

**Mehrkriterielle Optimierung anhand von Evolutionären Algorithmen unter
genauerer Betrachtung des SMS-EMOA**

*Im Rahmen der Bachelorarbeit
HT 2020*

Marcel Hagen Francke
1184646
Abgabetermin: 31.01.2021

Aufgabenstellung:
PD Dr. habil. Silja Meyer-Nieberg
Apl. Professor Dr. Marko Hofmann

Betreuung:
Steven Künzel

Universität der Bundeswehr München
Fakultät für Informatik
Institut für Technische Informatik

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Optimierung und der Problemlösung durch evolutionäre Algorithmen. Es werden Grundlagen und Strukturen dieser Algorithmen dargestellt und verdeutlicht, dass eine Betrachtung mehrerer Ziele durch evolutionäre Algorithmen vorteilhaft in der Lösung komplexer Probleme sein kann. *Fighting Games*, der Kampf zwischen zwei Spielern in Echtzeit [1], stellen solch ein komplexes Problem dar. Aufbauend auf diesem Wissen wird ein Controller für die Fighting Game AI Competition [2], der einen evolutionären Algorithmus zur Entscheidungsfindung nutzt implementiert. In der Fighting Game AI Competition werden Kämpfe zweier künstlicher Intelligenzen simuliert [3]. Der entwickelte Controller *SMSEMOAai* basiert auf dem, in der KI-Videospielindustrie erfolgreichen, Schema der *Monte-Carlo-Tree-Search* [4], mehrere Spielszenarien zu simulieren, um die beste Aktionskette auszuwählen, greift aber auf die Fähigkeit von evolutionären Algorithmen zurück, große Suchräume vergleichsweise effizient zu durchsuchen [5]. Die durchgeführte Fallstudie zeigt, dass die *SMSEMOAai* erste Erfolge gegen etablierte und starke Referenzgegner der Fighting Game AI Competition erzielen konnte und präsentiert mit der Nutzung des SMS-EMOA ein bemerkenswertes Potential, um sich gegen diese Gegner in der Zukunft auch durchzusetzen. Es werden zudem Ansätze für die Weiterentwicklung der *SMSEMOAai* vorgestellt, welche vielversprechende Erkenntnisse im Bereich der Echtzeitproblemlösung durch evolutionäre Algorithmen erzielen könnten.

Inhaltsverzeichnis

1	Einleitung.....	1
1.1	Aufgabenstellung.....	1
1.2	Zielsetzung.....	1
1.3	Aktueller Stand der Forschung.....	2
1.4	Gliederung der Arbeit.....	4
2	Grundlagen.....	5
2.1	Evolutionäre Algorithmen und deren Komponenten.....	5
2.1.1	Anwendungsfelder von Evolutionären Algorithmen.....	6
2.1.2	Repräsentation des Problems durch Individuen.....	7
2.1.3	Fitnessfunktion.....	9
2.1.4	Population.....	11
2.1.5	Selektion.....	13
2.1.6	Variation.....	14
2.1.7	Initialisierung.....	16
2.1.8	Terminierung.....	17
2.2	Grenzen der monokriteriellen Optimierung.....	18
2.3	Sortierung im mehrkriteriellen Umfeld durch mehrkriterielle Optimierung..	21
2.3.1	Pareto Dominanz.....	21
2.3.2	Hypervolumen.....	23
2.3.3	Evolutionäre Mehrziel-Optimier-Algorithmen.....	24
3	Entwicklung eines Controllers für die Fighting Game AI Competition.....	26
3.1	Das Konzept der Fighting Game AI Competition.....	26
3.2	Ziele der Fallstudie.....	27
3.3	SMS-EMOA.....	29
3.3.1	Architektur des Algorithmus.....	29
3.3.2	jMetal zur Implementierung des SMS-EMOA.....	30
3.4	Das Framework der FightingICE.....	31
3.4.1	Spielsystem der FightingICE.....	31
3.4.2	Rahmenbedingungen bei der Programmierung des Controllers.....	32
3.5	Repräsentation des Problems und Modellierung der SMSEMOAai.....	33

3.6	Fitnessfunktionen	36
3.7	Kontrollparameter	38
3.8	Experimentelle Analyse	40
3.8.1	Durchführung der Experimente	41
3.8.2	Ergebnisse gegen Referenzgegner	42
3.8.3	Analyse des Verhaltens	44
3.9	Erkenntnisse der Fallstudie	45
3.10	Weiterentwicklung der SMSEMOAai	47
4	Zusammenfassung und Ausblick.....	49
5	Literaturverzeichnis.....	51
Anhang A.....		55
	Untersuchte Fitnessfunktionen in der FTG	55

Abbildungsverzeichnis

Abbildung 1: Ein vereinfachtes TSP. Stellt den Problemraum dar.	9
Abbildung 2: Das Genotyp-Phänotyp-Mapping und zwei Individuen. Stellt den Suchraum dar.	9
Abbildung 3: Transformation eines Schmetterlings zu einer Fledermaus mittels Picbreeder [35].	11
Abbildung 4: Generationszyklus eines EA [7]	12
Abbildung 5: Mutation einer Route aus dem TSP.	15
Abbildung 6: Ein mögliches Crossover, bei einer Repräsentation durch einen Bitstring.	16
Abbildung 7: Durch Eben exemplarisch dargestellte Entwicklung eines EAs [3, pp. 42 - 43].	17
Abbildung 8: Der mögliche Lösungsraum eines TSP. Die rote Linie stellt die Pareto-Front dar.	22
Abbildung 9: Verdeutlichung der Dominanzzahl (Dz).	23
Abbildung 10: Hypervolumen des TSP markiert mit Grün. Die Einzelbeiträge der Lösungen sind rot umrandet.	24
Abbildung 11: Ein Kampf zwischen Zen (P1) und Lud (P2) in der FightingICE.	27
Abbildung 12: Flowchart des SMS-EMOA.....	30
Abbildung 13: Visualisierung der drei Stufen einer Aktion (Schlag) [2].	32

Tabellenverzeichnis

Tabelle 1: Kontrollparameter der SMSEMOAai nach manuellem durchgeführten Parameter tuning.....	39
Tabelle 2: Von jMetal definierte Standardwerte der Kontrollparameter.	40
Tabelle 3: Ergebnisse des Experiments, mit Konfiguration aus Tabelle 2.	43
Tabelle 4: Ergebnisse des Experiments, mit Konfiguration aus Tabelle 1.	44

1 Einleitung

Die Optimierung ist ein Prozess, der den Alltag bestimmt. Immer wenn der Mensch über die Verbesserung eines Objekts oder eines Prozesses nachdenkt, wird unterbewusst optimiert [6]. Die Menschheit strebt nach Perfektion und zielt darauf ab, Probleme immer mit dem besten Ergebnis und unter Aufwendung der geringsten Ressourcen zu lösen [7, S. 21]. Unter diesem Aspekt wurden Algorithmen entwickelt, welche nach dem Schema einer der stärksten natürlichen Problemlöser der Welt arbeiten, dem evolutionären Prozess [8, S. 20].

1.1 Aufgabenstellung

Diese Arbeit befasst sich mit der Optimierung und Problemlösung durch Evolutionäre Algorithmen. Insbesondere wird die mehrkriterielle Optimierung durch Evolutionäre Algorithmen betrachtet, welche sich mit dem Lösen und Optimieren von Probleminstanzen befasst, die oftmals komplex aufgebaut sind und daher i. d. R. mehrere Ziele beinhalten. Dies könnte Relevanz für die Probleminstanz der Fighting Games haben, welche einen Kampf zweier Gegner bis zu einem bestimmten Ereignis simulieren [1]. Ein Kampf kann durch die immer unterschiedlichen Spielstände in Bezug auf z. B. Position der Spieler ein komplexes Problem darstellen. Die Fighting Game AI Competition untersucht dieses komplexe Problem anhand von Kämpfen zwischen KI-gesteuerten Agenten [3]. Diese Arbeit führt eine Fallstudie durch, die sich mit der Entwicklung eines Controllers für die Fighting Game AI Competition befasst. Die Entscheidungsfindung des Controllers wird ausschließlich durch den SMS-EMOA gesteuert. Der SMS-EMOA ist ein Evolutionärer Algorithmus, der mehrere konfliktäre Ziele gleichzeitig optimieren kann [9].

1.2 Zielsetzung

Ein Ziel dieser Arbeit ist es, einen Überblick über den Aufbau von evolutionären Algorithmen zu geben und Methoden darzustellen, welche genutzt werden, um Probleme mit mehreren konfliktären Zielen effektiv anhand von evolutionären Algorithmen zu lösen.

Ein weiteres Ziel ist es, anhand des entwickelten Controllers für die Fighting Game AI Competition zu überprüfen, ob mehrkriterielle evolutionäre Algorithmen, insbesondere der SMS-EMOA, für das Problem der Fighting Game AI Competition geeignet sind. Hierfür wird ein Controller entwickelt, über den anhand von Ergebnissen gegen Referenzgegner und einer Verhaltensanalyse ein erstes Fazit und ein Ausblick zur Weiterentwicklung formuliert wird.

1.3 Aktueller Stand der Forschung

Dieser Abschnitt stellt den aktuellen Stand der Forschung bezogen auf die in dieser Arbeit behandelten Themengebiete der evolutionären Mehrziel-Optimier-Algorithmen (EMOA) und der Fighting Game AI Competition dar.

Ein Anwendungsfeld von evolutionären Mehrziel-Optimier-Algorithmen ist die Medizin [10]. Yaghoobi [11] stellt ein Tool zur Früherkennung von Eierstockkrebs vor. Das vorgestellte Tool basiert auf einem evolutionären Mehrziel-Optimier-Algorithmus und kann potenzielle Biomerkmale für Eierstockkrebs erkennen. Eierstockkrebs ist für Frauen die tödlichste gynäkologische Malignität, hauptsächlich aufgrund von Einschränkungen bei der Früherkennung. Ein Vergleich der Ergebnisse gegenüber anderen Methoden zur Früherkennung bestätigt die Dominanz des vorgestellten Tools [11]. Ein weiteres Optimierungsproblem ist die *Community-Detection* in komplexen Netzwerken [12]. Solche Netzwerke können z. B. soziale Netzwerke oder das World Wide Web sein. *Community-Detection* befasst sich mit der Aufteilung dieser Netzwerke in Teilmengen, welche als *Communities* bezeichnet werden können. Es wird von einer *Community* gesprochen, wenn Knoten in dieser Teilmenge stark vernetzt sind, also eine Beziehung untereinander haben [12]. Knoten würden im genannten Beispiel der Sozialen Medien z. B. durch Profile der Nutzer repräsentiert werden. Solch eine Erkennung von Teilmengen hat zahlreiche Anwendungsfelder in der E-Commerce und der Betrugserkennung [12]. Die Lösung dieses Problems wird meistens anhand einer monokriteriellen Optimierung approximiert. Shaik et al. [12] stellen in ihrer Veröffentlichung eine Optimierung anhand von drei Zielen und einer Optimierung dieser Ziele mithilfe von drei evolutionären Mehrziel-Optimier-Algorithmen vor und erzielte damit vergleichbare und teils bessere Resultate als andere Verfahren [12], wie z. B. MOGA-net [13].

Ein neuer evolutionärer Algorithmus zur mehrkriteriellen Optimierung wurde von Falcón-Cardona [14] veröffentlicht, welcher die Abhängigkeit von EMOA in Bezug auf die Pareto-Front Geometrien ausgleicht. Die Pareto-Front wird in der mehrkriteriellen Optimierung als optimale Lösungsmenge bezeichnet. Die Performance des CRI-EMOA [14] korreliert somit nicht mit der Form der Pareto-Front. Deshalb konnte CRI-EMOA die Leistungen von etablierten EMOA, welche das Hypervolumen als Indikator nutzen [15], in Bezug auf bekannte Referenzprobleme übertreffen [14], wie z. B. DTLZ [16] und WFG [17].

Weiterhin wurde mit dem R2HCA-EMOA [15] ein Algorithmus vorgestellt, welcher den $R2$ Indikator zur Approximation des Hypervolumens nutzt und dabei aufbauend auf der grundlegenden Struktur des SMS-EMOA entwickelt wurde. Testergebnisse mit dem R2HCA-EMOA zeigen, dass er effizienter als Hypervolumen basierte EMOA optimierte und Überlegenheit gegenüber allen verglichenen EMOA in Bezug auf die Ergebnisse der Referenzprobleme aufweist [18].

Für den aktuellen Stand der Forschung bezüglich der Fighting Game AI Competition ist es zielführend, den Sieger des durchgeführten Wettkampfs zu betrachten. Der von Tang [19] programmierte Controller mit dem Namen *ERHEA_PI* [19] setzte sich gegen den viermaligen Gewinner *Thunder* durch und stellt somit die neue Referenz in der Fighting Game AI Competition dar. Der vorgestellte Controller basiert auf dem Optimierungsprozess durch den *Rolling-Horizon-Evolution-Algorithm* [20], welcher Sequenzen an Aktionen entwickelt, um eine Entscheidung zu finden, welche Aktion als nächstes ausgeführt werden soll. Zudem erstellt der Controller während des Kampfes ein Modell des Gegners welches für eine Vorhersage der Aktionen genutzt wird, um die eigenen Aktionssequenzen zu simulieren und zu evaluieren [19]. Weiterhin wurde der Bereich der Publikumsbeteiligung in Videospiele erforscht. So entwickelten Paliyawan et al. [21] zwei künstliche Intelligenzen BEAI (Believable Entertaining AI) und eAI (Entertaining AI), die unter Verwendung des MCTS agieren, jedoch ihre Stärke anhand der Beteiligungen von Zuschauern während des Kampfes variieren, um die Theorie der *Social-Facilitation* [22] zu imitieren.

1.4 Gliederung der Arbeit

Dem/der Leser/-in werden zuerst die Grundlagen über evolutionäre Algorithmen in Kapitel 2 vermittelt. Um einen Überblick über evolutionäre Algorithmen in der Praxis zu erlangen, werden die Anwendungsfelder dargestellt und kategorisiert. Anschließend werden die Komponenten von evolutionären Algorithmen vorgestellt und ihre Arbeitsweise anhand von Beispielen erläutert. Um eine Überleitung zu der in dieser Arbeit durchgeführten Fallstudie zu schaffen, werden Methoden zur mehrkriteriellen Optimierung vorgestellt.

Kapitel 3 befasst sich mit der Entwicklung des Controllers für die Fighting Game AI Competition. Hierzu wird das Framework der FightingICE dargestellt und die Funktionsweise des entwickelten Controllers erläutert. Die Fallstudie wird mit der Darstellung der Erkenntnisse und Ideen zur Weiterentwicklung des Controllers abgeschlossen. Es werden einige Ideen zur Optimierung des Controllers vorgeschlagen, die das in dieser Arbeit festgestellte Potenzial des Controllers realisieren könnten.

Anschließend wird in Kapitel 4 eine Zusammenfassung der Arbeit präsentiert und ein Ausblick für die Weiterentwicklung des Controllers gegeben.

2 Grundlagen

Evolutionäre Algorithmen (EA) befassen sich mit der Lösung und Optimierung von Problemen. Dieses Kapitel schafft eine Grundlage zum Verständnis, wie EAs funktionieren, welche Anwendungsgebiete sie abdecken und worauf bei der Arbeit mit einem EA geachtet werden muss. Zudem wird die Optimierung von mehrkriteriellen Problemen im Besonderen betrachtet. Erkenntnisse aus dieser Betrachtung dienen als Basis für die in Kapitel 3 durchgeführte Fallstudie.

2.1 Evolutionäre Algorithmen und deren Komponenten

Dieser Abschnitt erläutert, wie EA aufgebaut sind und gibt einen generellen Überblick über einzelne Komponenten, deren Funktionen und darüber, wie diese zusammenarbeiten. Um jedoch eine Definition des Begriffes „Evolutionärer Algorithmus“ zu schaffen, ist es wichtig diesen in seine Bestandteile zu zerlegen und die Wörter „Evolution“ und „Algorithmus“ zu definieren.

Das Teilwort Evolution verweist hierbei auf die Erkenntnisse von Charles Darwin`s Buch „On the Origin of Species“ [23]. In diesem Buch formuliert Darwin den Prozess der Evolution anhand von natürlicher Selektion. Demnach passt sich eine Spezies im Laufe der Zeit an seine Umgebung an. Diese Anpassung drückt sich in Form von Veränderung des Verhaltens oder der Physik des Organismus aus. Je besser sich ein Organismus an die Umgebung angepasst hat, desto länger ist es ihm möglich zu überleben. Dies wird durch Zeugung von Nachfahren realisiert. Organismen, die sich nicht an die Umgebung anpassen sterben aus [24]. Unter einem Algorithmus versteht man eine Anleitung, wie ein Problem zu lösen ist. In dieser Anleitung ist Schritt für Schritt erläutert, wie das Problem gelöst werden soll und klar definiert, welche Reihenfolge dabei eingehalten werden muss [25].

Grundlegend ist ein EA eine Technik zum automatisierten Lösen von Problemen, die sich anhand von „natürlicher“ Selektion weiterentwickelt und sich so Generation für Generation immer mehr an die Anforderung des Problems anpasst [8].

2.1.1 Anwendungsfelder von Evolutionären Algorithmen

Die Anwendungsfelder von Evolutionären Algorithmen (EA) erstrecken sich über einen weiten Raum. So werden EAs zum Beispiel in der Medizin eingesetzt, um Moleküle zu modellieren [26] oder Röntgenbilder zu segmentieren und für den Menschen interpretierbar zu gestalten [27]. Ein großes Anwendungsgebiet findet sich zudem in der Vernetzung und Kommunikation. Hier werden EAs genutzt, um die Planung und Verwaltung von Rechnernetzen zu modellieren, da Netztopologien auf viele verschiedene Probleme, wie zum Beispiel die Übertragungsgeschwindigkeit oder die Redundanz gewisser Routen aufgeteilt werden können [28]. Weitere Anwendungsgebiete sind mathematische Probleme, der Bereich der Wirtschaft und Finanzen, die Schaltkreismodellierung und viele weitere Bereiche [7].

