technische universität
dortmund

Abschlussbericht

Projektgruppe 529: Spielercharaktere - Modellierung menschenähnlicher Gegenspieler in Strategiespielen mit Techniken der Computational Intelligence

Niels Ackermann, Alireza Gholaman,
Marc Gorzala, Matthias Grochowski,
Thomas Harweg, Markus Kemmerling,
Daniel Spierling, Sebastian Uellenbeck,
Wolfgang Walz
29. Juli 2009

INTERNE BERICHTE

Betreuer:
Nicola Beume
Boris Naujoks
Mike Preuß

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.uni-dortmund.de

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Anreize zur Durchführung der Projektgruppe 1
1.2 Menschenähnliche Gegenspieler in Strategiespielen 1
1.2.1 Stärken und Schwächen von NPCs 2
1.2.2 Kommunikation mit Computerspielern 2

2 CI-Methoden **4**
2.1 Neuronale Netze 4
2.2 Evolutionäre Algorithmen 6
2.3 Fuzzy-Systeme 7
2.3.1 Fuzzifizierung 8
2.3.2 Inferenzsystem 8
2.3.3 Defuzzifizierung 8
2.4 Influence Map 9

3 Erstes Projekt: Poker **11**
3.1 Ziele des Projektes 12
3.2 Planung des Projektes 13
3.3 Verwirklichung des Projektes 13
3.3.1 Vermenschlichung: Ansatz für einen menschenähnlichen Computerspieler 13
3.3.2 Emoticons: Neue Kommunikationswege werden geschaffen 14
3.3.3 CI-Methoden: Der Kern des Spielens 14
3.3.4 Gegnermodellierung: Ein aufmerksames Gedächtnis 15
3.4 Erkenntnisse aus der Projektdurchführung 16

4 Zweites Projekt: Diplomacy **17**
4.1 Regeln 19
4.2 Grundlegende Züge 19
4.2.1 Halten 19
4.2.2 Bewegen/Angreifen 19
4.2.3 Unterstützen 20
4.2.4 Konvoi 20
4.3 Strategien 20
4.4 Vorhandener Rahmen 21
4.4.1 Kommunikationsprotokoll 21
4.4.2 Diplomacy-Server 22
4.4.3 DAIDE-Mapper 22
4.4.4 DAIDE Java AI Communication API 23
4.5 Existierende NPCs 23
4.5.1 DumbBot 23
4.5.2 Diplomat-Bot 23
4.5.3 BlabBot 24
4.5.4 The Israeli Diplomat 24
4.5.5 The Bordeaux Diplomat 25
4.5.6 LA Diplomat 25
4.5.7 Diplominator 25
4.5.8 Holdbot 26

5 Entwickelte Diplomacy NPCs **27**
5.1 NICE 27

5.1.1 Konzept . . . 27
5.1.2 Taktik-Module . . . 28
5.1.2.1 Cluster: Die Phalanx des NICE-Bots . . . 28
5.1.2.2 EA: Die diverse Evolution der Staatengebilde . . . 28
5.1.2.3 Eröffnungsbuch: Ein guter Start in *Diplomacy* . . . 30
5.1.2.4 Priorisierung taktischer Zielsetzungen . . . 30
5.1.3 Verhandlungsmodule . . . 33
5.1.3.1 ALY: Das Schmieden von Allianzen . . . 33
5.1.3.2 XDO: Der Schrei nach Unterstützung . . . 34
5.1.3.3 DRW: Das Angebot einer friedlichen Koexistenz . . . 36
5.1.3.4 DMZ: Die Befriedung von Provinzen . . . 36
5.1.4 Modultest . . . 36
5.1.5 Fazit . . . 39
5.2 Stragotiator . . . 40
5.2.1 Planung . . . 40
5.2.1.1 Paperinhalte . . . 42
5.2.1.2 Grundidee . . . 47
5.2.1.3 Ziele . . . 47
5.2.2 Aufbau und Realisierung . . . 48
5.2.2.1 Negotiator . . . 48
5.2.2.2 Strategie Kern . . . 59
5.2.2.3 EA-Optimierung . . . 65
5.2.2.4 Gegnerbewertung . . . 67
5.2.3 Parameteroptimierung . . . 68
5.2.3.1 Negotiator . . . 68
5.2.3.2 Strategie Kern . . . 75
5.2.3.3 EA-Optimierungstests . . . 77
5.2.4 Fazit . . . 79
5.2.5 Ausblick . . . 79
5.2.5.1 Negotiator . . . 79
5.2.5.2 Strategie Kern . . . 80
5.2.5.3 EA-Optimierung . . . 82
5.2.5.4 Gegnerbewertung . . . 84
5.3 FrySpi . . . 84
5.3.1 Ziele des NPC . . . 84
5.3.2 Ursprüngliche Planung . . . 85
5.3.2.1 Movement - Truppenbewegung . . . 85
5.3.2.2 Attacking - Angriff . . . 85
5.3.2.3 Defending - Verteidigung . . . 85
5.3.2.4 SupportDetector - Unterstützung . . . 86
5.3.2.5 BlockSupport - Zusammenhalt . . . 86
5.3.2.6 DynamicLocationWeight - Bewertung der Provinzen . . . 86
5.3.2.7 Negotiation - Verhandlung . . . 86
5.3.2.8 Emotion - Modellierung der Emotionen . . . 86
5.3.3 Entwicklungsframeworks . . . 87
5.3.3.1 Kommunikation . . . 87
5.3.3.2 Game . . . 88
5.3.3.3 KI . . . 88
5.3.4 Entwicklungsstufen KI . . . 88
5.3.4.1 Spielstärke . . . 89
5.3.4.2 Analyzer . . . 90
5.3.4.3 Menschlichkeit . . . 93
5.3.5 Erkenntnisse aus der Durchführung . . . 94
5.3.6 Fazit . . . 95

6 Statistische Auswertung **96**
6.1 Computergestütze Anayse . . . 96
6.1.1 die unterschiedlichen Spielkonstellationen . . . 96
6.1.1.1 Jeder gegen jeden . . . 97
6.1.1.2 CrySpi vs. FrySpi . . . 98
6.1.1.3 Strago vs. Strago (mit EA) . . . 100
6.1.1.4 CrySpi vs. Nice-Bot . . . 100
6.1.1.5 CrySpi vs. Strago . . . 101
6.1.1.6 Nice-Bot vs. Strago . . . 103
6.1.2 Gesamtergebnis . . . 105
6.1.3 Angriffs- und Erfolgsanalyse der Bots . . . 110
6.1.3.1 CrySpi . . . 110
6.1.3.2 FrySpi . . . 110
6.1.3.3 Diplominator . . . 112
6.1.3.4 Nice . . . 112
6.1.3.5 Strago mit EA . . . 113
6.1.3.6 Strago ohne EA . . . 113
6.1.4 Fazit . . . 114
6.2 Contest Software . . . 115
6.3 Durchführung der Contests . . . 116
6.4 Turing Test . . . 116
6.4.1 Durchführung . . . 117
6.4.2 Ergebnisse des Turingtest . . . 117
6.4.3 Fazit . . . 119

7 Fazit **121**
7.1 Verbessungspotential und Erkenntnisse für andere Projekte . . . 121

A Anhang **123**
A.1 Diplomacy Regeln . . . 123
A.2 Diplomacy Abkürzungen . . . 124
A.2.1 Großmächte . . . 124
A.2.2 Provinzen . . . 125

B Seminararbeiten **126**
B.1 Avatare und Kommunikation mit Computerspielern . . . 126
B.2 Emotionen in der klassischen KI . . . 126
B.3 Evolutionäre Algoithmen . . . 126
B.4 Fuzzy-Systeme . . . 126
B.5 Künstliche Intelligenz in den Spielen Poker, Diplomacy und Junta . . . 127
B.6 Ludologie . . . 127
B.7 Maschinelles Lernen/Regelbasierte Steuerung . . . 127
B.8 Künstliche Neuronale Netze . . . 127
B.9 Soziologie/Psychologie: Modelle für Emotionen/Verhalten von Spielern . 127

C PG-Ordnung **129**
C.1 PG-Vereinbarungen . . . 129
C.1.1 Abstimmungsmodus . . . 129
C.1.2 Organisatorisches . . . 129
C.2 Etikette . . . 129

Abbildungsverzeichnis **130**

Literaturverzeichnis **132**

1 Einleitung

Die Projektgruppe 529 (PG529) untersuchte die Methoden der Computational Intelligence (CI) für ihre Einsatzmöglichkeiten in Computerspielen. Sie wurde im Rahmen einer Pflichtveranstaltung der technischen Universität Dortmund im Sommersemester 2008 und Wintersemester 2008/2009 durchgeführt.

1.1 Anreize zur Durchführung der Projektgruppe

Die Produktion und der Vertrieb von Computerspielen stellt einen wesentlichen wirtschaftlichen Faktor dar, welcher in den vergangenen Jahren zunehmend an Bedeutung gewonnen hat. Im Gegensatz zur Film- und Phonoindustrie unterliegt die Spieleindustrie seit mehreren Jahren einem stetigen Umsatzwachstum und man geht davon aus, dass sie in naher Zukunft die Umsatzzahlen der Filmindustrie erreichen beziehungsweise sogar überflügeln wird.

Bei der Entwicklung neuer Computerspiele zeichnet sich der Trend ab, den Fokus auf immer komplexere und somit in der Entwicklung zeitaufwendigere Grafik zu legen. Der Entwicklung von menschenähnlicheren Computergegnern wird allerdings eine wesentlich geringere Bedeutung zugewiesen und so kommt es häufig vor, dass erfahrene Spieler Computergegner leicht als solche erkennen und in der Lage sind, die oft statischen Strategien effektiv zu analysieren und vorherzusagen. Durch dieses monotone, vorhersehbare Verhalten der Computergegner wird insbesondere der Langzeitspielspaß vermindert und der Unterhaltungswert sinkt. Da die Grafik in modernen Computerspielen immer realistischer wird und die Unterschiede diesbezüglich zwischen den einzelnen Produkten immer geringer werden, reicht die Grafik als alleiniges Qualitätsmerkmal eines Spieles nicht mehr aus. Authentische Interaktion mit der Spielwelt, insbesondere mit den Nichtspielercharakteren, sowie konsistente Handlungsweise dieser gewinnen zunehmend an Bedeutung und nehmen maßgeblich Einfluss auf den Spielspaß.

Mit Computational Intelligence (CI) werden in der Informatik jene Methoden bezeichnet, welche von einem biologischen Vorbild inspiriert wurden. Zu diesen zählen unter anderem evolutionäre Algorithmen, neuronale Netze und Fuzzy-Systeme, welche im nächsten Kapitel noch detailliert vorgestellt werden. Diese Methoden können zum Lernen und Optimieren eingesetzt werden und zeichnen sich dadurch aus, dass sie mit unvollständigen oder ungenauen Informationen umgehen können. Im Gegensatz zu statischen Methoden können sich diese CI-Methoden relativ leicht an neue Situationen anpassen. Die Idee, das vorhandene Gefüge statischer Entscheidungsfindung von Computergegnern aufzubrechen und mit Hilfe von CI-Methoden um Dynamik und Lernfähigkeit zu erweitern, scheint eine gute Möglichkeit zu sein, den Spielspaß durch gesteigerte Variation zu erhöhen.

1.2 Menschenähnliche Gegenspieler in Strategiespielen

Bei langandauernden Spielen hat die konsistente Spielweise beziehungsweise der dargestellte Charakter eines Computerspielers einen sehr großen Einfluss auf die Bewertung der Menschenähnlichkeit. Wird das Verhalten des Spielers als nicht nachvollziehbar eingeschätzt oder handelt der Spieler planlos, so wird sich dies zwangsläufig auf die Plausibilität auswirken. Auffällig bei vielen Computerspielen diesbezüglich ist, dass sich die zur Wahl stehenden Computerspieler häufig nur in ihrer strategischen Spielstärke unterscheiden und man zum Beispiel zwischen leichten, normalen und starken Gegenspielern

auswählen kann. Für den Spieler bietet sich nur die Herausforderung, nach und nach sämtliche Schwierigkeitsstufen zu bewältigen, und sollte er dies erreicht haben, wird das Spiel für ihn sehr schnell uninteressant, da keine weiteren Überraschungen zu erwarten sind. Möchte man dem Computerspieler durch Hinzufügen eines Charakters mehr Tiefe geben und so probieren, eine menschenähnlichere Spielweise zu erreichen und den Spielspaß zu erhöhen, so ist es sinnvoll, sich mit folgenden Fragen zu beschäftigen:

- Was sind menschliche Verhaltensweisen und wie sind sie motiviert?
- Welche Emotionen sind in dem Spiel sinnvoll und wie kann man sie darstellen?
- Wie kann man typische Verhaltensmuster und Motivationen mit Hilfe von CI-Methoden nachbilden?

Während der zweisemestrigen Arbeitsphase hat sich die Projektgruppe mit der Beantwortung dieser Fragen und der Realisierung von menschenähnlichen Spielern in zwei unterschiedlichen Computerspielen beschäftigt. Im ersten Semester stand die Modifikation von PokerTH im Vordergrund. Dies ist eine PC-Version von Texas Hold'em, einer Variante des Kartenspiels Poker. Im zweiten Semester wurde die Gruppe aufgeteilt in drei Teilgruppen, welche unterschiedliche Ansätze für menschenähnliche Spieler für das Spiel Diplomacy entwickelten. Diplomacy ist ein Brettspiel, bei welchem ein wesentlicher Aspekt die Kommunikation zwischen den unterschiedlichen Mitspielern ist. Die Kommunikation zwischen den Mitspielern wurde hierbei über ein existierendes Kommunikationsprotokoll ermöglicht. In den folgenden Kapiteln werden die notwendigen Grundlagen sowohl der CI-Methoden als auch der Spiele erläutert und die Ziele, Planungen und Realisierungen der beiden Projekte dokumentiert. Die Erkenntnisse aus der Projektdurchführung wurden zusammengefasst und in einem Fazit zusammengeführt.

