Wilhelm Büchner Hochschule
Fachbereich Informatik

Bachelorarbeit

Dienstplanerstellung in Unternehmen mittels genetischer Algorithmen

vorgelegt bei	Prof. Dr. Thomas Freytag
von	Stefan Specht
Matrikelnummer	888809
Anschrift	Triebstraße 47, 60388 Frankfurt
Abgabetermin	01.10.2016

Abstract

Die vorliegende Bachelorarbeit untersucht den Einsatz genetischer Algorithmen zur automatisierten Dienstplanerstellung in Unternehmen. Das Problem der Dienstplanerstellung wurde dazu am Beispiel eines konkreten Unternehmens analysiert und als Optimierungsmodell formuliert. Neben dem Optimierungsziel wurden dabei auch die einzuhaltenden Nebenbedingungen definiert.

Beim Entwurf des genetischen Algorithmus zur Lösung des Optimierungsproblems spielte sowohl der Umgang mit den Nebenbedingungen als auch die problemspezifische Anpassung von Codierung und Rekombinationsoperator eine zentrale Rolle. Die Einhaltung der Nebenbedingungen konnte überwiegend durch die gewählte Codierung und die Entwicklung eines speziellen Initialisierungsalgorithmus und Mutationsoperators sichergestellt werden. Eine der Nebenbedingungen musste in Form von Strafkosten in die Bewertungsfunktion integriert werden. Aufbauend auf theoretischen Vorüberlegungen wurden die Codierung und der Rekombinationsoperator so entworfen, dass der Algorithmus bestmöglich bei einer effizienten Verarbeitung von Building Blocks unterstützt wird.

Die experimentelle Evaluierung ergab, dass durch die Wahl einer hohen Populationsgröße und durch den Einsatz eines speziellen Rekombinationsoperators, der die Gültigkeit von Lösungen bewahrt, der genetische Algorithmus in Bezug auf die erzielte Güte der Dienstpläne noch deutlich verbessert werden kann. Der so angepasste Algorithmus ist in der Lage, mit einer sehr hohen Wahrscheinlichkeit optimale Lösungen bei gleichzeitig geringer Laufzeit zu erzeugen.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

In vielen Unternehmen werden Dienstleistungen rund um die Uhr erbracht. Klassische Beispiele hierfür sind u. a. die Betreuung von Patienten in Krankenhäusern und Telefonsupport in Call-Centern. In Zeiten der Globalisierung und Digitalisierung gilt auch für viele neue informationsbasierte Dienstleister der Leitgedanke: „Rund um die Uhr und rund um den Globus“[1]. Nur so können diese Unternehmen in der globalen Dienstleistungswirtschaft bestehen.

Um diese Dienstleistungen jederzeit in der gewünschten Qualität erbringen zu können, muss über die Personalplanung des Unternehmens sichergestellt werden, dass zu jedem Zeitpunkt die notwendige Anzahl entsprechend qualifizierter Mitarbeiter zur Ausübung der anfallenden Aufgaben zur Verfügung steht. Die Personalplanung besteht dabei aus mehreren Teilaspekten, u. a. aus der Personalbedarfsplanung und der Personaleinsatzplanung[2].

Während in der Personalbedarfsplanung, als Vorbereitung der Personalbeschaffung, der quantitative und qualitative Bedarf an Mitarbeitern im Unternehmen ermittelt wird[3], ist es Aufgabe der Personaleinsatzplanung, die vorhandenen Mitarbeiter zu Allokationsobjekten, wie z. B. Stellen oder Tätigkeiten, zuzuordnen[4]. Ein zentrales Problem, bei dem die temporale Dimension der Personaleinsatzplanung im Fokus steht, ist die Zuordnung von Mitarbeitern zu Dienstplänen. Dieser Vorgang wird auch als Dienstplanung bezeichnet[5].

An Dienstpläne werden neben obligatorischen Anforderungen, wie die Einhaltung gesetzlicher Bestimmungen oder die Vermeidung personeller Über- und Unterbesetzung, i. d. R. noch weitere Forderungen gestellt, deren Erfüllung wünschenswert ist. Dies können Vorgaben des Unternehmens oder der Abteilung sein, wie z. B. die Vermeidung bestimmter Schicht-Muster, oder auch individuelle Präferenzen der Mitarbeiter, wie z. B. Urlaubswünsche für bestimmte Tage.

Die große Anzahl möglicher Bedingungen, die darüber hinaus miteinander in Widerspruch stehen können, bedeutet, dass die Dienstplanerstellung keinesfalls

[1] Bullinger 2013, S. 167.
[2] Vgl. Bartscher/Wichert 2016.
[3] Vgl. Bartscher 2016.
[4] Vgl. Spengler 2011, S. 569.
[5] Vgl. Spengler 2006, S. 15–16.

eine triviale Aufgabe darstellt. Vielmehr kann sie als ein schwieriges kombinatorisches Optimierungsproblem aufgefasst werden und ist somit dem Operations Research zuzuordnen.

Die manuelle Dienstplanerstellung durch den Team- oder Bereichsleiter erfordert daher ein hohes Maß an Erfahrung und ist mit großem Zeitaufwand verbunden. Die Qualität dieser Dienstpläne kann darüber hinaus nicht genau bestimmt werden und ist somit nicht transparent. Auch die Gefahr der Bevorzugung einzelner Mitarbeiter ist gegeben.

Automatisierte Ansätze bergen großes Potenzial, diesen Prozess der Dienstplanerstellung und die Qualität der resultierenden Dienstpläne zu verbessern[6]. Mathematische und heuristische Ansätze können auf einfache Weise mehrere Lösungen generieren, Rückmeldung über die Qualität der erzeugten Pläne geben und dabei versuchen, die Arbeitslast gleichmäßig auf die Mitarbeiter zu verteilen[7]. Einer der größten Vorteile der automatisierten Dienstplanerstellung ist jedoch die beachtliche Zeitersparnis für die Verwaltungsmitarbeiter[8].

Diese Arbeit beschäftigt sich mit der automatisierten Dienstplanerstellung in Unternehmen am Beispiel der globalen IT-Organisation der Evonik Industries AG. Die Abteilungen, die den internen Kunden des Unternehmens konkrete IT-Dienste anbieten, verfügen jeweils über ein operatives Team, das für die Bereitstellung und Aufrechterhaltung dieser Dienste verantwortlich ist. Dies geschieht rund um die Uhr in einer speziellen Form des Schichtbetriebs nach dem „Follow-the-Sun“-Prinzip (drei Schichten, verteilt auf drei Zeitzonen).

In jeder dieser Schichten gibt es eine Reihe von Aufgaben, die jeweils von einer bestimmten Anzahl an Mitarbeitern besetzt sein müssen. Voraussetzung für eine gültige Zuordnung ist, dass der Mitarbeiter sowohl verfügbar ist, als auch über die nötige Qualifikation verfügt, die Aufgabe auszuführen. Eine Besonderheit ist dabei, dass in einer Schicht bestimmte zulässige Kombinationen von Aufgaben auch von einem einzelnen Mitarbeiter abgedeckt werden können (1:n-Zuordnung).

Aufgrund der bereits angesprochenen Komplexität des Problems und der damit verbundenen Schwierigkeit der manuellen Dienstplanerstellung werden in den operativen Teams derzeit nur rudimentäre Dienstpläne vom jeweilig zuständigen Teamleiter erstellt, d. h. es werden nur wenige, als besonders wichtig erachtete Aufgaben betrachtet. Da Regelungen für die Besetzung der übrigen

[6] Vgl. Burke u. a. 2004, S. 442.
[7] Vgl. ebd., S. 442.
[8] Vgl. ebd., S. 442.

Aufgaben fehlen, ist es bisher nicht selten, dass keiner der Mitarbeiter mit der erforderlichen Qualifikation anwesend ist und somit Kundenanfragen oder Störungen nicht umgehend bearbeitet werden können. Dies soll künftig durch die computergestützte Generierung qualitativ hochwertiger Dienstpläne verbessert werden.

1.2. Ziele

In anderen wissenschaftlichen Arbeiten wurden bereits viele gängige Lösungsverfahren des Operations Research auf verschiedenartige Probleminstanzen der Dienstplanerstellung angewandt und untersucht. Gute Ergebnisse konnten dabei u. a. durch die Anwendung sog. Meta-Heuristiken erzielt werden, insbesondere mittels genetischer Algorithmen und Tabu-Suche[9,10,11,12]. Keines der bisher formulierten Problemmodelle lässt sich jedoch unverändert auf die Probleminstanzen der Evonik Industries AG anwenden.

Ziel dieser Arbeit ist es, bereits existierende, ähnliche Problemmodelle so zu erweitern, dass damit das zugrunde liegende Problem vollständig formuliert werden kann. Für das so formulierte Problem soll dann, basierend auf aktuellen Erkenntnissen und Empfehlungen der wissenschaftlichen Literatur, ein genetischer Algorithmus entworfen und in einer prototypischen Anwendung implementiert werden.

Dieser Algorithmus soll nach einer ersten Evaluierung weiter verbessert werden, indem experimentell möglichst gute Parametereinstellungen ermittelt werden und die bisher eingesetzten Verfahren und Operatoren angepasst und optimiert werden.

Ein Vergleich mit weiteren Lösungsverfahren kann im Rahmen dieser Bachelorarbeit nicht angestellt werden.

1.3. Aufbau der Arbeit

Diese Arbeit gliedert sich in drei Teilbereiche.

[9] Vgl. Burke/Causmaecker/Vanden Berghe 1999.
[10] Vgl. Burke u. a. 2001.
[11] Vgl. Adamuthe/Bichkar 2011.
[12] Vgl. Aickelin 1999.

Zur Einführung in die Thematik werden im *ersten Teil* der Arbeit die theoretischen Grundlagen genetischer Algorithmen ausführlich dargestellt und erläutert. In Abschnitt 2 werden dazu zunächst die prinzipiellen Mechanismen und Begriffe der biologischen Evolution dargelegt, um sie anschließend auf die Optimierung mittels genetischer Algorithmen zu übertragen und deren grundlegenden Ablauf zu skizzieren. Einer theoretischen Betrachtung der inneren Funktionsweise folgt dann die vertiefende Beleuchtung der einzelnen Komponenten genetischer Algorithmen. Dies liefert bereits wertvolle Hinweise für den späteren Entwurf.

Im *zweiten Teil* der Arbeit wird auf die gewonnenen Erkenntnisse des ersten Teils aufgebaut, um einen genetischen Algorithmus zu entwerfen, der das beschriebene Optimierungsproblem effizient lösen kann. Dazu wird in Abschnitt 3 das Problem der Dienstplanerstellung zunächst einer umfassenden Analyse unterzogen und aus den Ergebnissen ein mathematisches Optimierungsmodell abgeleitet. In Abschnitt 4 erfolgt dann der eigentliche Entwurf des genetischen Algorithmus.

Im *dritten Teil* der Arbeit wird dieser Algorithmus schließlich evaluiert und optimiert. Die im zweiten Teil gewählten Methoden und Parameterwerte werden in Abschnitt 5 experimentell überprüft und entsprechend der Ergebnisse angepasst.

2. Einführung in genetische Algorithmen

2.1. Biologischer Hintergrund und Terminologie

Genetische Algorithmen sind naturanaloge Suchalgorithmen, die auf den Mechanismen natürlicher Selektion und der Genetik beruhen[13] und stellen eine klassische Variante evolutionärer Algorithmen dar. Allen Ausprägungen evolutionärer Algorithmen ist gemein, dass sie Vorgänge und Begriffe aus der Biologie entlehnen, um in einer simulierten Evolution für ein nahezu beliebig wählbares Optimierungsproblem möglichst gute Näherungswerte an eine exakte Lösung zu erzeugen[14].

Um die Arbeitsweise genetischer Algorithmen zu verstehen, werden zunächst die grundlegenden Begriffe und Mechanismen der Evolution in der Natur er-

[13] Vgl. Goldberg 1989, S. 1.
[14] Vgl. Weicker 2015, S. 1 und 20.

läutert.

Ein einzelnes Lebewesen wird in der Biologie meist als *Individuum* bezeichnet[15], während man eine Menge mehrerer Individuen eine *Population* nennt[16]. Komplexe biologische Individuen, wie Pflanzen oder Tiere, sind aus sog. eukaryotischen Zellen aufgebaut. Diese Zellen beinhalten jeweils einen Zellkern, in dem sich die Träger der Erbinformationen des Individuums, die *Chromosomen*, befinden. In diesen Chromosomen ist der „Bauplan“ zur Konstruktion des Erscheinungsbildes des Individuums codiert[17]. Jedes Einzelmerkmal dieses Plans, wie z. B. die Augenfarbe, ist durch einen Teilabschnitt des Chromosoms, der als *Gen* bezeichnet wird, festgelegt. Die verschiedenen Möglichkeiten der Ausprägung eines bestimmten Gens werden *Allele* genannt[18]. Am Beispiel des Gens, das die Augenfarbe bestimmt, sind die Allele also die Menge aller Augenfarben, wie z. B. „blau“ oder „braun“. Die Summe aller in den Genen durch die Allele festgelegten Erbinformationen heißt *Genotyp*[19,20]. Das durch den Genotyp bestimmte Erscheinungsbild des Individuums, also z. B. die Pflanze oder das Tier selbst, wie es von uns wahrgenommen wird, wird als *Phänotyp* bezeichnet[21,22].

Evolution folgt dem darwinistischen Prinzip „Survival of the fittest“. Dieses Prinzip besagt, dass diejenigen Lebewesen überleben, die am besten an ihre Umwelt angepasst sind[23]. Aufgrund ihrer höheren Lebensdauer können diese Lebewesen vermehrt Nachkommen erzeugen, wodurch sich ihr Genmaterial bevorzugt durchsetzt und insgesamt zu einer Optimierung der Lebewesen führt[24].

Die Evolution verändert die Häufigkeit von Merkmalen in einer Population also dahingehend, dass deren Individuen im Laufe der Zeit immer besser an die Umwelt angepasst sind, d. h. eine höhere Fitness erreichen. Während die Beurteilung der Fitness eines Individuums nur auf phänotypischer Ebene erfolgen kann, läuft die Veränderung der Individuen auf der Ebene des Genotyps ab. Diese Veränderung, die im Spannungsfeld zwischen Konservation und Diversifikation der Erbinformation operiert, wird durch drei grundlegende Mechanismen erreicht:

[15] Vgl. Weicker 2015, S. 8.
[16] Vgl. Heistermann 1994, S. 21.
[17] Vgl. Goldberg 1989, S. 21.
[18] Vgl. Weicker 2015, S. 8.
[19] Vgl. Heistermann 1994, S. 16.
[20] Vgl. Goldberg 1989, S. 21.
[21] Vgl. Heistermann 1994, S. 16.
[22] Vgl. Goldberg 1989, S. 21.
[23] Vgl. Heistermann 1994, S. 12.
[24] Vgl. ebd., S. 12.

Selektion Je besser der Phänotyp eines Individuums an seine Umgebung angepasst ist, desto höher sind dessen Chancen, in dieser Umgebung zu überleben. Dies erhöht sogleich die Wahrscheinlichkeit einer hohen Fortpflanzungsrate und somit auf eine (mehrfache) Weitergabe des bewährten Erbguts. Bei der Selektion handelt es sich demnach um eine „Veränderung der Allelenhäufigkeit durch unterschiedlich viele Nachkommen der einzelnen Allele“[25]. Sie gibt die Richtung der Veränderung an[26].

Mutation Mutation bezeichnet das Auftreten möglicher Vervielfältigungsfehler bei der Reproduktion und ist die Grundlage für Veränderung in der Evolution[27]. Sie wirkt dem durch die Selektion bedingten Diversitätsverlust entgegen, indem sie mit einer geringen Wahrscheinlichkeit neue oder verlorengegangene Allele einführt. Diese kleinen, zufälligen Veränderungen des Erbguts können sich sowohl positiv als auch negativ auf die Fitness des resultierenden Phänotyps des Individuums auswirken.

Rekombination Die Rekombination findet bei der geschlechtlichen Fortpflanzung zweier Individuen statt, wodurch das genetische Material der Eltern neu kombiniert wird. Es werden also keine neuen Allele eingeführt, sondern lediglich vorhandene neu zusammengesetzt[28]. Durch die Durchmischung des Erbguts sollen so Kinder mit einer insgesamt höheren Fitness als die der Eltern entstehen.

2.2. Ablauf genetischer Algorithmen

Die Prinzipien und Mechanismen der Evolution können nun auf die Optimierung mittels genetischer Algorithmen übertragen werden. Dabei ist eine abstraktere Betrachtungsweise auf die beschriebenen biologischen Vorgänge nötig.

Die Evolution wird hierzu als Suchprozess aufgefasst, dessen Suchraum durch alle möglichen Allel-Kombinationen eines Individuums definiert ist. Das Ziel der Suche besteht darin, eine Genbelegung zu finden, sodass die dadurch repräsentierte Lösung möglichst gut an die Umwelt, also das gegebene Optimierungsproblem, angepasst ist. Die in der Biologie schwer fassbare Umwelt wird

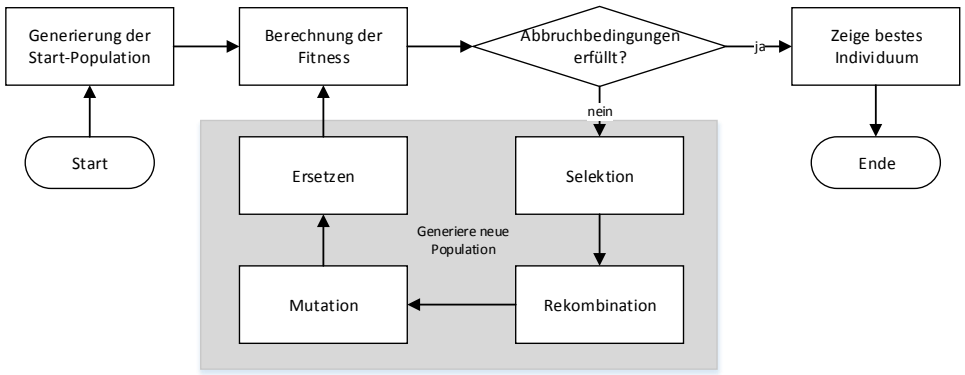
[25] Weicker 2015, S. 10.
[26] Vgl. Buttelmann/Lohmann 2004, S. 152.
[27] Vgl. Weicker 2015, S. 10.
[28] Vgl. ebd., S. 10.

im Kontext genetischer Algorithmen durch eine klar definierte Bewertungsfunktion[29] ersetzt[30].