Generell lassen sich die meisten Probleme in den oben genannten Anwendungsfeldern als schwierige Probleme klassifizieren. Schwierige Probleme werden der Komplexitätsklasse NP-hart zugeordnet, da sie vermutlich in polynomial Zeit nicht effizient lösbar sind und einen zu umfassenden Lösungsraum besitzen [29]. Ein Beispiel für ein solches Problem ist das „Travelling Salesman Problem“ (TSP), das sich mit der Frage befasst: „Welche Route durch eine Menge von Städten ist die schnellste, die ein/eine Verkäufer/-in auf seiner Reise nehmen kann?“. Der/die Verkäufer/-in sowie auch die Städte sind hierbei nur symbolisch zu interpretieren und die schnellste Route kann auch als günstigste verstanden werden, wenn die Metrik von einem Punkt zum anderen als Kosten dargestellt werden. Dieses Problem ist für die Informatik von besonderem Interesse, da es eine Analogie zum Routen von Datenpaketen durch das Internet darstellt. Zudem gibt es keinen Algorithmus wie bei allen NP-harten Problemen, der dieses Optimierungsproblem in polynomialer Zeit lösen kann [30]. Dennoch ist es mithilfe von heuristischen Verfahren und Metaheuristiken möglich, eine Lösung für solche Probleme zu approximieren. Eine in allen Wissenschaften einheitliche Definition des Begriffes Heuristik existiert nicht [31]. Diese Arbeit definiert Heuristik als die „Untersuchung der Mittel und Methoden des Aufgabenlösendens“ (Georg Pólya) [32]. Metaheuristiken sind Algorithmen, um das lokale Suchverfahren einer Problemlösung, also Heuristiken, zu steuern. Ein Merkmal von Metaheuristiken ist, dass sie generisch aufgebaut sind. Dies bedeutet, dass sie nicht speziell an ein Problem gebunden sind, sondern für die Lösung von beliebigen Optimierungsproblemen genutzt werden können [33]. Evolutionäre Algorithmen gehören zur Klasse der Metaheuristiken. Wenn es um die Wahl eines

Optimierungsverfahren, gerade im Hinblick auf Metaheuristiken geht, muss das No-Free-Lunch-Theorem (NFL) in Betracht gezogen werden. Das NFL besagt, dass über der Summe aller Probleme jedes Optimierungsverfahren im Durchschnitt gleiche Ergebnisse erzeugt. Wenn ein Algorithmus für eine Klasse von Problemen besonders gut geeignet ist, bedeutet dies, dass dieser ein Defizit an Performance für eine andere Klasse von Problemen aufweist [34].

Evolutionäre Algorithmen besitzen neben dem Lösen von Problemen und deren Optimierung, also die Verbesserung von bereits gefundenen Lösungen, noch weitere Klassen von Anwendungsgebieten. So werden EAs zudem auch für Computer gestützte Experimente genutzt, die zu kostenintensiv oder zeitaufwändig wären, um sie in der realen Welt durchführen zu lassen. Der evolutionäre Aufbau von EAs lässt vermuten, dass eine Simulation mit EAs dem echten Geschehen in der realen Welt identisch ist, dennoch ist es wichtig, vorsichtig mit einer Interpretation der durch Computer erzeugten Simulationsergebnisse zu sein, da sie immer durch simplifizierte Modellierung eine Abweichung der Ergebnisse zur realen Welt generieren. Simulationen müssen jedoch nicht immer reale Bedingungen nachbilden und können, der menschlichen Neugier zugrunde liegend, auch experimentell genutzt werden [8].

Eine Teilmenge an Problemen, die in den letzten Jahren besonders viel Aufmerksamkeit von Forschern im Bereich der Optimierung bekommt, sind mehrkriterielle Probleme (MOP). Solche Probleme befassen sich mit der Frage, wie ein Problem mit mehreren miteinander im Konflikt stehenden Zielen gelöst werden kann. Anhand des vorherigen Beispiels des TSP werden nun zum Beispiel nicht nur die Reisekosten zwischen zwei Städten in Betracht gezogen, sondern ebenfalls die Reisezeit, die der/die Verkäufer/-in zurücklegen muss. Um den Aufbau und das Verfahren zur Lösung von MOPs anhand von EAs nachvollziehen zu können, was für das Verständnis der in dieser Arbeit behandelten Fallstudie notwendig ist, ist es wichtig, die Komponenten und deren Relation von EAs zuerst anhand der monokriteriellen Optimierung zu erläutern.

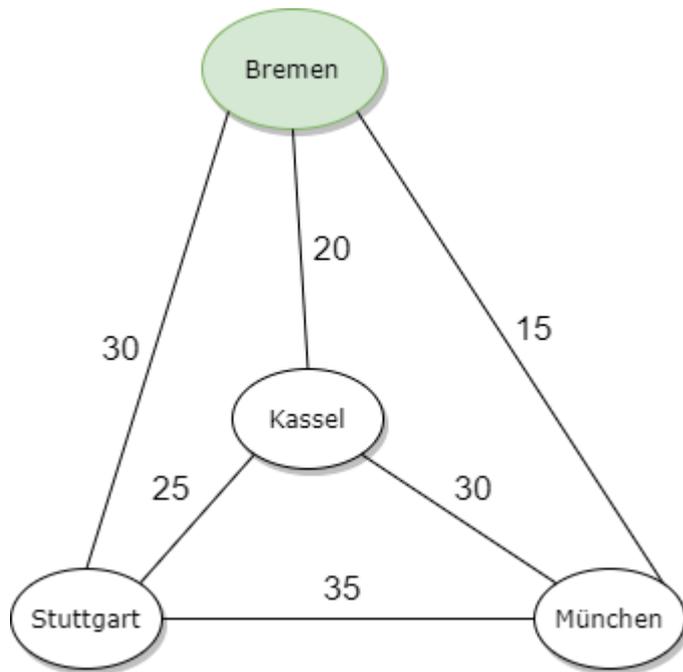
2.1.2 Repräsentation des Problems durch Individuen

Um ein Problem mit einem EA lösen zu können, gilt es beim ersten Schritt immer eine Repräsentation des Problemraums zu finden, die eine Verknüpfung zwischen der realen Welt

und dem Computer darstellt. Diese Repräsentation wird Suchraum genannt. Oftmals muss dabei das Problem aus der realen Welt simplifiziert werden, da die Repräsentation für den Computer modifizierbar sein muss. Dieser Suchraum gilt als Lebensraum für Lösungen, die sich an die Bedingungen, welche durch das Problem definiert werden, im Suchraum anpassen müssen. Dort findet die Suche und der Optimierungsprozess des EAs statt. Kandidatenlösungen oder auch Individuen werden im Suchraum als Genotypen bezeichnet und enthalten Metainformationen über die Lösung in Form von Genen. Dabei ist zu beachten, dass Kandidatenlösungen keine geeignete oder angemessene Lösung für ein Problem darstellen müssen, Kandidatenlösungen erfüllen lediglich die Bedingungen des Suchraums und befinden sich in der Menge der möglichen Lösungen. Im Laufe der Arbeit stellt der Begriff „Lösung“ meistens eine Kurzform von Kandidatenlösung dar. Die Evaluation der Lösung des Problems findet im Problemraum statt, welcher mithilfe des Suchraums eine virtuelle Lösung erstellt, die evaluiert werden kann. Dieser Prozess des Genotyp-Phänotyp-Mapping ist problemabhängig und hat einen großen Einfluss auf die Interpretation der Lösungen. Wird ein Problem ungenau repräsentiert, wurden z. B. einige Faktoren des Problems nicht mit einbezogen, hat die durch einen EA generierte Lösung einen geringeren Wert für die finale Anwendung [8, S. 28 - 30].

Ein anschauliches Beispiel für eine Repräsentation ist das bereits erwähnte Travelling Salesman Problem. In dieser vereinfachten Version des TSP möchte der/die Verkäufer/-in von Bremen aus durch Deutschland reisen und auf seiner Reise drei Städte besuchen, bevor er nach Bremen zurückkehrt (siehe Abbildung 1). Die Metriken zwischen zwei Städten dienen rein zum Verständnis des Problems. Damit der Computer dieses Problem lösen kann, ist es wichtig zu verstehen, wie eine mögliche Lösung aussehen kann. Da nach einer Route gesucht wird, die der/die Verkäufer/-in nehmen kann, wie zum Beispiel: Bremen, Kassel, München, Stuttgart, Bremen, wäre eine naheliegende Repräsentation dieser Städte eine Nummerierung mithilfe von Integer und eine Speicherung in einer Liste oder einem Array. Dies beendet das Genotyp-Phänotyp-Mapping, da der Computer nun Lösungen manipulieren kann. Mögliche Lösungen, also Individuen, werden nun als eine Liste der Größe fünf repräsentiert. Eine wichtige Einschränkung, die es bei der Initialisierung der Individuen zu beachten gilt ist, dass der Startpunkt gleich dem Ziel der Reise ist. Dies bedeutet, dass sowohl das erste als auch das letzte Gen bei allen Individuen gleich und konstant bleibt. Zudem haben die mittleren drei

Gene einen begrenzten Zahlenraum, der aus der Menge $A = \{2, 3, 4\}$ besteht und sie müssen bei Initialisierung den gesamten Raum der Menge A abdecken (siehe Abbildung 2).



Genotyp-Phänotyp-Mapping	
Bremen	1
Kassel	2
Stuttgart	3
München	4

Route 1				
1	2	4	3	1

Route 2				
1	4	3	2	1

Abbildung 1: Ein vereinfachtes TSP. Stellt den Problemraum dar.

Abbildung 2: Das Genotyp-Phänotyp-Mapping und zwei Individuen. Stellt den Suchraum dar.

2.1.3 Fitnessfunktion

Durch die Fitnessfunktion werden die Anforderungen an die Individuen dargestellt, an die sie sich anpassen müssen, um zu überleben. Individuen werden anhand ihrer Fitnesswerte vergleichbar gemacht. Außerdem dient die Fitness eines Individuums als Indikator, inwiefern eine Lösung sich verbessert. Um die Analogie der von Darwins vorgeschlagenen Evolutionstheorie fortzuführen, beschreibt die Fitness eines Organismus ihren Grad der Anpassung an ihre Umgebung, welches als Basis zur Fortpflanzung und Selektion dient [8, S. 30]. Der Unterschied in Bezug auf die Selektion zur realen Welt und dem Suchraum des EAs ist, dass Organismen in der realen Welt sich ihre Partner meistens selbst aussuchen, wohingegen Individuen im Suchraum ihre Partner i. d. R. nach ihrer Fitness zugewiesen bekommen [35].

Zur beispielhaften Erläuterung einer Fitnessfunktion dient das bereits im Abschnitt 2.1.2 definierte TSP. Um eine mögliche Route des Verkäufers zu bewerten, ist eine Summierung

aller Teil-Routen zwischen zwei Städten naheliegend. Eine mögliche Definition einer Fitnessfunktion lautet:

Sei F die Fitnessfunktion und x ein beliebiges Individuum aus der Population. Unter der Voraussetzung, dass: $b, c, d \in \{2,3,4\} \wedge a = 1$, wird die Fitness des Individuums x wie folgt berechnet:

$$F(x) = R(a, b) + R(b, c) + R(c, d) + R(d, a).$$

Es ist anzumerken, dass die „Berechnung“ einer Teilroute R grundlegend nur auf das Genotyp-Phänotyp-Mapping zurückzuführen ist. Die Anwendung dieser Funktion auf die Beispielrouten (siehe Abbildung 2) ergibt folgendes Ergebnis:

$$F(\text{Route 1}) = 115 \text{ bzw. } F(\text{Route 2}) = 95.$$

Eine Fitnessfunktion wird grundsätzlich einmal pro Generationszyklus auf alle neuen Individuen angewendet. Eine Ausnahme stellen zum Beispiel „Coevolutionary algorithms“ (CoEAs) dar [36]. Diese funktionieren nach den gleichen Prinzipien wie reguläre EAs jedoch mit dem Unterschied, dass die Evaluation der Individuen nicht aufgrund des Vergleiches ihrer Fitnesswerte, die durch eine Fitnessfunktion definiert wurden, geschieht, sondern auf der Basis der Ergebnisse von Interaktionen mit anderen Individuen. Dieser Unterschied hat zur Folge, dass sich die Beurteilung von Individuen bei CoEAs über Generationen hinweg ändern kann. Im Gegensatz zu EAs, die typischerweise mit objektiven Werten arbeiten, wird bei CoEAs von einer subjektiven Fitness gesprochen, da sich die Werte auf die sich ändernde(n) Population(en) beziehen [37].

Eine objektive Betrachtung der Individuen ist zwar die Norm bei den meisten EAs, dennoch gibt es ebenfalls evolutionäre Ansätze, die eine subjektive Bewertung der Individuen vornehmen. Ein Beispiel für einen solchen Ansatz ist die Website picbreeder.org¹. Sie basiert auf der Idee der evolutionären Kunst, genauer der Manipulation von Computer generierten Bildern. Der/die Nutzer/-in hat die Möglichkeit, sich ein zufälliges Bild generieren zu lassen, oder bereits entwickelte Werke anderer Nutzer weiter zu entwickeln. Für dieses Bild werden dem Nutzer verschiedene Variationen in Form einer Population angezeigt. Um die Individuen der Population zu bewerten, gibt es keine Fitnessfunktion. Es wird rein subjektiv nach Belieben

¹ <http://picbreeder.org/> aufgerufen am 18.12.2020

des/der Nutzers/-in entschieden, wie die Individuen bewertet werden. So ist es zum Beispiel möglich, ein Bild mit Charakteristiken eines Schmetterlings, durch Aussuchen von Eltern für die nächste Generation, welche Charakteristiken von Fledermäusen aufzeigen, einen Schmetterling in eine Fledermaus zu entwickeln (siehe Abbildung 3) [38].



Abbildung 3: Transformation eines Schmetterlings zu einer Fledermaus mittels Picbreeder [38].

Die direkte Suche nach einer optimalen Lösung oder einer Approximation, die Inspiration in der Evolutionstheorie findet, ist manchmal nicht der richtige Ansatz, um das Problem zu lösen und erfordert nicht zwangsläufig eine Fitnessfunktion [39]. Novelty Search befasst sich mit dieser Thematik und bewertet Individuen nicht danach, zu welchem Grad sie sich an den Problemraum angepasst haben, sondern belohnt die Individuen, die ein neues vorteilhaftes Verhalten aufzeigen [40]. Trotzdem ist dieses Verhalten auch eine Form der Fitness, auch wenn kein objektiver Wert zur Evaluation genutzt wird.

2.1.4 Population

Eine Population fasst die Menge der Individuen zusammen. Meistens beschreibt eine Population wie viele Individuen maximal in einer Generation existieren können. Die Anzahl an Individuen in einer Population bleibt i. d. R. über die Generationszyklen (siehe Abbildung 4) hinweg gleich und wird vor der Initialisierung definiert. Eine feste Größe der Population generiert eine begrenzte Menge an Ressourcen und es wird ein Wettkampf zwischen den Individuen forciert.

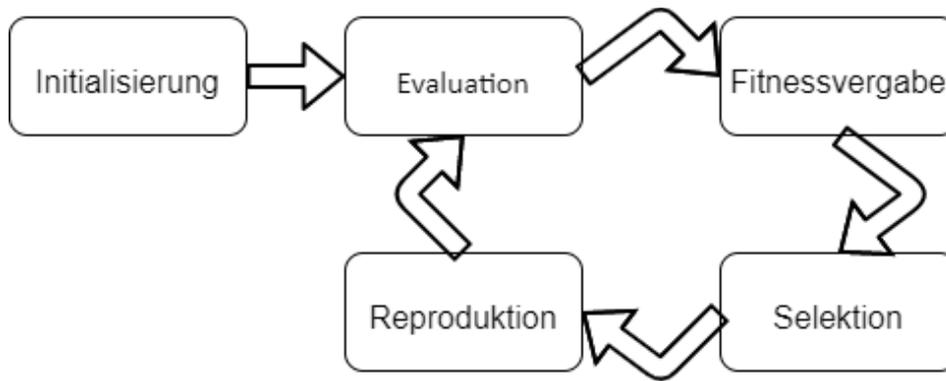


Abbildung 4: Generationszyklus eines EA [7].

Ein wichtiges Detail in Bezug auf die Population ist, dass sich nicht die Individuen verändern oder anpassen, sondern dass es die Population ist, die sich von Generation zu Generation verändert. Ein Merkmal einer Population ist ihre Diversität. Populationen sind Multimengen, daher sind Duplikate von Individuen möglich. Die Diversität beschreibt die Anzahl an Individuen, welche unterschiedlich voneinander sind. Dieses Prinzip kann ebenfalls auf weitere Kriterien, wie Fitnesswerte, Genotypen oder Phänotypen angewendet werden, wodurch es kein einheitliches Maß für Diversität gibt [8, S. 30 - 31].

Eine naheliegende Vermutung ist, dass wenn sich alle Individuen angleichen, diese Komposition von Genomen die optimale Lösung für das Problem ist und weitere Optimierungen nicht notwendig wären. Diese Aussage ist jedoch nur teilweise richtig. Eine Widerlegung dieser Vermutung bestätigt das Phänomen der vorzeitigen Konvergenz. Bezüglich der EAs bedeutet vorzeitige Konvergenz eine zu frühe Ansammlung von Lösungen um ein lokales Minimum im Zielraum herum und die damit verbundene Fokussierung einen kleinen Bestandteil des Suchraums, was die Stagnierung der Evolution und die Unerreichbarkeit eines globalen Optimums zur Folge haben könnte. Das lokale Minimum wird vermutlich eine gute Lösung bereitstellen, dennoch schließt eine weitere Optimierung in diesen Teilbereich des Suchraums das Erreichen der optimalen Lösung aus [7, S. 58 - 59].

Eine Schwierigkeit, die sich bei der Optimierung feststellen lässt, ist die Einordnung, ob ein Optimum global oder lokal ist. Dieses Phänomen lässt sich auch in der Natur beobachten. Die Entscheidung zu treffen, ob der Fokus auf die Erforschung des Suchraums (Exploration), eine Verfeinerung des bekannten Optimums (Exploitation) oder sogar ein Stoppen des Optimierungsverfahrens (Terminierung) zielführend ist, ist oftmals nicht klar. Deshalb ist es wichtig, eine Balance zwischen der Exploration und der Exploitation zu schaffen. Diese Balance

kann anhand der Goldsuche gut dargestellt werden. Das Goldschürfen in einem großen und unerforschten Bereich (Exploration), gleicht der zufälligen Suche, kann aber auch den Fund einer großen Goldader bedeuten. Die Suche in einem bereits erforschten Gebiet, in dem Gold bereits gefunden und abgebaut wurde (Exploitation), hat eine große Chance auf den Fund von Gold. Vielleicht enthält aber die Ader nicht mehr viel Gold. Bei einem Fund wird die Suche auf dieses Gebiet konzentriert und erst wieder verlassen, wenn kein Gold mehr gefunden wird (mögliche vorzeitige Konvergenz). Die Diversität korreliert dabei stark mit der Exploration, also der Suche nach neuen, unerforschten Punkten im Suchraum und wird teilweise durch Variationsoperatoren, die im Abschnitt 2.1.6 behandelt werden, gefördert. Die Exploitation schlägt eine Suche in der Nähe von vielversprechenden Lösungen vor, mit der Annahme, dass Lösungen bezogen auf ihre nachbarschaftlichen Beziehungen sowohl im Zielraum als auch im Suchraum korrelieren. Ein entscheidender Faktor ist hierbei aber die Repräsentation, welche die Interpretation einer nachbarschaftlichen Beziehung im Suchraum verfälschen könnte [41].

