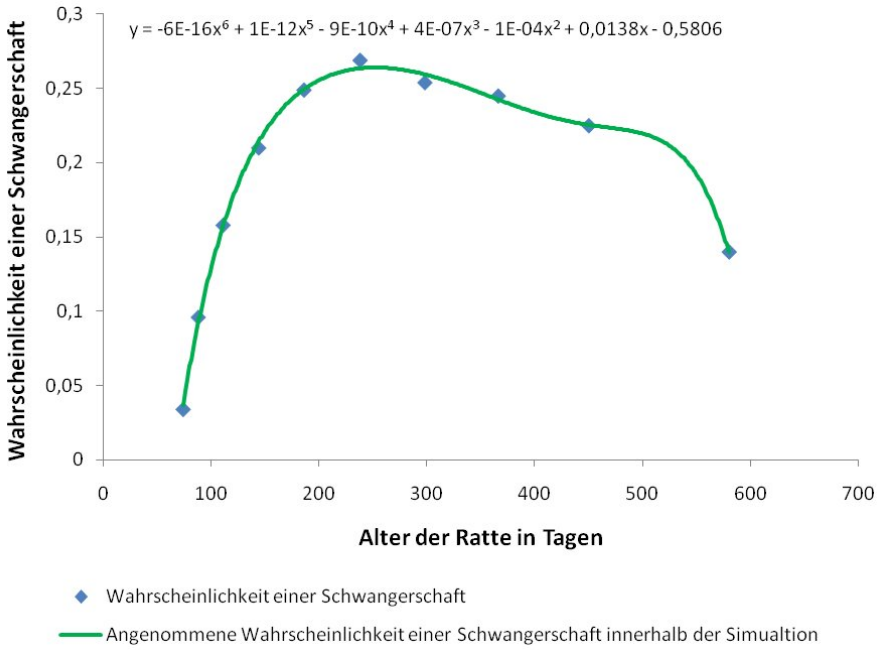

Abbildung 3.3: Wahrscheinlichkeit von Rattenweibchen in Gefangenschaft in einem bestimmten Alter trächtig zu sein nach King [Kin39] und die in der Simulation abgenommenen Wahrscheinlichkeiten trächtig zu werden.

eventuell wieder abgelegt zu werden. Futterstücke beispielsweise können transportiert werden, erben somit also von PickableThing. Duftmarken sind dagegen statisch und können nicht transportiert werden und erben somit von der Klasse Thing aber nicht von PickableThing.

3.3.5 Food

Food erbt von PickableThing und kann im Bau von Lebewesen transportiert werden. Agenten können Futter fressen um ihren Hunger zu stillen. Frisst ein Agent von einem Futterobjekt, verringert sich dessen Größe um die gefressene Menge. Entsprechend verringert sich der Hungerwert der Ratte. Dies bedeutet, dass in der jetzigen Form alle Futterstücke denselben Energiegehalt haben. Wird das Futterobjekt von einem Agenten ganz gefressen, verschwindet es aus der Simulation.

Futterobjekte können von Agenten auch außerhalb des Baus gefunden werden. Ist dies der Fall, können die Agenten das Futter in den Bau eintragen oder außerhalb des Baus fressen (siehe 3.5). Die Wahrscheinlichkeit eines Agenten, Futter zu finden, steigt mit der Aufenthaltsdauer außerhalb des Baus (siehe hierzu 3.3.6), die Futtergröße der Futterstücke innerhalb der Simulation wird als normalverteilt angenommen (näheres hierzu siehe 3.3.6). Prinzipiell ist es auch möglich, unterschiedliche Futterstücke zu bauen, zum Beispiel mit unterschiedlichen Energieniveaus.

Hierdurch könnten verschiedene Nahrungsquellen der Ratten nachempfunden werden. Auch die Wahrscheinlichkeit mit dem diese verschiedenen Futterstücke von Ratten gefunden werden, könnte angepasst werden.

3.3.6 Implementierte Verhaltensweisen

Aktion

Das Vergehen von Zeit wird innerhalb der Simulation durch Ticks repräsentiert. Ein Tick entspricht hierbei einer simulierten Sekunde. Eine Aktion ist die Tätigkeit, die ein Wesen in dieser Sekunde ausführt. Ein Agent hat mehrere mögliche Aktionen, die er ausführen kann. Jede Instanz der Klasse Aktion gehört zu einem Agenten und kennt diesen. Um eine sinnvolle Aneinanderreihung von Aktionen zu gewährleisten, aber auch die Möglichkeit zu haben in diesen Ablauf aktiv einzugreifen und trotzdem auf auftretende Ereignisse zu reagieren, besitzt jedes Objekt der Klasse Rat eine Aktionsliste. Diese Liste enthält alle Aktionen, die die Ratte plant in den nächsten Ticks auszuführen. Aktionen können dabei andere Aktionen auslösen. Diese Aktionen schreiben dann noch folgende Aktionen an den Beginn der Aktionsliste. Soll eine Aktion später ausgeführt werden, wird die Aktion ans Ende der Aktionsliste geschrieben. Auch Interaktionen mit anderen Agenten können so einfach realisiert werden. So kann eine Agent die angestrebte Aktion in die Aktionsliste des Agenten schreiben, der diese mit dem anderen Agenten ausführen möchte. Der so angesprochene Agent kann dann entscheiden, ob er die Interaktion eingeht oder nicht. Je nachdem kann die Aktion in beiden Agenten ausgeführt oder aus den Listen der Agenten wieder entfernt werden.

Ist die Aktionsliste leer, werden die Bedürfnisse des Agenten betrachtet. Ist ein bestimmter Wert über einen kritischen Punkt erhöht, wird eine entsprechende Aktion in die Aktionsliste aufgenommen. Sind alle Werte des Agenten in einem unkritischen Bereich, wählt der Agent zufällig aus einer Reihe verfügbaren Aktionen aus.

Bisher muss jede Aktion zwei Methoden besitzen: Mit der Methode executeAction wird die Aktion in der Simulation ausgeführt. Zusätzlich verfügt jede Aktion über eine toString Methode. Sie dient zum Loggen der entsprechenden Aktion, kann also zur Ausgabe der Aktion auf der Konsole oder zum Schreiben in ein Logfile benutzt werden.

RatAction

RatAction erbt von Aktion und repräsentiert Handlungen die von Ratten ausgeführt werden können. Als Aktion verfügt jede RatAction ebenfalls über eine executeAction und eine toString Methode.

In der Simulation wurden verschiedene einfache Verhaltensweisen implementiert und damit verschiedene Verhaltensstrategien getestet. Es sollte den Agenten

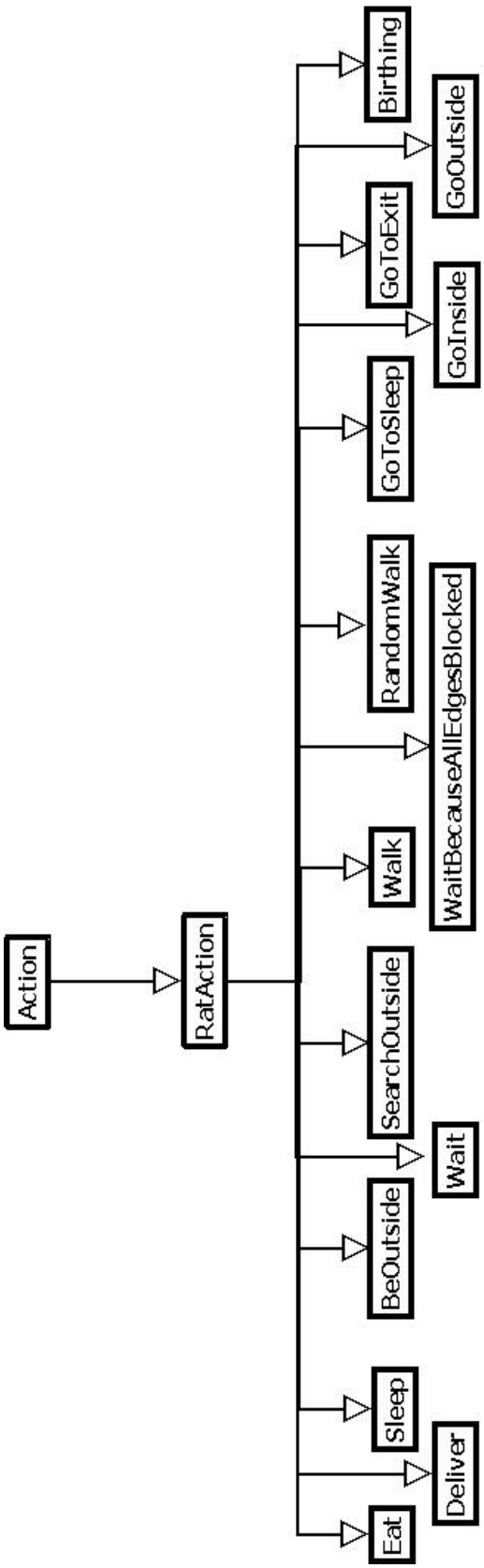

Abbildung 3.4: Vereinfachter Ausschnitt aus dem Klassendiagramm der Simulation mit den implementierten Verhaltensweisen.

ermöglicht werden sich selbstständig im Bau zu bewegen, außerhalb des Baus Futter zu suchen, dieses zu fressen oder im Bau einzulagern und selbstständig in regelmäßigen Abständen einen Schlafplatz aufzusuchen, um dort zu ruhen. Um diese Verhaltensweisen darzustellen, wurden mehrere mögliche RatActions implementiert. Diese können nach ihrer Art und Weise in unterschiedliche Bereiche eingeteilt werden.

Die Aktionen Walk und RandomWalk dienen den Agenten zur Fortbewegung innerhalb des Baus, GoToExit, GoToSleep, GoOutside und GoInside dienen zur Wegfindung und zielgerichteten Fortbewegung auf das entsprechende Ziel. WaitBecauseAllEdgesBlocked wird aufgerufen, wenn sich ein Tier fortbewegen möchte, dies aber aus Platzgründen nicht möglich ist. Das Objekt verharrt dann an seiner Position. Eat dient zur Futteraufnahme. Birthing beinhaltet das Verhalten einer weiblichen Ratte bei der Geburt und lässt einen neuen Wurf von Ratten-Objekten entstehen, Sleep wird aufgerufen wenn eine Ratte schläft, Deliver, wenn eine Ratte ein Futterobjekt in einer Kammer abliefern möchte. BeOutside, Wait und SearchOutside sind Aktionen außerhalb des Baus. Sie regeln wie lange eine Ratte sich draußen aufhält, ob sie hierbei Futter findet und was für Aktionen sie eventuell außerdem bei oder nach einem Aufenthalt außerhalb des Baus ausführt.

Ist die Aktionsliste leer, werden die Bedürfnisse der Ratte betrachtet. Ist ein bestimmter Wert über einen kritischen Punkt erhöht, wird eine entsprechende Aktion in die Aktionsliste aufgenommen. Sind alle Werte der Ratte in einem unkritischen Bereich, wählt die Ratte zufällig mit gleichen Wahrscheinlichkeiten aus den Aktionen GoOutside, Sleep oder RandomWalk. Wird GoOutside aufgerufen, sucht die Ratte Futter außerhalb des Baus und lagert dieses mit einer hohen Wahrscheinlichkeit bei erfolgreicher Suche im Bau ein, bei Sleep sucht die Ratte die nächste Kammer auf um sich etwas auszuruhen und bei RandomWalk geht die Ratte an einen zufälligen Punkt im Bau.

Eat

Diese Aktion wird aufgerufen, wenn der Hungerwert der Ratte einen kritischen Wert erreicht hat. Befindet sich die Ratte innerhalb des Baus, sucht sie das nächstgelegene Futterstück, um dieses aufzunehmen und zu fressen. Hierbei kennt jeder Agent alle Futterstücke innerhalb des Baus. Hat der Agent ein Futterstück bei sich, frisst er es. Befindet sich kein Futter im Bau, muss der Agent Futter außerhalb des Baus suchen. Es wird die Aktion GoToExit an den Beginn der Aktionsliste geschrieben, so dass die simulierte Ratte als darauffolgende Aktion den nächsten Ausgang aufsuchen wird, um hinauszugehen und Futter außerhalb des Baus zu suchen.

Außerhalb des Baus darf nicht bei jeder Teststrategie gefressen werden. Soll sich die Ratte möglichst wenig außerhalb des Baus befinden, frisst die Ratte nur kurz vor ihrem Hungertod außerhalb des Baus (Notfallessen siehe weiter unten). Ansonsten wird nur innerhalb des Baus gefressen. Hierbei ist zu beachten, dass Ratten ihr

Futter innerhalb des Baus nur in Kammern fressen. Auch hier ist das Notfallessen wiederum eine Ausnahme. In der Teststrategie, bei der sich die Ratte möglichst viel außerhalb des Baus befindet, frisst die Ratte auch außerhalb des Baus, falls sie Futter bei sich hat und der Müdigkeitswert nicht zu hoch ist. Ratten mit einem erhöhten Müdigkeitswert gehen in den Bau und schlafen und fressen dann gegebenenfalls dort. Befindet sich die Ratte außerhalb des Baus und hat einen erhöhten Hungerwert aber kein Futter bei sich, sucht sie weiter Futter außerhalb des Baus und geht nicht wieder rein. Befinden sich Ratten hingegen innerhalb des Baus und befindet sich dort Futter, fressen sie bei beiden Eintragestrategien das Futter innerhalb des Baus und verlassen den Bau dann nur, wenn sich kein Futter im Bau befindet.

Bei jedem Aufruf von Eat mit einem verfügbaren Futterstück wird pro Eat-Aktion allerdings nur eine bestimmte Futtermenge aufgenommen. Diese Futtermenge entspricht der Menge, die ein Agent innerhalb einer Sekunde fressen kann. Der Hungerwert der Ratte sinkt um die entsprechend aufgenommene Futtermenge und das Futterobjekt verkleinert sich um die verzehrte Menge. Ist der Hungerwert des Agenten nach der Aktion noch erhöht und das Futterstück noch nicht ganz verzehrt, wird eine neue Eat Aktion an den Beginn der Aktionsliste geschrieben. Hat die simulierte Ratte keinen erhöhten Hungerwert mehr, legt sie das restliche Futterstück in einer Kammer ab. Ist das Futterstück aufgegessen wird keine neue Eat Aktion von dieser Aktion aufgerufen, allerding führt eine erneute Überprüfung des Futterwertes in diesem Fall zu einer erneuten Eat Aktion, wenn nicht bereits andere Aktionen in der Aktionsliste stehen.

Die einzige Ausnahme ist das sogenannte Notfallessen: Ist ein Agent kurz vor dem Verhungern und trägt ein Futterstück bei sich, frisst er dieses auf der Stelle, egal wo er sich befindet und welche weiteren Aktionen in der Liste stehen. Erst nach dem Verzehr des Futterstücks werden weitere Aktionen durchgeführt.

Sleep

Die Aktion Sleep wird aufgerufen, wenn der Müdigkeitswert einer simulierten Ratte beim Überprüfen ihres Zustands zu hoch ist. Die Ratte geht zum nächsten Schlafplatz und schläft dort. Befindet sich die Ratte in einer Kammer, die als Schlafplatz dient, wird bei jedem Aufruf der Sleep Aktion der Müdigkeitswert der simulierten Ratte verringert. Ist der Müdigkeitswert des Agenten nach der Aktion größer als Null, wird eine neue Sleep Aktion gestartet. Simulierte Ratten schlafen also immer bis ihr Müdigkeitswert wieder bei Null ist.

GoToExit

Diese Aktion wird aufgerufen, wenn eine Ratte zu einem Ausgang gehen möchte, um den Bau zu verlassen. Die Ratte sucht den nächstgelegenen erreichbaren Ausgang und geht dort hin. Ist die Ratte am Ausgang angekommen, wird GoOutside aufgerufen.

GoToSleep

Diese Aktion wird von einer Ratte aufgerufen, wenn sie zu einer Kammer gehen möchte. Es gibt mehrere Möglichkeiten, warum eine Kammer im Bau aufgesucht werden soll: Beispielsweise kann eine Ratte Futter suchen oder sie möchte sich in einer Kammer ausruhen. Je nach Motivation der Ratte wird ein anderer Konstruktor der Klasse benutzt. Außerdem wird die Motivation der Bewegung im Rattenobjekt gespeichert. So kann die Intention der Bewegung jederzeit abgerufen werden. Eine Ratte die sich ausruhen möchte, kann jede Kammer zum Ausruhen benutzen. Sie sucht die nächstgelegene, erreichbare Kammer auf. Hingegen sucht eine Ratte auf Futtersuche lediglich Kammern auf, die Futter enthalten. In der Simulation wird davon ausgegangen, dass eine Ratte weiß, wo sich Futter befindet. Ist sie allerdings auf dem Weg zu einem Futterstück und wird dieses dann von einer anderen simulierten Ratte entfernt, stellt sie dies erst beim Erreichen des Futterplatzes fest. Außerdem werden Kammern zum Ablegen von Futtervorräten benutzt. Hat eine Ratte Futter außerhalb des Baus gefunden und möchte dies nun im Bau deponieren, sucht sie sich ebenfalls die nächstgelegene, erreichbare Kammer, da jede Kammer als Schlafplatz und Vorratskammer dienen kann. Das Ablegen von Futter in anderen Bereichen des Tunnelsystems wurde nicht implementiert, dies entspricht dem Verhalten realer Ratten [Cal63].

GoOutside

Diese Aktion wird aufgerufen wenn eine Ratte an einem Ausgang den Bau verlässt. Sie befindet sich dann außerhalb des Baus. Da die Welt außerhalb des Baus nicht mit vertretbarem Aufwand simuliert werden kann, befindet sich die Ratte anschließend auf einer Kante, die keinerlei Verbindung zum Bau hat und die Welt draußen symbolisiert.

Befindet sich die Ratte beim Aufruf dieser Aktion noch nicht auf einem Ausgang, wird die Aktion GoToExit an den Beginn der Aktionsliste geschrieben, so dass die Ratte erst zu einem Ausgang geht. Nach dem Verlassen des Baus wird die Aktion BeOutside an den Beginn der Aktionsliste der Ratte geschrieben.

BeOutside

Diese Aktion legt zum einen fest, wie lange sich eine Ratte außerhalb des Baus befindet und zum anderen, ob eine Ratte Futter gefunden hat. Die genauen Simulationsbedingungen sind einstellbar. In diesen Simulationen soll allerdings die Annahme gelten, dass die Wahrscheinlichkeit Futter zu finden mit der Aufenthaltsdauer außerhalb des Baus steigt. BeOutside schreibt hierbei eine bestimmte Anzahl an Wait-Aktionen (siehe 3.3.6) in die Aktionsliste der Ratte. Die genaue Anzahl an auszuführenden Wait Aktionen wird zufällig gewählt. Es wird angenommen, dass Ratten eine normalverteilte Zeit außerhalb des Baus aufhalten, so dass die Anzahl

an auszuführenden Wait Aktionen gaußverteilt gewählt wird. In den vorliegenden Simulationen ist die Anzahl der Wait-Aktionen normalverteilt mit einem Erwartungswert von 300 und einer Standardabweichung von 60. Dies bedeutet, dass die Ratten im Mittel 5 Minuten außerhalb des Baus bleiben. Dies wurde gewählt, unter der Annahme, dass diese Ratten einen Bereich von rund 150 m vom Bau besichtigen und hierbei eine Geschwindigkeit von rund 0,6 m/sec aufweisen. Bei diesem Modell flossen zum einen gemessene Futtereintragegeschwindigkeiten von Ratten von [NF95] ein und zum anderen gemessene Homeranges von städtischen Ratten [DDE48, TTM$^+$06, GSNF$^+$09], es entspräche somit der mittleren Aufenthaltsdauer, mit der die Ratte jeden Punkt in diesem Bereich erreichen kann, auch wenn sie nicht den direkten Weg zu einem Punkt läuft. Entsprechend kann diese Zeit als unterster Bereich für die Aufenthaltsdauer außerhalb gelten. Je nach verwendeter Futtereintragestrategie wird anschließend unter unterschiedlich eingestellten Versuchseinstellungen neu entschieden, ob die Ratte dann den Bau aufsucht oder weiterhin außerhalb des Baus bleibt. Eine minimale Dauer, nach der eine Ratte immer neu entscheidet, ob sie weiterhin außerhalb des Baus bleibt oder wieder in den Bau zurückkehrt, kann eingestellt werden. In den vorliegenden Simulationen wurde eine Obergrenze von 1 Stunde festgelegt. Entsprechend kann eine einzige BeOutside Aktion höchstens 3600 Wait Aktionen nach sich ziehen. Nach spätestens einer Stunde testet eine Ratte neu, ob sie den Bau wieder betritt oder weiterhin außerhalb des Baus bleibt. Außerdem legt BeOutside fest, ob eine Ratte Futter außerhalb des Baus gefunden hat. Die Futterfundwahrscheinlichkeit wurde hierbei mit

$$y = (0{,}6/360) * i \tag{3.1}$$

simuliert, wobei i der Aufenthaltsdauer außerhalb des Baus in Sekunden entspricht. Auch kann hier ein Prädatorenangriff simuliert werden. Die genauen Umstände, wann ein Prädatorenangriff stattfindet, sind wiederum frei wählbar. Sinnvoll erscheint die Annahme, dass die Wahrscheinlichkeit für einen Prädatorenangriff mit der Aufenthaltsdauer außerhalb ansteigt. Eine mögliche Entscheidung könnte durch folgende Überprüfung getestet werden:

$$a \leq (1/20412) * i \tag{3.2}$$

wobei a eine zufällige (float) Zahl zwischen 0 und 1,0 ist und i der Aufenthaltsdauer außerhalb des Baus in Sekunden entspricht.

Hier wäre die Wahrscheinlichkeit, einem Prädator zu begegnen gering. Begegnet eine Ratte einem Prädator, stirbt sie.

Wait

Die Aktion Wait wird aufgerufen, falls sich eine Ratte außerhalb des Baus befindet. Die Anzahl der hintereinander aufgerufenen Wait Aktionen repräsentiert die Zeit,

die eine Ratte außerhalb des Baus verbringt und wird in BeOutside bestimmt (siehe hierzu 3.3.6). Da das Verhalten außerhalb des Baus nicht simuliert werden soll macht die Ratte hier nichts. Es werden allerdings alle Listener verständigt, die die Aufenthaltsdauer außerhalb des Baus benötigen. Nachdem alle hintereinander aufzurufenden Wait-Aktionen der Aktionsliste abgearbeitet wurden, wird die Aktion SearchOutside (siehe hierzu 3.3.6) aufgerufen.

SearchOutside

Diese Aktion repräsentiert die Wahrscheinlichkeit Futter außerhalb des Baus zu finden. Hat eine Ratte Futter außerhalb des Baus gefunden (siehe hierzu 3.3.6), sind verschiedene Futterarten möglich. In der aktuellen Implementierung unterscheiden sich die gefundenen Futterstücke allein in ihrer Größe. Hier sind viele mögliche Futtergrößen und deren Wahrscheinlichkeit gefunden zu werden denkbar. So wäre es beispielsweise denkbar, dass kleinere Futterstücke häufiger vorkommen als größere. In den vorliegenden Simulationen werden die Futtergrößen als normalverteilt angenommen. Beispielsweise wurden normalverteilte Futterstücke mit einem Erwartungswert von 220 mg und einer Standardabweichung von 60 mg angenommen oder eine normalverteilte Futtergröße mit einem Erwartungswert von 250 mg und einer Standardabweichung von 60 mg. Je nach verwendeter Futtereintragestrategie (3.5) sind anschließend verschiedene Rattenaktionen möglich. Entweder geht die Ratte dann wieder in den Bau um dort Futter abzuliefern oder auf einem Schlafplatz zu ruhen, dann wird GoInside aufgerufen. Oder die Ratte bleibt weiterhin außerhalb des Baus und es folgt als nächstes die Aktion BeOutside oder, falls die Ratte Futter gefunden hat und hungrig ist, die Aktion Eat.

GoInside

Diese Aktion wird aufgerufen, wenn eine Ratte den Bau betritt. Die Ratte betritt durch einen zufälligen Ausgang den Bau. Die Wahrscheinlichkeit für jeden Ausgang ist hierbei gleich. Im Bau wird nun überprüft, ob die Ratte Futter bei sich hat. Ist dies der Fall bringt die Ratte das Futter in eine Kammer und die Aktion Deliver wird an den Beginn der Aktionsliste der Ratte geschrieben. Ist der Ausgang durch andere Ratten blockiert, bleibt sie draußen und BeOutside wird an den Beginn der Aktionsliste der Ratte hinzugefügt. Die Ratte bleibt also wieder eine zufällige Zeit außerhalb des Baus und versucht anschließend erneut den Bau zu betreten. Kann die Ratte den Bau betreten, hat aber kein Futter bei sich, wird nichts an die Aktionsliste geschrieben. Die Rattendaten werden dann überprüft und je nach Zustand der Ratte wird die nächste Aktion ausgewählt.

Deliver

Diese Aktion wird aufgerufen, wenn eine Ratte ein Futterobjekt in einer Vorratskammer ablegen möchte. Die Ratte bewegt sich zur nächstgelegenen, erreichbaren Kammer. Ist sie dort angekommen, legt sie das Futterobjekt ab.

Birthing

Weibliche Ratten können ab dem 36. Tag trächtig werden. Zur Darstellung wird ein Ovulationszyklus simuliert (vgl. 3.3.2). Die Wahrscheinlichkeit trächtig zu werden hängt hierbei vom Alter der Ratte ab. Die Daten wurden hierbei von Untersuchungen von King und Calhoun entnommen (vgl. [Cal63], [Kin39]).

Wenn eine weibliche Ratte trächtig wird und die Tragzeit überschritten ist, geht die Ratte in eine Kammer und bringt die Jungen zur Welt. Als Modell für die Wurfgröße wurden hier Untersuchungen von Mohan ([Moh74]) herangezogen. Je nach Alter der Ratte bei der Empfängnis, bekommt die Ratten zwischen 4 und 11 Jungtiere (vgl. Abb. 3.5). Das Geschlecht der Jungtiere wird zufällig gewählt, männlich und weiblich sind dabei gleich wahrscheinlich. Die Jungtiere sind sofort voll entwickelte Ratten bis auf die Ausnahme, dass sie noch nicht zeugungsfähig sind. Die Zeugungsfähigkeit erlangen die Tiere erst im entsprechenden Alter (siehe hierzu 3.3.2).

Walk

In der Simulation wird davon ausgegangen, dass sich alle Ratten mit derselben Geschwindigkeit fortbewegen. Eine Geschwindigkeit von 30 cm/sec wird hier auf Grund von Ganganalyseuntersuchungen [Sch10] angenommen. Die Aktion Walk wird aufgerufen, wenn sich eine Ratte im Bau von einer Startposition zu einem Ziel fortbewegt. Eine WalkAktion entspricht hier, wenn der Weg nicht durch eine andere Ratte blockiert ist, einem Schritt der Ratte (Aufruf der Methode MakeAStep bzw. MakeAStepWithAStar). MakeAStep wird aufgerufen, wenn eine Ratte bisher keiner anderen Ratte auf ihrem Weg begegnet ist und ihr somit keinerlei Kanten bekannt sind, die sie auf ihrem Weg zum Ziel nicht betreten darf. MakeAStepWithAStar wird hingegen verwandt, wenn eine Ratte bereits einer anderen Ratte begegnet ist und somit einen Weg suchen muss, der die bekannten blockierten Kanten nicht benutzt. Bei diesem Ausweichen wird zur Wegfindung der Algorithmus A* benutzt (siehe 3.4.2). Ist ein kompletter Schritt nicht möglich, wird die mögliche Strecke zurückgelegt. Ist das Ziel dann noch nicht erreicht, wird im nächsten Tick eine weitere WalkAktion aufgerufen.

Es gibt verschiedene Kontruktoren von Walk: Walk wird meist mehrmalig aufgerufen, da jede Walk Aktion nur einem Schritt der Ratte entspricht. Jeder Konstruktor wird in verschiedenen Situationen aufgerufen. Jede Walk-Aktion kennt die ausführende Ratte, deren Position innerhalb des Baus, zu der die Ratte gelangen

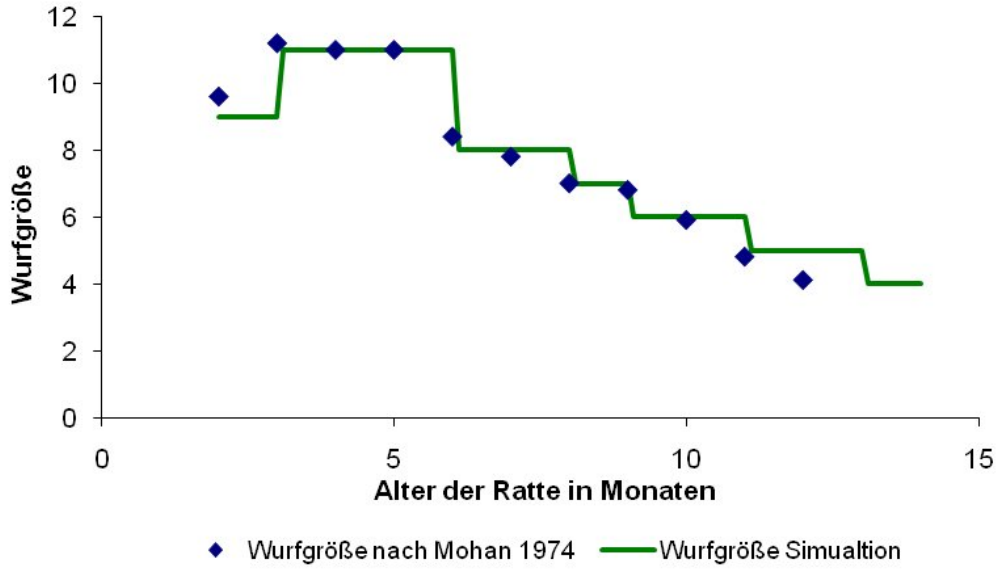

Abbildung 3.5: Wurfgrößen nach Mohan [Moh74] und die in der Simulation erzeugten Wurfgrößen

möchte, die Startposition der Ratte und den Zeitpunkt, zu dem sie sich auf den Weg zu der Zielposition gemacht hat. Je nach vorhandenen weiteren Angaben, werden unterschiedliche Methoden aufgerufen. Ein boolean gibt hierbei an, ob es sich um einen Bewegung zu einem neuen Ziel handelt. Ist dies der Fall, wird die Startposition auf die Position der Ratte gesetzt und der Zeitpunkt wird festgehalten. Zusätzlich kann eine Liste von blockierten Kanten übergeben werden. Die Ratte wird diese Kanten auf ihrem Weg zum Ziel nicht betreten. Zur Wegfindung der Ratte wurden verschiedene Wegfindungsalgorithmen implementiert, die je nach Situation benutzt werden (näheres hierzu siehe 3.4).

RandomWalk

Die Aktion RandomWalk wird aufgerufen, wenn eine Ratte keinen möglichen Weg zu ihrem ursprünglichen oder einem entsprechenden Ziel gefunden hat und nun eine Ausweichbewegung startet. Die Ratte geht zu einer zufällig im Bau gewählten Po-

sition. Ist diese Bewegung wiederum nicht möglich, wird die Aktion abgebrochen. Eine RandomWalk Aktion, entspricht dabei einem Schritt der Ratte, wenn der Weg nicht erneut durch eine andere Ratte blockiert ist (Aufruf der Methode MakeAStep bzw. MakeAStepWithAStar). MakeAStep wird aufgerufen, wenn eine Ratte bisher keiner anderen Ratte auf ihrem Weg begegnet ist und ihr somit keinerlei Kante bekannt ist, die sie auf ihrem Weg zum Ziel nicht betreten darf. MakeAStepWithAStar wird hingegen aufgerufen, wenn eine Ratte bereits einer anderen Ratte begegnet ist und somit einen Weg suchen muss, der die bekannten blockierten Kanten nicht benutzt. Ist ein kompletter Schritt nicht möglich, wird im nächsten Tick die mögliche Strecke zurückgelegt. Ist das Ziel dann noch nicht erreicht, wird eine weitere RandomWalkAktion aufgerufen. Auch hier ist der Aufruf mit verschiedenen Kontruktoren von RandomWalk möglich: Jeder Konstruktor dient hier ebenfalls denselben Unterscheidungen verschiedener Situationen wie bei Walk. Daher besitzen die Konstruktoren dieselben Parameter wie bei Walk.

WaitBecauseAllEdgesBlocked

Diese Aktion wird aufgerufen, wenn eine Ratte durch andere Ratten in ihrer Bewegungsfreiheit so eingeschränkt ist, dass sie sich nur noch auf ihrer Kante bewegen kann. Dann wartete die Ratte an ihrem Platz. Die Aktion kann sich selbst wieder aufrufen. Dies geschieht mit einer Wahrscheinlichkeit von 50%. Die Anzahl der hintereinander aufgerufenen Aktionen WaitBecauseAllEdgesBlocked variiert. Es können aber nie mehr als zehn Aktionen hintereinander aufgerufen werden. Danach wird die Aktionsliste weiter abgearbeitet. Dies bedeutet, dass eine Ratte zwischen einer und zehn Sekunden an einem Ort verharrt, bevor sie erneut versucht, ihr ursprüngliches Ziel zu erreichen.

3.4 Wegfindungsalgorithmen und Ausweichstrategien

Es gibt unterschiedliche Tierarten, die unterirdisch in größeren Populationen leben. Ihr Bau wird somit von mehreren Individuen gleichzeitig benutzt. Da die entsprechenden Tunnelsysteme aber immer über eine begrenzte Ausdehnung verfügen, kann nicht ausreichend Platz für eine beliebige Anzahl an Tieren herrschen. Somit müssen die Tiere über Ausweich- und Kollisionsvermeidungsstrategien verfügen. Das Tunnelsystem von Ratten besitzt in den meisten Tunnelabschnitten lediglich genügend Platz für eine Ratte. Ausnahmen von dieser Regel sind hierbei Kammern, die Platz für mehrere Individuen bieten. Wenn sich nun die Individuen innerhalb des Baus zielgerichtet bewegen sollen, sind unterschiedliche Ausweichstrategien denkbar. Es wäre zum einen denkbar, dass jedes Tier lediglich einen bestimmten Bereich des Tunnelsystems verwendet. Dies würde zu einer räumlichen Trennung

führen und die Tiere würden sich somit nur selten gegenseitig blockieren. Feldstudien widerlegen dies allerdings bei Ratten, da die Tiere keinerlei Präferenz für bestimmte Eingänge aufweisen [Cal63]. Ein weiteres Indiz gegen diese Bauaufteilung liefern Beobachtungen in Bezug auf Futter innerhalb des Baus. Hier scheint das Futter als allgemeiner Vorrat der gesamten Population zu dienen. Ratten zeigen keinerlei Aggressivität anderen, zur Population gehörenden Tieren gegenüber, wenn sich diese aus dem Futtervorrat bedienen, um etwas zu fressen oder dieses an einen anderen Ort zu bringen [Cal63]. Allerdings ist anzumerken, dass sich diese Beobachtungen auf Populationen mit genügend Futter beziehen. Futterknappheit wird eventuell zu aggressivem Verteidigen von Futtervorräten führen. Nimmt man an, dass alle Individuen einer Population den gesamten Bau nutzen, sind wiederum verschiedene Bewegungsstrategien möglich. Eine denkbare Ausweichstrategie könnte im Verlassen des Baus liegen. Eine Lösung von Kollisionen läge darin, dass ein Individuum, wenn es nicht ans Ziel kommt, da ein anderes Individuum ihm entgegen kommt, den Bau auf dem nächsten Weg verlässt und dann einen anderen Eingang benutzt, der eventuell näher am Ziel liegt oder bei dem die Begegnung mit einem anderen Baubewohner unwahrscheinlicher wird. Diese Strategie führt allerdings zu einer erhöhten Aufenthaltsdauer um den Bau herum. Um aber einen möglichst geringen Prädatorendruck zu ermöglichen, sollten sich die Individuen möglichst viel innerhalb des sicheren Tunnelsystems befinden. Dies führt zu einer anderen Ausweichstrategie bei der die Individuen einen neuen Weg, ohne die Benutzung des blockierten Tunnelabschnitts durch ein anderes Lebewesen, finden müssen. Um ein möglichst einfaches und realistisches Modell zu entwerfen, erscheint es sinnvoll, den jeweiligen Lebewesen möglichst wenig Wissen über andere Lebewesen im Bau zu geben. So wäre ein Ansatz, der in eine bestimmte Wegfindung alle anderen Objekte und deren Bewegungen einplant ideal. Dieser würde aber keine mit der Realität vergleichbaren Daten liefern.