Zur Simulation des Evolutionsprozesses in einem Computer werden die Individuen, die Punkte im Suchraum darstellen, analog zum biologischen Vorbild als Chromosomen codiert. Die Darstellung eines Chromosoms mit l Genen erfolgt dabei als Zeichenkette der Länge l, wobei jede Komponente der Zeichenkette ein Gen repräsentiert.

Wie in der Natur erfolgt die Suche nach immer besseren Lösungen innerhalb des Suchraums populationsbasiert[31]. Somit wird „das Wechselspiel von Variation und Selektion gemäß Darwin [...] als sequentielles Abarbeiten einzelner Phasen auf einer Population interpretiert“[32].

Der grundlegende Ablauf eines kanonischen genetischen Algorithmus ist in Abbildung 1 schematisch dargestellt und kann wie folgt skizziert werden:



Quelle: Eigene Darstellung in Anlehnung an Buttelmann/Lohmann 2004

Abbildung 1: Ablaufplan des kanonischen genetischen Algorithmus

Der Algorithmus startet mit der Initialisierungsphase. In dieser Phase wird eine Anfangspopulation $G(0)$ der Größe n erzeugt. Die Belegung der Gene ist dabei meist zufällig gewählt.

In jeder Iteration des nachfolgenden evolutionären Zyklus wird aus einer Po-

[29] In der Literatur auch als „Fitnessfunktion“ oder „Objektfunktion“ bezeichnet.
[30] Vgl. Weicker 2015, S. 24.
[31] Vgl. Heistermann 1994, S. 21.
[32] Weicker 2015, S. 24.

pulation $G(t)$ eine neue Population $G(t+1)$ erzeugt[33]. Dazu wird zunächst der Phänotyp jedes Individuums der aktuellen Population bewertet und ein entsprechender Fitnesswert berechnet. Anschließend werden die aus der Natur bekannten Phasen Selektion, Rekombination und Mutation durchlaufen.

Die Selektion nutzt den Fitnesswert, um für jedes Individuum festzulegen, wie viele Kindindividuen daraus erzeugt werden sollen[34]. Individuen mit einer höheren Fitness werden dabei statistisch häufiger für die nachfolgenden Reproduktionsschritte herangezogen.

Auf diese Weise werden $\frac{n}{2}$-mal jeweils zwei Eltern ausgewählt, die dann durch Rekombination zwei Kindindividuen erzeugen. Diese werden anschließend mit einer geringen Wahrscheinlichkeit mutiert. Analog zur Biologie hat die Rekombination dabei die Aufgabe, vorhandene Merkmale der Population zu durchmischen und neu zu kombinieren, während die Mutation potenziell neue oder verlorene Merkmale einführt.

In der letzten Phase des Zyklus werden schließlich Individuen der aktuellen Population $G(t)$ gemäß eines Ersetzungsschemas durch die erzeugten Kindindividuen ersetzt und so $G(t+1)$ gebildet. Je nach Implementierung des Algorithmus kann diese Ersetzung auf verschiedene Arten erfolgen. Beim kanonischen Basis-Algorithmus, der in dieser Arbeit im Fokus steht, werden jedoch alle Individuen der aktuellen Population durch die erzeugten Kinder ersetzt.

Dieser Zyklus wird so lange durchlaufen, bis eine der Abbruchbedingungen erfüllt ist. Dazu wird oft das Erreichen einer maximalen Rechenzeit oder die Konvergenz im Suchraum herangezogen[35].

2.3. Schema-Theorem

Für den erfolgreichen Entwurf effizienter genetischer Algorithmen ist es hilfreich, deren innere Funktionsweise zu verstehen. Insbesondere ein Verständnis dafür, warum die Lösungen eines genetischen Algorithmus zu einem Optimum konvergieren, ist von Interesse. Einen Erklärungsansatz hierfür liefert das Schema-Theorem von John H. Holland.

Dazu führt Holland (1975, S. 66-68) erstmals das Konzept der Schemata ein. Diese sollen hier, ohne Beschränkung der Allgemeinheit, am Beispiel von Chro-

[33] Die Population zu einem definierten Zeitpunkt t wird auch *Generation* genannt.
[34] Vgl. Weicker 2015, S. 24.
[35] Vgl. Buttelmann/Lohmann 2004, S. 152.

Schema H	$o(H)$	$\delta(H)$
(0 * * 1)	2	3
(* * 1 1)	2	1
(1 * * *)	1	0
(1 0 1 *)	3	2

Quelle: Eigene Darstellung

Tabelle 1: Schema Eigenschaften

mosomen verdeutlicht werden, die als Zeichenketten der Länge 4 über dem Alphabet {1,0} codiert sind. Ein Beispiel für solch eine Zeichenkette ist (1 0 0 1).

Zeichenketten, bei denen nicht alle Gene mit konkreten Werten belegt sind, werden Schemata genannt[36]. Sie repräsentieren Ähnlichkeitsmuster mehrerer Chromosomen[37]. Zur Darstellung dieser Schemata wird das sog. „Don't Care"-Symbol * als zusätzliches Allel ins Alphabet eingeführt[38]. Das Schema (1 0 * 1) repräsentiert z. B. die Zeichenketten (1 0 0 1) und (1 0 1 1).

Schemata H sind durch zwei grundlegende Eigenschaften charakterisiert: die Ordnung des Schemas $o(H)$ und dessen definierte Länge $\delta(H)$[39]. Die Ordnung eines Schemas ist durch die Anzahl der definierten Positionen bestimmt. Dies sind diejenigen Genpositionen, die mit einem konkreten Wert belegt sind. Die definierte Länge eines Schemas ist die maximale Entfernung zweier solcher definierter Positionen. Beispiele dazu sind in Tabelle 1 dargestellt.

Für Zeichenketten der Länge l über einem Alphabet der Kardinalität k gibt es insgesamt $(k+1)^l$ verschiedene Schemata[40]. Wie viele dieser Schemata in einer konkreten Population existieren ist nur schwer bestimmbar. Es kann jedoch eine untere und obere Schranke angegeben werden. So bewegt sich die Anzahl der Schemata in einer Population der Größe n, je nach Diversität der Individuen, zwischen k^l und $n \cdot k^l$ [41]. Bei der Verarbeitung der Individuen wird so implizit die Information aus einer Vielzahl von Schemata durch den Algorithmus ausgewertet. Dies wird als *impliziter Parallelismus* bezeichnet[42].

Es wurde untersucht, inwiefern die Selektion, Rekombination und Mutation dazu beitragen, Schemata mit einer überdurchschnittlichen mittleren Fitness[43] in

[36] Vgl. Goldberg 1989, S. 19.
[37] Vgl. ebd., S. 19.
[38] Vgl. ebd., S. 19.
[39] Vgl. ebd., S. 29.
[40] Vgl. ebd., S. 29.
[41] Vgl. ebd., S. 29.
[42] Vgl. ebd., S. 20.
[43] Die mittlere Fitness eines Schemas ergibt sich aus dem arithmetischen Mittel der Fitness der Individuen, die von diesem Schema repräsentiert werden.

der Population zu verbreiten. Individuen mit überdurchschnittlicher Fitness, und damit implizit deren Schemata, werden mit einer höheren Wahrscheinlichkeit durch die Selektion für die Fortpflanzung ausgewählt. So trägt diese dazu bei, dass sich solche Schemata von Generation zu Generation überdurchschnittlich stark ausbreiten[44]. Bei der Rekombination besteht allerdings die Gefahr, dass diese Schemata wieder aufgespalten werden. Je größer die definierte Länge eines Schemas, desto höher ist die Wahrscheinlichkeit, dass es durch die Rekombination aufgebrochen wird[45]. Da die Mutation nur mit einer sehr geringen Wahrscheinlichkeit ein Allel verändert, hat diese keinen großen Einfluss auf die Verbreitung guter Schemata[46].

Insgesamt besagt das Schema-Theorem, dass sich überdurchschnittlich fitte Schemata niedriger Ordnung und mit einer kurzen definierten Länge zu überdurchschnittlich fitten Schemata höherer Ordnung zusammensetzen und sich so exponentiell in der Population ausbreiten[47]. Solche Schemata werden „Building Blocks“ genannt.

Das Schema-Theorem ist allerdings nicht unumstritten. Zum einen werden zur Herleitung des Theorems Erwartungswerte genutzt, die von einer unendlich großen Population ausgehen. Die Aussagekraft für die i. d. R. sehr kleinen Populationen genetischer Algorithmen muss daher kritisch hinterfragt werden[48]. Zum anderen wird das Phänomen der Epistase, d. h. der wechselseitigen Abhängigkeit von Genen, vernachlässigt[49]. Epistase entsteht immer dann, wenn die Gesamtfitness eines Individuums keine Linearkombination der Fitness einzelner Gene darstellt[50]. Dies ist z. B. dann der Fall, wenn der Beitrag eines Allels zur Gesamtfitness von der Ausprägung anderer Allele abhängig ist[51].

2.4. Codierung

Einer der ersten Schritte beim Entwurf genetischer Algorithmen ist die Festlegung der Codierung der Individuen. Chromosomen werden im Computer als Zeichenfolgen dargestellt, wobei jede Position ein Gen repräsentiert und jedes Zeichen ein Allel. Das zur Darstellung der Zeichen verwendete Alphabet ist dabei prinzipiell beliebig wählbar.

[44] Vgl. Goldberg 1989, S. 20.
[45] Vgl. ebd., S. 20.
[46] Vgl. ebd., S. 20.
[47] Vgl. ebd., S. 33.
[48] Vgl. Weicker 2015, S. 94.
[49] Vgl. Davidor 1991, S. 23–24.
[50] Vgl. Aickelin 1999, S. 75.
[51] Vgl. ebd., S. 75.

Für gewöhnlich gibt es viele mögliche Codierungen für dasselbe Problem. Beim Entwurf sollten allerdings diese grundlegenden Prinzipien beachtet werden:

Prinzip der sinnvollen Building Blocks Es ist eine Codierung zu wählen, die den Algorithmus möglichst gut bei einer effizienten Schema-Verarbeitung unterstützt. Demnach ist vor allem darauf zu achten, dass der Abstand zwischen den Positionen der Gene, die bezüglich der Fitness voneinander abhängig sind, möglichst gering ist[52].

Prinzip des minimal möglichen Alphabets Für die Codierung der Allele soll das Alphabet mit der niedrigsten Kardinalität gewählt werden, dass zur natürlichen Darstellung des Problems geeignet ist[53]. Dies wird damit begründet, dass ein minimales Alphabet, wie z. B. das binäre Alphabet $\{1, 0\}$, die Zahl der Schemata, die gleichzeitig pro Individuum verarbeitetet werden, maximiert[54].

Außerdem besteht die Möglichkeit, bereits bei der Wahl der Codierung die Einhaltung von Nebenbedingungen implizit zu gewährleisten[55]. So kann beispielsweise die Einhaltung der Nebenbedingung $x \in \mathbb{N}$ und $0 \leq x \leq 15$ dadurch erreicht werden, dass die Werte per 4-stelligem Binärcode dargestellt werden.

2.5. Initialisierung

Bei der Initialisierung genetischer Algorithmen müssen zwei grundlegende Entscheidungen getroffen werden: Die Populationsgröße und die Art der Erzeugung der Anfangspopulation.

Die Wahl der Populationsgröße ist stark problemspezifisch und hängt u. a. von der Schwierigkeit und dem Charakter des Optimierungsproblems, dem involvierten Selektionsoperator, sowie dem Zusammenspiel von Mutation und Rekombination ab[56]. Es ist zu erwarten, dass mit steigender Populationsgröße, aufgrund der höheren Diversität der Individuen, bessere Lösungen gefunden werden können[57]. Andererseits steigt mit der Populationsgröße auch die be-

[52] Vgl. Goldberg 1989, S. 80.
[53] Vgl. ebd., S. 80.
[54] Vgl. ebd., S. 81–82.
[55] Vgl. Aickelin 1999, S. 29–30.
[56] Vgl. Weicker 2015, S. 87.
[57] Vgl. Aickelin 1999, S. 55.

nötigte Rechenzeit pro Generation[58] und somit die Zeit, die insgesamt bis zu einer Konvergenz des Algorithmus benötigt wird. Aufgrund der starken Abhängigkeit vom zugrunde liegenden Problem muss eine genauere Bestimmung der optimalen Populationsgröße letztendlich experimentell erfolgen.

Die Individuen der Anfangspopulation werden häufig zufällig erzeugt, um so durch eine hohe Diversität der Individuen den gesamten Suchraum abzudecken. Alternativ kann der Algorithmus auch mit Individuen initialisiert werden, von denen bereits bekannt ist, dass sie hohe Fitnesswerte aufweisen oder gültige Individuen darstellen. Dies hat allerdings den Nachteil, dass sich der Algorithmus so meist schon zu Beginn in einem lokalen Optimum befindet, was das Erreichen des globalen Optimums erschwert[59].

2.6. Selektion

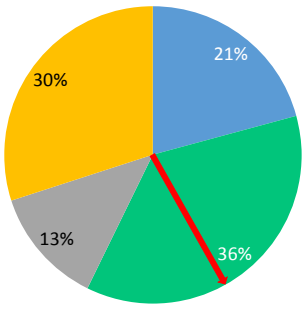
Wie zuvor bereits erwähnt, bestimmt die Selektion die Richtung des Suchprozesses, indem sie Individuen mit einer höheren Fitness bevorzugt für die Erzeugung der Kindindividuen auswählt. Die Auswahl der Selektionsmethode spielt dabei eine entscheidende Rolle für das Konvergenzverhalten des genetischen Algorithmus[60]. Drei gängige Methoden sollen daher kurz vorgestellt werden: fitnessproportionale Selektion, stochastisches universelles Sampling und rangbasierte Selektion.

Fitnessproportionale Selektion Beträgt die Populationsgröße n, so wird bei der fitnessproportionalen Selektion n-mal nacheinander jeweils ein Individuum für die Erzeugung der Kindindividuen ausgewählt. Bei jedem Selektionsvorgang verhält sich die Wahrscheinlichkeit $P(S_i)$ für die Auswahl des i-ten Individuums proportional zu dessen Fitnesswert und kann nach folgender Formel berechnet werden[61,62]:

$$P(S_i) = \frac{f(i)}{\sum_{k=1}^{n} f(k)} .$$

Diese Selektionsmethode wird häufig durch eine Analogie zum Roulette-Rad in Casinos verdeutlicht und deshalb auch *Roulette-Selektion* genannt. Wie in

[58] Vgl. Aickelin 1999, S. 55.
[59] Vgl. Burke/Causmaecker/Vanden Berghe 1999, S. 190.
[60] Vgl. Buttelmann/Lohmann 2004, S. 154.
[61] Vgl. Weicker 2015, S. 72.
[62] Die Funktion $f(i)$ berechnet die Fitness des i-ten Individuums.



Quelle: Eigene Darstellung

Abbildung 2: Fitnessproportionale Selektion

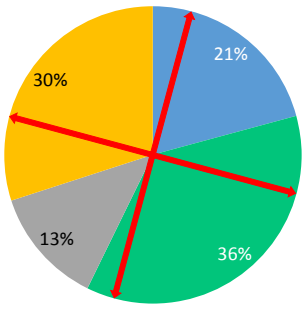
Abbildung 2 dargestellt, wird dazu jedem Individuum ein Segment auf dem Roulette-Rad zugewiesen, wobei die Größe des jeweiligen Segments der berechneten Auswahlwahrscheinlichkeit entspricht[63]. Die Selektion der n Individuen erfolgt dann durch n-maliges Drehen des Rades. Je besser die Fitness eines Individuums und damit je größer das Segment, desto höher ist die Wahrscheinlichkeit, mehrmals für die Erzeugung von Nachkommen ausgewählt zu werden.

Ein großer Nachteil dieses Verfahrens ist jedoch die relativ hohe Varianz[64], d. h. durch mehrmaliges Drehen am Rad kann es durchaus vorkommen, dass die tatsächliche Auswahlhäufigkeit eines Individuums erheblich vom Erwartungswert $n \cdot P(S_i)$ abweicht.

Stochastisches universelles Sampling Das stochastische universelle Sampling (SUS) begegnet dem Problem der Varianz damit, dass das Roulette-Rad nur einmal, aber dafür mit n äquidistant angeordneten Kugeln, gedreht wird (siehe Abbildung 3). Die Auswahlhäufigkeit entspricht so dem Erwartungswert und ein Individuum mit einer Auswahlwahrscheinlichkeit von $P(S_i) > \frac{1}{n}$ wird mindestens einmal selektiert[65].

Rangbasierte Selektion Beide bisher vorgestellten Selektionsmethoden haben die Eigenschaft, dass sie zu Beginn der Suche dazu neigen, sehr schnell zu konvergieren und somit die Gefahr einer Stagnation in einem lokalen Optimum besteht[66]. Gegen Ende des Suchprozesses entsteht hingegen das entgegengesetzte Problem: Da hier die Gütewerte der Individuen nahe beieinander

[63] Vgl. Buttelmann/Lohmann 2004, S. 154.
[64] Vgl. Weicker 2015, S. 75.
[65] Vgl. ebd., S. 75.
[66] Vgl. Buttelmann/Lohmann 2004, S. 154.



Quelle: Eigene Darstellung

Abbildung 3: Stochastisches universelles Sampling

liegen, haben die besseren Individuen kaum noch einen Selektionsvorteil, sodass das Verhalten fast einer gleichverteilten zufälligen Auswahl entspricht[67]. Eine Konvergenz zum Optimum wird so erschwert.

Eine Methode, die dieses problematische Konvergenzverhalten verhindert, ist die rangbasierte Selektion. Die absoluten Fitnesswerte sind bei dieser Methode nicht von Bedeutung, da nur das relative Verhältnis der Werte zueinander betrachtet wird[68]. Die Individuen einer Population werden gemäß ihrer Fitnesswerte von Rang 1 (beste Fitness) bis Rang n (schlechteste Fitness) sortiert. Aus diesem Rang r kann dann die jeweilige skalierte Auswahlwahrscheinlichkeit direkt berechnet werden, meist linear durch die Formel[69,70]

$$P(S_r) = \frac{2}{n} \cdot \left(1 - \frac{r-1}{n-1}\right) .$$

Auf Basis der so skalierten Wahrscheinlichkeiten erfolgt die eigentliche Auswahl anschießend per stochastischem universellem Sampling[71].