2.1.5 Selektion

Selektion beschreibt eine Auswahl von Individuen aufgrund von vordefinierten, einheitlichen Kriterien. Grundlegend lässt sich die Selektion in zwei Kategorien einordnen: Elternselektion und die Überlebenden-Selektion. Beide Varianten sind Teil des Generationszyklus. Der entscheidende Unterschied ist die Reihenfolge der Ausführung der Selektionsvarianten und die Intention hinter den ausgewählten Individuen. Zudem ist die Entscheidung, welche Individuen ausgewählt werden bei der Elternselektion oft stochastisch, wohingegen die Überlebenden-Selektion oftmals auf einer deterministischen Entscheidung basiert [7, S. 121 - 122].

Elternselektion

Bei dieser Variante der Selektion werden die Individuen ausgewählt, die die Eltern für die nächste Generation werden sollen. Dabei sind EAs, anders als in der Natur, nicht auf zwei Eltern beschränkt, sondern können beliebig viele Individuen als Eltern deklarieren. Eine Auswahl wird meistens anhand der Qualität, also der Fitness der Individuen getroffen. Dies

ermöglicht Individuen, die sich am besten an den Problemraum angepasst haben Variation² zu durchlaufen. Unter Betrachtung der Evolution in der Natur haben Organismen, die sich der Umgebung und ihren Gefahren am besten angepasst haben eine höhere Wahrscheinlichkeit sich fortzupflanzen. Da eine simple Auswahl der besten Individuen jedoch eine starke Exploitation eines bestimmten Bereichs des Suchraums zur Folge hätte, erhalten Individuen mit niedrigerer Fitness meistens ebenso eine geringe prozentuale Chance, Eltern der nächsten Generation zu werden. Diese Maßnahme hat eine Steigerung der Diversität zur Folge, was wiederum die Gefahr einer vorzeitigen Konvergenz verringert [8, S. 31].

Überlebenden-Selektion

Im Gegensatz zur Eltern Selektion wird die Überlebenden-Selektion typischerweise nach der Variation durchgeführt. Wie im Abschnitt 2.1.4 bereits erläutert wurde, haben die meisten EAs eine konstante Populationsgröße über ihre Generationszyklen hinweg. Dies bedingt einen Überschuss an Individuen. Einen Ausgleich dieses Überschusses zu schaffen, geschieht durch die Überlebenden-Selektion. Es gibt viele Kriterien, an denen die Individuen selektiert werden können, möglich ist z. B. die Betrachtung der Qualität oder des Alters der Individuen. Die Überlebenden-Selektion wird auch als „Motor der Evolution“ [8, S. 26] bezeichnet, weil sie die direkte Analogie der natürlichen Selektion in der realen Welt darstellt [8, S. 34].

2.1.6 Variation

Variation beschreibt den Prozess der Erstellung neuer Individuen und das Verändern von Genomen alter Individuen. Die Intention hinter der Erstellung und Veränderung von Individuen ist die Erwartung, neue und hoffentlich bessere Lösungen zu finden. Zusätzlich, wie im Abschnitt 2.1.4 bereits erwähnt, trägt die Variation häufig zur Diversität der Population bei. Um eine Variation der Individuen zu erreichen, gibt es im Wesentlichen zwei Variationsoperatoren.

² Erläuterung folgt in Abschnitt 2.1.6

Mutation

Die Mutation betrachtet den Aspekt der Veränderung eines Individuums. Für diesen Prozess wird meistens auf zufälliger Basis ein oder mehrere Individuen pro Generation ausgewählt und mutiert. Für die Mutation wird meistens ein Gen des ausgewählten Individuums zufällig manipuliert. Die Mutation bei einem Individuum wird meistens stochastisch durchgeführt. Eine stochastische Manipulation des Genoms sollte dabei das Erreichen von jedem Punkt im Suchraum garantieren, was wichtig für die Diversität der Population ist [8, S. 31 - 32].

Abbildung 5 zeigt beispielhaft die Mutation eines Individuums aus dem TSP auf Basis einer Permutations-Repräsentation.

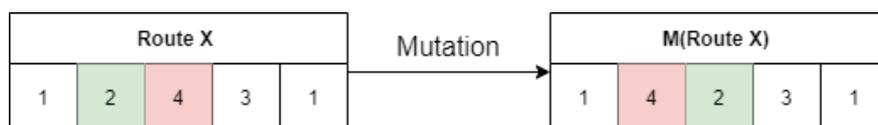


Abbildung 5: Mutation einer Route aus dem TSP.

So bewirkt der Tausch von zwei Städten eine wohlmöglich komplett neue Route, welche vielleicht die Abdeckung des Suchraums durch die Population vergrößert und eine Diversität der Lösungen zur Folge haben könnte.

Rekombination

In bisherigen Abschnitten wurde oft der Begriff der Fortpflanzung im Zusammenhang mit der Erstellung von neuen Organismen durch Eltern erwähnt. Die Rekombination bindet dieses Konzept in einen Generationszyklus eines EAs ein, mit der Erwartung, ähnlich wie beim natürlichen Prozess der Evolution, Nachkommen zu schaffen, die sich besser als Ihre Eltern an den Problemraum anpassen. Die Auswahl der Eltern nimmt die Elternselektion vor. Eine Kreuzung dieser Eltern generiert meistens ein oder zwei neue Individuen, deren Erbgut, also ihre Genome, aus den Informationen der oft stochastisch ausgewählten Gene der Eltern zusammengesetzt wird. Der spezifische Vorgang der Kreuzung von zwei Eltern, zur Erstellung zwei neuer Individuen wird als „crossover“ bezeichnet. Ein anschauliches Beispiel bei der Rekombination ist eine Kreuzung zweier Bitstrings (Abbildung 6) [8, S. 32 - 33].

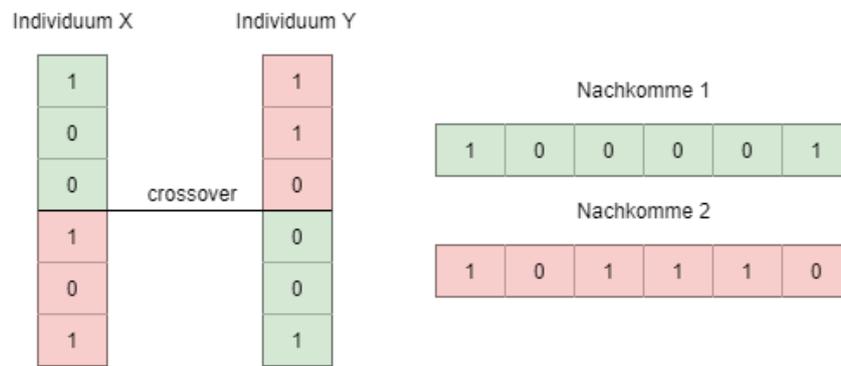


Abbildung 6: Ein mögliches Crossover, bei einer Repräsentation durch einen Bitstring.

Variationsoperatoren obliegen einer Abhängigkeit zur Probleminstanz. Dies bedeutet, dass Variationsoperatoren abhängig von der Repräsentation gewählt werden müssen [8, S. 32 - 33]. Der in Abbildung 6 dargestellte Operator würde beim Problem des TSP zu ungültigen Lösungen führen. Deshalb müsste für das TSP auf einen passenden Operator wie z. B. PMX [42] zurückgegriffen werden. Für eine detaillierte Einführung in den PMX-Operator wird auf [8, S. 80] verwiesen. Der in Abbildung 6 gezeigte Operator verdeutlicht den Sachverhalt ausreichend. Einen Bezug auf die reale Welt bietet die Kreuzung von Hunderassen. Nachkommen von gekreuzten Hunderassen übernehmen äußerliche Merkmale wie z. B. Fellfarbe, Form der Ohren oder Schwanztyp von ihren Vorfahren.

2.1.7 Initialisierung

Der erste Schritt eines EAs ist immer die Erstellung der ersten Generation. Dieser Prozess wird Initialisierung genannt und geschieht bei den meisten EAs, zu denen keine Lösung bekannt ist, zufällig. Somit wird vom Zeitpunkt der ersten Generation eine Diversität erzeugt [8, S. 42]. Nach Rahnamayan et al. [43] ist ein genaueres Betrachten dieses Prozesses aber vorteilhaft, da die Initialisierung einen Einfluss auf die Qualität der Lösung und die Geschwindigkeit der Konvergenz haben könnte. Hierfür schlagen sie eine Methode vor, die angewendet auf eine Menge von 34 numerischen Benchmark Problemen eine schnellere Konvergenz zur Lösung als eine zufallsbasierte Variante erzielt [43]. Um bei der ersten Generation Individuen mit einer besseren Fitness als bei der Zufallsinitialisierung zu generieren wurde das Prinzip des *Opposition-based learning* [44] genutzt.

Eiben bezeichnet den zusätzlichen Rechenaufwand der Initialisierung einer, in Bezug auf ihrer Fitness, besseren Anfangspopulation im Kontrast zum zufälligen Aufbau als unnötig und nicht lohnend [8, S. 42]. Seine Argumentation stützt er durch die Darstellung des *anytime behaviour*

und den natürlichen Verlauf eines EAs. Eine Suche oder Optimierung, kann laut des *anytime behaviour* jederzeit gestoppt werden, um dem/der Nutzer/-in ein Individuum wiederzugeben, welches zur Lösung der Problem Instanz verwendet werden kann. Zudem stellt er die Entwicklung bezüglich der besten Fitness einer Population exemplarisch anhand einer logarithmischen Kurve dar (siehe Abbildung 7). Diese Kurve zeigt eine deutliche Steigerung der Fitness in den ersten Generationen. Ausnahmen bezüglich der Entwicklung der Population und der dargestellten Kurve sind möglich.

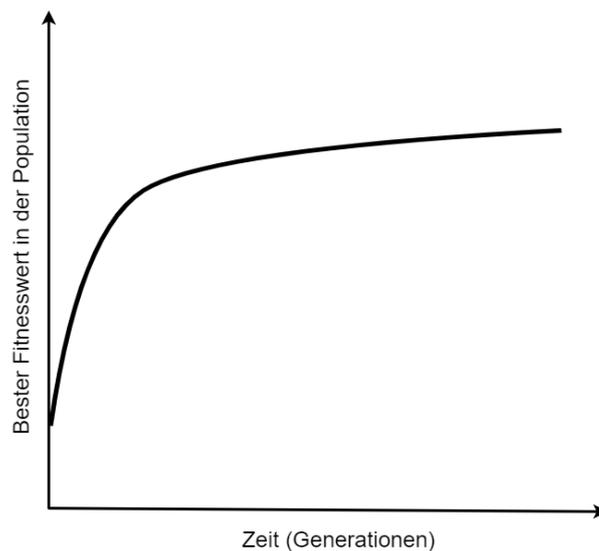


Abbildung 7: Durch Eiben exemplarisch dargestellte Entwicklung eines EAs [3, pp. 42 - 43].

Dieses Verhalten von EAs lässt den Ansatz einer Anfangsinitialisierung von besseren Individuen in einer Population fragwürdig wirken, da eine Zufallspopulation häufig wenige Generationen benötigen könnte, um ein gleiches Level an Fitness zu erreichen [8, S. 42 - 43]. Viel interessanter scheint deshalb die Festlegung eines Zeitpunkts zur Terminierung des Algorithmus.

2.1.8 Terminierung

Evolutionäre Algorithmen oder Optimierungsverfahren im Allgemeinen verfolgen im Grunde zwei Ziele. Das Erreichen des globalen Optimums einer Problem Instanz und die Verwendung des minimalen Rechenaufwands beim Optimieren. Eine Terminierung des Algorithmus beeinflusst beide Ziele. Kriterien dafür müssen deshalb mit Bedacht gewählt werden [44]. Der bestmögliche Fall, der beide Ziele erfüllt scheint die Terminierung zu einem Zeitpunkt, zu dem

ein Individuum mit adäquater Fitness gefunden wurde. Dieses Kriterium einer Terminierung setzt jedoch voraus, dass die passende Fitness einer Probleminstanz bekannt ist. Oftmals ist dies jedoch unrealistisch, da wie im Abschnitt 2.1.1 dargestellt wurde, die meisten Optimierungsprobleme NP-hart oder sogar NP-vollständig sind und somit eine bestmögliche Lösung nicht bekannt ist. Eine übliche Art, dieses Problem zu lösen, ist die Festlegung von Schwellenwerten bezüglich der Fitness zum Abbruch des Algorithmus. Ein Erreichen der optimalen Lösung wird dadurch nicht garantiert, dennoch wird bei sorgfältigem Auswählen eines Schwellenwertes eine wohlmöglich zufriedenstellende Lösung generiert. Wird der Schwellenwert nicht erreicht, ist es möglich durch ein sekundäres Abbruchkriterium, z. B. eine zeitliche Grenze, den Algorithmus zu terminieren oder eine Neubewertung des Problems durchzuführen, um einen Schwellenwert zu finden, der ebenfalls zufriedenstellend ist. Ein Faktor, welcher eine Nichterreichung des Schwellenwertes zur Folge haben könnte, ist die Verwendung von unpassenden Parameterwerten. Dies könnte z. B. die Mutationsrate sein, die die Wahrscheinlichkeit einer Mutation in einem Zyklus beschreibt. Bei besonders zeitkritischen Problemen, die ein schnelles Lösen über ein optimales Ergebnis setzen, kann ein Abbruchkriterium durch eine Festlegung eines Zeitintervalls geschehen. Häufig gewählte Kriterien sind z. B. die Prozessorlaufzeit oder die verstrichene Realzeit. Ein beliebter Ansatz ist die Festlegung einer Anzahl von Generationszyklen oder Evaluationen, die der EA ausführen soll, bevor terminiert wird [8, S. 34].

2.2 Grenzen der monokriteriellen Optimierung

Ein genereller Überblick über die Komponenten eines EAs und ihre Funktionsweise untereinander wurde im letzten Abschnitt gegeben. Dabei wurden auch mehrmals die Ziele eines EAs verdeutlicht. Das Hauptziel eines EAs ist das Finden einer Lösung zu einem Problem, bzw. die Optimierung einer bereits vorhandenen Lösung. Dies wurde anschaulich mithilfe des TSP in Abschnitt 2.1 dargestellt. Diese Probleminstanz betrachtete jedoch nur ein Ziel, das Finden der günstigsten Route, was mithilfe einer einzigen Fitnessfunktion realisierbar ist und meistens zu einer annähernd optimalen Lösung führt. EAs mit diesen Eigenschaften lassen sich grundsätzlich der monokriteriellen Optimierung (MO) zuordnen. Probleme wie z. B. die schnellste Zeit, der größte Profit oder auch wie im Fall des TSP die günstigste Route, lassen generell die Minimierung oder Maximierung eines einzigen Fitnesswertes vermuten. Oftmals

haben jedoch Probleme in der realen Welt viele verschiedene Ziele, wie in Abschnitt 2.1.1 dargestellt wurde. Eine Umwandlung mehrerer Ziele zu einem einzigen kann daher die Interpretationskraft der erstellten Lösung verfälschen. Dieser Abschnitt beschäftigt sich mit diesen und weiteren Einschränkungen, denen die MO unterliegen [45].

Ein Zusammenfassen von Zielen für eine Fitnessfunktion wird als Skalarisieren bezeichnet. Bei einer Skalarisierung werden die verschiedenen Ziele durch eine Gewichtung repräsentiert, wodurch ein lineares Optimierungsproblem mit einer gewichteten Zielfunktion entsteht [46]. Was dabei jedoch oftmals nicht beachtet wird, sind die Beziehungen, die die verschiedenen Ziele in der realen Welt untereinander haben, welche sehr komplex sein können. Am Beispiel nach der Suche des besten Hotels kann dieses Problem verdeutlicht werden. Grundsätzlich kann davon ausgegangen werden, dass ein Gast ein Hotel aussucht, das das bestmögliche Erlebnis bietet. Weiterhin wird angenommen, dass die Kosten des Hotels mit dem Faktor des Erlebnisses korrelieren. Zu definieren, was ein Erlebnis für einen Gast ist und in welchem Zusammenhang dies mit den Kosten steht, ist für eine MO viel zu komplex [45].

Eine weitere Einschränkung einer MO ist der fehlende Entscheidungsprozess nach der Optimierung, da die finale Population immer eine Lösung mit der besten Fitness enthält, auch wenn es nicht die optimale oder beste Lösung ist. Um dies jedoch zu gewährleisten, wird eine Entscheidungsfindung bereits vor der eigentlichen Optimierung durchgeführt, indem definiert wird, wie der Fitnesswert eines Individuums berechnet wird und ob dieser minimiert oder maximiert werden soll. Diese Prozedur verbietet eine spätere Auswahl von Lösungen anhand persönlicher Präferenzen und fokussiert die Optimierung auf einen/eine speziellen Nutzer/-in. Da es oftmals eine Ungleichheit zwischen persönlichen Präferenzen und der optimalen Lösung gibt, ist es wichtig, dem Gast Alternativen anhand einer Menge von Lösungen zu geben, um einen möglichst breiten Raum an Nutzerpräferenzen abzudecken. Ein Gast könnte einen kurzen, aber luxuriösen und ein weiterer Gast einen langen, aber weniger luxuriösen Aufenthalt erleben. Eine Pauschalisierung einer Präferenz ist deshalb uneffektiv [45].

Weiter ist es ebenfalls möglich, dass Ziele einer Optimierungsaufgaben konfliktär sind und deshalb nicht vereinheitlicht werden können, ohne eine Abweichung zur Realität zu riskieren [45]. Unter Betrachtung eines TSP, welches nicht wie in Abschnitt 2.1 eine einzelne Zielfunktion zur Optimierung besitzt, sondern sich neben der günstigsten Route auch an eine

Zeitmetrik zwischen den Routen richten muss, um diese zu minimieren, wird deutlich, dass beide Ziele voneinander abhängig sind und somit parallel optimiert werden sollten.