In der vorliegenden Simulation wurden mehrere Annahmen und Bedingungen festgelegt, die eine sinnvolle Bewegungsstrategie der Lebewesen ermöglichen. Es wurde darauf geachtet, dass jedes Lebewesen möglichst wenig Wissen über andere Lebewesen besitzt und dieses zuvor erwerben musste z.B. durch direkte Begegnung mit einem anderen Individuum. Allerdings geht die Simulation davon aus, dass alle Lebewesen ihren Bau genau kennen. Außerdem bewegt sich jedes Lebewesen immer auf dem kürzesten möglichen Weg zu seinem jeweiligen Ziel. Ist eine Bewegung der Tiere innerhalb des Baus möglich, verlassen sie den Bau nicht.

Die Agenten befinden sich innerhalb des Baus immer auf einer Kante. Diese Kante kann jeweils nur eine bestimmte Anzahl an Agenten beherbergen. Bewegt sich nun ein Agent innerhalb des Baus, muss er zu einem bestimmten Zeitpunkt die Kante wechseln. Dabei wird überprüft, ob auf der Kante noch genügend Kapazität für den Agenten frei ist. Ist dies nicht der Fall, dreht die simulierte Ratte um und sucht sich einen neuen Weg zum Ziel ohne diese Kante zu benutzen. Dabei auftretende mögliche Probleme werden wir später betrachten. Für diese zwei unterschiedlichen Wegfindungen, zum einen eine Wegfindung mit angenommenem freien Weg

zum Ziel und zum anderen ein Wegfindung mit bekannten blockierten Kanten durch andere Agenten, wird mit unterschiedlichen Algorithmen der kürzest mögliche Pfad berechnet, um eine optimale Rechenzeit zu erreichen. Für eine Wegfindung ohne bisherige Zusammentreffen mit anderen Agenten, wird der Floyd-Warshall Algorithmus (3.4.1) benutzt. Der A* Algorithmus (3.4.2) wird zur Wegfindung bei bekannten blockierten Kanten benutzt.

3.4.1 *Floyd-Warshall Algorithmus*

Der *Floyd-Warshall Algorithmus* [Flo62, War62] stammt aus der Graphentheorie und berechnet den kürzesten Weg zwischen jeweils zwei Knoten innerhalb eines gewichteten, gerichteten Graphen. Der Bau wird bereits als Graph repräsentiert. Die Gewichtung des Graphen entspricht der Distanz zwischen den entsprechenden Knoten einer Kante. Der Weg zwischen zwei Knoten ist immer in beide Richtungen möglich. Der Algorithmus bildet alle Knotenpaare zwischen zwei Knoten innerhalb des Graphen. Anschließend wird folgende Beobachtung genutzt: Geht der kürzeste Weg von Knoten u nach Knoten v durch Knoten w, dann sind die enthaltenen Teilpfade von u nach w und von w nach v bereits minimal. Nimmt man also an, man kennt bereits die kürzesten Wege zwischen allen Knotenpaaren, die nur über Knoten mit Index kleiner als k führen und man sucht alle kürzesten Wege über Knoten mit Index kleiner oder gleich k, dann hat man für einen Pfad von u nach v zwei Möglichkeiten: Entweder er geht über den Knoten k, dann setzt er sich zusammen aus schon bekannten Pfaden von u nach k und von k nach v, oder es ist schon der bekannte Weg von u nach v über Knoten kleiner als k. Der Algorithmus bestimmt somit zuerst die kürzesten Wege(u,v,1) für alle (u,v) Knotenpaare, und dies wird dann benutzt, um die kürzesten Pfade(u,v,2) zu finden für alle (u,v) Knotenpaare, etc. Dieser Prozess wird fortgeführt bis k = n wobei n die Anzahl aller Knoten ist und wir den kürzesten Pfad für alle (u,v) Knotenpaare gefunden haben, da wir alle möglichen dazwischenliegenden Knoten besucht haben und den Weg mit der kürzesten Distanz uns merken. Alle Pfade sind somit mit v^3 Vergleichen bestimmt. Der *Floyd-Warshall-Algorithmus* wird zur Wegfindung innerhalb des Baus benutzt, unter der Voraussetzung, dass keinerlei blockierte Kanten bekannt sind. Hierbei wird einmalig zu Beginn der Simulation der kürzeste Weg für jeden Knoten zu jedem anderen Knoten bestimmt und in einer Matrix abgelegt. Somit kann dann schnell der Weg von einem Knoten zu einem anderen Knoten innerhalb des Baus bestimmt werden. Außerdem wird eine zusätzliche Tabelle erstellt, die angibt welcher Knoten als nächstes besucht werden muss, wenn man sich auf Knoten a befindet und nach Knoten b gehen möchte. Diese Tabelle wird von Agenten genutzt, um eine Wegfindung von der Startposition zum Ziel zu bestimmen. Dazu muss der Agent auf einer beliebigen Kante lediglich feststellen, welcher Knoten am nächsten zum Ziel liegt und welcher Endknoten hier aufgesucht werden muss. Anschließend muss der Agent den Startknoten aufsuchen und dann bei jedem weiter erreichten Knoten lediglich in der Tabelle nachschauen, welcher Knoten als nächstes besucht werden soll. Die

durch diesen Algorithmus erstellten Tabellen können aber lediglich für eine Wegfindung ohne Behinderung genutzt werden. Begegnet ein Agent, auf seinem Weg zum Ziel, einem anderen Agenten und kann seinen Weg nicht fortführen, können diese erstellten Tabellen nicht genutzt werden. Daher wird hier zur Wegfindung der A* Algorithmus (siehe 3.4.2) eingesetzt.

3.4.2 *A* Algorithmus*

Der A*Algorithmus ([HPE68]) verwendet eine Schätzfunktion, um zielgerichtet einen möglichen Weg zwischen zwei Punkten zu suchen und damit die Laufzeit zu verringern. Der Algorithmus ist optimal. Das heißt, es wird immer die optimale Lösung gefunden, falls eine existiert. Der A*-Algorithmus untersucht immer die Knoten zuerst, die wahrscheinlich schnell zum Ziel führen. Um den vielversprechendsten Knoten zu ermitteln, wird allen bekannten Knoten x jeweils ein Wert f(x) zugeordnet, der angibt, wie lang der Pfad vom Start zum Ziel unter Verwendung des betrachteten Knotens im günstigsten Fall ist. Der Knoten mit dem niedrigsten f-Wert wird als nächstes untersucht. Für einen Knoten x bezeichnet g(x) die bisherigen Kosten vom Startknoten aus, um x zu erreichen. h(x) bezeichnet die geschätzten Kosten von x bis zum Zielknoten. Die verwendete Heuristik darf die Kosten nie überschätzen. Daher wird als Heuristik der Koordinatenabstand, der zwei Punkte, genommen.

$$f(x) = g(x) + h(x) \tag{3.3}$$

Dieser Algorithmus wird verwendet, wenn ein Agent auf einen anderen Agenten trifft und der ursprüngliche Weg blockiert ist. Mit diesem Algorithmus werden dann neue Wege gesucht, die die blockierten Kanten nicht verwenden um ans Ziel zu kommen. Hierbei merkt sich die ausweichende Ratte immer nur die gerade direkt um sie herum blockierten Kanten. Blockierte Kanten aus anderen Situationen werden nicht berücksichtigt. Dies ist insofern sinnvoll, da sich alle Lebewesen innerhalb des Baus bewegen und somit Ausweichbewegungen innerhalb des Baus immer nur auf die momentane Situation reagieren sollten. Da die Berechnung dieser Wege aber mitunter sehr rechenintensiv ist je mehr Tiere sich im Bau aufhalten, wurde ein PathCache eingeführt. Hier werden die häufigsten Wege, die mit diesem Algorithmus berechnet werden, zwischengespeichert, so dass besonders häufige Wege der Agenten schnell abgerufen werden können. Sucht ein Agent also einen Weg von Ort A nach Ort B innerhalb des Baus und ist hierbei von einem anderen Agenten auf seinem Weg blockiert worden, wird zuerst im PathCache überprüft, ob sich dieser Weg bereits im PathCache befindet. Erst wenn dies nicht der Fall ist, wird mit Hilfe dieses Algorithmus der Weg berechnet und gegebenenfalls dem PathCache hinzugefügt.

3.4.3 Bewegungsstrategie

Die Lebewesen bewegen sich innerhalb des Baus wie bereits erwähnt mit Hilfe mehrerer Klassen und Methoden. Das genaue Zusammenspiel dieser Methoden ist mit Hilfe der FlowCharts im Anhang 6.2 einzusehen. Die beteiligten Klassen sind Walk und RandomWalk, sowie die Methoden MakeAStep und MakeAStepWithA-Star der Klasse Creature. Die Schwierigkeit lag darin, eine realistische Bewegungsstrategie zu entwickeln, die andererseits aber starren Regeln folgt. Dabei sollten die Agenten möglichst naiv sein und so wenig Wissen wie möglich über andere Agenten im Bau einsetzen. Außerdem sollte dieses Wissen zuvor angeeignet werden, d.h. die Position anderer simulierter Ratten sollte nur mit in die Wegplanung integriert werden, wenn die Agenten diesen gerade im Moment begegnet waren. Andererseits sollte es ermöglicht werden, eine sinnvolle Anzahl an Agenten im Bau zu beherbergen. Die größte Problematik war hierbei das Auflösen größerer Ansammlungen an Agenten. Dies gelang mit der hier implementierten Bewegungsstrategie am erfolgreichsten.

Wie bereits zuvor erwähnt, dient die Aktion Walk zur Fortbewegung einer Ratte im Bau von einer Startposition zu einem Ziel. Hierbei ist das Ziel zunächst ein fester Ort. Da aber blockierte Kanten zu Ausweichbewegungen führen können, besitzt die Ratte zusätzlich einen gespeicherten Wert, der die Intention jeder Bewegung wiederspiegelt. Hierbei sind unterschiedliche Intentionen möglich. Die Ratte kann beispielsweise eine Kammer zum Schlafen, eine zum Futter ablegen oder eine die Futter enthält suchen. Außerdem können Ausgänge gesucht werden, um den Bau zu verlassen oder zufällige Ausweichbewegungen ausgeführt werden (6.2.2). Auch andere Intentionen sind denkbar und können leicht hinzugefügt werden. Wird Walk ausgeführt, wird zuerst überprüft, ob es sich um eine neue Bewegung handelt. Ist dies der Fall wird die aktuelle Position der Ratte und die Zeit notiert. Dies kann zu weiteren Auswertungen benutzt werden. Außerdem muss die Ratte überprüfen, ob ein Weg zum Ziel möglich ist. Hierbei kann es vorkommen, dass die Ratte allerdings keinen Weg zum Ziel findet. In diesem Fall wird eine Exception 'Kein Weg gefunden! A' geworfen. Gibt es einen Weg wird überprüft, ob die Ratte bereits einer anderen Ratte auf ihrem Weg begegnet ist und blockierte Kanten bekannt sind. Ist dies nicht der Fall wird die Methode MakeAStep aufgerufen. Andernfalls wird MakeAStepWithAStar aufgerufen. Die Methoden unterscheiden sich hauptsächlich in Bezug auf die Wegfindung (vgl. 3.4). Anschließend wird überprüft, ob die Ratte nun am Ziel angekommen ist. Ist dies nicht der Fall, wird Walk erneut mit oder ohne blockierte Kanten aufgerufen.

Wurde eine Exception 'Kein Weg gefunden! A' geworfen, muss die Ratte nun eine Ausweichstrategie verfolgen. Prinzipiell ist es in dem vorliegenden Bau immer möglich alle Punkte im Bau von jedem anderen Punkt zu erreichen, wenn keine anderen Agenten den Weg blockieren. Dies wird also nur aufgerufen, wenn die Ratte bereits anderen Ratten begegnet ist. Würde die Ratte nun an ihrem Platz bleiben, würde dies auf jeden Fall zu Problemen führen. Daher merkt sich die Ratte ihr al-

tes Ziel, um diese Bewegung später auszuführen. Dies wird programmintern gelöst, indem die gerade nicht mögliche Walk-Aktion an die Aktionsliste vorne angefügt wird und die aktuelle Intention der Aktion ebenfalls in einer zusätzlichen Variable oldIntention zwischengespeichert wird. Anschließend wird eine neue RandomWalk (Erläuterungen hierzu siehe ebenfalls weiter unten) an die Aktionsliste der Ratte hinzugefügt und die Intention der Bewegung auf eine Ausweichbewegung gestellt. Die Ratte versucht nun beim nächsten Aufruf der Aktionsliste an einen zufälligen Ort im Bau zu gehen. Ist dies nicht möglich, wird die Aktion beendet und die Ratte versucht erneut das ursprüngliche Ziel zu erreichen. Somit gibt es eine gewisse Wahrscheinlichkeit, dass die Ratte eine Bewegung ausführt, andere Ratten dadurch den blockierten Bereich passieren können und somit Blockierungen gelöst werden.

Die Methode MakeAStep der Klasse Creature wird aufgerufen, wenn eine Ratte auf ihrem Weg zu einem Ziel bisher noch keinen anderen Agenten begegnet ist und somit keinerlei blockierte Kanten kennt. Die Methode kennt die zu erreichende Zielposition und die noch verbleibende Schrittlänge. Beim Eintritt in die Funktion wird überprüft, ob sich der Agent bereits auf dem Ziel befindet. Ist dies der Fall wird die Funktion verlassen. Ist dies nicht der Fall, wird überprüft ob sich die Ratte auf der Zielkante befindet. Ist dies eingetreten, wird überprüft, ob der Agent das Ziel mit der noch verbleibenden Schrittlänge erreichen kann. Ist dies möglich, wird die Position der Ratte auf das Ziel gesetzt und die Funktion verlassen. Andernfalls geht der Agent die restliche Schrittlänge in Richtung Ziel.

Befindet sich die Ratte nicht auf der Zielkante, muss sie sich davor befinden. Der Agent holt sich den verbleibenden Weg aus der mit Hilfe des Floyd-Warshall Algorithmus (3.4.1) erstellten Tabelle. Nun wird überprüft, ob er sich auf einem Knoten befindet. Ist dies der Fall, wird der nächste zu besuchende Knoten auf dem Weg zum Ziel bestimmt. Nun muss überprüft werden, ob sich die simulierte Ratte auf der korrekten Kante befindet oder ob sie die Kante wechseln muss. Befindet sie sich auf der korrekten Kante, bewegt sie sich entlang der Kante, entweder bis die übrige Schrittlänge gelaufen wurde oder die Kante gewechselt werden muss. Es wird die übrige Schrittlänge aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf.

Befindet sich der Agent nicht auf der korrekten Kante, muss die Kante gewechselt werden. Besitzt die Kante Richtung Ziel ausreichend Platz, wird die Kante gewechselt. Es wird dann wiederum überprüft, ob der Agent sich nun auf der Zielkante befindet. Befindet sich der Agent auf der Zielkante, wird MakeAStep erneut aufgerufen. Befindet er sich nicht auf der Zielkante, bewegt er sich entlang der Kante in Richtung Ziel entweder bis die übrige Schrittlänge gelaufen wurde oder die Kante gewechselt werden muss. Es wird die übrige Schrittlänge aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf. Befand sich der Agent hingegen zu Beginn der Funktion weder auf der Zielkante, noch auf einem Knoten, hat er sich auf einer Kante vor dem Ziel befunden. Hier geht der Agent auf der Kante entlang bis entweder die übrige Schrittlänge gelaufen wurde oder die Kante

gewechselt werden muss. Die übrige Schrittlänge wird aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf.

Beim Aufruf der MakeAStep Funktion kann es allerdings vorkommen, dass eine Kante, die betreten werden soll, nicht mehr ausreichend Platz für einen Agenten hat. Dann wird eine Exception 'Blockierte Kante' geworfen. Dies bedeutet, dass die Funktion verlassen wird und eine Exception wirft. Diese Exception wird nun vom aufrufenden Programm abgefangen. In unserem Fall muss nun also Walk darauf reagieren. Hierbei wird nun eine Ausweichstrategie benötigt. Diese Strategie ist essentiell für die Auflösung von gegenseitigen Blockierungen innerhalb des Baus.

Beim Aufruf der Exception 'Blockierte Kante' wird zuerst getestet, ob die Ratte sich tatsächlich auf einem Knoten befindet. Dies sollte immer der Fall sein. Ist dies nicht der Fall, wird die Walk-Aktion mit einer Fehlermeldung abgebrochen. Dies ist in den vorliegenden Simulationen nicht aufgetreten. Befindet sich die Ratte auf einem Knoten, wird nun getestet, ob alle Kanten um die Ratte herum blockiert sind. Ist dies der Fall, kann sich die Ratte nur noch auf ihrer Kante bewegen. Die gerade nicht mögliche Walk-Aktion wird an die Aktionsliste vorne angefügt und zusätzlich die gerade aktuelle Intention der Aktion ebenfalls in einer extra Variablen oldIntention gespeichert. Anschließend wird festgestellt, ob die Kante auf der sich die Ratte akut befindet einen Ausgang hat. Ist dies der Fall, geht die Ratte nach draußen, indem die Aktion GoToExit an den Beginn der Aktionsliste gefügt wird. Ist dies nicht der Fall verharrt die Ratte zwischen einer und zehn Sekunden an ihrem Platz, indem die Aktion WaitBecauseAllEdgesBlocked an den Beginn der Aktionsliste geschrieben wird (3.3.6). Die ursprüngliche Walk Aktion wird verlassen. Sind nicht alle Kanten um die Ratte herum blockiert, wird ein neuer Weg gesucht, der die entsprechende Intention ebenfalls erfüllt. Hierbei wird ein neuer Weg zum ursprünglichen Ziel gesucht. Wäre dieser aber länger als ein anderer Weg zu einem Ort im Bau, der die Intention der Ratte ebenfalls erfüllt, wird der kürzere Weg zu einem neuen Ziel genommen. Ausnahmen sind Bewegungen mit RandomWalk. Hat die Ratte gerade eine Ausweichbewegung gemacht und ist nun an eine blockierte Kante gekommen, wird die alte Intention einer Bewegung geladen und die Ausweichbewegung gestoppt. Es wird die nächste Aktion in der Aktionsliste geladen. Dies ist im Normalfall die Bewegung zum ursprünglichen Ziel. Eine Bewegung zu einem zufälligen Punkt im Bau ohne Ausweichcharakter wird einfach abgebrochen. Ein Beispiel wäre, wenn eine Ratte zu einem Schlafplatz möchte, um sich dort auszuruhen. Auf dem Weg dorthin trifft sie allerdings auf eine andere Ratte, die ihr den Weg versperrt. Nun sucht sie sich einen neuen Weg zum ursprünglichen Ziel ohne diese Kante zu benötigen. Außerdem wird getestet, ob ein anderer Schlafplatz unter diesen Bedingungen nicht näher erscheint. Ist dies der Fall, geht sie zu dieser Kammer. Doch kann es möglich sein, dass weder das ursprüngliche Ziel und auch kein Ersatzziel erreichbar sind. Die Ratte kann sich zum Beispiel in einer Sackgasse befinden die kein Futter enthält, sie hat allerdings einen erhöhten Futterwert und möchte Futter finden. Der Weg aus der Sackgasse heraus wird allerdings von einer anderen Ratte blockiert. In diesem Fall wird die Exception 'Keinen Weg ge-

funden! B' aufgerufen. Hier wird nun an den Beginn der Aktionsliste die gerade nicht mögliche Walk-Aktion geschrieben. Außerdem wird wiederum die Intention der Aktion in oldIntention gespeichert. Nun wird getestet, ob die Ratte sich auf einer Kante befindet, die einen Ausgang hat. Ist dies der Fall, verlässt die Ratte den Bau. Ist dies nicht der Fall, macht sie anschließend eine Ausweichbewegung mittels RandomWalk an eine zufällige Stelle im Bau.

MakeAStepWithAStar funktioniert ähnlich wie MakeAStep. Die Wegfindung ist hierbei aber unterschiedlich, da hier ein Weg gesucht werden muss, der die blockierten Kanten nicht benutzt. Die Methode MakeAStepWithAStar der Klasse Creature wird also aufgerufen, wenn eine Ratte auf ihrem Weg zu einem Ziel bisher mindestens einem anderen Agenten begegnet ist und somit mindestens eine blockierte Kante kennt. Die Methode wird solange ausgeführt, bis die gesamte Schrittlänge aufgebraucht ist oder das Ziel erreicht wurde. Ist eine dieser Bedingungen erreicht, wird die Aktion beendet. Bei jedem Durchlauf wird zuerst überprüft, ob sich der Agent bereits auf der Zielkante befindet. Ist dies der Fall, wird überprüft, ob der Agent das Ziel mit der noch verbleibenden Schrittlänge erreichen kann. Ist dies möglich, wird die Position der Ratte auf das Ziel gesetzt und die Funktion verlassen. Ist dies nicht möglich, geht der Agent die restliche Schrittlänge in Richtung Ziel. Befindet sich die Ratte nicht auf der Zielkante, muss sie sich davor befinden. Der Agent holt sich den verbleibenden Weg ohne die blockierten Kanten zu benutzten. Dieser Weg wird mit Hilfe des A* Algorithmus berechnet (3.4.2). Ist kein Weg möglich wird eine Exception 'Kein Weg gefunden A!' aufgerufen. Ansonsten wird überprüft, ob sich die Ratte auf einem Knoten befindet. Ist dies der Fall, wird der nächste zu besuchende Knoten auf dem Weg zum Ziel bestimmt. Nun muss überprüft werden, ob sich der Agent auf der korrekten Kante befindet oder ob er die Kante wechseln muss. Befindet er sich auf der korrekten Kante bewegt er sich entlang der Kante entweder bis die übrige Schrittlänge gelaufen wurde oder die Kante gewechselt werden muss. Es wird die übrige Schrittlänge aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf. Befindet sich der Agent nicht auf der korrekten Kante, muss die Kante gewechselt werden. Besitzt die Kante zum nächsten Ziel ausreichend Platz, wird die Kante gewechselt. Es wird dann wiederum überprüft, ob die simulierte Ratte sich nun auf der Zielkante befindet. Befindet sich die Ratte auf der Zielkante wird MakeAStepWithAStar erneut aufgerufen. Befindet sie sich nicht auf der Zielkante, bewegt sie sich entlang der Kante in Richtung Ziel entweder, bis die übrige Schrittlänge gelaufen wurde oder die Kante gewechselt werden muss. Es wird die übrige Schrittlänge aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf. Befand sich der Agent hingegen zu Beginn der Funktion weder auf der Zielkante, noch auf einem Knoten, hat er sich auf einer Kante vor dem Ziel befunden. Dann geht der Agent auf der Kante entlang bis entweder die übrige Schrittlänge gelaufen wurde oder die Kante gewechselt werden muss. Es wird die übrige Schrittlänge aktualisiert und die Funktion ruft sich selbst mit den aktualisierten Werten erneut auf. Beim Aufruf der MakeAStepWithAStar Funktion kann es allerdings ebenfalls vorkommen, dass eine

Kante, die betreten werden soll nicht mehr ausreichend Platz für einen Agenten hat. Dies führt dazu, dass eine Exception 'Blockierte Kante' geworfen wird. Dies bedeutet, dass die Funktion verlassen wird und eine Exception wirft. Diese Exception muss nun vom aufrufenden Programm abgefangen werden.

RandomWalk ähnelt Walk, nur dass es sich hierbei um eine Ausweichbewegung handelt. Der executeAction Teil gleicht hier dem Programmteil von Walk mit ein paar Änderungen. Wird RandomWalk ausgeführt, wird zuerst überprüft, ob es sich um eine neue Bewegung handelt. Ist dies der Fall wird die jetzige Position der Ratte und die Zeit notiert. Dies kann zu weiteren Auswertungen benutzt werden. Außerdem muss überprüft werden, ob ein Weg zum Ziel möglich ist. Hierbei kann es vorkommen, dass die Ratte allerdings keinen Weg zum Ziel findet. In diesem Fall wird eine Exception 'Kein Weg gefunden! A' geworfen. Gibt es einen Weg wird überprüft, ob die Ratte bereits einer anderen Ratte auf ihrem Weg begegnet ist und blockierte Kanten bekannt sind. Ist dies nicht der Fall wird die Methode MakeAStep aufgerufen. Sind blockierte Kanten bekannt, wird MakeAStepWithAStar aufgerufen. Anschließend wird überprüft, ob die Ratte nun am Ziel angekommen ist. Ist dies nicht der Fall, wird RandomWalk erneut mit oder ohne blockierte Kanten aufgerufen.

Wurde eine Exception 'Kein Weg gefunden! A' geworfen, konnte die Ratte die Ausweichbewegung nicht vollbringen, da auch dieser zufällige Punkt nicht erreicht werden konnte. Es wird eine neue zufällige Position im Bau gewählt und eine neue RandomWalk-Aktion an die entsprechende zufällige Position an den Beginn der Aktionsliste geschrieben.

Auch kann der Aufruf von MakeAStep und MakeAStepWithAStar zu einer Exception führen, die dann von RandomWalk abgefangen werden muss. Tritt die Exception 'Kein Weg gefunden B' auf, so wird eine neue RandomWalk zu einer neuen zufälligen Stelle im Bau an den Beginn der Aktionsliste geschrieben und die Funktion verlassen.

Beim Aufruf der Exception 'Blockierte Kante' wird zuerst wiederum getestet, ob die Ratte sich tatsächlich auf einem Knoten befindet. Dies sollte immer der Fall sein. Ist dies nicht der Fall, wird die RandomWalk-Aktion mit einer Fehlermeldung abgebrochen. Dies ist in der vorliegenden Simulation nicht aufgetreten. Befindet sich die Ratte auf einem Knoten wird nun getestet, ob alle Kanten um den Agenten herum, blockiert sind. Anschließend wird festgestellt, ob die Kante auf der sich die Ratte aktuell befindet einen Ausgang hat. Ist dies der Fall, geht die Ratte nach draußen, indem die Aktion GoToExit an den Beginn der Aktionsliste gefügt wird. Ansonsten wird die Funktion beendet. Im Normalfall wird dann als nächstes eine Bewegung zum ursprünglichen Ziel versucht, die die RandomWalk Aktion ursprünglich ausgelöst hatte. Sind nicht alle Kanten um die Ratte herum besetzt, sollte die Intention der Ratte auf RandomWalk stehen. Ist dies nicht der Fall wird die Aktion mit einer Fehlermeldung abgebrochen. Dies war in den vorliegenden Simulationen nie der Fall. Ansonsten hat die Ratte gerade versucht eine Ausweichbewegung zu

machen und ist nun an eine blockierte Kante gekommen, daher wird die alte Intention einer Bewegung geladen und die Ausweichbewegung gestoppt. Es wird die nächste Aktion in der Aktionsliste geladen. Dies ist im Normalfall die Bewegung zum ursprünglichen Ziel.

3.4.4 Auswirkungen der Bewegungsstrategie

Mit der hier vorgestellten Simulation können nun die Auswirkungen einer solchen Bewegungsstrategie der implementierten Agenten getestet werden. Durch den Aufbau von Bedürfnissen, die jedes Lebewesen innerhalb der Simulation stillen muss, wird hierbei eine realistischere Simulation geschaffen und die Bewegungsstrategie muss sich als sinnvolle Strategie zur Bewegung innerhalb des Baus bewähren, so dass die simulierten Tiere ihre Bedürfnisse ausreichend stillen können. Eine nicht sinnvolle Bewegungsstrategie führt zum Tod aller Agenten. Nur sinnvolle Bewegungsstrategien führen zum Überleben der Agenten. Hierbei können außerdem mehrere weitere Fragestellungen geklärt werden. Zum einen wurde getestet wie viele Agenten den Bau unter der gegebenen Bewegungsstrategie bewohnen können. Die hier gewonnenen Daten können mit realistischen Populationengrößen innerhalb des Baus verglichen werden. Außerdem kann die Baunutzung untersucht werden, d.h. welche Bereiche im Bau besonders häufig genutzt werden und welche selten aufgesucht werden. Zudem können die Ausgänge genauer untersucht werden, also wie oft Ratten den Bau an bestimmten Stellen verlassen bzw. betreten. Des Weiteren kann getestet werden, ob sich eine konstante Populationsgröße einstellen kann, wenn man davon ausgeht, dass sich die Tiere nach dem aufgestellten Vermehrungsmodell fortpflanzen. Da ein Vermehrungsmodell auch Einfluss auf die benötigte Futtermenge hat, da ja weibliche trächtige Ratten mehr Nahrung benötigen, kann hier ebenfalls festgestellt werden, wie hoch der Einfluss hierbei ist, d.h. inwieweit hier die benötigte Futtermenge insgesamt ansteigt bzw. wie sich dadurch die mittlere Futterfundgröße unterscheiden muss.

3.5 Futtereintragestrategien

Nachdem eine sinnvolle Bewegungsstrategie implementiert wurde, sind weitere Untersuchungen möglich. Innerhalb dieses Simulationsframeworks sind nun viele unterschiedliche Verhaltensweisen untersuchbar. Nach der Erstellung des Frameworks sollte nun in einem ersten Schritt eine einfache Verhaltensweise weiter untersucht werden und hierbei Auswirkungen auf den Bau und die gesamte Population festgestellt werden. Hierzu wurden unterschiedliche Futtereintragestrategien der simulierten Ratten implementiert, um festzustellen, wie sich diese auf die Baunutzung und Aufenthaltsdauer außerhalb des Baus auswirken. Des Weiteren kann getestet werden, wie sich diese unterschiedlichen Strategien auf die mögliche Populationsgröße innerhalb des Baus auswirken. Zusätzlich kann hier die Baunutzung aufgezeichnet

werden, um festzustellen, ob sich durch unterschiedliche Strategien Auswirkungen feststellen lassen.

Es wurden zwei unterschiedliche Futtereintragestrategien getestet:

3.5.1 Futtereintragestrategie 1

Bei der ersten Futtereintragestrategie wird angenommen, dass die Tiere sich möglichst wenig außerhalb des Baus aufhalten und immer möglichst schnell wieder in den Bau gehen, egal ob sie Futter gefunden haben oder nicht. Dies würde einer sinnvollen Strategie entsprechen, um Fressfeinden zu entgehen. Die Tiere sollten sich ausschließlich zur Futtersuche außerhalb des Baus begeben. Andere Aktionen werden nur innerhalb des Baus ausgeführt, d.h. die Tiere fressen auch nur innerhalb des Baus. Dies bedeutet, dass sich die simulierten Ratten möglichst kurz außerhalb des Baus aufhalten. Andere Aspekte wie Hunger, Müdigkeit oder ein außerhalb eingetretenes Ereignis wie etwa ein Futterfund führen hier zu keiner Strategieänderung.

3.5.2 Futtereintragestrategie 2

Bei der zweiten Futtereintragestrategie wird angenommen, dass die Tiere sich möglichst wenig innerhalb des Baus aufhalten. Dies bedeutet, die Ratten nutzen den Bau hauptsächlich als Schlafplatz und Vorratskammer, außerdem zur Aufzucht von Jungratten. Hierbei bleibt eine Ratte, sobald sie den Bau verlassen hat, möglichst lange außerhalb des Baus. Hier frisst sie das gefundene Futter, falls sie welches findet und betritt den Bau nur, wenn sie Futter im Bau ablegen möchte oder zu müde ist, d.h. einen Müdigkeitswert über 396 hat. Ansonsten bleibt die Ratte außerhalb des Baus. Diese Strategie berücksichtigt somit also die aktuellen Bedürfnisse einer Ratte, doch wird gleichzeitig die Aufenthaltsdauer außerhalb des Baus erhöht. Dies führt somit zu einer effizienteren Zeitnutzung, da die Ratte weniger Wegstrecke zurücklegen muss. Außerdem steigt in diesem Modell die Futterfundwahrscheinlichkeit mit der Aufenthaltsdauer außerhalb vom Bau. Dies entspricht in einer gewissen Näherung einer realistischen Umwelt, wenn man hier davon ausgeht, dass Futterstücke an bestimmten Stellen in größerer Anzahl vorliegen. Dies könnte in der Realität etwa der Fall sein, wenn ein Tier etwa ein Aasstück findet oder ein Strauch mit Früchten. Allerdings ist es klar, dass dieses Modell hier nur sehr vereinfacht die Welt außerhalb darstellt.

Diese beiden Futtereintragestrategien wurden unter verschiedenen Aspekten betrachtet. Es sollte hierbei festgestellt werden welche Regelkreise sich unter den Futtereintragestrategien bei einer bestimmten Futtergrößenverteilung einspielen und welche Rolle hierbei die Größe und die Struktur des Baus wie auch die Bewegungsstrategie der Agenten und deren Aktionswahl und Anzahl spielen. Außerdem

wurde getestet, unter welchen Bedingungen sich eine stetige Population einstellt, wenn man davon ausgeht, dass sich die Tiere nach dem aufgestellten Vermehrungsmodell fortpflanzen und welche Menge an Futter die Tiere hierzu benötigen. Um die vorgestellten Futtereintragestrategien unter einem weiteren Aspekt zu betrachten wurde in einem letzten Schritt ein Prädator eingeführt. Die Wahrscheinlichkeit einem Prädator zu begegnen hing hierbei von der Aufenthaltsdauer der simulierten Ratte außerhalb des Baus ab. Mit längerer Aufenthaltsdauer stieg die Wahrscheinlichkeit einem Prädator zu begegnen. Hierbei wurde getestet, in wie weit die beiden Futtereintragestrategien hier unterschiedliche Ergebnisse erzielten in Hinblick auf das Überleben einer konstanten Population. Für den Vergleich der beiden Strategien musste eine Reihe von Daten innerhalb der Simulation möglichst effizient und einfach aufgenommen und weiterverarbeitet werden. Wie dies möglich ist, wird im nächsten Abschnitt erklärt.

3.6 Ermittlung der Daten

Während der Simulation sollen die Agenten und der Bau beobachtbar sein. Außerdem sollen mehrere Daten aktuell dargestellt werden. Außerdem kann es interessant sein den zeitlichen Verlauf einiger Daten während der Simulation, etwa wie viel Futter sich zu verschiedenen Zeitpunkten innerhalb des Baus befand, zu betrachten. Es wäre denkbar, dies mit Hilfe von Logmeldungen zu bewerkstelligen, doch führt dies schnell zu Verwirrungen und extrem vielen Daten. Diese müssen dann in einem weiteren Schritt verarbeitet werden. Daher ist die aktuelle Verfolgung der Simulation über diesen Weg, nur sehr begrenzt möglich. Es ist zwar möglich dies zu tun und es können auch alle Strukturen des Baus, sowie die Agenten und deren jeweilige Aktionen geloggt werden, doch ist eine Auswertung allein aufgrund dieser Daten sehr mühsam und zeitaufwendig. Trotzdem kann das Logsystem sehr gut bei der Weiterentwicklung der Simulation genutzt werden, da hier prinzipiell alles geloggt werden kann. Um allerdings eine einfacher und benutzerfreundlichere Verfolgung der Simulation zu ermöglichen, wurde entschieden dies besser mit Hilfe mehrerer Fenster, Views genannt, zu bewerkstelligen. Jedes dieser Fenster muss über die aktuellen Geschehnisse innerhalb der Simulation, die das jeweilige Fenster für seine Datenanzeige benötigt, unterrichtet werden. Da jedes Fenster oftmals nur einen bestimmten Teil der Simulation benötigt, wurde das Listener-Konzept genutzt.

