Fahrzeugeinsatzplaung

Entwicklung einer Anwendung zur Optimierung

Oliver Pohling, David Mittelstädt, Henrik Voß

15. Juli 2014

Was für ein Projekt!

Inhaltsverzeichnis

1 Einleitung

1.1 Problembeschreibung

1.2 Anforderungen

- Fahrzeug
  - Ein Fahrzeug kann einen Produkttyp transportieren
  - Geschwindigkeit
  - Kapazität
  - Zeitfenster
  - Start- und End-Depot
- Produkt
  - Name
- Auftrag
  - Beliebige Auf- und Ablade-Station
  - Zeitfenster
- Station
  - Namen
  - Koordinaten

2 GenetischerAlgorithmus

Das nachfolgende Kapitel stellt den genetischen Algorithmus, als Verfahren zur Lösung komplexer Optimierungsprobleme, vor. Dazu werden im ersten Schritt die benötigten Grundlagen geschaffen und im Anschluss die speziellen Operatoren im Zusammenhang mit dem „PDPTW“ vorgestellt.

2.1 Grundlagen

Bei den genetischen Algorithmen handelt es sich um Verfahren, die zur Lösung komplexer Optimierungsaufgaben eingesetzt werden. Sie beruhen auf Methoden und Erkenntnisse der biologischen Genetik. Dabei dient insbesondere die Evolutionstheorie als Vorbild für die Entwicklung der genetischen Algorithmen, da die Evolution bislang gute Ergebnisse in der Natur geliefert hat.

Die grundlegende Arbeit, zur Schaffung dieser Verfahrenstechnik, wurde von I. Rechenberg mit dem 1960 erschienenen Werk „Evolutionsstrategie“ geschaffen. Auf diesem Wissen aufbauend, erfand John Holland 1975 die genetischen Algorithmen, die in seinem Werk „Adaption in Natural and Artificial Systems“ niedergeschrieben wurden.

Der genetische Algorithmus beruht dabei auf eine simple Verfahrenstechnik, welche nachfolgend dargestellt und erklärt wird:

- Initialisiere Population P
- Evaluiere alle Individuen in P
- Wiederhole, solange Abbruchbedingung nicht erreicht
  - Selektiere ein Individuenpaar x, y aus P als Eltern
  - Erzeuge zwei Nachkommen x‘, y‘ aus x und y durch Anwendung von Crossover-Operationen mit der Wahrscheinlichkeit p(cross)
  - Erzeuge modifizierte Nachkommen x‘‘, y‘‘ aus x‘ und y‘ durch Anwendung von Mutations-Operatoren mit der Wahrscheinlichkeit p(mut)
  - Füge x‘‘ und y‘‘ der Population P hinzu und entferne schlechtere Individuen
- Gib das beste Individuum aus P als Lösung aus.

Die einzelnen Schritte des genetischen Algorithmus werden dabei in den nachfolgenden Kapiteln anhand einer speziellen Problemstellung (PDPTW) konkretisiert und erläutert.

2.2 Repräsentation

Eine der größten Herausforderungen für die Entwicklung und den Erfolg des genetischen Algorithmus ist die Wahl einer geeigneten Repräsentationsebene. Dazu weißt der PDPTW zwei strukturell heterogene Teilprobleme auf. Zum einen muss eine Zusammenfassung von Aufträgen mit ihrer Zuordnung zu einem Fahrzeug als Gruppierungsproblem dargestellt werden. Auf der anderen Seite ist zusätzlich ein Reihenfolgeproblem zu lösen, indem eine zulässige Route bestimmt wird, in der die Ladeorte anzufahren sind. Aus diesen Einschränkungen folgt, dass beispielsweise eine binäre Kodierung, die im klassischen genetischen Algorithmus angewandt wird, keine

geeignete Repräsentationsform darstellt.

Bevor nun eine Möglichkeit der Darstellung der beiden genannten Aspekte vorgestellt wird, müssen die Begriffe des Individuums und des Gens in den Kontext des PDPTW gebracht werden. Ein Individuum ist im nachfolgenden ein kompletter Tourenplan, der aus der Gesamtheit aller Touren besteht, die zur Erfüllung aller Aufträge notwendig sind. Ein Gen hingegen besteht aus exakt einer Tour, der ein Fahrzeug und die verschiedene Aufträge zugeordnet sind. Dies soll die nachfolgende Abbildung 1 verdeutlichen:

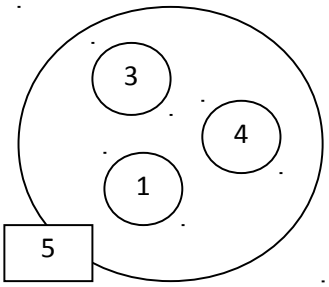
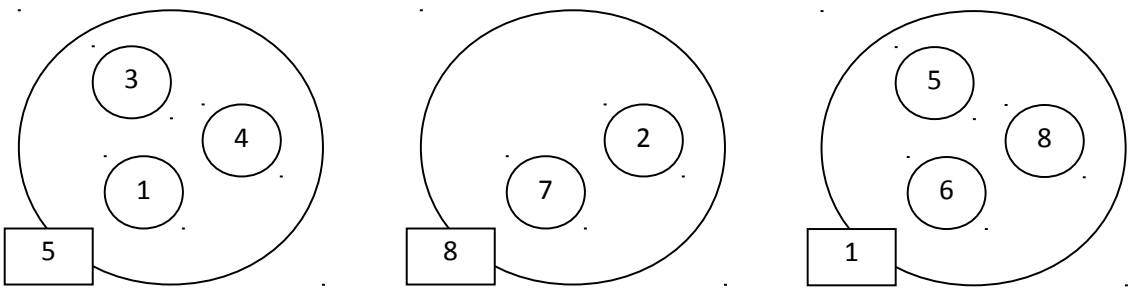


Abbildung 1:

Hier ist im unteren linken Bereich das verwendete Fahrzeug (Fahrzeug 5) zu finden, das für die Auslieferung der Aufträge 1, 3 und 4 verantwortlich ist. Desweiteren soll im späteren Verlauf zusätzlich die Reihenfolge identifizierbar sein, um genau zu wissen, wann welcher Auftrag ab- bzw. aufgeladen wurde. Ein Individuum stellt demnach die Gesamtheit aller Touren dar, die zur Erledigung aller Aufräge gebildet werden müssen. Dies verdeutlich die Abbildung 2:



(a) sd (b) df (c) sd

Abbildung 2:

Hierbei handelt es sich um ein konkretes Individuum, das drei Fahrzeuge verwendet, um die gezeigten acht Aufträge zu erfüllen. Zusätzlich muss jedes Gen mit einer ergänzenden Datenstruktur assoziiert werden, so dass im späteren Verlauf der Reihenfolgeaspekt dargestellt werden kann. Diese Darstellung wird im Umfeld des genetischen Algorithmus als Phänotyp bezeichnet. Dies veranschaulicht die Abbildung 3 auf der nächsten Seite zu einer gegebenen Tour:

Auf diese Informationen haben die genetischen Operatoren unmittelbaren Zugriff, um eine möglichst große Varianz in die spätere Population einzubringen.