2.3 Sortierung im mehrkriteriellen Umfeld durch mehrkriterielle Optimierung

Da nun die Grenzen und Einschränkungen der MO dargestellt wurden, ist es vorteilhaft zu betrachten, wie eine Optimierung eines Problems mit mehreren konfliktären Zielen durchgeführt wird. Wie bereits in vorherigen Abschnitten erläutert, decken solche Probleme einen großen Bereich der in der realen Welt behandelten Aufgaben ab, was ein großes Interesse an der Forschung in diesem Bereich in letzten Jahren zur Folge hatte [8, S. 195]. Solche Probleme werden im Umfeld der mehrkriteriellen Optimierung behandelt. Grundsätzlich lässt sich die Aufgabe einer mehrkriteriellen Optimierung zusammenfassen auf die Bereitstellung einer Menge von optimierten Lösungen, welche Kompromisse in Bezug auf ihre Fitnessfunktionen eingehen und die jeweilige Entscheidungsfindung einer präferierten Lösung aus eben dieser Menge [44, S. 7]. Dieser Abschnitt behandelt eine wichtige Methode zur Sortierung im mehrkriteriellen Umfeld anhand der Pareto Dominanz, stellt zudem den Qualitätsindikator des Hypervolumens dar und befasst sich anschließend mit der Thematik der Evolutionäre Mehrziel-Optimier-Algorithmen (EMOA). All diese Themengebiete vermitteln ein Grundwissen, was für das Verständnis der Fallstudie und dem dafür verwendeten Algorithmus unabdingbar ist.

2.3.1 Pareto Dominanz

Die Pareto Dominanz betrachtet die Vergleichbarkeit der möglichen Lösungen im Problemraum. So wird eine Lösung A von einer weiteren Lösung B bei einem Minimierungsproblem dominiert, falls jedes Skalar des Fitnessvektors von A gleich dem von B ist und mindestens ein Wert von B kleiner ist. Formal lässt sich unter der Betrachtung von n-Dimensionen, und im Hinblick auf die Fitnessvektoren \bar{a} und \bar{b} der Lösungen A und B repräsentieren, so wird \bar{a} von \bar{b} dominiert, wenn [8, S. 196]:

$$\forall i \in \{1, \dots, n\} \bar{a}_i \geq \bar{b}_i, \wedge \exists j \in \{1, \dots, n\}, \bar{a}_j > \bar{b}_j$$

Bezogen auf das Beispiel des TSP, unter der Betrachtung der Minimierung der Kosten und Zeit, lässt sich die Pareto Dominanz durch Abbildung 8 verdeutlichen. Hier wird der Punkt C von

jeder anderen Lösung dominiert. Die Abbildung verdeutlicht zudem, dass es vier nicht dominierte Lösungen gibt.

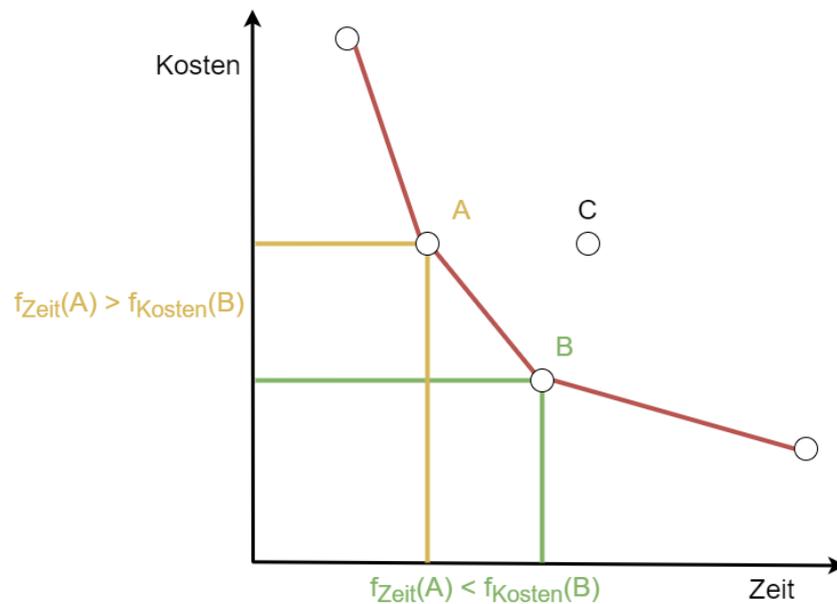


Abbildung 8: Der mögliche Lösungsraum eines TSP. Die rote Linie stellt die Pareto-Front dar.

Die Menge der Lösungen, welche nicht durch andere dominiert wird, wird als Pareto-Menge bezeichnet. Oftmals ist diese Menge eine Approximation der Pareto-Front, welche in der MO als die optimale Lösung angesehen wird. Eine Charakteristik der Pareto-Front ist, dass keine Lösungen in dieser Menge ihre Position im Raum, also ihre Fitness verändern kann (Generationsübergreifend betrachtet), ohne mindestens eine Zielfunktion negativ zu beeinflussen. Es wird meistens bei der mehrkriteriellen Optimierung angestrebt, die Pareto-Front zu approximieren. Ein Problem stellt hierbei jedoch ebenfalls die Unwissenheit der optimalen Lösung bei realen Problemen dar, was ein Erreichen oder eine Approximation der Pareto-Front nicht verifizierbar macht [8, S. 196 - 197].

Um die Pareto-Dominanz im Kontext der EA als Selektionskriterium zu nutzen, dient z. B. die Dominanzzahl als geeigneter Indikator. Die Dominanzzahl eines Individuums ist hierbei die Anzahl an Individuen, die es dominieren, wie Abbildung 9 grafisch verdeutlicht [9].

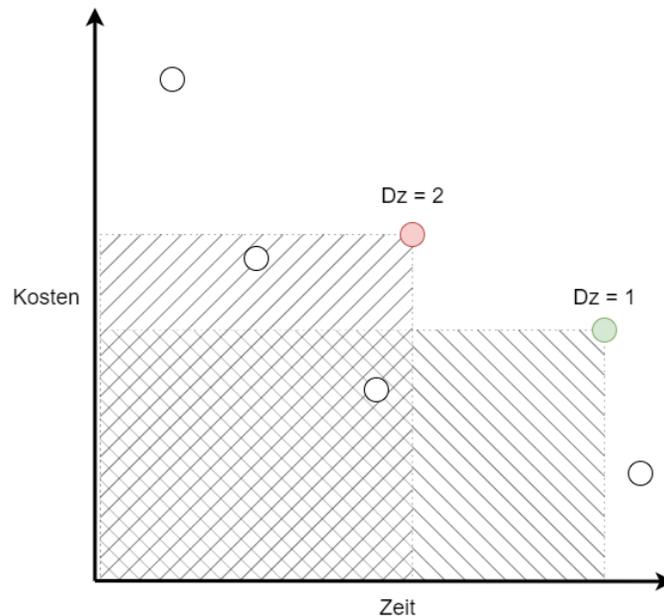


Abbildung 9: Bestimmung der Dominanzzahl (Dz).

Das Individuum mit der höchsten Dominanzzahl wird aus der Population entfernt, um den Überschuss an Individuen in der Population auszugleichen.

2.3.2 Hypervolumen

Wurde eine Pareto-Front approximiert, ist es wichtig die Lösungen in dieser Menge zu ordnen und zu sortieren. Ein weitverbreitetes Qualitätsmaß ist das Hypervolumen oder auch die S-Metrik. Das Hypervolumen nimmt diese Sortierung anhand der Einzelbeiträge der Lösungen zum gesamten Problemraum vor. Diese Berechnung benötigt jedoch einen Referenzpunkt, welcher von allen Elementen in der Lösungsmenge dominiert werden muss [9]. Dieser Referenzpunkt wird i. d. R. statisch am Anfang der Optimierung festgelegt, dennoch gibt es Studien, die einen dynamischen Ansatz eines Referenzpunktes als vielversprechend in der Optimierung ansehen [47].

Das Hypervolumen setzt sich aus den Einzelbeiträgen der Lösungen zusammen. In der mehrkriteriellen Optimierung erscheint es günstig, eine Maximierung dieser Einzelbeiträge anzustreben. Denn je weiter eine Lösung vom Referenzpunkt entfernt ist, desto besser ist die entsprechende Fitness dieser Lösung. Es gibt zudem an, wie gut die Lösungsmenge den gesamten Lösungsraum abdeckt (Abbildung 10).

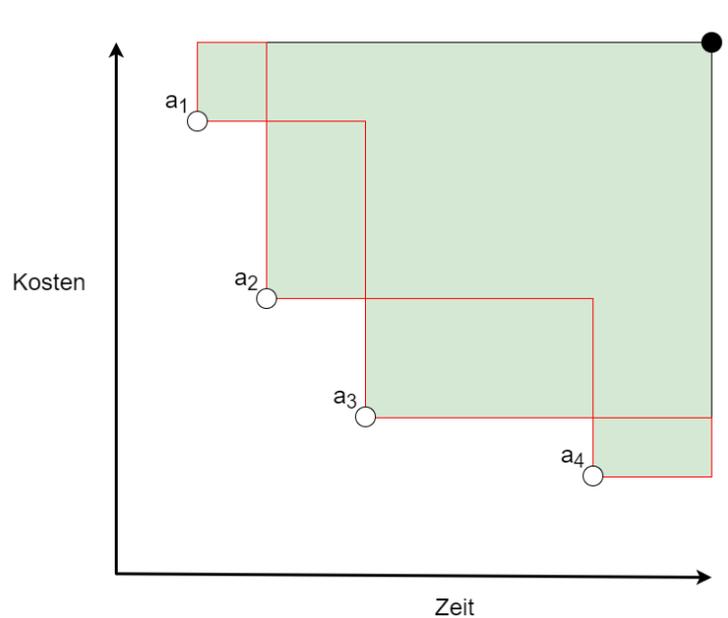


Abbildung 10: Hypervolumen des TSP markiert mit Grün. Die Einzelbeiträge der Lösungen sind rot umrandet.

Ist das Hypervolumen einer Menge bekannt, ist es wichtig, die Einzelbeiträge zu berechnen. Ein Einzelbetrag einer Lösung $a \in A$ und eines Referenzpunktes r wird durch folgende Formel berechnet:

$$HV(a, A, r) = HV(A, r) - HV(A \setminus \{a\}, r) \quad [48].$$

Ein interessanter Aspekt des Hypervolumens ist, dass die Entfernung zum Nachbarn einen Einfluss auf die Qualität der Lösung hat, da ein großer Abstand zwischen zwei Lösungen auch einen größeren Einzelbetrag dieser Lösungen ermöglicht.

2.3.3 Evolutionäre Mehrziel-Optimier-Algorithmen

Um mehrkriterielle Probleme (MOP) zu lösen, wurden Evolutionäre Mehrziel-Optimier-Algorithmen (EMOA) entwickelt, welche Teilmengen der Pareto-Front automatisiert durch eine schrittweise Annäherung mithilfe des evolutionären Prozesses approximieren [44, S. 69]. Dabei übernehmen EMOA, neben der Suche (Optimierung) der Lösungen, meist auch die Entscheidungsfindung. Bezüglich der Reihenfolge dieser Aufgaben unterscheiden EMOA drei Techniken.

A priori: Der Algorithmus benötigt einen Entscheidungsfinder (DM) bereits vor der eigentlichen Optimierung, der das auszuwählende Zielkriterium priorisiert [44, S. 3].

A Posteriori: Ein großer Suchraum wird angestrebt, um eine möglichst große Verteilung der Lösungen auf der Pareto-Front zu gewährleisten. Aus diesem großen Lösungsraum wird nach der Optimierung durch den DM entschieden, welche Lösung präferiert wird [44, S. 148].

Progressive: Eine interaktive Technik, welche meistens progressiv nach jeder Evaluation Präferenzen setzt, um die optimale Lösung zu finden [44, S. 28].

A priori und progressive Techniken verlangen genaue Zielvorstellung und Problemverständnis, weshalb die meisten EMOA können als a posteriori klassifiziert werden.

3 Entwicklung eines Controllers für die Fighting Game AI Competition

In den letzten Jahrzehnten gab es im Bereich der künstlichen Intelligenz (KI) signifikante Fortschritte in der Videospielindustrie, bedingt durch die immer leistungsfähigeren Rechner sowie die wissenschaftlichen Erkenntnisse verschiedener Lernparadigmen u. a. den Evolutionären Algorithmen. Das Ziel, sich an die Fähigkeiten der menschlichen Intelligenz anzugleichen wurde bereits übertroffen. Moderne KIs sind Menschen in runden-basierten Spielen wie Schach und Go, aber auch in Strategiespielen weit voraus [49]. Diese Spielgenre sind selten zeitkritische Anwendungen. KIs haben deshalb genügend Zeit, um Entscheidungen zu treffen oder zu optimieren. Eine Herausforderung für KIs stellen Echtzeit-Spiele dar, da die KI nur einen kurzen Zeitraum hat, um Entscheidungen zu treffen. Unter dieses Spielgenre fallen auch Fighting Games (FTG), welche den Kampf zwischen zwei Charakteren simulieren [1]. Bei diesem Genre müssen schnelle Entscheidungen getroffen werden, da der Spielstand sich schnell ändert und bisherige Berechnungen verworfen werden müssen.

Dieses Kapitel beschäftigt sich mit der Entwicklung einer KI für ein Fighting Game, welche basierend auf einem EMOA Entscheidungen trifft und einer anschließenden Fallstudie in Bezug auf die Performance und das Verhalten des Controllers. Als erstes wird das Konzept des Fighting Games erläutert, daraufhin werden die Ziele der in diesem Kapitel dargestellten Fallstudie definiert. Nach einer Beschreibung des verwendeten EMOA für den Controller wird ein Überblick über die Plattform gegeben, welche zur Entwicklung genutzt wurde und erläutert, welche Herausforderungen sich bei der Programmierung darstellten. Anschließend werden der Aufbau des Controllers, die verwendete Repräsentation der Problem Instanz, sowie die definierten Fitnessfunktionen und verwendete Kontrollparameter vorgestellt. Daraufhin werden die Rahmenbedingungen sowie die Durchführung der experimentellen Analyse des Controllers betrachtet. Die Erkenntnisse aus dieser Analyse werden am Ende des Kapitels präsentiert.

3.1 Das Konzept der Fighting Game AI Competition

Organisiert vom *Intelligent Computer Entertainment Lab. (ICE Lab.)* der Ritsumeikan Universität in Kyoto, Japan, wird jährlich seit 2013 die Fighting Game AI Competition (FTGAIC) durchgeführt. Die FTGAIC basiert dabei auf dem Computerspiel Genre der 2D Fighting Games.

Fighting Games simulieren den Kampf zwischen zwei Gegnern bis zu einem bestimmten Ergebnis, welches meistens in Form der Erschöpfung eines Spielers oder der Terminierung einer Zeitleiste realisiert wird [1]. Die Gegner bei der FTGAIC sind jedoch keine menschlichen Spieler, sondern Controller, welche mithilfe der Spiel-Engine der FightingICE (Abbildung 11) meistens in Java oder Python programmiert wurden.



Abbildung 11: Ein Kampf zwischen Zen (P1) und Lud (P2) in der FightingICE.

Die Motivation hinter der Entwicklung der FightingICE und die Organisation eines Wettbewerbs basiert auf der Frage, ob es eine vereinheitlichte künstliche Intelligenz (KI) für Fighting Games gibt und wie sie realisiert wird. Vereinheitlicht bedeutet, dass sie stark gegen jeden Gegner, unter Betrachtung jedes Charakters sein soll [3]. Ein weiteres Ziel, das Thawonmas et al. bei der Entwicklung der Plattform und dem Organisieren des Wettbewerbs verfolgen, ist die Bereitstellung eines Tools, das Studierende ermutigt, sich mit dem Thema der *Computational Intelligence* zu beschäftigen [2]. Zudem können die Studierende direkt ihren Quellcode mithilfe der FightingICE untereinander testen, was in einem Wettbewerb eine stetige Weiterentwicklung der Fighting KIs bedeutet [2].

3.2 Ziele der Fallstudie

Ein Großteil der Controller in der FTGAIC basiert auf dem Monte-Carlo-Tree-Search (MCTS) [4] Algorithmus, welcher anhand einer Baumstruktur verschiedene Spielszenarien mithilfe eines

playouts, simuliert. Ein *playout* bedeutet meistens, dass eine Simulation von Knoten durchgeführt wird, bis ein Spiel entschieden oder ein bestimmtes Zeitlimit erreicht ist. Knoten stellen einen Spielstand dar. Anschließend werden durch eine *backpropagation* alle Knoten, die durch den *playout* simuliert wurden, bewertet. Diese Bewertung der Knoten dient als Grundlage für den Entscheidungsprozess des MCTS [4]. Der MCTS Algorithmus zeigt sich als sehr starke und effiziente künstliche Intelligenz (KI) in Brettspielen wie Schach oder Go und ist dem Menschen in dieser Hinsicht bereits überlegen [50].

Unter den 14 Teilnehmern der FTGAIC im Jahr 2020 befanden sich fünf Controller, welche auf einem MCTS-Schema basierten [3]. Zudem verwendeten vier der Sieger der letzten fünf Jahre einen MCTS Ansatz. Eine große Schwäche des MCTS in einer Echtzeitanwendung ist die große Anzahl an Iterationen, die durchgeführt werden müssen, um die bestmögliche Aktion zu finden. Da die FTGAIC eine sehr zeitkritische Problem-Instanz ist, ist zu vermuten, dass trotz der Erfolge der letzten Jahre, ein Controller mit einem MCTS basierten Algorithmus nicht die optimale Lösung darstellt.

Diese Fallstudie befasst sich mit der Entwicklung eines Controllers namens *SMSEMOAai*. Die *SMSEMOAai* basiert auf der mehrkriteriellen Optimierung unter Verwendung des SMS-EMOA, welcher in Abschnitt 3.3 dargestellt wird. Dieser Algorithmus führt die Entscheidungsprozess in Kämpfen gegen andere KIs durch. Im Fokus steht die Performance des Controllers gegenüber Referenzgegnern, welche auf MCTS basieren. Wir definieren diesbezüglich folgende Forschungsfrage:

Kann der Controller sich gegen etablierte Referenzgegner durchsetzen?

Weiterhin wird eine Verhaltensanalyse durchgeführt, welche auf den Fitnessfunktionen des Controllers basiert. Über den Zeitraum der Entwicklung des Controllers wurden insgesamt zehn Fitnessfunktionen definiert, die eine Optimierung vieler verschiedener Aspekte des Kampfes anstreben. Ein interessanter Aspekt ist die Beobachtung mehrerer Kämpfe unter Betrachtung der Komposition der Fitnessfunktionen, die für die Beantwortung der ersten Forschungsfrage genutzt wurde. Die folgende Forschungsfrage kann mit der Durchführung eines adäquaten Experimentes beantwortet werden:

Lässt sich ein Verhaltensmuster der SMSEMOAai bei Kämpfen gegen den Referenzgegner NewHighlightMcts [51] erkennen?