2.7. Genetische Operatoren

2.7.1. Rekombination

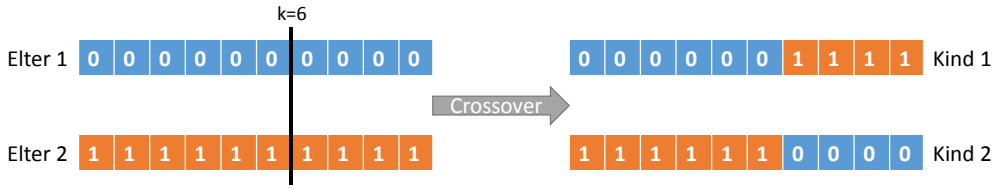
Die Rekombination ist der wichtigste Operator eines genetischen Algorithmus[72]. Durch eine Durchmischung der Allele zweier Elternindividuen entstehen bei der Rekombination zwei neue Kindindividuen. Als Eltern stehen dazu die per Selektion ausgewählten Individuen zur Verfügung. Die Menge dieser

[67] Vgl. Weicker 2015, S. 73.
[68] Vgl. ebd., S. 73.
[69] $P(S_r)$ bezeichnet die Selektionswahrscheinlichkeit des Individuums mit Rang r.
[70] Vgl. Weicker 2015, S. 75.
[71] Vgl. Buttelmann/Lohmann 2004, S. 155.
[72] Vgl. Goldberg 1989, S. 14.

Eltern wird auch Heiratspool genannt[73]. Allerdings kommt die Rekombination nur mit einer gewissen Wahrscheinlichkeit $P(C)$ zur Anwendung. Die restlichen Individuen werden unverändert an den nachfolgenden Mutationsoperator weitergereicht[74]. Die Wahl von $P(C)$ ist stark problemabhängig. In praktischen Anwendungen haben sich jedoch Werte zwischen $0,6$ und $1,0$ bewährt[75].

Die Rekombination wird in zwei Schritten vollzogen. Zuerst werden zwei Individuen zufällig dem Heiratspool entnommen, dann wird mit der Wahrscheinlichkeit $P(C)$ eine der Varianten des Crossovers auf die Eltern angewandt, um so die Nachkommen zu erzeugen[76]. Die bekanntesten Varianten dieses Operators sind das klassische *1-Punkt-Crossover*, *n-Punkt-Crossover* und *uniformes Crossover*.

1-Punkt-Crossover Beim 1-Punkt-Crossover zweier Chromosomen der Länge l wird zunächst eine ganzzahlig positive Kreuzungsposition k zufällig gewählt, sodass $k \in \{1, \ldots, l-1\}$. An dieser Position werden nun die Chromosomen der Eltern aufgespalten, sodass insgesamt 4 Teilabschnitte entstehen. Diese Teilabschnitte werden dann über Kreuz zu den Kindern zusammengesetzt, d. h. Kind 1 besteht aus den ersten k Genen des Elters 1 und den letzten $l-k$ Genen des Elters 2. Für Kind 2 verhält sich dies genau entgegengesetzt.



Quelle: Eigene Darstellung

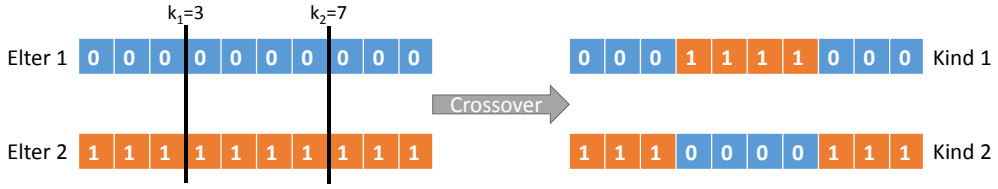
Abbildung 4: 1-Punkt-Crossover

Da bei dieser Variante immer nur ein Kreuzungspunkt gewählt wird, tritt folgender Effekt auf: Je weiter entfernt zwei Gene auf dem Chromosom voneinander positioniert sind, desto höher ist die Wahrscheinlichkeit, dass diese durch das Crossover getrennt werden. Anfangs- und Endposition werden darüber hinaus immer getrennt. Diese als „positional bias“ bezeichnete Eigenschaft er-

[73] Goldberg 1989, S. 12.
[74] Vgl. Weicker 2015, S. 128.
[75] Vgl. Goldberg 1989, S. 111.
[76] Vgl. ebd., S. 12.

schwert oft das Erreichen einiger Regionen des Suchraums, während andere bevorzugt werden[77].

n-Punkt-Crossover Beim n-Punkt-Crossover zweier Chromosomen der Länge l werden, statt nur einer, insgesamt n Kreuzungspositionen $(k_1, \dots k_n)$ zufällig gewählt, sodass $\forall i : k_i \in \{1, \dots, l-1\}$ und $n \in \{2, \dots, l-1\}$. Für die Zusammensetzung der Kinder werden dann immer abwechselnd ein Teilstück des Elters 1 und des Elters 2 verwendet. Abbildung 5 zeigt dies am Beispiel eines 2-Punkt-Crossovers.



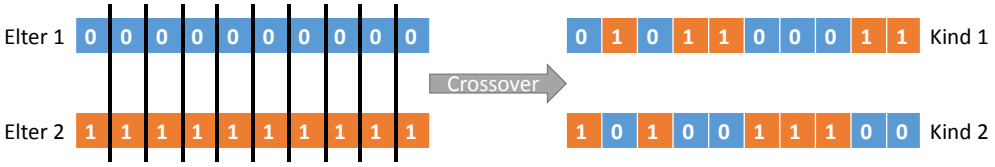
Quelle: Eigene Darstellung

Abbildung 5: n-Punkt-Crossover

Je höher n gewählt wird, desto mehr wird der „positional bias“-Effekt abgeschwächt[78]. Auf der anderen Seite wird allerdings auch die Vererbung größerer Building Blocks an die Nachkommen erschwert.

Uniformes Crossover Uniformes Crossover verhält sich recht ähnlich einem n-Punkt-Crossover mit $n = l - 1$. Die Chromosomen der Eltern werden auch hier mit maximaler Granularität aufgespalten. Anders als beim n-Punkt-Crossover werden dann aber bei der Zusammensetzung der Kinder die Gene nicht abwechselnd von einem der Eltern übernommen, sondern zufällig (siehe Abbildung 6). Die Wahrscheinlichkeit, dass ein einzelnes Gen von Elter 1 übernommen wird, kann dabei definiert werden. Übliche Werte liegen zwischen $0,5$ und $0,8$[79].

[77] Vgl. Buttelmann/Lohmann 2004, S. 156.
[78] Vgl. ebd., S. 156.
[79] Vgl. ebd., S. 156.



Quelle: Eigene Darstellung

Abbildung 6: Uniformes Crossover

2.7.2. Mutation

Bei der Mutation werden einzelne Allele zufällig mit einer geringen Wahrscheinlichkeit $P(M)$ verändert (siehe Abbildung 7). Obwohl die Rekombination für die Suche nach besseren Individuen der primär entscheidende Operator ist, spielt auch die Mutation eine wichtige Rolle. Durch sie können Allele, die entweder durch Rekombination verloren gegangen sind oder bereits in der Anfangspopulation nicht vorhanden waren, wieder eingeführt werden[80]. Sie dient also primär der Erkundung des Suchraums und kann so einer frühzeitigen Konvergenz gegen ein lokales Optimum entgegenwirken[81].

Quelle: Eigene Darstellung

Abbildung 7: Mutation

Die optimale Mutationsrate ist erneut stark problemabhängig. Goldberg (1989, S. 14) empfiehlt einen Wert in der Ordnung $P(M) = 0,001$.

2.8. Elitismus

Das Ersetzungsschema eines kanonischen genetischen Algorithmus arbeitet generationsbasiert, d. h. alle Individuen der vorherigen Generation werden durch die neu generierte Population vollständig ersetzt. Dies birgt jedoch die Gefahr, dass das beste bisher gefundene Individuum dabei verloren geht[82]. Dem wird

[80] Vgl. Goldberg 1989, S. 14.
[81] Vgl. ebd., S. 14.
[82] Vgl. Buttelmann/Lohmann 2004, S. 155.

häufig durch Anwendung des sog. Elitismus entgegengewirkt. Dazu werden vor der eigentlichen Ersetzung der Population die k schlechtesten Nachkommen durch die k besten Eltern ersetzt.

3. Optimierungsmodell der Dienstplanerstellung

3.1. Problemanalyse

3.1.1. Definition und Terminologie

Bevor das Problem der Dienstplanerstellung[83] formal als Optimierungsmodell angegeben werden kann, soll das Problem und die damit verbundene Terminologie zunächst definiert und erläutert werden.

Dienstplanerstellung bezeichnet den Vorgang der Zuordnung von Mitarbeitern zu Aufgaben in einer Reihe von Schichten, innerhalb eines definierten Planungszeitraums[84]. Eine Aufgabe wird, im Kontext der Dienstplanerstellung, während einer vordefinierten Zeitspanne durchgeführt, die als Schicht bezeichnet wird[85]. Diese Schichten sind zeitlich fixiert (z. B. „Frühschicht“)[86].

Dies soll am Beispiel der IT-Organisation der Evonik Industries AG verdeutlicht werden. Die operativen Teams erbringen ihre Dienstleistungen rund um die Uhr in einer speziellen Form des Schichtbetriebs nach dem „Follow-the-Sun“-Prinzip. Räumlich sind die Mitarbeiter dieser Teams dabei auf drei Kontinente (Amerika, Europa und Asien) und somit auf drei Zeitzonen verteilt. Daraus ergeben sich täglich drei Schichten:

- AMERICAS-Schicht (Amerika)
- EMEA-Schicht (Europa, mittlerer Osten, Afrika)
- APAC-Schicht (Asien-Pazifik)

In jeder dieser Schichten fallen eine Reihe von Aufgaben an, die von in dieser Zeitzone verfügbaren Mitarbeitern besetzt werden müssen. Der Bedarf an Mitarbeitern pro zu besetzender Aufgabe kann dabei, abhängig vom Wochentag,

[83] Im englischsprachigen Raum auch „Employee Timetabling Problem“ (ETP) genannt.
[84] Vgl. Meisels/Schaerf 2003, S. 1.
[85] Vgl. ebd., S. 2.
[86] Vgl. ebd., S. 2.

variieren. Einige dieser Aufgaben, wie z. B. „Funktionsexperte Virtualisierung“ oder „Funktionsexperte Backup“, sind dabei allerdings keine schichtfüllenden Aufgaben, sondern können vielmehr als Rollen verstanden werden, von denen ein einzelner Mitarbeiter eine als zulässig definierte Kombination übernehmen kann.

Ein Dienstplan soll im Folgenden als „gültig“ bezeichnet werden, wenn alle der sog. „harten Nebenbedingungen“ erfüllt sind. Zusätzlich zu diesen harten Nebenbedingungen werden „weiche Nebenbedingungen“ formuliert, deren Verletzung prinzipiell toleriert werden kann. Jede Verletzung einer weichen Nebenbedingung mindert allerdings die Qualität des betroffenen Dienstplans. Die Summe all dieser Verletzungen bestimmt somit die Güte eines Dienstplans.

3.1.2. Harte und weiche Nebenbedingungen

In diesem Abschnitt erfolgt die verbale Beschreibung der einzuhaltenden harten und weichen Nebenbedingungen. Die formale Darstellung ist Teil des mathematischen Optimierungsmodells in Abschnitt 3.2.

Harte Nebenbedingungen

Die Einführung der harten Nebenbedingungen H1—H3 erfolgt in Anlehnung an das Problemmodell aus Meisels/Schaerf (2003, S. 3–4). Nebenbedingung H4 stellt eine Erweiterung dieses Modells dar.

H1: Personalbedarf Jede Schicht besteht aus verschiedenen Aufgaben, die, teils mehrfach, zu besetzen sind. Jeder zu besetzenden Aufgabe einer Schicht ist genau ein Mitarbeiter zuzuordnen. Sowohl eine Unter- als auch Überdeckung des Personalbedarfs ist unzulässig.

H2: Qualifikation Jeder Mitarbeiter verfügt über Qualifikationen, die ihn befähigen, bestimmte Aufgaben durchzuführen. Einem Mitarbeiter können nur Aufgaben zugewiesen werden, für die er qualifiziert ist.

H3: Verfügbarkeit Für jede der Schichten steht i. d. R. nur ein Teil der Mitarbeiter zur Verfügung. Gründe sind z. B. unumgängliche Abwesenheiten (ex-

terne Schulung, bereits genehmigter Urlaub etc.) oder vertragliche Regelungen. Dies schränkt die Anzahl möglicher Zuordnungen zusätzlich ein.

H4: Aufgabenkombinationen Jedem verfügbaren Mitarbeiter können pro Schicht jeweils keine, eine oder mehrere Aufgaben zugeordnet werden. Welche Aufgabenkombinationen den Mitarbeitern zugeordnet werden dürfen, ist durch eine vordefinierte Menge an zulässigen Kombinationen bestimmt.

Weiche Nebenbedingungen

Art und Umfang der weichen Nebenbedingungen können, je nach Unternehmen, stark variieren. In dieser Arbeit werden daher nur diejenigen Bedingungen betrachtet, die für das gewählte Beispielunternehmen von Bedeutung sind.

W1: Präferenzen Jeder Mitarbeiter hat die Möglichkeit, Präferenzen zu formulieren, bestimmten Schichten nicht zugeordnet zu werden (z. B. Urlaubswunsch).

W2: Vollständige Wochenenden Meist ist es wünschenswert, dass einem Mitarbeiter, der zu einer Schicht am Wochenende eingeteilt wird, auch die Schicht des zweiten Tags des Wochenendes zugeteilt wird. Eine Zuordnung kompletter Wochenenden reduziert insgesamt die Anzahl an Mitarbeitern, die an diesen unbeliebten Schichten arbeiten müssen.

W3: Maximale Zuweisungen pro Woche Um eine Überlastung der Mitarbeiter zu vermeiden und einen Überstundenabbau zu ermöglichen, ist es außerdem wünschenswert, die maximale Anzahl an zugewiesenen Schichten pro Woche für jeden Mitarbeiter limitieren zu können.

3.2. Formale Darstellung des Optimierungsmodells

Aufbauend auf den bisherigen Überlegungen kann nun das mathematische Optimierungsmodell wie folgt angegeben werden.

Mengen und Indizes

m	Anzahl der Mitarbeiter
n	Anzahl der Schichten
t	Anzahl der Aufgaben
u	Anzahl der zulässigen Aufgabenkombinationen (inklusive Einzelaufgaben und „Keine Aufgabe“)
v	Anzahl der weichen Randbedingungen

I	Indexmenge für Mitarbeiter $I := \{1, \ldots, m\}$
J	Indexmenge für Schichten $J := \{1, \ldots, n\}$
K	Indexmenge für Aufgaben $K := \{1, \ldots, t\}$
L	Indexmenge für zulässige Aufgabenkombinationen $L := \{0, \ldots, u-1\}$
B	Indexmenge für weiche Randbedingungen $B := \{1, \ldots, v\}$

E	Menge aller Mitarbeiter $E := \{e_i \mid i \in I\}$
S	Menge aller Schichten im Planungszeitraum $S := \{s_j \mid j \in J\}$
T	Menge aller Aufgaben $T := \{t_k \mid k \in K\}$
TC	Menge aller zulässigen Aufgabenkombinationen $TC := \{tc_l \mid l \in L, tc_l \subseteq T\}$
TA_{ij}	Menge aller Aufgaben, denen Mitarbeiter e_i in Schicht s_j zugeordnet ist $TA_{ij} \subseteq T$
W	Menge aller weichen Randbedingungen $W := \{w_b \mid b \in B\}$

Entscheidungsvariable

$$x_{ijk} = \begin{cases} 1 & \text{Mitarbeiter } e_i \text{ bearbeitet Aufgabe } t_k \text{ in Schicht } s_j \\ 0 & \text{sonst} \end{cases}$$

Modellparameter

R Positiv ganzzahlige $n \times t$ - Personalbedarfs-Matrix (H1)
$R = (r_{jk})$
r_{jk} = Anzahl Mitarbeiter, die Aufgabe k in Schicht j zuzuordnen sind

Q Binäre $m \times t$ - Qualifikations-Matrix (H2)
$Q = (q_{ik})$

$$q_{ik} = \begin{cases} 1 & \text{Mitarbeiter } e_i \text{ ist für Aufgabe } t_k \text{ qualifiziert} \\ 0 & \text{sonst} \end{cases}$$

A Binäre $m \times n$ - Verfügbarkeits-Matrix (H3)
$A = (a_{ij})$

$$a_{ij} = \begin{cases} 1 & \text{Mitarbeiter } e_i \text{ ist für Schicht } s_j \text{ verfügbar} \\ 0 & \text{sonst} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & TA_{ij} \in TC \\ 0 & \text{sonst} \end{cases}$$

$\Phi_{bi}(x_{ijk})$ Straffunktion zur Berechnung des Verletzungsgrades der weichen Nebenbedingung w_b, bezogen auf Mitarbeiter e_i, abhängig von dessen Dienstplan

λ_{bi} Gewichtungs-Koeffizient der Straffunktion Φ_{bi}

Zielfunktion

$$\text{Minimiere} \quad \sum_{i=1}^{m} \sum_{b=1}^{v} (\lambda_{bi} \cdot \Phi_{bi}(x_{ijk}))$$

u. d. N.[87]

$$\sum_{i=1}^{m} x_{ijk} = r_{jk}, \qquad \forall j \in J, \forall k \in K \qquad \text{(H1)}$$

$$x_{ijk} \leq q_{ik}, \qquad \forall i \in I, \forall j \in J, \forall k \in K \qquad \text{(H2)}$$

$$x_{ijk} \leq a_{jk}, \qquad \forall i \in I, \forall j \in J, \forall k \in K \qquad \text{(H3)}$$

$$y_{ij} = 1, \qquad \forall i \in I, \forall j \in J \qquad \text{(H4)}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i \in I, \forall j \in J, \forall k \in K$$

[87] Unter den Nebenbedingungen.