1.2.1 Stärken und Schwächen von NPCs

Bisher lag das Hauptaugenmerk bei der Entwicklung von Nicht-Personen-Charakteren (NPC) darauf, dass sie möglichst spielstark sind. Dies ist auch ganz gut verständlich, dass dies erstmal als am wichtigsten betrachtet wurde, um überhaupt einen Gegner zu haben, gegen den das Spielen nicht absolut lächerlich ist. Nachdem man dieses Ziel - rein strategisch betrachet - nun in vielen Fällen erreicht hat, fiel auf, dass auch Bots die zwar spielstark sind, doch häufig auffallen, indem sie Strategien haben, die nicht viel mit denen zu tun haben, die auch Menschen wählen würden. Dass sie immer die gleichen Lösungswege wählen, diese aber aufgrund von verschiedenen überlegenen Eigenschaften des Computers, doch zu Ziel führen.

Aus diesen Gründen ist der Spielspaß häufig kleiner, wenn man gegen den Computer spielt und nicht gegen einen Menschen.

1.2.2 Kommunikation mit Computerspielern

Bei der Kommunikation zwischen menschlichen Spielern wird häufig ein Textchat[1] oder *Voice over IP*[2] eingesetzt. Für die Kommunikation mit Computerspielern sind diese beiden Ansätze problematisch. Der Einsatz von gesprochener Sprache wie bei "Voice over IP" scheitert alleine schon an der sehr schwierigen Spracherkennung. Auch bei geschriebener Sprache tritt dieses Problem in ähnlicher Form auf. Da auch die Erkennung von Grammatik- bzw. Rechtschreibfehlern und deren Behandlung schwierig ist. Und selbst wenn es gelingt, die exakte Wortfolge zu ermitteln, muss die Bedeutung jedes

[1] Mitteilungen werden als Text per Tastatur eingegeben und verschickt.
[2] Ein Programm, das parallel zum Spiel arbeitet und die in ein Mikrofon gesprochenen Nachrichten an den Mitspieler übermittelt, der ihn dann über Lautsprecher hört.

Wortes bekannt sein und der Satz, eventuell bezogen auf einen Kontext (z.B. was im Spiel passiert) und dem vorher Gesagten, interpretiert werden, um anschließend zu antworten. Es ist also sehr schwierig, sich mit einer Maschine wie mit einem Menschen zu unterhalten.

Der 1950 von Alan Turing erfundene Turing-Test (vgl. [Tur50]) testet die kommunikativen und die damit zusammenhängenden kognitiven Fähigkeiten einer Maschine. Bei diesem Test kommuniziert ein Mensch nur über Tastatur und Bildschirm mit zwei Gesprächspartnern, von denen einer ein Mensch und einer eine Maschine ist, und stellt ihnen Fragen. Kann der Fragende am Ende des Tests nicht genau sagen, wer von den beiden die Maschine ist, hat sie den Test bestanden. Bis heute ist das noch keinem Computerprogramm gelungen.

In Spielen wird deswegen bislang keine natürliche Sprache für die Kommunikation mit Computerspielern verwendet. Stattdessen kann ein menschlicher Spieler durch Auswahl vorgegebener Sätze oder Satzstücke, die nach bestimmten Regeln kombiniert werden können, (für ein Beispiel siehe Abschnitt 4.4.1 auf Seite 21) mit seinen computergesteuerten Mitspielern kommunizieren.

2 CI-Methoden

Die Entwicklungen der Projektgruppe 529 basieren auf Methoden der Computational Intelligence. Das folgende Kapitel befasst sich mit den grundlegenden Methoden um auch Lesern ohne Vorkenntnisse den Zugang zu diesem Endbericht zu ermöglichen. Die Komplexität dieser Artikel richtet sich primär nach der Bedeutung der Verfahren für diese Projektgruppe.

2.1 Neuronale Netze

Neuronale Netze sind eine Nachbildung des menschlichen Gehirns durch gewichtete gerichtete Graphen. Falls der Graph azyklisch ist, wird das Netz „Feed Forward“, sonst „Rekursiv“ genannt. Die Knoten eines *Neuronalen Netzes* werden als *Neuronen* bezeichnet und stellen Recheneinheiten dar. Die einfachste Form eines *Neurons* ist das *McCullon-Pitts Neuron* (siehe Abbildung 1).

Es besitzt n erregende und m hemmende binäre Eingänge sowie einen binären Ausgang y und einen Schwellenwert θ. Das *Neuron* berechnet die Funktion

$$y(x_1, \ldots, x_n, \ldots, x_{n+m}, \theta) = \begin{cases} 1 & \text{falls } \sum_{i=1}^{n} x_i \geq \theta \text{ und } \sum_{i=n+1}^{n+m} x_i = 0 \\ 0 & \text{falls } \sum_{i=1}^{n} x_i < \theta \text{ oder } \sum_{i=n+1}^{n+m} x_i > 0 \end{cases}$$

Neuronale Netze, die mit dieser Art von Neuronen arbeiten, werden *McCullon-Pitts Netze* genannt.

Eine Erweiterung der *McCullon-Pitts Netze* bilden gewichtete Netze mit *Perceptronen* (siehe Abbildung 2) als *Neuronen*.

Perceptronen besitzen ebenfalls einen Schwellenwert θ, eine binäre Ausgabe y und n binäre Eingänge, die im Gegensatz zu den *McCullon-Pitts Neuron* Gewichte $w_1, \ldots, w_n$ haben. Die von ihnen berechnete Funktion ist

$$y(x_1, \ldots, x_n, w_1, \ldots, w_n, \theta) = \begin{cases} 1 & \text{falls } \sum_{i=1}^{n} x_i \cdot w_i \geq \theta \\ 0 & \text{sonst} \end{cases}$$

Um eine differenzierbare Ausgabe zu erhalten, berechnen *Neuronen* jedoch häufig eine sigmoide Funktion mit einer zu wählenden Konstante c. Sie werden als *sigmoide Neuronen* bezeichnet.

$$y(x_1, \ldots, x_n, w_1, \ldots, w_n, \theta) = \frac{1}{1 + e^{-c \cdot ((\sum_{i=1}^{n} x_i \cdot w_i) - \theta)}}$$

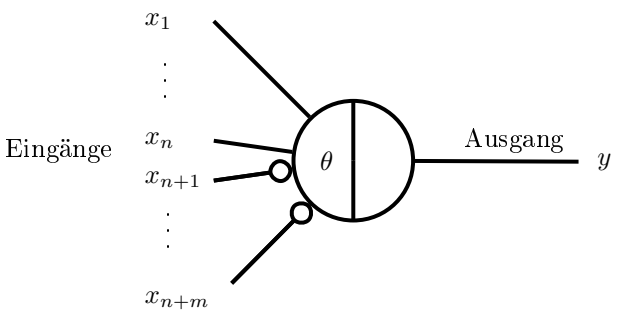


Abbildung 1: McCullon-Pitts Neuron

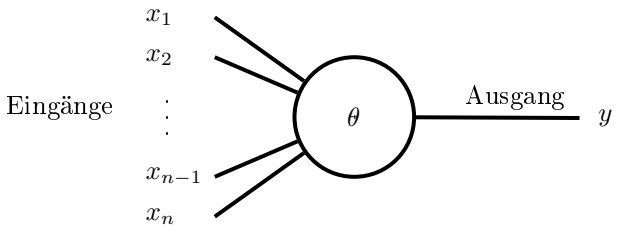


Abbildung 2: Perceptron

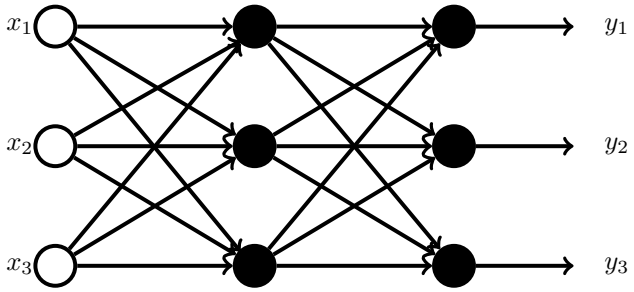


Abbildung 3: Ein in Schichten unterteilt Neuronales Netz

Häufig werden *Neuronale Netze* in Schichten unterteilt (siehe Abbildung 3). Dabei gibt es nur Kanten zwischen zwei aufeinander folgenden Schichten i und $i+1$ und niemals zwischen i und $j > i+1$. Die Eingabeschicht, im Bild ganz links, berechnet nichts, sondern stellt lediglich die Eingaben des Netzes zur Verfügung. In der Ausgabeschicht, im Bild ganz rechts, werden die Ausgaben des Neuronalen Netzes berechnet. Die dazwischen liegenden Schichten werden *hidden layer*, auf Deutsch „versteckte Schichten", genannt.

Bei einem in Schichten unterteilten gewichteten *Neuronalen Netz*, welches *sigmoide Neuronen* benutzt, kann zur Optimierung der Kantengewichte das sogenannten *Backpropagation* genutzt werden. Dabei wird die mit der Ableitung der sigmoiden Funktion multiplizierte Differenz zwischen der Soll- und Ist-Ausgabe - dem Fehler - eines *Neurons* mit den bisherigen Gewichten der eingehenden Kanten verrechnet. Befindet sich das *Neuron* nicht in der Ausgabeschicht, errechnet sich der Fehler aus der mit den Kantengewichten gewichteten Summe der Fehler der nachfolgenden *Neuronen*. Der Fehler wird also von hinten nach vorne zurück propagiert. Durch dieses Verfahren, was mathematisch einem Gradientenabstieg entspricht, wird der Fehler sukzessiv minimiert und die Kantengewichte optimiert. Es konvergiert nach endlicher Zeit in einem lokalen Optimium.

Generell kann zur Verbesserung der Kantengewichte, auch bei Verwendung von nicht *sigmoiden Neuronen*, jedes allgemeine Optimierungsverfahren, zum Beispiel ein *Evolutionärer Algorithmus* (siehe Abschnitt 2.2 auf Seite 6), genutzt werden. Bei einem *EA* ist allerdings nicht sicher gestellt, dass er in einem lokalen Optimum konvergiert.

2.2 Evolutionäre Algorithmen

Evolutionäre Algorithmen(EA) stellen randomisierte Optimierungsverfahren dar, welche die biologische Evolution als Vorbild haben. Der Grundgedanke dabei ist, dass sich aus gefundenen Lösungen für das Optimierungsproblem ähnliche und hoffentlich auch bessere Lösungen ableiten lassen. In der Botanik verwendet man, um beispielsweise dem Ziel der Züchtung schneller wachsender und witterungsresistenterer Getreidesorten näher zu kommen, zunächst Ableger einer Sorte, welche die gewünschten Eigenschaften noch nicht ausreichend besitzen. Diese werden dann in Kreuzungsverfahren veredelt, so dass sie die gewünschten Eigenschaften annehmen. Ähnlich ist das Vorgehen bei EA.

Um für ein Problem eine gute Lösung zu finden, muss bestimmbar sein, wie gut diese Lösung ist. Eine solche Bewertungsfunktion wird auch als „Ziel"- oder „Fitnessfunktion" bezeichnet. Je besser eine Lösung ist, desto höher (oder wenn es ein Minimierungsproblem ist: desto niedriger) der Fitnesswert. In Anlehnung an die Biologie werden bei EAs Lösungen auch als „Individuen" und eine Lösungsmenge als „Population" bezeichnet.

Aus einer Population werden, meistens auf Grundlage ihrer Fitnesswerte, eine Menge von Individuen, Eltern genannt, ausgewählt. Aus diesen Eltern werden dann neue Individuen erzeugt, welche Nachkommen genannt werden.

Für die Erzeugung der Nachkommen gibt es zwei grundlegende Konzepte:

- *Mutation*
- *Rekombination*

Bei der Mutation wird ein Kind, welches zuerst eine exakte Kopie seines Elternindividuums darstellt, durch Zufallseinfluss verändert. Der Grad der Veränderung lässt sich dabei parametrisieren. Als Beispiel könnte der folgender Bitstring $(1, 0, 1, 0, 1)$ mit einer zufälligen Invertierung der Bits (hier das dritte und vierte Bit) den Bitstring $(1, 0, 0, 1, 1)$ bilden.

Die Rekombination wählt dagegen zwei (oder mehr) Elternindividuen und bildet aus zufällig ausgesuchten Teilen der Eltern ein neues Kindindividuum. Die Individuen x_1, x_2, x_3 und y_1, y_2, y_3 können beispielsweise zu folgendem neuen Individuum rekombiniert werden: x_1, x_2, y_3
Anschließend wird in der Regel das durch Rekombination gebildete Kindindividuum mutiert. Bei beiden Verfahren hat man die Hoffnung, dass gute Eigenschaften der Eltern erhalten bleiben oder sogar verstärkt werden.

Trotz der beschränkten Darstellungsweise von reellen Zahlen und einem beschränkten Speichermodell, gibt es zu viele mögliche Kandidaten für Individuen. Sie alle zu erzeugen ist praktisch nicht durchführbar, weswegen man sich auf die Untersuchung einer kleineren Population beschränkt. Somit müssen einzelne bisher berechnete Individuen gelöscht werden. Dies sollten diejenigen Individuen sein, die wahrscheinlich nicht mehr zur Verbesserung der Population beitragen können. Es stellt sich nun die Frage wie man diese „guten" oder potentiell guten Individuen (Individuen, welche durch entsprechende Mutationen oder Rekombinationen befriedigende Ergebnisse erzielen) findet bzw. auswählt. Ein Ansatz ist aus der gesamten Population nach einer vorgegebenen Definition „gute" Individuen auszuwählen. Dabei werden Individuen verschiedener Generationen gleich bewertet und haben (abgesehen von ihrem Fitness-Wert) die gleichen Chancen ausgewählt zu werden. Dieser Selektionstyp wird auch Plus-Selektion genannt. Steht man auf dem Standpunkt, dass nur neue Individuen erhaltenswert sind, bietet sich die Komma-Selektion an. Dabei werden die Überlebenden nur unter den Nachkommen gesucht.