Der Referenzgegner *NewHighlightMcts* wurde für das Experiment ausgewählt, da er von dem Entwicklerteam der FightingICE implementiert wurde und ein Benchmark für die Entwicklung eines Controllers darstellen soll.

3.3 SMS-EMOA

In der Fallstudie wurde der S-Metrik-Selektion-EMOA (SMS-EMOA) [9] zur Optimierung verwendet. Ein Merkmal des SMS-EMOA ist die Fähigkeit in hoch-dimensionalen Zielfunktionsräumen (größer drei) Optimierungen durchzuführen. Unter Betrachtung der Problem Instanz der Fighting Game AI Competition, könnte dies ein großer Vorteil sein. Viele Parameter wie z. B. die bisher durchgeführten Aktionen, die gegnerische Position oder auch die HP (Health Points) beider Spieler können betrachtet und als Grundlage für die Entscheidungsfindung verwendet werden. SMS-EMOA weist eine hohe *Usability* auf, weil keine weiteren Kontrollparameter, neben den für EAs üblichen, siehe Abschnitt 2.1.6, für die Optimierung benötigt werden und so ein tieferes Expertenwissen in diesem Bereich nicht vorausgesetzt wird. Kontrollparameter sind Werte, welche sich z. B. auf die Mutation oder Rekombination beziehen. Zusätzlich zeigt der SMS-EMOA Praxistauglichkeit, da er bereits zur Optimierung des Designs z. B. von Tragflügeln und Getriebewellen verwendet wurde [9].

3.3.1 Architektur des Algorithmus

Der SMS-EMOA strebt direkt die Maximierung des Hypervolumens, in Abschnitt 2.3.2 bereits definiert, einer Lösungsmenge an und wird deshalb als ein Indikator-basierter Algorithmus klassifiziert [9]. Der Ablauf des SMS-EMOA ist in Abbildung 12 anhand eines Flowcharts dargestellt.

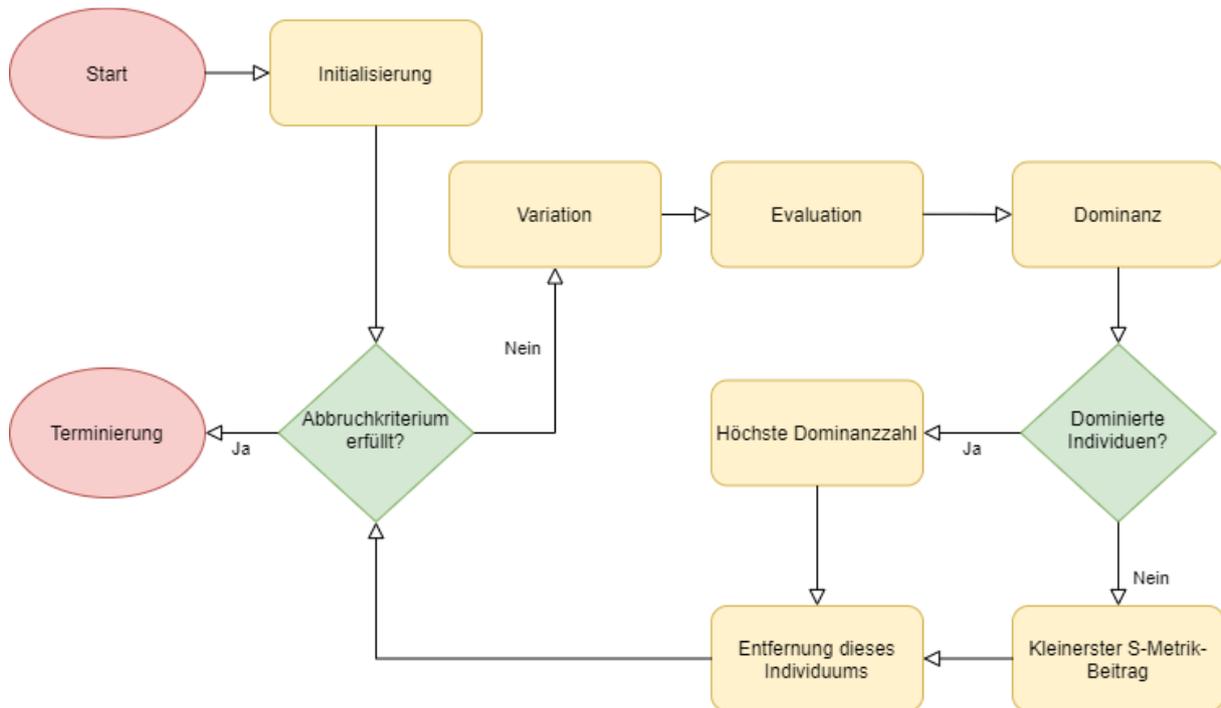


Abbildung 12: Flowchart des SMS-EMOA.

Der Algorithmus beginnt mit der Initialisierung einer Population mit vordefinierter Größe. Die Individuen der ersten Generation werden zufällig generiert. Mithilfe von Variationsoperatoren, die durch den Nutzer definiert werden müssen, wird ein neues Individuum erstellt. Da ein Überschuss bezogen auf die Populationsgröße entstanden ist, muss ein Individuum aus der Population entfernt werden. Die Entscheidung, welches Individuum entfernt werden soll, basiert auf der Einteilung der Population in eine nicht-dominierte und dominierte Menge. Ist die dominierte Menge ungleich der Leeren Menge, so wird das Individuum mit der höchsten Dominanzzahl aus der Population entfernt. Falls es keine dominierte Menge gibt, wird das Individuum mit dem kleinsten Beitrag zum Hypervolumen entfernt, was einen monoton steigenden S-Metrik Wert über die Generationen hinweg garantiert. Diese Prozedur wird wiederholt, bis das Abbruchkriterium erfüllt ist [9].

3.3.2 jMetal zur Implementierung des SMS-EMOA

Zur Implementierung des SMS-EMOA in den Controller wurde das Java-Framework jMetal [52] (Metaheuristic Algorithms in Java) verwendet. Dieses stellt neben Implementierungen bekannter EMOA auch Algorithmen zur monokriteriellen Optimierung bereit [53].

3.4 Das Framework der FightingICE

Dieser Abschnitt gibt eine Übersicht über die Plattform und die Rahmenbedingungen, die für die Programmierung eines Controllers in der Umgebung der FightingICE eingehalten werden müssen [3]. Dies ist wichtig, um die Funktionsweise des entwickelten Controllers und den Ablauf eines Kampfes in der FightingICE nachzuvollziehen.

3.4.1 Spielsystem der FightingICE

Ein Kampf in der FightingICE besteht aus drei Runden, welche jeweils 60 Sekunden dauern, was bei einer Bildwiederholungsrate von 60 Bildern bzw. Frames pro Sekunde (fps) eine Anzahl von 3.600 Frames pro Kampf ergibt. Dies bedeutet, dass ein Frame 16,67 Millisekunden angezeigt wird [3]. Der Spieler, der anschließend am wenigsten Schaden erlitten hat, gewinnt die Runde. Nach jeder Runde werden die Position und die HP der Charaktere zurückgesetzt. Den Kampf gewinnt der Spieler, welcher die Mehrheit der Runden gewonnen hat [3].

In der FightingICE gibt es drei verschiedene Charaktere: *Zen*, *Lud* und *Garnet*. Alle drei Charaktere besitzen das gleiche *Moveset* von 42 Aktionen. Ein *Moveset* beschreibt die Aktionen, die ein Spieler ausführen kann. Dies kann z. B. ein Vorwärtssprung oder ein geduckter Schlag sein. Um gewisse Aktionen wie z. B. einen Feuerball auszuführen, benötigt der Spieler eine bestimmte Anzahl an Energie (EP). Diese EP wird generiert, wenn man dem Gegner Schaden zufügt, aber auch wenn einem selbst Schaden zugefügt wird. Der Unterschied zwischen den Charakteren ist die visuelle Repräsentation durch unterschiedliche Bildmodelle (*Sprites*) und eine unterschiedliche *MotionData* in Bezug auf die 42 Aktionen. Bei der *MotionData* handelt es sich um die Informationen, wie eine Aktion aufgebaut ist. Dies sind Parameter wie *frameNumber*, welche die Dauer der Frames angibt, die benötigt werden, um diese Aktion auszuführen oder auch Parameter wie *startUp*, *active*, *recovery* oder *cancelAbleFrames*, die die Dauer der drei Stufen einer Aktion beschreiben (siehe Abbildung 13) [2].

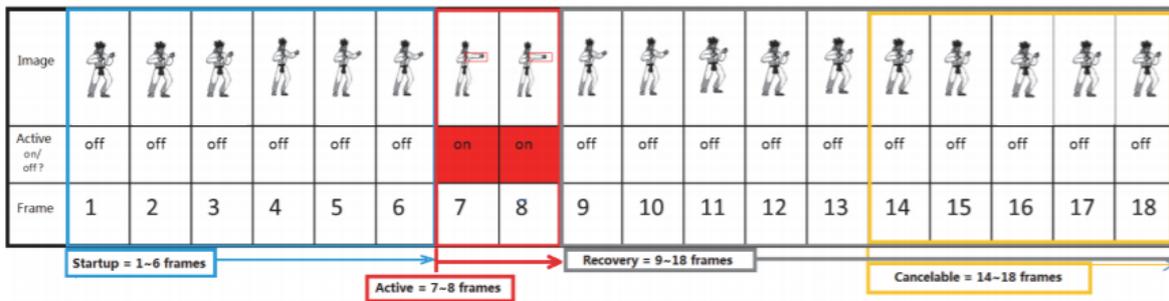


Abbildung 13: Visualisierung der drei Stufen einer Aktion (Schlag) [2].

In jedem Frame des Kampfes wird vom *GameManager*, der für die Initialisierung der Kämpfe zuständig ist und den Ablauf des Kampfes steuert, eine *FrameData* erstellt. Die *FrameData* enthält Informationen über den derzeitigen Zustand des Spiels. Hierzu gehören Informationen wie die Position, die HP oder auch die derzeitige Aktion beider Spieler. Diese Informationen werden mit einer Verzögerung von 15 Frames in jedem Frame an beide Controller übergeben, um die menschliche Reaktionszeit zu simulieren. Wie die *FrameData* ausgewertet wird, hängt von der Programmierung und dem gewählten KI-Ansatz des Controllers ab [3].

3.4.2 Rahmenbedingungen bei der Programmierung des Controllers

Die *FightingICE* stellt durch die Schnittstelle *AllInterface* Methoden zur Verfügung, die zur Programmierung des Controllers genutzt werden müssen [1]. Eine Auflistung der wichtigsten Methoden mit einer kurzen Beschreibung verschafft einen Überblick über den Prozess der Datenverarbeitung und Entscheidungsfindung der Controller:

```
initialize(GameData gd, boolean playerNumber) :
```

Diese Methode wird nur einmal am Anfang des Kampfes aufgerufen und initialisiert die KI. Die *GameData* enthält Informationen, welche sich anders als bei der *FrameData*, über den Zeitraum des Matches nicht ändern. Dazu gehören Information wie z. B. die *MotionData* des ausgewählten Charakters oder die Namen der Controller. Die *playerNumber* zeigt dem Controller an, welcher Spieler er ist; *true* für P1 und *false* für P2 [54].

```
getInformation(FrameData fd, boolean isControl) :
```

In jedem Frame wird diese Methode von der *Fighting* Instanz aufgerufen, um der KI die Informationen (*FrameData*) bereitzustellen. Das *isControl* Flag gibt an, ob der Charakter eine Aktion ausführen kann [54].

```
processing() :
```

Die *processing* Methode wird in jedem Frame ausgeführt und ist für die Datenverarbeitung des Controllers verantwortlich. Hier findet der Entscheidungsprozess der KI für die nachfolgende Aktion statt [54].

`input () :`

Der *input* wird von der Fighting Instanz nach 16,67 Millisekunden abgerufen und gibt den nächsten *Key* zurück. Eine Kombination von *Keys* bewirkt die Ausführung einer Aktion [54].

Ein typischer Ablauf nach der Initialisierung des Controllers, welcher jeden Frame, also alle 16,67 Millisekunden durchgeführt wird, ist zuerst die Informationsbeschaffung der aktuellen (15 Frames alten) *FrameData* durch die *getInformation* Methode. Anschließend wird die Verarbeitung (*processing*) dieser Daten und eine Eingabe (*input*) der auszuführenden Aktion durchgeführt. Dadurch, dass in jedem Frame eine Entscheidung getroffen werden muss, besteht eine begrenzte Rechenzeit, die bei der Modellierung der KI zu beachten ist [2].

Die FightingICE stellt eine Simulationsklasse zur Verfügung, welche genutzt werden kann, um eine neue *FrameData* zu erstellen, die dann analysiert wird. Zudem ist es möglich durch den Simulator einen Kampf auf Basis gegebener *FrameData* auch weiter zu simulieren, ohne dass Aktionen von Charakteren übergeben werden müssen [54].

3.5 Repräsentation des Problems und Modellierung der SMSEMOAai

Der erste Schritt, um eine Repräsentation des Problems zu finden, bereits in Abschnitt 2.1.2 erläutert, ist die genaue Analyse und Modellierung des Problems. Bei der Repräsentation ist zudem darauf zu achten, dass die Problem Instanz so weit simplifiziert ist, dass der Computer mögliche Lösungen durch Variation verändern kann. Einfache Datentypen (Integer, Bitstring etc.) werden oft für eine Kodierung genutzt [8, S. 49]. Die Formulierung des Problems und die Modellierung einer möglichen Lösung, sowie die Simplifizierung der Problem Instanz waren die Hauptaufgaben, welche bei der Programmierung des Controllers für die FTGAIC in Bezug auf die Repräsentation zu lösen waren.

Da in jedem Frame, also 16,67 Millisekunden, in der FightingICE eine Entscheidung getroffen werden muss, ist es notwendig, dass in diesem Zeitraum eine Lösung für das Problem gefunden wird. Dies bedeutet, dass in jedem Frame der SMS-EMOA ausgeführt werden muss. Dies setzt eine kleine Population und wenige Evaluationen voraus, da die Rechenzeit sonst

den Schwellenwert von 16,67 Millisekunden überschreitet. Die *SMSEMOAai* repräsentiert das Problem als die Auswahl optimaler Aktionen aus einem Aktionspool von insgesamt 42, unter Berücksichtigung der derzeitigen *FrameData*. Da die FightingICE, wie in Abschnitt 3.4 dargestellt, die *FrameData* mit einer Verzögerung von 15 Frames dem Controller übergibt, ist es wichtig diese Verzögerung auszugleichen, damit der Algorithmus die Optimierung auf Grundlage der derzeitigen *FrameData* durchführen kann. Hierfür berechnet die *SMSEMOAai* mithilfe der Simulator Klasse die nächsten 15 Frames auf Basis des vorhandenen Spielstands vor der Initialisierung des SMS-EMOA voraus. Diese Simulation approximiert zwar nur die derzeitige *FrameData*, da innerhalb des Zeitraums der 15 Frames bereits Aktionen von beiden Charakteren durchgeführt werden könnten, diese Approximation genügt jedoch wie später deutlich wird.

Um unmögliche Aktionen wie z. B. das Ausführen einer Attacke in der Luft, während der Controller sich auf den Boden befindet zu vermeiden, filtert die *SMSEMOAai* vor der Initialisierung auf Basis der vorhandenen EP und dem derzeitigen Zustand (Boden, Luft etc.) Aktionen aus, die möglich sind. Neben der Vermeidung von unmöglichen Kombinationen schränkt dies den Suchraum an möglichen Zusammenstellungen von Individuen ein, was wiederum ein schnelleres Approximieren der Pareto-Front zur Folge hat.

Ein Individuum bei der *SMSEMOAai* besteht aus einer Kombination von drei Aktionen, welche durch jeweils eine Ganzzahl repräsentiert werden. Diese verweist auf den Index, der in einem Array gespeicherten Aktionen. Durch die 42 möglichen Aktionen bedeutet eine solche Repräsentation einen maximalen Suchraum von 42^3 möglichen Lösungen. Eine Evaluation der Individuen in einer Population geschieht durch den bereitgestellten Simulator der FightingICE. Ein Individuum, also die Folge von drei Aktionen, wird mit einer Paarung von drei gegnerischen Aktionen³ über die Dauer von 60 Frames simuliert, was eine neue *FrameData* generiert. Die Auswertung gewisser Attribute (HP, Distanz, EP etc.) und der Vergleich der resultierenden *FrameData* mit der derzeitigen stellt eine Vergleichbarkeit der Individuen her und bietet die Grundlage, auf der Fitnessfunktionen definiert werden können. Die Auswahl, also der Entscheidungsprozess, welches Individuum aus der Lösungsmenge ausgewählt wird, wird durch die Gewichtung einer bestimmten Fitnessfunktion realisiert. Die verwendeten Fitnessfunktionen werden in Abschnitt 3.6 eingeführt. Eine nutzerbasierte Auswahl unter den

³ Die Auswahl dieser Aktionen wird später erläutert.

finalen Individuen der Pareto Front ist in der FightingICE, bedingt durch das geringe Zeitfenster von 16,67 Millisekunden nicht möglich. Deshalb wird das Individuum aus der Zielmenge ausgewählt, welches die größte Fitness in Bezug auf das Ziel: Maximierung des verursachten Schadens unter Berücksichtigung des erlittenen Schadens, aufweist. Anschließend werden die drei Aktionen des Individuums zur Ausführung bereit in einer Warteschlange gespeichert.

Einen großen Einfluss auf die neu generierte *FrameData* und damit auch auf die Approximation der Pareto-Front durch die entstandene Lösungsmenge hat dabei die Auswahl der gegnerischen Aktionen, welche zur Simulation verwendet werden. Die *SMSEMOAai* entscheidet anhand einer Vorhersage der nächsten drei gegnerischen Aktionen. Diese Vorhersage basiert auf einer *Prediction* Klasse, welche aus dem Controller des Referenzgegners *NewHighlightMcts* übernommen wurde. Diese Klasse filtert genau wie die *SMSEMOAai* zuerst die unmöglichen Aktionen aus der Menge der 42 Aktionen und ermittelt anschließend basierend auf der Distanz zum Gegner und den bereits getätigten Aktionen des Gegners im Kampf, welche Aktionen am wahrscheinlichsten sind [51].