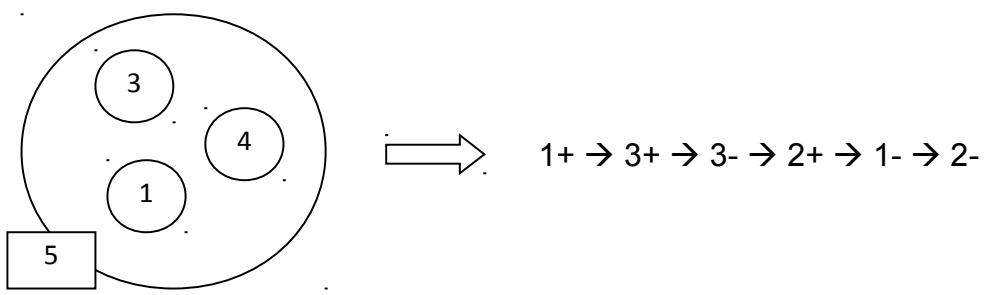


Abbildung 3:

2.3 Aufbau der Startpopulation

Bevor der genetische Algorithmus arbeiten kann, muss im ersten Schritt eine Startpopulation erzeugt werden, die für den Algorithmus als Ausgang dient. Dazu gibt es prinzipiell zwei verschiedene Vorgehensweisen:

1. Zufällige Auswahl von Individuen mit zufälligen Merkmalen
2. Auswahl von „guten“ Individuen, die durch bestimmte Konstruktionsheuristiken erzeugt wurden

Die zweite Variante bringt ein gewisses Risiko mit, da es beim genetischen Algorithmus passieren kann, dass die Suche, aufgrund bereits guter Lösungen, vorzeitigt konvergiert. Um diesem Problem entgegen zu wirken, wurde der Aufbau der Startpopulation mit beiden Möglichkeiten gekoppelt. Dies soll die Abbildung 4 verdeutlichen:

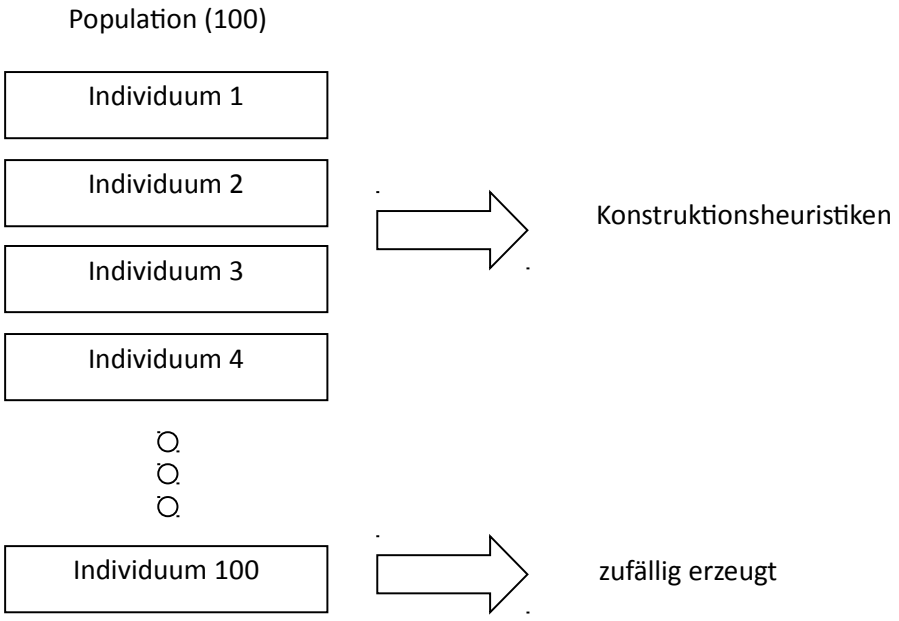


Abbildung 4:

Der Großteil der Population wird dabei zufällig erzeugt, wobei ein ausgewählter Teil sich den Konstruktionsheuristiken bedient. Dabei handelt es sich zum einen um das Nearest-Neighbor- und zum anderen um das Sweep-Verfahren, die in den folgenden Kapiteln kurz vorgestellt werden.

2.3.1 Nearest-Neighbor

Beim Nearest-Neighbor handelt es sich um ein bekanntes heuristisches Eröffnungsverfahren was beispielsweise zur Lösung des „Problems des Handlungsreisenden“ verwendet wird. Da das PDPTW eine Abwandlung von dieser Problemstellung ist, eignet sich diese Heuristik ideal für die Erzeugung von potentiellen Startindividuen. Dabei wird bei dieser Technik von einem beliebigen Startpunkt ausgehend der nächstdichteste Knoten besucht. Dies wird sukzessiv fortgesetzt, bis alle Knoten besucht wurden.

2.3.2 Sweep

Der Sweep-Algorithmus bedient sich einer anderen Technik zum Finden des nächstmöglichen Knotens. Hier wird von einem beliebigen Knoten ausgehend, der Knoten gewählt, der den geringsten Winkel von einer definierten Startkante aufweist. Dies kann sowohl im oder entgegengesetzt dem Uhrzeigersinn durchgeführt werden.

2.4 Individuenauswahl (Selektion)

Die Selektion bestimmt welche Individuen sich paaren dürfen, und erzeugt aus Ihnen die Menge der Nachkommen. Dabei kann die Auswahl entweder komplett zufällig stattfinden oder ins direkte Verhältnis zur Fitness eines Individuums gesetzt werden. Wenn man sich für eine reine fitnessproportionale Selektion entscheidet, so ist ein verhältnismäßig niedriger Selektionsdruck vorhanden, da der genetische Algorithmus verhältnismäßig lange konvergiert. Abhilfe schafft dort eine Ranglistenbasierte Selektion. Bei diesem Verfahren wird die Selektionswahrscheinlichkeit nicht mehr ins direkte Verhältnis zur Fitness gesetzt. Stattdessen werden die unterschiedlichen Individuen nach ihrer Fitness sortiert, so dass die Tour mit dem besten Ergebnis an oberster Stelle der Population steht. Im nächsten Schritt erfolgt die Auswahl der möglichen Kinder für die oberen Elemente der Rangliste mit einer höheren Wahrscheinlichkeit als tieferliegende Individuen.

Zusätzlich wird auch hier auf ein Verfahren zurückgegriffen, welches die möglichen Eltern rein zufällig wählt. Dies hat den Vorteil, dass auch Individuen mit schlechter Fitness als potentieller Elternteil gewählt wird. Dadurch werden ggf. mehr genetische Informationen in die nächste Population übertragen und eine zu frühe Konvergenz, aufgrund von sehr guten Fitnesswerten, verhindert.

Die Kombination der beiden Techniken veranschaulicht die Abbildung 5 auf der nächsten Seite. Dort wird als mögliches Elternpaar zum einen das Individuum 2 aufgrund der Fitness ausgewählt. Zum anderen wird Individuum 100 rein zufällig gewählt. Dadurch besteht der Vorteil, dass das Individuum, trotz schlechter Fitness, ggf. eine gute Kombination hervorbringt.