Im Weiteren ist die Frage zu klären, wann ein Individuum gut genug ist, um zu überleben. Ein naheliegendes Auswahlverfahren ist die Bestenselektion, bei der die besten Individuen ausgewählt werden, um zukünftige Eltern zu werden. Dieses Verfahren ist sehr einfach zu implementieren und findet auch häufig Anwendung. Eine weitere Möglichkeit ist die q-stufige Turnierselektion, bei der jedes Individuum mit q zufällig ausgesuchten Individuen verglichen wird. Für jeden Vergleich, bei dem das Individuum besser ist als sein zufällig ausgewählter „Gegner", erhält es einen Punkt. Zum Schluss werden die Individuen selektiert, welche die meisten Punkte erhalten haben. Der Aufwand dieses Verfahrens steigt mit wachsendem q. Auch kann dieses Verfahren zu einer „weicheren" Selektion führen, da ein schwächeres Individuum auch überleben kann, wenn dessen Gegner noch schwächer waren.

Als letztes ist zu klären, wie lange dieses Optimierungsverfahren dauern soll. Dies wird durch ein zuvor definiertes Abbruchkriterium festgelegt. Hierzu kann man die Anzahl der Generationszyklen oder die Rechenzeit in Minuten beschränken. Ebenfalls bieten sich qualitative Kriterien an, wie das Erreichen einer gewissen Mindestgüte einer gefundenen Lösung oder ein mögliches Stagnieren der Qualität der Population. Letzteres kann so realisiert werden, dass ein Abbruch erfolgt, wenn innerhalb von k Generationen keine Verbesserung erreicht wird. Natürlich kann man es auch dem Benutzer selbst ermöglichen, den Optimierungsprozess durch ein externes Abbruchsignal vorzeitig zu beenden.

Zusammenfassend lässt sich der Aufbau eines EA wie folgt darstellen:

- *Initialisierung* einer Population
- *Evaluierung* der Individuen mit Zielfunktion
- solange *Abbruchkriterium* nicht erfüllt ist, starte Generationszyklus:
  - ◇ *Selektion* zur *Reproduktion* (Elternindividuenauswahl)
  - ◇ *Reproduktion* durch Variation (Mutation/Rekombination)
  - ◇ *Evaluierung* der Nachkommen
  - ◇ *Selektion* der *Überlebenden* (Schrumpfen der Population)
- *Ausgabe* des besten Individuums

2.3 Fuzzy-Systeme

Die Fuzzy-Logik stellt eine Erweiterung der klassischen binären Logik dar. Im Gegensatz zur Binärlogik, die nur zwischen zwei Zuständen unterscheidet, wird diese Restriktion aufgehoben und sämtliche reelle Zahlen im Bereich zwischen null und eins zugelassen. Durch diese Erweiterung wird die Möglichkeit zur Behandlung von Unsicherheiten und Unschärfen in einem System geschaffen. Ihren Ursprung findet die Fuzzy-Logik bereit in der griechischen Antike durch den Philosophen Platon, jedoch gewinnt sie erst nach der Entwicklung der Fuzzy-Set-Theorie im Jahr 1965 durch Lotfi A. Zadeh[Zad65] an Bedeutung. Ein klassisches Fuzzy-System besteht aus drei verschiedenen Teilen:

- Fuzzifizierung
- Regelsystem
- Defuzzifizierung

2.3.1 Fuzzifizierung

Die Fuzzifizierug wandelt Eingabedaten zur Verarbeitung durch das Regelsystem vor. Hierbei wird scharfen Eingabedaten ein Zugehörigkeitsgrad zu Mengen zugeordnet, die vom spezifischen Anwendungsfall abhängig sind. Ein Eingabedatum wird dabei in so genannte Fuzzy-Sets unterteilt. Dabei repräsentiert jede Menge eine Abstufung einer bestimmten Eigenschaft. Die Zugehörigkeit zu mehreren Mengen stellt dabei keine Ausnahme dar und ist primär an den Übergängen zu finden. Verschiedene Zugehörigkeitsgrade zu den Mengen ermöglichen im Gegensatz zur Binärlogik einen weichen Übergang zwischen zwei natürlichsprachlichen Eigenschafen, den sogenannten linguistischen Termen.

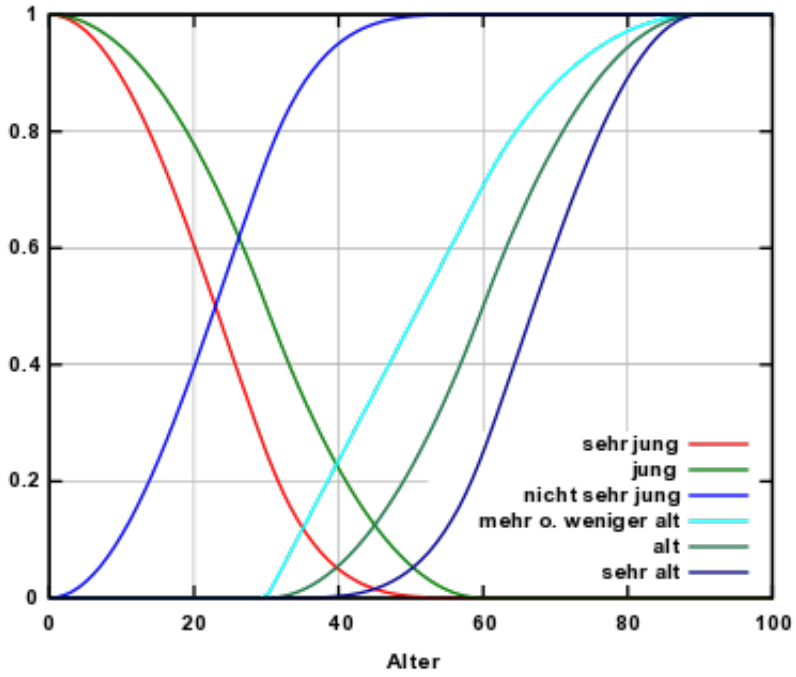


Abbildung 4: Fuzzy Sets: Zuordnung eines Alters in Fuzzy-Sets

2.3.2 Inferenzsystem

Der Kern eines Fuzzysystems ist das Regelsystem, das anhand der modifizierten Eingabedaten Folgerungen durchführt. Der Aufbau des Inferenzsystems ist sehr stark an ein binäres Regelsystem angelehnt. Der Unterschied liegt darin, dass die Verknüpfungen von Werten für den erweiterten Eingabebereich angepasst werden und ein ebenfalls reellwertiges Resultat erzeugen. Als Ersatz für die Und- und Oder-Verknüfung wird häufig die Minimum- bzw. Maximum-Funktion verwendet. Eine Alternative dazu stellen Algebraisches Produkt und Summe dar, die insbesondere in Neuro-Fuzzy-Systemen verwendet werden, da für den Backpropagation-Algorthmus eine differenzierbare Funktion verwendet werden muss.

2.3.3 Defuzzifizierung

Das Ergebnis des Inferenzsystems ist eine Zuordnung zu Fuzzy-Sets. In vielen Anwendungsbereichen, wie beispielsweise der Steuerungs- und Regeltechnik, ist es daher not-

wendig diese Zuordnungen in eine Ausgabe für das System transformieren. Der letzte Schritt eines Fuzzy-Systems liegt folglich darin, aus diesen Mengenzugehörigkeiten je nach Anforderung des Systems wieder binäre oder reellwertige Ausgaben z. B. als Anweisungen zu produzieren.

2.4 Influence Map

das Erstellen einer Influence Map ist eine CI-Methode, welche häufig eingesetzt wird, um Computerspieler in Strategiespielen bei der Bewertung von Spielsituationen sowie der Entscheidungsfindung zu unterstützen. Sie sind eine topographische Darstellung eines bestimmten Aspekts der aktuellen Spielsituation und repräsentieren den Einfluss der Einheiten und Spieler an den unterschiedlichen Positionen der Spielkarte. Die Influence Map kann dafür gebraucht werden, die Grenzen unterschiedlicher Mächte anzuzeigen. Obwohl Influence Maps in jeglicher Topologie eingesetzt werden können, wird sich das folgende erklärende Beispiel auf eine zweidimensionale Umgebung beschränken, da dies für die meisten Strategiespiele und den Kontext der Projektgruppe vollkommen ausreichend ist.

Auf dem Spielfeld befinden sich drei eigene Einheiten sowie eine Einheit von einem gegnerischen Spieler (Abbildung 5 auf Seite 9 zeigt die Ausgangssituation).

Abbildung 5: Influence Maps: die Ausgangssituation des Beispiels

Bei der Erstellung der Influence Map wird die Spielkarte zunächst in Zellen aufgeteilt. Um die Zuordnung zu den einzelnen Zellen zu ermöglichen, wird ein Gitternetz über die Karte gelegt. Abbildung 6 auf Seite 9 zeigt die gerasterte Karte unseres Beispiels.

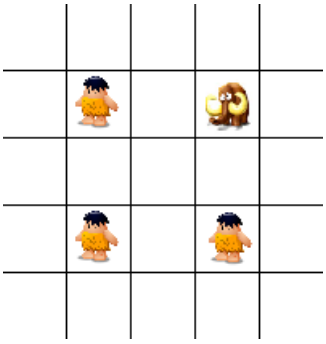


Abbildung 6: Influence Maps: die Rasterung der Spielkarte

Nachdem die Karte in verschiedene Zellen aufgeteilt wurde, wird nun der Einfluss für jede Einheit berechnet. Der Einfluss einer Einheit kann von einem oder mehreren Faktoren, wie zum Beispiel der Anzahl der Lebenspunkte, der Bewaffnung etc., abhängen. In unserem Beispiel hat jeder der Krieger eine Stärke von fünf und die gegnerische Einheit eine Stärke von zehn. Diese Werte werden zunächst auf der Karte eingetragen (vgl. Abbildung 7 auf Seite 10).

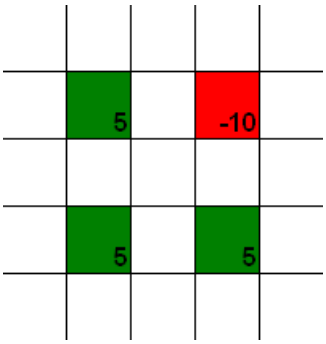
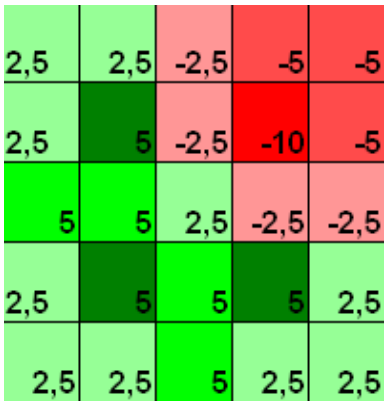
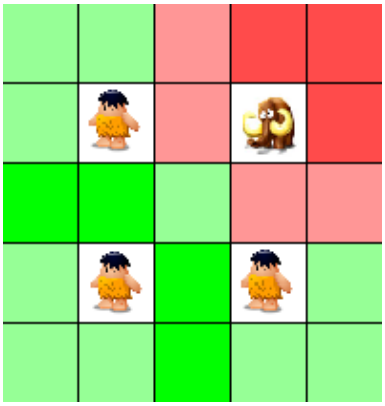


Abbildung 7: Influence Maps: die Stärke der Spieler

Der Einfluss der Einheiten wird dann mit Hilfe einer Funktion über die Karte propagiert, wobei sehr häufig der Wert der Propagierungsfunktion mit größer werdendem Abstand abnimmt. So gibt es unter anderem lineare beziehungsweise exponentielle Propagierungsfunktionen, bei welchen der Funktionswert in linearer oder exponentieller Weise in Abhängigkeit von der Distanz abnimmt. In unserem sehr rudimentären Beispiel sei der Einflussradius auf einen Quadranten in jede Richtung beschränkt und der Einfluss sei 50% des initialen. Wenn sich ein Quadrant unter dem Einfluss von zwei Einheiten befindet, dann können sich die Effekte verstärken (zum Beispiel durch Addition, wenn die Einheiten befreundet sind) oder im entgegengesetzten Fall vermindern beziehungsweise aufheben. In unserem konstruierten Beispiel addieren sich die Werte sich überlagernder Einflüsse. Die folgende Abbildung zeigt die für die Spielkarte berechneten Einflusswerte. Je kräftiger eine Farbe ist, desto größer der Einfluss der entsprechenden Macht. Der Einfluss der eigenen Macht ist grün dargestellt, der Einfluss der feindlichen Macht rot.





Die berechnete Influence Map kann nun zur Entscheidungsfindung hinzugezogen werden. Truppenbewegungen im grünen Bereich gelten als sicher, da der Einfluss der eigenen Einheiten dort größer ist als der der gegnerischen Einheit. Der Einfluss der eigenen Einheiten ist in der Mitte der Karte sogar so groß, dass sie durch die Kombination der Stärken in der Lage sind, sich in das direkte Einflussgebiet des Gegners zu bewegen.

3 Erstes Projekt: Poker

Im ersten Semester der Projektphase sollte die Projektgruppe einen Bot für ein rundenbasiertes Strategiespiel entwickeln. Da wir uns ganz auf die Entwicklung einer künstlichen Intelligenz konzentrieren wollten, suchten wir ein Spiel, das nach Möglichkeit schon vollständig und möglichst fehlerfrei implementiert war. Sowohl eine möglichst freie Lizenz, als auch ein gut dokumentierter Quellcode würden dabei die Entwicklung stark vereinfachen, damit das Hauptaugenmerk auf die wesentlichen Ziele gerichtet werden konnte.