3.6.1 Listener-Konzept

Bei der objektorientierten Programmierung sollt man darauf achten, dass Klassen nicht fest miteinander verzahnt, sondern lediglich lose gekoppelt sind. Dies beinhaltet ebenfalls das Prinzip der Datenkapselung. Der direkte Zugriff auf interne Datenstrukturen anderer Klassen sollte unterbunden werden. Interaktionen sollten besser über wohldefinierte Schnittstellen erfolgen, so dass die Klassen später noch

verändert werden können. Die lose Kopplung hat viele Vorteile, unter anderem wird die Wiederverwendung des Codes erhöht und das Programm ist änderungsfreundlicher. Das Prinzip eines Listeners geht auf das Konzept eines Observer-Pattern zurück. Das Observer-Pattern hat seine Ursprünge in Smalltalk-80. Hierbei werden zwei Komponenten unterscheiden: zum einen der *Observables (Beobachtbare)* und die *Observer (Beobachter)*. Eine anschauliche Erklärung des Prinzips ist in [Ull09] zu finden: Die zwei Komponenten werden anhand einer Partygesellschaft erklärt. Auf dieser Party befinden sich zurückaltende passive Gäste und aktive Erzähler. Die Zuhörer sind interessiert an den Gesprächen der Unterhalter. Da die Erzähler nun von den Zuhörern beobachtet werden, bekommen sie den Namen *Beobachtete* oder *Beobachtbare* (englisch: *Observables*). Die Erzähler interessieren sich hingegen nicht dafür, wer ihnen zuhört. Für sie sind alle Zuhörer gleich. Sie schweigen allerdings, wenn ihnen gar niemand zuhört. Die Zuhörer reagieren auf die Geschichten der Erzähler, und werden dadurch zu `Beobachtern`. Um bei diesem Beispiel zu bleiben, erzählt der Erzähler einen Witz, lachen sie. Das Beispiel wird in Java anhand der Klassenbibliothek *Observable* repräsentiert. Der Beobachter wird durch die Schnittstelle *Observer* abgedeckt, und ist der, der informiert werden will, wenn sich eine Datenstruktur verändert. Jedes Exemplar der *Observable*-Klasse informiert alle seine Horcher, wenn sich sein Zustand ändert. Eine Erweiterung der Möglichkeit über *Observer/Observable* zu kommunizieren sind *Listener*. Es gibt bestimmte Ereignisauslöser, die spezielle Ereignis-Objekte aussenden. Außerdem Interessenten, die sich bei den Auslösern an- und abmelden können. Sie werden in Java *Listener* genannt. Die Interessenten implementieren als `Listener` eine Java-Schnittstelle, die `XXXListener` heißt - `XXX` steht hierbei für einen frei wählbaren Namen. Die Operation der Schnittstelle kann beliebig lauten und kann ebenfalls mehrere Operationen vorschreiben. Der Ereignisauslöser bietet die Methoden `addXXXListener(XXXListener)` und `removeXXXListener(XXXListener)` an, um Interessenten an- und abzumelden. Immer wenn ein bestimmtes Ereignis eintritt, erzeugt der Auslöser das Ereignisobjekt und informiert jeden Listener, der in der Liste eingetragen ist, über einen Aufruf der Methode aus dem Listener über das entsprechende Ereignis.

Mit Hilfe solcher Listener können die entsprechenden Views benachrichtigt werden. Hierbei wurden in der Simulation mehrere Listener-Interfaces implementiert. Jeder Listener wird hierbei bei mehreren Methoden der jeweiligen Klasse verständigt und kann dann entsprechend darauf reagieren. Das Interface BurrowListener dient zur Weitergabe bei Veränderungen des Baus z.B. wenn eine Kante hinzugefügt wird oder ein Agent eine Kante betritt. Das Interface CreatureListener dient zur Weitergabe von Informationen, wenn ein Agent etwas macht, wie etwa fressen oder schlafen oder sich innerhalb des Baus bewegt. Das Interface RatListener erbt von Creature und dient zur Weitergabe rattenspezifischer Daten, wie etwa der Aktionen, die eine Ratte ausführt oder der außerhalb des Baus verbrachten Zeit. Ein FoodListener zeigt an, wenn ein Futterstück gefunden wird, gefressen wird oder ganz verspeist wurde. Zusätzlich wurde ein TestListener eingeführt. Er

wird aufgerufen, wenn ein Agent eine Bewegung an sein Ziel beendet hat. Dies kann zu Geschwindigkeitsberechnungen und ähnlichem benutzt werden. Weitere Listener können einfach und schnell hinzugefügt werden, indem sie in die Liste der jeweiligen Klasse hinzugefügt werden.

Das Listener Konzept bewährt sich somit als einfache und klare Lösung zur Kommunikation zwischen den Anzeigefenstern und den entsprechenden Klassen der Simulation. Die Datenkapselung führt zu einer einfachen Ausbaufähigkeit des Systems, so dass neue Fenster in das vorhandene System schnell und einfach implementiert werden können. Ebenfalls ist eine Feststellung neuer Daten einfach und schnell erreichbar, indem ein entsprechendes Ereignisobjekt erstellt wird.

3.6.2 Views

Wie bereits erwähnt, kann die Simulation mit Hilfe mehrere Fenster beobachtet werden. Die Funktionen, über die jedes Fenster verfügen soll, werden in der Klasse RatpopDisplay zusammengefasst. Jeder View erbt von dieser Klasse. Beim Start der Simulation entsteht ein Kontrollfenster. Mit diesem Fenster kann die Simulation gestoppt werden. Die einzelnen Fenster können auf Wunsch über eine dropdown Liste innerhalb des Kontrollfensters aufgerufen werden. Außerdem kann jedes Fenster als png-Datei von Hand jederzeit abgespeichert werden. Zusätzlich werden alle Fenster jeden Monat gespeichert. Aber auch ein anderer Turnus ist einfach zu erreichen. Außerdem wird jedes Fenster regelmäßig aktualisiert. Der jeweilige Zeitpunkt einer Aktualisierung ist hierbei von Fenster zu Fenster unterschiedlich. Ausgenommen hiervon ist die monatliche Aktualisierung des Fensters, so dass hier immer ein fester Zeitpunkt aufgenommen wird. Im weiteren Verlauf soll lediglich auf die wichtigsten Fenster eingegangen werden. Viele dieser Fenster zeigen Diagramme. Diese werden mit Hilfe des Frameworks JFreeChart erstellt. JFreeChart ist ein Framework, mit dessen Hilfe komplexe Diagramme, wie Säulen- oder Balken- oder Kreisdiagramme und Histogramme leicht erstellt werden können. Außerdem ist ein einfacher Export der Diagramme als png Grafik möglich.

View3D

Mit Hilfe des Fensters View3D wird der Bau in dreidimensionaler Weise dargestellt mit seinen Schlafplätzen, Ausgängen und dem Tunnelsystem. Alle im Bau befindlichen Ratten und Futterstücke werden ebenfalls dargestellt. In diesem Fensters können alle Agenten und Futterstücke direkt beobachtet werden. Bei Simulationen über einen längeren Zeitraum sollte der View besser deaktiviert werden, da er relativ rechenintensiv ist und die Beobachtung der Rattenbewegungen bei maximaler Simulationsgeschwindigkeit schwerfällt. Hier empfiehlt es sich dann den ViewOneRat zu verwenden, der die zurückgelegten Wege einer Ratte anzeigt. Falls nur eine kurze Zeitspanne simuliert werden soll, kann auch mit einer eingebauten Pause zwi-

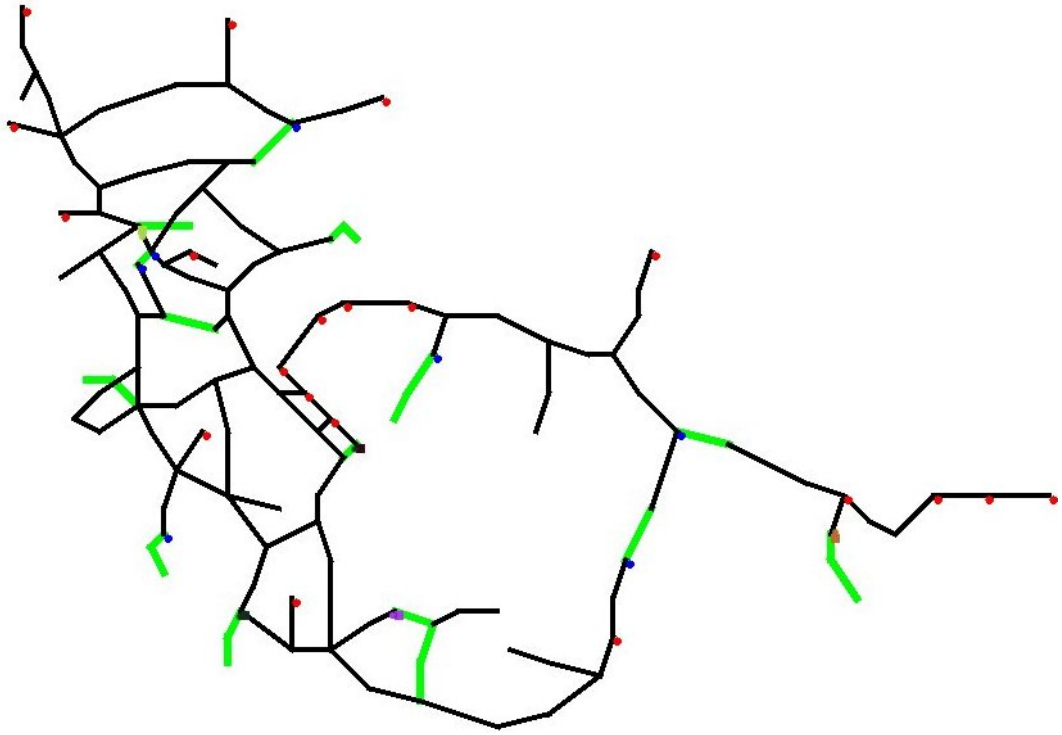

Abbildung 3.6: Beispiel einer Bauansicht innerhalb der Simulation mit den darin befindlichen Ratten und Futterstücken.

schen den einzelnen Rechenschritten die Simulation verlangsamt werden und somit auch die einzelnen Bewegungen eines Agenten besser nachvollzogen werden. Dies kann über die Variable timeBetweenTicks der Klasse Test eingestellt werden. Hier kann die Zeit in Millisekunden angegeben werden, die zwischen zwei Ticks in der Realität liegen soll. Die Darstellung des Baus und der darin befindlichen Agenten (Abbildung 3.6) erfolgte mit Hilfe der Klassenbibliothek Java3D (3.1). Das Tunnelsystem wird als Graph mit schwarzen Linien dargestellt. Die Schlafplätze innerhalb des Baus sind hierbei grün unterlegt. Ratten werden aus zwei Komponenten dargestellt: sie bestehen aus einem Zylinder und einem Kreis. Die Zylinderspitze zeigt die Blickrichtung der Ratte an. Ausgänge werden als rote Kugeln dargestellt, abgelegte Futterstücke im Bau als blaue.

ViewBurrowUsage

Mit diesem Fenster kann die Baunutzung betrachtet werden. Eine Farbcodierung zeigt an, welche Bereiche häufig genutzt werden und welche selten von Agenten betreten werden. Rote Bereiche zeigen stark genutzte Wege, wohingegen blaue Strukturen nur sehr selten betretene Bereiche zeigen. Grüne Bereiche werden durch-

schnittlich oft aufgesucht. Jede Kante speichert wie oft sie innerhalb einer Simulation betreten wurde. Betritt also eine Ratten eine Kante wird dieser Wert um eins erhöht. Zusätzlich werden die Benutzungen der Ausgänge festgehalten. Auch hier gilt wiederum, rote Farben geben eine häufige Nutzung an, blaue Ausgänge werden selten benutzt. Da ein Ausgang sowohl als Ausgang, als auch als Eingang dienen kann, wurde dies auch unterschieden. Der innere Kreis gibt die Nutzung als Eingang wieder, der äußere Kreis als Ausgang. Da die Tiere beim Betreten des Baus einen zufälligen Eingang wählen, zeigen die Werte für die Nutzung als Eingang keinen großen Unterschied. Die Anzahl der Ein- und Austritte und die Anzahl der Kantenbesuche wurde wie folgt normalisiert:

$$norm = visits/max \tag{3.4}$$

wobei:

visits = Anzahl, die diese darzustellende Kante bisher betreten wurde

max = maximale Anzahl, die eine Kante bisher betreten wurde

Außerdem wurde der normierte Mittelwert berechnet, wie oft eine Ratte eine Kante betritt:

$$mean = (allVisitsTogether/max)/allEdges \tag{3.5}$$

wobei:

allVistitsTogether = Summe aller Eintritte aller Kanten

max = maximale Anzahl, die eine Kante bisher betreten wurde

allEdges = Anzahl, der im Bau befindlichen Kanten

Diese Werte verändern dann die Farbgebung der Kante: Es liegt eine Farbgebung durch drei Farbwerte vor: ein roter Bereich, ein grüner und ein blauer. Die Werte für den grünen Bereich werden konstant auf 1 gesetzt. Die Werte für den roten und blauen Bereich sollen Werte zwischen 0 und 1 annehmen. Es werden folgende Gleichungen benutzt:

Der Wert für den roten Bereich:

$$r = 0{,}5 + norm - mean \tag{3.6}$$

Falls $r \leq 0$ wird $r = 0$ gesetzt, falls $r \geq 0$ wird $r = 1$ gesetzt.

Der Wert für den blauen Bereich:

$$b = 0{,}5 - norm + mean \tag{3.7}$$

Falls $b \leq 0$ wird $b = 0$ gesetzt, falls $b \geq 0$ wird $b = 1$ gesetzt.

ViewAllActions

In diesem Fenster wird angezeigt, wie viel Prozent ihrer Zeit die Agenten mit den einzelnen Aktionen verbringen. Die Aktionen werden eingeteilt in Eat, Sleep, Born, Die, SearchOutside und MakeAStep. Eat ist die reine Zeit, die Ratten benötigen, um ihr Futter zu Fressen. Sleep ist die Zeit, die eine Ratte mit Schlafen verbringt, Born ist die Aktion, wenn eine Ratte geboren wird, Die wenn sie stirbt. Da dies jeweils eine einmalige Aktion jeder Ratte ist, sind die Werte hierfür immer 0% und wird daher oftmals nicht angezeigt. SearchOutside ist die mit Futtersuche verbrachte Zeit der Ratten außerhalb des Baus. MakeAStep ist die Zeit, die sich Ratten innerhalb des Baus fortbewegen.

ViewOfOneRat

Dieses Fenster dient zum genaueren betrachten einzelner Agenten. Da das View3D-Fenster (3.6.2) nur den Bau mit allen darin befindlichen Ratten anzeigt, aber oftmals sowohl der Weg einer Ratte nur schlecht verfolgt werden kann als auch, was genau ein Agent gerade macht, wurde dieses weitere Fenster eingeführt. Durch eine dropdown Anzeige kann jede Ratte, die sich in der Simulation befindet, ausgewählt werden. Es können für jeden Agenten folgende Werte abgelesen werden: Alter, Geschlecht und eine etwaige Trächtigkeit, ihre Position innerhalb der Simulation, die Farbe der Ratte in der View3D Anzeige und der aktuelle Hunger und Müdigkeitswert der Ratte. Außerdem die Größe ihrer Aktionsliste und die aktuelle Aktion der Ratte. Außerdem kann mit diesem Fenster eine Routenverfolgung der Ratte innerhalb des Baus stattfinden. Hierbei wird die zurückgelegte Strecke der Ratte innerhalb des Baus angezeigt. Der untere Teil des Fensters zeigt den Rattenbau an. Der zurückgelegte Weg der Ratte ist farblich hervorgehoben. Der zeitlich zuletzt zurückgelegte Weg wird hierbei rot dargestellt, der letzte aufgezeichnete Weg wird gelb dargestellt. Wege zwischen diesen Zeitpunkten sind in einem Farbverlauf von gelb nach rot dargestellt, so dass anhand der farblichen Codierung die zurückgelegten Wege der Ratte dargestellt werden. Als Angabe wie weit der Weg dargestellt werden soll, kann die Anzahl verwendeter Positionen der Ratte eingestellt werden. Als Beispiel soll hier eine Abbildung dienen, bei dem die letzten 100 Position innerhalb des Baus einer Ratte herangezogen wurden (Abbildung 3.7).

BeOutsideOrInside

In diesem Fenster, wird angezeigt, wie viel Prozent ihrer Zeit die Tiere innerhalb oder außerhalb des Baus verbringen.

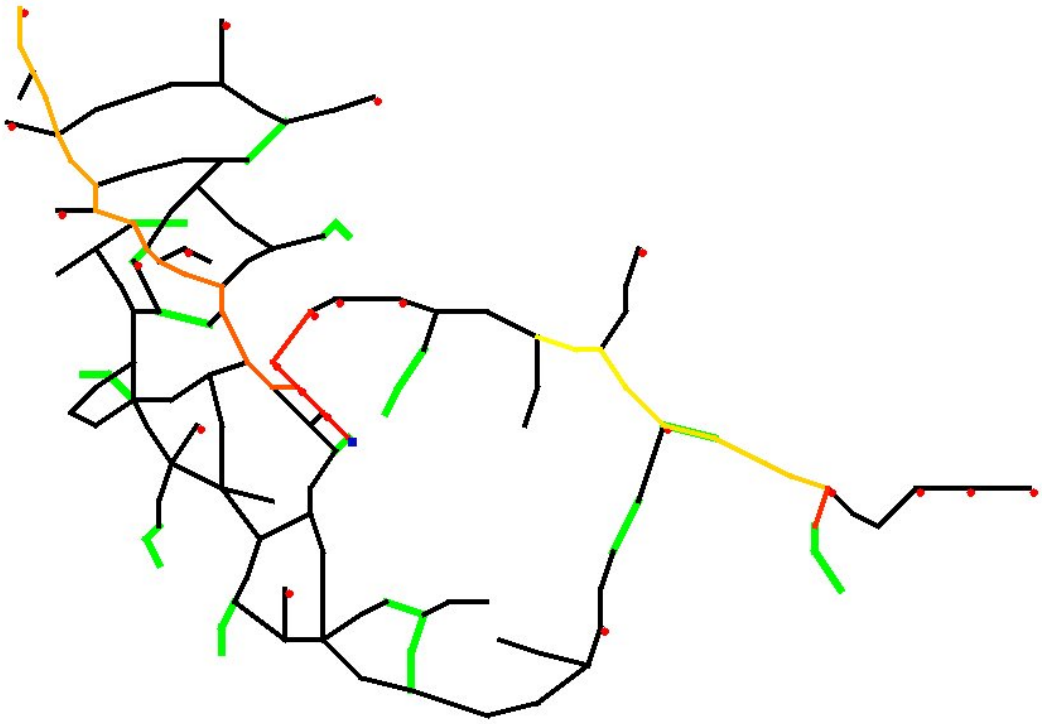

Abbildung 3.7: Beispiel einer Ansicht des Baus mit der zurückgelegten Strecke einer Ratte. Hierbei wurden die letzen 100 Positionen der Ratte innerhalb des Baus herangezogen. Der zeitliche Verlauf der Bewegungen ist farblich codiert: rote Bereiche zeigen zuletzt besuchte Orte an, gelb die zeitlich ältesten Positionen.

ViewTimesOutside

In diesem Fenster wird ein Histogramm angezeigt, dass die Häufigkeit angibt, wie oft Agenten eine bestimmte Zeit außerhalb des Baus verbracht haben. Dies unterscheidet sich je nach Futtereintragestrategie fundamental und kann hier genau betrachtet werden. Die Auflösung des Histogramms wurde auf zwei Minuten gewählt, d.h. jede Säule entspricht einem Zeitraum von 2 Minuten oder 120 Sekunden.

ViewHungerOfAllRats

In diesem Fenster können die Hungerwerte aller Ratten im zeitlichen Verlauf der Simulation betrachtet werden. Die Kurve des Hungerwerts einer Ratte hat die in ViewOneRat angegebene Farbe, so dass die Kurve einer speziellen Ratte zugeordnet werden kann. Stirbt eine Ratte, sinkt ihr Hungerwert auf null und wird anschließend nicht mehr angezeigt.

ViewTirednessOfAllRats

In diesem Fenster kann der zeitliche Verlauf der Müdigkeitswerte aller Ratten angezeigt werden. Auch hier gilt: Die Kurve des Müdigkeitswert einer Ratte hat die in ViewOfOneRat angegebene Farbe, so dass die Kurve einer speziellen Ratte zugeordnet werden kann. Stirbt eine Ratte, sinkt ihr Müdigkeitswert auf null und verschwindet aus der Anzeige.

ViewPopulation

Mit diesem Fenster kann die Entwicklung einer Population nachvollzogen werden, d.h. hier werden alle Geburten und Sterbeprozesse festgehalten und zeitlich aufgetragen. Es kann also leicht verfolgt werden, ob eine Population gerade wächst, konstant bleibt oder sich verringert. Zu jedem Zeitpunkt wird die Anzahl der Ratten innerhalb des Baus aufgetragen.

ViewFood

Dieses Fenster ist zweigeteilt. Die linke Seite zeigt die zeitliche Entwicklung der Futtermenge, die rechte die zeitliche Entwicklung der Anzahl an Futterstücken innerhalb des Baus an. Die Anzahl an Futterelementen steigt nicht über einen bestimmten Wert an. Dies liegt an einer internen Verwaltung von Futterobjekten. Da die Futterobjekte, falls sie nicht gefressen werden eine ewige Lebensdauer innerhalb der Simulation aufweisen, führte dies bei längeren Simulationen zu einer extrem hohen Anzahl an Futterobjekten innerhalb des Baus. Dies wiederum führte zu einem enormen Rechenaufwand und somit einer langsameren Simulation. Um diesen Problemen vorzubeugen, werden nun immer nur 20 Futterobjekte pro Kante zugelassen. Befinden sich nach dem Ablegen des Futterobjektes mehr Futterobjekte auf einer Kante verschmelzen zwei Futterobjekte miteinander zu einem größeren Futterobjekt. Dies ist für das implementierte Verhalten der Agenten allerdings irrelevant, da abgelegte Futterobjekte nur noch auf der entsprechenden Kante gefressen werden und es hierbei für das Rattenobjekt keinen Unterschied macht, ob sie zwei Objekte aufnimmt und frisst oder nur eines. Außerdem ist die Anzahl an möglichen Futterobjekten auf einer Kante immer größer als die Anzahl möglicher Agenten, so dass im Normalfall auch jeder Agent ein Objekt dort finden sollte, vorausgesetzt es wären mehr als 20 Futterobjekte auf diese Kante.

ViewTraveledDistanceWithoutMeetingAnotherRat

Dieses Fenster zeigt die mittlere Distanz über alle Ratten innerhalb der bisherigen Simulation, zu dem angegebenen Zeitpunkt an, die die Ratten im Mittel zurückgelegt haben, bevor sie einer anderen Ratte begegnet sind.

ViewVelocityOfIdealPath

Die Geschwindigkeit, die hier angegeben ist, ist eine theoretische Geschwindigkeit, die angeben soll, wie erfolgreich eine Ratte ihr festgelegtes Ziel erreicht. Hierbei fließen nur Wege der Ratten ein, die auch abgeschlossen wurden. Tatsächlich bewegen sich alle Agenten mit derselben Geschwindigkeit innerhalb der Simulation. Die hier dargestellte Geschwindigkeit ist die theoretische Geschwindigkeit, die eine Ratte aufweisen würde, wenn sie den kürzesten Weg gelaufen wäre. Da aber der direkte Weg durch andere Agenten blockiert sein kann, müssen sich die Agenten oftmals einen anderen Weg innerhalb der Simulation zum Ziel suchen. Eventuell werden auch noch Ausweichbewegungen gestartet. Somit benötigen die Ratten mehr Zeit um zum Ziel zu kommen. Hierbei wird dann die Zeit gemessen, die die Ratte benötigt um ans Ziel zu kommen. Die Geschwindigkeit wird dann wie folgt gebildet:

$$x_i = shortestPath/time \tag{3.8}$$

wobei:

x_i = theoretische Geschwindigkeit eines ideal zurückgelegten Weges

shortestPath = kürzest mögliche Distanz zwischen Startpunkt und Zielpunkt

time = benötigte Zeit um das Ziel tatsächlich zu erreichen

Die hier angegebene theoretische Geschwindigkeit ist somit ein Mittelwert aller bisher zurückgelegten Wege aller Ratten. Die Geschwindigkeit wird hierbei in m/sec angegeben.

$$v = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.9}$$

wobei:

v = Mittlere Geschwindigkeit aller Ratten zum Zeitpunkt t

n = Anzahl aller zurückgelegten Wege

x_i = theoretische Geschwindigkeit eines ideal zurückgelegten Weges

ViewRelationIdealPathTraveledPath

Es wird bei jedem Agenten die zurückgelegte Distanz festgehalten, die er vom Startpunkt der zielgerichteten Bewegung bis zum Erreichen des Ziels zurückgelegt hat. Dann wird der Quotient r gebildet aus der zurückgelegten Distanz, die der Agent tatsächlich zurückgelegt hat, um an sein Ziel zu gelangen und der kürzest möglichen Distanz, die der Agent innerhalb des Baus zurücklegen musste, um vom Startpunkt an das Ziel zu gelangen.

$$r_i = traveledWay/shortestPossibleWay \tag{3.10}$$

wobei:

traveledWay = tatsächlich zurückgelegte Distanz, die ein Agent zurückgelegt hat, um vom Startpunkt der Bewegung zu seinem Ziel zu gelangen

shortestPossibleWay = kürzest möglichen Distanz, die ein Agent innerhalb des Baus zurücklegen muss, um vom Startpunkt an das Ziel zu gelangen

Werden nun alle diese Quotienten über alle Agenten addiert und über die Anzahl der zurückgelegten Wegstrecken dividiert, erhält man einen Einblick, in wie weit die gegenseitigen Behinderungen einen Einfluss auf die zurückgelegten Strecken haben und somit eventuell auch einen Einfluss auf weitere Behinderungen haben könnten.

$$m = \frac{1}{n} \sum_{i=1}^{n} r_i \tag{3.11}$$

wobei:

m = Mittlerer Quotient über alle Wege der Agenten innerhalb der Simulation

n = Anzahl aller zurückgelegten Wege

r_i = theoretische Geschwindigkeit eines ideal zurückgelegten Weges

Kapitel 4

Ergebnisse

In diesem Kapitel wird eine Auswahl an Simulationen präsentiert, die mit Hilfe der in Abschnitt 3.5 vorgestellten Futtereintragestrategien erstellt wurden. Für diese Simulation wurde eine Aufenthaltsdauer außerhalb des Baus von N(300/60) eingestellt (3.3.6), bis der Agent den Bau wieder betrat (Futtereintragestrategie 1) oder die simulierte Ratte neu entschied, ob sie weiterhin außerhalb des Baus blieb oder den Bau erneut betrat (Futtereintragestrategie 2). Die entsprechenden Werte werden immer in Sekunden angegeben. Der Erwartungswert für die Aufenthaltsdauer außerhalb betrug also 5 Minuten, bis die simulierten Tiere wieder den Bau betraten oder sich neu entschieden. Die Standardabweichung betrug eine Minute. Außerdem wurde eine Futterfundwahrscheinlichkeit von $y = (0{,}6/360) * i$ (vgl. Formel 3.1), wobei i die Aufenthaltsdauer außerhalb des Baus in Sekunden angibt, verwendet. Die Futtergrößen wurden hierbei als normalverteilt angenommen. Die entsprechenden Werte werden immer in mg angegeben. Eine Normalverteilung wird mit N(x/y) angegeben. X entspricht dabei dem Erwartungswert der Futtergröße in mg und y gibt die Standardabweichung in mg an.

4.1 Benötigte Futtergröße für eine konstante Populationsgröße

Das Überleben der Agenten ist gewährleistet, wenn Hunger- und Müdigkeitsstatus nicht zu sehr ansteigen. Gerät einer dieser Werte außer Kontrolle stirbt der Agent entweder direkt an einem zu hohen Hungerwert oder indirekt im Schlaf, da hier der Hungerwert weiter ansteigt und somit bei zu großer Müdigkeit ebenfalls ein Hungertod eintreten kann.

Als Erstes sollte nun festgestellt werden, welche Futtergröße für das Überleben mehrerer simulierter Ratten im Bau gewählt werden muss und welche Auswirkung die Futtergröße auf die Hunger- und Müdigkeitswerte der Tiere hat. Außerdem soll aufgezeigt werden welchen Einfluss eine zu geringe Futtergröße auf die

Hunger- und Müdigkeitswerte und das Verhalten der simulierten Ratten hat. Zusätzlich wurde der Einfluss der Populationsgröße auf die Entwicklung der Hunger- und Müdigkeitswerte untersucht. Hierbei zeigt sich, dass die Agenten eine Futtergröße von N(220/60) für das Überleben benötigten. Dies ist von der Futtereintragestrategie unabhängig. Die Tiere zeigten eine geringe Schwankung innerhalb der Hunger- und Müdigkeitswerte (Abbildung 4.1 A und Abbildung 4.2 A) bei einer geringen Populationsdichte. Es zeigt sich, dass die simulierten Ratten mit dieser Bedingung kürzere Schlafphasen durchführen und auch der Hungerwert auf einem sehr geringen Level verbleibt. Nimmt hingegen die Populationsdichte zu, steigen auch die gemessenen Hunger- und Müdigkeitslevel und es kann zum Tod einiger simulierter Agenten kommen (Abbildung 4.1 B und Abbildung 4.2 B).

Bei akutem Futtermangel gehen die Tiere bei der Futtereintragestrategie 2 erst ab einem Wert von 396 schlafen (Abbildung 4.3 B). Beobachtungen der Hungerwerte bei einem zu geringen Futteraufkommen bei einer Futtergröße von N(210/60) (Abbildung 4.3 A) zeigen, dass die simulierten Tiere verhungern. Dies ist durch einen steigenden Hungerwert zu erkennen, der lediglich kurzzeitig durch einen Futterfund reduziert werden kann. Die Agenten können ihren Hunger nicht vollständig stillen und sterben bei einem zu hohen Hungerstatus.

Ein zu geringes Futterangebot führt zu einem Anstieg der Aufenthaltszeit der Tiere außerhalb des Baus. Bei ausreichendem Futterangebot verbringen die Tiere der Futtereintragestrategie 1 rund 47% ihrer Zeit außerhalb des Baus, ist das Futterangebot zu gering mit einer Futtergrößenverteilung von N(210/60), verbringen sie 49% ihrer Zeit außerhalb des Baus und bewegen sich weniger im Innern. Hier sinkt der Wert von MakeAStep von 3%, bei ausreichendem Futter, auf 1% bei einem zu geringen Futterangebot von N(210/60). Bei der Futtereintragestrategie 2 bleiben die Tiere so lange wie möglich außerhalb des Baus und gehen nur in den Bau, um zu schlafen. Dies ist zum einen an ihren Müdigkeitswerten erkennbar, denn die Tiere schlafen erst nach Erreichen des Wertes 396, der sie zwingt schlafen zu gehen und zum anderen an Hand der außerhalb verbrachten Zeit von 58%. Bei einem ausreichenden Futterangebot verbringen die Tiere hingegen höchstens 55% ihrer Zeit außerhalb des Baus. Dies bedeutet, dass die Tiere bei einem zu geringen Futterangebot ihr gesamtes Futter außerhalb des Baus fressen und ihre gesamte Wachphase mit Futtersuchen und fressen verbringen. In den Bau gehen die Tiere lediglich zum Schlafen. Beides deutet darauf hin, dass kein Futtervorrat innerhalb des Baus angelegt werden kann, sondern die Tiere das gefundene Fressen entweder sofort innerhalb des Baus verzehren (Futtereintragestrategie 1) oder bereits außerhalb des Baus verzehren (Futtereintragestrategie 2). Genauere Untersuchung zum Futtervorrat werden im anschließenden Abschnitt 4.2.3 vorgestellt.

Der Einfluss der Populationsgröße zeigt sich bei einer größeren Population. Hier führen einzelne auftretende Erhöhungen der Schlaf- und Hungerwerte zum Sterben einzelner Ratten. Ist die Populationsdichte zu hoch, verbleibt der Hunger- und Müdigkeitslevel auf einem erhöhten Level (Abbildung 4.1 B und Abbildung 4.2

B) und es kommt zu Todesfällen und somit einer Reduktion der Rattenanzahl bis der Bau und das Futterangebot eine ausreichende Grundlage für die verbleibenden Ratten bieten.

Im weiteren Verlauf dieser Studie sollte nun aufgezeigt werden, welche Rolle die Futtermenge, aber auch die Größe und die Struktur des Baus, wie auch die Bewegungsstrategie der Agenten und deren Aktionswahl, unter den unterschiedlichen Futtereintragestrategien, unter anderem bei Sterbeprozessen, spielen. Möglichkeiten für Todesfälle sind etwa gegenseitige Blockaden der Ratten, so dass sie nicht an Schlafplätze und Vorratskammern gelangen können und auch nicht den Bau verlassen können oder eine kurzzeitige Nahrungsknappheit. Ein genauerer Blick auf die Aktionen der Tiere, das Futterangebot und die Bewegungen der Agenten innerhalb und außerhalb des Baus soll in den folgenden Abschnitten geworfen werden.

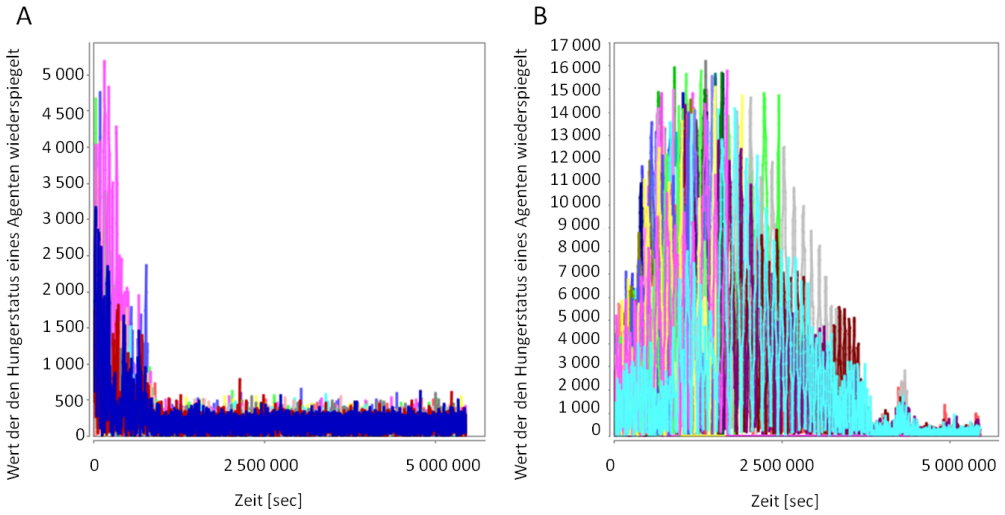

Abbildung 4.1: Entwicklung des simulierten Hungerlevels bei einer normalverteilten Futtergröße von N(220/60). Simulationen mit unterschiedlicher Anzahl an Agenten und einer normalverteilten Futtergrößenwahrscheinlichkeit von N(220/60) unter Verwendung der Futtereintragestrategie 1. Hierbei entspricht jede farbliche Linie dem zeitlichen Verlauf des Hungerwertes einer simulierten Ratte. Stirbt eine Ratte sinkt der Hungerlevel auf null ab und die Linie wird in der Zukunft nicht mehr angezeigt. A) Übersicht über den Hungerlevel bei einer Population aus 10 Agenten. B) Übersicht über den Hungerlevel bei einer Population aus 40 Agenten.