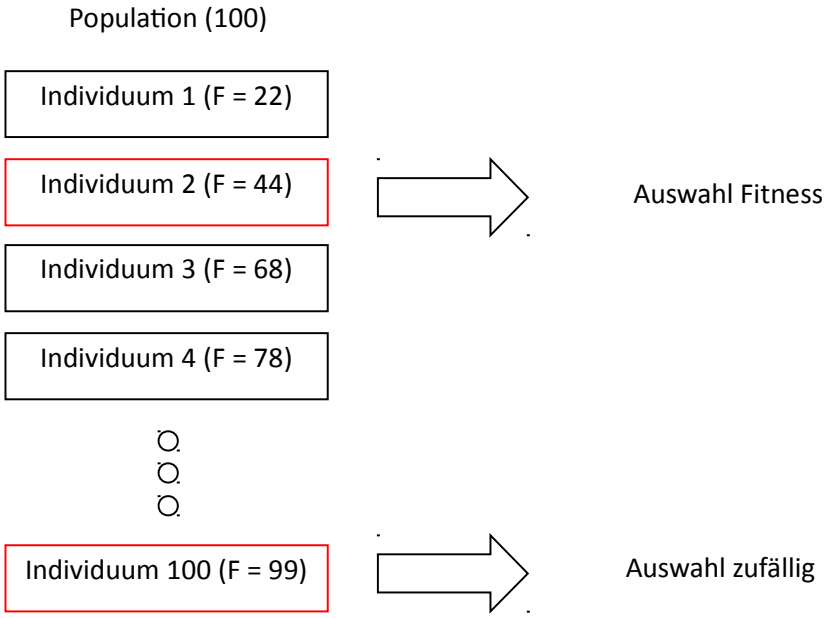


Abbildung 5:

2.5 Rekombination (Crossover)

Bei der Rekombination, auch als Crossover bezeichnet, handelt es sich um den Vorgang, aus zwei Elternpaaren neue Individuen zu erzeugen. Dieses Verfahren stellt dabei einen der wichtigsten Suchoperatoren für genetische Algorithmen dar. Dabei werden die Nachkommen aus den Informationen der gekreuzten Elternpaare systematisch zusammengesetzt. Gesteuert wird der Crossover durch eine Wahrscheinlichkeit, die bestimmt ob entweder nur einfache Kopien der Eltern erzeugt werden, oder ein Operator zur Erzeugung der Nachkommen verwendet wird. Dazu werden nachfolgend zwei verschiedene Varianten vorgestellt.

2.5.1 Crossover mit Control-String

Bei dieser Variante der Rekombination wird, auf Grundlage der gewählten Eltern, ein Bit-String gebildet, der letztendlich zu der Entscheidung führt, ob sich der Nachkomme aus einem Teil des ersten oder zweiten Elternteils zusammensetzt. Im Kontext der Fahrzeugeinsatzplanung soll dies die Tabelle 1 verdeutlichen:

Tabelle 1: Crossover mit Control-String

Fahrzeug 1 (Plan 1):	$1+ \rightarrow 4+ \rightarrow 1- \rightarrow 7+ \rightarrow 7- \rightarrow 4-$
Control-String:	1 0 0 0 1 1 0 1 0 1
Fahrzeug 1 (Plan 2):	$1+ \rightarrow 5+ \rightarrow 5- \rightarrow 1-$
Nachkomme:	$1+ \rightarrow 5+ \rightarrow 5- \rightarrow 1- \rightarrow 4+ \rightarrow 7+ \rightarrow 7- \rightarrow 4-$

Oben und unten ist jeweils eine Tour dargestellt, die jeweils einem Elternindividuum entnommen wurde. In der Mitte befindet sich der „Control-String“, wobei die Länge der Summe der Aufträge (Be- und Entladung) der beiden Eltern entspricht. Nun nimmt ein Element des Strings zufällig

eine Ausprägung von „1“ oder „0“ an, wobei die „1“ besagt, dass das Element aus Fahrzeug 1 von Plan 1 für den Nachkommen gewählt wird. Auf gleiche Weise geschieht dies mit der „0“ für Fahrzeug 1 aus Plan 2. Bereits gewählte Aufträge werden dabei nicht mehr berücksichtigt. Letztendlich entsteht der unten in der Tabelle 1 auf der vorherigen Seite aufgeführte Nachkomme.

2.5.2 Crossover nach Falkenauer

Die zweite Variante für einen möglichen Crossover orientiert sich an dem gruppenzentrierten Operator, der von Falkenauer vorgeschlagen wurde. Im Wesentlichen besteht er aus vier Phasen die nachfolgend in der Abbildung 6 dargestellt und erläutert werden.

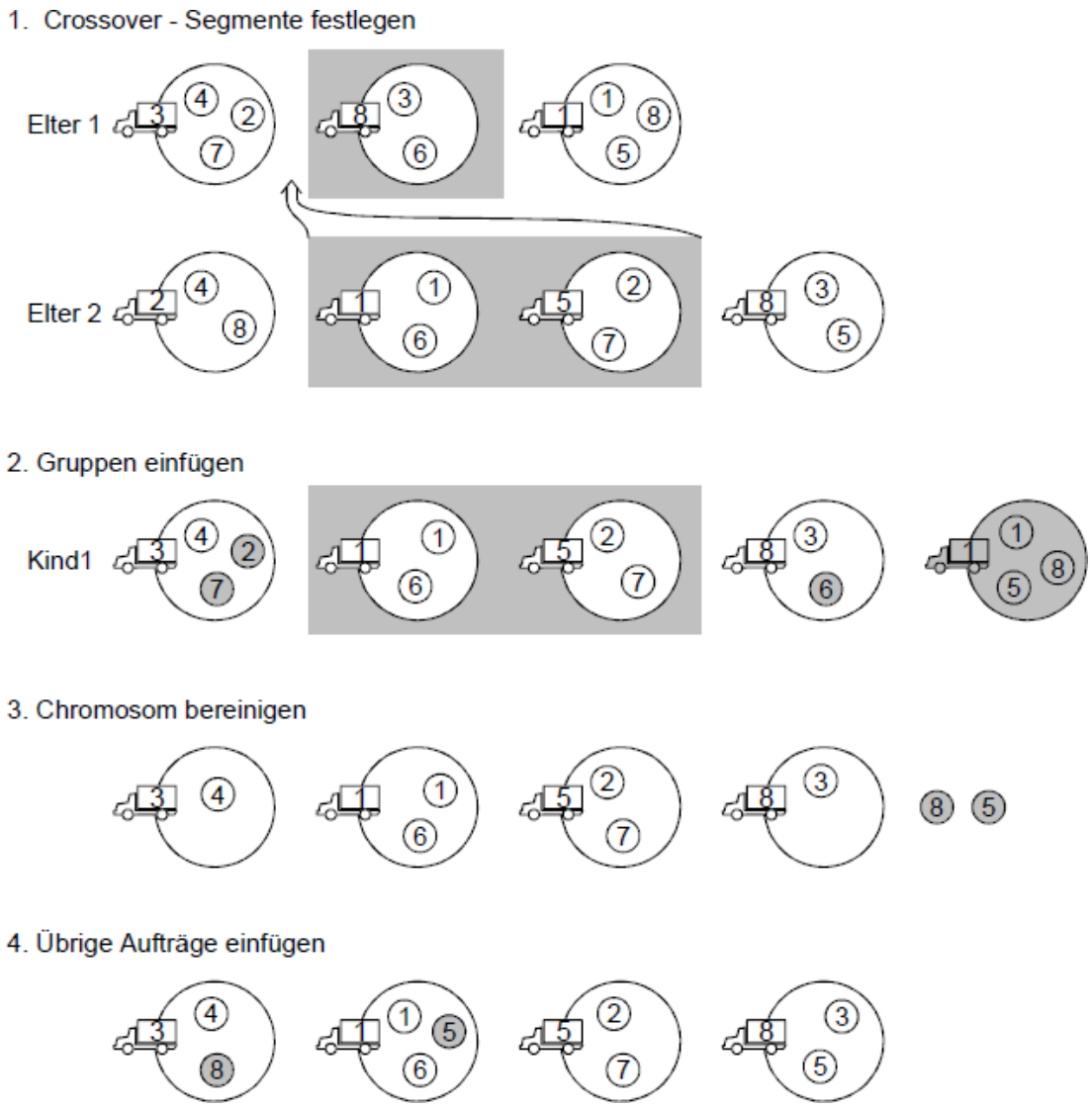


Abbildung 6:

1. In der ersten Phase werden in jedem der beiden Eltern zufällig je zwei Crossover-Punkte gebildet, die innerhalb von jedem Elternteil den Anfang und das Ende des Crossover-Segments darstellen.