4. Entwurf des genetischen Algorithmus

4.1. Nebenbedingungen

In Abschnitt 3.1.1 wurde bereits definiert, dass sich die Güte eines Dienstplans aus der Summe der Verletzungen der weichen Nebenbedingungen ergibt, welche es zu minimieren gilt. Gültig sind allerdings nur Dienstpläne, die alle formulierten harten Nebenbedingungen einhalten. Somit müssen zunächst geeignete Methoden zum Umgang mit diesen harten Nebenbedingungen gefunden werden. Zwei gängige Herangehensweisen sind dazu die Anwendung restriktiver Methoden und die Anwendung toleranter Methoden:

Restriktive Methoden Restriktive Methoden verfolgen den Ansatz, das Vorkommen ungültiger Individuen zu verhindern[88].

Dies kann z. B. dadurch erreicht werden, dass die Einhaltung von Nebenbedingungen durch eine entsprechende *Codierung* der Individuen garantiert wird. Ein Beispiel dazu wurde bereits in Abschnitt 2.4 gegeben. Diese Methode ist allerdings nur für spezielle Arten von Nebenbedingungen möglich, was deren Anwendbarkeit stark einschränkt[89].

Eine weitere Möglichkeit besteht darin, bei Entstehung ungültiger Individuen, diese sofort wieder zu verwerfen (*Krippentod*). Dies ist auf sehr einfache Weise zu implementieren, schränkt aber auch die Diversität unter den Individuen erheblich ein. Das kann zur Folge haben, dass die Suche nach dem globalen Optimum stark erschwert oder sogar verhindert wird, da der Weg zu diesem Optimum beispielsweise nur über eine Rekombination mit ungültigen Individuen erreichbar ist[90].

Alternativ können ungültige Individuen auch über einen speziellen *Reparaturalgorithmus* in gültige Individuen überführt werden. Da jedes erzeugte Individuum so in einem gültigen Individuum resultiert, müssen nicht mehr so viele Individuen erzeugt werden als bei Anwendung des Krippentods[91]. Allerdings ist ein sehr hohes Problemwissen für den Entwurf eines solchen Algorithmus nötig. Darüber hinaus bleiben, wie bei Anwendung des Krippentods, die Nachteile der geringen Diversität bestehen.

[88] Vgl. Weicker 2015, S. 191.
[89] Vgl. Aickelin 1999, S. 29.
[90] Vgl. Weicker 2015, S. 194.
[91] Vgl. ebd., S. 194.

Die Entstehung ungültiger Individuen kann außerdem durch die geschickte Kombination aus speziellem Initialisierungsalgorithmus und speziell angepasster Crossover- und Mutationsoperatoren verhindert werden[92]. Dabei wird die Anfangspopulation durch einen entsprechenden Algorithmus nur mit gültigen Individuen initialisiert und die genetischen Operatoren so entworfen, dass diese die Gültigkeit der Individuen bewahren.

Tolerante Methoden Tolerante Methoden lassen die Entstehung ungültiger Individuen grundsätzlich zu, benachteiligen diese jedoch gegenüber gültigen Individuen durch die Selektion. Dies wird meist erreicht, indem die Fitness ungültiger Individuen proportional zum jeweiligen Verletzungsgrad schlechter bewertet wird[93]. Je nachdem, ob es sich um ein Maximierungs- oder Minimierungsproblem handelt, muss dazu in der Fitnessfunktion ein entsprechender Strafwert subtrahiert respektive addiert werden.

Beim Entwurf eines genetischen Algorithmus ist man keineswegs auf die Verwendung nur einer der beschriebenen Methoden beschränkt. Eine Kombination ist möglich[94] und oftmals sinnvoll und angebracht. Auch in dieser Arbeit wird eine kombinierte Strategie verfolgt.

Für möglichst viele Nebenbedingungen soll deren Verletzung dadurch vermieden werden, dass betroffene Individuen gar nicht erst entstehen. Im Verlauf der nächsten Unterabschnitte wird sich zeigen, dass sich Nebenbedingung H4 durch die Wahl einer geeigneten Codierung implizit erfüllen lässt[95]. Verletzungen der Nebenbedingungen H2 und H3 lassen sich mit akzeptablem Aufwand durch eine geeignete Initialisierung und spezielle Crossover- und Mutationsoperatoren vermeiden[96].

Wie sich herausstellen wird, ist die Verletzung der Nebenbedingung H1 jedoch nicht ohne weiteres vollständig vermeidbar, sodass an dieser Stelle zwischen der Anwendung einer restriktiven Methode, wie Krippentod oder Reparaturalgorithmus, und einem toleranten Ansatz per Straffunktion gewählt werden muss. Da aufgrund der Vermeidungsstrategie in Bezug auf die Nebenbedingungen H2 — H4 die Diversität der Individuen ohnehin bereits eingeschränkt ist, soll diese nicht noch zusätzlich durch eine restriktive Behandlung der Nebenbedingung H1 verringert werden. Deshalb wird der Grad der Verletzung dieser Nebenbedingung in die Fitnessfunktion mit einfließen.

[92] Vgl. Weicker 2015, S. 194.
[93] Vgl. Goldberg 1989, S. 85.
[94] Vgl. Aickelin 1999, S. 194.
[95] Vgl. Abschnitt 4.3.
[96] Vgl. Abschnitt 4.6.1 und 4.6.2.

4.2. Fitnessfunktion

Basierend auf den soeben gewonnenen Erkenntnissen kann nun die Fitnessfunktion zur Bewertung der Dienstpläne entworfen werden. Diese kann aus der Zielfunktion des Problemmodells abgeleitet werden, muss jedoch noch um eine zusätzliche Funktion zur Berechnung der Strafkosten, die bei Verletzung der Nebenbedingung H1 entstehen, erweitert werden.

Dazu wird die Straffunktion Ψ eingeführt, die für jede einzelne Schicht die mit ihr verbundenen Strafkosten, abhängig vom Grad der Unter- oder Überdeckung des Personalbedarfs, ermittelt. Die gesamten Strafkosten ergeben sich dann aus der durch ρ gewichteten Summe der einzelnen Strafkosten. Formal wird diese Funktion wie folgt definiert:

$$\Psi_j = \sum_{k=1}^{t} \left(|r_{jk} - \sum_{i=1}^{m} x_{ijk}| \right) .$$

Durch Addition der so ermittelten Strafkosten zur eigentlichen Zielfunktion ergibt sich die finale Fitnessfunktion

$$\sum_{i=1}^{m} \sum_{b=1}^{v} (\lambda_{bi} \cdot \Phi_{bi}(x_{ijk})) + \rho \cdot \sum_{j=1}^{n} \Psi_j(x_{ijk}) .$$

Aus dieser zusammengesetzten Fitnessfunktion resultiert nun eine mehrkriterielle Optimierung des Problems: Zum einen werden Dienstpläne gesucht, die den Wünschen und Vorgaben jedes einzelnen Mitarbeiters entsprechen, zum anderen werden Dienstpläne gesucht, die bezogen auf jede Einzelschicht eine optimale Personaldeckung gewährleisten.

Der Beitrag der Zuordnung eines Mitarbeiters zur Gesamtfitness des Dienstplans ist daher auch abhängig von den Zuordnungen aller anderen Mitarbeiter, die selbe Schicht betreffend. Dieses Phänomen der Epistase[97] muss beim Entwurf der Codierung und der genetischen Operatoren berücksichtigt werden.

4.3. Codierung

Vor dem Entwurf einer Codierung ist es sinnvoll, sich zunächst Gedanken darüber zu machen, welche Merkmale eines Dienstplans entscheidend für dessen

[97] Vgl. Abschnitt 2.3.

Fitnessbewertung sind. Wie man an der Fitnessfunktion erkennen kann, hängt die Bewertung vollständig von den Werten der Entscheidungsvariable x_{ijk} ab, d. h. von der jeweiligen Zuordnung eines Mitarbeiters zu einer Aufgabe in einer Schicht.

In Abschnitt 4.1 wurde gefordert, die Einhaltung möglichst vieler harter Nebenbedingungen implizit, z. B. durch die Codierung, zu erfüllen. Nebenbedingung H4 scheint hierfür ein geeigneter Kandidat zu sein. Statt pro Gen die Zuweisung eines Mitarbeiters zu einer einzelnen Aufgabe abzubilden, erfolgt deshalb eine Zuweisung zu einer zulässigen Aufgabenkombination. Welche Kombinationen dabei zulässig sind, ist vom Anwender zu definieren. Ein Beispiel einer solchen Definition zeigt Tabelle 2. Wählt man nun als Alphabet zur Darstellung der Allele die Menge aller Indizes der definierten Aufgabenkombinationen $\{0, \ldots, u-1\}$, so ist die Einhaltung von H4 garantiert.

Kombinations-ID	0	1	2	3	4	5
Aufgaben-ID(s)	/	1	2	3	1;2	2;3

Quelle: Eigene Darstellung

Tabelle 2: Definition gültiger Aufgabenkombinationen (Beispiel)

Zur Repräsentation eines kompletten Dienstplans sind nun zwei natürliche Darstellungsformen denkbar. Beiden Möglichkeiten ist gemein, dass jedes Individuum aus insgesamt $m \cdot n$ Genen besteht. Die Unterschiede ergeben sich in der Anordnung dieser Gene.

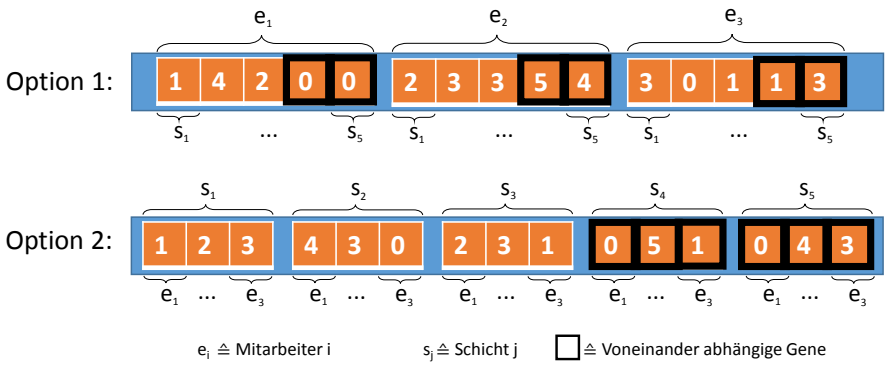
Option 1 geht dabei mitarbeiterbasiert vor, d. h. die Gene $(1, \ldots, n)$ repräsentieren die Zuordnungen des ersten Mitarbeiters zu Aufgabenkombinationen in allen n Schichten. Die Zuordnungen für den zweiten Mitarbeiter erfolgt über die Gene $(n+1, \ldots, 2n)$ usw.

Option 2 hingegen folgt einer schichtbasierten Darstellung, d. h. die Gene $(1, \ldots, m)$ repräsentieren die Zuordnungen aller Mitarbeiter zu einer Aufgabenkombination in der ersten Schicht. Die Zuordnungen für die zweite Schicht erfolgen über die Gene $(m+1, \ldots, 2m)$ usw.

Beide Möglichkeiten sollen bezüglich ihrer Entsprechung des geforderten „Prinzips der sinnvollen Building Blocks“[98] unter Berücksichtigung der epistatischen Effekte bewertet werden. Der Abstand zwischen denjenigen Positionen der Gene, die bezüglich der Fitness voneinander abhängig sind, sollte demnach möglichst gering sein.

[98] Vgl. Abschnitt 2.3.

Dass Option 2 diesem Prinzip deutlich besser entspricht, kann am besten an einem Beispiel demonstriert werden. Abbildung 8 zeigt dazu die Repräsentation eines Dienstplans in beiden Codierungsvarianten.



Quelle: Eigene Darstellung

Abbildung 8: Codierungsoptionen

Ein Merkmal, das in diesem Beispiel die Fitness des Plans positiv beeinflusst, könnte die Tatsache sein, dass Mitarbeiter 1 in den Schichten 4 und 5 (gemäß seinem Wunsch) keinen Aufgaben zugeordnet ist. Die Gesamtfitness hängt jedoch auch davon ab, welche Zuordnungen in diesen Schichten für Mitarbeiter 2 und 3 erfolgen. Wie anhand der grafischen Darstellung erkennbar ist, liegen diese voneinander abhängigen Gene bei Option 2 sehr nahe beieinander, während sie bei Option 1 über das gesamt Chromosom verteilt sind.

Um den Algorithmus bei einer effektiven Verarbeitung guter Building Blocks zu unterstützten, wird daher eine Codierung nach Option 2 festgelegt.

4.4. Initialisierung

Bevor auf den eigentlichen Initialisierungsalgorithmus zur Erzeugung der Anfangspopulation eingegangen wird, muss zuerst ein geeigneter Parameterwert für die Populationsgröße gefunden werden.

Da die Populationsgröße stark problemabhängig ist, sind in der Literatur keine eindeutigen Vorgaben zu deren Bestimmung zu finden. In praktischen Anwendungen mit binärer Codierung werden oft Werte zwischen 30 und 100 angege-

ben[99,100]. Für Anwendungen mit einer Codierung höheren Alphabets werden jedoch auch höhere Werte bis 1000 verwendet[101,102]. Der Wert wird daher zunächst intuitiv auf 500 festgelegt und im Rahmen der Evaluierung genauer bestimmt.

Um die Einhaltung der Nebenbedingungen H2 und H3 zu gewährleisten, wurde in Abschnitt 4.1 die Initialisierung durch einen Algorithmus vorgeschlagen, der dies für alle Individuen der Anfangspopulation sicherstellt. Dazu wird ein Algorithmus entworfen (siehe Algorithmus 1), der für jede zu besetzende Aufgabe zunächst prüft, welche Mitarbeiter sowohl für diese Aufgabe qualifiziert als auch in der entsprechenden Schicht verfügbar sind. Aus dieser Menge wird dann, unter Berücksichtigung der Forderung nach Gültigkeit der Aufgabenkombinationen (H4), ein zufälliger Mitarbeiter für die Zuweisung ausgewählt. Die vollständige Deckung des Personalbedarfs (H1) wird dabei angestrebt, kann vom Algorithmus allerdings nicht garantiert werden.

Algorithm 1 Initialisierung

```
1:  procedure INIT
2:      n ← Anzahl der Schichten
3:      t ← Anzahl der Aufgaben
4:      for each j ∈ {1, ..., n} do
5:          for each k ∈ {1, ..., t} do
6:              r_jk ← Personalbedarf für Aufgabe k in Schicht j
7:              I_verfügbar ← Indizes aller verfügbaren Mitarbeiter
8:              while r_jk > 0 und I_verfügbar ≠ ∅ do
9:                  i ← wähle zufälligen Index aus I_verfügbar
10:                 I_verfügbar ← I_verfügbar \ {i}
11:                 if Mitarbeiter i ist für Aufgabe k qualifiziert then
12:                     K_vorher ← Menge der derzeit zugewiesenen Aufgaben
13:                     K_nachher ← K_vorher ∪ {k}
14:                     if K_nachher ist gültige Kombination then
15:                         ordne zu: Aufgabe k; Mitarbeiter i; Schicht j
16:                         r_jk ← r_jk − 1
```

4.5. Selektionsmethode

In Abschnitt 2.6 wurden die Vorteile einer rangbasierten Selektion in Bezug auf Varianz- und Konvergenzverhalten bereits ausführlich dargelegt. Aus den angeführten Gründen wird die rangbasierte Selektionsmethode für den genetischen Algorithmus ausgewählt.

[99] Vgl. Goldberg 1989, S. 71 und 101.
[100] Vgl. Weicker 2015, S. 130.
[101] Vgl. Raghavjee/Pillay 2013, S. 74.
[102] Vgl. Aickelin 1999, S. 53–57.

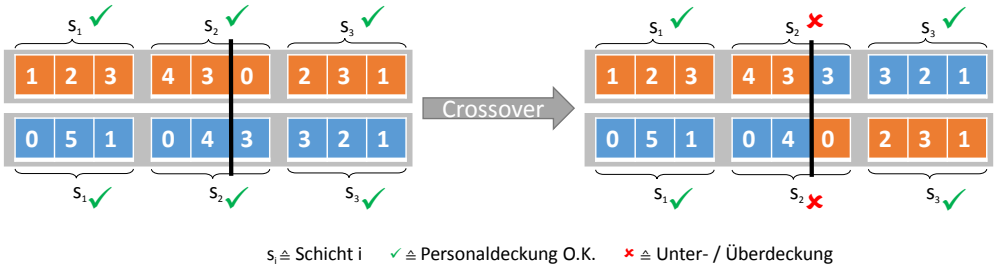
4.6. Genetischen Operatoren

4.6.1. Rekombination

Bei der Wahl des Rekombinationsoperators soll darauf geachtet werden, dass durch die Rekombination zweier gültiger Individuen mit hoher Wahrscheinlichkeit auch gültige Nachkommen entstehen.

In Bezug auf die Nebenbedingungen H2 und H3 ist dies in jedem Fall gegeben. Da durch den Initialisierungsalgorithmus die Verfügbarkeit und Qualifikation der zugeordneten Mitarbeiter in beiden Elternindividuen bereits sichergestellt ist, ist dies auch für deren Kinder garantiert.

Die Erfüllung der Nebenbedingung H1 kann durch die Rekombination jedoch gestört werden. Wird bei der Rekombination gültiger Eltern ein Kreuzungspunkt innerhalb eines Chromosombereichs gewählt, der die Zuordnungen einer bestimmten Schicht definiert, besteht für die resultierenden Kinder eine hohe Wahrscheinlichkeit der Über- oder Unterdeckung des Personalbedarfs in der betroffenen Schicht. Nur wenn der zufällig gewählte Kreuzungspunkt genau zwischen zwei solcher Segmente liegt, wirkt sich die Rekombination nicht auf H1 aus. Abbildung 9 soll dies noch einmal verdeutlichen.



Quelle: Eigene Darstellung

Abbildung 9: Auswirkung Rekombination auf H1

Daraus lässt sich schließen, dass sich mit steigender Anzahl von Kreuzungspunkten die Wahrscheinlichkeit der Erzeugung ungültiger Nachkommen erhöht. Um dies zu vermeiden, ist daher eine Crossovermethode mit geringer Anzahl von Kreuzungspunkten zu wählen. Aufgrund der bereits erwähnten Nachteile des 1-Punkt-Crossovers[103] wird die Verwendung eines 2-Punkt-Crossovers

[103] Vgl. Abschnitt 2.7.1.

festgelegt.