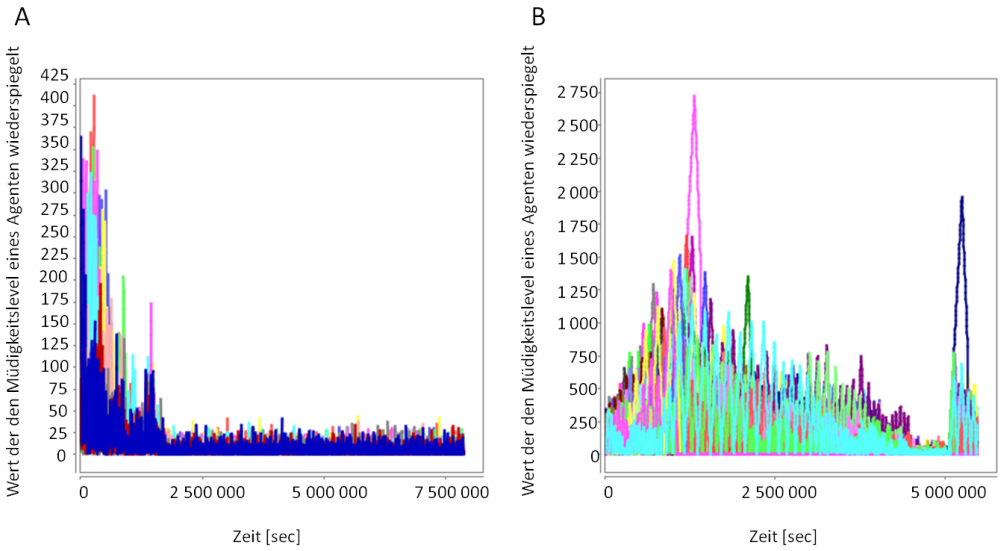

Abbildung 4.2: Entwicklung des simulierten Müdigkeitslevels bei einer normalverteilten Futtergröße von N(220/60). Simulationen mit unterschiedlicher Anzahl an Agenten und einer normalverteilten Futtergrößenwahrscheinlichkeit von N(220/60) unter Verwendung der Futtereintragestrategie 1. Hierbei entspricht jede farbliche Linie dem zeitlichen Verlauf des Müdigkeitslevels einer simulierten Ratte. Stirbt eine Ratte sinkt der Müdigkeitslevel auf null ab und die Linie wird in der Zukunft nicht mehr angezeigt. A) Übersicht über den Müdigkeitslevel bei einer Population aus 10 Agenten. B) Übersicht über den Müdigkeitslevel bei einer Population aus 40 Agenten.

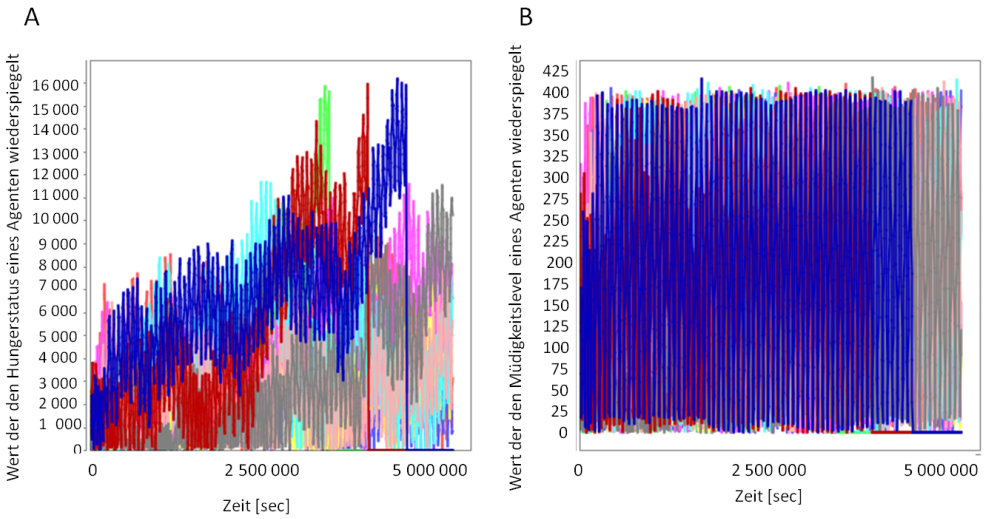

Abbildung 4.3: Entwicklung des simulierten Hungerlevels und Müdigkeitslevel bei mehreren simulierten Ratten bei einem zu geringen Futterangebot. Simulation mit 10 Agenten und einer normalverteilten Futtergrößenwahrscheinlichkeit von N(210/60) unter Verwendung der Futtereintragestrategie 2. Hierbei entspricht jede farbliche Linie dem zeitlichen Verlauf des Hunger- bzw. Müdigkeitslevels einer simulierten Ratte. Stirbt eine Ratte sinkt der Hungerwert und der Müdigkeitswert auf null ab und die Linie wird in der Zukunft nicht mehr angezeigt. A) Übersicht über den Hungerlevel bei einer Population aus 10 Agenten. B) Übersicht über den Müdigkeitslevel bei einer Population aus 10 Agenten.

4.2 Simulationen mit unterschiedlichen Futtereintrageverhaltensweisen und Futtergrößenverteilungen bei konstanter Populationsgröße

Als nächstes wurde nun untersucht, welche Veränderungen zum einen bei den beiden verwendeten Futtereintragestrategien innerhalb der Simulation auftreten und zum anderen welche Veränderungen das Futtervorkommen bzw. die Futtergröße, bei ausreichendem Futter und einem gewissen Futterüberfluss haben. Außerdem sollte gezeigt werden, was die Größe und die Struktur des Baus, was die Bewegungsstrategie der Agenten und was deren Aktionswahl unter den unterschiedlichen Futtereintragestrategien bewirken. Es wurden Simulationen mit einer festen Anzahl an Ratten zu Beginn der Simulation gemacht. Die Anzahl der Ratten zu Beginn wurde zwischen 5 und 100 variiert. Die Tiere konnten sich nicht fortpflanzen und benötigten somit alle jeden Tag die gleiche Futtermenge. Hierbei wurden die beiden Futtereintragestrategien verwendet und zwei unterschiedliche Futtergrößen angenommen. Es gab eine Untersuchung mit einer geringen Futtergröße von N(220/60) und einer größeren Futtergröße von N(250/60). Es soll nun ein Überblick über diese Simulationen gegeben werden. Die betrachteten Parameter waren die Aktionszusammensetzungen der Ratten, die unterschiedliche verbrachten Zeiten außerhalb und innerhalb des Baus, die Baunutzung und im Bau gemessene Bewegungsparameter, der angelegte Futtervorrat innerhalb des Baus und die Sterberate der Tiere in den unterschiedlichen Simulationssituationen.

4.2.1 Aktionszusammensetzung

Hier soll nun aufgezeigt werden wie sich die verbrachte Zeit der Tiere in verschiedene Aktionen gliedert und wie viel Zeit die Agenten jeweils mit den verschiedenen Aktionen verbracht haben. Hierzu wurde das Fenster ViewActions (3.6.2) erstellt. Die aufgezeigten Aktionen sind aufgeteilt nach Zeit, die mit der jeweiligen Aktion in einer Simulation über zwei Monate verbracht wurde. Diese Mittelwerte erwiesen sich als besonders stabil über alle Simulationen einer Art und die Anzahl an Agenten im Bau hatte auf diese Werte keinen Einfluss, solange eine sinnvolle Bewegungsstrategie innerhalb des Baus möglich war und sich nicht zu viele simulierte Ratten gegenseitig dauerhaft oder sehr lange blockiert haben. Solche Vorfälle können zu einer kurzfristigen Veränderung der Aktionszusammensetzungen führen. Da aber eine Vielzahl von Blockaden der Agenten zu deren Tod führen kann, da sie durch immer wieder auftretende Blockaden nicht mehr genügend Zeit haben, um ihre Bedürfnisse dauerhaft sinnvoll zu stillen, tritt diese Veränderung immer nur für eine bestimmte Zeit auf. Dies führt dann zum Sterben einiger Agenten. Ist die Anzahl der simulierten Ratten dann innerhalb der Simulation auf ein bestimmtes Level gesunken, spielt sich wieder eine feste Zusammensetzung an Aktionen ein.

Die beiden Futtereintrageverhaltensweisen zeigen hierbei unterschiedliche Ergebnisse. Außerdem zeigen sich Unterschiede bei einem knappen, aber ausreichenden Futtervorkommen und einem gewissen Futterüberfluss. Um diese Unterschiede aufzuzeigen sollen hier Daten bei einer Rattenanzahl von 10 Ratten aufgezeigt werden (Abbildung 4.4). Die Werte für Eat und Sleep entsprechen den Werten, die zum Stillen der Hunger- und Müdigkeitswerte benötigt werden. So schlafen die Ratten hier im Schnitt 10,8 h pro Tag (rund 42%) und verbringen 1,92 h am Tag mit der Nahrungsaufnahme (rund 8%). Diese Zeit muss von den simulierten Ratten somit aufgebracht werden, um ihre Hunger- und Müdigkeitswerte immer wieder zu senken. Sind dieser Werte geringer, kann die Population ihre Grundbedürfnisse nicht ausreichend stillen und zumindest ein Teil der Population stirbt.

Die restliche Zeit kann von den Tieren zur Nahrungssuche außerhalb des Baus und zu Bewegungen innerhalb des Baus genutzt werden. Betrachtet man nun die verbrachte Zeit, die die Agenten innerhalb des Baus mit Bewegungen verbringen, also die Werte von MakeAStep, so fällt auf, das die Ratten sich am wenigsten innerhalb des Baus bewegen, wenn wenig Futter vorhanden ist und die Ratten die Futtereintragestrategie 2 verwenden. Bei höherem Futtervorkommen bewegen sich die Ratten mehr innerhalb des Baus als bei geringem Futtervorkommen. Die meisten Bewegungen innerhalb des Baus werden von Tieren bei einem Futtergrößenvorkommen von N(250/60) unter der Futtereintragestrategie 1 ausgeführt. Die simulierten Ratten verbringen bei der Futtereintragestrategie 2 bei einer geringen Futtergröße mit 49% die meiste Zeit mit Futtersuchen, knapp hinter den Werten der Futtereintragestrategie 2 bei einer höheren Futtergröße und der Futtereintragestrategie 1 bei einer geringen Futtergröße mit jeweils 48 bzw. 47%. Am wenigsten Zeit verbringen simulierte Ratten mit der Futtersuche bei der Futtereintragestrategie 1 bei einer großen Futtergröße. Hier verbringen die Ratten lediglich 47% ihrer Zeit mit der Futtersuche.

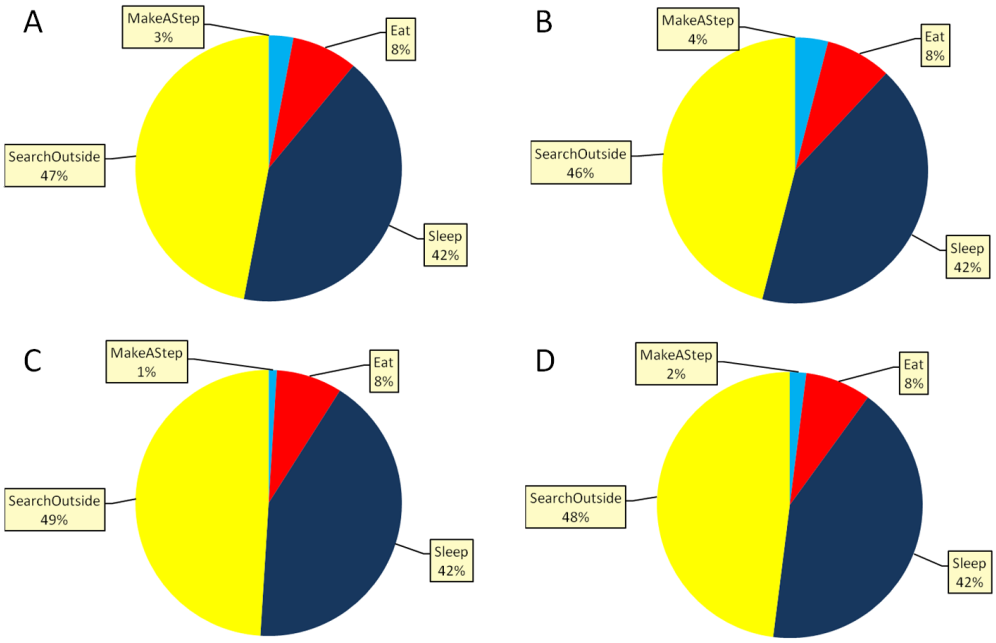

Abbildung 4.4: Beispielhafte Aufzeichnungen von Aktionszusammensetzungen bei vier verschiedenen Simulationen mit jeweils 10 Ratten pro Simulation. A) Aktionszusammensetzung unter Verwendung der Futtereintragestrategie 1 und einer Futtergröße von N(220/60). B) Aktionszusammensetzung unter Verwendung der Futtereintragestrategie 1 und einer Futtergröße von N(250/60). C) Aktionszusammensetzung unter Verwendung der Futtereintragestrategie 2 und einer Futtergröße von N(220/60). D) Aktionszusammensetzung unter Verwendung der Futtereintragestrategie 2 und einer Futtergröße von N(250/60).

4.2.2 Aufenthaltsdauer innerhalb und außerhalb des Baus

Als nächstes soll nun die unterschiedliche Zeit betrachtet werden, die die Agenten innerhalb und außerhalb des Baus verbringen und welchen Einfluss die Futtereintragestrategien und die verschiedenen Futtervorkommen hierauf haben. Bei der Futtereintragestrategie 1 und einer Futtergrößenverteilung von N(220/60) verbrachten die Tiere 47%, bei einer Futtergrößenverteilung von N(250/60) 46% ihrer Zeit außerhalb des Baus und somit 53 bzw. 54% innerhalb. Bei der Futtereintragestrategie 2 hingegen verbrachten die Tiere bei einer Futtergröße von N(220/60) 55% der Zeit außerhalb des Baus und bei einer Futtergröße von N(250/60) verbrachten die Tiere 54% ihrer Zeit außerhalb des Baus und somit 45 bzw. 46% innerhalb des Baus. Abbildung 4.5 zeigt vier Simulationen bei denen immer 10 Ratten simuliert wurden.

Da die Tiere bei der Futtereintragestrategie 2 nicht nur Futter suchen können, sondern auch außerhalb des Baus fressen können, bedeutet dies nun im Rückschluss, dass die Tiere unter dieser Strategie das meiste Futter außerhalb des Baus fressen. Von den 8% ihrer Zeit, die sie mit fressen verbringen, verwenden die Ratten 6% davon, um außerhalb des Baus zu fressen. Dies bedeutet wiederum, dass die Tiere rund 75% ihres Futters außerhalb des Baus verzehren. Die Anzahl an Ratten der Simulation hatte auch wiederum keinen Einfluss auf die Zusammensetzung der außerhalb und innerhalb des Baus verbrachten Zeit, solange auch hier eine sinnvolle Bewegungsstrategie innerhalb des Baus möglich war.

Andere Fragen die sich stellten waren zum einen, wie oft verlassen die Tiere den Bau und zum anderen, wie viel Zeit am Stück verbringen die Tiere außerhalb des Baus. Auch hier sollte wiederum festgestellt werden, ob es einen Einfluss der Agentenanzahl gibt. Daher wurde die Anzahl der Agenten innerhalb der Simulation konstant gehalten, indem die Agenten in diesen Simulationen bei einem zu hohen Hungerwert nicht aus der Simulation entfernt wurden. War innerhalb der Simulation allerdings bei mehr als 90% der Agenten der Hungerwert über den normalerweise zum Tod führenden Wert gestiegen, wurde auf eine weitere Erhöhung der Rattenanzahl innerhalb der Simulationen verzichtet. Es wurden jeweils 10 Simulationen für jede Rattenanzahlen durchgeführt und hierbei über 2 Monate aufgezeichnet, wie oft ein Bau verlassen wurde (vgl. Abbildung 4.6).

Beide Strategien zeigen ein selteneres Verlassen des Baus auf einen Agenten gerechnet, je mehr Agenten sich innerhalb der Simulation befinden. Hierbei zeigt sich, dass Tiere der Futtereintragestrategie 1 den Bau oft mehr als doppelt so oft verlassen als Tiere der Futtereintragestrategie 2. Bei der Futtereintragestrategie 1 hat das Futtergrößenvorkommen keinen erkennbaren Einfluss auf die Frequenz mit dem die Tiere den Bau verlassen (siehe Abbildung 4.6 A). Beim Futtereintrageverhalten 2 hingegen zeigt sich ein Einfluss der Futtergröße auf das Verlassen des Baus. Bei einer größeren Futtergröße gehen die Ratten öfters nach draußen als bei einer geringeren Futtergröße (siehe Abbildung 4.6 B).

Zusätzlich interessierte uns was für eine Aufenthaltszeitenverteilung sich außer-

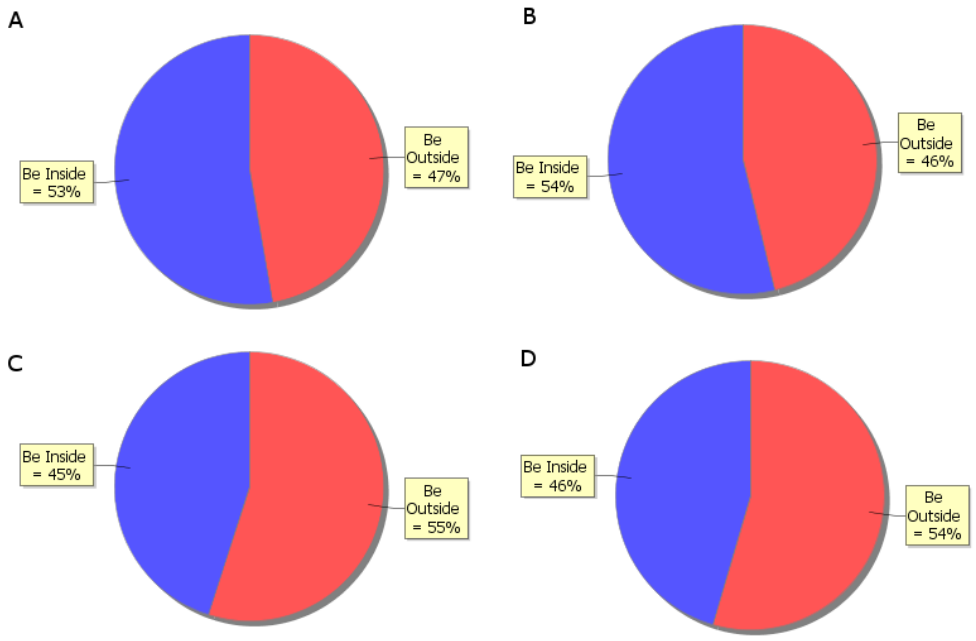

Abbildung 4.5: Beispielhafte Aufzeichnungen der Aufenthaltsdauer innerhalb und außerhalb des Baus bei vier verschiedeneren Simulationen mit jeweils 10 Ratten und einem Simulationszeitraum von 2 Monaten pro Simulation. A) Aufenthaltsdauer innerhalb und außerhalb des Baus unter Verwendung der Futtereintragestrategie 1 und einer Futtergröße von N(220/60). B) Aufenthaltsdauer innerhalb und außerhalb des Baus unter Verwendung der Futtereintragestrategie 1 und einer Futtergröße von N(250/60). C) Aufenthaltsdauer innerhalb und außerhalb des Baus unter Verwendung der Futtereintragestrategie 2 und einer Futtergröße von N(220/60). D) Aufenthaltsdauer innerhalb und außerhalb des Baus unter Verwendung der Futtereintragestrategie 2 und einer Futtergröße von N(250/60).

halb des Baus unter den Futtereintragestrategien einstellt. Hierzu wurde der ViewTimesOutside (3.6.2) entwickelt. Tiere unter der Futtereintragestrategie 1 zeigen, wie erwartet und programmiert, eine normalverteilte Aufenthaltszeitenverteilung mit einem Erwartungswert von 300 Sekunden und einer Standardabweichung von 60 Sekunden. Da die Aufenthaltsdauer von der Futtermenge oder dem Hunger der Ratten in dieser Futtereintragestrategie unabhängig ist, zeigt sich unter den verschiedenen Futtergrößenverteilungen die gleiche Verteilung (siehe Abbildung 4.7 A). Tiere unter der Futtereintragestrategie 2 zeigen eine andere Verteilung der Aufenthaltszeiten (Abbildung 4.7 B, bei einer geringen Futtergröße und C bei einer größeren Futtergröße). Hier zeigt sich wiederum, dass die simulierten Ratten bei einem geringen Futtervorkommen weniger oft den Bau verlassen als bei einem höheren Futtervorkommen. Außerdem sind die Aufenthaltszeiten außerhalb des Baus bei der Futtereintragestrategie 2 länger als bei der Futtereintragestrategie 1.

Bei der Futtereintragestrategie 2 hängt die Aufenthaltsdauer außerhalb des Baus von verschiedenen Faktoren ab. Der Zeitrahmen, in dem die Entscheidung getroffen wird, ob der Bau betreten wird oder nicht, ist in den beiden Futtergrößenvorkommen gleich. Allerdings entscheidet die simulierte Ratte unter den beiden Futtergrößenvorkommen aufgrund unterschiedlicher Zustände, ob sie den Bau wieder betritt. Prinzipiell gilt in unserer Simulation: wird sie sehr müde, d.h. befindet sich ihr Müdigkeitslevel über 396 oder möchte sie Futter innerhalb des Baus ablegen, betritt sie den Bau erneut. Ansonsten verbleibt sie weiterhin außerhalb des Baus. Welche Bedingungen hier zu dem Entschluss einer simulierten Ratte führen den Bau zu verlassen, kann zu diesem Zeitpunkt noch nicht aufgezeigt werden.

Bei einem zu geringen Futterangebot bei einer Futtergröße von N(220/60) zeigt sich, dass die Tiere sehr viel seltener aus dem Bau gehen und längere Aufenthaltszeiten außerhalb des Baus aufweisen (siehe Abbildung 4.7 D). Hierbei fällt auf, dass die Ratten oftmals fast 40 000 Sekunden außerhalb des Baus verbringen. Dies entspricht der am längsten möglichen Aufenthaltszeit von simulierten Ratten, da sie hierzu direkt nach dem Schlafen aus dem Bau gehen müssen und erst bei einem Müdigkeitswert von 396 in den Bau zurückkehren. Dies bedeutet, die Tiere fressen in diesem Fall das ganze gefundene Fressen außerhalb des Baus.

Die Agentenanzahl innerhalb des Baus zeigte in keiner Simulationsbedingung einen Einfluss auf die Aufenthaltszeitenverteilungen außerhalb des Baus, wenn die entsprechenden Verteilungen auf einen Agenten bezogen werden und die Agentenanzahl noch eine sinnvolle Bewegungsstrategie innerhalb des Baus zuließ.

Um weitere Aussagen über die Faktoren die zum Verlassen des Baus führen und zu den Zeiten, die die Tiere außerhalb des Baus verbringen, treffen zu können, wurden zwei neue Views entworfen, die ebenfalls die Aufenthaltszeiten außerhalb des Baus und deren Häufigkeit aufzeigen. Hierbei wurde unterschieden, ob die Tiere den Bau verließen, da sich kein Futter innerhalb des Baus befand, die Agenten aber einen Hungerwert erreichten, der ein erneutes Futtersuchen und anschließendes Futterfressen veranlasste oder ob die simulierten Ratten aufgrund einer zufälligen Wahl bei einer guten Verfassung zwischen Sleep, RandomWalk und GoOutside, sich für das Futtersuchen außerhalb des Baus entschieden hatten.

Bei der Futtereintragestrategie 1 zeigten die Ratten in allen Bedingungen eine normalverteilte Aufenthaltszeitenverteilung mit einem Erwartungswert von 300 Sekunden und einer Standardabweichung von 60 Sekunden (Abbildung 4.8). Dies entspricht der vom Programm vorgeschriebenen Aufenthaltszeitenverteilung außerhalb des Baus. Hierbei zeigt sich, dass bei einem geringen Futtergrößenvorkommen von N(220/60), die Agenten häufig den Bau verließen auf Grund von akutem Hunger, der gestillt werden musste (Abbildung 4.8 A mittlere Abbildung). Allerdings zeigte sich auch, dass die Tiere bereits bei einem geringen Futtervorkommen genauso häufig den Bau zur reinen Futtersuche und nicht auf Grund eines hohen Hungerwertes (Abbildung 4.8 A rechte Abbildung) verließen.

Bei einer größeren Futtergröße von N(250/60) zeigen die Agenten seltener eine

Futtersuche auf Grund von akutem Hunger (Abbildung 4.8 B mittlere Abbildung). Die Mehrzahl der Entschlüsse aus dem Bau zu gehen wird hier zufällig bei guten Hunger- und Müdigkeitswerten gefällt (Abbildung 4.8 B rechte Abbildung). Dies führt zu einem weiteren Futtereintragen innerhalb des Baus und schlussendlich einer Vergrößerung des Futtervorrats.

Bei der Futtereintragestrategie 2 sieht man wiederum, dass die Agenten den Bau bei einer größeren Futtergröße von N(250/60) häufiger verlassen, als bei einer geringeren Futtergröße von N(220/60). Außerdem verlassen die Agenten bei einer geringen Populationsdichte den Bau fast nur noch, um mehr Futter in den Bau einzutragen (Abbildung 4.9 C und D). Aus akutem Hunger verlassen die Tiere bei beiden getesteten Futtergrößenverteilungen von N(220/60) und N(250/60) den Bau nur noch sehr selten und die Mehrzahl der Bauaustritte geschieht bei einer guten Grundverfassung der Tiere zufällig, so dass die Mehrzahl der Tiere den Futtervorrat innerhalb des Baus vergrößern wird. Man kann außerdem erkennen, dass die Tiere, die den Bau mit einem akuten Hungerwert verlassen länger außerhalb des Baus bleiben als simulierte Tiere die einen guten Hunger- und Müdigkeitswert beim Austritt aus dem Bau aufweisen. Während die nicht hungrigen Tiere das Futter in den Bau eintragen und dort in Vorratskammern ablegen, fressen die hungrigen Tiere das Futter direkt außerhalb des Baus und suchen dann erneut weiter Futter. Erst wenn diese Tiere ihren Hunger gestillt haben oder zu müde werden, gehen sie in den Bau.

Bei zu geringem Futtervorkommen verlassen die Agenten den Bau seltener, gleichzeitig erhöht sich allerdings die Aufenthaltszeit der Ratten außerhalb des Baus (siehe Abbildung 4.9 E) und es zeigt sich, dass die Tiere häufig aus akutem Hunger den Bau verlassen (Abbildung 4.9 E Mitte) und selten zum Futtereintragen. Außerdem verbringen die hungrigen Rattenoftmals soviel wie möglich Zeit außerhalb vom Bau und gehen erst bei einem zu hohen Müdigkeitswert in den Bau.

Dies bedeutet nun zusammenfassend: Bei der Futtereintragestrategie 1 gehen die Ratten bei kleinerer Futtergröße (N(220/60)) öfters aus akutem Hunger aus dem Bau als bei einer höheren Futtergröße (N(250/60)). Dafür verlassen die Agenten den Bau zur Futtervorratsanlegung bei geringer Futtergröße seltener als bei einer größeren Futtergröße. Die Verteilung der Aufenthaltszeit außerhalb ist hierbei gleich. Bei der Futtereintragestrategie 2 verlassen die Tiere bei einer zu geringen Futtergröße (N(210/60)) den Bau häufiger aus akutem Hunger und fehlendem Futter innerhalb des Baus als bei größeren Futtergrößenverteilungen. Bei den beiden getesteten Futtergrößen, die ein ausreichendes Futterangebot darstellen, wird nur noch sehr selten der Bau aus akutem Hunger verlassen. Dies spricht für einen schnellen Aufbau eines Futtervorrats. Außerdem verbringen Agenten, die einen großen Hunger haben mehr Zeit außerhalb des Baus, als Tiere, die keinen erhöhten Hungerwert zeigen, da diese Tiere das gefundene Futter sofort verspeisen und weiteres Futter suchen. Agenten, die Futter in den Bau eintragen möchten, verbringen eine kürzere Zeit außerhalb des Baus, da sie das Futter nicht ganz außerhalb des Baus verspeisen sondern in den Bau eintragen. Außerdem zeigte sich, dass die simulierten Ratten den Bau mit stei-

gender Futtergröße häufiger verlassen. Dies ist durch ein Anstieg der Austritte bei einem niedrigen Hunger- und Müdigkeitswert zur Futtereinlagerung zu erklären.

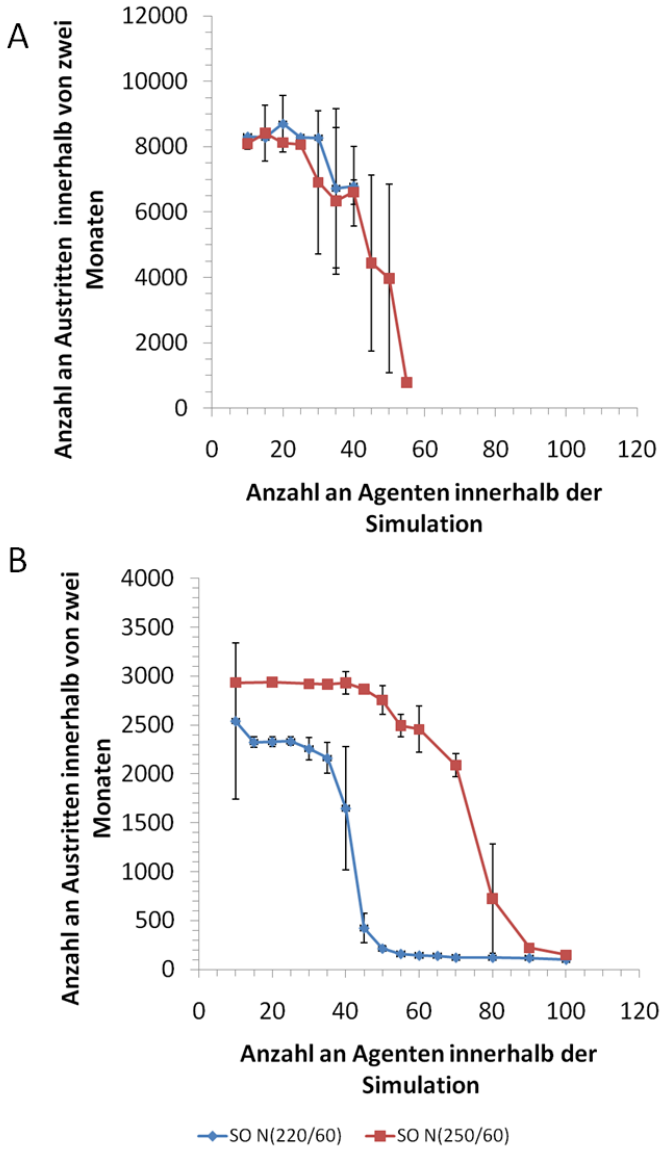

Abbildung 4.6: Beispiel der Anzahl der Ausgangsnutzungen innerhalb von zwei Monaten. Verließ eine Ratten den Bau durch einen Ausgang, wurde dies festgestellt und vermerkt. Die Anzahl an Agenten wurde während der Simulationen konstant gehalten. Es wurden Simulationen mit den Futtergrößen N(220/60) und N(250/60) berechnet. A) Anzahl an Ausgangsnutzungen für eine simulierte Ratte bei Verwendung der Futtereintragestrategie 1. B) Anzahl an Ausgangsnutzungen für eine simulierte Ratte bei Verwendung der Futtereintragestrategie 2.

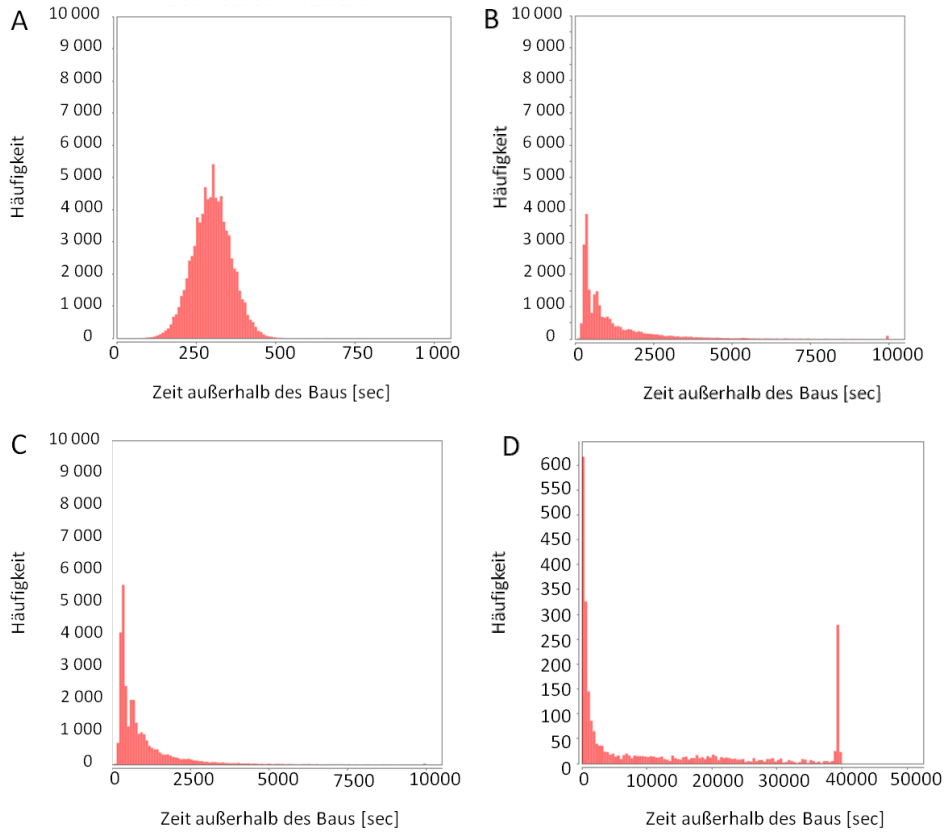

Abbildung 4.7: Verteilung der Aufenthaltszeiten außerhalb des Baus innerhalb einer zweimonatigen Simulation bei unterschiedlichen Futtergrößen und Futtereintragestrategien. Es handelte sich um zweimonatige Simulationen mit 10 Agenten. Hierbei wird die unterschiedliche Verteilung von den verschiedenen Aufenthaltszeiten außerhalb des Baus gezeigt. A) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 1. Die Simulationen unterscheiden sich nicht bei unterschiedlichen Futtergrößen, da die Tiere immer in den Bau zurückkehren. B) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(220/60) angenommen. C)Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(250/60) angenommen. D)Verteilung der Aufenthaltszeit außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(210/60) angenommen.