2. Hier wird das Crossover-Segment des zweiten Elternteils an den ersten Crossover-Punkt des ersten Elternteils angefügt. Die dazugehörigen Touren werden vom Elternteil unmittelbar übernommen, also vorerst ohne Neukonstruktion. Als Folge kann eine Anzahl von Aufträgen mehreren Fahrzeugeinsatzplänen zugeordnet sein. Desweiteren kann es auch vorkommen, dass mehrere Touren vom gleichen Fahrzeug bedient werden.
3. In dieser Phase werden alle auftretenden Probleme aus Punkt 2 aufgelöst. Dabei wird so vorgegangen, dass vorerst alle neuerhaltenden Touren unangetastet bleiben und die Touren des empfangenden Elternteils modifiziert werden, um alle Konflikte zu lösen. Dazu werden alle Touren eliminiert, die auf ein Fahrzeug referenzieren, das neu hinzugefügt wurde (im Beispiel Fahrzeug 1). Falls anschließend noch einzelne Aufträge doppelt vorhanden sind, so werden auch diese aus den Touren des empfangenden Elternteils eliminiert (im Beispiel die Aufträge 2, 6 und 7). Nach Abschluss dieser Operation sind einzelne Aufträge (im Beispiel 5 und 8) noch vorerst keiner Tour zugeordnet.
4. Die verbleibenden Aufträge werden rein zufällig an die vorhandenen Touren vergeben. Dazu kann beispielsweise auch ein komplett neues Fahrzeug allokiert werden. Nach Abschluss von Schritt 4 liegt letztendlich als Ergebnis ein vollständiges Kind vor.

Gegebenenfalls können die Schritte 2-4 mit vertauschten Rollen der Eltern wiederholt werden, falls man ein zweites Kind erzeugen möchte.

2.6 Mutation

Unter der Mutation, im Zusammenhang zum Genetischen Algorithmus, versteht man die Art und Weise, um aus einem Elternteil ein neues Individuum durch zufällige Veränderung zu erzeugen. Diese Änderung wird dabei auch hier durch eine Wahrscheinlichkeit gesteuert, die angibt, ob eine Mutation an einem Individuum stattfindet oder nicht. Sie dienen hauptsächlich dazu, eine gewisse Inhomogenität und Divergenz mit in die Population herein zu bringen, was durch die Rekombination unter Umständen nicht möglich ist. Zusätzlich wird eine frühzeitige Konvergenz des Algorithmus verhindert und der Selektionsdruck abgeschwächt.

Innerhalb dieser Arbeit wurden insgesamt sieben verschiedene Mutationsoperatoren entwickelt, welche mit einer Gewichtung dem genetischen Algorithmus hinzugefügt werden können. Diese werden im nachfolgenden jeweils mit einer Abbildung vorgestellt.

2.6.1 Aufteilung der längsten Route

Dieser Operator überprüft die Länge von jeder Tour innerhalb einer Fahrzeugeinsatzplanung. Die Tour, welche die längste Strecke anhand der gefahrenen Kilometer aufweist (im Beispiel Fahrzeug 5), wird entfernt und auf die verbleibenden Touren aufgeteilt (siehe Abbildung 7 auf der nächsten Seite und Abbildung 8 auf der nächsten Seite). Dabei werden die Aufträge rein zufällig an die anderen Touren vergeben.

2.6.2 Aufteilung der kleinsten/zufälligen Tour

Diese beiden Mutationsoperatoren arbeiten auf die gleiche Weise wie unter Abschnitt 2.6.1 beschrieben. Dabei wird hier der mögliche Kandidat entweder zufällig oder anhand der kleinsten

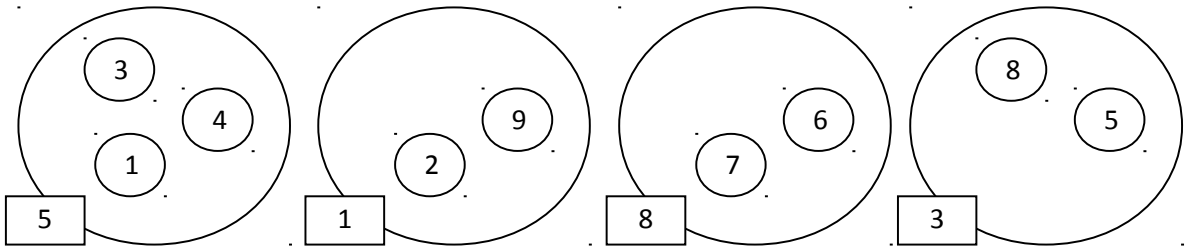


Abbildung 7: Ausgangsindividuum

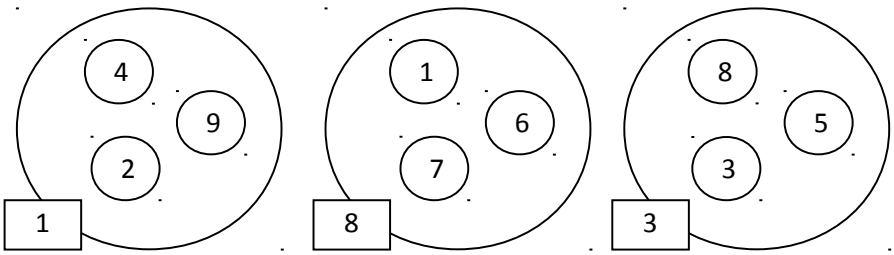


Abbildung 8: Neues Individuum

Tour gewählt. Da die Arbeitsweise nahezu identisch ist, wurde hier auf eine zusätzliche Abbildung verzichtet.

2.6.3 Verschiebung einer/mehrerer Aktion(en) innerhalb einer Tour

Dieser Operator geht noch einen Schritt weiter und verschiebt innerhalb einer Tour eine Aktion. Bei einer Aktion kann es sich um ein Auf- oder Abladen eines bestimmten Auftrags handeln, was über die assoziierte Datenstruktur zu einer Tour gewährleistet wird. (siehe Abbildung 9). Ein weiterer Operator verschiebt zusätzlich nicht nur eine Aktion, sondern ganze Subrouten, die in ihren Grenzen zufällig gewählt werden.