Eine weitere Anforderung an das Spiel war eine möglichst große Popularität. Dies schien uns notwendig, da wir unser Ergebnis von möglichst vielen Probanden auf das Erreichen unserer Ziele überprüfen lassen wollten. Aus gleichem Grund sollten die Regeln möglichst leicht sein und das Spiel nicht zu lange dauern.

Mit diesen Anforderungen entschieden wir uns dann dazu, eine künstliche Intelligenz für die Poker-Variante Texas Hold'em zu entwickeln. Unter den für diese Variante äußerst zahlreich zur Verfügung stehenden Implementierungen wählten wir PokerTH als Grundlage für unsere Erweiterungen aus.

Der nachfolgende Abschnitt vermittelt grundlegende Informationen über die gewählte Poker-Variante, die für das weitere Verständnis unumgänglich sind.

Hold'em im Allgemeinen bezeichnet all jene Poker-Varianten, bei denen fünf offene Karten (engl. *Board cards* bzw. *Community cards*) in die Mitte des Tisches gelegt werden, die von jedem Spieler verwendet werden können, um eine Hand (die fünf besten Karten, die ein Spieler nutzen kann) zu bilden, während jeder Spieler seine Hand-Karten (engl. *Hole cards*) verwenden darf. Bei den verwendeten Karten handelt es sich um französische bzw. anglo-amerikanische Karten zu 52 Blatt. Es können zwei bis maximal vierzehn Personen an einem Spiel teilnehmen. Ziel ist es, die höchste Poker-Kombination zu erhalten oder durch geschickte Spielweise die anderen Spieler zur Aufgabe zu bewegen.

Die Rangfolge der einzelnen Kartenkombinationen ist wie folgt festgelegt, beginnend mit dem besten Blatt:

1. **Royal Flush** (z.B.: 10♣, B♣, D♣, K♣, A♣): Zehn bis Ass der gleichen Farbe.
2. **Straight Flush** (z.B.: 2♡, 3♡, 4♡, 5♡, 6♡): Fünf aufeinander folgende Karten in der gleichen Farbe.
3. **Four of a Kind** (z.B.: A♣, A♠, A♡, A♢): Vier Karten mit dem gleichen Zahlen- oder Bildwert.
4. **Full House** (z.B.: 10♠, 10♢, A♡, A♠, A♣): Drei Karten des gleichen Zahlen- oder Bildwertes und zwei andere Karten mit dem gleichen Zahlen- oder Bildwert.
5. **Flush** (z.B.: 3♠, 5♠, 9♠, 10♠, Q♠): Fünf Karten der gleichen Farbe.
6. **Straight** (z.B.: 4♡, 5♠, 6♡, 7♣, 8♢): Fünf aufeinander folgende Karten, egal in welcher Farbe.
7. **Three of a Kind** (z.B.: 7♢, 7♣, 7♠): Drei Karten mit dem gleichen Zahlen- oder Bildwert.
8. **Two Pair** (z.B.: 7♢, 7♣, A♢, A♡): Zweimal zwei Karten mit dem gleichen Zahlen- oder Bildwert.
9. **One Pair** (z.B.: 7♢, 7♣): Zwei Karten mit dem gleichen Zahlen- oder Bildwert.
10. **High Card** (z.B. A♠): Einfach eine hohe Karte, sonst nichts.

Zu Beginn muss ein sogenannter Kartengeber (engl. *dealer*) bestimmt werden. Hierzu wird der Spieler mit der höchsten Karte als *dealer* bestimmt und erhält eine Markierung zur Kenntlichmachung (engl. *dealer button*). Die Rolle des Kartengebers wechselt immer nach jedem einzelnen Spiel im Uhrzeigersinn. Der Spieler zur Linken des Gebers muss einen vorgeschriebenen Einsatz (engl. *small blind*), zum Beispiel zehn Geldeinheiten, setzen, sein Nachbar einen erhöhten *small blind*, den sogenannten *big blind* (meistens der doppelte *small blind*). Die übrigen Spieler müssen vor Erhalt der ersten Karten keinen Grundeinsatz leisten. Das Spiel verläuft in maximal vier Wettrunden. Jeder Teilnehmer erhält eine Starthand bestehend aus zwei verdeckten Karten (engl. *hole cards*). Nun müssen die Spieler im Uhrzeigersinn Ihre Einsätze machen. Dabei ist es möglich, den momentan höchsten Einsatz am Tisch mitzugehen (engl. *call*), den aktuell höchsten Einsatz der Runde weiter zu erhöhen (engl. *raise*), was als *bet* bezeichnet wird, wenn es sich um die erste Erhöhung der Runde handelt, oder seine beiden *hole cards* zu verwerfen (engl. *fold*). Wenn kein Spieler mehr erhöhen will und jeder den gleichen Einsatz erbracht hat, gilt die erste Wettrunde als beendet (engl. *flop*). Nachdem der Kartengeber drei Gemeinschaftskarten aufgedeckt hat, folgt die zweite Wettrunde (engl. *turn*). Diese Runde verläuft analog zur vorherigen, nur dass am Ende der Runde nicht drei Karten in der Mitte des Spielfeldes aufgedeckt werden, sondern eine. Nach einer weiteren Wettrunde (engl. *river*) werden die Gemeinschaftskarten komplettiert und sind dann fünf. Es folgt eine abschließende Wettrunde. Alle Spieler, die in den vorherigen Runden ihr Blatt nicht verworfen haben, können (wenn sie der Meinung sind, mit den Karten den Pot zu erlangen) nach dieser Runde ihre beiden verdeckten Handkarten den anderen Spielern zeigen, um den endgültigen Sieger dieser Runde zu bestimmen. Derjenige mit dem besten Blatt gewinnt alle Einsätze, die in dieser Runde gemacht wurden.

3.1 Ziele des Projektes

Grundsätzlich bestand für unsere beiden Projekte die Aufgabe, einen möglichst menschenähnlichen Computerspieler zu entwickeln. So stellten wir uns der Frage, wo man bei einem Poker-Bot ansetzen könnte, um Menschenähnlichkeit zu erreichen. Bei Poker fiel uns auf, dass Gefühle eine große Rolle spielen.

Es ist die Aufgabe des Spielers, die eigenen Gefühle möglichst gut kontrollieren zu können. Bekommt er ein gutes Blatt, so will er nach Möglichkeit nicht, dass man es ihm an seiner Mimik oder Gestik ansieht. Auch den Erhalt eines schlechten Blattes möchte er mit einen Bluff verbergen.

Ebenso ist es für den Spieler wesentlich, die Gefühle seiner Mitspieler einzuschätzen. Nur wenn ein Spieler seine Gegner einschätzen kann, dann weiß er, ob die Stärke ihrer Karten ihre Spielweise rechtfertigt. Diese beiden Aspekte wurden in den uns bekannten Pokerimplementierungen gar nicht oder nur sehr schwach umgesetzt.

Deswegen konzentrierten wir uns darauf, diese beiden Aufgaben des Spielers in möglichst menschenähnlicher Art und Weise in einem NPC zu realisieren.

Alle Spieler sollten in der grafischen Oberfläche durch Bilder repräsentiert werden. Diese Bilder würden einen Eindruck des emotionalen Zustands des Spielers vermitteln. Menschliche Spieler könnten dann aus einem kategorisierten Vorrat ein Bilder auswählen. So war geplant, dass es zum Beispiel sowohl eine Kategorie Wut/Hass, als auch eine für Freude geben würde. Es war klar, dass es damit dem menschlichen Spieler leicht gemacht wird, Gefühle vorzutäuschen. Da wir allerdings in erster Linie darauf abzielten den Spielspaß durch menschenähnliche Gegner zu erhöhen, wurde dieses Manko nicht weiter berücksichtigt.

Für die NPCs war geplant, dass sie ein Emotionsmodell besitzen sollten. Dieses hatte nicht nur die Aufgabe, den aktuellen Gemütszustand zu modellieren, sondern auch dessen Veränderungen im Abhängigkeit vom Spielgeschehen. Das Emotionsmodell sollte unterschiedliche Charaktere repräsentieren können.

Damit eine Kommunikation zwischen den Spielern stattfinden konnte, wurde eine Möglichkeit implementiert, Textbausteine verschicken zu können. Freitext wurde aufgrund der Komplexität der menschlichen Sprache (vgl. [Wei66]) als zu aufwändig betrachtet. Die Textbausteine wurden, wie auch die oben erwähnten Bilder, kategorisiert. Somit sollte dann das Spielgeschehen durch pokertypische Kommunikation ergänzt werden, damit der Spieler den Eindruck hätte, seine Gegner wären Menschen. Es gab auch eine Seminararbeit, die sich mit dem Thema der Kommunikation in Computerspielen beschäftigte (siehe 1.2.2).

3.2 Planung des Projektes

Maßgeblich für unsere Zeiteinteilung waren die „Milestones“, die wir uns zu Beginn der Projektgruppe gesetzt hatten. Abbildung 8 zeigt unsere angestrebten Milestones. Unser Zeitplan sah vor, dass zunächst in einer sogenannten Planungsphase die Ziele

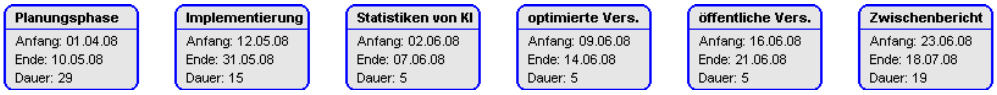


Abbildung 8: Die Projektplanung zu PokerTH. Die Grafik zeigt unseren ersten Entwurf eines Zeitplans des Projekts

der einzelnen Teilnehmer genauer definiert werden und anhand dieser sich bestimmte Aufgabenstellungen ergeben sollten. Differenziertere Konzepte sollten in Untergruppen erarbeitet und immer weiter verfeinert werden. Anschließend mussten die entwickelten Konzepte umgesetzt werden. Dies sollte in der Implementierungsphase stattfinden. Nachdem eine erste lauffähige Version implementiert wurde, soll der nächste Abschnitt, den wir unter der Überschrift „Statistiken für KI“ zusammengefasst haben, eine Feldstudie mit ca. 50 Probanden beinhalten, wobei die gesammelten Daten aufgezeichnet werden sollten. Nach sorgfältiger Validierung sollte die aktuelle Version soweit adaptiert werden, dass eine optimierte Version entsteht, die vorherige Fehler behebt und suboptimale Algorithmen verbessert. Diese optimierte Version sollte nach einem erneuten Test zu einer endgültigen Version führen, die der Öffentlichkeit zugänglich gemacht wird. Abschließend sollte ein Zwischenbericht unsere Arbeit des Semesters zusammenfassend wiedergeben.

3.3 Verwirklichung des Projektes

Der folgende Abschnitt vermittelt dem Leser, welche Inhalte im Rahmen des Poker-Projektes im ersten Semester der Projektgruppe umgesetzt wurden. Die Gliederung erfolgte anhand der vier zentralen Punkte für die Verwirklichung des Projektes.

3.3.1 Vermenschlichung: Ansatz für einen menschenähnlichen Computerspieler

Um das Kernziel der Projektgruppe, das Erzeugen eines menschenähnlichen Computerspielers, zu verwirklichen, sollten Emotionen ein essenzieller Bestandteil des Computerspielers werden. Diese Emotionen sollten sich, wie bei einem Menschen, durch

Ereignisse während des Pokerspieles verändern und sich im Verhalten, insbesondere in der Spielweise, widerspiegeln. Darüber hinaus war es ein Ziel, dass der Computerspieler einen individuellen Charakter erhält, welcher ebenfalls an seinem Spielverhalten erkannt werden können sollte.

Hierfür wurden diverse Charakterwerte modelliert, die eine charakteristische Prägung des Computerspielers ermöglichen. Um einen möglichst menschlichen Pokerspieler zu gewährleisten, wurden die Charakterwerte in einer Art beschrieben, die direkt auf die Anforderungen eines Pokerspiels abzielen.

Zusätzlich wurden einige Emotionen implementiert, um kurzfristig stattfindende Wutausbrüche oder Nervosität modellieren zu können. Die Emotionen wurden hierfür in Gruppen zusammengefasst, mit dem Ziel, ein möglichst klar erkennbares Emotionsmodell vorweisen zu können. Die Initialisierung wird durch den darzustellenden Charakter, anhand der gegebenen Charakterwerte, definiert. Die dynamische Veränderung während des Spiels wird in einer deterministischen Weise ausgeführt. Sie erfolgt bei Aktivität anderer Spieler, neutralen Ereignissen, wie dem Aufdecken von neuen Gemeinschaftskarten, und besonders intensiv bei eigenem Handeln. Näheres zu Charakteren und Emotionen findet sich im Zwischenbericht der Projektgruppe [AGG+09] in Kapitel 3.5.

3.3.2 Emoticons: Neue Kommunikationswege werden geschaffen

Ein weiteres Ziel von uns war es, einen neuen Kommunikationskanal zu konstruieren. Über diesen sollten insbesondere die Emotionen signalisiert und dargestellt werden können, durch sogenannte Emoticons. Die Emoticons werden auf der grafischen Benutzeroberfläche angezeigt.

Emoticons sind - den aktuellen emotionalen Zustand darstellende - leicht verständliche Sinnbilder mit großer Ausdruckskraft.

Zusätzlich zu den auswählbaren Emoticons sollten auch Textnachrichten verschickt werden können. Aufgrund der Schwierigkeit des Turing-Tests war sofort klar, dass keine natürliche Sprache benutzt werden sollte. In der fertigen Implementierung ist es daher nur möglich, vorgefertigte Nachrichten zu versenden und zu erhalten.