Die Grundidee bei der Entwicklung der *SMSEMOAai* und die Begründung der Repräsentation eines Individuums in Form einer Aneinanderreihung von Aktionen basiert auf dem Verhalten des MCTS Algorithmus, der in Abschnitt 3.2 dargestellt wurde. Während MCTS eine Baumstruktur zur Auswahl der nächsten Aktion erstellt und solange simuliert, bis die 16,67 Millisekunden erreicht sind, wählt *SMSEMOAai* eine mögliche Aktionsfolge in einem dreidimensionalen Suchraum der Größe 42^3 aus. Dieser Ansatz ist inspiriert durch das Schema der MCTS, greift aber auf die Fähigkeit von EAs zurück, große Suchräume vergleichsweise effizient zu durchsuchen [5]. Da nur in der Tiefe von drei Aktionen nach einer Lösung gesucht wird, bevor der SMS-EMOA im nächsten Frame wieder mit einer neueren *FrameData* durchlaufen wird, erwartet man, dass die Umsetzung in der *SMSEMOAai* ressourcensparender ist als die MCTS.

Eine wichtige Eigenschaft der *SMSEMOAai* ist, dass der Entscheidungsprozess ausschließlich auf den Ergebnissen der Optimierung des SMS-EMOA basiert. Es wurden keine hartcodierten Verhaltensmuster vorgegeben, die diese Entscheidungsfindung beeinflussen. Viele der für die FTGAIC entwickelten Controller, unter anderem auch die, die den MCTS zur Optimierung nutzen, basieren neben einem Algorithmus zur Optimierung auch auf hartcodiertem Verhalten [1]. Mit der *SMSEMOAai* soll ein Controller programmiert werden, welcher ohne

Vorkenntnisse der Charaktere oder den durch Studien über die FightingICE erlangten Erkenntnissen, wie z. B. die optimale Distanz zum Gegner, Entscheidungen trifft. Die *SMSEMOAai* strebt somit direkt die Erfüllung des im Abschnitt 3.1 definierten Gesamtziels der FightingICE an, eine vereinheitlichte KI für Fighting Games zu entwickeln. Zudem hat der Verzicht auf hartcodiertes Verhalten den Vorteil, dass die Performance des SMS-EMOA in Verbindung mit dem erläuterten Konzept in dieser Problem Instanz unverfälscht dargestellt wird.

3.6 Fitnessfunktionen

Um die Individuen der *SMSEMOAai* anhand ihrer Fitness durch den SMS-EMOA im Zielraum zu sortieren und zu selektieren, wurden Fitnessfunktionen definiert. Wie im Abschnitt 3.5 erläutert, basieren die meisten Funktionen auf der Auswertung des Deltas der *FrameData* vor und nach der Simulation. Während des Entwicklungsprozesses des Controllers wurden insgesamt zehn verschiedene Fitnessfunktionen definiert⁴ und in vorläufigen Experimenten miteinander verglichen. Da eine Evaluation aller zehn Ziele technisch gegenwärtig nicht umsetzbar ist, um bei einer Populationsgröße von zehn Individuen und unter der Terminierungsbedingung von 150 Evaluationen alle 16,67 Millisekunden eine Pareto-Menge zu generieren, wurden vier Fitnessfunktionen ausgewählt. Dies hat neben der Minimierung der Rechenzeit noch weitere Gründe. Diese vier Funktionen waren zwingend nötig. Neben dem wesentlichen Ziel eines Fighting Games, nämlich zu gewinnen, wurden weitere Hilfsfunktionen etabliert, welche menschliches Verhalten approximieren sollen. Für diesen Ansatz eignet sich der SMS-EMOA, da andere EMOA bei einer Zielfunktionsdimension größer als drei versagen [9]. Zudem ließen sich oftmals mehrere Ziele einer gemeinsamen Fitnessfunktion zusammenfassen, ohne einen Interpretationsverlust, siehe Abschnitt 2.2, in Kauf nehmen zu müssen, wie beispielsweise die Maximierung des verursachten Schadens und die Minimierung des erlittenen Schadens.

Die Auswahl dieser vier Zielfunktionen basiert auf der Analyse von Kämpfen gegen den Referenzgegner *NewHighlightMcts* [51]. Diese Analyse wurde im Hinblick auf die Beobachtung

⁴ Auflistung im Anhang A

des Verhaltens der *SMSEMOAai* und dem Ergebnis der Kämpfe durchgeführt. Wichtig war, dass der Controller ein ausgeglichenes Verhalten zeigt. Er sollte eine Balance zwischen aggressivem und passivem Verhalten präsentieren.

Eine Auflistung der vier ausgewählten Ziele und eine kurze Erläuterung, der erwarteten Auswirkungen einzelner Funktionen auf den Controller, vermittelt einen Überblick über das Verhalten. Die finale Konfiguration der *SMSEMOAai*, bezogen auf das Verhalten, wird in Abschnitt 3.8.3 genauer betrachtet.

1. Maximierung des verursachten Schadens unter Berücksichtigung des erlittenen Schadens.

Dies ist die wichtigste Zielfunktion, da sie das Hauptziel eines FTG darstellt. Sie ist essenziell für das Verhalten der *SMSEMOAai*, denn sie treibt den Controller an, in jedem Simulationsdurchlauf mehr Schaden als der Gegner zu verursachen. Zugleich berücksichtigt sie die Balance zwischen Aggressivität und Passivität. Wie schon in Abschnitt 3.5 erläutert, wird die Entscheidungsfindung, welches Individuum aus der Zielmenge gewählt wird, aufgrund dieser Fitnessfunktion getätigt.

2. Minimierung der Distanz zum Gegner.

Damit der Controller nach erfolgreichen Angriffen nicht zu passiv agiert, ist es zweckmäßig, eine geringe Distanz zum Gegner herzustellen.

3. Maximierung des Zeitintervalls zwischen einem erfolgreichen Treffer einer Aktion und einem erfolgreichen Treffer des Gegners.

Dieses Ziel soll den Controller motivieren nach einem erfolgreichen Angriff eine Distanz zum Gegner aufzubauen. Im Hinblick auf das Ziel, die Distanz zum Gegner zu minimieren, entsteht ein Konflikt. Die Erwartung hinter der Erzeugung eines Konflikts bezogen auf die Distanz ist, dass der Controller den Konflikt anhand einer Kombination schneller Attacken löst. Dies könnte den Gegner in einem Zustand versetzen, indem er keine Möglichkeit hat, einen Gegenangriff zu initialisieren und daraufhin wiederholend zu Boden geht (*isDown*).

4. Maximierung, der Anzahl an Aktion, welche dem Gegner Schaden zufügen.

Um bei einem Rückzug oder einer Verringerung der Distanz zum Gegner keine Schwachstellen zu schaffen, soll der Controller möglichst immer eine Attacke ausführen. Diese Funktion schließt Aktionen wie einen Sprung oder das Ducken nicht aus, sondern versucht sie mit Aktionen zu paaren, welche Schaden verursachen. Dies könnte sich z. B. bei einem Rückzug in einem Sprungtritt oder einem geduckten Schlag ausdrücken.

3.7 Kontrollparameter

Da der SMS-EMOA über keine eigenen Kontrollparameter verfügt [9], mussten lediglich fünf Parameter bei der Programmierung des Controllers berücksichtigt werden. Als Operatoren für die Variation wurde für die Mutation, die von jMetal bereitgestellte *IntegerPolynomialMutation* Klasse benutzt. Dieser Operator führt eine Mutation bezogen auf die Grenzen der Variablen und in Bezug auf den ursprünglichen Wert des Genoms aus. Genauer wird diese von Deb [55] beschrieben. Als Operator für die Rekombination wurde der, ebenfalls von jMetal bereitgestellte, *IntegerSBXCrossover* gewählt [52]. Die Abkürzung SBX steht für *Simulated Binary Crossover*. Dieser Operator simuliert eine binäre *single-point* Kreuzung. Die *single-point* Kreuzung wird oft bei EAs gewählt, die Individuen durch einen Bitstring repräsentieren. In diesem Verfahren wird meistens ein Punkt definiert, an dem die Bitstrings der Eltern geteilt werden und jeweils eine Hälfte des Bitstrings beider Individuen zur Erzeugung eines neuen Individuums benutzt werden [56].

Anschließend werden die fünf Parameter dargestellt und deren Funktionsweise beschrieben.

- *Mutationswahrscheinlichkeit*: Dieser Parameter beschreibt prozentual das Auftreten einer Mutation während eines Generationszyklus. Ein Wert von z. B. 0,5 bedeutet eine Wahrscheinlichkeit von 50%.
- *Kreuzungswahrscheinlichkeit*: Diese gibt an, wie wahrscheinlich eine Rekombination, also das Kreuzen zweier Individuen, in einer Generation ist.
- *Mutations-Distributions-Index*: Dieser Wert berücksichtigt, wie stark sich die Mutation auf die ausgewählte Variable eines Individuums auswirkt. Niedrige Werte (eins bis 15) werden als starke Mutation bezeichnet. Dies bedeutet, dass der neue, durch Mutation generierte

Wert der Variable weit vom ursprünglichen Wert entfernt ist. Dementsprechend bedeutet ein größerer Wert (> 15) eine höhere Wahrscheinlichkeit, Werte zu generieren, die dem des ursprünglichen Wertes sehr nahe sind [57]. Bezogen auf die Individuen der *SMSEMOAai* bedeutet dies, dass eine Aktion nach einer Mutation mit einem starken Mutations-Distributionsindex stark verändert wird. Die 42 Aktionen der Charaktere sind kategorisch nach Aktionstypen (Bodenattacken, Luftattacken etc.) geordnet. So wird z. B. aus einem Sprung ein geduckter Schlag. Deshalb hat eine starke Mutation eine größere Chance, ein mögliches lokales Minimum zu verwerfen und die Diversität der Population zu steigern.

- *Kreuzungs-Distributions-Index*: Ähnlich wie bei dem Mutations-Distribution-Index gibt dieser Parameter an, wie wahrscheinlich es ist, dass sich das neue Individuum in der Nähe der Eltern im Problemraum einordnet. Ein höherer Wert resultiert in einer höheren Wahrscheinlichkeit, dass sich das neu erzeugte Individuum in der Nähe der Eltern befindet. Ein niedriger Wert bedeutet, dass Nachkommen entfernte Punkte im Suchraum als Lösungen repräsentieren können [56].
- *Populationsgröße*: Anhand der Populationsgröße wird definiert, wie viele Individuen gleichzeitig in einer Population existieren dürfen.

Für die in Abschnitt 3.8 durchgeführten Experimente wurde die Konfiguration der Parameter, wie in Tabelle 1 dargestellt, genutzt.

Tabelle 1: Kontrollparameter der SMSEMOAai nach manuellem durchgeführten Parametertuning.

Kontrollparameter	Werte
Mutationswahrscheinlichkeit	0,2
Kreuzungswahrscheinlichkeit	0,01
Mutations-Distributions-Index	1
Kreuzungs-Distributions-Index	1
Populationsgröße	10

Tabelle 2: Von jMetal definierte Standardwerte der Kontrollparameter.

Kontrollparameter	Werte
Mutationswahrscheinlichkeit	0,33
Kreuzungswahrscheinlichkeit	0,9
Mutations-Distributions-Index	20
Kreuzungs-Distributions-Index	20
Populationsgröße	10

Durch manuelles Tuning der Parameter (Tabelle 1) konnte die Performance bei den durchgeführten Experimenten gegenüber der Standardkonfiguration von jMetal (Tabelle 2) bereits verbessert werden, siehe Abschnitt 3.8. Für das Tuning wurden Werte für die Parameter randomisiert ausgewählt und die Ergebnisse aus insgesamt zehn Kämpfen gegen den Referenzgegner der *NewHighlightMcts* miteinander verglichen. Die Fachliteratur ist sich einig, dass das Tuning von Parametern sich positiv auf die Performance von EAs auswirkt [58]. Deshalb ist es sinnvoll für die zukünftige Optimierung der *SMSEMOAai*, die Auswirkung der Kontrollparameter genauer zu untersuchen. Gegebenenfalls sollten zudem Methoden zur Parameterkontrolle, also der dynamischen Parameteranpassung während des evolutionären Prozesses, untersucht werden [59]. Die *SMSEMOAai* könnte von diesen Methoden profitieren, da die Problem Instanz der FTGAIC sich ebenfalls dynamisch verhält, da in jedem Frame ein neuer Spielstad erschaffen wird.

3.8 Experimentelle Analyse

Dieser Abschnitt befasst sich mit der Beantwortung der in Abschnitt 3.2 formulierten Forschungsfragen. Zudem wird dargestellt, wie die Experimente durchgeführt wurden. Anschließend werden die Ergebnisse ausgewertet und interpretiert, um ein Fazit über die *SMSEMOAai* verfassen zu können.

Die FTGAIC stellt eine visuelle Darstellung der FightingICE zum Testen des entwickelten Controllers zur Verfügung, wie sie bereits in Abbildung 11 dargestellt wurde. Um die erste Forschungsfrage zu beantworten, die sich auf die Performance der *SMSEMOAai* gegen sechs Referenzgegner bezieht, ist eine visuelle Repräsentation des Kampfes nicht notwendig. Zudem

müsste händisch ein Kampf nach drei Runden neu gestartet und das Ergebnis manuell notiert werden. Deshalb wurde speziell in Hinblick auf die Automatisierung der Kämpfe, die Klasse *GameInitializer* programmiert, welche eine Simulationsumgebung der *FightingICE* implementiert. Mithilfe dieser Klasse lassen sich Kämpfe automatisiert starten und auswerten.

3.8.1 Durchführung der Experimente

Das erste Experiment dient der Beantwortung der Forschungsfrage: „Kann der Controller sich gegen etablierte Referenzgegner durchsetzen?“. Hierfür kämpfte die *SMSEMOAai* gegen jeden der sechs Referenzgegner, siehe Abschnitt 3.8.2, 100 Mal. Alle Kämpfe wurden von beiden Controllern mit dem Charakter *Zen* durchgeführt, um ein vergleichbares Ergebnis zu schaffen. Die Kämpfe wurden mithilfe des *GameInitializer* durchgeführt. Zudem wurden bei allen Gegnern, die den MCTS Algorithmus nutzen, die Iterationen des MCTS auf einen zufälligen Wert zwischen 150 und 160 limitiert, um das Ergebnis unabhängig in Bezug auf den benutzten Computer zu modellieren und eine Vergleichbarkeit zwischen dem SMS-EMOA, der 150 Evaluationen nutzt, herzustellen. Die Ergebnisse der Kämpfe wurden in einem Datenträger gespeichert. Gegenüber einem Gegner durchgesetzt hat sich die *SMSEMOAai*, wenn eine Erfolgsrate von mindestens 50% erzielt wurde.

Die zweite Forschungsfrage befasst sich mit der Frage „Lässt sich ein Verhaltensmuster der *SMSEMOAai* bei Kämpfen gegen die *NewHighlightMcts* erkennen?“. Um diese Frage zu beantworten, wurde die Zusammenstellung von Fitnessfunktionen betrachtet, welche in Abschnitt 3.6 bereits definiert wurde. Diese Kombination sollte sich per Definition sehr ausgeglichen verhalten. Die Beobachtungen wurden durch die visuelle Verfolgung von Kämpfen gegen den Referenzgegner *NewHighlightMcts* durchgeführt. Die Ergebnisse der Analyse des Verhaltens basieren deshalb auf der subjektiven Wahrnehmung des Autors.

3.8.2 Ergebnisse gegen Referenzgegner

Bevor die Ergebnisse der Simulationen dargestellt und interpretiert werden, folgt zunächst eine Darstellung und Beschreibung der sechs berücksichtigten Referenzgegner. Zudem werden sie basierend auf den Ergebnissen der FTGAIC in den letzten Jahren nach ihrer Stärke geordnet, um abschließend festzustellen, wo sich die *SMSEMOAai* einordnet.

- *Thunder*: Dieser Controller basiert auf den MCTS und limitiert den Suchraum durch das Entfernen von unnötigen Bewegungen und Attacks [60]. Zudem wurde ein Verhalten für den Charaktere Zen programmiert, welches Gegner in die Ecke drängt und kontinuierlich mit einem geduckten Tritt zu Boden bringt, was für die meisten Controller schwer zu verteidigen ist. Er nutzt also die Schwachstellen der FightingICE bezogen auf die *MotionData* der Charaktere aus. Wenn er einen vordefinierten Betrag an HP-Vorsprung generiert hat, nimmt er Abstand vom Gegner und bewegt sich in eine Ecke der Arena. Von hieraus verteidigt er sich, bis er ausreichend viel EP hat (150), um einen Feuerball auszuführen, welcher den meisten Schaden (120) von allen 42 Aktionen verursacht. Verschiedenste Versionen von Thunder gewannen bereits vier Mal die FTGAIC, deshalb wird er als der stärkste Gegner gegen die *SMSEMOAai* betrachtet [60].
- *FooAI*: Es handelt sich hierbei um eine optimierte Version der *SampleMctsAI*⁵ [61]. Ähnlich wie Thunder passt FooAI ebenfalls das Verhalten an die Auswahl des Charakters an. Es wurden zudem viele Verhaltensmuster hart codiert, die laut den Ergebnissen der bisherigen FTGAICs ein vorteilhaftes Verhalten zeigen. Hierzu gehört z. B. die Prozedur, bei einer gegnerischen Energie von 300 zu springen, um einem möglichen Feuerball auszuweichen. In Bezug auf das Experiment wird er basierend auf den Ergebnissen in der FTGAIC als zweitbesten Gegner vermutet [61].
- *JayBot_GM*: Dieser Referenzgegner benutzt eine Kombination aus MCTS und evolutionären Algorithmen für den Charakter Zen. Dieser Controller basiert auf einem ähnlichen Ansatz wie die *SMSEMOAai*. Individuen werden anhand einer Kombination von Aktionen repräsentiert. Es wird jedoch anhand einer einzigen Fitnessfunktion evaluiert. Die Fitness bezieht sich nur auf den HP-Unterschied der Spieler nach der Simulation,

⁵ Wird später definiert.

welche mithilfe des MCTS durchgeführt wurde. Basierend auf den Ergebnissen der FTGAIC wird er als drittstärkster Gegner vermutet [5].