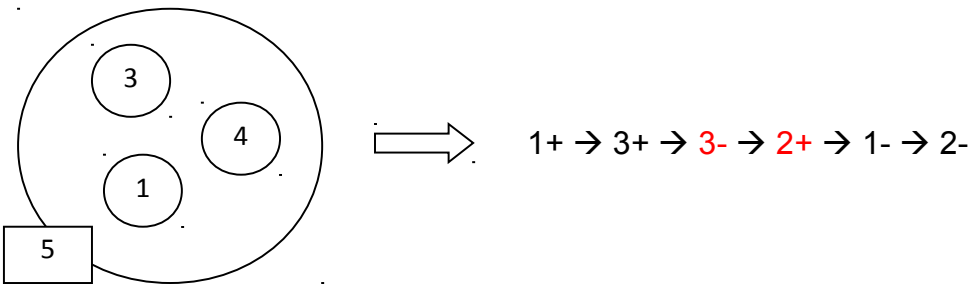


Abbildung 9: Verschiebung einer/mehrerer Aktion(en) innerhalb einer Tour

Die Repräsentation (links in Abbildung 9) bleibt davon unberührt, da es sich um eine reine Manipulation der Datenstruktur handelt. Konkret wird in diesem Beispiel das Abladen des Auftrags 3 mit dem Aufladen des Auftrags 2 vertauscht. Dies kann zur Folge habe, dass falsche Lösungen entstehen, da nach dem Operator keine Überprüfung und ggf. eine Korrektur stattfindet. Der

Umgang mit der genannten Problematik wird jedoch im Abschnitt 2.7 auf Seite 14 erläutert.

2.6.4 Verschiebung von Subtouren über eine Tour hinaus

Die bisherigen Operatoren haben lediglich ein Gen des Individuums mutiert. Um eine größere Varianz, unter dem Einfluss der Mutation, zu erschaffen wurde zusätzlich eine Möglichkeit entwickelt, wo ein Tauschen von Subtouren zwischen den unterschiedlichen Genen eines Individuums stattfindet. Dies veranschaulicht die Abbildung 10:

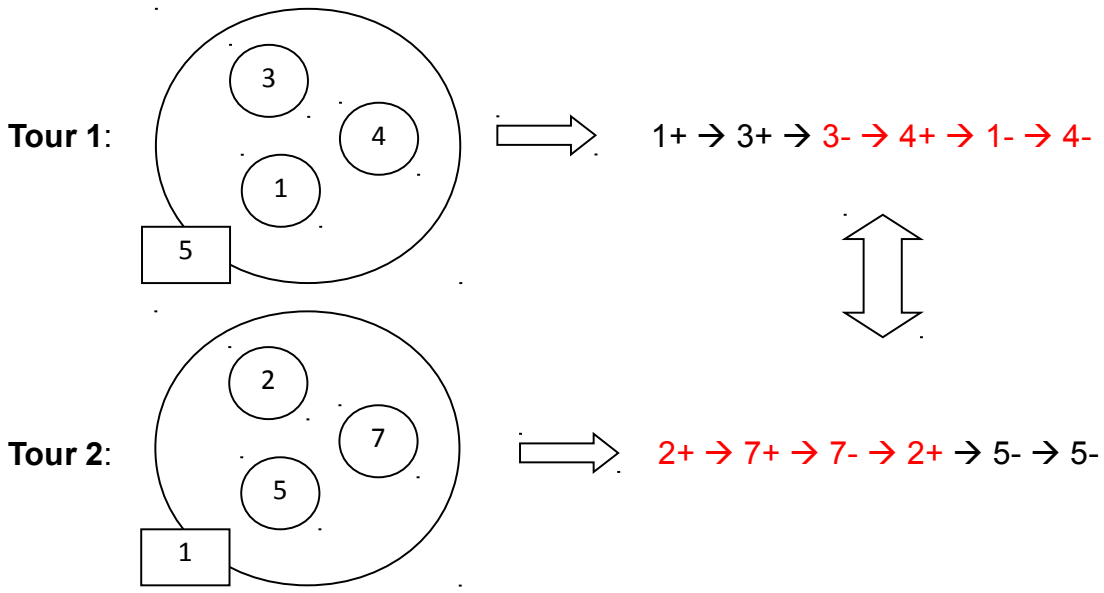


Abbildung 10: Verschiebung von Subtouren über eine Tour hinaus

Dadurch entstehen letztendlich zwei komplett neue Touren innerhalb einer Fahrzeugeinsatzplanung, wobei diese Mutation Einfluss auf die Repräsentationsebene hat, da beispielsweise die Order 2 und 7 von Tour 2 nun von der Tour 1 ausgeführt wird.

2.6.5 Kombination von 2 Touren

Den letzten genetischen Operator für eine Mutation stellt die Kombination von 2 Touren zu einer neuen Tour dar. Das Fahrzeug, das die Aufträge abliefert, wird zufällig aus den beiden vorhandenen gewählt. Dies wird durch die Abbildung 11 auf der nächsten Seite veranschaulicht:

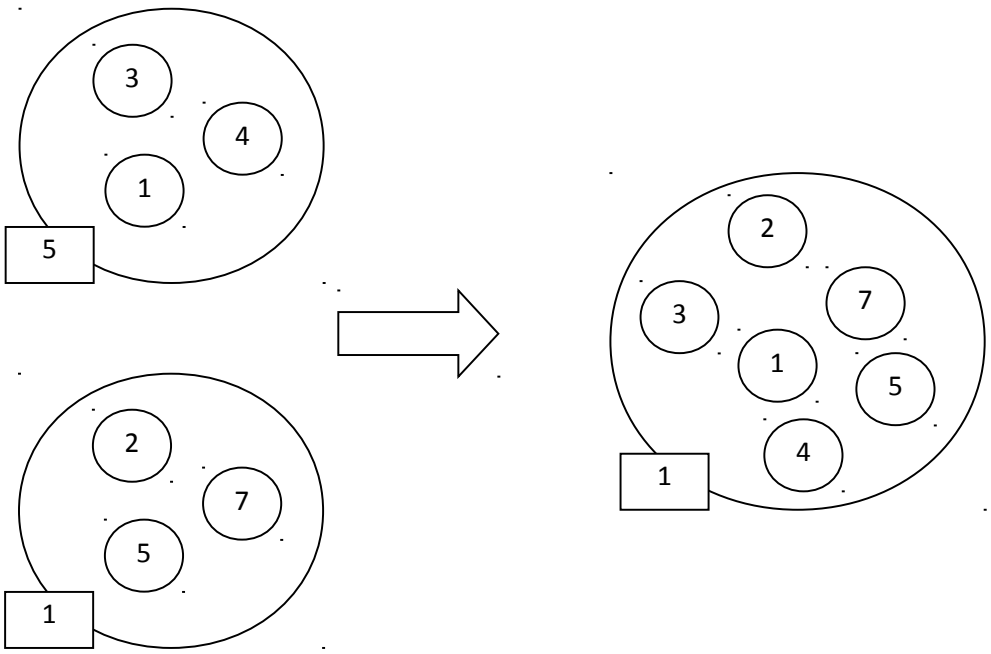


Abbildung 11: Verschiebung von Subtouren über eine Tour hinaus

2.7 Bildung der Nachfolgegeneration

Über die Mechanismen der Selektion hinaus, die in Abschnitt 2.4 auf Seite 7 definiert wurden, wird sowohl die komplette Ausgangspopulation, als auch die neue Population, für die Nachfolgepopulation berücksichtigt. Dazu findet auch an der Neuen wiederum eine Sortierung statt. Zum einen werden alle invaliden Lösungen, d.h. Lösungen, wo beispielsweise ein Ent- vor dem Beladen erfolgt, ans Ende der Population verschoben. Danach erfolgt eine Sortierung anhand der Fitness. Demnach werden beiden Populationen zu einer Nachfolgepopulation der definierten Menge (im Beispiel 100) zusammengefasst. Zur Veranschaulichung dient die Abbildung 12:

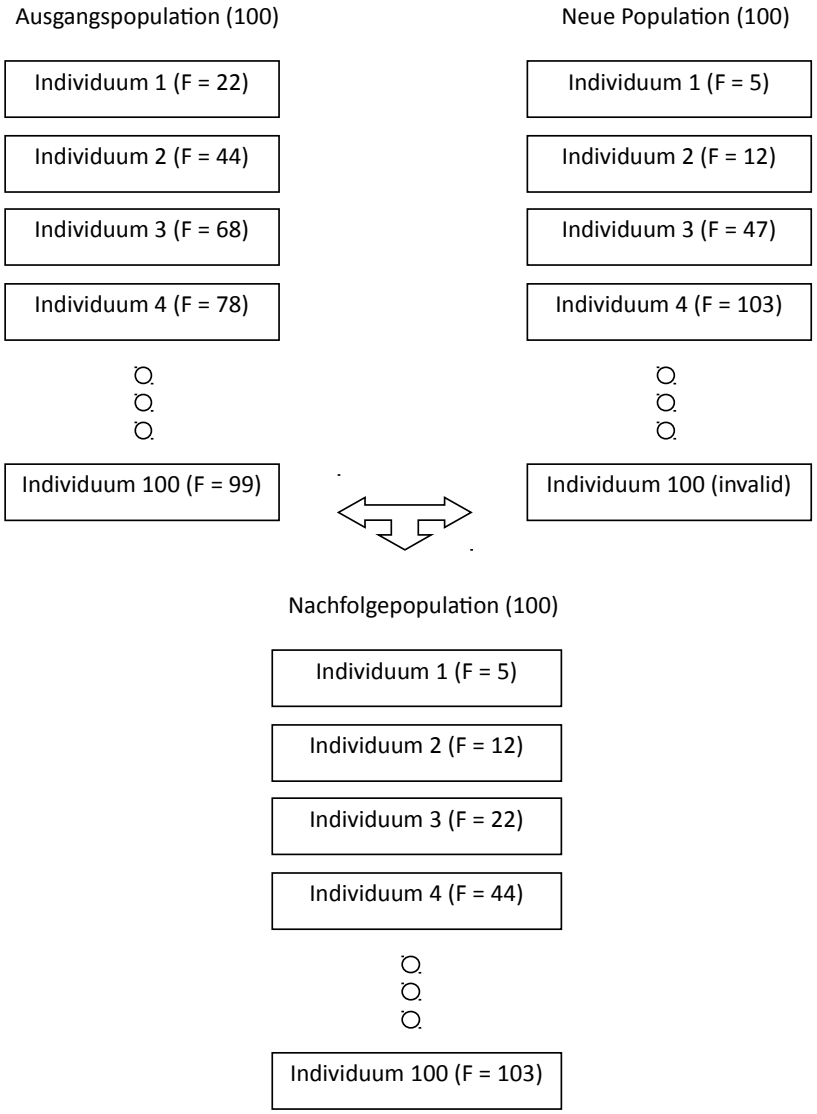


Abbildung 12: Bildung der Nachfolgegeneration

Das Resultat ist die neue Nachfolgepopulation, die wieder in den genetischen Algorithmus einfließt, bis eine Abbruchbedingung erreicht wird. Diese Verfahrenstechnik hat den Vorteil, dass bereits potentiell gute Lösungen, die in der Ausgangspopulation vorhanden waren, erhalten bleiben und nicht durch die neugebildete Population überschrieben werden.

2.8 Abbruchbedingung

Als letzten entscheidenden Punkt muss beim genetischen Algorithmus eine Abbruchbedingung definiert werden, wann die Suche abgeschlossen ist. Dazu wurden zwei Varianten entwickelt:

1. Der genetische Algorithmus wird solange durchlaufen, bis eine definierte Anzahl von Iterationen (z.B. 5000) erreicht wurde.
2. Die Fitness der besten Lösung entspricht einem definierten Abstand von der Durchschnittsfitness von allen Lösungen zu dieser Lösung

Die erste Bedingung wird für den Fall benötigt, dass der genetische Algorithmus nicht unendlich oft durchlaufen wird. Eine Iteration entspricht dabei dem Weg von einer Ausgangspopulation bis hin zu Neubildung der Nachfolgegeneration.

Die zweite Bedingung ist die eigentlich entscheidende, da der Algorithmus solange laufen soll, bis keine bessere Lösung mehr gefunden wird. Ist dies der Fall, so kann die genetische Suche abgeschlossen werden und das erste Individuum in der Population entspricht der optimalen Lösung für eine gegebene Konfiguration.

3 Softwarearchitektur

3.1 Systemdesign

Das Systemdesign setzt sich im wesentlichen aus den folgenden vier Modulen zusammen:

1. Konfigurationsgenerierung
2. Konstruktionsverfahren
3. Genetisches Optimierungsverfahren
4. Ergebnisvisualisierung

Desweiteren sind zwei Datenmodelle definiert:

1. Konfiguration-Datenmodell
2. Ergebnis-Datenmodell

3.2 Konfigurations-Datenmodell

Das Konfiguration-Datenmodell beschreibt die Konfiguration, die vom Anwender zur Verfügung gestellt wird. Sie umfasst Stationen, Fahrzeuge, Produkte, Aufträge und Zeitfenster (siehe Abbildung 14 auf der nächsten Seite).