Abbildung 9 zeigt einen Spielausschnitt mit den von den Bots kommunizierten Textbausteinen im Chat-Fenster (1) und den Emoticons (2).

3.3.3 CI-Methoden: Der Kern des Spielens

Im Gegensatz zu den rein mathematischen Methoden des PokerTH-Bots, war es unser Ziel, mit Hilfe von CI-Methoden einen dynamischen Computergegner zu entwerfen.

Um dies zu gewährleisten und einen Aufbau zu wählen der Lernfähig ist, haben wir uns entschieden, die CI-Methode des neuronales Netzes (siehe 2.1 auf S.4) für das Pokerspiel zu benutzen. Bei dem Gewählten handelte es sich um ein dreischichtiges Feedforward-Netz, wie im Zwischenbericht [AGG+09] in Kapitel 3.6 beschrieben, dessen Komplexität ausreichend erschien, um die geforderten Zusammenhänge zu erlernen. Der Ausgabevektor des Netzes ergibt reelwertige Zahlen für die pokerspezifischen Anweisungen, Geld auf sein Blatt zu setzen (bet und raise), mit anderen Geboten mitzugehen (chall und check) oder sein Blatt zu verwerfen (fold), um das Spielgeschehen zu kontrollieren. Es wird diejenige Aktion gewählt, welche die höchstwertigste Zahl erhält. Im Falle des Bietens wird durch die Zahl auch die Höhe des Wetteinsatzes bestimmt. Die Textausgabe und Darstellung der Emoticons wurde separat durch ein Regelsystem kontrolliert.

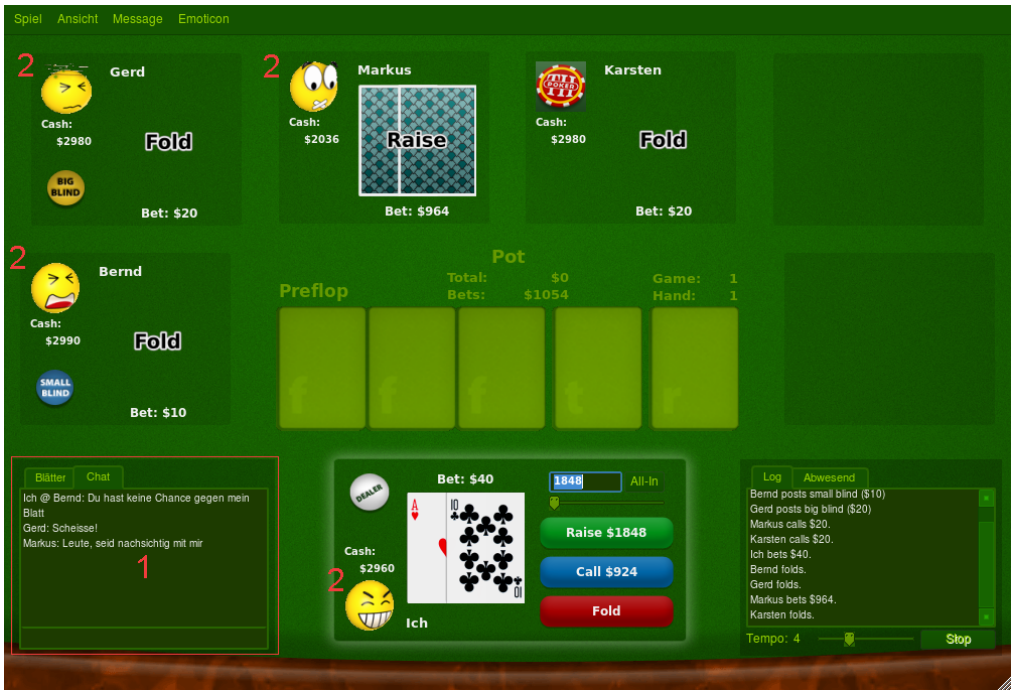


Abbildung 9: PokerTH PG529 Edition (mit Emoticons und Textbausteinen)

Das Anlernen - beziehungsweise die Optimierung - der computergesteuerten Pokerspieler sollte durch die Anpassung an zuvor erspielte Trainingsmuster realer Spieler erfolgen. Ein evolutionärer Algorithmus (siehe 2.2 auf S.6) optimierte die Kantengewichte des Feedforward-Netzes in einem offline-Lernprozess dahingehend, das gewünschte Verhalten (bet, raise, fold) in den diversen Spielsituationen dem Verhalten des menschlichen Spielers anzupassen. Wenn der Mensch in einer Situation gefoldet hatte, so sollte der computergesteuerte Pokerspieler in der gleichen Situation ebenfalls so handeln. Das primäre Optimierungsziel des von uns erstellten Spielers war es also nicht - wie bei vielen anderen künstlichen Spielern - die Profitmaximierung zu gewährleisten, sondern die Menschenähnlichkeit zu erhöhen.

3.3.4 Gegnermodellierung: Ein aufmerksames Gedächtnis

Zur Verbesserung von Spiel und Menschlichkeit, erschien es ratsam, eine Einschätzung der Gegenspieler zu liefern, sowohl was deren Charakter bzw. Emotionen, als auch was ihre Spielweise und Strategie betrifft.

Basierend auf den Einschätzungen, die aus den Handlungen der Gegner während des Spiels ermittelt werden, sollte nun für die künstliche Intelligenz ein Gedächtnis aufgebaut werden, um auf das Verhalten der Gegenspieler sowohl über kurze, als auch über längere Zeiträume reagieren zu können. Eine Funktion zum Speichern der ermittelten Werte in eine Datei sollte dies auch über einzelne Spiele hinaus ermöglichen.

Das Gedächtnis, siehe Zwischenbericht [AGG+09] Kapitel 3.7, funktionierte dabei in mehreren Ebenen, um auf alle Bereiche des Pokerspiels einzugehen. Es basiert auf den drei zeitlichen Einteilungen der strategischen, taktischen und operativen Sicht. Während die strategische Sicht die eher langfristige Einschätzung des Charakters bietet, erlaubt die taktische Komponente mögliche mittelfristige Taktiken des Spielers zu erkennen und die Anpassung des Spielers an den Tisch zu berücksichtigen. Der operative

Einblick gewinnt einen Eindruck über die Güte der aktuellen Hand des Spielers und erreicht somit eine recht kurzfristige Interessenseinschätzung. Neben der Modellierung des Spielercharakters, kann nun auch geeignet auf dynamische Spielverhaltensänderungen reagiert werden.

Der computergesteuerte Pokerspieler war somit in der Lage Verhalten von PG-Mitgliedern zu imitieren, als auch, Spieler einzuschätzen und dieses Wissen in sein Spiel einfließen zu lassen.

3.4 Erkenntnisse aus der Projektdurchführung

Da es sich für die meisten Mitglieder um das erste größere Projekt gehandelt hat, hat die Gruppe vielerlei Erfahrung sammeln können. Hierbei wurden sowohl technische als auch soziale Erfahrungen gemacht:

- Das Ticketsystem Trac[1] wurde erfolgreich eingesetzt. Außerdem betrieb die Gruppe für das Projekt ein eigenes Wiki und das Versionsverwaltungssystem Subversion[2].
- Die Programmiersprache C++ wurde erlernt beziehungsweise bei den Teilnehmern mit Vorkenntnissen vertieft.
- Es wurden CI-Methoden der evolutionären Algorithmen und neuronalen Netze implementiert und in komplexen Zusammenhängen eingesetzt.
- Es wurde die Zusammenarbeit im Team geschult. Ein großes Projekt wurde in mehrere Teilprojekte aufgeteilt und die Teilgruppen mussten wiederum Programmieraufgaben an die einzelnen Mitglieder verteilen.
- In den Sitzungen mussten die Ergebnisse präsentiert werden. Dies erforderte eine gezielte Erarbeitung der Teamleistung, um den benötigten Informationsaustausch zwischen den Gruppen zu gewährleisten. Die Fähigkeit zum freien Vortrag und zur Diskussion wurden in den Sitzungen trainiert.
- Durch die regelmäßige Rolle als Moderator bzw. Protokollant mussten die PG-Mitglieder lernen, Diskussionen zielführend zu lenken und durch Trennung von wichtigen und weniger wichtigen Aspekten zu differenzieren. Der Protokollant musste lernen, während der Sitzung die wesentlichen Äußerungen, Fragen und Abstimmungen so zu dokumentieren, dass in der folgenden Sitzung nahtlos an die vorausgegangene angeschlossen werden konnte.

Durch die Erfahrungen, welche im ersten Projekt gemacht wurden, gelang es der Gruppe, den Verwaltungsaufwand für das zweite Projekt zu reduzieren, so dass man sich verstärkt auf wichtige Entwicklungsdetails konzentrieren konnte.

[1] `http://trac.edgewall.org/`
[2] `http://subversion.tigris.org/`

4 Zweites Projekt: Diplomacy

> *Anfang des 20. Jahrhunderts war Europa ein Hexenkessel politischer Intrigen. Sie stehen am Anfang einer Reise in diese historisch höchst interessante Zeit. Ihre Aufgabe besteht darin, den Verlauf der Geschichte zum Vorteil Ihrer Nation zu verändern...*[1]

So beginnt die Spielanleitung des im Jahre 1959 von Allan B. Callhamer entwickelten Brettspiels Diplomacy. Hier entscheidet nicht nur das strategische Geschick über Sieg oder Niederlage, sondern auch das Aushandeln, Aufrechterhalten und Brechen von diplomatischen Beziehungen und Bündnissen zwischen den Spielern.

Callhamer schloss die Entwicklung des Spiels größtenteils im Jahre 1954 ab, hatte dann aber große Probleme, eine Firma für die Produktion zu finden. Daraufhin investierte er im Jahre 1959 selber in die Produktion von 500 Exemplaren und übertrug ein Jahr später die Produktion an die Firma Games Research, die im darauf folgenden Jahr das Spiel veröffentlichten (vgl. [Cal74]). Im Jahre 2009, als dieser Bericht erstellt wurde, besaß Hasbro die Rechte an dem Spiel.

Diplomacy simuliert den ersten Weltkrieg in Europa im 20. Jahrhundert. Zu dieser Zeit herrschte Kaiser Wilhelm II über das Deutsche Reich (im weiteren Verlauf Deutschland), König Eduard VII über das Vereinigte Königreich (im weiteren Verlauf England) und Präsident Émile Loubet über Frankreich. Italien wurde von König Viktor Emanuel III regiert, Österreich-Ungarn von Kaiser Joseph I, das Osmanische Reich (im weiteren Verlauf Türkei) von Sultan Abdülhamid II und Herrscher im Russischen Reich (im weiteren Verlauf Russland) war Zar Nikolaus II. Der Spieler schlüpft in die Rolle eines der sieben Herrscher.

Das Spielfeld ist eine Landkarte Europas mit politischer Einteilung der Gebiete. Die 75 Spielfelder sind unterteilt in 56 Land- und 19 Seegebiete, die jeweils durch eindeutige Namen gekennzeichnet sind. 34 der Landgebiete sind Versorgungszentren, die eine besondere Rolle spielen. Eine Einheit auf dem Spielfeld ist entweder eine Armee oder eine Flotte, und auf jedem Feld darf sich immer nur eine Einheit befinden. Im Folgenden werden die geläufigen Abkürzungen für Gebiete und Großmächte, wie im Anhang A.2 auf Seite 124 beschrieben, verwendet.

Das Spiel beginnt im Jahr 1901. Jeder Spieler hat zu diesem Zeitpunkt eine festgelegte Anzahl von Armeen und Flotten auf vorher definierten Gebieten stationiert. Bis auf Russland, das zu Beginn über vier Einheiten verfügt, beginnen alle anderen Spieler mit drei Einheiten. In der Standard-Variante ist die Aufstellung wie folgt:

AUS Flotte: TRI, Armeen: BUD und VIE

ENG Flotten: EDI und LON, Armee: LVP

FRA Flotte: BRE, Armeen: MAR und PAR

GER Flotte: KIE, Armeen: BER und MUN

ITA Flotte: NAP, Armeen: ROM und VEN

RUS Armeen: MOS und WAR, Flotten: STP und SEV

TUR Armeen: CON und SMY, Flotte: ANK

Die Spielfelder, auf denen die Einheiten zu Beginn des Spiels stationiert sind, sind die Heim-Versorgungszentren, die noch eine wichtigere Rolle spielen als alle anderen Versorgungszentren, da nur hier neue Einheiten gebaut werden können.

[1] `http://www.hasbro.de/manuals/41307.pdf`

Diplomacy ist ein rundenbasiertes Brettspiel, bei dem alle Spieler zuerst ihre Züge aufschreiben und zu einem festgesetzten Zeitpunkt gleichzeitig ausführen. Das Spielgeschehen ist in Jahre unterteilt, und jedes Jahr besteht aus fünf Phasen. Es gibt zwei Phasen, in denen Einheiten bewegt werden dürfen. Auf jede dieser Phasen folgt eine Rückzug-Phase und am Ende jedes Jahres eine Bau-Phase.

In der Bewegungs-Phase plant jeder Spieler, welche Einheiten auf welche Provinz gezogen werden sollen. Auch darf in dieser Phase mit Gegenspielern verhandelt werden. So kann ein Zug durch einen Gegenspieler unterstützt werden, oder einem Gegenspieler kann bei einer Aktion geholfen werden. Die Spieler versuchen durch geschickte Verhandlung, eine Allianz oder einen Friedenspakt zu bilden, um gegen andere Spieler zu bestehen. Die genauen Regeln für Diplomacy sind unter 4.1 auf Seite 19 zusammengefasst. Wichtigster Punkt, der hier vorweg genommen werden soll, ist die Tatsache, dass alle Einheiten gleich stark sind. Eine Einheit kann also nur geschlagen werden, wenn sie von mindestens zwei anderen Einheiten gleichzeitig angegriffen wird.