Für den Parameter $P(C)$ wird ein Wert innerhalb des in Abschnitt 2.7.1 angegebenen Wertebereichs gewählt. Er wird auf $P(C) = 0,9$ festgelegt.

4.6.2. Mutation

Durch die Mutation wird jede Zuweisung einer Aufgabenkombination mit einer geringen Wahrscheinlichkeit verändert. Die Neuzuweisung einer rein zufälligen anderen Aufgabenkombination würde allerdings dazu führen, dass durch Verletzung der Nebenbedingungen H1, H2 und H3 vorwiegend ungültige Dienstpläne entstehen. Um dies zu vermeiden, muss ein spezieller Mutationsoperator entwickelt werden (siehe Algorithmus 2). Dieser unterscheidet zwei Fälle:

Ist dem durch die Genposition bestimmten, verfügbaren Mitarbeiter in der entsprechenden Schicht derzeit keine Aufgabe zugeordnet, so sucht der Algorithmus nach einer Aufgabe, die der Mitarbeiter von einem anderen Mitarbeiter übernehmen kann. Die Prüfung der potenziell zuweisbaren Aufgaben erfolgt dabei in zufälliger Reihenfolge. Wurde auf diese Weise eine Aufgabe gefunden, für die der Mitarbeiter qualifiziert ist, so wird sie ihm zugewiesen und die ursprüngliche Zuordnung aufgehoben (Pull-Mutation, Algorithmus 3).

Ist dem durch die Genposition bestimmten Mitarbeiter in der entsprechenden Schicht bereits eine oder mehrere Aufgaben zugewiesen, so wird für jede dieser Aufgaben nach jeweils einem Mitarbeiter gesucht, der diese übernehmen kann. Dazu wird jeweils für die verfügbaren und entsprechend qualifizierten Mitarbeiter in zufälliger Reihenfolge geprüft, ob eine Zuweisung der Aufgabe möglich ist. Dies ist der Fall, wenn die von ihm derzeit ausgeführten Aufgaben und die neue Aufgabe zusammen eine gültige Kombination darstellen. Wurde auf diese Weise ein geeigneter Abnehmer gefunden, so wird ihm die neue Aufgabe zugewiesen und die ursprüngliche Zuordnung aufgehoben (Push-Mutation, Algorithmus 4).

Die Mutationsrate wird in Anlehnung an die Empfehlung von Goldberg (1989, S. 14) auf $P(M) = 0,001$ festgelegt.

Algorithm 2 Mutation

```
1: procedure MUTATE(Genposition g)
2:     i ← betroffener Mitarbeiter an Position g
3:     j ← betroffene Schicht an Position g
4:     l ← Index der zugeordneten Aufgabenkombination
5:     if l = 0 und Mitarbeiter i ist für Schicht j verfügbar then
6:         MUTATEPULL(i, j)
7:     else if l > 0 then
8:         MUTATEPUSH(i, j)
```

Algorithm 3 Pull-Mutation

```
1: procedure MUTATEPULL(Mitarbeiter i_ziel, Schicht j)
2:     i_ziel ← betroffener Mitarbeiter
3:     j ← betroffene Schicht
4:     I_eingeteilt ← Indizes aller zu Schicht j eingeteilter Mitarbeiter
5:     pulled ← false
6:     while pulled = false und I_eingeteilt ≠ ∅ do
7:         i_quelle ← wähle zufällig aus I_eingeteilt
8:         I_eingeteilt ← I_eingeteilt \ {i_quelle}
9:         K_vorher ← Indizes der Aufgaben von Mitarbeiter i_quelle in Schicht j
10:        K_nachher ← K_vorher
11:        while pulled = false und K_vorher ≠ ∅ do
12:            k_quelle ← wähle zufällig aus K_vorher
13:            K_vorher ← K_vorher \ {k_quelle}
14:            if Mitarbeiter i_ziel ist für Aufgabe k_quelle qualifiziert then
15:                K_nachher ← K_nachher \ {k_quelle}
16:                ordne zu: Schicht j, Mitarbeiter i_quelle, Aufgaben K_nachher
17:                ordne zu: Schicht j, Mitarbeiter i_ziel, Aufgabe k_quelle
18:                pulled ← true
```

4.7. Ersetzung und Termination

Um auszuschließen, dass bei der generationsbasierten Ersetzung die besten Individuen verloren gehen, wird deren Erhaltung durch den Einsatz von Elitismus sichergestellt. Dabei wird ein relativ niedriger Wert von 10 Individuen festgelegt, um eine zu starke Dominanz weniger sehr guter Individuen vorzubeugen.

Für den Abbruch des Algorithmus werden folgende Kriterien definiert:

- Erreichen des Optimums (Fitness = 0)
- Keine Verbesserung der Fitness über 50 Generationen

Algorithm 4 Push-Mutation

procedure MUTATEPUSH(Mitarbeiter i_{quelle}, Schicht j)
  $i_{\text{quelle}} \leftarrow$ betroffener Mitarbeiter
  $j \leftarrow$ betroffene Schicht
  $K_{\text{quelle}} \leftarrow$ Indizes aller Aufgaben von Mitarbeiter i_{quelle} in Schicht j
  $I_{\text{verfügbar}} \leftarrow$ Indizes aller für Schicht j verfügbarer Mitarbeiter
  for each $k_{\text{quelle}} \in K_{\text{quelle}}$ **do**
    $pushed \leftarrow false$
    $I_{\text{quali}} \leftarrow$ Indizes aller für Aufgabe k_{quelle} qualifizierter Mitarbeiter
    $I_{\text{gültig}} \leftarrow I_{\text{verfügbar}} \cup I_{\text{quali}}$
    while $pushed = false$ und $I_{\text{gültig}} \neq \emptyset$ **do**
      $i_{\text{ziel}} \leftarrow$ wähle zufällig aus $I_{\text{gültig}}$
      $I_{\text{gültig}} \leftarrow I_{\text{gültig}} \setminus i_{\text{ziel}}$
      $K_{\text{ziel}} \leftarrow$ Indizes zu Mitarbeiter i_{ziel} zugewiesener Aufgaben
      $K_{\text{ziel\_neu}} \leftarrow K_{\text{ziel}} \cup \{k_{\text{quelle}}\}$
      if $K_{\text{ziel\_neu}}$ ist gültige Kombination **then**
        $K_{\text{quelle\_neu}} \leftarrow K_{\text{quelle}} \setminus \{k_{\text{quelle}}\}$
        ordne zu: Schicht j, Mitarbeiter i_{ziel}, Aufgaben $K_{\text{ziel\_neu}}$
        ordne zu: Schicht j, Mitarbeiter i_{quelle}, Aufgaben $K_{\text{quelle\_neu}}$
        $pushed \leftarrow true$

4.8. Zusammenfassung

In Abschnitt 4 wurde ein auf das gegebene Optimierungsproblem abgestimmter genetischer Algorithmus entworfen. Dabei mussten sowohl geeignete Methoden und Operatoren ausgewählt und angepasst werden als auch Parameterwerte festgelegt werden. Die bisherigen Arbeitsergebnisse sind in Tabelle 3 zusammengefasst.

Parameter / Methode	**Wert**
Codierung	schichtbasiert
Populationsgröße	500
Selektion	rangbasierte Selektion
Rekombination	2-Punkt-Crossover
$P(C)$	$0,9$
Mutation	Pull-/Push-Mutation
$P(M)$	$0,001$ pro Gen
Elitismus	10 Individuen
Terminationskriterien	Stagnation über 50 Generationen; Optimum erreicht

Quelle: Eigene Darstellung

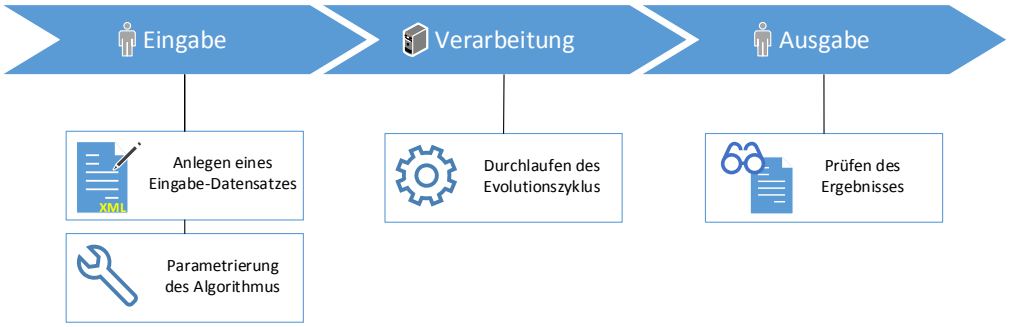
Tabelle 3: Gewählte Parameter und Methoden

5. Evaluierung und Optimierung des genetischen Algorithmus

5.1. Evaluierungsumgebung

Zur Durchführung der nachfolgenden Evaluierung wurde der genetische Algorithmus zur Dienstplanerstellung als prototypische Anwendung implementiert. Dies erfolgte auf einem Windows 7 PC in der Programmiersprache Java 8. Als integrierte Entwicklungsumgebung (IDE) wurde Eclipse, Version 4.5, eingesetzt.

Für die Implementierung genetischer Algorithmen in Java existieren eine Reihe von Frameworks[104], auf die zurückgegriffen werden kann. Für die Entwicklung dieser Anwendung wurde das „Watchmaker Framework“ gewählt. Dieses Framework ist flexibel erweiterbar, bietet eine performante multi-threaded Evolutions-Engine[105] und ist das einzige der Frameworks, das Java Generics unterstützt um Typsicherheit zu gewährleisten[106]. Zur Erstellung der grafischen Benutzeroberfläche wurde Swing, ein Bestandteil der Java Foundation Classes (JFC), verwendet. Die automatisierte Erstellung von Dienstplänen unter Verwendung dieser Anwendung ist in Abbildung 10 als mehrstufiger Prozess dargestellt.



Quelle: Eigene Darstellung

Abbildung 10: Prozess automatisierte Dienstplanerstellung

Zuerst legt der Anwender einen Eingabe-Datensatz in Form einer XML-Datei (Extensible Markup Language) an. Dieser Datensatz enthält alle relevanten

[104] Zum Beispiel ECJ (Evolutionary Computation for Java), JGAP (Java Genetic Algorithms Package) oder Watchmaker Framework.
[105] Ermöglicht eine parallele Fitnessevaluation der Individuen.
[106] Vgl. Dyer 2015, Kapitel 2.

Informationen zur Definition der jeweiligen Probleminstanz. Ein Beispiel eines solchen Datensatzes befindet sich in Anhang A.

Danach wird die Parametrierung des genetischen Algorithmus vorgenommen. Alle bisher angesprochenen Parameter, wie z. B. Populationsgröße oder Mutationsrate, sind über die grafische Benutzeroberfläche einstellbar.

Anschließend kommt der eigentliche genetische Algorithmus zur Dienstplanerstellung zur Ausführung. Der Evolutionszyklus wird bis zum Erreichen einer der Terminationskriterien oder bis zum manuellen Abbruch durch den Nutzer durchlaufen. Währenddessen erhält der Anwender kontinuierlich Rückmeldung zum aktuellen Suchfortschritt.

Nach Termination des Algorithmus wird schließlich der beste gefundene Dienstplan grafisch in der Benutzeroberfläche dargestellt und kann vom Anwender geprüft werden. Sollte das Ergebnis nicht zufriedenstellend ausgefallen sein, besteht die Möglichkeit, den Eingabe-Datensatz und/oder die Parametereinstellungen anzupassen und den Suchprozess erneut zu starten.

Screenshots der Anwendung sind in Anhang C zu finden.

5.2. Methodisches Vorgehen

In den nachfolgenden Abschnitten wird der genetische Algorithmus zur Dienstplanerstellung in Bezug auf die Güte des besten gefundenen Dienstplans und die Laufzeit des Algorithmus evaluiert. Dabei sollen insbesondere die in Tabelle 3 zusammengefassten Parameter und Methoden überprüft und optimiert werden.

Aufgrund der auftretenden Varianz der Messergebnisse wird der Algorithmus für jede zu prüfende Parametereinstellung oder Methode 200 mal ausgeführt und ein arithmetisches Mittel über die gemessenen Werte gebildet. Für den Vergleich werden dann diese Mittelwerte herangezogen. Alle Versuchsläufe werden dabei unter gleichen Bedingungen auf einem x86-PC mit Intel Core i5-3210M Prozessor[107] durchgeführt. Eine detaillierte Beschreibung der Umgebung ist in Anhang B angegeben.

Als Probleminstanz dienen reale Daten einer IT-Abteilung der Evonik Industries AG. Als Planungszeitraum wurde der Monat Juli ausgewählt, da für die-

[107] 2,5 GHz Zwei-Kern-Prozessor mit aktiviertem Hyper-Threading.

sen Monat für gewöhnlich die meisten Urlaubswünsche formuliert werden. Der verwendete Datensatz befindet sich in Anhang A.

5.3. Initiale Parameter

Der Algorithmus wird zunächst unter Verwendung der Methoden und Parametereinstellungen aus Tabelle 3 evaluiert. Dabei können folgende Werte ermittelt werden:

- Fitness des besten Individuums: 6,11
- Laufzeit bis zur Termination: 4084 ms

Diese Werte bilden die Basis für den Vergleich mit den nachfolgenden Versuchsläufen.

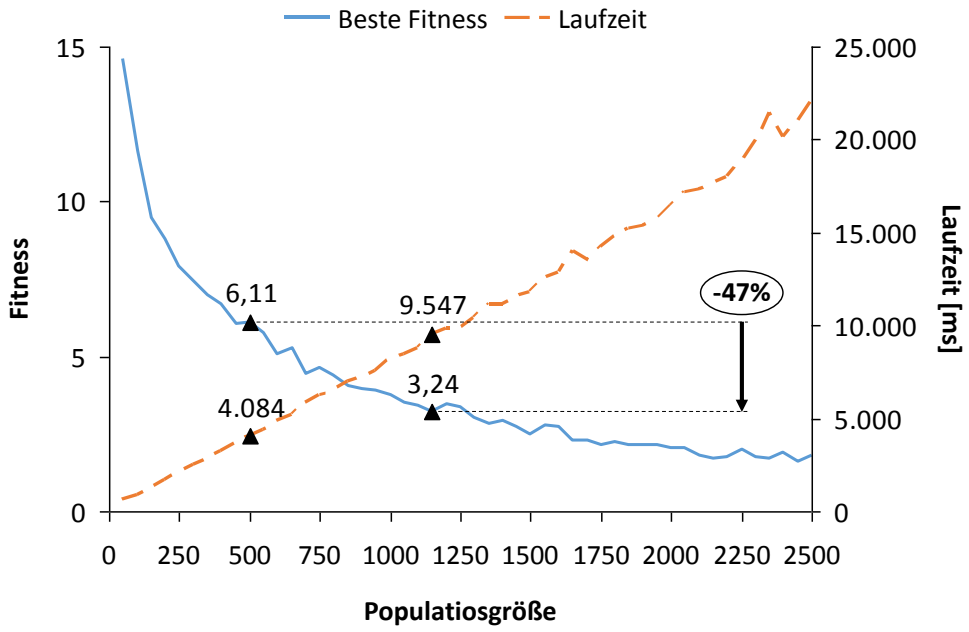
5.4. Populationsgröße

Der erste zu evaluierende Parameter ist die Populationsgröße. Dazu wird ein breiter Wertebereich von 50 bis 2500 Individuen mit einer Granularität von 50 getestet. Die Ergebnisse sind in Abbildung 11 dargestellt.

Wie vermutet steigt mit der Populationsgröße sowohl die Fitness des besten Individuums als auch die Laufzeit des Algorithmus. Während der Anstieg der Laufzeit annähernd linear verläuft, flacht der Anstieg der Fitness mit steigender Populationsgröße zunehmend ab.

Wie in der Abbildung zu erkennen ist, wurde die Populationsgröße mit 500 Individuen deutlich zu niedrig gewählt. Eine Festlegung des Wertes auf 1150 führt zu einer Verbesserung der Fitness um insgesamt 47 % auf einen Wert von 3,24 zu Kosten einer noch akzeptablen Laufzeit von ca. 9,5 Sekunden.

Dieser Parameterwert liegt mit über 1000 Individuen auch über dem Wert, der in den meisten in der Literatur gefundenen praktischen Anwendungen verwendet wird. Dies ist hauptsächlich auf die Kardinalität des zur Codierung verwendeten Alphabets zurückzuführen. Mit steigender Zahl der möglichen Allele pro Gen sinkt die Wahrscheinlichkeit, dass bei gleichbleibender Populationsgröße alle Allele in der initialen Population vertreten sind, wodurch nicht mehr alle Punkte des Suchraums alleine durch die Rekombination erreichbar



Quelle: Eigene Darstellung

Abbildung 11: Evaluierung Populationsgröße

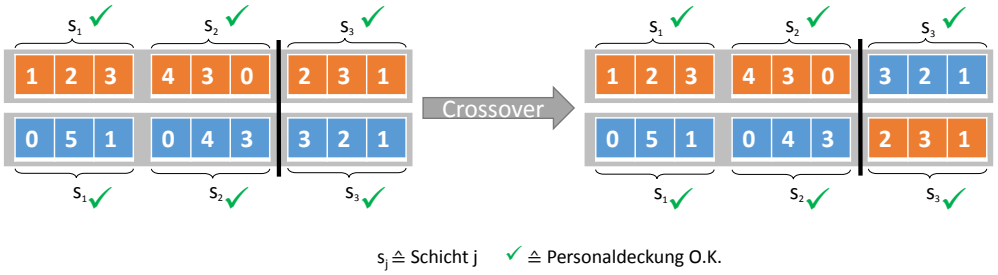
sind. In dem hier betrachteten genetischen Algorithmus wurde zur Repräsentation der Allele ein Alphabet recht hoher Kardinalität gewählt, die der Zahl aller zuweisbaren Aufgabenkombinationen entspricht. Um die Wahrscheinlichkeit zu erhöhen, dass für jedes Gen alle Allele in der Population vertreten sind, ist die Populationsgröße daher entsprechend hoch zu wählen.