3.3 Ergebnis-Datenmodell

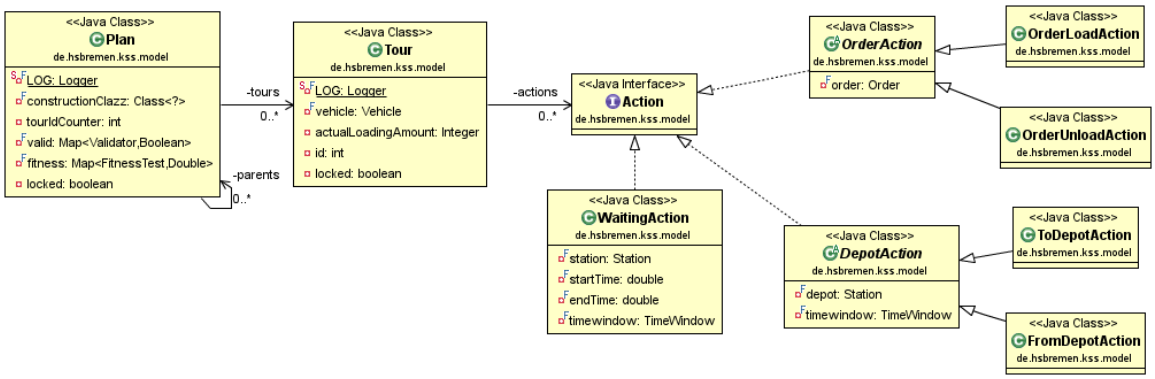


Abbildung 13: Repräsentation eines Plans mit Touren und Aktionen

3.4 Konfigurationsgenerierung

3.5 Genetischer Algorithmus

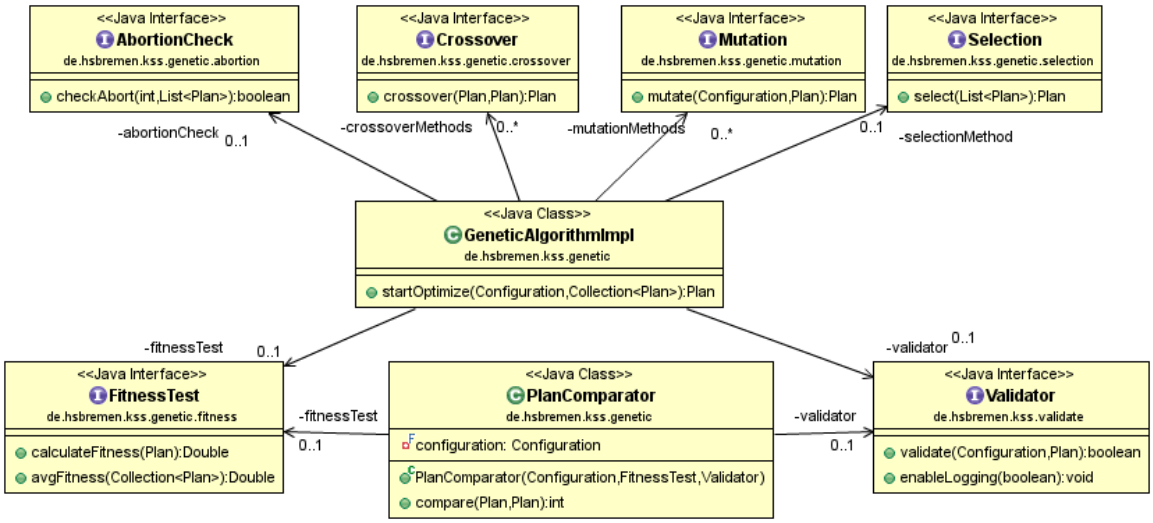


Abbildung 14: Klassendiagramm vom genetischen Algorithmus

4 Umsetzung

4.1 Verwendete Bibliotheken

Im Rahmen der Entwicklung wurden diverse OpenSource - Bibliotheken eingeführt, um die Entwicklung zu beschleunigen und die Qualität der Software zu steigern. Im folgenden werden die verwendeten Bibliotheken beschrieben:

Apache Commons Math (Version 3.2, Apache License 2.0) [11] Stellt mathematische Funktionen zur Verfügung. Wird unteranderem für die Berechnung der Entfernungen verwendet.

Apache Commons Lang (Version 3.3.1, Apache License 2.0) [10]

Apache Commons Collections (Version 4.0, Apache License 2.0) [9] Erweitert das Java-Collections-Framework.

Joda-Time (Version 2.3, Apache License 2.0) [4]

JUnit (Version 4.11, Eclipse Public License) [5] JUnit ist die Standard-Bibliothek für Unit-Tests unter Java.

Hamcrest (Version 1.3, BSD 3-Clause) [3] Mit Hamcrest ist es möglich bei Tests „sprechendere“ Ausdrücke zu formulieren.

EasyMock (Version 3.2, Apache License 2.0) [1] EasyMock ermöglicht ein einfaches Erstellen von Mock-Objekten für den Unit-Test.

Simple Logging Facade for Java (SL4J) (Version 1.7.6, MIT license) [8] SLF4J ist eine Logging-Schnittstelle und wird zur Ausgabe auf der Konsole verwendet.

Logback (Version 1.1.1, Eclipse Public License v1.0 und LGPL 2.1) [7] Wird als Implementierung für SLF4J eingesetzt.

Google Guave (Version 17.0, Apache License 2.0) [2] Guave ist eine Sammlung von Softwarebibliotheken. Es wird ausschließlich der EventBus für die Kommunikation zwischen den Softwarekomponenten verwendet.

JFreeChart (Version 1.0.17, LPGL) [6] JFreeChart wird zur Darstellung der Graphen verwendet.