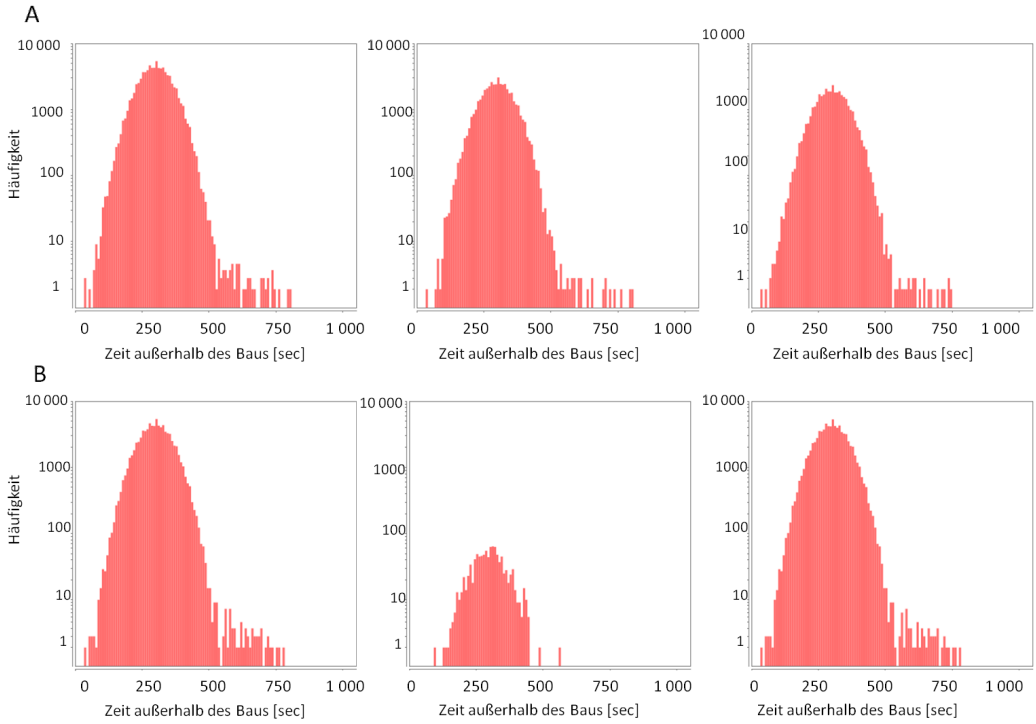

Abbildung 4.8: Verteilung der Aufenthaltszeiten außerhalb des Baus bei unterschiedlicher Intention des Aufenthalts innerhalb einer zweimonatigen Simulation bei unterschiedlichen Futtergrößen und Futtereintragestrategien. Es handelte sich um zweimonatige Simulationen mit 10 Agenten. Das linke Bild zeigt die Aufenthaltszeitverteilung der Ratten außerhalb des Baus insgesamt, die mittlere Abbildung die Aufenthaltszeitverteilung außerhalb des Baus, wenn die Agenten aufgrund von Hunger den Bau verließen, die rechte Abbildung zeigt die Aufenthaltszeitverteilung außerhalb des Baus, wenn die Agenten zur Futtersuche den Bau verließen ohne dass ihr Hungerwert den kritischen Wert zur Futteraufnahme überschritten hatte. In diesem Fall wird das gefundene Futter anschließend im Bau eingelagert. A) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 1. Die Futtergröße wurde hierbei normalverteilt mit N(220/60) angenommen. B) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(250/60) angenommen.

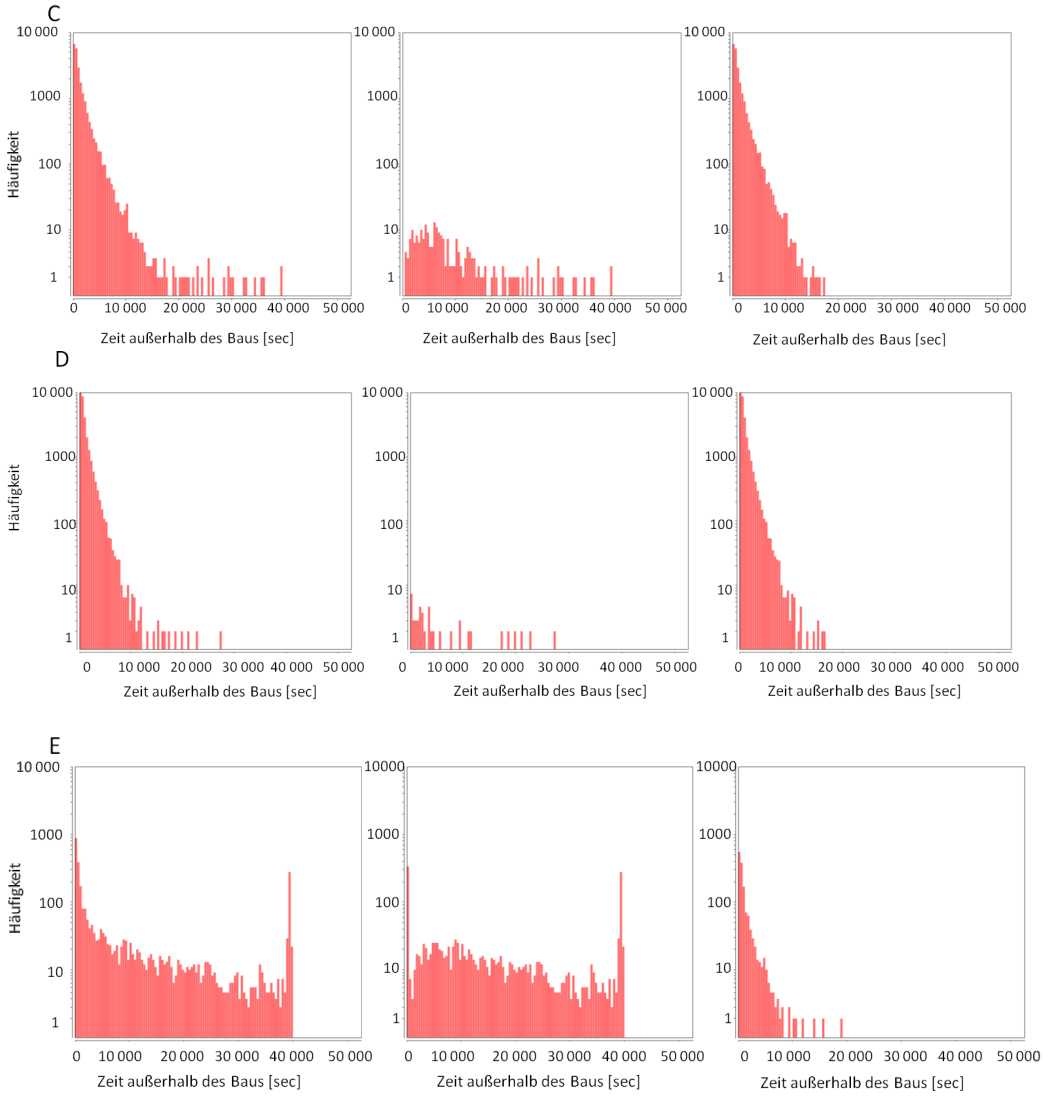

Abbildung 4.9: Verteilung der Aufenthaltszeiten außerhalb des Baus bei unterschiedlicher Intention des Aufenthalts innerhalb einer zweimonatigen Simulation bei unterschiedlichen Futtergrößen und Futtereintragestrategien. Es handelte sich um zweimonatige Simulationen mit 10 Agenten. Das linke Bild zeigt die Aufenthaltszeitverteilung der Ratten außerhalb des Baus insgesamt, die mittlere Abbildung zeigt die Aufenthaltszeitverteilung außerhalb des Baus, wenn die Agenten aufgrund von Hunger den Bau verließen, die rechte Abbildung zeigt die Aufenthaltszeitverteilung außerhalb des Baus, wenn die Agenten zur Futtersuche den Bau verließen ohne dass ihr Hungerwert den kritischen Wert zur Futteraufnahme überschritten hatte. In diesem Fall wird das gefundene Futter anschließend im Bau eingelagert. C) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(220/60) angenommen. D) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(250/60) angenommen. E) Verteilung der Aufenthaltszeiten außerhalb des Baus unter Verwendung der Futtereintragestrategie 2. Die Futtergröße wurde hierbei normalverteilt mit N(210/60) angenommen.

4.2.3 Futtervorrat innerhalb des Baus

Es zeigt sich, dass die Agenten der Futtereintragestrategie 1 bei einer geringen Futtergröße von N(220/60) noch keinen dauerhaften Futtervorrat innerhalb des Baus aufbauen können (siehe Abbildung 4.10 A), wohingegen Agenten, die die Futtereintragestrategie 2 verwenden, durchaus einen stetig wachsenden Futtervorrat anlegen können (siehe Abbildung 4.10 B), wenn sich nicht mehr als 40 simulierte Ratten innerhalb des Baus befinden. Befinden sich mehr als 40 Agenten innerhalb des Baus, kann auch hier kein dauerhafter Futtervorrat angelegt werden. Bei einer Futtergrößenverteilung von N(250/60) führen beide Futtereintragestrategien zu einem stetig wachsenden Futtervorrat. Hier konnte ein stetig wachsender Futtervorrat bei allen durchgeführten Simulationen der Futtereintragestrategie 1 festgestellt werden, bei der Futtereintragestrategie 2 konnte, bei bis zu 70 Tieren, immer eine stetig wachsende Futtermenge verzeichnet werden. Ab einer Populationsdichte von über 80 Ratten konnte allerdings kein dauerhafter Futtervorrat mehr angelegt werden. Es wurden nur vereinzelte Futterstücke abgelegt. Ein Vergleich der beiden Futtervorräte bei den unterschiedlichen Futtereintragestrategien zeigt Abbildung 4.11. Hierbei zeigt sich, dass Futtereintragestrategie 2 erfolgreicher ist als die Futtereintragestrategie 1. Außerdem sieht man, dass sich die Anzahl an Ratten, bei dem die meiste Futtermenge im Bau eingetragen wurde, bei den beiden Strategien unterscheiden. So wird bei einer Anzahl von 35 Agenten innerhalb der Futterstrategie 1 die höchste absolute Futtermenge eingetragen. Bei der Futtereintragestrategie 2 hingegen wird das meiste absolute Futter bei einer Rattenanzahl von 50 Ratten eingetragen. Betrachtet man nun allerdings die Menge an Futter die sich pro Agent innerhalb des Baus befindet, erkennt man, dass die Futtermenge immer mehr sinkt, je mehr Tiere sich innerhalb der Simulation befinden, egal welche Futtereintragestrategie verwendet wird (siehe Abbildung 4.12).

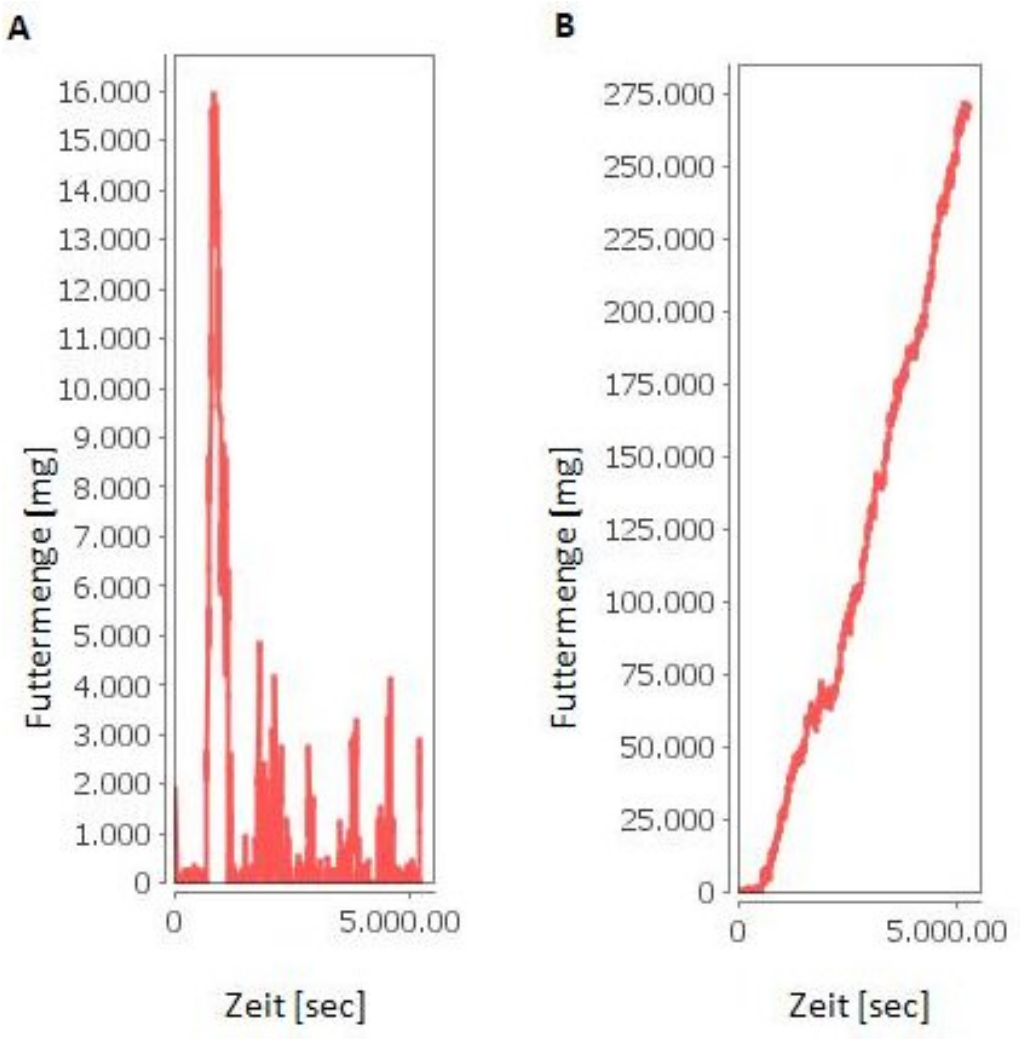

Abbildung 4.10: Entwicklung des Futtervorrats bei geringem Futterangebot und verschiedener Futtereintragestrategie. Simulation mit 10 Ratten und einer normalverteilten Futtergrößenwahrscheinlichkeit von N(220/60). A) Futtereintragestrategie 1: es wird kein wachsender Futtervorrat angelegt. B) Futtereintragestrategie 2: es kann ein wachsender Futtervorrat angelegt werden.

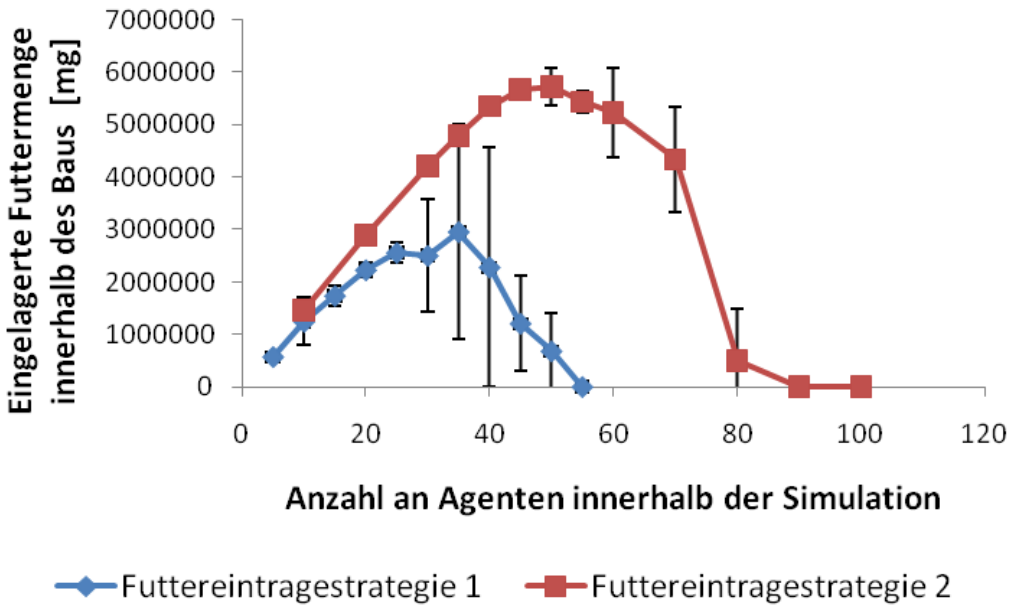

Abbildung 4.11: Gesamte Futtermenge innerhalb des Baus nach einer zweimonatigen Simulationszeit bei einer normalverteilten Futtergröße von N(250/60) und einer konstanten Anzahl an Agenten.

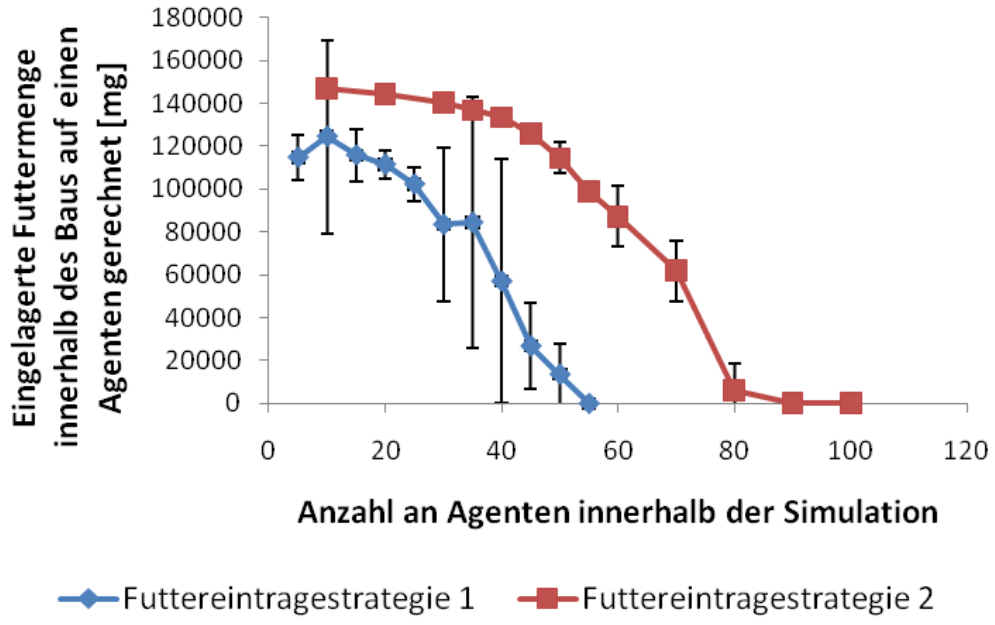

Abbildung 4.12: Futtermenge pro simulierte Ratte innerhalb des Baus nach einer zweimonatigen Simulationszeit bei einer normalverteilten Futtergröße von N(250/60) und einer konstanten Anzahl an Agenten.

4.2.4 Baunutzung

Zusätzlich soll nun die Baunutzung betrachtet werden. Hierzu wurde die Ansicht ViewBurrowUsage erstellt. Beispielhaft werden hier Ansichten mit einer unterschiedlichen Futtergrößenverteilung gezeigt. Die Simulationen, die diese Baunutzungsabbildungen erzeugten, verfügen wiederum über keine Vermehrungsrate und auch keinen simulierten Prädator. Dies führt somit bei einer ausreichenden Futtermenge und einer nicht zu hohen Rattenanzahl zu einer gleichbleibenden Populationsgröße.

Innerhalb des Baus zeigen die beiden Futtereintragestrategien bei ausreichend vorhandenem Futter keine unterschiedliche Baunutzung. Dies verwundert nicht, da die Bewegungsstrategie innerhalb des Baus zwischen den beiden Futtereintragestrategien nicht verändert wurde. Auch die Erhöhung der Rattenanzahl änderte die Baunutzung nicht, auch wenn man die Anzahl der Ratten soweit ansteigen lässt, dass eine gleichbleibende Populationsgröße nicht mehr möglich ist, da sich die Tiere zu sehr behindern und daher Teile der Population sterben.

Abbildung 4.13 zeigt eine solche Baunutzung bei ausreichendem Futter. Hierbei werden Kanten besonders häufig benutzt, die auf direktem Weg zwischen mehreren Ausgängen und einer Kammer liegen. Dies zeigt, dass die Tiere, bei einer niedrigen Populationsanzahl, die direkten Wege zwischen einem Ausgang und einer Kammer häufig benutzen. Hierbei werden die Tiere, vor allem nach einem Aufenthalt außerhalb, ihr Futter in eine Kammer bringen, um es dort zu fressen oder einzulagern oder die Tiere verlassen den Bau häufig nach dem Schlafen um außerhalb des Baus Futter zu suchen. Außerdem ist die Nutzung der Ausgänge als Ausgang und Eingang aufgezeichnet. Die Tiere betreten den Bau innerhalb der Simulation durch einen zufälligen Eingang. Daher sind hier keine Unterschiede sichtbar. Bei der Nutzung als Ausgänge werden drei Ausgänge besonders häufig benutzt. Sie sind die nächsten Ausgänge zu mehreren Kammern. Daher werden sie häufiger benutzt als andere Ausgänge.

Wird die Simulation allerdings bei einem zu knappen Nahrungsangebot durchgeführt, zeigt sich eine unterschiedliche Baunutzung bei den zwei verwendeten Futtereintragestrategien.

Bei der Futtereintragestrategie 1 werden von den Tieren hauptsächlich die direkten Verbindungen zwischen Kammern und Ausgängen stark benutzt (Abbildung 4.14). Zuvor stark genutzte Bereiche werden seltener aufgesucht, allerdings wird der gesamte Bau weiterhin genutzt und hierbei alle Tunnelsysteme mindestens durchschnittlich stark genutzt. Bei der Wahl der Ausgänge ist keine Veränderung sichtbar.

Bei der Futtereintragestrategie 2 ändert sich die Baunutzung drastisch (Abbildung 4.15). Auch hier werden nur noch Wege zwischen einem Schlafplatz und einem Ausgang besucht. Bewegungen zu anderen Bereichen des Baus finden nicht mehr statt.

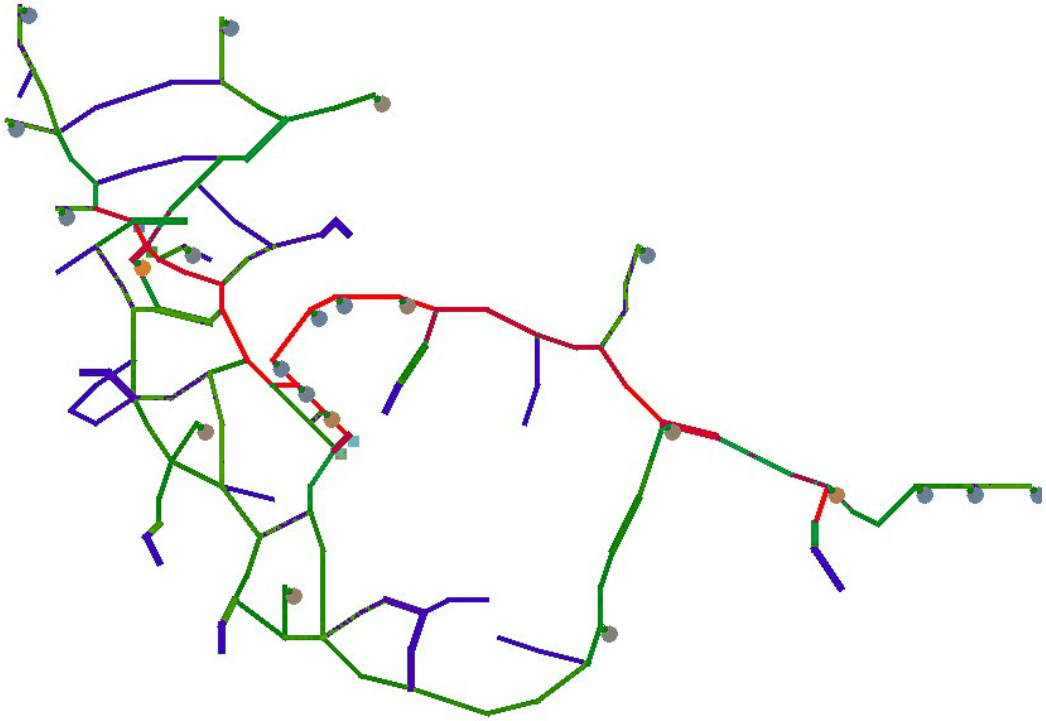

Abbildung 4.13: Beispiel einer Ansicht von ViewBurrowUsage mit 15 simulierten Ratten. Es wurde die Futtereintragestrategie 1 benutzt. Die Futtergrößen waren normalverteilt mit N(220/60). Die Nutzung des Baus ist farblich codiert: rote Bereiche zeigen häufig benutzte Kanten, grün im mittleren Durchschnitt benutzte Kanten und Ausgänge, blau selten benutzte Kanten und Ausgänge. Kammern verfügen über eine etwas dickere Kante. Außerdem werden Ausgänge als runde Kreise dargestellt. Hierbei zeigen die inneren Kreise die Nutzung als Eingang, die äußeren Kreise die Nutzung als Ausgang.

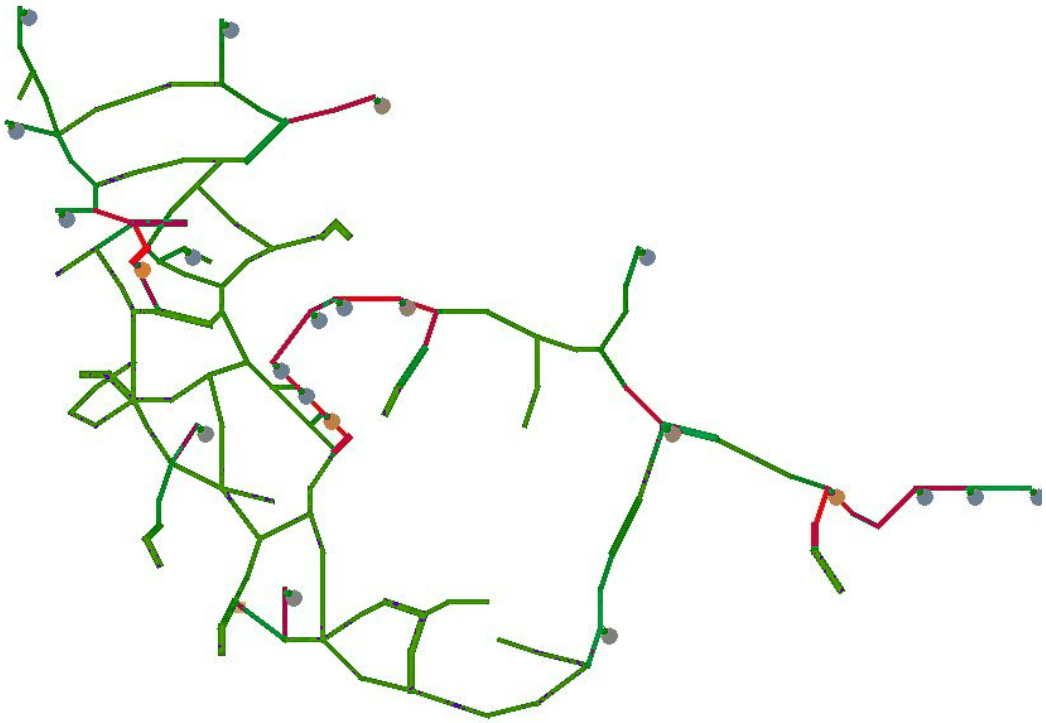

Abbildung 4.14: Beispiel einer Ansicht von ViewBurrowUsage mit 10 simulierten Ratten bei einem zu geringen Futterangebot. Es wurde die Futtereintragestrategie 1 benutzt. Die Futtergrößen waren normalverteilt mit N(210/60). Dies entspricht einem zu geringen Nahrungsangebot und die Tiere verhungern mit der Zeit. Die Nutzung des Baus ist hierbei farblich codiert: rote Bereiche zeigen häufig benutzte Kanten und Ausgänge des Baus, grün im mittleren Durchschnitt und blau selten benutzte Kanten und Ausgänge. Kammern verfügen über eine etwas dickere Kante. Außerdem werden Ausgänge als runde Kreise dargestellt. Hierbei zeigen die inneren Kreise die Nutzung als Eingang, die äußeren Kreise die Nutzung als Ausgang.

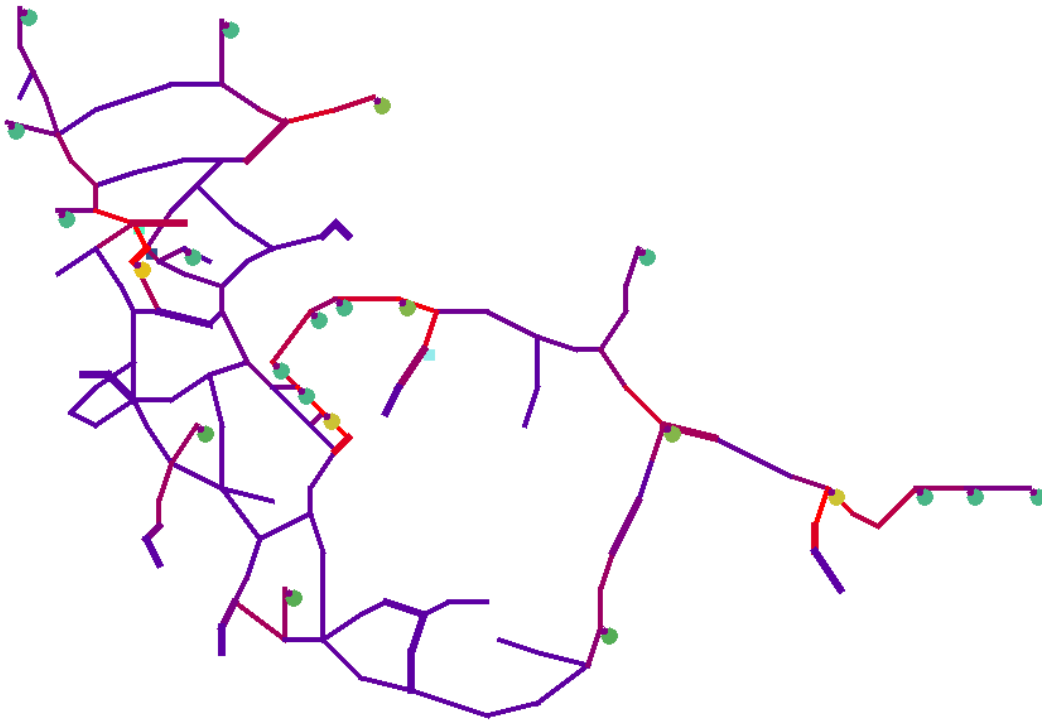

Abbildung 4.15: Beispiel einer Ansicht von ViewBurrowUsage mit 10 simulierten Ratten bei einem zu geringen Futterangebot. Es wurde die Futtereintragestrategie 2 benutzt. Die Futtergrößen waren normalverteilt mit N(210/60). Dies entspricht einem zu geringen Nahrungsangebot und die Tiere verhungern mit der Zeit. Die Nutzung des Baus ist farblich codiert: rote Bereiche zeigen häufig benutzte Kanten und Ausgänge, grün im mittleren Durchschnitt und blau selten benutzte Kanten und Ausgänge. Kammern verfügen über eine etwas dickere Kante. Außerdem werden Ausgänge als runde Kreise dargestellt. Hierbei zeigen die inneren Kreise die Nutzung als Eingang, die äußeren Kreise die Nutzung als Ausgang.

4.2.5 Bewegungsfreiheit der Agenten innerhalb des Baus

Als nächstes sollte nun ein genauer Blick auf die Bewegungsstrategie innerhalb des Baus geworfen werden und festgestellt werden, wie sich diese in Zusammenhang mit den unterschiedlichen Futtereintragestrategien auf die Population und die einzelnen Ratten in Bezug auf ihre Bewegungsfreiheit innerhalb des Baus auswirkt. Es wurden mehrere Parameter genauer betrachtet. Um eine feste Anzahl an Agenten innerhalb der Simulationen zu gewährleisten, konnten die Agenten innerhalb der hier vorgestellten Simulationen nicht an einem zu großen Hungerwert sterben, sondern verblieben auch mit einem zu hohen Hungerwert weiterhin in der Simulation.

Zu Beginn wurde betrachtet, welche Wegstrecke eine simulierte Ratte im Durchschnitt zurücklegt bis sie einer anderen Ratte begegnet. Betrachtet man den zeitlichen Verlauf dieser Werte (vgl. 3.6.2), ist erkennbar das sich bei einer geringen Anzahl an Ratten innerhalb der Simulation ein bestimmter Wert innerhalb von 2 Monaten nur sehr schwer einspielt (siehe Abbildung 4.16). Ist die Anzahl der Ratten hingegen größer, sind oftmals keine größeren Schwankungen mehr erkennbar bzw. kann eine längere Blockade mehrerer Ratten zu einem Absinken der mittleren Distanz führen, doch ist dies oftmals der Fall, so dass das Auftreten einer solchen Blockade keinen so großen Einfluss auf den Verlauf des mittleren Gesamtwertes hat. Diese Schwankungen treten bei einer geringen Rattenanzahl allerdings auf, wenn sich mehrere Ratten gleichzeitig blockieren. In diesem Fall werden viele kurze Strecken aufgezeichnet, bei denen sich viele Ratten immer wieder nach kürzester Distanz treffen. Dies kann zu einem Abfall der durchschnittlichen Wegstrecke führen, die die Agenten innerhalb der Simulation zurückgelegt haben. Löst sich diese gegenseitige Blockade auf, führt dies im Anschluss wiederum zu einer Erhöhung der durchschnittlichen zurückgelegten Wegstrecke ohne einer anderen Ratte zu begegnen. So haben mehrere kürzere gegenseitige Blockaden einen geringeren Einfluss auf die durchschnittlich zurückgelegte Wegstrecke und eine langanhaltende Blockade mit vielen Ratten hingegen einen größeren Einfluss. Bei einer geringen Anzahl an Ratten konnte sich innerhalb der zwei Monate oftmals noch kein konstanter Wert einspielen und die Tendenz war oftmals auf einen höheren Wegstreckenwert orientiert. Die zu Beginn auftretenden Blockaden hatten oftmals einen sehr starken Einfluss. Hier ist die Wahrscheinlichkeit einer Blockade höher, da sich zu Beginn der Simulation alle Tiere innerhalb des Baus befinden. Außerdem werden bei einer geringeren Anzahl an Ratten auch weniger Wegstrecken in derselben Zeit festgehalten. Aus diesen Gründen kommt es zu einem größeren Schwanken und damit zu einer höheren Standardabweichung als bei den Aufnahmen mit mehr Agenten innerhalb des Baus. Diese Schwankungen legten sich bei einer höheren Rattenanzahl. Als Beispiel kann eine Aufzeichnung mit 45 Ratten herangezogen werden (Abbildung 4.17). In diesem Fall kommt es sozusagen immer zu gegenseitigen Blockaden von mehreren Ratten. Trotzdem soll eine durchschnittliche Momentaufnahme gezeigt werden, die die durchschnittliche zurückgelegte Wegstrecke angibt nach dem einem anderen Agenten begegnet wurde nach einer zweimonati-

gen Simulation. Es wurden die Futtereintragestrategien wiederum mit den beiden normalverteilten Futtergrößen von N(220/60) und N(250/60) getestet. Es zeigt sich, dass sich die durchschnittlich zurückgelegten Strecken innerhalb des Baus, nach der die simulierten Ratten auf eine andere Ratte trafen, zwischen den verschiedenen Futtergrößen und Futtereintragestrategien nicht stark unterschieden. Die Anzahl der Ratten zeigt einen starken Einfluss auf diesen Wert (siehe Abbildung 4.18). Je mehr Ratten sich innerhalb einer Simulation befinden, desto weniger Wegstrecke kann zurückgelegt werden bis einer anderen Ratte begegnet wird. Ab einer Rattenanzahl von 35 Ratten zeigten alle Simulationen einen durchschnittlichen Wert unter einem Meter. Dies bedeutet, die Ratten begegnen sich innerhalb des Baus besonders häufig und müssen sich im Schnitt bei einer höheren Populationsdichte öfters gegenseitig ausweichen.