5.5. Rekombinationsoperator

5.5.1. Segmentiertes Crossover

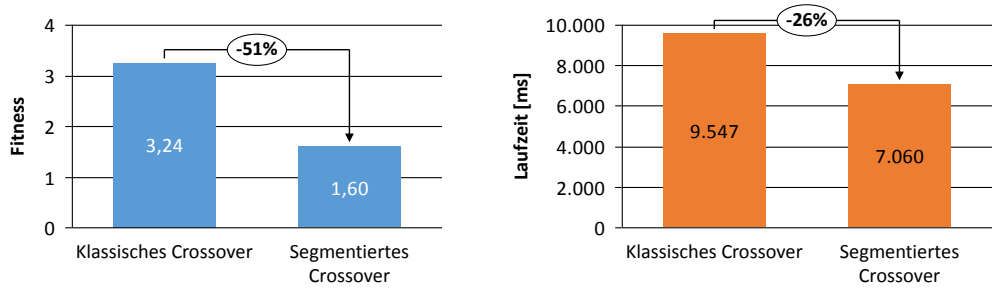
In Abschnitt 4.6.1 wurde bereits auf ein Problem des klassischen Crossoveroperators in Bezug auf die gewählte Codierung der Dienstpläne hingewiesen: Durch die Anwendung des Operators auf zwei gültige Individuen entstehen mit hoher Wahrscheinlichkeit, durch Verletzung der Nebenbedingung H1, ungültige Kindindividuen.

Dies kann durch den Einsatz eines speziellen Crossoveroperators verhindert werden, der nur Kreuzungspunkte zwischen zwei Chromosomenbereichen (Segmenten) zulässt, die jeweils die Zuordnungen in Bezug auf eine bestimmte Schicht definieren. Ein Beispiel ist in Abbildung 12 dargestellt.



Quelle: Eigene Darstellung

Abbildung 12: Auswirkung segmentiertes Crossover auf H1



Quelle: Eigene Darstellung

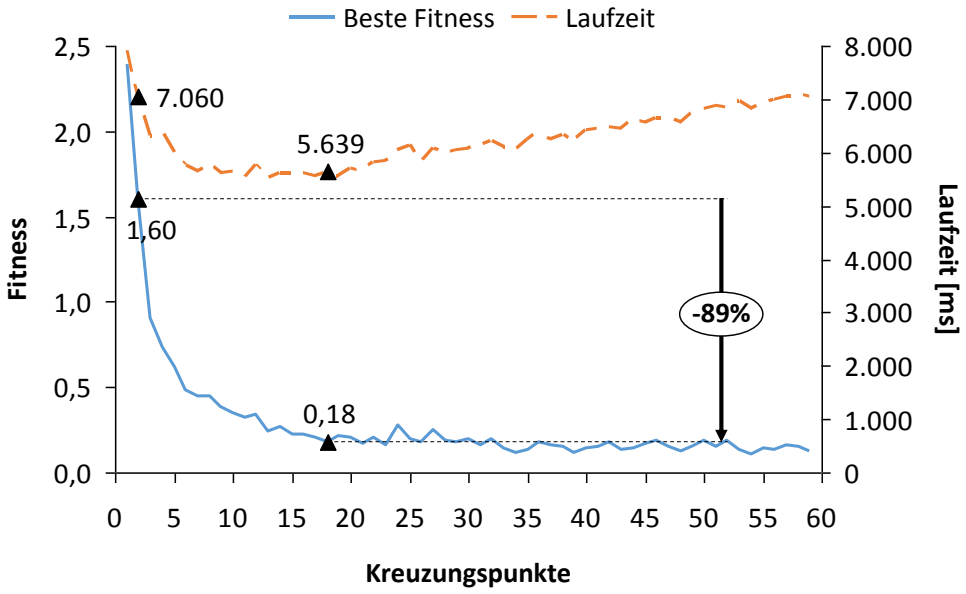
Abbildung 13: Evaluierung segmentiertes Crossover

Die in Abbildung 13 dargestellte Gegenüberstellung der Evaluierungsergebnisse des klassischen und des segmentierten Crossovers zeigt, dass der angepasste Operator bezogen sowohl auf die Fitness als auch auf die Laufzeit bessere Resultate erzielt. Die Fitness des besten Individuums konnte so um weitere 51 % verbessert und die Laufzeit um 26 % verringert werden.

5.5.2. Anzahl Kreuzungspunkte

Das bisher verwendete klassische Crossover wurde mit nur zwei Kreuzungspunkten ausgeführt, da mit steigender Anzahl eine vermehrte Erzeugung ungültiger Kindindividuen erwartet wurde. Mit Einführung des segmentierten Crossovers, dass aus gültigen Individuen die Erzeugung gültiger Nachkommen garantiert, ist dies nun hinfällig und der Parameter muss neu bestimmt werden. Der Algorithmus wurde dazu erneut unter Verwendung des segmentierten Crossovers mit 1 bis 59 Kreuzungspunkten evaluiert.

Abbildung 14 zeigt, dass mit steigender Anzahl von Kreuzungspunkten die er-



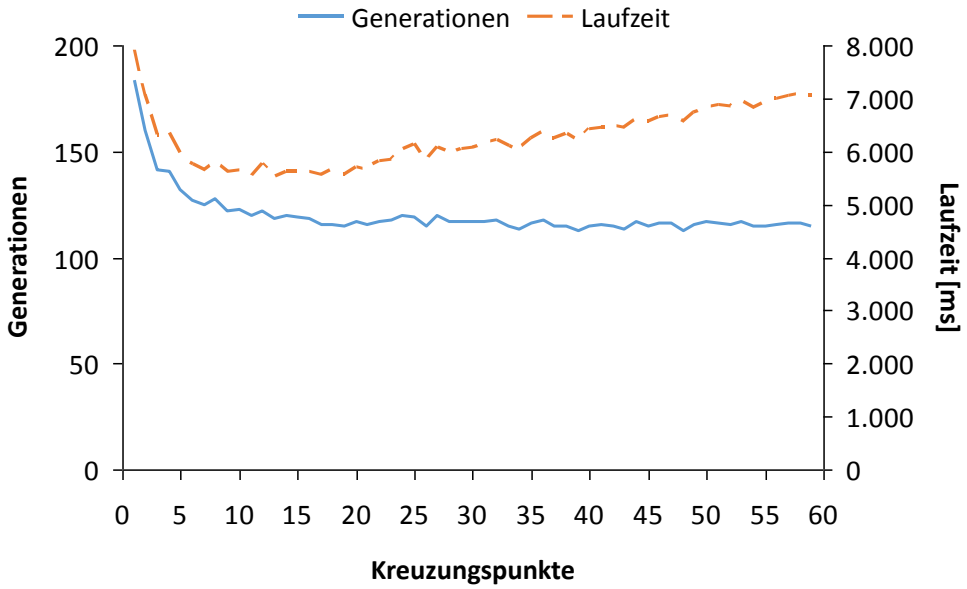
Quelle: Eigene Darstellung

Abbildung 14: Evaluierung Kreuzungspunkte

reichte Fitness des besten Individuum zunächst deutlich verbessert wird. Ein optimales Fitness-Laufzeit-Verhältnis ist bei einem Wert von ca. 18 erreicht. Durch eine weitere Erhöhung des Wertes werden dann tendenziell keine nennenswerten Verbesserungen mehr erzielt.

Mit steigender Anzahl von Kreuzungspunkten steigt auch die benötigte Rechenzeit zur Bestimmung der Kreuzungspositionen und somit die benötigte Zeit pro Generationszyklus. Die Gesamtlaufzeit des Algorithmus sinkt jedoch im Verlauf der ersten 18 Messpunkte zunächst, da die Zahl der Generationszyklen, die bis zu einer Konvergenz nötig sind, abnimmt. Nachdem sich die Anzahl nötiger Generationen nicht weiter verbessert, erhöht sich die Gesamtlaufzeit des Algorithmus mit weiter steigender Anzahl von Kreuzungspunkten wie erwartet. Dieser Zusammenhang wird in Abbildung 15 verdeutlicht.

Eine Erhöhung der Anzahl der Kreuzungspunkte von 2 auf 18 führt also zu einem deutlich besseren Konvergenzverhalten des Algorithmus, d. h. in insgesamt weniger Generationszyklen werden bessere Fitnesswerte erreicht. So kann die Fitness um 89 % auf einen Wert von 0,18 verbessert werden bei einer gleichzeitig um ca. 1,5 Sekunden verkürzten Laufzeit.



Quelle: Eigene Darstellung

Abbildung 15: Korrelation Generationen zu Laufzeit

5.6. Mutationsrate

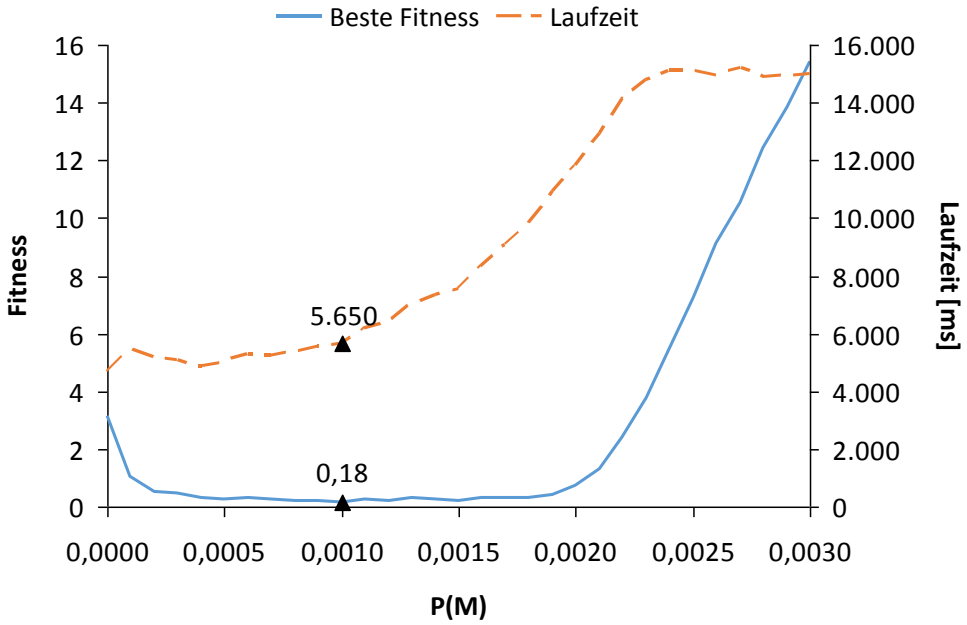
Als letzter Parameter wird abschließend die Mutationsrate $P(M)$ evaluiert. Die Wahrscheinlichkeit einer Mutation pro Gen wird dazu im Bereich von 0 bis $0,003$ mit einer Granularität von 10^{-4} getestet.

Die in Abbildung 16 dargestellten Messergebnisse zeigen, dass mit einem Wert von $0,001$ in Bezug auf die erzielten Fitnesswerte bereits eine optimale Parametereinstellung gefunden wurde.

Eine sehr geringe Mutationsrate mit $P(M) < 3 \cdot 10^{-4}$ führt aufgrund der zu geringen Diversität in der Population zu einer frühzeitigen Konvergenz des Algorithmus in ein lokales Optimum[108]. Wird die Mutationsrate hingegen mit $P(M) > 0,002$ zu hoch gewählt, wird die Bildung und Verbreitung guter Schemata so stark gestört, dass eine Konvergenz in ein Optimum gänzlich verhindert wird.

Der zuvor gewählte Wert von $0,001$ wird daher beibehalten.

[108] Vgl. Abschnitt 2.7.2.



Quelle: Eigene Darstellung

Abbildung 16: Evaluierung Mutationsrate

5.7. Zusammenfassung

Durch die Evaluierung konnten die verwendeten Parametereinstellungen und Methoden so optimiert werden, dass der Fitnesswert des besten gefunden Individuums von zuvor $6,11$ auf $0,18$ verbessert wurde. Dies wurde zu Lasten einer vergleichsweise geringen Verlängerung der Laufzeit von 4084 Millisekunden auf 5650 Millisekunden erreicht. Eine Übersicht über die optimierten Werte gibt Tabelle 4.

Parameter/ Methode	**Alter Wert**	**Optimierter Wert**
Codierung	schichtbasiert	schichtbasiert
Populationsgröße	500	*1150*
Selektion	rangbasierte Selektion	rangbasierte Selektion
Rekombination	2-Punkt-Crossover	*Segmentiertes 18-Punkt-Crossover*
$P(C)$	$0,9$	$0,9$
Mutation	Pull-/Push-Mutation	Pull-/Push-Mutation
$P(M)$	$0,001$ pro Gen	$0,001$ pro Gen
Elitismus	10 Individuen	10 Individuen

Quelle: Eigene Darstellung

Tabelle 4: Optimierte Parameter und Methoden

6. Fazit und Ausblick

Ziel dieser Arbeit war die Entwicklung und Optimierung eines genetischen Algorithmus zur Dienstplanerstellung in Unternehmen am Beispiel der Evonik Industries AG.

Zu diesem Zweck wurde das Problem zunächst analysiert. Dabei wurden sowohl Anforderungen an die Gültigkeit der Dienstpläne definiert (harte Nebenbedingungen) als auch Kriterien zur Beurteilung deren Güte gefunden (weiche Nebenbedingungen). Auf dieser Grundlage konnte das Problem als mathematisches Optimierungsmodell formuliert werden, dessen Zielfunktion den Verletzungsgrad weicher Nebenbedingungen unter Einhaltung aller harten Nebenbedingungen minimiert.

Bei dem anschließenden Entwurf des genetischen Algorithmus standen zwei zentrale Herausforderungen im Mittelpunkt:

Da genetische Algorithmen originär nicht auf die Optimierung unter Nebenbedingungen ausgelegt sind, musste zunächst eine geeignete Vorgehensweise zum Umgang mit den harten Nebenbedingungen gefunden werden. Dabei wurde versucht, möglichst viele Nebenbedingungen zu erfüllen, indem deren Verletzung implizit vermieden wird. Dies konnte für die meisten der Bedingungen durch die Wahl einer geeigneten Codierung oder durch den Entwurf speziell angepasster Initialisierungs- und Mutationsalgorithmen gewährleistet werden. Die Forderung nach der Deckung des Personalbedarfs jeder Schicht konnte durch diese Methoden jedoch nicht vollständig garantiert werden. Um die Diversität innerhalb der Population nicht zu stark einzuschränken, wurde für diese Nebenbedingung der Einsatz einer toleranten Methode gewählt. Ungültige Individuen werden so vom Algorithmus zwar zugelassen, jedoch durch die Addition von Strafkosten zu deren Fitnesswert bei der Selektion benachteiligt. Es wurde festgestellt, dass hierdurch ein mehrkriterielles Optimierungsproblem entsteht, was zu einer gesteigerten gegenseitigen Abhängigkeit der Allele in Bezug auf deren Beitrag zur Gesamtfitness des Individuums führt (Epistase).

Die zweite Herausforderung bestand darin, den Algorithmus so zu entwerfen, dass dieser bestmöglich bei einer effizienten Suche unterstützt wird. Auf Grundlage des in Abschnitt 2.3 erläuterten Schema-Theorems war bekannt, dass dies vor allem dann der Fall ist, wenn Codierung und Rekombinationsoperator so entworfen und aufeinander abgestimmt sind, dass die Entstehung und Erhaltung guter Building Blocks gefördert wird. Es wurde erkannt, dass dies erreicht wird, indem voneinander abhänge Einzelmerkmale, die zusammen zu

einer guten Fitnessbewertung eines Dienstplans beitragen, so codiert werden, dass diese auf dem Chromosom nahe beieinander positioniert sind. Der Rekombinationsoperator war dann so zu wählen, dass dieser diese Muster möglichst selten wieder aufbricht. Durch eine Analyse der Building Blocks guter Dienstpläne konnte festgestellt werden, dass eine schichtbasierte Repräsentation der Dienstpläne in Kombination mit einem 2-Punkt-Crossover diesem Prinzip am besten entspricht.

Eine abschließende Evaluierung des genetischen Algorithmus ergab weiter, dass die zuvor festgelegte Populationsgröße mit 500 Individuen deutlich zu niedrig gewählt wurde. Die Notwendigkeit einer hohen Populationsgröße von 1150 Individuen wurde auf die hohe Kardinalität des zur Codierung verwendeten Alphabets zurückgeführt.

Besonders gute Verbesserungen sowohl der erreichten Fitness als auch der Laufzeit konnten darüber hinaus durch die Einführung eines speziellen segmentierten Crossovers und die anschließende Anpassung der Anzahl der Kreuzungspunkte erzielt werden. Dies unterstreicht erneut, wie wichtig ein auf die Codierung abgestimmter Rekombinationsoperator in Hinblick auf eine effiziente Verarbeitung von Building Blocks ist.

Aufgrund des beschränkten Umfangs dieser Arbeit erfolgte die Evaluierung dabei auf Basis nur einer einzigen Probleminstanz. Eine Evaluierung des Algorithmus bezogen auf weitere Instanzen mit abweichenden Parametern (z. B. Planungszeitraum, Ausprägung der weichen Nebenbedingungen, Anzahl der Mitarbeiter, Aufgaben und Schichten), ist daher notwendig, um allgemeinere Aussagen bezüglich guter Parameterwerte und Methoden treffen zu können. Um die erzielten Ergebnisse besser einordnen zu können, sollten diese darüber hinaus mit den Ergebnissen weiterer Lösungsverfahren, wie z. B. Tabu-Suche, verglichen werden.

Bezogen auf die untersuchte Probleminstanz konnte insgesamt ein genetischer Algorithmus entworfen und implementiert werden, der mit einer hohen Wahrscheinlichkeit optimale Lösungen bei gleichzeitig geringer Laufzeit erzeugt.