4.2 Mutation

4.3 Grafische Oberfläche

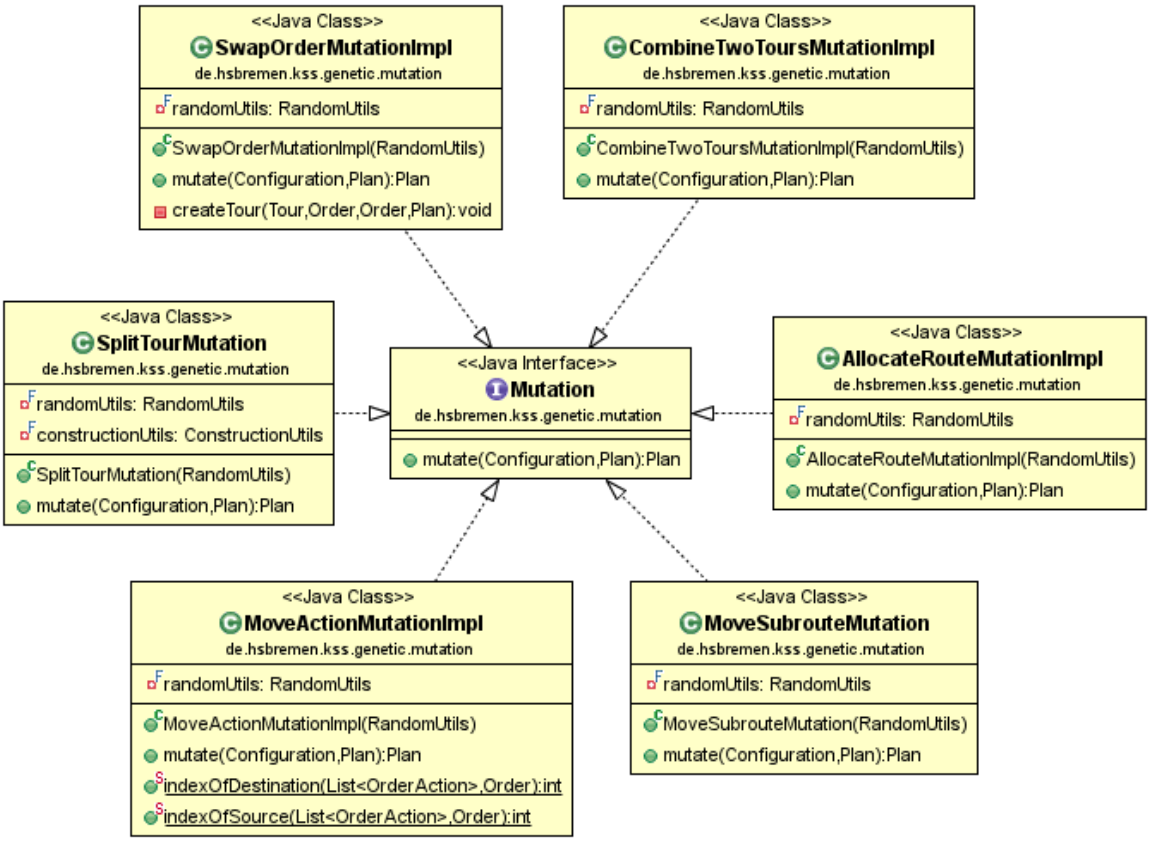


Abbildung 15: Implementierte Mutationsalgorithmen

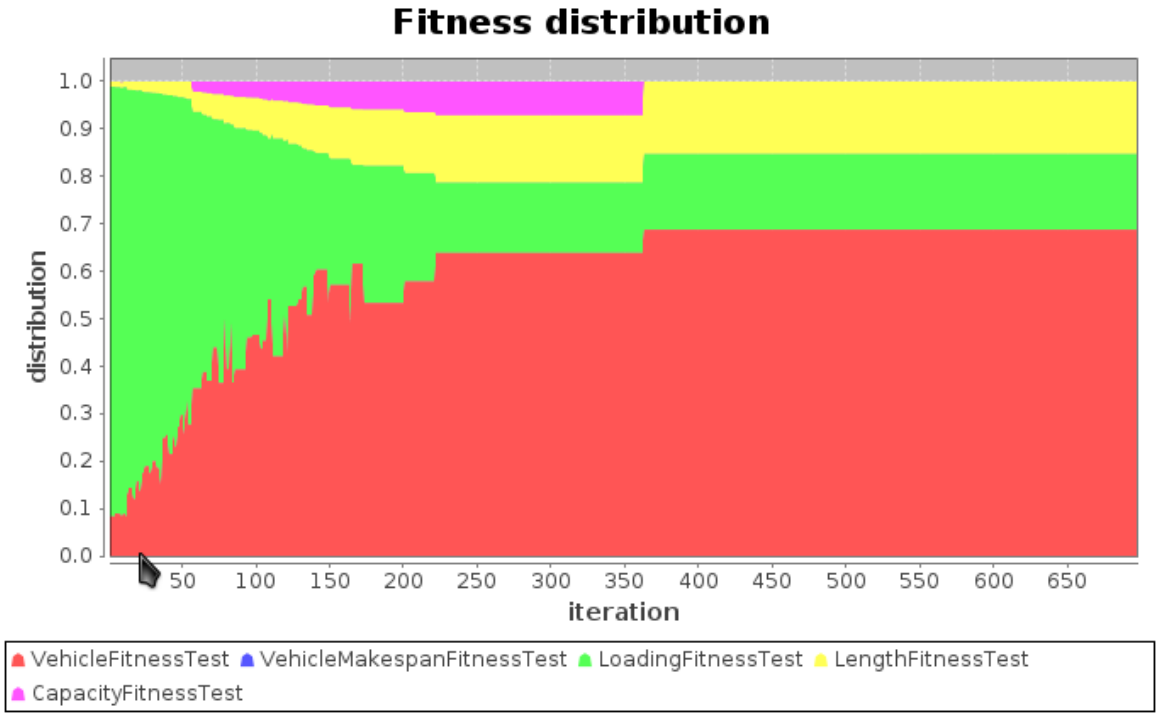


Abbildung 16: Verteilung des Fitnesswertes

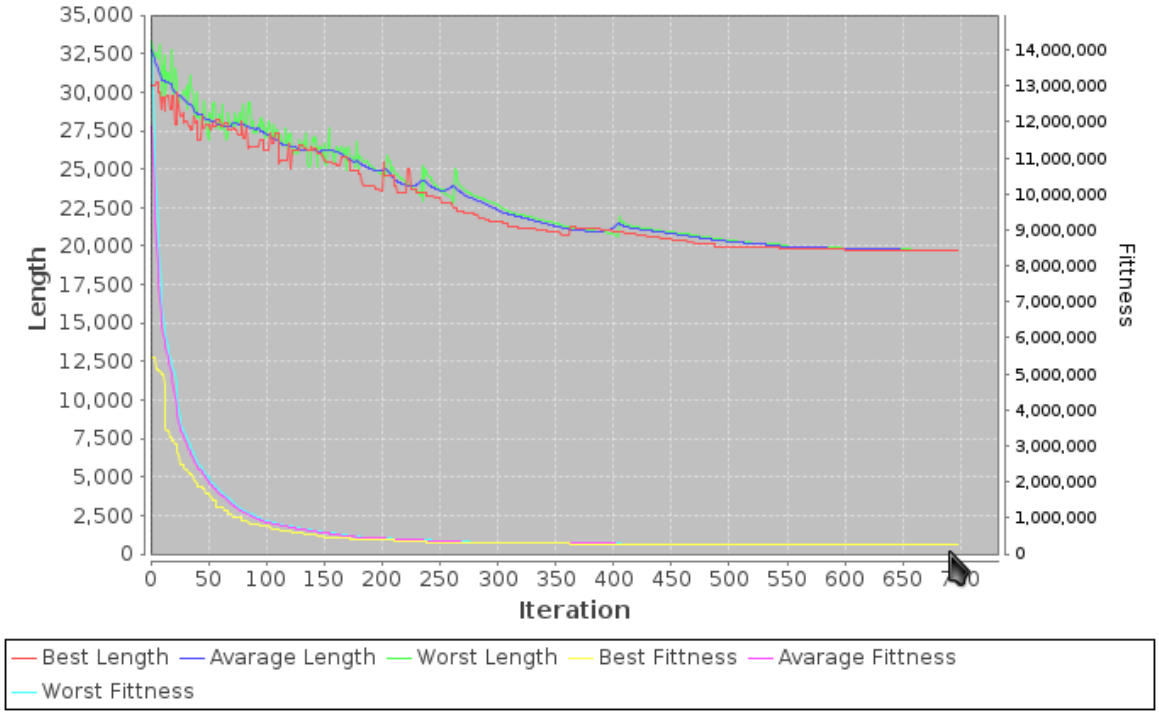


Abbildung 17:

Abbildung 18:

5 Experimente

6 Ausblick

7 Fazit

Literatur

[1] EasyMock contributors: *EasyMock*. http://easymock.org/. Version: 2014. – [Online; 9. Juli 2014]

[2] Google: *Guave*. https://code.google.com/p/guava-libraries/. Version: 2014. – [Online; 9. Juli 2014]

[3] hamcrest.org: *Hamcrest*. http://hamcrest.org/. Version: 2014. – [Online; 9. Juli 2014]

[4] Joda: *Joda-Time - Java date and time API*. http://www.joda.org/joda-time/. Version: 2014. – [Online; 9. Juli 2014]

[5] JUnit: *JUnit*. http://junit.org/. Version: 2014. – [Online; 9. Juli 2014]

[6] Object Refinery Limited: *JFreeChart*. http://www.jfree.org/jfreechart/. Version: 2014. – [Online; 9. Juli 2014]

[7] QOS.ch: *Logback Project*. http://logback.qos.ch/. Version: 2014. – [Online; 9. Juli 2014]

[8] QOS.ch: *Simple Logging Facade for Java (SLF4J)*. http://http://www.slf4j.org/. Version: 2014. – [Online; 9. Juli 2014]

[9] The Apache Software Foundation: *Commons Collections*. http://commons.apache.org/proper/commons-collections/. Version: 2014. – [Online; 9. Juli 2014]

[10] The Apache Software Foundation: *Commons Lang*. http://commons.apache.org/proper/commons-lang/. Version: 2014. – [Online; 9. Juli 2014]

[11] The Apache Software Foundation: *Commons Math: The Apache Commons Mathematics Library*. http://commons.apache.org/proper/commons-math/. Version: 2014. – [Online; 9. Juli 2014]