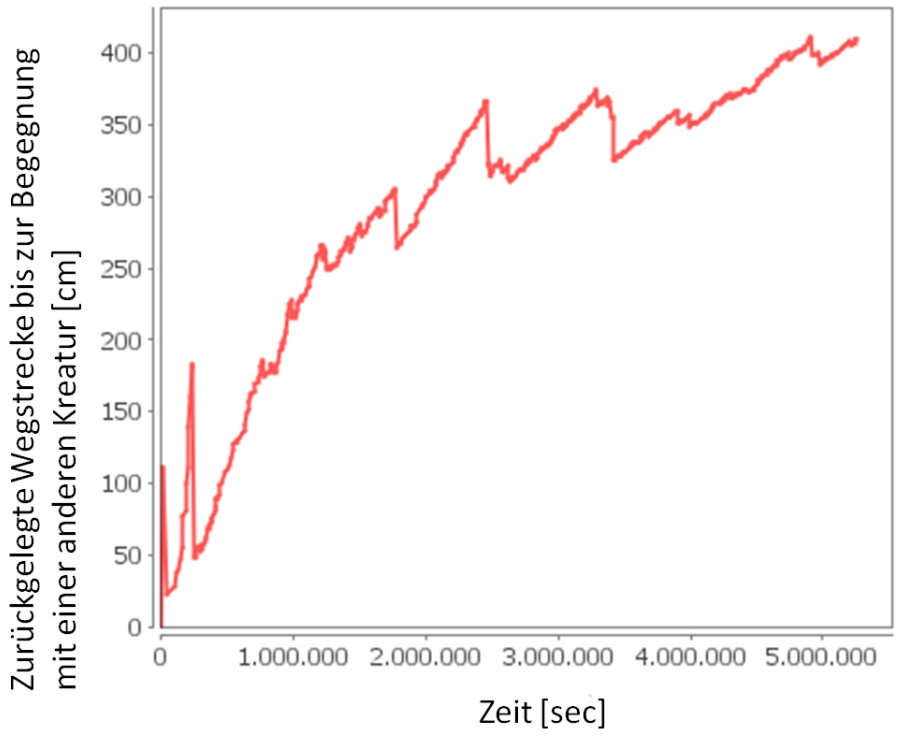

Abbildung 4.16: Durchschnittlich zurückgelegte Strecke innerhalb des Baus nach der die simulierte Ratte auf eine andere Ratte getroffen ist. Simulation mit 10 Agenten unter Verwendung der Futtereintragestrategie 2 und normalverteilter Futtergröße von N(220/60). Simulationszeitraum von 2 Monaten.

Um zusätzlich zu sehen, in wie weit Tiere sich innerhalb einer Simulation gegenseitig innerhalb des Baus behindern, sind weitere Datenerhebungen möglich.

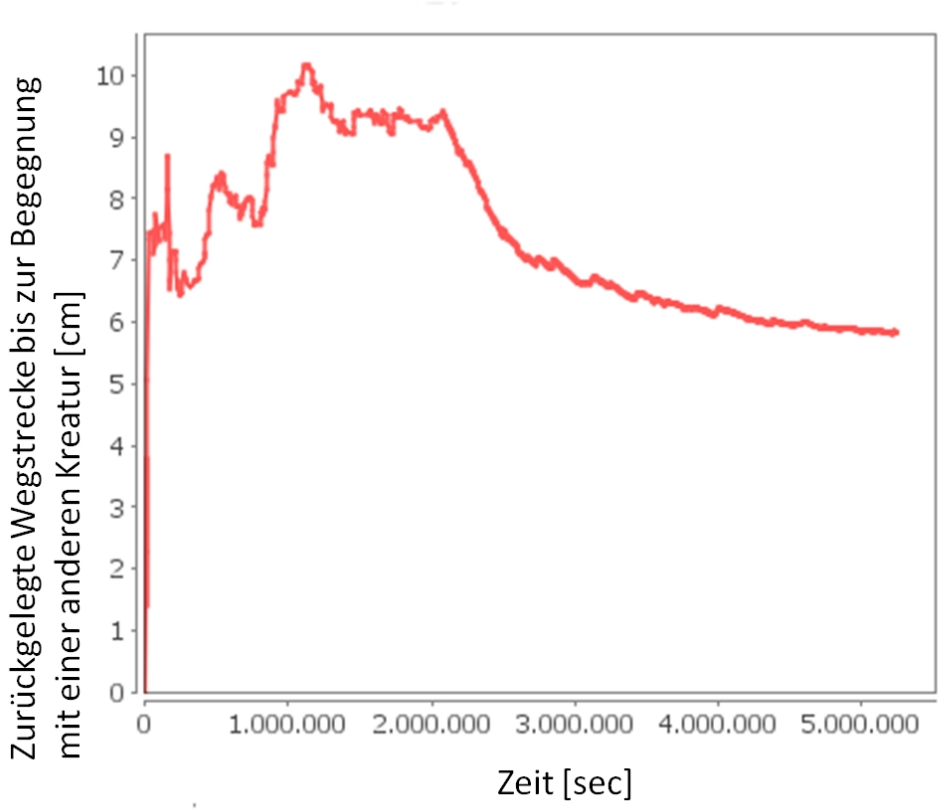

Abbildung 4.17: Durchschnittlich zurückgelegte Strecke innerhalb des Baus nach der die simulierte Ratte auf eine andere Ratte getroffen ist. Simulation mit 45 Agenten unter Verwendung der Futtereintragestrategie 2 und normalverteilter Futtergröße von N(220/60). Simulationszeitraum von 2 Monaten.

Zum einen wurde das Verhältnis von tatsächlich zurückgelegtem Weg und kürzest möglichem Weg betrachtet (vgl. 3.6.2). Es wurde der Mittelwert aus jeweils 10 zweimonatigen Simulationen gebildet. Die festgestellten Simulationsdaten lassen eine exponentielle Entwicklung vermuten (vgl. Abbildung 4.19). Die unterschiedlichen Bedingungen scheinen ein Wachstum der Verhältnisse zu beeinflussen. Das stärkste exponentielle Wachstum scheint die Futtereintragestrategie 1 zu zeigen mit einer geringen Futtergröße. Bei einer größeren Futtergröße wird bereits ein geringeres Wachstum erreicht. Ein nur geringfügig geringeres Wachstum weist die Futtereintragestrategie 2 bei einem geringen Futterangebot auf und die Futtereintragestrategie 2 mit höherer Futtergröße zeigt das geringste Wachstum bei steigernder Agentenanzahl innerhalb der Simulationen. Wobei auch zu beachten ist, dass die unterschiedlich wachsenden Funktionen erst bei einer sehr hohen Rattenanzahl drastische Auswirkungen zeigen. Bereits Simulationen mit einem geringen Anstieg des

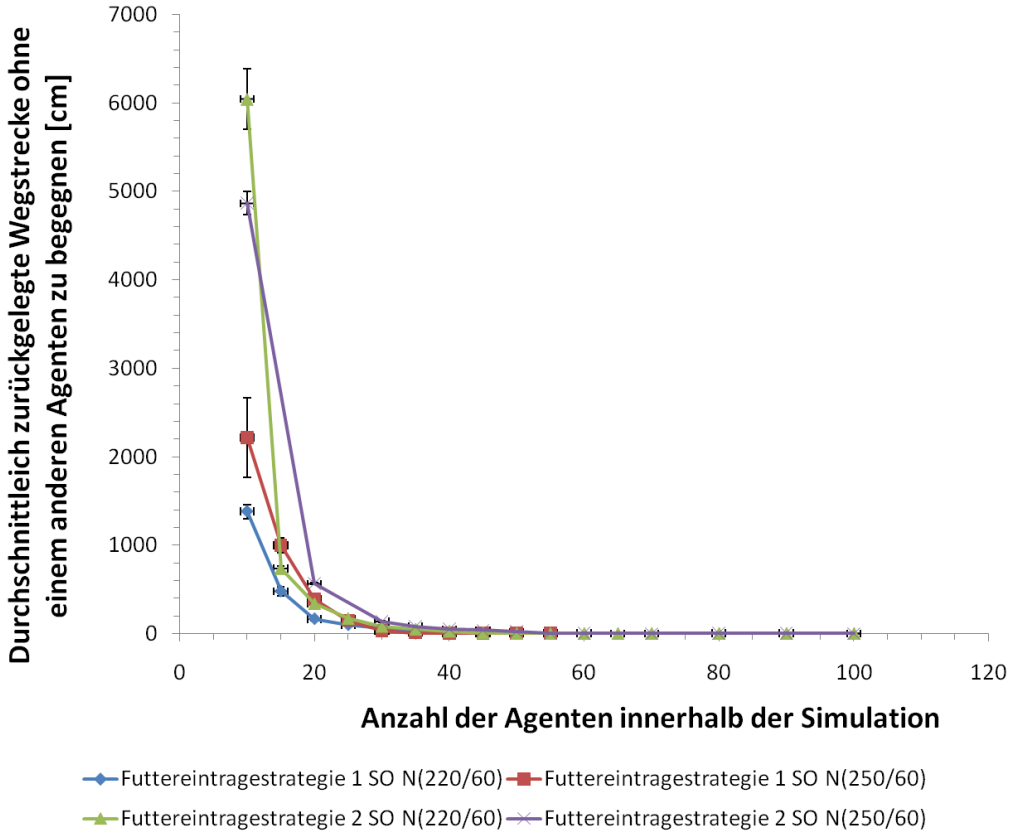

Abbildung 4.18: Übersicht der durchschnittlich zurückgelegten Strecken innerhalb des Baus, nach der die simulierten Ratten auf eine andere Ratte getroffen sind.

Verhältnisses führten allerdings bereits zum Tod eines großen Anteils der Ratten (siehe auch 4.2.6) in dem räumlich begrenzten Bau.

Außerdem sollte die Zeit betrachtet werden, die ein Agent benötigt, um an ein Ziel innerhalb des Baus zu kommen. Es sollte allerdings auch der Weg berücksichtigt werden, den die Tiere auf jeden Fall zurücklegen müssen, um an ihr Ziel zu gelangen. Da alle Tiere sich prinzipiell mit identischer Geschwindigkeit innerhalb der Simulation bewegen, wurde entschieden eine neue theoretische Geschwindigkeit zu definieren, die angibt welche durchschnittliche Geschwindigkeit ein Agent aufweisen würde, wenn er immer den idealen Weg nutzen würde. Es wurde das Fenster ViewVelocityOfIdealPath (vgl. 3.6.2) erstellt.

Es wurden Mittelwerte aus jeweils 10 zweimonatigen Simulationen gebildet und daraus eine theoretische Geschwindigkeit gebildet. Es zeigt sich, dass mit ansteigender Agentenanzahl die theoretische Geschwindigkeit eines ideal zurückgelegten Weges sinkt (siehe Abbildung 4.20). Bis zu eine Populationsgröße von 30 Agenten zeigen sich keine Unterschiede zwischen den verschiedenen Bedingungen. Ab

einer Populationsgröße von 40 Agenten erscheint die Futtereintragestrategie 1 bei einer Futtergröße von N(220/60) den stärksten Geschwindigkeitsverlust zu haben. Die Futtereintragestrategie 1 mit einer Futtergröße von N(250/60) und die Futtereintragestrategie 2 mit einer Futtergröße von N(220/60) liefern vergleichbare Geschwindigkeitsverluste und die Futtereintragestrategie 2 mit einer Futtergröße von N(250/60) scheint die höchste Geschwindigkeit bei einer größeren Agentenanzahl aufzuweisen.

Es war außerdem von Interesse, ob es im Bau Regionen gibt, in denen besonders häufig gegenseitige Blockaden stattfinden. Die Agenten weichen anderen Ratten immer an einem Knoten aus. Nun sollte festgestellt werden, wie oft an jedem Knoten innerhalb einer zweimonatigen Simulation einer anderen Ratte ausgewichen wird. Es wurden Simulationen mit den zwei Futtereintragestrategien und den unterschiedlichen Futtergrößenverteilungen von N(220/60) und N(250/60) mit jeweils 10 und 30 simulierten Ratten durchgeführt.

Bei beiden Futtereintragestrategien zeigt sich, dass verschiedene Punkte im Bau deutlich mehr Blockaden auslösen, als andere Regionen. Besonders häufige Blockaden zeigen Abschnitte mit vielen Ausgängen und einer Kammer, die in einer Sackgasse endet (Abbildung 4.21, 4.22, 4.23 und 4.24). Es handelt sich vermehrt um Bereiche mit einer hohen Baubelastung, wie bereits in Abschnitt 4.2.4 vorgestellt. Die simulierten Ratten nutzten den Bau in diesen Bereichen besonders häufig, da sich in ihnen viele Ausgänge befinden, durch die die simulierten Ratten den Bau betreten und verlassen können und der kürzeste Weg von mehreren Ausgängen zu einer Kammer, alle zur selben Kammer führen. Für diese Theorie spricht auch die stärkere Blockadenanzahl, die sich bei mehr Ratten scheinbar von diesen Kammern aus ausbreitet. Dies lässt sich erklären, da die Agenten dann alle die nächste Kammer aufsuchen, die aber dann oftmals auch wiederum von mehreren Agenten aufgesucht wird. Somit kann eine solche Konstellation weitgreifend auf größere Areale des Baus Auswirkungen zeigen, hier erkennbar an einer höheren Anzahl an Blockaden um bestimme Kammern herum.

Bei der Futtereintragestrategie 1 fällt auf, dass die Futtergröße scheinbar einen starken Einfluss auf die Baunutzung und die hervorgerufenen Blockaden hat. Hierbei zeigt sich, dass sich die Agenten bei einer geringen Futtergröße deutlich stärker blockieren (Abbildung 4.21 oben) als bei einer größeren Futtergröße (Abbildung 4.21 unten).

Bei einer höheren Rattenanzahl innerhalb der Simulation (Abbildung 4.22) steigt die Anzahl der Blockaden an und auch selten benutzte Knoten, die bei 10 Agenten noch keine Blockaden aufzeigten, zeigen nun z.T. bei der Futtereintragestrategie 1 über hundert Blockaden innerhalb der zweimonatigen Simulation an. Die Blockaden der reich frequentierten Bereiche zeigen bis über 1 000 000 Blockaden an. Die Bereiche mit sehr vielen Blockaden verändern sich allerdings nicht. Dies deutet darauf hin, dass sich die simulierten Ratten bei 30 Agenten trotz der Ausweichbewegungen und der daraus resultierenden ausgedehnteren Baunutzung,

sehr viel stärker gegenseitig blockieren. Allerdings kann man erkennen, dass eine größerer Futtergröße von N(250/60) weniger Blockaden hervorruft. Dies ist unter anderem durch eine größere Anzahl an zufälligen Bewegungen innerhalb des Baus zu erklären.

Bei der Futtereintragestrategie 2 zeigt sich kein erkennbarer Unterschied zwischen den verschiedenen Futtergrößenverteilungen, weder bei 10 Ratten (Abbildung 4.23), noch bei 30 Ratten (Abbildung 4.24).

Ein Vergleich der beiden Futtereintragestrategien zeigt einen deutlichen Unterschied zwischen den auftretenden Blockaden. Während die Futtereintragestrategie 1 sehr viele Blockaden in allen Baubereichen aufzeigt, sind die auftretenden Blockaden bei der Futtereintragestrategie 2 sehr viel geringer. Man erkennt bei der Futtereintragestrategie 2 ebenfalls einen Trend zu mehr Blockaden bei einer größeren Futtergröße. Dies erklärt sich durch die höhere Zahl an Agenten innerhalb des Baus, die nun vermehrt innerhalb des Baus nach Futter suchen und somit tendenziell mehr Blockaden an Vorratskammern hervorrufen. Außerdem verlassen diese Tiere den Bau häufiger und müssen daher vermehrt Ausgänge aufsuchen. Auch wählen diese Tiere bei einem geringen Hunger und Müdigkeitslevel häufiger die Aktion Random-Walk. Sie führt zu zufälligen Bewegungen im Bau.

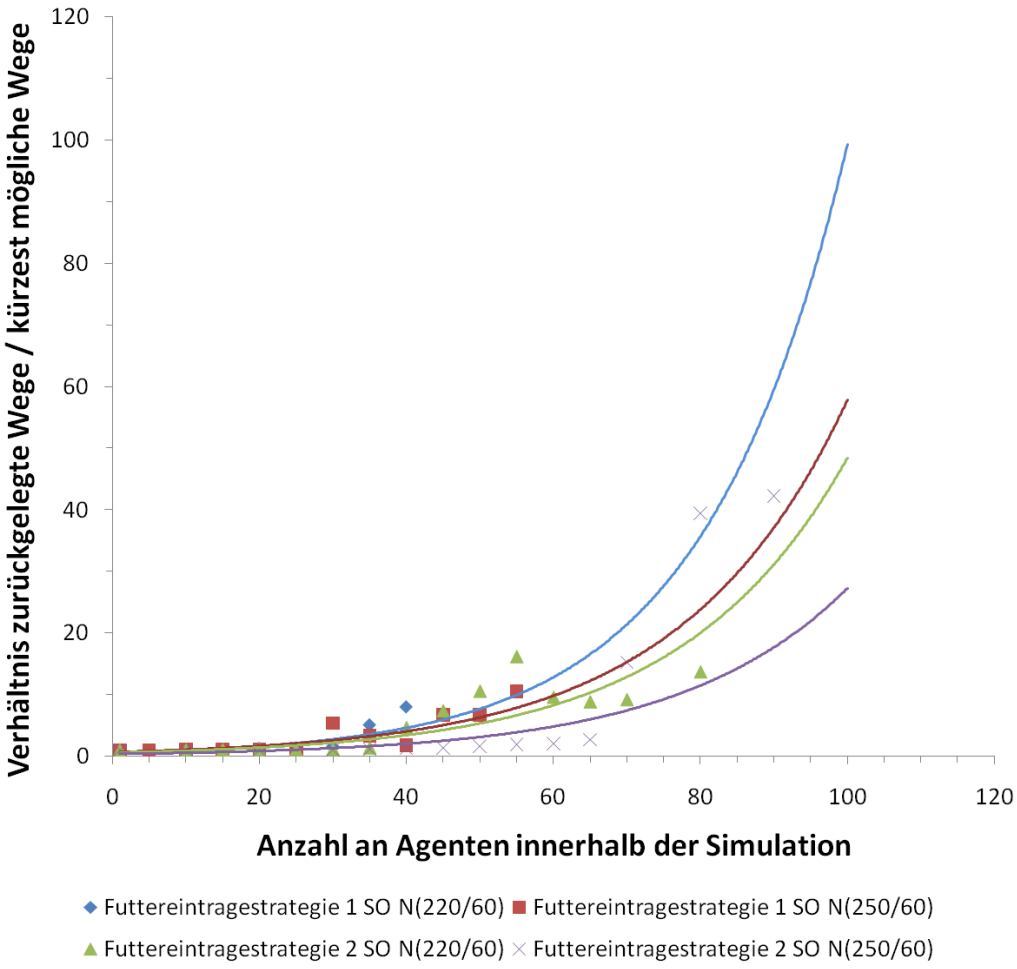

Abbildung 4.19: Verhältnis der zurückgelegten Wegstrecken zu den kürzest möglichen Wegstrecken innerhalb des Baus bei einer zweimonatigen Simulation. Es wurden die Quotienten gebildet über die zurückgelegten Distanzen, die die Agenten vom Startpunkt der zielgerichteten Bewegung bis zum Erreichen des Ziels zurücklegen mussten. Dann wurde die Summe über alle Quotienten gebildet und diese durch die Anzahl der Quotienten geteilt. Anschließend wurde der Mittelwert über alle Simulationen gebildet und aufgetragen. Die aufgezeigten Funktionen zeigen den Verlauf bei einem angenommenen exponentiellen Wachstum der Verhältnisse.

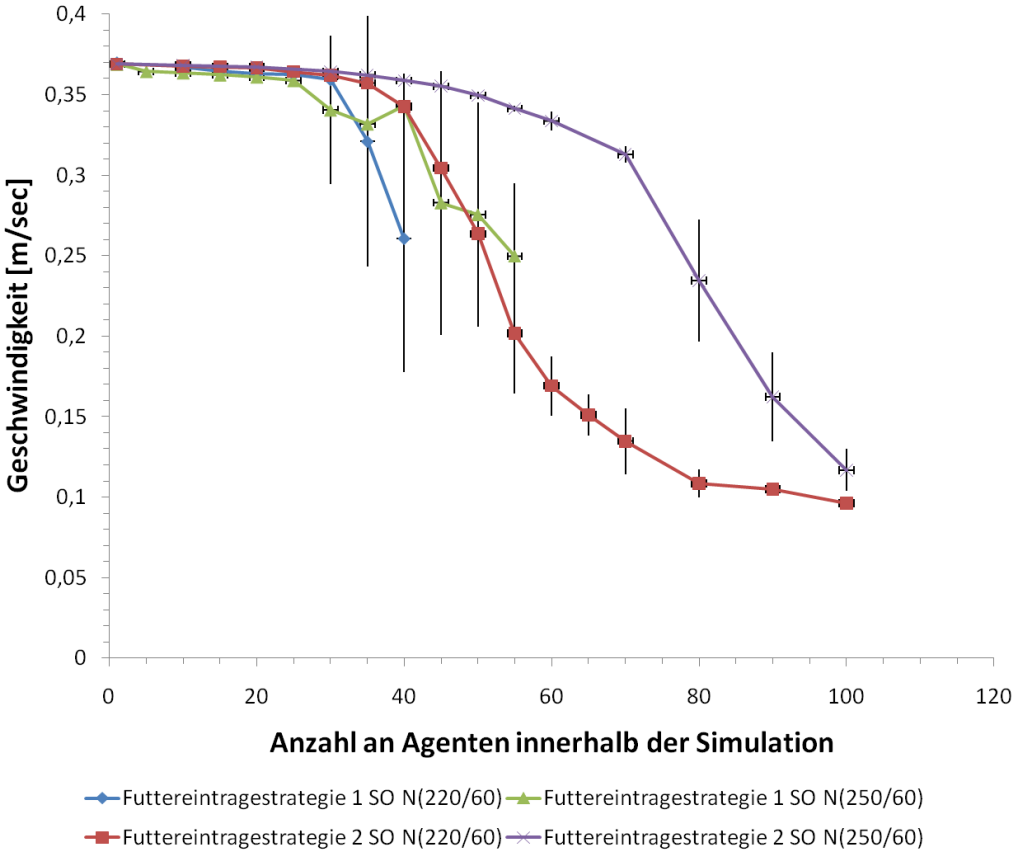

Abbildung 4.20: Übersicht der durchschnittlichen theoretischen Geschwindigkeit innerhalb des Baus, die sich durch Ausweichbewegungen ergibt. Es wird die theoretische Geschwindigkeit bestimmt, die ein Agent aufweisen würde, wenn er immer den perfekten Weg zwischen zwei Punkten im Bau nutzen würde. Tatsächlich bewegen sich die Agenten innerhalb der Simulation mit identischer Geschwindigkeit. Diese würde bei dieser Messung 0,369 m/sec entsprechen.

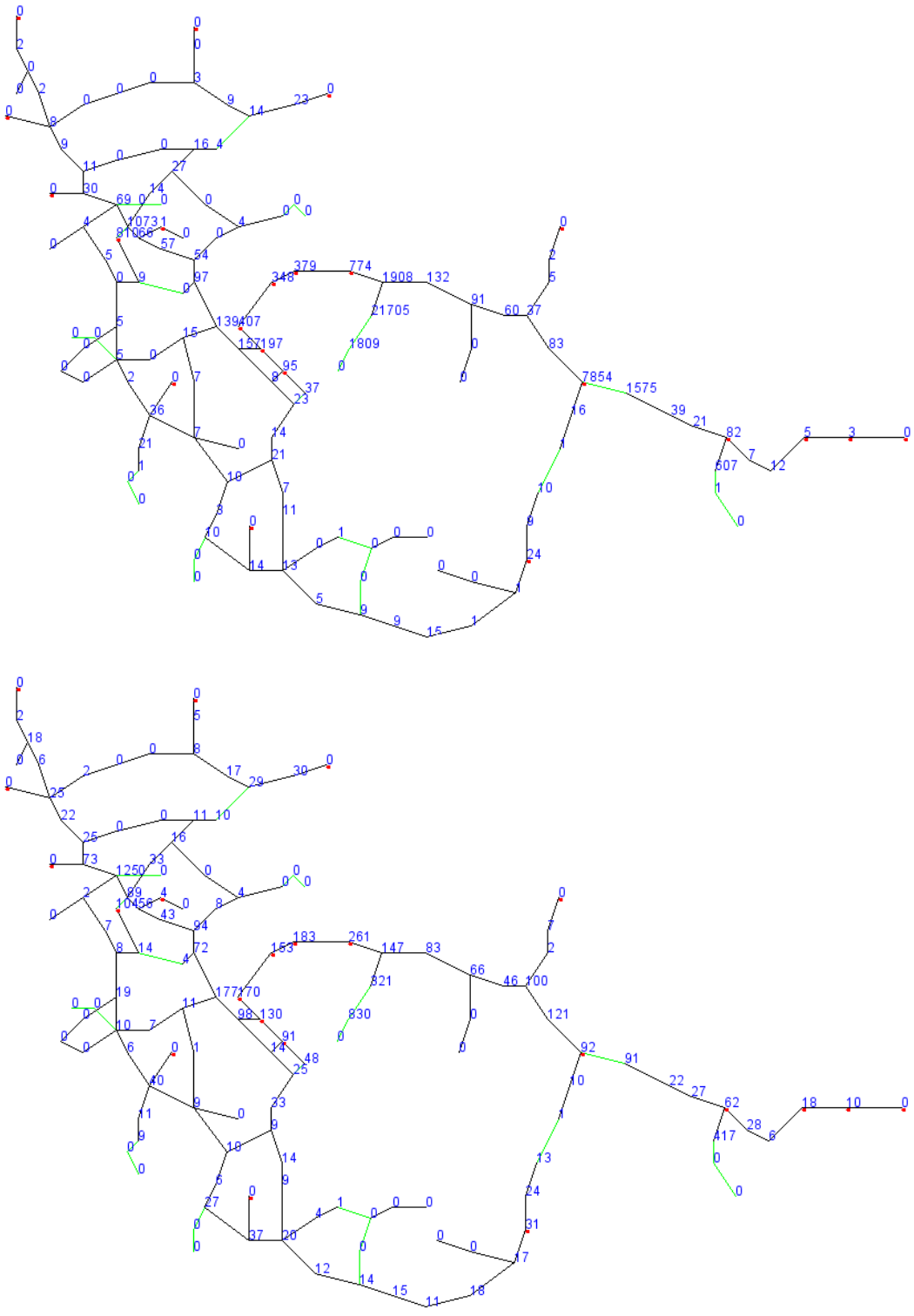

Abbildung 4.21: Anzahl der Ausweichbewegungen bei jedem Knoten innerhalb des Baus bei der Futtereintragestrategie 1 und unterschiedlicher normalverteilten Futtergröße bei 10 simulierten Ratten und einer Simulationszeit von zwei Monaten. Der obere Bau zeigt das Futtereintrageverhalten bei einer Futtergrößenverteilung von N(220/60) und der untere Bau bei einer Futtergrößenverteilung von N(250/60).

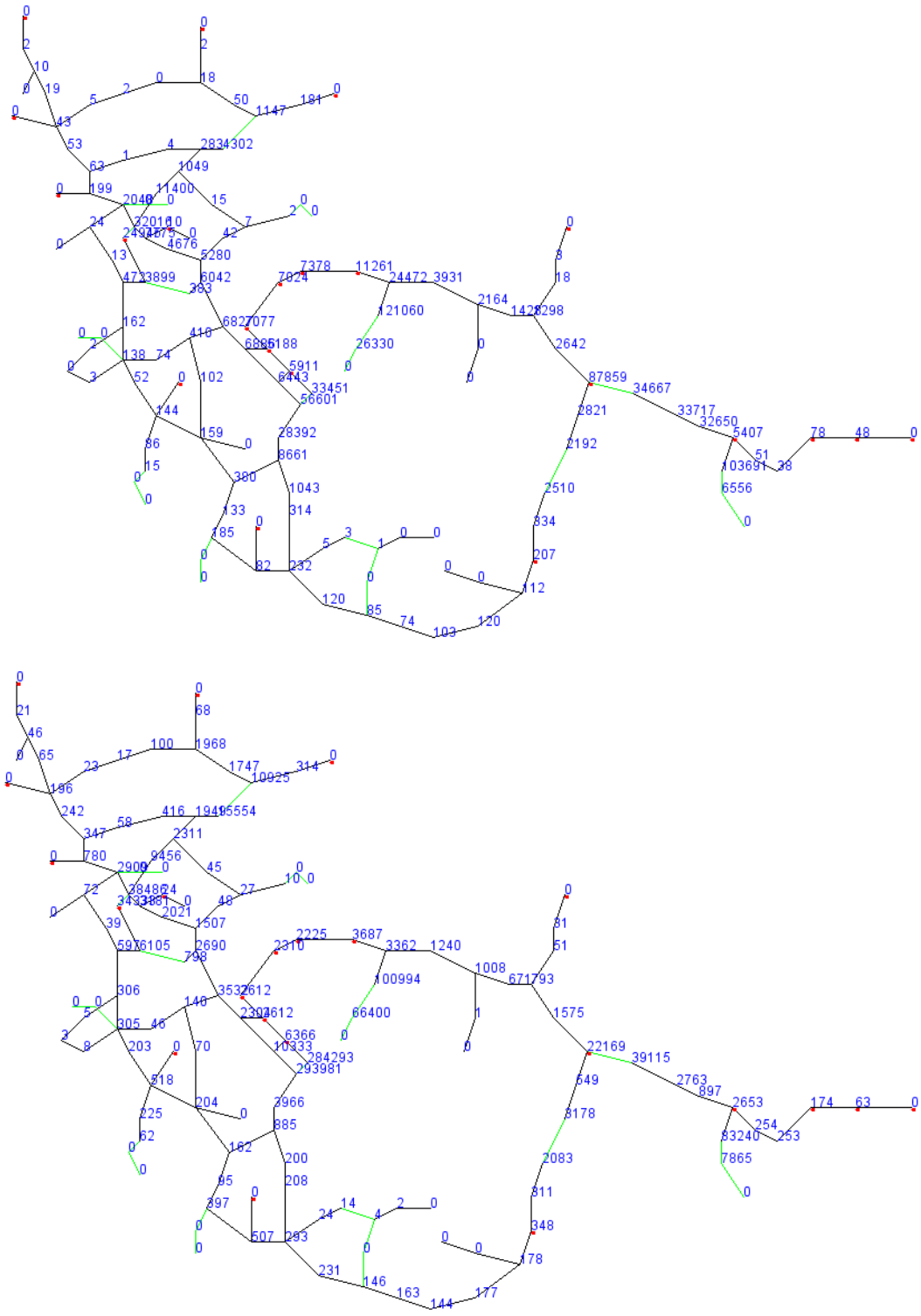

Abbildung 4.22: Anzahl der Ausweichbewegungen bei jedem Knoten innerhalb des Baus bei der Futtereintragestrategie 1 und unterschiedlicher normalverteilten Futtergröße bei 30 simulierten Ratten und einer Simulationszeit von zwei Monaten. Der obere Bau zeigt das Futtereintrageverhalten bei einer Futtergrößenverteilung von N(220/60) und der untere Bau bei einer Futtergrößenverteilung von N(250/60).

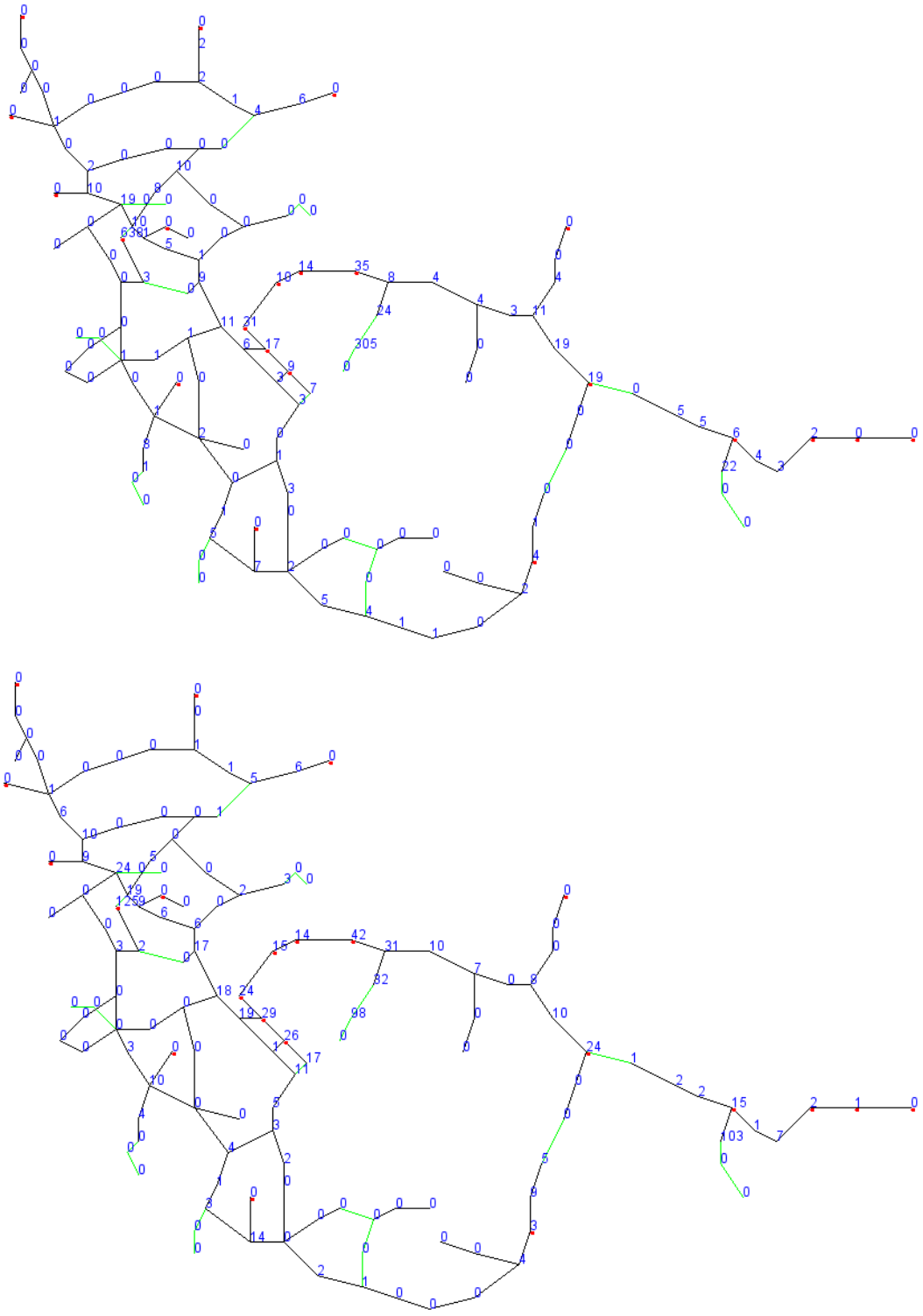

Abbildung 4.23: Anzahl der Ausweichbewegungen bei jedem Knoten innerhalb des Baus bei der Futtereintragestrategie 2 und unterschiedlicher normalverteilten Futtergröße bei 10 simulierten Ratten und einer Simulationszeit von zwei Monaten. Der obere Bau zeigt das Futtereintrageverhalten bei einer Futtergrößenverteilung von N(220/60) und der untere Bau bei einer Futtergrößenverteilung von N(250/60).