A. Eingabe-Datensatz

```
<SchedulingPeriod>

    <!--========================================================================-->
    <!--                        Planning Horizon                                -->
    <!--========================================================================-->
    <!--
    The planning horizon is defined by a start date and an end date. The end
    date is inclusive
    -->
    <StartDate>2016-07-01</StartDate>
    <EndDate>2016-07-31</EndDate>


    <!--========================================================================-->
    <!--                           ShiftTypes                                   -->
    <!--========================================================================-->
    <!--
    Here we define the shift types to be assigned. The required information
    for each shift type is a unique ID, StartTime and EndTime. A description is
    optional
    -->
    <ShiftTypes>
        <Shift ID="AP">
            <Description>APAC shift</Description>
            <StartTime>00:00</StartTime>
            <EndTime>07:59</EndTime>
        </Shift>
        <Shift ID="EM">
            <Description>EMEA shift</Description>
            <StartTime>08:00</StartTime>
            <EndTime>15:59</EndTime>
        </Shift>
        <Shift ID="AM">
            <Description>AMERICAS shift</Description>
            <StartTime>16:00</StartTime>
            <EndTime>23:59</EndTime>
        </Shift>
    </ShiftTypes>


    <!--========================================================================-->
    <!--                                Tasks                                   -->
    <!--========================================================================-->
    <!--
    Here we define the tasks that can be assigned to shifts later. The
    required information for each task is a unique ID. A description is optional
    -->

    <Tasks>
        <Task ID="MCC">
                <Description>Mission Control Center</Description>
        </Task>
        <Task ID="EoD">
                <Description>Engineer on Duty</Description>
        </Task>
        <Task ID="FEV">
                <Description>Functional Expert Virtualization</Description>
        </Task>
        <Task ID="FEF">
                <Description>Functional Expert File-Service</Description>
```

```
    </Task>
    <Task ID="FEW">
            <Description>Functional Expert Windows</Description>
    </Task>
    <Task ID="FES">
            <Description>Functional Expert Security</Description>
    </Task>
    <Task ID="FEH">
            <Description>Functional Expert Hardware</Description>
    </Task>
    <Task ID="FEB">
            <Description>Functional Expert Backup</Description>
    </Task>
    <Task ID="OCS">
            <Description>On-Call Service</Description>
    </Task>
</Tasks>
<!--=========================================================================-->
<!--                            TasksCombinations                            -->
<!--=========================================================================-->
<!--
Here we define the feasible tasks combinations. Single task assignments
are always considered to be feasible and must not be defined here.
Each task-element must refer to a valid task ID.
-->
<TaskCombinations>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FEV</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FEF</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FEW</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FES</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FEH</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>EoD</Task>
        <Task>FEB</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEV</Task>
        <Task>FEF</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEV</Task>
        <Task>FEW</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEV</Task>
```

```
        <Task>FES</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEV</Task>
        <Task>FEH</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEV</Task>
        <Task>FEB</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEF</Task>
        <Task>FEW</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEF</Task>
        <Task>FES</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEF</Task>
        <Task>FEH</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEF</Task>
        <Task>FEB</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEW</Task>
        <Task>FES</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEW</Task>
        <Task>FEH</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FEW</Task>
        <Task>FEB</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FES</Task>
        <Task>FEH</Task>
    </TaskCombination>
    <TaskCombination>
        <Task>FES</Task>
        <Task>FEB</Task>
    </TaskCombination>
</TaskCombinations>

<!--============================================================================-->
<!--                              ShiftGroups                                   -->
<!--============================================================================-->
<!--
Shifts can be placed in groups to limit the scope for some constraints.
The groups are defined here. The required information for each group is
a unique ID. Each Shift element must refer to a valid shift ID.
-->
<ShiftGroups>
    <ShiftGroup ID="EMEA">
        <Shift>EM</Shift>
    </ShiftGroup>
```

```
    <ShiftGroup ID="APAC">
        <Shift>AP</Shift>
    </ShiftGroup>
    <ShiftGroup ID="AMERICAS">
        <Shift>AM</Shift>
    </ShiftGroup>
</ShiftGroups>

<!--=========================================================================-->
<!--                          Employees                                      -->
<!--=========================================================================-->
<!--
Here we define the employees to be scheduled in the planning horizon. The
only required information for each employee is a unique ID. Each employee
can get one or multiple task qualification assigned. Each task qualification
must refer to a valid task ID
-->

<!--============================== EMEA ===================================-->

<Employees>
    <Employee ID="EM1">
        <Name>Stefan</Name>
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEV"/>
            <TaskQualification ID="FEF"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM2">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEW"/>
            <TaskQualification ID="FES"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM3">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM4">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEV"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM5">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
```

```
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM6">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FES"/>
            <TaskQualification ID="FEV"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM7">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEF"/>
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM8">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM9">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEW"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM10">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEW"/>
            <TaskQualification ID="FES"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM11">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="EM12">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
```

```
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEV"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="EM13">
<TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEW"/>
        <TaskQualification ID="FEB"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="EM14">
<TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEF"/>
        <TaskQualification ID="FEH"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>

<!--============================ AMERICAS ==============================-->

<Employee ID="AM1">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEV"/>
        <TaskQualification ID="FEF"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM2">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEW"/>
        <TaskQualification ID="FES"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM3">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEH"/>
        <TaskQualification ID="FEB"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM4">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="FEV"/>
        <TaskQualification ID="FEB"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
```

```
<Employee ID="AM5">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEW"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM6">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEF"/>
        <TaskQualification ID="FEH"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM7">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEW"/>
        <TaskQualification ID="FEB"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AM8">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEH"/>
        <TaskQualification ID="FES"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>

<!--============================== APAC ==================================-->

<Employee ID="AP1">
<TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEV"/>
        <TaskQualification ID="FEF"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AP2">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
        <TaskQualification ID="FEW"/>
        <TaskQualification ID="FES"/>
        <TaskQualification ID="OCS"/>
    </TaskQualifications>
</Employee>
<Employee ID="AP3">
    <TaskQualifications>
        <TaskQualification ID="MCC"/>
        <TaskQualification ID="EoD"/>
```

```
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="AP4">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEV"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="AP5">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FES"/>
            <TaskQualification ID="FEW"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="AP6">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEF"/>
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="AP7">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEW"/>
            <TaskQualification ID="FEB"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
    <Employee ID="AP8">
        <TaskQualifications>
            <TaskQualification ID="MCC"/>
            <TaskQualification ID="EoD"/>
            <TaskQualification ID="FEH"/>
            <TaskQualification ID="FES"/>
            <TaskQualification ID="OCS"/>
        </TaskQualifications>
    </Employee>
</Employees>

<!--========================================================================-->
<!--                              EmployeeGroups                            -->
<!--========================================================================-->
<!--
Employees can be placed in groups to limit the scope for some constraints.
The groups are defined here. The required information for each group is
a unique ID. Each Employee element must refer to a valid employee ID.
-->
```

```
<EmployeeGroups>
    <EmployeeGroup ID="EMEA">
        <Employee>EM1</Employee>
        <Employee>EM2</Employee>
        <Employee>EM3</Employee>
        <Employee>EM4</Employee>
        <Employee>EM5</Employee>
        <Employee>EM6</Employee>
        <Employee>EM7</Employee>
        <Employee>EM8</Employee>
        <Employee>EM9</Employee>
        <Employee>EM10</Employee>
        <Employee>EM11</Employee>
        <Employee>EM12</Employee>
        <Employee>EM13</Employee>
        <Employee>EM14</Employee>
    </EmployeeGroup>
    <EmployeeGroup ID="APAC">
        <Employee>AP1</Employee>
        <Employee>AP2</Employee>
        <Employee>AP3</Employee>
        <Employee>AP4</Employee>
        <Employee>AP5</Employee>
        <Employee>AP6</Employee>
        <Employee>AP7</Employee>
        <Employee>AP8</Employee>
    </EmployeeGroup>
    <EmployeeGroup ID="AMERICAS">
        <Employee>AM1</Employee>
        <Employee>AM2</Employee>
        <Employee>AM3</Employee>
        <Employee>AM4</Employee>
        <Employee>AM5</Employee>
        <Employee>AM6</Employee>
        <Employee>AM7</Employee>
        <Employee>AM8</Employee>
    </EmployeeGroup>
</EmployeeGroups>

<!--=========================================================================-->
<!--                            TaskCoverRequirements                        -->
<!--=========================================================================-->
<!--
Here we define the task cover requirements for each
ShiftType-WeekDay-Combination. The required information for each cover
requirement is a reference to a valid ShiftType and a minimum of one defined
weekday. Multiple weekdays can be concatenated by semicolons. The required
information for each cover-element is a reference to valid task ID and
quantity.
-->
<TaskCoverRequirements>
    <TaskCoverRequirement ShiftType="EM" WeekDays=
    "MONDAY;TUESDAY;WEDNESDAY;THURSDAY;FRIDAY">
            <Cover>
                <Task>MCC</Task>
                <Quantity>2</Quantity>
            </Cover>
            <Cover>
                <Task>EoD</Task>
                <Quantity>1</Quantity>
```

```
        </Cover>
        <Cover>
            <Task>FEV</Task>
            <Quantity>2</Quantity>
        </Cover>
        <Cover>
            <Task>FEF</Task>
            <Quantity>2</Quantity>
        </Cover>
        <Cover>
            <Task>FEW</Task>
            <Quantity>2</Quantity>
        </Cover>
        <Cover>
            <Task>FES</Task>
            <Quantity>2</Quantity>
        </Cover>
        <Cover>
            <Task>FEH</Task>
            <Quantity>2</Quantity>
        </Cover>
        <Cover>
            <Task>FEB</Task>
            <Quantity>2</Quantity>
        </Cover>
</TaskCoverRequirement>
<TaskCoverRequirement ShiftType="AP" WeekDays=
"MONDAY;TUESDAY;WEDNESDAY;THURSDAY;FRIDAY">
        <Cover>
            <Task>MCC</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>EoD</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FEV</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FEF</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FEW</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FES</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FEH</Task>
            <Quantity>1</Quantity>
        </Cover>
        <Cover>
            <Task>FEB</Task>
            <Quantity>1</Quantity>
        </Cover>
```

```
    </TaskCoverRequirement>
    <TaskCoverRequirement ShiftType="AM" WeekDays=
    "MONDAY;TUESDAY;WEDNESDAY;THURSDAY;FRIDAY">
            <Cover>
                <Task>MCC</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>EoD</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FEV</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FEF</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FEW</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FES</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FEH</Task>
                <Quantity>1</Quantity>
            </Cover>
            <Cover>
                <Task>FEB</Task>
                <Quantity>1</Quantity>
            </Cover>
    </TaskCoverRequirement>
    <TaskCoverRequirement ShiftType="EM" WeekDays="SATURDAY;SUNDAY">
            <Cover>
                <Task>OCS</Task>
                <Quantity>1</Quantity>
            </Cover>
    </TaskCoverRequirement>
    <TaskCoverRequirement ShiftType="AP" WeekDays="SATURDAY;SUNDAY">
            <Cover>
                <Task>OCS</Task>
                <Quantity>1</Quantity>
            </Cover>
    </TaskCoverRequirement>
    <TaskCoverRequirement ShiftType="AM" WeekDays="SATURDAY;SUNDAY">
            <Cover>
                <Task>OCS</Task>
                <Quantity>1</Quantity>
            </Cover>
    </TaskCoverRequirement>
</TaskCoverRequirements>

<!--=========================================================================-->
<!--                          ShiftCoverRequirements                         -->
<!--=========================================================================-->
<!--
```

```
Here we define the shift cover requirements for each weekday. The required
information for each cover requirement is a day and a minimum of one
reference to a valid ShiftType ID
-->
<ShiftCoverRequirements>
    <DayOfWeekCoverRequirement>
        <Day>MONDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>TUESDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>WEDNESDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>THURSDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>FRIDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>SATURDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
    <DayOfWeekCoverRequirement>
        <Day>SUNDAY</Day>
        <Cover Shift="EM"/>
        <Cover Shift="AP"/>
        <Cover Shift="AM"/>
    </DayOfWeekCoverRequirement>
</ShiftCoverRequirements>

<!--============================================================================-->
<!--                        ShiftUnavailabilities                               -->
<!--============================================================================-->
<!--
Here we define which employees are unavailable on which shift groups. The
required information for each ShiftUnavailability element is a reference
to a valid EmployeeGroup ID or Employee ID alternativly, and a reference
to a valid ShiftGroup ID.
Optionally, dates can be specified to limit the unavailability to these
dates. Dates must be specified in "YYYY-MM-DD" format
-->
```

```
<ShiftUnavailabilities>
    <ShiftUnavailability>
        <EmployeeGroup>AMERICAS</EmployeeGroup>
        <ShiftGroup>EMEA</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability>
        <EmployeeGroup>AMERICAS</EmployeeGroup>
        <ShiftGroup>APAC</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability>
        <EmployeeGroup>APAC</EmployeeGroup>
        <ShiftGroup>EMEA</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability>
        <EmployeeGroup>APAC</EmployeeGroup>
        <ShiftGroup>AMERICAS</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability>
        <EmployeeGroup>EMEA</EmployeeGroup>
        <ShiftGroup>APAC</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability>
        <EmployeeGroup>EMEA</EmployeeGroup>
        <ShiftGroup>AMERICAS</ShiftGroup>
    </ShiftUnavailability>
    <ShiftUnavailability> <!-- Reise Frankfurt -->
        <Employee>AM3</Employee>
        <ShiftGroup>AMERICAS</ShiftGroup>
        <Date>2016-07-09</Date>
        <Date>2016-07-10</Date>
        <Date>2016-07-11</Date>
        <Date>2016-07-12</Date>
        <Date>2016-07-13</Date>
        <Date>2016-07-14</Date>
        <Date>2016-07-15</Date>
        <Date>2016-07-16</Date>
        <Date>2016-07-17</Date>
    </ShiftUnavailability>
</ShiftUnavailabilities>

<!--=========================================================================-->
<!--                        ShiftUnavailabilityfExceptions                   -->
<!--=========================================================================-->
<!--
Here we define exceptions for the ShiftUnavailabilities just defined. The
required information for each ShiftUnavailabilityfException element is a
reference   to a valid EmployeeGroup ID or Employee ID alternatively,
and reference to a valid ShiftGroup ID.
Optionally, dates can be specified to limit the exception to these
dates. Dates must be specified in "YYYY-MM-DD" format
-->
<ShiftUnavailabilityfExceptions>
    <ShiftUnavailabilityfException>
        <Employee>AM3</Employee>
        <ShiftGroup>EMEA</ShiftGroup>
        <Date>2016-07-11</Date>
        <Date>2016-07-12</Date>
        <Date>2016-07-13</Date>
        <Date>2016-07-14</Date>
```

```
        <Date>2016-07-15</Date>
    </ShiftUnavailabilityfException>
</ShiftUnavailabilityfExceptions>

<!--=========================================================================-->
<!--                          DayUnavailabilities                            -->
<!--=========================================================================-->
<!--
Here we define which employees are unavailable on which days. The
required information for each DayUnavailability element is a reference
to a valid EmployeeGroup ID or Employee ID alternatively, and a minimum of
on date. Dates must be specified in "YYYY-MM-DD" format
-->
<DayUnavailabilities>
    <DayUnavailability> <!-- VCE Schulung -->
        <Employee>EM3</Employee>
        <Date>2016-07-27</Date>
        <Date>2016-07-28</Date>
        <Date>2016-07-29</Date>
    </DayUnavailability>
     <DayUnavailability>  <!--VCE Schulung -->
        <Employee>EM12</Employee>
        <Date>2016-07-27</Date>
        <Date>2016-07-28</Date>
        <Date>2016-07-29</Date>
    </DayUnavailability> -->
    <DayUnavailability> <!--Klausur -->
        <Employee>EM7</Employee>
        <Date>2016-07-09</Date>
    </DayUnavailability>
    <DayUnavailability> <!-- ESK Seminar -->
        <Employee>AP3</Employee>
        <Date>2016-07-03</Date>
    </DayUnavailability>
    <DayUnavailability>
        <Employee>EM2</Employee>
        <Date>2016-07-16</Date>
        <Date>2016-07-17</Date>
    </DayUnavailability>
    <DayUnavailability>
        <Employee>AP3</Employee>
        <Date>2016-07-18</Date>
    </DayUnavailability>
</DayUnavailabilities>

<!--=========================================================================-->
<!--                           DayOffPreferences                             -->
<!--=========================================================================-->
<!--
Employees can express their preference to get certain days off. These
preferences are defined here. The required information for each
DayPreference element is reference to a valid Employee ID and a minimum of
one date. Dates must be specified in "YYYY-MM-DD" format
-->
<DayOffPreferences>
    <DayOffPreference>
        <Employee>EM1</Employee>
        <Date>2016-07-04</Date>
        <Date>2016-07-05</Date>
        <Date>2016-07-06</Date>
```

```
        <Date>2016-07-07</Date>
        <Date>2016-07-08</Date>
        <Date>2016-07-09</Date>
        <Date>2016-07-10</Date>
        <Date>2016-07-11</Date>
        <Date>2016-07-12</Date>
        <Date>2016-07-13</Date>
        <Date>2016-07-14</Date>
        <Date>2016-07-15</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>EM4</Employee>
        <Date>2016-07-04</Date>
        <Date>2016-07-05</Date>
        <Date>2016-07-06</Date>
        <Date>2016-07-07</Date>
        <Date>2016-07-08</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>EM5</Employee>
        <Date>2016-07-04</Date>
        <Date>2016-07-05</Date>
        <Date>2016-07-06</Date>
        <Date>2016-07-07</Date>
        <Date>2016-07-08</Date>
        <Date>2016-07-09</Date>
        <Date>2016-07-10</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>EM13</Employee>
        <Date>2016-07-18</Date>
        <Date>2016-07-19</Date>
        <Date>2016-07-20</Date>
        <Date>2016-07-21</Date>
        <Date>2016-07-22</Date>
        <Date>2016-07-23</Date>
        <Date>2016-07-24</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>EM14</Employee>
        <Date>2016-07-18</Date>
        <Date>2016-07-19</Date>
        <Date>2016-07-20</Date>
        <Date>2016-07-21</Date>
        <Date>2016-07-22</Date>
        <Date>2016-07-23</Date>
        <Date>2016-07-24</Date>
        <Date>2016-07-25</Date>
        <Date>2016-07-26</Date>
        <Date>2016-07-27</Date>
        <Date>2016-07-28</Date>
        <Date>2016-07-29</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>AM5</Employee>
        <Date>2016-07-11</Date>
        <Date>2016-07-12</Date>
        <Date>2016-07-13</Date>
        <Date>2016-07-14</Date>
        <Date>2016-07-15</Date>
```

```
        <Date>2016-07-16</Date>
        <Date>2016-07-17</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>AM2</Employee>
        <Date>2016-07-18</Date>
        <Date>2016-07-19</Date>
        <Date>2016-07-20</Date>
        <Date>2016-07-21</Date>
        <Date>2016-07-22</Date>
    </DayOffPreference>
    <DayOffPreference>
        <Employee>AP7</Employee>
        <Date>2016-07-04</Date>
        <Date>2016-07-05</Date>
        <Date>2016-07-06</Date>
    </DayOffPreference>
</DayOffPreferences>

<!--=========================================================================-->
<!--                            Constraints                                  -->
<!--=========================================================================-->
<!--
Here the setting for soft constraints are defined.
The required information for each ConstraintSet element is a reference to
a valid Employee ID, or EmployeeGroup ID alternatively. The existence of
a CompleteWeekends element means that this constraint get enabled for the
Employee(s). AssignmentsPerWeek limits the number of assignments per week.
-->
<ConstraintSets>
    <ConstraintSet>
        <EmployeeGroup>EMEA</EmployeeGroup>
        <CompleteWeekends/>
        <AssignmentsPerWeek>6</AssignmentsPerWeek>
    </ConstraintSet>
    <ConstraintSet>
        <EmployeeGroup>APAC</EmployeeGroup>
        <CompleteWeekends/>
        <AssignmentsPerWeek>6</AssignmentsPerWeek>
    </ConstraintSet>
    <ConstraintSet>
        <EmployeeGroup>AMERICAS</EmployeeGroup>
        <CompleteWeekends/>
        <AssignmentsPerWeek>6</AssignmentsPerWeek>
    </ConstraintSet>
</ConstraintSets>

<!--=========================================================================-->
<!--                          Global Parameters                              -->
<!--=========================================================================-->
<!--
Here the the global settings are defined.
-->
<GlobalSettings>
    <Termination>
        <StagnationLimit>50</StagnationLimit>
    </Termination>
    <Experiments>
        <PopulationSize>
            <Min>100</Min>
```

```
                <Max>105</Max>
                <Granularity>1</Granularity>
            </PopulationSize>
            <CrossoverPoints>
                <Min>1</Min>
                <Max>59</Max>
                <Granularity>1</Granularity>
            </CrossoverPoints>
            <CrossoverRate>
                <Min>0.5</Min>
                <Max>1.0</Max>
                <Granularity>0.1</Granularity>
            </CrossoverRate>
            <MutationRate>
                <Min>0.0</Min>
                <Max>0.003</Max>
                <Granularity>0.0001</Granularity>
            </MutationRate>
        </Experiments>
    </GlobalSettings>

</SchedulingPeriod>
```