- *Toothless*: Er bezieht sein Verhalten von der Auswahl des Charakters. Bei der Auswahl von Zen generiert *Toothless* seine Entscheidungen aus einer Kombination von Attacken, die ähnlich wie bei Thunder in einer Schleife enden können, um den Gegner in die Ecke zu drängen. Zudem wird ein Minimax Algorithmus zur Entscheidungsfindung verwendet. Minimax wird i. d. R. im Bereich der Optimierung verwendet, um die Minimierung eines *worst-case* Szenariums, also den schlimmsten Fall anzustreben und wird seit Jahrzehnten für KIs für Videospiele verwendet [62].
- *SampleMctsAi & NewHighlightMcts*: Diese beiden Controller wurden vom Entwicklerteam der FightingICE bereitgestellt, um bei der Entwicklung eines Controllers als Referenzgegner zu fungieren. Der *SampleMctsAi* [60] basiert ausschließlich auf Simulation durch den MCTS, ohne hartcodierte Funktionen zu nutzen, die bei der Entscheidungsfindung den Controller beeinflussen. Der *NewHighlightMcts* stellt zusätzliche Prozeduren wie die Vorhersage gegnerischer Angriffe oder das Ausweichen von Feuerbällen zur Verfügung, die durch eine Konfigurationsklasse aktiviert oder deaktiviert werden können. Für die Durchführung des Experiments wurden sämtliche Zusatzfunktionen, ausschließlich der Funktion *Highlight*, deaktiviert.

Es ist zu beachten, dass die gewählten Referenzgegner zu den stärksten Controllern der FTGAIC überhaupt gehören.

Bei einer Konfiguration der Kontrollparameter durch die in Tabelle 2 dargestellten Werte, resultierte das Experiment folgende Ergebnisse (siehe Tabelle 3).

Tabelle 3: Ergebnisse des Experiments, mit Konfiguration aus Tabelle 2.

Gegner	Simulationen	Gewonnen	Verloren	Unentschieden	Erfolgsrate in %
NewHighlightMcts	100	15	85	0	15
Toothless	100	3	97	0	3
JayBot_GM	100	0	100	0	0
SampleMctsAi	100	0	100	0	0
FooAI	100	0	100	0	0
Thunder	100	0	100	0	0

Unter der Konfiguration der Kontrollparameter in Tabelle 1, wurden folgende Ergebnisse ausgewertet (siehe Tabelle 4).

Tabelle 4: Ergebnisse des Experiments, mit Konfiguration aus Tabelle 1.

Gegner	Simulationen	Gewonnen	Verloren	Unentschieden	Erfolgsrate in %
NewHighlightMcts	100	20	80	0	20
Toothless	100	8	92	0	8
JayBot_GM	100	0	100	0	0
SampleMctsAi	100	1	99	0	1
FooAI	100	1	99	0	1
Thunder	100	0	100	0	0

Die Ergebnisse zeigen, dass sich die *SMSEMOAai* gegen keinen der Referenzgegner durchsetzen konnte. Eine Teilnahme in der nächsten FTGAIC für 2021 mit der derzeitigen Version der *SMSEMOAai* ist daher nicht anzustreben. Dennoch sind die Ergebnisse des Experiments nicht als Misserfolg zu werten, denn es wurden Siege erzielt und dem Controller gelang es, jedem Gegner Schaden zuzufügen, was teils zu einem sehr knappen Ausgang der Kämpfe führte. Eine tabellarische Übersicht über den durchschnittlich verursachten und erlittenen Schaden der *SMSEMOAai* wird aus Platz- und Zeitgründen nicht bereitgestellt. Dieser Aspekt wird jedoch in zukünftigen Untersuchungen betrachtet.

3.8.3 Analyse des Verhaltens

Bei der Betrachtung der Kämpfe gegen *NewHighlightMcts* konnte insbesondere ein auffälliges Merkmal des Verhaltens beobachtet werden. Die *SMSEMOAai* ergreift die Initiative zumeist mithilfe eines Sprungtritts. Vermutlich resultiert dieses Verhalten aus der Kombination der Zielfunktionen, die Distanz zum Gegner zu verringern und möglichst immer Aktionen auszuwählen, welche dem Gegner Schaden zufügen. Zudem hat dieses Verhalten eine hohe

Chance den Gegner zu treffen, da Attacken aus der Luft schwierig zu blocken sind. Die *SMSEMOAai* nutzt diesen Vorteil oftmals aus.

Auffällig ist zudem der Mangel an Energiemanagement. Es wird viel Energie verschwendet dadurch, dass die *SMSEMOAai* motiviert wird ständig Attacken auszuführen, auch wenn diese nicht unbedingt treffen, sodass die *SMSEMOAai* kaum die Gelegenheit hat, einen Feuerball auszuführen. Dadurch, dass Energie nicht nur durch Treffer, sondern auch durch erlittenen Schaden erzeugt wird, scheint aber eine Fitnessfunktion zur Maximierung der EP nicht zielführend. Deshalb ist es für zukünftige Untersuchungen wichtig, unnötige Ausführungen von Attacken zu verhindern und dem Controller eine Richtlinie zu definieren, wann welche Attacken sinnvoll sind.

Interessant ist die Tatsache, dass die *SMSEMOAai* scheinbar das Ausweichen von Projektilen versucht. Die meisten Controller haben dieses Verhalten hart codiert. Vermutlich erkennt die *SMSEMOAai* die Gefahr eines Projektils durch die vorhersehbare Flugbahn frühzeitig genug, um auszuweichen.

Eine weitere Beobachtung ist der oftmals durchgeführte *BACK_STEP* nachdem ein erfolgreicher Angriff durchgeführt wurde. Ein *BACK_STEP* bezeichnet das schnelle Zurückweichen eines Spielers. Dies dient dem Ausweichen eines möglichen Konters und ist höchstwahrscheinlich durch die Fitnessfunktion der Maximierung der Zeit zwischen Treffern des Gegners und Minimierung der Zeit zwischen getroffenen Attacken bedingt.

Insgesamt lässt sich das Verhalten als ausgeglichen bezeichnen, da die *SMSEMOAai* neben der Initialisierung eines Angriffes auch eine gewisse Distanz zum Gegner sucht, wenn er in eine Ecke gedrängt wird. Um ein objektives Urteil über das Verhalten zu erlangen, ist es dennoch wichtig, in zukünftigen Forschungen eine statistische Analyse durchzuführen, um festzustellen was die Stärken und Schwächen der *SMSEMOAai* sind.

3.9 Erkenntnisse der Fallstudie

Es scheint nicht zielführend 42 Aktionen zu betrachten, um die beste auszuwählen. Dies wird durch den Erfolg des MCTS dargestellt, der mehrere Aktionen durch eine Tiefensuche

analysiert. Unter Berücksichtigung dieses Aspektes, ist der Ansatz, den die *SMSEMOAai* repräsentiert nachvollziehbar.

Alle Controller, bis auf *SampleMctsAi* und *NewHighlightMcts*, streben das Ausnutzen von Schwächen in der FightingICE, bezogen auf die *MotionData* der Charaktere an. Dies wird über hartcodiertes Verhalten realisiert. Für die weitere Optimierung der *SMSEMOAai* scheint es vorteilhaft zu sein, ebenfalls ein Verhaltensmuster bezogen auf den Charakter zu implementieren. Da die FTGAIC jedoch die Entwicklung einer universellen KI für Fighting Games anstrebt, gilt ein solcher Ansatz nicht als Antwort auf die vom ICE Lab. definierten Forschungsfragen (Abschnitt 3.1). Da trotzdem Controller, die durch Charakterspezifisches hartcodiertes Verhalten agieren am erfolgreichsten sind, sollten bereits bekannte und von Controllern ausgenutzte Schwachstellen bezogen auf die Charaktere in der FightingICE ausgebessert werden.

Die in Abschnitt 3.5 beschriebene Vorhersage der gegnerischen Aktionen zeigt sich im direkten Vergleich gegen den MCTS, wie er in der *SampleMctsAi* oder *NewHighlightMcts* genutzt wurde, als nicht vielversprechend. Der Erfolg, der auf MCTS basierenden Controller zeigt, dass eine Betrachtung einer Abfolge von Aktionen der richtige Ansatz für einen EA ist, jedoch die Betrachtung der gegnerischen Aktionen ebenso wichtig ist wie die der eigenen. Dies wird durch die Tiefensuche einer Baumstruktur in Bezug auf die möglichen Aktionen des Gegners durch den MCTS kompensiert. Die *SMSEMOAai* hingegen stützt sich lediglich auf die Vorhersage von drei möglichen Aktionen des Gegners. Wenn jedoch schon eine der vorhergesagten Aktionen nicht eintritt, verfälscht dies das gesamte Ergebnis der Optimierung, was möglicherweise in einer für den derzeitigen Spielstand nicht optimalen Lösung resultiert. Eine mögliche Lösung dieses Problems wäre eine Optimierung der möglichen Aktionen des Gegners durch einen EA durchzuführen.

Die Tatsache, dass *Toothless* und *Thunder* anstreben, den Gegner in einer Schleife festzuhalten und die *SMSEMOAai* dennoch Schaden verursacht, lässt vermuten, dass EMOA in Bezug auf die FightingICE einen schnellen Weg aus Situationen finden, welche sich als äußerst ungünstig erweisen. Hierzu gehören z. B. das in die Ecke drängen des Gegners oder das durch spezielle Aktionen ständige zu Boden bringen, was über die Verhaltensanalyse festgestellt werden konnte. Diese Eigenschaft gilt es in Zukunft weiterhin zunutze zu machen, um präventiv gegen das Forcieren einer Schleife handeln zu können.

Trotz der geringen Erfolgsrate im ersten Experiment, scheint die Nutzung eines EMOAs zur Entscheidungsfindung als vielversprechend. Dies zeigt insbesondere die Verhaltensanalyse, da beobachtet werden konnte, dass bei vorhersehbaren Ereignissen wie z. B. der Flugbahn eines Projektils, der Erzeugung einer Schleife durch Attacken oder der Flugkurve des Gegners oftmals schnell und richtig reagiert wird.

3.10 Weiterentwicklung der *SMSEMOAai*

Anhand der Ergebnisse aus der Fallstudie geht hervor, dass die *SMSEMOAai* ein vielversprechendes Konzept für einen Controller der FTGAIC darstellt. Dennoch ist eine Weiterentwicklung nötig, um das Durchsetzen gegen die in der Fallstudie betrachteten Referenzgegner zu gewährleisten. Dieser Abschnitt behandelt Ideen, welche zu einer Verbesserung der *SMSEMOAai* führen könnten.

Bei der Entwicklung der *SMSEMOAai* und der damit verbundenen Einbindung des SMS-EMOA wurde festgestellt, dass der Algorithmus zwar in hohen Dimensionen optimieren kann, jedoch die Berechnung des Hypervolumens einen hohen Rechenaufwand benötigt [9]. Bezogen auf die FTGAIC entstanden dadurch Limitierungen in Bezug auf die Simulationszeit, der Dimension des Zielfunktionsvektors und die Begrenzung von höchstens drei Aktionen als Repräsentation eines Individuums. Zudem musste eine Populationsgröße von zehn und ein Evaluationslimit von 150 definiert werden, um die Rechenzeit von 16.67 Millisekunden einzuhalten, es entsteht also ein Engpass. Das Hypervolumen beinhaltet aber für die Problem Instanz der FTGAIC viele Vorteile, wie in Abschnitt 3.3 dargestellt wurde. Deshalb ist es wichtig, die Rechenzeit zur Berechnung des Hypervolumens zu verringern, um der *SMSEMOAai* mehr Ressourcen für die Optimierung bereitzustellen. Für die Weiterentwicklung der *SMSEMOAai* sollte deshalb die Approximation des Hypervolumens und das dadurch resultierende Selektionskriterium des niedrigsten Beitrages durch den R2 Indikator erforscht werden. Der R2 Indikator basiert auf einer vielfältigen Gewichtung von Fitnessvektoren und ermöglicht ebenso die Approximation des Hypervolumens [63]. Eine vorgestellte Implementation des R2 durch Shang [18] oder Künzel und Meyer-Nieberg [64] verspricht eine gute Approximation des Hypervolumens und könnte genannte Engpässe der *SMSEMOAai* relativieren. Das Framework jMetal bietet zwar die Implementation eines R2 Indikators an, stellt jedoch keinen R2-EMOA

zur Verfügung, welcher zur Weiterentwicklung der *SMSEMOAai* genutzt werden könnte [53]. Deshalb würde dieser Ansatz eine eigene Implementation verlangen, welche durch zusätzliche Optimierungen und Performancesteigerungen unterstützt werden könnte.

Neben der möglichst effizienten Nutzung der Ressourcen wurden weitere Erkenntnisse durch die Experimente mit der *SMSEMOAai* gewonnen. Als äußerst notwendig stellte sich die Berücksichtigung der möglichen gegnerischen Aktionen bei der Optimierung und der Entscheidungsfindung der bestmöglichen Lösung heraus. Dieses Problem wurde von der *SMSEMOAai* nur teilweise durch eine grobe Vorhersage der gegnerischen Aktion gelöst. Die resultierenden Ergebnisse der Experimente zeigten, dass dieses Implementieren eine große Schwachstelle des Controllers darstellt. Um dieses Problem zu lösen scheint eine Implementierung eines CoEAs, welcher in Abschnitt 2.1.4 bereits erläutert wurde, als vielversprechend. Es ließen sich mehrere Populationen initialisieren, die verschiedenste Ziele anstreben z. B. aggressives oder passives Verhalten und dann Spielstand spezifisch eine entschieden wird welches Verhalten vorteilhafter ist, um dynamisch das Verhalten des Controllers zu verändern. Zudem ließe sich mit mehreren Populationen, eine vielversprechende Vorhersage der Aktionskette des Gegners generieren. So könnte sich eine Population auf die Optimierung der Auswahl der gegnerischen Aktionen fokussieren und eine weitere die Optimierung der eigenen Aktionen anstreben. Das Forcieren eines Wettkampfs zwischen diesen Populationen, nach Liu [36], hätte Potenzial die Vorhersagegenauigkeit der gegnerischen Aktionen zu steigern. Für diesen Ansatz, müsste jedoch die Repräsentation eines Individuums verändert werden. Da eine Population auch eine Optimierung der gegnerischen Aktionen anstreben soll, könnte sich der Suchraum, bei einer Betrachtung von jeweils drei Aktionen von 42^3 auf 42^6 ausbreiten.

Eine weitere Idee, um eine genaue Vorhersage des Gegners zu erstellen, wäre anhand eines EAs eine Population von künstlichen neuronalen Netzen (KNN) zur Laufzeit zu trainieren. Neuronale Netze sind dabei dem Aufbau und der Funktionsweise des biologischen Gehirns nachempfunden und eignen sich, um verschiedenste Problemstellungen zu lösen [65]. Diese Netze streben das Vorhersagen der nächsten Aktion des Gegners auf Basis des aktuellen Spielstands an. Die Fitness eines KNN wird durch die Genauigkeit der Vorhersage bestimmt. Um die Genauigkeit bestimmen zu können muss auf eine Datenbank zurückgegriffen werden, die Samples der entstanden *FrameData* bei der Auswahl von Aktionen bezogen auf den

aktuellen Spielstand besitzt. Zudem werden während des Kampfes ebenfalls Samples vom Gegner erstellt, sodass ein Modell des Verhaltens vom Gegner erstellt wird. Die Vorhersage der KNN in Bezug auf gegnerischen Aktionen, kann danach wie in Abschnitt 3.5 eingebunden und verarbeitet werden.

4 Zusammenfassung und Ausblick

Evolutionäre Algorithmen sind leistungsfähige Werkzeuge, um Optimierungsprobleme zu lösen oder zu approximieren. Zudem haben sie viele Anwendungsfelder und können mithilfe von Frameworks wie jMetal sehr praxisnah und unkompliziert implementiert werden. Es ist zu erwarten, dass das Themengebiet der EMOA weiterhin viel Aufmerksamkeit in der Forschung bekommt, gerade weil die Anwendungsbereiche so vielseitig sind und viele Lösungen für reale Probleme bereits durch EMOA approximiert werden konnten. So wurden im Zeitraum zwischen 1990 und 2018 insgesamt 76.358 Publikationen veröffentlicht, welche sich mit der evolutionären mehrkriteriellen Optimierung befassen. Allein etwa 7.500 von den bisherigen Publikationen wurden im Jahr 2018 veröffentlicht [10].

Die in dieser Arbeit durchgeführte Fallstudie behandelt die Entwicklung des Controllers *SMSEMOAai* für die Fighting Game AI Competition, die einen Kampf, basierend auf den Rahmenbedingungen eines 2D Fighting Games [1] zwischen zwei künstlichen Intelligenzen simuliert [2]. Dieser Controller bezieht die Entscheidungsfindung, also welche Aktion als nächstes ausgeführt wird, ausschließlich durch die Optimierung von Aktionsketten durch den SMS-EMOA [9]. Der Algorithmus selektiert anhand der Dominanzzahl und den Einzelbeiträgen der Individuen zum Hypervolumen [48]. Somit wird eine Approximation der optimalen Lösungsmenge erreicht. Die *SMSEMOAai* generiert zuerst anhand einer Vorhersage eine Aktionskette für den Gegner. Die Vorhersage basiert auf den bisherigen ausgeführten und den derzeitigen möglichen Aktionen des Gegners. Anschließend simuliert die *SMSEMOAai* auf Grundlage der Vorhersage und eigener Aktionsketten Spielszenarien durch, welche dann vom SMS-EMOA bewertet werden, um die beste Aktionskette bezogen auf die gegnerische Vorhersage zu finden.

Mit der *SMSEMOAai* wurde ein vielversprechender Controller entwickelt, der bereits gegen etablierte Referenzgegner erste Erfolge erzielen konnte. Außerdem konnte anhand einer Verhaltensanalyse und unter Verwendung von vier Fitnessfunktionen ein ausgeglichenes Verhalten, was Offensive und Defensive betrifft, implementiert werden. Dennoch ist die *SMSEMOAai* als ein Prototyp zu betrachten, welcher noch nicht wettkampffähig ist. Das Potenzial von EMOAs in der Fighting Game AI Competition gegenüber dem MCTS [4], wie es schon der Sieger des Wettkampfes 2020, *ERHEA_PI* [19], beweisen konnte, ist bemerkenswert.

Es ist anzunehmen, dass die *SMSEMOAai* mithilfe einer Optimierung des SMS-EMOA, bezogen auf Qualitätsmaß und Kontrollparameter und einer Implementierung eines Gegnermodells zur Vorhersage der gegnerischen Aktionen weiter aufschließen kann. Bezogen auf die Kontrollparameter des SMS-EMOA müsste ein ausführliches Parametertuning durchgeführt werden, um die optimalen Werte für die Problem Instanz der Fighting Games zu finden. Es konnte festgestellt werden, dass die Implementierung der Berechnung des Hypervolumens durch jMetal äußerst rechenaufwändig ist. Dadurch dass die FightingICE eine schnelle Entscheidungsfindung (innerhalb von 16,67 Millisekunden) [2] verlangt, könnte eine Ersetzung durch den R2 Indikator, der das Hypervolumen effizient approximiert [64], die Performance der *SMSEMOAai* steigern. Das größte Defizit der *SMSEMOAai* ist jedoch die Vorhersage der gegnerischen Aktionskette, da diese zu ungenau geschieht und die Performance der *SMSEMOAai* sehr stark an die Vorhersage gekoppelt ist. Dieses Defizit könnte anhand einer Modellierung eines Gegnermodells durch ein neuronales Netz, während der Laufzeit behoben werden. Weiterhin scheinen *Coevolutionary algorithms* [37] ein interessanter und vielversprechender Ansatz für die Weiterentwicklung der *SMSEMOAai* zu sein. *Coevolutionary algorithms* initialisieren und optimieren mehrere Populationen gleichzeitig [37]. Dadurch wäre es möglich neben der Optimierung der eigenen Aktionskette durch eine Population auch eine genauere Vorhersage der gegnerischen Aktionen anhand einer weiteren Population durch einen Wettkampf [36] zu forcieren.