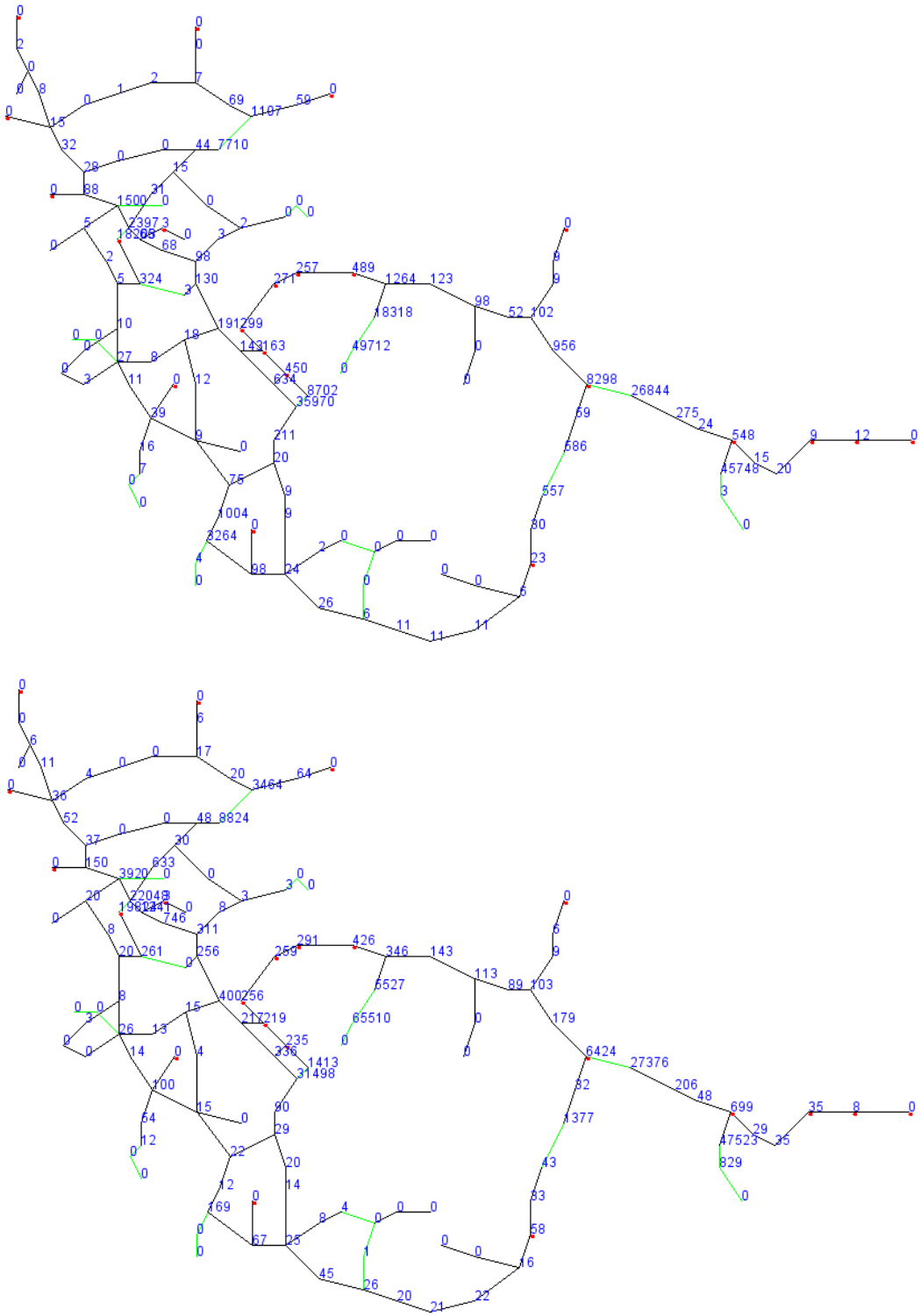

Abbildung 4.24: Anzahl der Ausweichbewegungen bei jedem Knoten innerhalb des Baus bei der Futtereintragestrategie 2 und unterschiedlicher normalverteilten Futtergröße bei 30 simulierten Ratten und einer Simulationszeit von zwei Monaten. Der obere Bau zeigt das Futtereintrageverhalten bei einer Futtergrößenverteilung von N(220/60) und der untere Bau bei einer Futtergrößenverteilung von N(250/60).

4.2.6 Sterberate

Zum Abschluss der Betrachtung von zweimonatigen Simulationen soll nun ein Blick auf die Sterberate geworfen werden. Es wird untersucht, wie viel Prozent der Tiere innerhalb der zweimonatigen Simulation verstorben sind. Dies kann nun abschließend als Hinweis dienen, wie viele Tiere tatsächlich innerhalb der simulierten Welt und dem darin befindlichen Bau unter den unterschiedlichen Bedingungen überleben können. Man kann zwei verschiedene Ansätze wählen: der Strebeprozess kann entweder nur vermerkt werden, die Ratte wird allerdings nicht aus der Simulation genommen oder die Tiere werden aus der Simulation genommen. Dies bedeutet die Anzahl der Agenten ist während der Simulation konstant oder veränderlich.

Abbildung 4.25 zeigt die Sterberate bei einer nicht abnehmenden Populationsgröße, das heißt wie viel Tiere bei einem bestimmten Baubesatz sterben würden. Man sieht, dass die verschiedenen Futtereintragestrategien und Futtergrößenwahrscheinlichkeiten einen unterschiedlichen Einfluss auf das Überleben der Rattenpopulation haben.

Bei der Futtereintragestrategie 1 können 25 bis 30 Agenten innerhalb des Baus leben. Die Todesrate führt bei einer größeren Populationsdichte schnell zum Tod der gesamten Population. Ein größeres Futterangebot führt zu einer geringeren Sterberate bei einer größeren Population, allerdings kann auch hier nicht eine größere Gesamtpopulation überleben, sondern zeigt sich hier ebenfalls eine sinnvolle Populationsgröße von rund 25 Agenten.

Bei der Futtereintragestrategie 2 können bei einem gewissen Überschuss an Futter bis zu 45 Ratten innerhalb des Baus überleben. Auch bei einem geringen Futterangebot können bis zu 35 Ratten innerhalb des Baus überleben. Erst bei einer höheren Populationsdichte kommt es zu gegenseitigen Blockaden, so dass ein immer größerer prozentualer Anteil an Tieren stirbt.

Werden nun die sterbenden Agenten während der Simulation entfernt, kann sich bei ausreichend Futter eine steigende Sterberate einspielen, die allerdings nicht unbedingt die gesamte Population umfasst (Abbildung 4.26). Dies ist besonders gut bei der Futtereintragestrategie 2 bei einer Futtergrößenverteilung von N(250/60) zu erkennen. Hier bildet sich offensichtlich eine konstante Rattenanzahl aus, die wahrscheinlich aus bis zu 45 Tieren bestehen kann. Hingegen führt eine zu geringe Futtergröße von N(220/60) nicht zur Einstellung dieses Verhältnisses. Stattdessen wurde hier beobachtet, dass bei mehr als 50 Tieren die gesamte Population zu Grunde geht. Eventuell kann hier über einen längeren Zeitraum nicht genügend Futter eingetragen werden. Selbst nach einer Reduktion der Agentenanzahl in den entsprechenden Simulationen, konnte sich auch kein kleiner Anteil an der ursprünglichen Population halten. Dies bedeutet, dass das Zusammenspiel von zu geringem Futter und zu vielen Agenten in unserer Simulation zum Tod der gesamten Population führt. Bei der Futtereintragestrategie 1 konnten diese Trends nicht bestätigt werden. In den vorliegenden Simulationen stieg die Sterberate konstant an. Allerdings wurden keine Simulationen mit mehr als 55 Ratten durchgeführt. Daher kann

nicht abschließend festgestellt werden, ob sich die Sterberate auf einen bestimmten prozentualen Anteil einspielen würde oder ob, bei einer höheren Agentenanzahl, die gesamte Population ausgelöscht werden würde. Allerdings zeigten anschließende Simulationen mit einer Vermehrungsrate, dass sich bei beiden Eintragestrategien bei ausreichendem Futterangebot eine konstante Populationsgröße ausbilden kann (vgl. 4.3)

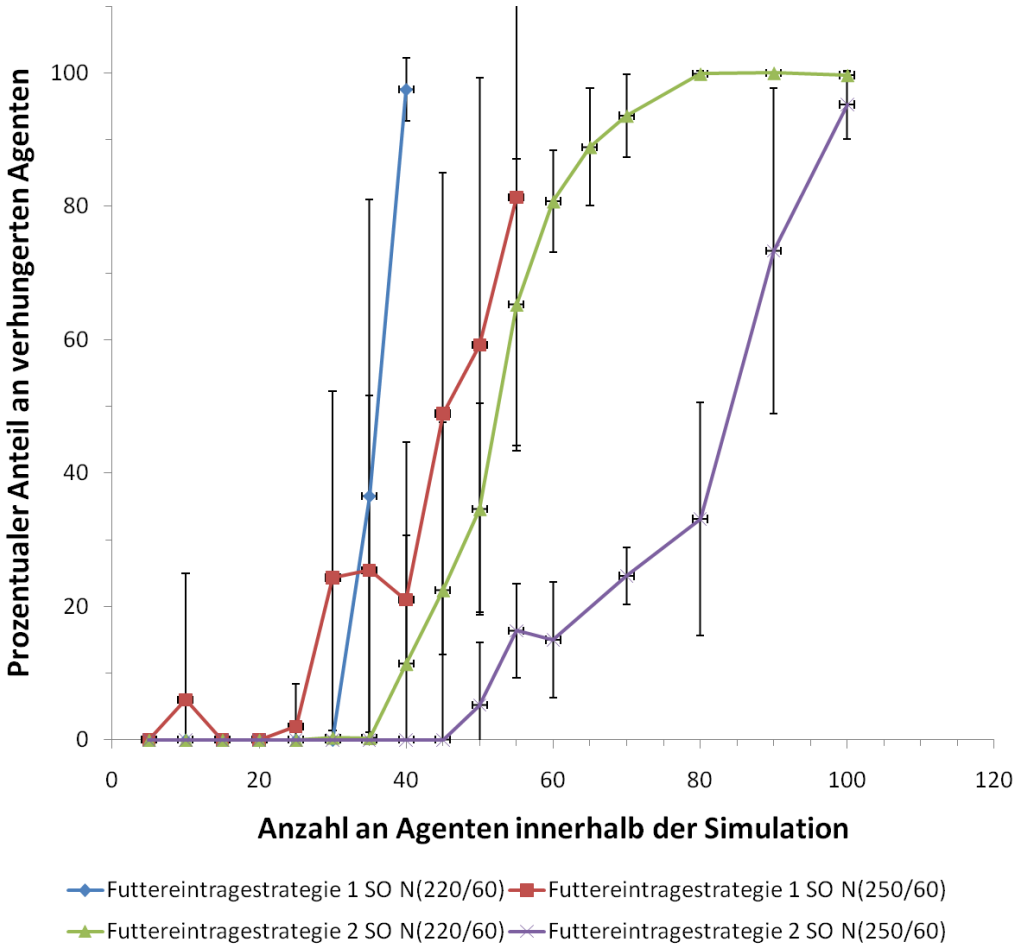

Abbildung 4.25: Prozentualer Anteil der Agenten, die innerhalb einer zweimonatigen Simulation versterben. Die Agenten wurden nach ihrem Tod nicht aus der Simulation entfernt, sondern der Tod wurde mittels eines boolean festgehalten. Die Agenten versuchten allerdings weiterhin ihre Bedürfnisse zu stillen und verhielten sich wie zuvor. Somit gibt dies eine Übersicht über den Anteil an Agenten, der innerhalb eines zweimonatigen Zeitraums, bei einer jeweiligen unveränderlichen Agentenanzahl innerhalb des Baus, stirbt.

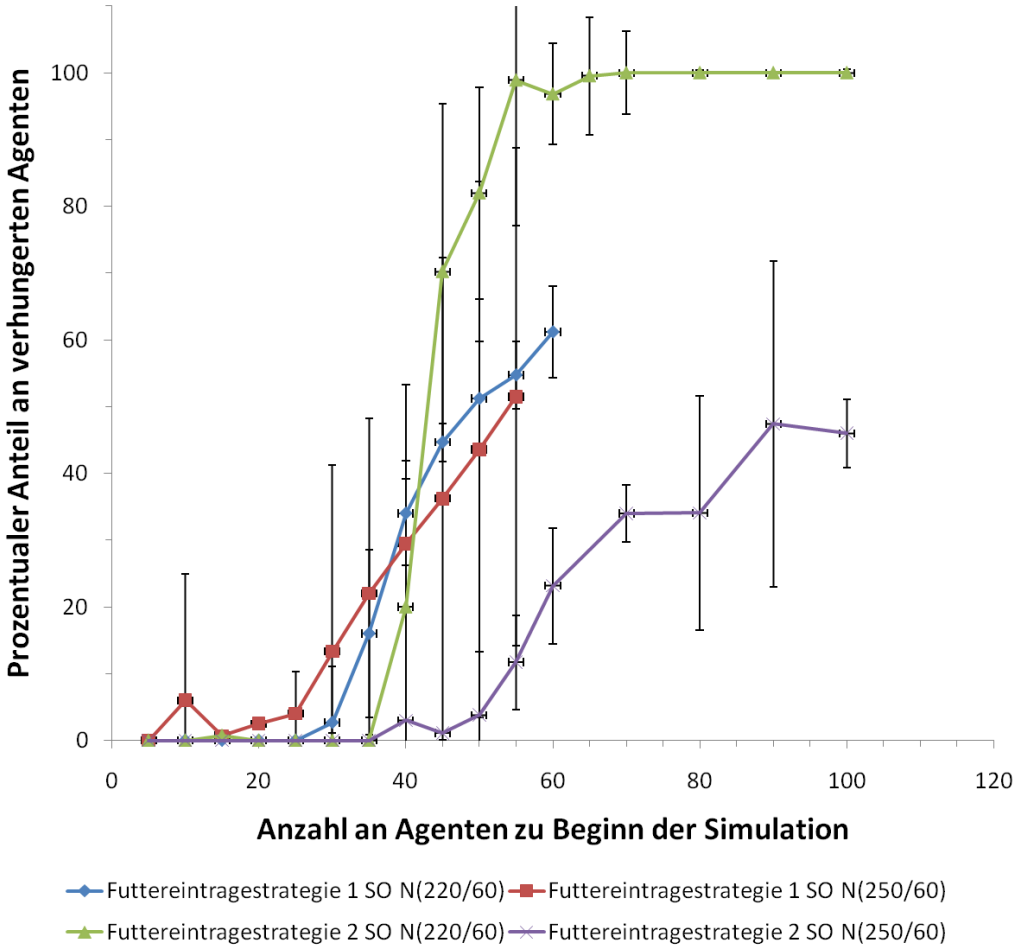

Abbildung 4.26: Prozentualer Anteil der Agenten, die innerhalb einer zweimonatigen Simulation versterben. Die Agenten wurden nach ihrem Tod aus der Simulation entfernt. Somit wird auf der x-Achse nur die Anzahl der Agenten zu Beginn der Simulation angegeben.

4.3 Simulationen mit Vermehrung

Als nächster Schritt wurde nun eine Vermehrung, wie bereits in Abschnitt 3.3.2 vorgestellt, eingeführt. Hierbei wurden Simulationen zwischen 6 Monaten und einem Jahr durchgeführt. Die Simulationsbedingungen wurden nicht verändert. Entsprechend wurde wiederum eine Aufenthaltsdauer von N(300,60) eingestellt (3.3.6) bis der Agent den Bau wieder betritt (Futtereintragestrategie 1) oder die simulierte Ratte neu entscheidet, ob sie weiterhin außerhalb des Baus bleibt oder den Bau erneut betritt (Futtereintragestrategie 2). Zudem wurde wiederum eine Futterfundwahrscheinlichkeit von $y = (0{,}6/360) * i$ (vgl. Formel 3.1), wobei i die Aufenthaltsdauer außerhalb des Baus in Sekunden angibt, angewendet. Allein die Futtergrößenwahrscheinlichkeit musste neu festgelegt werden, da die Tiere nun ein unterschiedliches Futterbedürfnis aufwiesen, da weibliche trächtige Ratten ein Futterbedürfnis von 35 g am Tag aufwiesen. Dies führt zu einem größeren Futterbedürfnis der gesamten Population. Die Futtergrößen mussten somit angepasst werden, um das gestiegene Futterbedürfnis der Weibchen zu stillen. Die Simulationen wurden mit 10 Agenten gestartet. Die Vermehrung setzte erst nach 36 Tage ein, dies bedeutet die Tiere hatten ausreichend Zeit einen Futtervorrat innerhalb des Baus aufzubauen, bevor die weiblichen Ratten trächtig wurden.

Während Tiere mit der Futtereintragestrategie 1 mindestens eine Futtergröße von N(300/60) benötigten, reichte bei der Futtereintragestrategie 2 bereits eine Futtergröße von N(260/60). War hingegen das Futterangebot in einer Vermehrungsphase zu gering eingestellt, starben alle weiblichen Tiere einer Population innerhalb ihrer ersten Trächtigkeit. Die männlichen Tiere überlebten, aber ohne weibliche Tiere ist ein Fortbestand der Population nicht sinnvoll möglich.

Außerdem führt das größere Futterbedürfnis der Ratten während der Trächtigkeit zu mehr Eat-Aktionen. Verbrachten die Tiere zuvor 8% ihrer Zeit mit Fressen, verbringen sie nun durchschnittlich 9-10% der Zeit mit Fressen je nach geschlechtlicher Populationszusammensetzung. Dies führte zu einer Veränderung in den Zeitverhältnissen, die die Tiere zur Futtersuche und zum Schlafen aufwendeten.

Bei der Futtereintragestrategie 1 verbrachten die Tiere zuvor rund 42% ihrer Zeit mit Schlafen, rund 46% mit Futtersuche außerhalb des Baus und 4% mit Bewegungen innerhalb des Baus. Nun verbringen sie nur noch 43% mit Futtersuche außerhalb des Baus und ebenfalls 43% der Zeit mit Schlafen.

Bei der Futtereintragestrategie 2 verbrachten die Tiere zuvor rund 42% ihrer Zeit mit Schlafen und rund 48% mit Futtersuche außerhalb des Baus. Nun verbringen sie nur noch 46% ihrer Zeit mit der Futtersuche außerhalb des Baus und 43% der Zeit mit Schlafen. Außerdem sind die Ratten weniger außerhalb des Baus. Statt zuvor 54% der Zeit sind sie nun 52% außerhalb.

Die simulierte Vermehrung zeigte eine mittlere Geburtenrate von 138,64 Geburten mit einer Standardabweichung von 34,23 innerhalb der simulierten sechs Monate. Dies bedeutet eine enorme Vermehrung innerhalb von sechs Monaten. Allerdings

verhinderte die Baugröße das Überleben aller Ratten. Bei der Futtereintragestrategie 1 konnten zwischen 25 und 40 Tiere innerhalb des Baus leben (siehe Abbildung 4.28), bei der Futtereintragestrategie 2 hingegen zwischen 40 und 60 Tiere (siehe Abbildung 4.29).

Dies bedeutet, dass die Futtereintragestrategie 2 im vorliegenden Modell eine höhere Populationsgröße innerhalb des Baus zulässt und daher als vorteilhaftere Strategie erscheint, da mehr Agenten innerhalb des Baus überleben können. Betrachtet man die beiden Todesraten vergleichend, starben bei der Futtereintragestrategie 1 durchschnittlich 100 Tiere innerhalb der Simulation mit einer Standardabweichung von 23, hingegen starben bei der Futtereintragestrategie 2 rund 80 Tiere innerhalb der Simulation mit einer Standartabweichung von 44. Auch hier zeigt sich, dass die Futtereintragestrategie 2, bei keinem Prädatorendruck, die sinnvollere Strategie für die simulierten Ratten darstellt.

Vergleicht man nun zusätzlich die Werte des Futtervorrats innerhalb des Baus, die Wegstrecke der Tiere bis sie einem anderen Agenten begegneten und das Verhältnis vom tatsächlich zurückgelegtem Weg der Ratte zum kürzesten möglichen Weg zum Ziel innerhalb des Baus (vgl. Tabelle 4.27) zeigt sich nur bei der eingetragenen Futtermenge ein eindeutiger Vorteil der Futtereintragestrategie 2, da hier rund doppelt so viel Futter eingetragen wurde als bei der Futtereintragestrategie 1.

	Futtereintragestrategie 1		Futtereintragestrategie 2	
	Mittelwert	Standartabweichung	Mittelwert	Standartabweichung
Futtermenge innerhalb des Baus [mg]	10731274	3133218,86	21900000	4614751
Zurückgelegte Strecke bis einer anderen Ratte begegnet wird [cm]	16,91	3,89	20,08	14,71
Konstruierte Geschwindigkeit [m/sec]	0,34946	0,00354	0,33923	0,01182
Verhältnis kürzest mögliche Wegstrecke / zurückgelegte Wegstrecke	1,513	0,149	1,503	0,205

Abbildung 4.27: Tabelle über die durchschnittlichen aufgezeichneten Daten von jeweils acht sechsmonatigen Simulation mit Vermehrung und einer Futtergrößenverteilung von N(320/60)

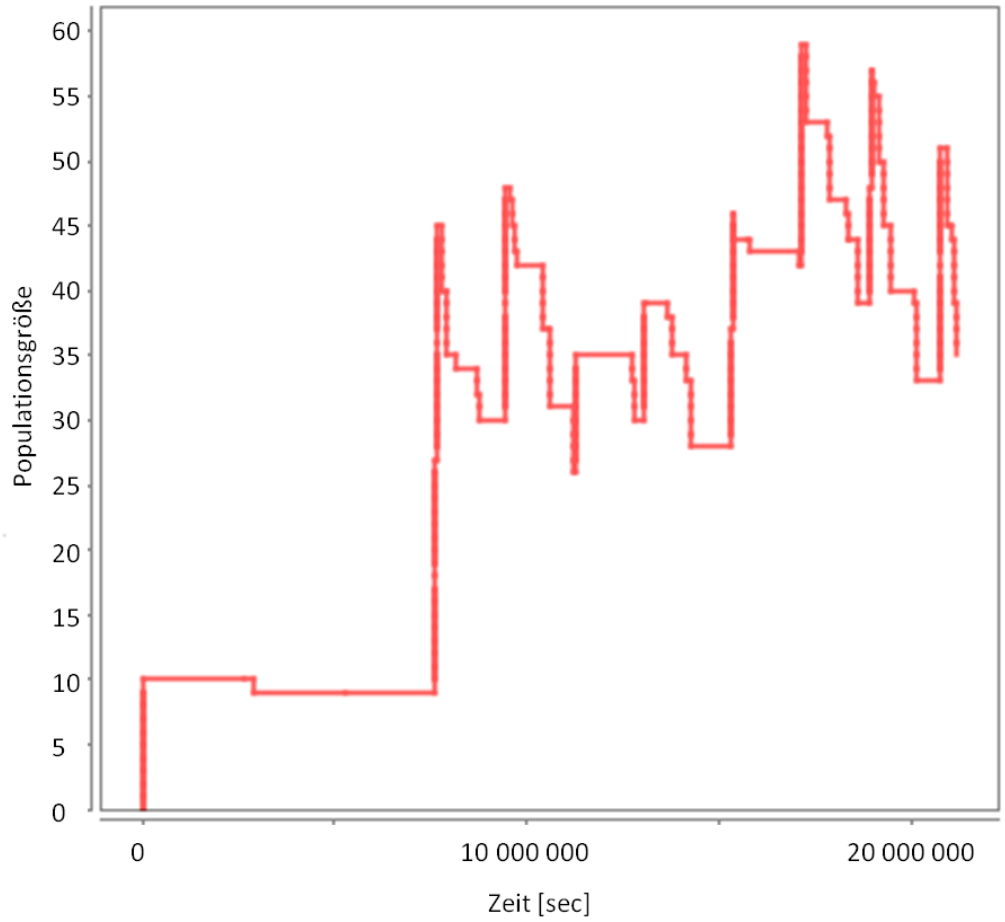

Abbildung 4.28: Populationsentwicklung innerhalb einer Simulation über mehrere Monate mit Vermehrung und keinem Prädator. Die Tiere verwendeten die Futtereintragestrategie 1 bei einer Futtergrößenverteilung von N(320/60)

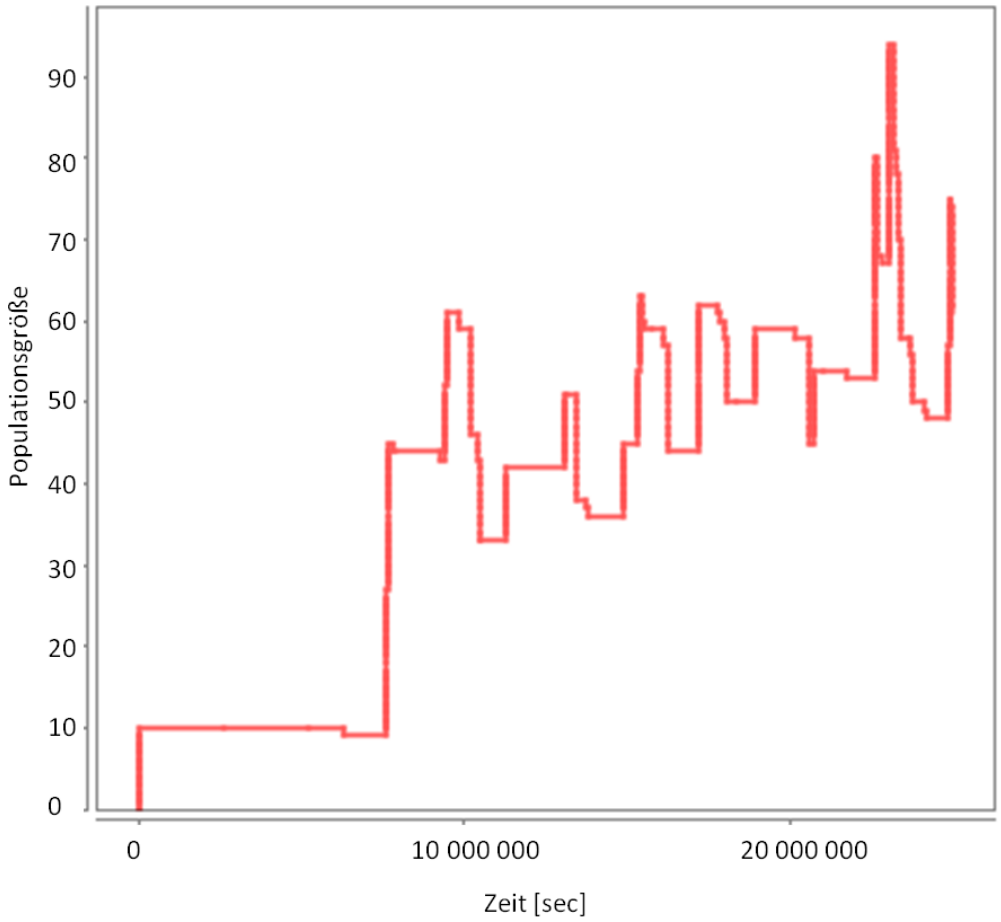

Abbildung 4.29: Populationsentwicklung innerhalb einer Simulation über mehrere Monate mit Vermehrung und keinem Prädator. Die Tiere verwendeten die Futtereintragestrategie 2 bei einer Futtergrößenverteilung von N(320/60)

4.4 Simulation mit Prädatorendruck und Vermehrung

In einer realistischen Umwelt existieren Prädatoren. Die von ihnen ausgehende Gefahr ist außerhalb des Baus eventuell hoch und kann unter Umständen die Auslöschung einer gesamten Population hervorrufen. Daher wurde nun ein Prädatordruck eingeführt. Hierbei hängt die Wahrscheinlichkeit einem Prädator zu begegnen von der Aufenthaltszeit außerhalb des Baus ab. Je länger ein Tier außerhalb des Baus Zeit verbringt, je höher ist die Wahrscheinlichkeit einem Prädator zu begegnen. Aus Zeitgründen konnte hier allerdings nur eine Prädatorenwahrscheinlichkeit getestet werden. Diese wurde unter beiden Futtereintrageverhaltensweisen getestet. Als Simulationszeitraum wurden sechs Monate gewählt, außerdem eine Futtergrößenverteilung von N(320/60). Dies entspricht einer ausreichenden Futtermenge bei beiden Futtereintragestrategien, so dass ein stetig steigender Futtervorrat innerhalb des Baus abgelegt werden kann. Dieser Futtervorrat ist auch bei einer größeren Anzahl an trächtigen Weibchen ausreichend. Die Aufenthaltsdauer außerhalb des Baus und die Futterfundwahrscheinlichkeit wurden nicht verändert und wie bei den anderen Simulationen angenommen. Die Prädatorenwahrscheinlichkeit und damit der Tod der simulierten Ratte stiegen außerhalb des Baus linear an mit der Aufenthaltszeit außerhalb des Baus. Die hierbei untersuchte Prädatorenwahrscheinlichkeit wurde in der Simulation mit

$$y = (0{,}00001/500) * i \tag{4.1}$$

angenommen, wobei i die Zeit außerhalb des Baus in Sekunden angibt. Es zeigte sich, dass bei der Futtereintragestrategie 1 im Mittel 87,5 Tiere, mit einer Standartabweichung von 19,32, starben. Davon verhungerten rund 83,87, mit einer Standardabweichung von 18,96, der Ratten aufgrund der Platzproblematik innerhalb des Baus, da genügend Futter innerhalb des Baus vorhanden war und nur rund 3,63 Ratten, mit einer Standardabweichung von 1,19, auf Grund eines Prädatorenangriffs. Bei der Futtereintragestrategie 2 hingegen starben rund 49 Ratten, mit einer Standardabweichung von 16,8. Hiervon starben 28 Tiere, bei einer Standardabweichung von 15,05, aufgrund der Platzproblematik innerhalb des Baus, da auch hier genügend Futter vorhanden war und rund 21 Ratten, mit einer Standardabweichung von 4,81, aufgrund des simulierten Prädators. Man kann somit feststellen, dass die Futtereintragestrategie 2 hier mehr Opfer aufgrund des Prädatorenangriffs verzeichnen muss.

Allerdings zeigt sich hier im Vergleich der beiden Gesamttodesfälle, dass auch hier bei der Futtereintragestrategie 1 mehr Tiere sterben als bei der Futtereintragestrategie 2. Dies bedeutet, dass die Mehrzahl der Todesfälle der Ratten bei beiden Strategien noch hauptsächlich aufgrund des Baus und dessen Platzverhältnissen auftreten und weniger wegen des simulierten Prädators.

Es zeigt sich, dass bei einem ausreichend vorhandenem Futtervorrat im Bau

immer noch über 95% aufgrund der Platzprobleme innerhalb des Baus bei der Futtereintragestrategie 1 sterben und bei der Futtereintragestrategie rund 50% der Sterbeprozesse aufgrund der stetigen Baugröße sterben. Dies zeigt die Wichtigkeit eines ständig wachsenden Baus. Außerdem zeigt sich ein sehr interessantes Ergebnis im Vergleich der Sterberaten bei keinem vorhandenen Prädator und der hier getesteten Prädatorenwahrscheinlichkeit. In dem hier vorgestellten Modell scheint eine Prädatorenanwesenheit sich positiv auf die Gesamttodesrate auszuwirken. Sterben bei der Futtereintragestrategie 1 bei keinem vorhandenen Prädator insgesamt rund 100 Agenten, sind es bei der simulierten Prädatorenwahrscheinlichkeit nur noch rund 87 Agenten. Noch deutlicher zeigt sich dieser Trend bei der Futtereintragestrategie 2: Bei keinem vorhandenen Prädator sterben insgesamt rund 80 Agenten innerhalb von sechs Monaten, bei der simulierten Prädatorenwahrscheinlichkeit sind es nur noch rund 49 Agenten. Dies bedeutet, dass bei einer gewissen Prädatorenwahrscheinlichkeit weniger Tiere insgesamt Sterben als bei keinem vorhandenen Prädator. Dies genauer zu untersuchen könnte mit der vorhandenen Simulation gut bewerkstelligt werden, konnte aus zeitlichen Gründen innerhalb dieser Diplomarbeit aber leider nicht mehr realisiert werden.

Kapitel 5

Diskussion und Ausblick

Bei der hier vorgestellten Arbeit wurde ein Framework entwickelt, mit dem eine unterirdisch lebende Population dargestellt werden kann. Dabei wurde darauf geachtet, das Framework modular und objektorientiert aufzubauen und möglichst offen für viele verschiedene Fragestellungen und auch verschiedene Tierarten aufzubauen. So ermöglicht das Framework eine Simulation von unterschiedlichsten Baustrukturen, die mit Hilfe einer Textdatei eingelesen werden können. Außerdem wurden Agenten innerhalb der Simulation modelliert, die das Verhalten von Tieren nachbilden können.

Desweiteren sollte es ermöglicht werden, in einer sinnvollen Rechenzeit möglichst genau kurze Simulationen von wenigen Stunden, aber auch Simulationen von mehreren Monaten bis zu Jahren berechnen zu können. Auch dies wurde erfüllt und eine Simulation von 2 Monaten ist in 10 bis 30 Minuten möglich. Die hier vorgestellten Simulationen von sechs Monaten bis zu einem Jahr konnten jeweils auf einem einzelnen CPU-Core innerhalb von 48 h errechnet werden.

Desweiteren sollten sich die simulierten Agenten innerhalb des Baus selbstständig bewegen können, sich selbst für ihr Überleben sinnvolle Ziele setzen und ein einfaches Verhaltensrepertoire zeigen. Auch dies konnte mit Hilfe von Wegfindungsalgorithmen und einem Aktionssystem realisiert werden. Außerdem wurde darauf geachtet, dass eine Erweiterung oder Änderung der Bewegungsstrategie und der möglichen Aktionen der Agenten leicht zu realisieren ist. Auch Interaktionen zwischen verschiedenen Tieren können einfach ermöglicht werden.

Zu einer besseren Beobachtbarkeit und Übersicht über die Geschehnisse innerhalb der Simulation wurden verschiedene Listenersysteme eingeführt, die eine genaue Betrachtung jedes Agenten, der gesamten Population und des Baus ermöglichen. Hierzu wurden außerdem mehrere Fenster entwickelt. Sie ermöglichen unter anderem eine Betrachtung des Baus und den darin befindlichen Agenten und Futterstücken, der Populationsentwicklung innerhalb der Simulation oder die Betrachtung einzelner Agenten und deren Wegverfolgung. Außerdem wurde es ermöglicht Daten über die Bewegungen innerhalb der Simulation und die damit verbundenen

Blockaden festzuhalten, weiterzuverarbeiten und bildlich anzuzeigen. Zusätzlich zu den Anzeigefenstern ist es möglich alle erwünschten Daten in einem Logfile auszugeben und dort abzuspeichern. Auch das Abspeichern jedes Fensters ist von Hand oder automatisch, beispielsweise nach einer bestimmten Zeit oder beim Eintritt eines bestimmten Ereignisses, möglich. Das erstellte Framework ermöglicht es somit Aktionszusammensetzungen der Population, durchschnittliche Geschwindigkeiten der Agenten, aber auch die Baunutzung oder Blockaden innerhalb des Baus aufzuzeigen und einfach graphisch darzustellen. Hierbei ermöglichen die verschiedenen Fenster eine Beobachtung der Simulation in Echtzeit. Die Daten können jederzeit abgerufen werden und werden selbstständig dargestellt und abgespeichert.

Mithilfe dieses Frameworks wurde ein einfaches Rattenmodell entwickelt. Die simulierten Ratten zeigen einfache Verhaltensweisen, die zum Überleben ihrer Population notwendig sind. Dazu gehört unter anderem ein Energieverbrauch und ein Schlafzyklus, allerdings auch die damit verbundene selbstständige Befriedigung dieser Bedürfnisse. Um es den Agenten zu ermöglichen Futter zu finden, wurde ein einfaches Modell einer Welt außerhalb des Baus entwickelt. Die Simulation der Außenwelt beschränkt sich auf ein Verstreichen der Zeit. Zudem können die Ratten außerhalb des Baus Futter finden und von Predatoren getötet werden, wobei die Wahrscheinlichkeit dieser Ereignisse von der außerhalb verbrachten Zeit abhängt. Kann ein Agent seine Bedürfnisse nicht ausreichend stillen, stirbt er bei Erreichen eines bestimmten Hungerwerts. Außerdem wurde es ermöglicht, eine Vermehrung der Tiere zu simulieren. Es wurde ein einfaches Modell eines Ovulationszyklus entwickelt. Ein weiblicher Agent hat eine bestimmte Wahrscheinlichkeit beim Eintritt der Hitze trächtig zu werden. Diese Wahrscheinlichkeit ist vom Alter der jeweiligen Ratte abhängig. Nach einer Tragzeit von 21 Tagen bekommt die Ratte Junge. Die Wurfgröße ist vom Alter abhängig.