B. Evaluierungsumgebung

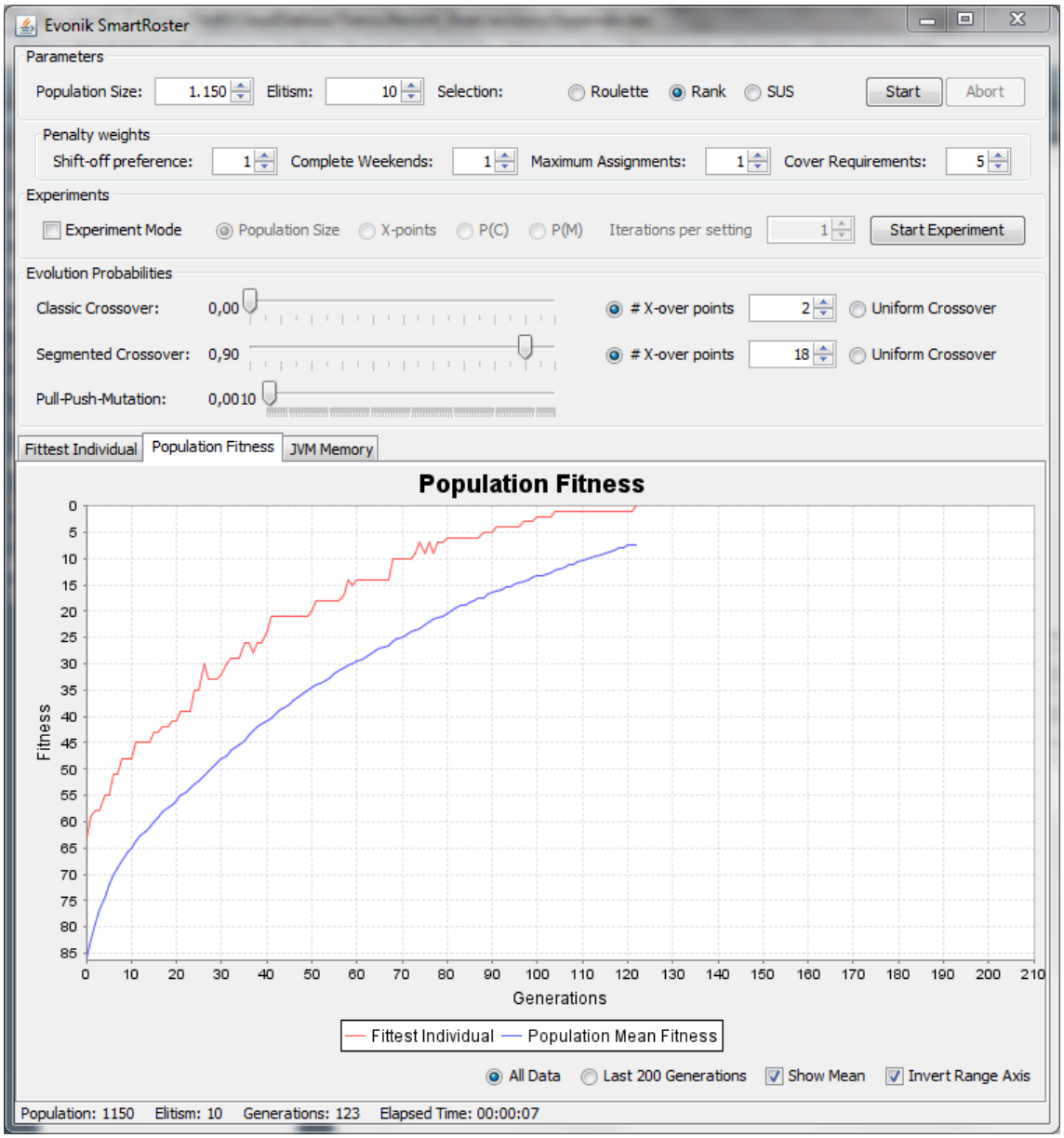
Alle Versuchsläufe wurden unter gleichen Bedingungen durchgeführt. Tabelle 5 gibt einen Überblick über die verwendete Evaluierungsumgebung.

Komponente	**Wert**
Rechnerarchitektur	x86-64
Prozessor	Intel Core i5-3210M (2,5 GHz, Dual-Core, Hyper-Threading)
Arbeitsspeicher	8 GB DDR3-1600
Betriebssystem	Microsoft Windows 7 Home Premium
Energie-Optionen	Maximale Leistung, Energiesparoptionen deaktiviert
Java Runtime Environment	Version 8, Update 66
Java Heapsize	1800 MB
Prozesspriorität	Hoch

Quelle: Eigene Darstellung

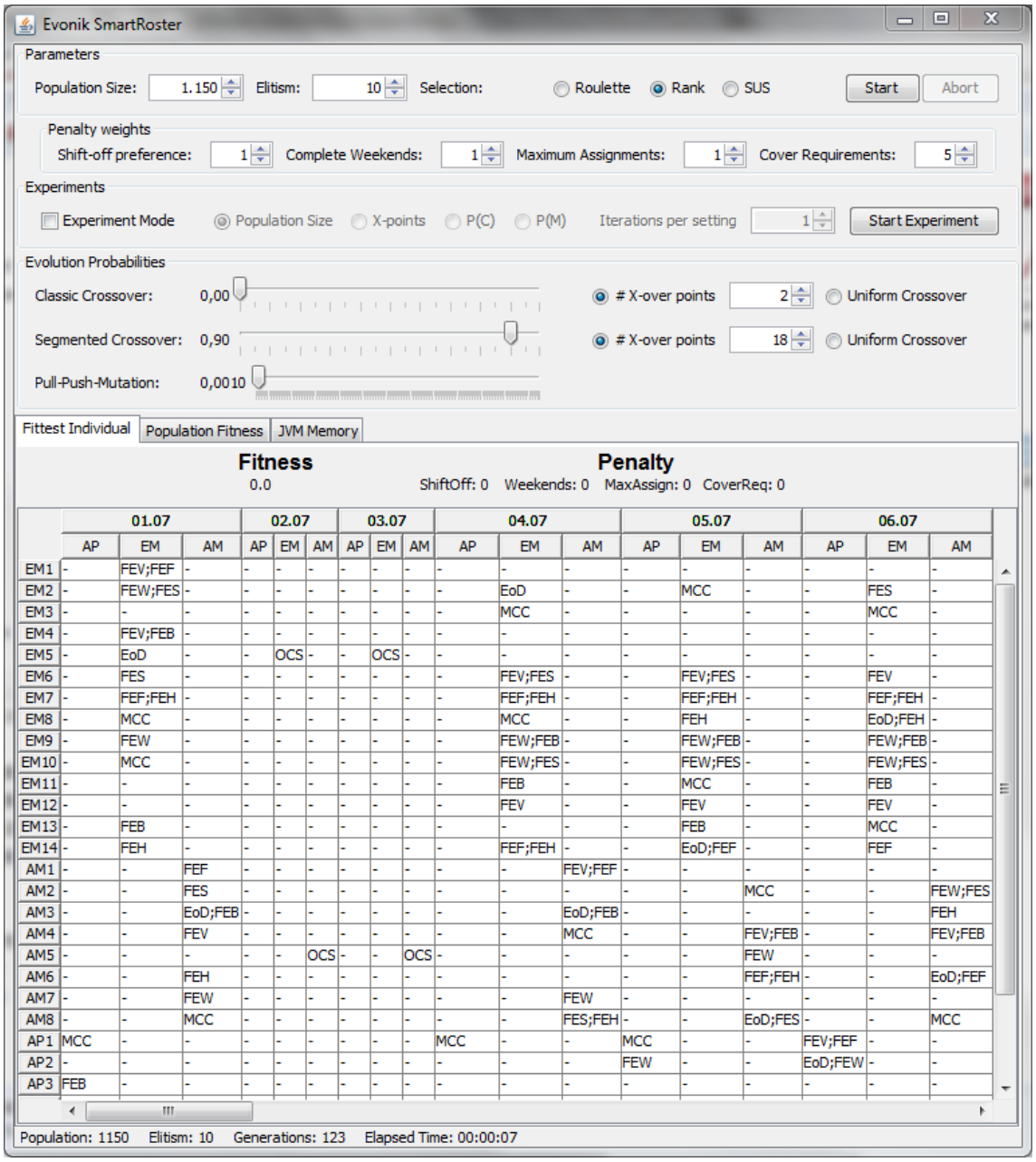
Tabelle 5: Evaluierungsumgebung

C. Screenshots



Quelle: Eigene Darstellung

Abbildung 17: Screenshot grafische Benutzeroberfläche



	01.07			02.07			03.07			04.07			05.07			06.07		
	AP	EM	AM	AP	EM	AM	AP	EM	AM	AP	EM	AM	AP	EM	AM	AP	EM	AM
EM1	-	FEV;FEF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EM2	-	FEW;FES	-	-	-	-	-	-	-	-	EoD	-	-	MCC	-	-	FES	-
EM3	-	-	-	-	-	-	-	-	-	-	MCC	-	-	-	-	-	MCC	-
EM4	-	FEV;FEB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EM5	-	EoD	-	-	OCS	-	-	OCS	-	-	-	-	-	-	-	-	-	-
EM6	-	FES	-	-	-	-	-	-	-	-	FEV;FES	-	-	FEV;FES	-	-	FEV	-
EM7	-	FEF;FEH	-	-	-	-	-	-	-	-	FEF;FEH	-	-	FEF;FEH	-	-	FEF;FEH	-
EM8	-	MCC	-	-	-	-	-	-	-	-	MCC	-	-	FEH	-	-	EoD;FEH	-
EM9	-	FEW	-	-	-	-	-	-	-	-	FEW;FEB	-	-	FEW;FEB	-	-	FEW;FEB	-
EM10	-	MCC	-	-	-	-	-	-	-	-	FEW;FES	-	-	FEW;FES	-	-	FEW;FES	-
EM11	-	-	-	-	-	-	-	-	-	-	FEB	-	-	MCC	-	-	FEB	-
EM12	-	-	-	-	-	-	-	-	-	-	FEV	-	-	FEV	-	-	FEV	-
EM13	-	FEB	-	-	-	-	-	-	-	-	-	-	-	FEB	-	-	MCC	-
EM14	-	FEH	-	-	-	-	-	-	-	-	FEF;FEH	-	-	EoD;FEF	-	-	FEF	-
AM1	-	-	FEF	-	-	-	-	-	-	-	-	FEV;FEF	-	-	-	-	-	-
AM2	-	-	FES	-	-	-	-	-	-	-	-	-	-	-	MCC	-	-	FEW;FES
AM3	-	-	EoD;FEB	-	-	-	-	-	-	-	-	EoD;FEB	-	-	-	-	-	FEH
AM4	-	-	FEV	-	-	-	-	-	-	-	-	MCC	-	-	FEV;FEB	-	-	FEV;FEB
AM5	-	-	-	-	-	OCS	-	-	OCS	-	-	-	-	-	FEW	-	-	-
AM6	-	-	FEH	-	-	-	-	-	-	-	-	-	-	-	FEF;FEH	-	-	EoD;FEF
AM7	-	-	FEW	-	-	-	-	-	-	-	-	FEW	-	-	-	-	-	-
AM8	-	-	MCC	-	-	-	-	-	-	-	-	FES;FEH	-	-	EoD;FES	-	-	MCC
AP1	MCC	-	-	-	-	-	-	-	-	MCC	-	-	MCC	-	-	FEV;FEF	-	-
AP2	-	-	-	-	-	-	-	-	-	-	-	-	FEW	-	-	EoD;FEW	-	-
AP3	FEB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Quelle: Eigene Darstellung

Abbildung 18: Screenshot grafische Benutzeroberfläche

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Adamuthe, Amol/Bichkar, Rajankumar: Genetic Algorithmic Approach for Personnel Timetabling, in: Shah, Ketan/Lakshmi Gorty, V. R./Phirke, Ajay (Hrsg.): Technology Systems and Management, Communications in Computer and Information Science, Bd. 145, Berlin/Heidelberg 2011, S. 69–76.

Aickelin, Uwe: Genetic Algorithms for Multiple-Choice Optimisation Problems, Diss., University of Wales, Swansea 1999.

Bartscher, Thomas: Gabler Wirtschaftslexikon: Stichwort „Personalbedarf“, Version 10 (2016), URL: `http://wirtschaftslexikon.gabler.de/Archiv/54931/personalbedarf-v10.html` (besucht am 01. 08. 2016).

Bartscher, Thomas/Wichert, Joachim: Gabler Wirtschaftslexikon: Stichwort „Personalplanung“, Version 9 (2016), URL: `http://wirtschaftslexikon.gabler.de/Archiv/56484/personalplanung-v9.html` (besucht am 01. 08. 2016).

Bullinger, Hans-Jörg: Technologiemanagement: Forschen und Arbeiten in einer vernetzten Welt, Berlin/Heidelberg 2013.

Burke, Edmund/Causmaecker, Patrick de/Vanden Berghe, Greet: A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem, in: McKay, Bob u. a. (Hrsg.): Simulated Evolution and Learning, Lecture Notes in Computer Science, Bd. 1585, Berlin/Heidelberg 1999, S. 187–194.

Burke, Edmund u. a.: A Memetic Approach to the Nurse Rostering Problem, in: Applied Intelligence 15 (3/2001), S. 199–214.

Burke, Edmund u. a.: The State of the Art of Nurse Rostering, in: Journal of Scheduling 7 (6/2004), S. 441–499.

Buttelmann, Maik/Lohmann, Boris: Optimierung mit Genetischen Algorithmen und eine Anwendung zur Modellreduktion (Optimization with Genetic Algorithms and an Application for Model Reduction), in: at - Automatisierungstechnik 52 (4/2004), S. 151–163.

Davidor, Yuval: Epistasis Variance: A Viewpoint on GA-Hardness, in: Rawlins, Gregory (Hrsg.): Foundations of Genetic Algorithms, Foundations of Genetic Algorithms, Bd. 1, San Mateo 1991, S. 23–35.

Dyer, Daniel W.: Evolutionary Computation in Java: A Practical Guide to the Watchmaker Framework (2015), URL: `http://watchmaker.uncommons.org/manual/` (besucht am 27. 08. 2016).

Goldberg, David E.: Genetic algorithms in search, optimization, and machine learning, Boston 1989.

Heistermann, Jochen: Genetische Algorithmen: Theorie und Praxis evolutionärer Optimierung, Teubner-Texte zur Informatik, Bd. 9, Stuttgart 1994.

Holland, John H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, Ann Arbor 1975.

Meisels, Amnon/Schaerf, Andrea: Modelling and Solving Employee Timetabling Problems, in: Annals of Mathematics and Artificial Intelligence 39 (1/2003), S. 41–59.

Raghavjee, Rushil/Pillay, Nelishia: A Study of Genetic Algorithms to Solve the School Timetabling Problem, in: Castro, Félix/Gelbukh, Alexander/González, Miguel (Hrsg.): Advances in soft computing and its applications, Lecture Notes in Computer Science, Bd. 8266, Berlin 2013, S. 64–80.

Spengler, Thomas: Modellgestützte Personalplanung, Working Paper Nr. 10, Otto-von-Guericke-Universität, Magdeburg 2006.

– Aktuelle Entwicklung in der Personaleinsatzplanung, in: Stock-Homburg, Ruth/Wolff, Birgitta (Hrsg.): Handbuch Strategisches Personalmanagement, Bd. 1, Wiesbaden 2011, S. 567–583.

Weicker, Karsten: Evolutionäre Algorithmen, Wiesbaden 2015.

Eidesstattliche Erklärung

Ich, Stefan Specht, versichere durch meine Unterschrift, dass ich die vorliegende Arbeit selbstständig erstellt habe. Andere als die angegebenen Hilfsmittel habe ich nicht verwendet.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum	Unterschrift