Wird eine Einheit erfolgreich angegriffen, so muss sie sich in der darauf folgenden Phase, der Rückzug-Phase, zurückziehen. Je nach Spielvariante sind Verhandlungen in der Rückzug-Phase entweder erlaubt oder verboten. Ist ein Rückzug nicht erfolgreich, so muss die Einheit vom Spielbrett entfernt werden. Hält eine Einheit in der zweiten Bewegungsphase eines Spiels ein Versorgungszentrum, so geht das Versorgungszentrum in den Besitz der Großmacht des Spielers über, dem die Einheit gehört.

In der Bau-Phase werden dann zum einen die Versorgungszentren und zum anderen die Einheiten jedes einzelnen Spielers aufsummiert. Kontrolliert ein Spieler weniger Einheiten als Versorgungszentren, so darf er in seine Heim-Versorgungszentren neue Einheiten stellen. Dies gilt nur, sofern sich in den Gebieten keine Einheit befindet. Auf Gebieten, die nicht Heim-Versorgungszentren sind, dürfen keine Einheiten erstellt werden.

Sobald einem Spieler 18 oder mehr Versorgungszentren gehören, wird das Spiel beendet und der Spieler zum Sieger erklärt. Es gibt neben der Standard-Variante, bei der es nur einen Sieger gibt, auch die Möglichkeit, dass mehrere Spieler, die zusammen mehr als 17 Versorgungszentren besitzen, einvernehmlich ein Unentschieden erklären. Dann werden diejenigen Spieler Sieger, die sich zusammengeschlossen haben.

Neben der Standard-Variante nach Calhamer werden noch viele andere Varianten gespielt, die sich in Kartenvarianten und Regelvarianten gruppieren lassen. Die Kartenvarianten zeichnen sich dadurch aus, dass das Spielfeld verändert wird. Bei den Regelvarianten wird mit geänderten Regeln gespielt. Motiviert sind die Variationen durch den Wunsch, das Spiel in einer anderen Zeit oder an einem anderen Ort zu spielen, wobei auch die Anzahl der Spieler variieren kann. Die Projektgruppe 529 beschäftigte sich mit der Standard-Variante.

Zum Ende des 20. Jahrhunderts wurden einige Implementierungen des Spiels für den Computer veröffentlicht. Die Schwachstelle in allen Implementierungen ist allerdings die Intelligenz der Computergegner, weshalb Verhandlungen, die wesentlicher Bestandteil von Diplomacy sind, nur unzureichend ausgeführt werden können.

Der DDB (Deutscher Diplomacy Bund[2]) ist die Anlaufstelle für passionierte Spieler und richtet seit einiger Zeit auch die Deutschen Diplomacy Meisterschaften aus. Im Jahr 2004 war er für die Ausrichtung der Diplomacy Europameisterschaft verantwortlich sowie 2006 für die Diplomacy Weltmeisterschaft.

Treffpunkt für Diplomacy Spieler im Internet ist unter anderem die Internetseite „The Diplomacy Pouch“ `http://www.diplom.org/index.py`.

[2] `http://www.diplomacy-bund.de/`

Entwickler von Diplomacy-Bots finden im Diplomacy AI Center auf `http://www.daide.org.uk/index.xml` ein Framework für die Programmierung von Clients sowie eine Vielzahl von Tools und auch fertige Bots zum Testen (vgl. 4.4 auf Seite 21).

4.1 Regeln

Diplomacy birgt für Einsteiger ein komplexes Regelwerk, das es zu verstehen gilt. Für einen einfachen Einstieg sind hier die wichtigsten Regeln vermerkt. Die vollständigen Regeln können im Anhang unter A.1 auf Seite 123 nachgelesen werden.

1. Alle Einheiten haben die gleiche Stärke.
2. In einem Gebiet kann sich immer nur eine Einheit aufhalten.
3. Wenn zwei oder mehr Einheiten gleicher Stärke versuchen, in dasselbe Gebiet zu ziehen, bleiben alle diese Einheiten in ihren Ausgangsgebieten.
4. Eine Pattsituation führt nicht dazu, dass sich eine Einheit zurückziehen muss, die sich zu Beginn des Spielzuges in dem Gebiet aufgehalten hat, in dem es zur Pattsituation gekommen ist.
5. Eine Einheit, die sich nicht bewegt oder nicht bewegen kann, kann dazu führen, dass sich mehrere Einheiten nicht bewegen.
6. Zwei Einheiten können ohne die Verwendung eines Geleitzuges nicht miteinander Plätze tauschen.
7. Drei oder mehr Einheiten können innerhalb eines Zuges ihre Plätze tauschen. Dabei dürfen aber keine zwei Einheiten direkt miteinander Plätze tauschen.
9. Wenn sich eine Einheit zu bewegen versucht, sind nur jene Unterstützungsbefehle gültig, bei denen die Bewegung der unterstützten Einheit korrekt angeführt wurde.
17. Wenn eine der Flotten, die eine Armee geleitet, zum Rückzug gezwungen wird, scheitert der ganze Geleitzug.

4.2 Grundlegende Züge

Auch wenn Diplomacy auf vielen Regeln basiert, so lassen sich die Züge in die vier Kategorien „Halten“, „Bewegen/Angreifen“, „Unterstützen“ sowie „Konvoi“ einteilen:

4.2.1 Halten

Einer Einheit, die ihre Position halten soll, gibt man den „Halte“-Befehl. Die Einheit bleibt auf ihrem Feld und verteidigt es gegebenenfalls gegen einen Angreifer. Der „Halte“-Befehl ist allerdings nicht nur für die Verteidigung von Nutzen. Hat eine Einheit ein feindliches oder freies Versorgungszentrum in der ersten Bewegungsphase eines Spieljahres erreicht, so ist ein „Halte“-Befehl solange sinnvoll, bis das Versorgungszentrum in den Besitz des Spielers übergeht (vgl. 4.3 ab Seite 20).

4.2.2 Bewegen/Angreifen

Der Befehl zum Bewegen/Angreifen veranlasst eine Einheit, ihr Feld zu verlassen und auf ein direkt anliegendes Feld zu ziehen. Befindet sich auf dem Feld keine Einheit, und auch keine andere Einheit versucht auf dieses Feld zu gehen, so ist der Befehl erfolgreich.

Das schließt auch ein Tauschen von Plätzen zwischen zwei Einheiten aus. Befindet sich bereits eine fremde Einheit auf dem Feld, die es auch nicht verlassen möchte oder kann, so wird diese Einheit angegriffen. Da alle Einheiten bei Diplomacy gleich stark sind, ist ein einfacher Angriff nie erfolgreich. Auch wenn zwei oder mehr Einheiten der Befehl zum Angriff gegeben wird, so wird jeder Befehl als einzelner Befehl angesehen und die angegriffene Einheit kann ihr Feld verteidigen. Um einen erfolgreichen Angriff durchzuführen, bedarf es einer Unterstützung, wie im nächsten Abschnitt beschrieben.

4.2.3 Unterstützen

Unterstützt werden können sowohl eigene als auch fremde Einheiten. Auch kann Unterstützung sowohl bei „Bewegen/Angreifen"-Befehlen als auch bei „Halte"-Befehlen gegeben werden. Bei einem „Halte"-Befehl muss sich die unterstützte Einheit auf einem direkt angrenzenden Feld befinden. Die Unterstützung bei einem „Bewegen/Angreifen"-Befehl setzt voraus, dass das Zielfeld direkt an dem Feld der unterstützenden Einheit liegt. Erwähnt werden sollte auch die Tatsache, dass eine Unterstützung nicht abgelehnt werden kann und auch feindliche Einheiten unterstützt werden können.

4.2.4 Konvoi

Flotten können sich nur auf Wasser oder Küstenfeldern bewegen und Armeen dürfen nur Land oder Küstenfelder nutzen. Da England als Insel von Wasserfeldern umgeben ist, Einheiten aber nur in den Heim-Versorungszentren gebaut werden dürfen, muss es eine Möglichkeit geben, Armeen auf das europäische Festland zu transportieren, damit auch England eine faire Siegeschance hat. Der „Konvoi"-Befehl realisiert dies. Ein Konvoi wird genau dann ausgeführt, wenn eine Armee von einem Küstenfeld auf ein anderes Küstenfeld zieht und auf allen dazwischen liegenden Wasserfeldern Flotten stehen, die die Armee geleiten.

4.3 Strategien

Genauso wie Schach enthält Diplomacy keine zufälligen Ereignisse, die das Spielgeschehen beeinflussen. Es gibt weder Würfel, noch werden vorher gemischte Karten gezogen. Man könnte also fast davon ausgehen, dass es möglich ist, aufgrund des deterministischen Spielansatzes und den auf den ersten Blick eingeschränkten Zugmöglichkeiten, einen Spielbaum zu berechnen. Nach [Loe95a] existieren allerdings allein 4.430.690.040.914.420 verschiedene Eröffnungszüge, weshalb der Ansatz des Spielbaums aufgrund der Komplexität des Problems nicht weiter verfolgt werden kann. Maschinell wie auch beim menschlichen Vorgehen kann also keine optimale Lösung in annehmbarer Zeit gefunden werden. Menschen entscheiden eher intuitiv, welche Züge gemacht werden sollten und werden in ihren Entscheidungen durch Emotionen beeinflusst. Hierbei gibt es verschiedenste Theorien, die sowohl das strategische, als auch das soziale Handeln der Spieler versuchen zu erklären. Paul D. Windsor beschreibt in [Win99] vier dieser Spielertyen und ihr Verhalten (siehe dazu 5.2.1.1.1 auf Seite 42). Ein Computerprogramm, das versuchen will, wie ein Mensch zu spielen, muss vergleichbare Ansätze verfolgen, um nicht enttarnt zu werden. Gleichwohl sollte aber beachtet werden, dass eine menschliche Strategie nicht zwangsläufig zum Sieg des Spielers führen muss, da es Charaktere gibt, denen der Spaß am Spiel wichtiger ist, als später als Sieger hervor zugehen. Gerade bei Diplomacy, wo der Verhandlungsaspekt eine zentrale Rolle spielt, kann eine vorausschauende Kommunikation mit den Mitspielern eine größere Befriedigung für den Spieler sein als der spätere Sieg.

4.4 Vorhandener Rahmen

Wie in unserem ersten Projekt wollten wir uns bei der Entwicklung eines Diplomacy-Bots auf das Wesentliche konzentrieren. Dies bedeutete für uns, dass wir uns nicht mit Themen wie dem Entwurf einer Client/Server-Anwendung oder der Art und Weise wie Bots miteinander kommunizieren beschäftigen wollten. Die notwendigen Routinen für die Überprüfung und Auflösungen von Zugwünschen sollten bereits implementiert sein. Daraus folgend wollten wir den Fokus wieder ganz auf die Entwicklung der künstlichen Intelligenz legen.

Hier erwies sich das DAIDE-Projekt[3](Diplomacy AI Development Environment) als eine sinnvolle Grundlage für unsere Arbeit.

So war für uns gleich am Anfang klar, dass eine wesentliche Herausforderung sein würde, wie man das Thema der Verhandlungen angehen sollte. Verständlich war, dass die Verwendung von freier Texteingabe ein unerreichbares Ziel darstellen würde. Einfache Textbausteine, wie wir sie beim Pokerprojekt genutzt hatten, wären allerdings auch keine Lösung des Problems.

Das DAIDE-Projekt hat hierzu ein Kommunikationsprotokoll definiert, das im folgenden Abschnitt näher beschrieben wird.

4.4.1 Kommunikationsprotokoll

In diesem Protokoll werden zwei Bereiche definiert, die ein Entwickler eines Diplomacy-Bots beherrschen muss. Einmal muss sich der Bot mit dem Spiel verbinden können, sich also Informationen über die Karte und den aktuellen Spielstand holen. Zum anderen muss er auch Möglichkeiten haben, mit den anderen Mächten Verhandlungen zu führen und seine Züge zu platzieren. Für uns ist gerade der letztere Punkt von besonderem Interesse gewesen, da hier genau die Fragen und Probleme behandelt wurden, mit denen wir uns Eingangs beschäftigt hatten. Konkreter wurde hier definiert, welche Vokabeln und Ausdrücke möglich sind. Dadurch sollte die Kommunikation nicht zu komplex werden. Es würden sich aber auch keine zu starken Einschränkungen für das Spielgeschehen ergeben.

Das Kommunikationsprotokoll definiert eine Vielzahl an Kommandos und teilt diese in 14 Sprachstufen (auch Press-Level genannt) ein. Die unterste Stufe definiert grundlegende Kommunikation, wie den Verbindungsaufbau zum Server oder die Abgabe von Zügen, gibt allerdings keine Chancen, Verhandlungen aufzubauen. Danach kommen mit jeder Stufe weitere Kommandos hinzu. Je niedriger die Stufe ist, desto wichtiger sind die zur Verfügung stehenden Kommandos.

Die letzte Stufe (8000) besteht aus freier Texteingabe. Zwischen Stufe 8000 und der oben beschriebenen Stufe 0 wird wie folgt differenziert:

Stufe 10 Friedenspakt und Allianzen

Stufe 20 Zugvorschläge unterbreiten

Stufe 30 mehrteilige Angebote unterbreiten

Stufe 40 Handel mit Support-Centern betreiben

Stufe 50 verschachtelte mehrteilige Angebote unterbreiten

Stufe 60 Anfragen und Insistenz

Stufe 70 Anfragen für (Zug-)Vorschläge

[3]http://www.daide.org.uk/index.xml

Stufe 80 Beschuldigungen

Stufe 90 Diskussionen über zukünftige Runden

Stufe 100 Bedingungen

Stufe 110 Marionetten und Begünstigungen

Stufe 120 Weiterleiten von Nachrichten

Stufe 130 Erläuterungen

Damit mussten wir uns nur noch auf eine Stufe einigen, die maßgeblich für die Entwicklung der Bots sein sollte. Wir schauten uns bestehende Implementierungen von Bots an, um einen Hinweis darauf zu bekommen, welches Press-Level realistisch wäre. Ein großer Teil der Bots arbeitet auf Stufe 0, wenige auf Stufe 10. Einen Bot, in dem mehr als Stufe 40 implementiert war, suchten wir vergeblich. Aus diesem Grund entschieden wir uns für ein Presslevel zwischen 20 und 40. Was davon dann später wirklich umgesetzt wurde, berichten wir in den Abschnitten der einzelnen Bots.