Im Kontext dieser Bachelorarbeit zeigt dieser Ausblick, dass auch eine zeitkritische Problem Instanz wie die Fighting Game AI Competition ein bedeutendes Anwendungs- und Testfeld für die Entwicklung künstlicher Intelligenz ist und EMOA dabei eine nicht zu vernachlässigende Rolle spielen. Eine tieferegehende Forschung im Bereich der

Echtzeitzeitproblemlösung durch evolutionäre Algorithmen könnte zeigen, dass EMOA ähnlich gut wie der MCTS [4] in die KI-Videospielindustrie etabliert werden können.

5 Literaturverzeichnis

- [1] Wikia, „A List and Guide to Game Genres“, 2012.
https://vsrecommendedgames.fandom.com/wiki/A_List_and_Guide_to_Game_Genres (zugegriffen März 02, 2021).
- [2] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee, und R. Thawonmas, „Fighting game artificial intelligence competition platform“, in *2013 IEEE 2nd Global Conference on Consumer Electronics, GCCE 2013*, 2013, S. 320–323, doi: 10.1109/GCCE.2013.6664844, ISBN:9781479908929.
- [3] R. Thawonmas, „Welcome to Fighting Game AI Competition“.
<http://www.ice.ci.ritsumei.ac.jp/~ftgaic/index.htm> (zugegriffen März 11, 2021).
- [4] J. Bradberry, „Introduction to Monte Carlo Tree Search“, *Jeff Bradberry*, Sep. 07, 2015.
<http://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/> (zugegriffen März 02, 2021).
- [5] M. J. Kim und C. W. Ahn, „Hybrid fighting game AI using a genetic algorithm and monte carlo tree search“, in *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion*, 2018, S. 129–130, doi: 10.1145/3205651.3205695, ISBN:9781450357647.
- [6] M. French, „Optimization: The Big Idea“, in *Fundamentals of Optimization*, Cham: Springer International Publishing, 2018, S. 1–13, doi: 10.1007/978-3-319-76192-3, ISBN:978-3-319-76192-3.
- [7] T. Weise, „Global Optimization Algorithms—Theory and Application“, URL [http://www. it-weise.de](http://www.it-weise.de), *Abrufdatum*, Bd. 1, 2009, doi: 10.1.1.64.8184.
- [8] A. E. Eiben und J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, doi: 10.1007/978-3-662-44874-8, ISBN:978-3-662-44873-1.
- [9] N. Beume, B. Naujoks, und G. Rudolph, „SMS-EMOA - Effektive evolutionäre Mehrzieloptimierung“, *At-Automatisierungstechnik*, Bd. 56, Nr. 7, S. 357–364, 2008, doi: 10.1524/auto.2008.0715.
- [10] K. Deb u. a., *Evolutionary Multi-Criterion Optimization: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings*, Bd. 11411. Springer, 2019, , ISBN:303012598X.
- [11] H. Yaghoobi, E. Babaei, B. M. Hussen, und A. Emami, „EBST: An Evolutionary Multi-Objective Optimization Based Tool for Discovering Potential Biomarkers in Ovarian Cancer“, *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, 2020, doi: 10.1109/tcbb.2020.2993150.
- [12] T. Shaik, V. Ravi, und K. Deb, „Evolutionary Multi-Objective Optimization Algorithm for Community Detection in Complex Social Networks“, *SN Comput. Sci.*, Bd. 2, Nr. 1, S. 1, 2020, doi: 10.1007/s42979-020-00382-x.
- [13] C. Pizzuti, „A multiobjective genetic algorithm to find communities in complex networks“, *IEEE Trans. Evol. Comput.*, Bd. 16, Nr. 3, S. 418–430, 2012, doi: 10.1109/TEVC.2011.2161090.

- [14] J. G. Falcón-Cardona, C. A. Coello Coello, und M. Emmerich, „CRI-EMOA: A pareto-front shape invariant evolutionary multi-objective algorithm“, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, Bd. 11411 LNCS, S. 307–318, doi: 10.1007/978-3-030-12598-1_25, ISBN:9783030125974.
- [15] K. Shang und H. Ishibuchi, „A New Hypervolume-Based Evolutionary Algorithm for Many-Objective Optimization“, *IEEE Trans. Evol. Comput.*, Bd. 24, Nr. 5, S. 839–852, 2020, doi: 10.1109/TEVC.2020.2964705.
- [16] K. Deb, L. Thiele, M. Laumanns, und E. Zitzler, „Scalable Test Problems for Evolutionary Multiobjective Optimization“, in *Evolutionary Multiobjective Optimization*, 2005, S. 105–145, doi: 10.1007/1-84628-137-7_6.
- [17] S. Huband, P. Hingston, L. Barone, und L. While, „A review of multiobjective test problems and a scalable test problem toolkit“, *IEEE Transactions on Evolutionary Computation*, Bd. 10, Nr. 5, S. 477–506, 2006, doi: 10.1109/TEVC.2005.861417.
- [18] K. Shang, M. L. Zhang, H. Ishibuchi, und Y. Liu, „A new R2 indicator for better hypervolume approximation“, in *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 2018, S. 745–752, doi: 10.1145/3205455.3205543, ISBN:9781450356183.
- [19] Z. Tang, Y. Zhu, D. Zhao, und S. M. Lucas, „Enhanced Rolling Horizon Evolution Algorithm with Opponent Model Learning“, *IEEE Trans. Games*, S. 1, 2020, doi: 10.1109/TG.2020.3022698.
- [20] D. Perez, S. Samothrakis, S. M. Lucas, und P. Rohlfshagen, „Rolling horizon evolution versus tree search for navigation in single-player real-time games“, in *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 2013, S. 351–358, doi: 10.1145/2463372.2463413, ISBN:9781450319638.
- [21] P. Paliyawan, K. Sookhanaphibarn, W. Choensawat, und R. Thawonmas, „Towards Social Facilitation in Audience Participation Games: Fighting Game AIs whose Strength Depends on Audience Responses“, in *IEEE Conference on Computational Intelligence and Games, CIG, 2020*, Bd. 2020-Augus, S. 686–689, doi: 10.1109/CoG47356.2020.9231633, ISBN:9781728145334.
- [22] C. F. Bond Jr., „Social facilitation“, in *Encyclopedia of psychology, Vol. 7.*, New York, NY, US: Oxford University Press, 2000, S. 338–440, doi: 10.1037/10522-145, ISBN:1-55798-656-8 (Hardcover).
- [23] C. Darwin, *On the Origin of the Species*, Bd. 5. 1859.
- [24] K. Than, „Darwin’s Theory of Evolution: Definition & Evidence | Live Science“, Feb. 27, 2018. <https://www.livescience.com/474-controversy-evolution-works.html> (zugegriffen März 11, 2021).
- [25] S. Augsten, „Was ist ein Algorithmus?“, Jan. 18, 2019. <https://www.dev-insider.de/was-ist-ein-algorithmus-a-788116/> (zugegriffen März 02, 2021).
- [26] D. E. Clark, *Evolutionary Algorithms in Molecular Design*, Bd. 8. 2008, doi: 10.1002/9783527613168, ISBN:9783527613168.
- [27] S. L. Smith, „A Review of Medical Applications of Genetic and Evolutionary Computation“, in *Genetic and Evolutionary Computation: Medical Applications*, 2010, S. 17–43, doi: 10.1002/9780470973134.ch3, ISBN:9780470748138.
- [28] P. Cortés, L. Onieva, J. Muñuzuri, und J. Guadix, „A revision of evolutionary computation techniques in telecommunications and an application for the network global planning

- problem“, *Stud. Comput. Intell.*, Bd. 92, S. 239–262, 2008, doi: 10.1007/978-3-540-76286-7_11, ISBN:9783540762850.
- [29] F. Hufsky, „Von leichten, schweren und vollständigen Problemen | BioinfoWelten“, Aug. 09, 2016. <http://bioinfowelten.uni-jena.de/2016/08/09/von-leichten-schweren-und-vollstaendigen-problemen/> (zugegriffen März 11, 2021).
- [30] W. Cook, „The Problem“. <http://www.math.uwaterloo.ca/tsp/problem/index.html> (zugegriffen März 11, 2021).
- [31] S. J. Chow, „Many meanings of ‘heuristic’“, *Br. J. Philos. Sci.*, Bd. 66, Nr. 4, S. 977–1016, 2015, doi: 10.1093/bjps/axu028.
- [32] A. Schreiber, *Heuristik - Kunst des Problemlösens*. 2015, doi: 10.13140/RG.2.1.1066.7368.
- [33] O. Bochkor und V. Krypczyk, „Metaheuristiken zur Lösung komplexer Probleme - JAXenter“, Juni 17, 2020. <https://jaxenter.de/java/metaheuristiken-loesung-komplexer-probleme-96390> (zugegriffen März 02, 2021).
- [34] D. H. Wolpert und W. G. Macready, „No free lunch theorems for optimization“, *IEEE Trans. Evol. Comput.*, Bd. 1, Nr. 1, S. 67–82, 1997, doi: 10.1109/4235.585893.
- [35] A. Leitão und P. Machado, „Mate choice in evolutionary computation“, in *Handbook of Genetic Programming Applications*, 2015, S. 155–177, doi: 10.1007/978-3-319-20883-1_7, ISBN:9783319208831.
- [36] J. Liu und W. Wu, „Coevolutionary optimization algorithm: With ecological competition model“, in *Communications in Computer and Information Science*, 2011, Bd. 237 CCIS, S. 68–75, doi: 10.1007/978-3-642-24282-3_10, ISBN:9783642242816.
- [37] E. Popovici, A. Bucci, R. P. Wiegand, und E. D. de Jong, „Coevolutionary principles“, in *Handbook of Natural Computing*, Bd. 2–4, 2012, S. 987–1033, doi: 10.1007/978-3-540-92910-9_31, ISBN:9783540929109.
- [38] J. Secretan, „Picbreeder“. <http://picbreeder.org/> (zugegriffen März 11, 2021).
- [39] J. Lehman und K. O. Stanley, „Novelty Search and the Problem with Objectives“, in *Genetic Programming Theory and Practice IX*, 2011, S. 37–56, doi: https://doi.org/10.1007/978-1-4614-1770-5_3, ISBN:978-1-4614-1770-5.
- [40] J. Lehman und K. O. Stanley, „Abandoning objectives: Evolution through the search for novelty alone“, *Evol. Comput.*, Bd. 19, Nr. 2, S. 189–222, 2011, doi: 10.1162/EVCO_a_00025.
- [41] M. Crepinsek, S. H. Liu, und M. Mernik, „Exploration and exploitation in evolutionary algorithms: A survey“, *ACM Computing Surveys*, Bd. 45, Nr. 3. 2013, doi: 10.1145/2480741.2480752.
- [42] A. Hussain, Y. S. Muhammad, M. Nauman Sajid, I. Hussain, A. Mohamd Shoukry, und S. Gani, „Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator“, *Comput. Intell. Neurosci.*, Bd. 2017, 2017, doi: 10.1155/2017/7430125.
- [43] S. Rahnamayan, H. R. Tizhoosh, und M. M. A. Salama, „A novel population initialization method for accelerating evolutionary algorithms“, *Comput. Math. with Appl.*, Bd. 53, Nr. 10, S. 1605–1614, 2007, doi: 10.1016/j.camwa.2006.07.013.
- [44] K. Zielinski, P. Weitkemper, R. Laur, und K.-D. Kammeyer, „Examination of stopping criteria for differential evolution based on a power allocation problem“, *Int. Conf. Optim. Electr. Electron. Equip.*, Bd. 3, S. 149–156, 2006.

- [45] J. Branke, K. Deb, K. Miettinen, und R. Slowinski, „Multiobjective Optimization: Interactive and Evolutionary Approaches“, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Bd. 5252 LNCS, Nr. January. Springer Science & Business Media, Berlin Heidelberg, S. 5–7, 2008, , ISBN:3540889078.
- [46] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, D. E. Goldberg, und J. R. Koza, *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition Genetic and Evolutionary Computation Series Editors Selected titles from this series* : New York: Springer, 2007, doi: 10.1007/978-0-387-36797-2, ISBN:978-0-387-33254-3.
- [47] Z. Yang, H. Wang, K. Yang, T. Back, und M. Emmerich, „SMS-EMOA with multiple dynamic reference points“, in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016*, 2016, S. 282–288, doi: 10.1109/FSKD.2016.7603187, ISBN:9781509040933.
- [48] N. Beume, B. Naujoks, und M. Emmerich, „SMS-EMOA: Multiobjective selection based on dominated hypervolume“, *Eur. J. Oper. Res.*, Bd. 181, Nr. 3, S. 1653–1669, 2007, doi: 10.1016/j.ejor.2006.08.008.
- [49] G. T. Lam, D. Logofătu, und C. Bădică, „A novel real-time design for fighting game AI“, *Evol. Syst.*, 2020, doi: 10.1007/s12530-020-09351-4.
- [50] D. Silver u. a., „Mastering chess and shogi by self-play with a general reinforcement learning algorithm“, *arXiv*, 2017.
- [51] „GitHub - TeamFightingICE/HighlightMCTS“. <https://github.com/TeamFightingICE/HighlightMCTS> (zugegriffen März 11, 2021).
- [52] A. Nebro, „GitHub - jMetal/jMetalPy: A framework for single/multi-objective optimization with metaheuristics“. <https://github.com/jMetal/jMetal> (zugegriffen März 11, 2021).
- [53] A. Nebro, „jMetal 5 Web site“. <http://jmetal.github.io/jMetal/> (zugegriffen März 11, 2021).
- [54] „GitHub - TeamFightingICE/FightingICE“. <https://github.com/TeamFightingICE/FightingICE> (zugegriffen März 11, 2021).
- [55] K. Deb und A. Deb, „Analysing mutation schemes for real-parameter genetic algorithms“, *Int. J. Artif. Intell. Soft Comput.*, Bd. 4, Nr. 1, S. 1, 2014, doi: 10.1504/ijaisc.2014.059280.
- [56] K. Deb und R. B. Agrawal, „Simulated Binary Crossover for Continuous Search Space“, *Complex Syst.*, Bd. 9, S. 1–34, 1994, doi: 10.1.1.26.8485, ISBN:0891-2513.
- [57] M. Hamdan, „Revisiting the distribution index in simulated binary crossover operator for evolutionary multiobjective optimisation algorithms“, in *2014 4th International Conference on Digital Information and Communication Technology and Its Applications (DICTAP). IEEE, 2014*, Mai 2014, S. 37–41, doi: 10.1109/DICTAP.2014.6821653, ISBN:978-1-4799-3724-0.
- [58] A. E. Eiben und S. K. Smit, „Parameter tuning for configuring and analyzing evolutionary algorithms“, *Swarm and Evolutionary Computation*, Bd. 1, Nr. 1. S. 19–31, 2011, doi: 10.1016/j.swevo.2011.02.001.
- [59] G. Karafotias, M. Hoogendoorn, und A. E. Eiben, „Parameter Control in Evolutionary Algorithms: Trends and Challenges“, *IEEE Transactions on Evolutionary Computation*, Bd. 19, Nr. 2. S. 167–187, 2015, doi: 10.1109/TEVC.2014.2308294.
- [60] E. Aoki und M. Kim, „FGAIC 2018 Competition Results“, Jan. 23, 2021. <http://www.ice.ci.ritsumei.ac.jp/~ftgaic/Downloadfiles/2018Competition.zip> (zugegriffen

März 02, 2021).

- [61] Y. I. Cherifi, „FGAIC 2017 Competition Results“, Aug. 22, 2017.
<http://www.ice.ci.ritsumei.ac.jp/~ftgaic/Downloadfiles/2017Competition.zip> (zugegriffen März 02, 2021).
- [62] J. Hedengren, „Minimax and Maximin Optimization“.
<http://apmonitor.com/me575/index.php/Main/MiniMax> (zugegriffen März 11, 2021).
- [63] H. Ishibuchi, N. Tsukamoto, Y. Sakane, und Y. Nojima, „Hypervolume approximation using achievement scalarizing functions for evolutionary many-objective optimization“, in *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, 2009, S. 530–537, doi: 10.1109/CEC.2009.4982991, ISBN:9781424429592.
- [64] S. Künzel und S. Meyer-Nieberg, „Coping with opponents: multi-objective evolutionary neural networks for fighting games“, *Neural Comput. Appl.*, Bd. 32, Nr. 17, S. 13885–13916, 2020, doi: 10.1007/s00521-020-04794-x.
- [65] S. Haykin, *Neural networks and learning machines*, 3/E. Pearson Education India, 2010, , ISBN:933258625X.

Anhang A

Untersuchte Fitnessfunktionen in der FTG:

1. Minimierung des erlittenen Schadens.
2. Maximierung des verursachten Schadens.
3. Minimierung der Distanz zum Gegner.
4. Maximierung des Zeitintervalls zwischen einem erfolgreichen Treffer einer Aktion und einem erfolgreichen Treffer des Gegners.
5. Minimierung des Zeitintervalls des letzten erfolgreichen Treffers durch eine Aktion.
6. Minimierung *startup* Frames von Aktionen bei der Aktionsauswahl.
7. Maximierung des verursachten Schadens unter Berücksichtigung des erlittenen Schadens.
8. Maximierung des Energiestandes.
9. Minimierung der *frameNumber* bei der Aktionsauswahl.
10. Maximierung, der Anzahl an Aktion, welche dem Gegner Schaden zufügen.

„Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst,
die Zitate ordnungsgemäß gekennzeichnet und keine anderen, als die im
Literatur/Schriftenverzeichnis angegebenen Quellen und Hilfsmittel benutzt zu haben.“

Ferner habe ich vom Merkblatt über die Verwendung von studentischen Abschlussarbeiten
Kenntnis genommen und räume das einfache Nutzungsrecht an meiner Bachelorarbeit der
Universität der Bundeswehr München ein.

A handwritten signature in black ink, appearing to be 'A. F. B.', written over a horizontal line.