Außerdem musste eine selbstständige Bewegungsstrategie innerhalb des Baus entwickelt werden, die ein Zusammenleben mehrerer simulierter Ratten innerhalb des Baus ermöglichten. Es wurde ein Modell erzeugt, dass davon ausgeht, dass die Tiere immer den kürzesten Weg zwischen ihrem Standort und einem Ziel wählen. Ist dieser Weg nicht möglich, wurde ein neuer möglicher Weg gesucht zu einem vergleichbaren Ziel oder dem ursprünglichen Ziel, immer darauf bedacht, eine möglichst kurze Wegstrecke zurückzulegen, um den Zweck der Bewegung auszuführen.

Mit dieser Simulation können nun unterschiedliche Szenarien untersucht werden. Die hier vorgestellten Simulationen haben sich auf den Einfluss der Futtergrößenverteilung und der Futtereintragestrategie konzentriert. Bereits bei diesem einfachen Modell zeigten sich interessante Zusammenhänge etwa zwischen der Rattenanzahl innerhalb des Baus, der Futtereintragestrategie und der Aktionszusammensetzung der simulierten Ratten. So wurde festgestellt, dass die Tiere eine Futtergrößenverteilung von N(220/60) bei den vorgestellten Simulationen benötigen, vorausgesetzt die Tiere benötigen alle eine Futtermenge von 15 g am Tag. Ist

das Futtervorkommen geringer, können beide Futtereintragestrategien keinen dauerhaften Futtervorrat im Bau aufbauen. Dies führt bei der Futtereintragestrategie 1 (möglichst kurzer Aufenthalt außerhalb vom Bau) zu einer erhöhten Aufenthaltsdauer außerhalb vom Bau und einer geringeren Laufbewegung innerhalb des Baus (vgl. Abbildung 4.1). Hierbei zeigt sich zusätzlich, dass die Tiere nun verstärkt nur noch Bereiche zwischen einem Schlafplatz und einem Ausgang benutzen (Abbildung 4.14). Dies erklärt sich durch einen dauerhaften hohen Hungerwert der Tiere. Damit werden die Agenten zu ständigen Futtersuchbewegungen veranlasst. Da sich oftmals keinerlei Futtervorrat im Bau befindet, müssen die Agenten außerhalb des Baus nach Futter suchen. Daher werden vermehrt Wege von Kammern zu Ausgängen und von Ausgängen zum nächstgelegenen Schlafplatz benutzt. Desweiteren verhindert ein erhöhter Hungerwert andere Aktionen, die nicht direkt aus dem Hunger- und Schlaflevel heraus getroffen werden. Es kommt zu einer geringeren Baunutzung der Tiere und einer erhöhten Aufenthaltsdauer der Tiere außerhalb des Baus. Außerdem gehen die Tiere oftmals erst bei einem Schlafwert von 396 schlafen. Es fehlen Situationen, bei der die simulierte Ratte bei guten Basiswerten, zufällig aus den Aktionen RandomWalk, Sleep und GoOutside wählt.

Die Futtereintragestrategie 2 zeigt ähnliche Verhaltensweisen, doch ist hier die Verhaltensweise noch etwas flexibler und kann mehr auf bestimmte Situationen eingehen. So führt hier ein Futterdefizit zu einer noch längere Aufenthaltszeit außerhalb des Baus. Hierbei verbleiben die Tiere oftmals so lange wie möglich außerhalb vom Bau (vgl. Abbildung 4.9), fressen auch direkt außerhalb das gesamte gefundene Futter auf und gehen oftmals erst in den Bau, wenn ihr Müdigkeitslevel über 396 gestiegen ist (vgl. Abbildung 4.3). Bei diesem Level ist die nächste Aktion zwingendermaßen eine Sleepaktion. Auch hier sind die Bewegungen innerhalb des Baus extrem gering und beschränken sich lediglich auf das Aufsuchen von Kammern, um entweder dort zu schlafen oder ein Futterstück abzulegen, falls der Hunger für einen kurzen Zeitraum befriedigt ist. Zufällige Aktionen werden durch einen dauerhaft erhöhten Hungerstatus verhindert und es kommt zu einer Baunutzung der Tiere, die sich lediglich auf die Wege zwischen Ausgängen und Kammern beschränkt (vgl. Abbildung 4.15). Da die Tiere rund 42% ihrer Zeit mit Schlafen innerhalb des Baus verbringen müssen, zeigt ein Wert von 58% außerhalb des Baus an, dass die Tiere im Grunde ihre gesamte Wachphase außerhalb des Baus verbringen und wirklich nur zum Schlafen den Bau betreten.

Die Tiere sind bei einem zu geringen Futterangebot allein durch ihre Hunger- und Müdigkeitswerte bestimmt und ihre gesamten Aktionen sind allein auf die Befriedigung dieser Bedürfnisse ausgerichtet.

Bei einem knappen, aber ausreichenden Futterangebot, bei einer Futtergrößenverteilung von N(220/60), zeigen sich erste messbare Unterschiede im Futtervorkommen innerhalb des Baus. Bei der Futtereintragestrategie 1 kann hier noch kein Futtervorrat im Bau angelegt werden. Die Futtereintragestrategie 2 hingegen führt zu einem stetig wachsenden Futtervorrat bei einer Populationsgröße unter 40 Rat-

ten. Bei einem Futtervorkommen über 40 Ratten kann hingegen auch hier kein dauerhafter Futtervorrat angelegt werden. Dieses Ergebnis zeigt wiederum mehrere Zusammenhänge innerhalb der Simulation. Die Regelkreise, die zu diesem Ergebnis führen können, sollen hier nun genauer betrachtet werden.

Die niedrigen Hunger- und Müdigkeitswerte bei einer geringen Populationsdichte (vgl. Abbildung 4.1 und 4.2) bei ausreichendem Futtervorkommen von N(220/60) erklären sich in beiden Futtereintrageverhaltensweisen durch die Aktionen bei geringem Müdigkeits- und Hungerlevel, da dann die simulierten Ratten zufällig zu gleichen Teilen aus den Aktionen GoOutside, Sleep und RandomWalk wählen. Befindet sich die Population in einem guten Zustand, führt dies daher zu regelmäßigen Schlafphasen der simulierten Ratten auch schon bei geringen Müdigkeitswerten. Wählt die simulierte Ratte zufällig die Aktion GoOutside führt dies bei der Futtereintragestrategie 1 und 2 zu einem Futtereintragen, da die Tiere außerhalb des Baus Futter finden und dieses dann bei einem niedrigen Hungerwert auch nicht sofort vollständig auffressen.

Der unterschiedliche Futtervorrat der beiden Futtereintragestrategien bei gleicher Futtergrößenverteilung von N(220/60) (vgl. Abbildung 4.10) lässt sich anhand mehrerer Faktoren erklären: Bei der Futtereintragestrategie 1 finden die Tiere durchschnittlich weniger Futter, da sie den Bau immer nur eine bestimmte Zeit verlassen und die Futterfundwahrscheinlichkeit von der Aufenthaltsdauer außerhalb des Baus abhängt. Somit können die Tiere auch bei einer geringen Populationsdichte keinen dauerhaften Futtervorrat innerhalb des Baus aufbauen, obwohl sie den Bau sehr oft verlassen, um Futter zu suchen. Dieses Futter wird allerdings sofort im Bau ganz aufgefressen. Zusätzlich müssen die Tiere alles Futter innerhalb des Baus ablegen und fressen. Dies ist wiederum ein Nachteil bei der Futtersuche, da hier mehr Zeit mit Bewegungen innerhalb des Baus verbracht werden muss.

Bei der Futtereintragestrategie 2 hingegen verbringen die Tiere mehr Zeit außerhalb des Baus und auch die Zeitintervalle sind länger. Somit erhöht sich die Wahrscheinlichkeit Futter zu finden und die Tiere gehen nur in den Bau, wenn sie Futter in den Bau einlagern möchten oder eine Kammer zum Schlafen aufsuchen. Dies führt bei einer geringen Populationsdichte zu einem dauerhaften Futtervorrat. Bei einer höheren Populationsdichte wiederum kann auch mit der Futtereintragestrategie 2 kein dauerhafter Futtervorrat aufgebaut werden. Dafür gibt es mehrere Gründe: Man erkennt, dass bei einer steigenden Anzahl an Tieren die eingetragene Futtermenge pro Tier sinkt (vgl. Abbildung 4.10). Zusätzlich verlassen die Tiere den Bau sehr viel seltener bei mehr als 40 Ratten innerhalb der Simulation (vgl. Abbildung 4.6 B). Dies ist mit der höheren Populationsdichte zu erklären. Sie führt zu mehr Blockaden und mehr Ausweichbewegungen. Hinweise liefern Daten, wie eine sinkende durchschnittliche zurückgelegte Wegstrecke ohne einer anderen Ratte zu begegnen (vgl. Abbildung 4.18), das exponentiell steigende Verhältnis der durchschnittlich zurückgelegten Wegstrecke zum kürzest möglichen Weg (vgl. Abbildung 4.19) und die sinkende theoretische Geschwindigkeit bei ansteigender Populations-

größe (vgl. Abbildung 4.20). Dies führt dazu, dass die simulierten Tiere oftmals hungriger aus dem Bau gehen, da sie mehr Zeit innerhalb des Baus benötigen um an ein selbstgewähltes Ziel zu kommen. Dies führt wiederum zu längeren Aufenthaltszeiten außerhalb des Baus und zu einem geringeren Futtereintrageverhalten, da mehr Futter außerhalb vom Bau gefressen wird.

Bei einer Futtergrößenverteilung von N(250/60) konnte in beiden Strategien ein Futtervorrat innerhalb des Baus aufgebaut werden. Es zeigt sich hierbei, dass mit der Futtereintragestrategie 2 ein deutlich größeren Futtervorrat anlegt werden kann (vgl. Abbildung 4.11). Es zeigte sich allerdings auch, dass die Futtermenge innerhalb des Baus pro Ratte bei einer höheren Populationsdichte sinkt (vgl. Abbildung 4.12).

Ein Futterüberfluss und der damit mögliche Futtervorrat führt dazu, dass die Tiere den Bau nicht mehr aus Hunger verlassen, da sie das eingelagerte Futter im Bau fressen können. Dies führt wiederum zu einer erhöhten Bewegungsrate innerhalb des Baus und dadurch bedingt zu mehr Blockaden. Außerdem führt der Futterüberfluss zu guten Basiswerten der Ratte, die Agenten wählen somit einen Großteil ihrer Aktionen zufällig aus. Dies führt dazu, dass die Ratten viele zufällige Bewegungen innerhalb des Baus ausführen, viele weitere Futtereintrageaktionen starten und auch kurze Schlafphasen ausführen.

Die Tiere der Futtereintragestrategie 1 verlassen den Bau nicht seltener bei einer größeren Futtermenge im Vergleich zu einer geringeren Futtergröße (vgl. Abbildung 4.6 A). Trotzdem verändert sich der Grund des Verlassens. Die Tiere verlassen den Bau bei einer größeren Futtergrößenverteilung nicht mehr aus akutem Hunger sondern mehr aufgrund der zufällig gewählten GoOutside Aktion bei guten Grundwerten der Tiere (siehe Abbildung 4.8).

Bei der Futtereintragestrategie 2 verändert die normalverteilte Futtergrößenwahrscheinlichkeit hingegen die Austrittsanzahl (vgl. Abbildung 4.6 B). Bei einer höheren Futtermenge gehen die simulierten Ratten aufgrund der zufälligen Wahl zwischen den Aktionen GoOutside, Sleep und RandomWalk nach draußen. Erst bei einer Population von 80 und mehr Agenten führen die gegenseitigen Blockaden der Ratten zu einer Herabsetzung der Austrittsanzahl, da die Tiere dann aufgrund der gegenseitigen Blockaden kein Futter in den Bau bringen können und sich somit kein Futtervorrat im Bau etabliert. Bei einem geringeren Vorkommen an Futter bei einer Futtergrößenverteilung von N(220/60) hingegen, verlassen die Tiere den Bau seltener. Man muss hierbei bedenken, dass bei bis zu 40 Ratten ein konstanter Futtervorrat angelegt werden kann. Dieser Futtervorrat führt dazu, dass es der Mehrzahl der Tiere sehr gut geht und sie den Bau wiederum allein zum Erweitern des Futtervorrates verlassen (vgl. Abbildung 4.9). Bei einer Populationsgröße über 40 Tiere kann allerdings bei einer Futtergröße von N(220/60) kein dauerhafter Futtervorrat mehr angelegt werden. Dies führt dazu, dass die Tiere wiederum mehr auf Grund eines akut hohen Hungerwertes den Bau verlassen. Der Bau wird dann für eine längere Zeit verlassen. Dies führt zu einem Abfall der Austrittsanzahl pro Ratte.

Daher zeigen die beiden Kurven aus Abbildung 4.9 einen unterschiedlichen Verlauf und eine unterschiedliche Rattenanzahl bei der die Austrittsanzahl abfällt. Dieser Wert hat jeweils einen direkten Zusammenhang mit der Futtermenge innerhalb des Baus bzw. die Austrittsanzahl sinkt extrem ab, sobald sich kein Futter mehr im Bau befindet.

Bei allen getesteten Futtereintragestrategien und Futtergrößen zeigte sich, dass die Agenten sich mit steigender Populationsgröße mehr blockieren. Vergleicht man allerdings die Futtereintragestrategien unter unterschiedlichen Futtergrößenbedingungen miteinander bei gleicher Rattenanzahl, zeigte sich, dass eine höhere Bewegung innerhalb des Baus (vgl. Abbildung 4.4) nicht unbedingt zu mehr Blockaden führen muss (vgl. Abbildung 4.21).

Bei der Futtereintragestrategie 1 fällt auf, dass die Futtergröße scheinbar einen starken Einfluss auf die Baunutzung und die hervorgerufenen Blockaden hat. Hierbei zeigt sich, dass sich die Agenten bei einer geringen Futtergröße deutlich stärker blockieren (Abbildung 4.21 oben und 4.22 oben) als bei einer größeren Futtergröße (Abbildung 4.21 unten und 4.22 unten). Dieser Unterschied wird vermutlich unter anderem durch die Menge an eingelagertem Futter hervorgerufen. Bei einer geringen Futtergröße wird kein dauerhafter Futtervorrat angelegt. Allerdings führt ein geringer Futtervorrat dazu, dass Tiere die hungrig sind, dann alle zu diesem Futtervorrat gehen. Daher werden diese Bereiche dann kurzzeitig besonders stark von vielen Agenten aufgesucht. Dies würde die extrem hohe Anzahl an Blockaden direkt vor einigen Kammern erklären. Bei einer größeren Futtergröße kann ein dauerhafter Futtervorrat angelegt werden. Die Tiere suchen daher das Futter auch in Kammern, allerdings befindet sich das Futter an mehreren Plätzen, so dass sich die Tiere besser verteilen. Außerdem bewegen sie sich häufiger innerhalb des Baus zu zufälligen Orten, da ihre Hunger- und Müdigkeitswerte oftmals gering sind (vgl. Abbildung 4.1 und 4.2) und sie daher mehr zufällige Bewegungen innerhalb des Baus ausführen. Dies führt zu einer größeren Verteilung der Ratten und die Blockaden insgesamt sind geringer.

Bei der Futtereintragestrategie 2 zeigt sich kein erkennbarer Unterschied zwischen den verschiedenen Futtergrößenverteilungen, weder bei 10 Ratten (Abbildung 4.23), noch bei 30 Ratten (Abbildung 4.24). Dies erklärt sich dadurch, dass bei beiden Futtergrößen bereits ein dauerhafter Futtervorrat angelegt werden konnte.

Bei den Simulationen mit Vermehrung zeigte sich zum einen, dass ein kurzfristiger Anstieg des Futterbedarf eines Teils der Population bereits zu einer starken Erhöhung der grundsätzlich benötigten Futtergröße führen kann. Es zeigte sich außerdem, dass die Futtereintragestrategie 2 einen klaren Vorteil hat, da sie eine viel geringere Futtergröße benötigte. Dies wird zum einen durch einen früheren möglichen dauerhaften Futtervorrat und zum anderen der Tatsache, dass die Tiere hier länger außerhalb vom Bau verbleiben können, das Futter direkt draußen fressen können und durch einen längeren Aufenthalt außerhalb des Baus eine höhere Wahrscheinlichkeit besitzen Futter zu finden. Desweiteren konnte sehr eindrücklich die

enorme Vermehrungsrate der Ratten aufgezeigt werden und wie stark die unterschiedlichen Futtereintragestrategien die Populationsgröße aufgrund der Baugröße und Baunutzung beschränken. Es zeigte sich, dass die Futtereintragestrategie 1 bei guten Futterverhältnissen eine Population von 20 bis 40 Tieren ermöglicht. Dagegen ermöglicht die Futtereintragestrategie 2 eine Populationsgröße von 35 bis 60 Ratten. Dies bedeutet, dass die Futtereintragestrategie einen enormen Einfluss auf die Populationsgröße unserer simulierten Rattenpopulation hat. Hierbei ist besonders der Anteil an Agenten wichtig, der einen niedrigen Hunger- und Müdigkeitslevel aufweist, da die simulierten Ratten dann einen Futtervorrat anlegen können und auch in Zukunft ein geringes Müdigkeitslevel aufweisen können.

Das entwickelte Rattenmodell mit der zugehörigen Bewegungsstrategie ermöglichte es den Agenten selbstständig ihre Bedürfnisse zu stillen und eine konstante Population zu erhalten.

Allerdings beachteten die bisherigen Simulationen nicht den Einfluss eines Prädators. Daher wurde es ermöglicht einen Prädator außerhalb des Baus zu simulieren. Bei Simulation mit einer Prädatorenwahrscheinlichkeit von $y = (0{,}00001/500) * (\text{Aufenthaltszeit außerhalb des Baus in Sekunden})$ zeigte sich, dass in diesem Fall in der Futtereintragestrategie 2 mehr Tiere aufgrund des Prädators starben als bei der Futtereintragestrategie 1. Allerdings zeigte sich auch, dass insgesamt weniger Tiere bei beiden Futtereintragestrategien mit Prädator starben als ohne Prädator. Auch starben bei der Futtereintragestrategie 2 wiederum weniger Tiere insgesamt als bei der Futtereintragestrategie 1. Dies ist allerdings auf den statischen Bau zurückzuführen. Dieser hatte in allen gezeigten Simulationen einen enormen Einfluss auf die Sterberate der Tiere. In der Realität ist ein bewohnter Rattenbau allerdings nicht statisch und verändert sich kontinuierlich mit der Rattenanzahl im Bau. Will man nun also weiter Simulationen mit einer wachsenden Population untersuchen, wäre es sinnvoll ein Modell zur Bauerweiterung zu entwickeln. Damit könnte der extreme Einfluss der Baugröße eventuell etwas abgeschwächt werden und es wäre möglich eine realistischere wachsende Population zu simulieren. Eine andere Möglichkeit den Einfluss des Baus weiter zu senken, wäre die Integration eines Migrationsmodells von Ratten.

Außer dieser Erweiterung sind viele weitere Ansätze und Untersuchungen möglich. So könnte zum Beispiel getestet werden, welchen Einfluss die Veränderung der Bewegungsstrategie innerhalb des Baus hat. Ein denkbarer Ansatz wäre beispielsweise, dass die Tiere nicht immer den kürzesten Weg zu einem Ziel wählen sondern eventuell einen zufälligen Ort wählen, der den Zweck des Ziels ermöglicht. Eine weitere Möglichkeit wäre etwa eine andere Verteilung der Aktionen bei einem niedrigen Hunger- und Müdigkeitswert. Bisher wählten die Agenten hier aus den drei Aktionen GoOutside, Sleep und RandomWalk zu gleichen Teilen. Nun könnte hier allerdings eine andere Verteilung angenommen werden oder es könnten noch zusätzliche Aktionen hinzugenommen werden. Desweiteren könnte man ein Hierarchiesystem einführen. So könnte ein Agent mit einem geringen Status einem Agen-

ten mit einer höheren Rang ausweichen, so dass höher gestellte Tiere weniger Ausweichbewegungen machen müssen als niedriger gestellte Tiere. Dies könnte große Auswirkungen auf die Baunutzung unterschiedlicher Tiere haben. Man würde somit erwarten, dass niedrig gestellte Tiere den gesamten Bau ausnützen und höher gestellte Tiere nur die Hauptverbindungsstrecken nutzen. Auch könnte betrachtet werden, welche Auswirkungen dieses System auf die Überlebenswahrscheinlichkeit der Tiere hat.

Außerdem kann die Baustruktur weiter untersucht und verändert werden und somit eventuell ein idealer Bau für die hier entwickelte Bewegungsstrategie entwickelt werden. Beispielsweise könnten weitere Rattenbauten eingelesen und getestet werden. Außerdem ist es möglich diese von Hand zu verändern und so eventuell ein Erweiterungsmuster für Baustrukturen zu entwickeln.

Zusätzlich könnte auch die Umwelt der Tiere außerhalb des Baus simuliert werden. Eine komplexere Umwelt könnte aus unterschiedlichem Futter bestehen, es könnten Jahreszeiten simuliert werden oder auch unterschiedliche Prädatoren simuliert werden. Außerdem könnte die Futtermenge, die zu einem bestimmten Zeitraum zur Verfügung steht begrenzt werden bzw. ein Verfallsdatum des Futters eingeführt werden.

Außerdem kann das Verhaltensrepertoir der Agenten jederzeit flexibel und einfach erweitert werden. Zusätzlich ist auch eine Simulation weiterer Tierarten und Baustrukturen denkbar. Mit der hier vorgestellten Framework können jederzeit neue Erkenntnisse über die betrachtete Tierart und ihre Bewegungsstrategie eingefügt und ein immer komplexeres und realistischeres Modell entwickelt werden.

Kapitel 6

Anhang

6.1 worldgraph.txt

Vertices are defined by a line like this: # V name x y z

V A01 3 4 0.5 exit=true
V A01a 3 7 0.5
V A02 19 5 0.8 exit=true
V A02a 19 7 0
V A03 4 9 0.5
V A03a 5 11 0.5
V A04 19 10 0
V A04a 15 10 0
V A04b 22 12 0
V A05 3 11 0
V A06 12 11 0
V A06a 9 12 0
V A07 31 11 0.8 exit=true
V A07a 28 12 -0.8
V A08 2 13 0.3 exit=true
V A09 24 13 -1.0
V A09a 21 16 -1.0
V A10 6 14 0
V A10a 7 16 0
V A11 16 16 0
V A11a 12 17 0
V A12 19 16 0
V A13 9 18 -1.0

```
V A14 17 18 0
V A14a 20 21 0
V A14b 15 20 0
V A15 6 20 0.3 exit=true
V A16 9 20 0
V A17 12 21 -3
V A17a 14 21 -3
V A18 16 21 -3
V A19 27 22 0
V A20 28 21 -2
V A21 29 22 -2
V A23 23 23 -1.5
V A23a 21 24 -1.5
V A24 18 24 0
V A24a 16 23 0.3 exit=true
V A25 6 25 0
V A26 9 23 0
V A26a 11 26 0
V A27 13 23 0
V A28 12 24 0.3 exit=true
V A29 14 24 0
V A29a 16 25 0
V A30 19 26 0
V A31 19 28 0
V A32 18 29 0
V A33 14 28 0
V A34 12 28 0
V A35 12 32 0
V A35a 9 34 0
V A36 7 36 -1.0
V A37 9 37 0
V A38 12 35 -2.0
V A38a 10 33 -2.0
V A38b 8 33 -2.0
V A39 13 37 0
V A40 15 35 0
V A41 17 37 0.3 exit=true
```

```
V A42 18 33 0
V A43 19 37 0
V A44 21 32 0
V A45 23 34 0
V A46 26 37 0
V A47 28 39 0
V A48 29 38 0
V A49 27 36 0.3 exit=true
V A50 25 34 0.3 exit=true
V A51 23 32 0.3 exit=true
V A52 26 28 0.3 exit=true
V A53 28 27 0.3 exit=true
V A54 33 27 0.3 exit=true
V A55 36 28 0
V A56 35 31 0
V A57 33 34 0
V A57a 32 36 0
V A58 40 28 0
V A59 43 37 0
V A59a 44 34 0
V A60 44 30 0
V B01 49 31 0
V B01a 51 34 0
V B01b 47 31 0
V B02 51 28 0
V B02a 51 26 0
V B03 52 23 0 exit=true
V B04 54 37 0 exit=true
V B05 58 38 0
V C01 15 40 0
V C01a 14 43 0
V C02 14 45 0
V C03 13 46 0
V C04 14 48 0
V C06 19 42 0
V C07 23 43 0
V C08 22 46 0
```

```
V C08a 21 49 0
V C09 26 44 0
V C09a 26 42 0
V C09b 27 47 0
V C10 20 51 0
V C10a 19 53 0
V C11 19 55 0
V C12 24 54 0
V C13 24 50 0.3 exit=true
V C14 27 49 0
V C15 27 54 0
V C15a 30 52 0
V C16 30 57 0
V C17 34 58 0
V C17a 37 59 0
V C18 34 55 0
V C19 35 52 -2
V C19a 37 51 -2
V C20 32 51 0
V C21 40 51 0
V C22 41 54 0
V C22a 44 55 0
V C23 40 60 0
V C24 44 59 0
V D01 48 56 0
V D02 49 53 0 exit=true
V D02a 49 50 0
V D03 50 47 0
V D04 52 43 0
V D04a 53 40 0
V D05 68 50 -2
V D06 66 47 -1
V D06a 66 45 0
V D07 67 42 0.3 exit=true
V D07a 62 40 0
V D07b 64 41 0
V D08 71 45 0
```

```
V D08a 69 44 0 V D09 74 42 0.3 exit=true
V D10 78 42 0.3 exit=true
V D11 83 42 0.3 exit=true
# Edges connect two vertices. They are defined like this: # E v1 v2 [cap=1.000]
# SECTOR A, LINE BY LINE, BOTTOM TO TOP
E A01 A01a
E A01a A03
E A02 A02a
E A02a A04
E A03 A05
E A03 A03a
E A03a A10
E A04 A04a
E A04a A06
E A04 A04b
E A04b A09
E A09 A07a
E A07a A07
E A06 A06a
E A06a A10
E A10 A08
E A09 A09a cap=5.0 sleep=true
E A09a A12
E A10 A10a
E A10a A13
E A11 A12
E A11 A11a
E A11a A13
E A12 A14
E A13 A16
E A14b A27
E A14 A14a
E A14 A14b
E A14a A23
E A15 A16
E A16 A17
E A20 A19 cap=3.0 sleep=true
```

```
E A20 A21 cap=3.0 sleep=true
E A17 A27
E A17 A18 cap=10 sleep=true
E A17 A26
E A19 A23
E A26 A25
E A26 A26a
E A26a A34
E A27 A28 cap=4.0 sleep=true
E A27 A29
E A24a A29
E A24a A24
E A23 A23a
E A23a A30
E A29 A29a
E A29a A30
E A30 A31
E A53 A54
E A53 A52
E A54 A55
E A34 A35
E A34 A33
E A28 A33
E A33 A32 cap=5.0 sleep=true
E A31 A32
E A31 A44
E A52 A51
E A55 A58
E A55 A56
E A58 A60
E A59a A59
E A60 A59a
E A60 B01b
E A56 A57 cap=3.0 sleep=true
E A57 A57a cap=5.0 sleep=true
E A35 A35a
E A35a A36
```

```
E A35 A38
E A38 A38a cap=4.0 sleep=true
E A38a A38b cap=4.0 sleep=true
E A44 A42
E A44 A45
E A51 A50
E A42 A40
E A42 A43
E A50 A45
E A50 A49
E A45 A46
E A38 A40
E A38 A37
E A38 A39
E A36 A37
E A49 A46
E A49 A48
E A43 C06
E A41 C01
E A46 A47
E A48 A47 cap=6.0 sleep=true
E A39 C01
E A47 C09a
# SECTOR B
E B01 B01b
E B03 B02a
E B02a B02
E B02 B01
E B01 B01a
E B01a B04
E B04 D04a
E B04 B05 cap=4.0 sleep=true
E B05 D07a
# SECTOR C
E C01 C06
E C01 C01a
E C01a C02
```

```
E C06 C07
E C06 C08
E C09 C08
E C09a C09
E C09 C09b
E C09b C14
E C14 C15
E C02 C03 cap=5.0 sleep=true
E C03 C04 cap=5.0 sleep=true
E C08 C08a
E C08a C10
E C10a C10 cap=5.0 sleep=true
E C10a C11 cap=5.0 sleep=true
E C10 C12
E C12 C13
E C12 C15
E C15 C16
E C15 C15a
E C15a C20
E C16 C17
E C20 C19 cap=4.0 sleep=true
E C17 C17a
E C17 C18 cap=3.0 sleep=true
E C19a C21
E C19 C19a
E C19 C18 cap=4.0 sleep=true
E C17a C23
E C23 C24
E C24 D01
E C22 C22a
E C22a D01
# SECTOR D
E D01 D02
E D02 D02a
E D02a D03
E D04a D04
E D03 D04 cap=4.0 sleep=true
```

```
E D06 D06a cap=5.0 sleep = true
E D07 D06a
E D06 D05 cap=5.0 sleep=true
E D07 D07b
E D07a D07b
E D07 D08a
E D08a D08
E D08 D09
E D09 D10
E D10 D11
```

6.2 Flow Charts

6.2.1 Flow Chart Walk

6.2.2 Flow Chart Random Walk

6.2.3 Flow Chart MakeAStep

6.2.4 FlowChart MakeAStepWithAStar

Literaturverzeichnis

[Bar58] S. A. Barnett. An analysis of social behaviour in wild rats. *Proc. Zool. Soc. Lond.*, 130:107–152, 1958.

[Cal63] J. Calhoun. *The Ecology and Sociology of the Norway Rat.* U.S. Dept. of Health, Education, and Welfare. Public Health Service, 1963.

[CB83] Peters L. C. and Kristal M. B. Suppression of infanticide in mother rats. *J Comp Psychol.*, 97(2):167–177, 1983.

[DDE48] Stokes A. W. Davis D. E., Emlen J.T. Jr. Studies on home range in the brown rat. *Journal of Mammalogy*, 29:207–225, 1948.

[E.48] Davis D. E. The survival of wild brown rats on a Maryland farm. *Ecology*, 29:437–448, 1948.

[EHPSDvdW00] M. J. Engelbregt, M. E. Houdijk, C. Popp-Snijders, and H. A. Delemarre-van de Waal. The effects of intra-uterine growth retardation and postnatal undernutrition on onset of puberty in male and female rats. *Pediatr. Res.*, 48:803–807, Dec 2000.

[EO95] Nakatsuyama E. and Fujita O. The influence of the food size, distance and food site on food carrying behaviour in rats (Rattus norvegicus). *J.Ethol.*, 13:95–103, 1995.

[Fab09] Gregor Fabritius. Burrowview path reconstruction and localization in a subterranean animal habitat observation system. Master's thesis, RWTH Aachen University, 2009.

[Flo62] Robert W. Floyd. Algorithm 97: Shortest Path. *Communication of the ACM*, 5:345, June 1962.

[Grz00] Bernhard Grzimek. *Grezimerks Tierleben Enzyklopädie des Tierreichs Band 11 Säugetiere 2*. Bechtermünz, 2000.

[GSNF+09] L. C. Gardner-Santana, D. E. Norris, C. M. Fornadel, E. R. Hinson, S. L. Klein, and G. E. Glass. Commensal ecology, urban landscapes, and their influence on the genetic characteristics of city-dwelling Norway rats (Rattus norvegicus). *Mol. Ecol.*, 18:2766–2778, Jul 2009.

[Han10] Anne Hanson. Rat behavior and biology. http://www.ratbehavior.org/, Feb 2010.

[HPE68] Raphael B. Hart P. E., Nilsson N. J. A Formel Basis for the Heuristic Determination of Minimum Cost Paths. *Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.

[KD78] Taylor KD. Range of movement and activity of common rats (Rattus norvegicus) on agricultural land. *Journal of Applied Ecology*, 15:663–677, 1978.

[Kin39] H. D. King. Life process in gray Norway rats during fourteen years in capitivity. *Amer. Anat. Memoirs*, 17:77pp., 1939.

[Kre93] Davies N. B. Krebs, J.R. *An Introduction to Behaviour Ecology*. Blackwell Publishers, 3. edition, 1993.

[Kri99] Georg J. Krinke. *The Laboratory Rat (Handbook of Experimantal Animals).* Academic Press, 1999.

[Lot84] D. Lott. Intraspecific variation in the social systems of wild vertebrates. *Behaviour*, 88:266–325, 1984.

[MD95] Fenn GP Macdonald DW. Rat ranges in arable areas. *Journal of Zoology, London*, 236:349–353, 1995.

[Moh74] C. Mohan. Age-dependent cannibalism in a colony of albino rats. *Lab. Anim.*, 8:83–84, Jan 1974.

[Moo99] J. Moore. Population density, social pathology, and behavioural ecology. *Primates*, 40:5–26, 1999.

[MP66] Robert H. MacArthur and Eric R. Pianka. On optimal use of a patchy environment. *The American Naturalist*, 100:603–609, Nov-Dec 1966.

[MW81] B. Morgan and M. Winick. A possible control of food intake during pregnancy in the rat. *Br. J. Nutr.*, 46:29–37, Jul 1981.

[NF95] E. Nakatsuyama and O. Fujita. The influance of the food size, distance and food site on food carrying behavior in rats (*Rattus norvegicus*). *Journal of Ethology*, 13:95–103, Jun 1995.

[P.08] Schwedhelm P. Ratpack: Analysis of rodent ultrasonic vocalization using wearable sensor nodes. Master's thesis, Tübingen University, 2008.

[PD93] Freeth G. Pass D. *The rat.* ANZCCART News 6(4):1-4, 1993.

[Rue10] L. Ruedas. Rattus norvegicus. In: IUCN 2009. IUCN Red List of Threatened Species. http://www.iucnredlist.org/, 23. Februar 2010.

[Sch10] Thomas Schulte. Ganganalyse und Pfadrekonstruktion mittels Dead Reckoning bei Ratten. Master's thesis, Eberhard Karls Universität Tübingen, 2010.

[TK78] Quy RJ Taylor KD. Long distance movements of a common rat (Rattus norvegicus) revealed by radio-tracking. *Mammalia*, 42:47–53, 1978.

[TTM+06] D. Traweger, R. Travnitzky, C. Moser, C. Walzer, and G. Bernatzky. Habitat preferences and distribution of the brown rat (Rattus norvegicus Berk.) in the city of Salzburg (Austria): implications for an urban rat management. *Journal of Pest Science*, 79:113–125, 2006.

[TW84] DeSantis D. T. and Schmaltz L. W. The mother-litter relationship in developmental rat studies: cannibalism vs caring. *Dev Psychobiol.*, 17(3):255–262, 1984.

[Ull09] C. Ullenboom. *Java ist auch eine Insel Programmieren mit der Java Standart Edition Version 6.* Galileo Computing, 8. edition, 2009.

[War62] Stephan Warshall. A Theorem on Boolean Matrices. *Journal of the ACM*, 9:11–12, Jan 1962.

[YJ03] Onuki Y. and Makino J. The influence of food deprivation and food size on food carrying in rats (Rattus norvegicus). *Tsukuba Psychol Res.*, 26:9–13, 2003.