4.4.2 Diplomacy-Server

Das DAIDE-Projekt bietet auch einen offiziellen Spiele-Server an, der das oben beschriebene Kommunikationsprotokoll implementiert. Dieser bietet viele Konfigurationsmöglichkeiten. Insbesondere wird die Möglichkeit, das maximale Sprachlevel für ein Spiel anzugeben und für die einzelnen Spielphasen (Zug, Rückzug und Bauphase) zeitliche Schranken anzugeben, unterstützt.

Den Server gibt es nur als Windows-Binary, was bei uns zu einigen Problemen führte. Viele Teilnehmer der Projektgruppe konnten aus verschiedenen Gründen nicht unter Windows arbeiten. Zwar konnte man den Server auch mit Hilfe des Wine-Projekts[4] starten, doch dies führte nicht immer zu einem stabilen Betrieb. Aus diesem Grund wurde nach einer Alternative gesucht, die auch direkt unter Linux lauffähig war.

Fündig wurden wir bei dem Parlance-Projekt[5]. Dieser Server wurde in Python geschrieben und ist so größtenteils plattformunabhängig. Die Dokumentation ließ ihn als einen Server mit vielen Konfigurationsmöglichkeiten erscheinen. Leider stellte sich in der Praxis bald heraus, dass viele dieser Optionen in Wirklichkeit nicht implementiert waren und er auch in vielerlei Hinsicht nicht stabil lief.

4.4.3 DAIDE-Mapper

Um ein laufendes Spiel zu betrachten, ist es möglich, sich mit Hilfe der DAIDE-Syntax als Beobachter mit dem Server zu verbinden und dann genügend Informationen zu bekommen, um z.B. die aktuelle Karte zeichnen zu können. Mann kann aber auch auf ein für diesen Fall erstelltes Programm (DAIDE-Mapper) zurückgreifen. Dieses bietet neben der oben genannten Möglichkeit - ein laufendes Spiel zu beobachten - auch die Option, als Schnittstelle zwischen dem Server und einem menschlichen Spieler zu fungieren. In dieser Betriebsart zeichnet der Mapper die Karte und ermöglicht durch seine GUI (engl. Graphical User Interface, grafische Benutzeroberfläche), Züge einzugeben. Ebenfalls bietet das grafisches Interface das Versenden von Nachrichten in DAIDE-Syntax an den Server an. Diese Fähigkeit wurde von uns später beim Turing-Test genutzt.

[4] http://www.winehq.org/
[5] http://pypi.python.org/pypi/Parlance/1.4.0

4.4.4 DAIDE Java AI Communication API

Nachdem einige Teilnehmer im ersten Projekt mit C++ einige Probleme hatten, entschieden wir uns im zweiten Projekt Java als Entwicklungplattform zu verwenden. Da es für C++ Bibliotheken vom DAIDE-Projekt gab, suchten wir etwas Vergleichbares für Java. Vom DAIDE-Projekt aus verlinkt war eine Klassen-Bibliothek von Henrik Bylund. Mit dessen Hilfe konnte man auf der Ebene der DAIDE-Syntax Kontakt mit einem DAIDE-Server halten. Auch wenn diese Unterstützung nicht so weit ging, wie sie von den C++-Bibliotheken bereitgestellt wurde, so wurde dies durch die im Team erheblich größere Erfahrung mit der Java-Plattform mehr als ausgeglichen.

4.5 Existierende NPCs

Es gibt eine Vielzahl existierender NPC für *Diplomacy*. Einige sind in der Lage, zu kommunizieren, aber die meisten sind nur darauf bedacht, taktische Züge der Einheiten zu berechnen und auszuführen. Hierfür werden eine Reihe verschiedenster Ansätze geboten, welche in den folgenden Unterabschnitten erläutert werden. Die meisten sind mit dem *DAIDE*-Server einsatzfähig und verwenden die *DAIDE*-Syntax. Im Folgenden sollen wichtige und interessante Ansätze bei Bots vorgestellt werden.

4.5.1 DumbBot

Der DumbBot ist ein von David Norman entwickelter Bot, welcher in der Lage ist, Spielanfänger zu schlagen. Er beherrscht keine Verhandlungen, ist aber in der Lage, in Spielsituationen auf Basis diverser Faktoren strategische Entscheidungen zu treffen. Diese Entscheidungsfindung setzt sich zusammen aus einer Landbewertungsphase und einer Zugauswahlphase.

Landbewertungsphase In dieser Phase wird jedem Land der Karte ein Wert zugeordnet. Dieser Wert umschließt Faktoren wie z. B. den Besitzer des Landes und dessen Stärke, in wie vielen angrenzenden Ländern feindliche und eigene Armeen stehen, sowie eine errechnete Wahrscheinlichkeit, wie hoch die Chance auf einen erfolgreichen Angriff ist. Die Phase umschließt außerdem, dass sich die Werte adjazenter Knoten gegenseitig beeinflussen, um eine globalere Sicht auf die Karte zu erhalten.

Zugauswahlphase Sobald für jedes Land der Wert berechnet wurde, beginnt die Zugauswahlphase. In dieser Phase wird für jede Einheit aus der Menge der angrenzenden Länder jenes ausgewählt, welches den höchsten berechneten Wert hat. Anschließend wird der Befehl erteilt, die jeweilige Einheit in das Zielgebiet zu bewegen und überprüft, ob schon eine andere eigene Einheit in dieses Land ziehen möchte. Wenn sich noch keine andere Einheit in das Zielgebiet bewegt, dann wird der Zugbefehl vergeben, ansonsten wird die Einheit unterstützt. Die Möglichkeit des Konvois wird vom DumbBot nicht unterstützt.

Das größte Problem des DumbBots ist es, dass er nicht in der Lage ist mit den anderen Spielern selbst einfachste Verhandlungen durchzuführen.

4.5.2 Diplomat-Bot

Der Diplomat-Bot ist - wie der Name schon erahnen lässt - der Versuch, den Schwerpunkt beim Spielen von Diplomacy auf Verhandlungen zu setzen und durch List zu gewinnen. Seine Vorgehensweise kann wie folgt beschrieben werden:

- Am Anfang führt der Diplomat-Bot seine Einheiten zu kleinen Gruppen zusammen und entscheidet, welche anliegenden Länder er erobern möchte.
- Danach entwirft er für jede Gruppe eine eigene Gruppenstrategie, um aus diesen Gruppenstrategien eine möglichst gute Gesamtstrategie zu entwickeln.
- Sobald der Diplomat-Bot mit seiner Armee ein Land erobern möchte, welches zu stark verteidigt ist, so dass er es mit seinen Armeen alleine nicht erobern kann, beginnt er, Diplomatie einzusetzen. Er wendet sich an die anderen Mitspieler und bittet sie um den Gefallen, ihn in diesem Kampf zu unterstützen. Als Gegenleistung bietet er dem Unterstützenden bei der Absprache an, ihm später bei einem Angriff zur Seite zu stehen.
- Der Diplomat-Bot schätzt in Abhängigkeit von der Spielsituation außerdem ein, wann es für ihn sinnvoll ist, sich an Absprachen zu halten und wann er Absprachen eher brechen sollte. Es kann also durchaus passieren, dass der DiplomatBot einem Spieler zwar für eine bestimmte Situation Unterstützung zusagt, er diese aber dann nicht durchführt. Sollte sich ein Spieler nicht an getroffene Abmachungen mit dem Bot halten, wirkt sich dies auf das Verhältnis zwischen Spieler und Bot aus. Um die Mitspieler einschätzen zu können, speichert der Bot nämlich für jeden Spieler Werte für Beständigkeit und Ehrlichkeit, die sich je nach Verhalten des Spielers verändern.

Man kann den Diplomat-Bot als das Gegenteil von DumbBot betrachten. Seine Stärke liegt eindeutig in der Diplomatie, jedoch hat er Schwächen in der strategischen Umsetzung, da die Analyse der Mitspieler oft sehr schwierig ist.

4.5.3 BlabBot

Der Blabbot[New08], konstruiert von John Newbury, ist ein PressLevel 20 Bot. Das bedeutet, dass er Verhandlungen mit anderen Diplomacy Spielern führen kann. Der BlabBot basiert auf dem DumbBot. Es werden einfachste Verhandlungstechniken benutzt, so wird z.B. an alle Mächte ein Friedensangebot geschickt und falls eine Macht ein Friedensangebot annimmt, wird mit einer geringeren Wahrscheinlichkeit ein Angriff auf die Provinzen der befriedeten Macht durchgeführt. Wenn alle noch im Spiel verbliebenen Mächte Frieden mit dem BlabBot geschlossenhaben, verhält sich der BlabBot genau wie zuvor, oder es wird jede Runde ein Unentschieden an alle Mächte versendet, so dass ein Remis ermöglicht werden soll.

4.5.4 The Israeli Diplomat

Bei der Entwicklung des *Israeli-Bot*, entwickelt von S. Kraus, D. Lehmann und E. Ephrati [Kra95], wurde besonders viel Wert auf Verhandlungen gelegt. Der *Israeli-Bot* verwendet eine Multi-Agenten-Architektur, das heißt, dass mehrere Agenten eingesetzt werden. Die Architektur ähnelt der einer Kriegsregierung. Die einzelnen Agenten – *Prime Minister*, *Secretary*, *Ministry of Defense*, *Foreign Office*, *Military Headquarers*, *Intelligence*, *Strategies Finder* – übernehmen verschiedene Aufgaben. Der *Prime Minister* steuert die Aktivitäten vom *Israeli-Bot* und besitzt Charakterwerte. Verhandlungen werden mit dem *Foreign Office*geführt. Das *Secretary* wird ebenfalls vom *Prime Minister* gesteuert und enthält die Regeln von Diplomacy sowie die aktuelle Spielsituation. Das *Ministry of Defense* ist verantwortlich für die Planung und Situationsanalyse und wird beeinflusst von den Persönlichkeitswerten des *Prime Ministers*. *Military Headquarters* entscheidet über die Bewegungen des *Israeli-Bots* am Ende jeder Saison. Das Modul *Intelligence* soll Informationen über die anderen Spieler einholen und entscheidet, ob den eingeholten Informationen vertraut wird. Der *Strategies Finder* bietet verschiedene

Strategien an. Der Israeli-Bot ist mit der *DAIDE*-Syntax nicht vertraut und kann mit dem *DAIDE*-Server nicht eingesetzt werden.

4.5.5 The Bordeaux Diplomat

Der *Bordeaux Diplomat*[Loe95a]wurde entwickelt von Daniel Loeb und Michael Hall. Davon ausgehend, dass Gegenspieler immer den besten Zug, der möglich ist, durchführen, versucht der Bordeaux-Bot, eine Gegenstrategie zu entwickeln. Mithin ist die Anzahl potentieller Züge bei der gegebenen Karte immer noch zu groß, daher wird das Spielfeld in kleinere Abschnitte unterteilt und somit die Anzahl möglicher Bewegungen reduziert. Zusätzlich verfolgt der Bot die Strategie, dass ausgehend von der Position seiner heimischen Versorgungszentren, ein bestimmter Einflussbereich rundum die heimischen Versorgungszentren gelegt wird, welcher sich beim auseinanderdriften der eigenen Einheiten vergrössert. Dieses Konzept mit den Einflussbereichen um die heimischen Versorgungszentren sorgt dafür, dass ein gewisser Schutz für die heimischen Versorgungszentren gewährleistet wird, weil der *Bordeaux Diplomat* sämtliche Einheiten in diesen Einflussbereichen platzieren wird.

4.5.6 LA Diplomat

LA Diplomat[WCW+08] ist ein Bot von A. Shapiro, G. Fuchs und R. Levinson, der keinerlei Verhandlungen führt (PressLevel 0). Er ist nicht mit der *DAIDE*-Syntax vertraut und kann nicht mit dem *DAIDE*-Server eingesetzt werden. Für jede Phase generiert der Bot alle möglichen Züge für jede einzelne Einheit, um anschliessend sämtliche Zugkombinationen aller Einheiten mit einem Vergleichswert – der schon zur Compilezeit initialisiert wird – zu vergleichen und daraus den Besten auszuwählen. Dieser Vergleichswert gibt an, ob ein Zug als vorteilhaft für den Bot einzuschätzen ist oder nicht. Der Vergleichswert wird zur Laufzeit angepasst und optimiert, so dass im Laufe eines Spiels sich die Spielstärke des Bots verbessern sollte.

4.5.7 Diplominator

Der Diplominator [WCW+08], entwickelt von Adam Webb, Jason Chin, Thomas Wilkins, John Pacye und Vincent Dedoyard, ist ein rudimentärer Press-Level 20 Bot, d. h., dass er in der Lage ist, Verhandlungen zu führen. Der Diplominator benutzt die DAIDE-Syntax und wurde in der Programmiersprache Java programmiert. Er basiert auf dem DumbBot. Mithin wurde er um einige Aspekte erweitert. Die wichtigsten Erweiterungen werden im folgenden zusammengefasst.

- Provinzen, die zwei Küsten haben, müssen gesondert betrachtet werden, daher wurden jeder Provinz drei Instanzen einer zusätzlichen Java-Klasse (*Node* – deutsch Knoten) hinzugefügt, die die Anzahl der Küsten gespeichert hat.
- Nach einer initialen Bewertung der einzelnen Knoten (Diese hängt davon ab, ob die Provinz ein Supply Centre ist) werden Zielwerte gesetzt (hierdurch weiss der Bot wohin die einzelnen Einheiten ziehen sollen). Diese Werte werden dahingehend modifiziert, dass durch jeden Knoten iteriert und ein neuer Zielwert berechnet wird, der sich als Summe aus dem Zielwert der letzten Iteration multipliziert mit den Zielwerten der adjazenten Knoten der letzten Iteration darstellen lässt (blurring algorithm). Formal:

$$\sum_{n=0}^{n=10} vorheriger\ Wert(n)\ \cdot\ k_n$$

- Für jede Provinz wird ein Stärkewert, der sich aus der Anzahl der Einheiten, die in diese Provinz einziehen können errechnen lässt, und ein Konkurrenzwert, welcher sich aus der maximalen Anzahl der Einheiten, die in diese Provinz einziehen können und zu einer einzelnen Macht gehören berechnen lässt, errechnet.
- Der endgültige Zielwert für jeden Knoten lässt sich formal folgendermassen darstellen:

  $$(blurring) + (Stärkewert) \cdot (Gewichtung) - (Konkurrenzwert) \cdot (Gewichtung)$$

- Die Verhandlungsmodule des Diplominators sind sehr trivial. Er implementiert jeweils ein PressLevel 10 und ein PressLevel 20 Verhandlungselement.
- Bei dem PCE-Modul wird zu Beginn jeder Verhandlungsrunde an alle Mächte ein Friedensangebot geschickt, falls diese noch keines angenommen haben. Alsdann werden die Zielwerte für diese Macht herabgesetzt, und sobald der Diplominator eine bestimmte Anzahl an Supply Centres eingenommen hat, wird den Mächte, die ein Friedensangebot angenommen haben, in den Rücken gefallen.
- Das reagieren auf XDO-Anfragen, d.h., Unterstützungsanfragen, die einen bestimmten Zug betreffen, wurde bereits vorbereitet, aber die Ausführung noch nicht umgesetzt. Dabei hat jeder Spieler ein bestimmtes Kreditlimit. Wenn dieses einen bestimmten Wert übersteigt, dann werden weiteren XDO-Anfragen nicht entgegengekommen.

4.5.8 Holdbot

Bei dem *Holdbot* handelt es sich um einen NPC, der lediglich in der Lage ist, zu *holden* (seine Postion Runde für Runde zu halten). Mithin führt er keinerlei Verhandlungen. Wenn es darum geht, einen NPC in einfachen Bedingungen zu testen, kann der Holdbot eingesetzt werden.

5 Entwickelte Diplomacy NPCs

Zur Durchführung des zweiten Projektes wurde die Gruppe in drei Untergruppen geteilt. Die Bildung der Gruppen erfolgte nach den Präferenzen, welche Untersuchungen durchgeführt werden sollten. Die Verwirklichung von drei parallelen Projekten ermöglichte einen Vergleich der Bots untereinander. Obwohl eine gewisse Konkurrenz unter den Teams herrschte, standen die Teilnehmer gruppenübergreifend für Empfehlungen bereit.

5.1 NICE

Dieser Abschnitt beschäftigt sich mit dem *Diplomacy*-Bot „Nice“, der von den Projektgruppenmitgliedern Niels Ackermann, Matthias Grochowski und Alireza Gholaman entwickelt wurde. Unser Ziel bei der Konstruktion des Nice war es, einen Verhandlungsspieler zu konstruieren, der die Spielweise eines Menschen möglichst gut simuliert. Hierbei wurde besonders Wert darauf gelegt, Verhandlungen mit anderen Spielern zu führen und diese Verhandlungen abhängig von der aktuellen Spielsituation zu steuern. Im Folgenden werden die einzelnen Module und die allgemeine Strategie der *Negotiation Intelligence with computional Emotion*(*NICE*) näher erläutert.

5.1.1 Konzept

Die Nicegruppe hatte als Konzept, einen bestehenden Diplomacy-Bot durch das Hinzufügen von Strategie- und Verhandlungsmodulen zu erweitern. Es wurde sich für die Erweiterung eines bestehenden Bots und gegen die Entwicklung eines komplett neuen Bots entschieden, da es bereits sehr gute taktische Bots gab, und die Gruppe durch Erweiterung eines solchen den Fokus verstärkt auf erweiterte Menschenähnlichkeit setzen konnte. Der Bot, auf welchen aufgebaut werden sollte, war der Diplominator (vgl. 4.5.7 auf Seite 25), welcher bereits strategische Entscheidungsfindung sowie grundlegende Verhandlungen beherrschte.

Die geplanten Strategiemodule umfassten:

- Das Clustern von Einheiten bei Truppenbewegungen.
- Die Optimierung von bestehenden Strategieroutinen.
- Das Finden von sinnvollen (Eröffnungs)-strategien.
- Die Priorisierung taktischer Zielsetzungen.

Neben der Erweiterung der strategischen Entscheidungsfindung ist ein wesentlicher Aspekt beim Spielen von Diplomacy die Verhandlung. Hierfür wurden folgende Verhandlungsmodule geplant:

- Wege zur sinnvollen Allianzbildung.
- Die Versendung von Zugnachrichten an befreundete Mächte.
- Das Erreichen eines Remis wenn ein Alleinsieg unmöglich scheint.
- Die Einrichtung von demilitarisierten Zonen.

Im Folgenden werden der zugrundeliegende Bot und die Ergebnisse der Entwicklung der unterschiedlichen Module vorgestellt. Abschließend werden die Auswirkungen auf die Spielstärke der unterschiedlichen Module in einem Modultest überprüft.

5.1.2 Taktik-Module

5.1.2.1 Cluster: Die Phalanx des NICE-Bots

Das Cluster-Modul ist dafür zuständig, die Einheitenbewegungen des Nice-Bots zu kontrollieren und dafür zu sorgen, dass die einzelnen Einheiten sich nicht zu weit von der Hauptarmee entfernen. Grund für ein solches Modul ist, dass es beim *Diplominator*, auf dem der Nice-Bot aufbaut, dazu kommen kann, dass Einheiten sich zu weit entfernen und isoliert von der Hauptarmee eine Provinz besetzen. Immer dann, wenn sich das aktuelle Spiel in einer Bewegungsphase befindet, wird das Cluster-Modul eingesetzt. Nachdem für jede Einheit der aktuellen Macht, mit der der Nice-Bot spielt, eine Provinz berechnet wurde, in die sie ziehen soll, kommt der Cluster-Algorithmus zum Einsatz. Eine Einheit zieht unter bestimmten Voraussetzungen nicht in eine Provinz. Nämlich genau dann, wenn sie nicht innerhalb von zwei Zügen eine Provinz, auf der eine andere Einheit derselben Macht steht, erreichen kann. Es wird für diese Einheit eine neue Provinz berechnet, in die sie ziehen soll. Dabei findet erneut eine Gegenprüfung (seitens des Cluster-Moduls) statt. Der Abstand von maximal zwei Provinzen wurde gewählt, weil ein grösserer Abstand (z.B. Abstand drei) zu Lücken innerhalb der Hauptarmee führt. Diese Tatsache beruht lediglich auf visuellen Erfahrungen beim Testen mit dem *DAIDE-Mapper* und wurde nicht genauer validiert. Besitzt der Nice-Bot nur noch eine Einheit, kommt das Cluster-Modul nicht zum Einsatz. Aufgrund der Tatsache, dass es dazu kommen kann, dass sich zwei Einheiten beliebig weit von der restlichen Armee entfernen, sich also zweier Cluster bilden, und der Ergebnisse des Modultests 5.1.4 auf Seite 36, wurde das Cluster-Modul im Nice-Bot deaktiviert.

5.1.2.2 EA: Die diverse Evolution der Staatengebilde

Da der Nice-Bot auf dem Diplominator aufbaut, werden die strategischen Entscheidungen des Nice-Bots stark durch die des Diplominators mitbestimmt. Die Gewichtung verschiedenster Parameter bei der Entscheidungsfindung des Diplominators sind dynamisch anpassbar, und die Gruppe hat vermutet, dass durch Optimierung der verschiedenen Parameterwerte die Spielstärke erhöht werden kann. Folgende Parameter wurden untersucht:

Parameter	Beschreibung	Standardwert
sprattack	Gewicht der offensiven Züge im Frühling	700
sprdefence	Gewicht der defensiven Züge im Frühling	200
fallattack	Gewicht der offensiven Züge im Herbst	600
falldefence	Gewicht der defensiven Züge im Herbst	300

Tabelle 5.1: Nice-Bot: Beschreibung der zu optimierenden Parameter

Ziel der Gruppe war es nun, durch Veränderung an der Parameterbelegung eine möglichst spielstarke Belegung zu finden und die Gruppe entschied sich für die Optimierung einen EA zu nutzen. Der EA wurde wie folgt eingesetzt:

- Jedes Individuum i wird durch seinen Parametervektor (sprattack sprdefence fallattack falldefence) definiert.
- Das erste Individuum ist ein Individuum, welches die Standardparameter des Diplominators besitzt also (700 200 600 300).

- Durch Mutation der Parameter wird versucht eine Parameterkombination zu finden, welche sich im Spiel als mindestens genauso spielstark erweist. Bei der Erstellung eines neuen Individuums wird jeder Wert des aktuellen Individuums mit Wahrscheinlichkeit $p = 0.25$ mutiert. Wenn ein Wert mutiert wird, so geschieht dies durch Addition einer normalverteilten Zufallsvariable mit dem Erwartungswert 50 und einer Varianz von 0.05. Sollte nach diesem Durchgang keiner der Werte modifiziert worden sein, wird einer der Werte zufällig ausgewählt und wie beschrieben mutiert.
- Insgesamt wird die Parameterkombination eines Individuums fünf Spiele lang eingesetzt.
- Als Zielfunktion wird, gemittelt über die fünf Spiele, die durchschnittliche Anzahl an Versorgungszentren am Ende eines Spiels bestimmt. Sollte dieser Wert beim aktuellen Individuum mindestens gleich dem des letzten Individuums sein, wird das alte Individuum durch das neue ersetzt.
- Der Definitionsbereich für die Parameter ist auf das Intervall $[0, 1000]$ festgelegt. Wenn der für einen Parameter durch die Mutation neu berechnete Wert x dieses Intervall verlässt, dann wird er mit der Funktion f in das Intervall zurückgespiegelt.

$$f(x) = \begin{cases} |x|, & \text{falls } x < 0 \\ x, & \text{falls } x \in [0, 1000] \\ 2000 - x, & \text{falls } x > 1000 \end{cases}$$

- Insgesamt wurden pro Nation 20 Generationen ausgewertet, also je 100 Spiele. Für die Nationen, welche gerade nicht optimiert wurden, wurde statisch die Standardkonfiguration (700 200 600 300) gewählt.

In der folgenden Tabelle 5.2 wird pro Land die Parameterkombination des stärksten Individuums in der Testreihe dargestellt. Auffällig ist, dass sich die Parameterwerte von Russland im Vergleich zu den Standardwerten des Diplominators nicht verändert haben. Das bedeutet, dass von unserem EA kein Individuum generiert wurde, welches eine durchschnittlich höhere Anzahl an Versorgungszentren in den gespielten Spielen erreichen konnte. Hieraus folgt nicht, dass es sich um eine optimale Belegung handelt sondern lediglich, dass unser EA den Wert nicht verbessern konnte.

Macht	sprattack	sprdefence	fallattack	falldefence
Österreich	753	201	639	300
England	782	195	630	307
Frankreich	729	210	670	320
Deutschland	746	200	600	273
Italien	755	211	588	284
Russland	700	200	600	300
Türkei	746	176	664	321

Tabelle 5.2: Nice-Bot: optimierte Parameterwerte

Abschließend wird in Tabelle 5.3 ein Bot, welcher die Parameter des Diplominators hat, mit einem Bot, welcher die modifizierten Parameter hat, verglichen.

Der erste Wert bezeichnet die durchschnittliche Anzahl an Versorgungszentren, welche bei dem ersten Individuum mit Diplominatorparametern nach fünf Spielen gemessen wurde. Der zweite Wert bezeichnet den durchschnittlichen Wert der Versorgungszentren, der beim Spielen mit der vom EA gefundenen spielstärksten Parameterbelegung

Macht	Standard	Optimiert	Verbesserung (in %)
Österreich	5	8	60
England	9	11	22,2
Frankreich	12	13	8,3
Deutschland	6	7	16,7
Italien	7	9	28,6
Russland	12	12	0
Türkei	7	8	14,3
Durchschnitt	**8,3**	**9,7**	**16,9**

Tabelle 5.3: Nice-Bot: Vergleich der durchschnittlichen Anzahl Supplycenter

gemessen wurde. Die gemessene Verbesserung schwankt zwischen 0% also keiner Verbesserung bei Russland und 60% Verbesserung bei Österreich. Im Durchschnitt wurde eine Verbesserung von ca. 16,9 % gemessen.

5.1.2.3 Eröffnungsbuch: Ein guter Start in *Diplomacy*

Auf der Internetplattform *The Diplomatic Pouch* [Han09] werden eine Reihe von Eröffnungszügen - gemeint sind die Bewegungen der Einheiten im Frühling 1901 - aufgelistet, die von menschlichen Spielern bevorzugt ausgeführt werden und strategisch den größten Nutzen bringen (ausgehend von der Position der Einheiten nach dem ersten Zug). Diese Eröffnungszüge können nur durchgeführt werden, weil die Startpositionen der Einheiten, in Abhängigkeit von der Macht mit der das Spiel begonnen wird, immer die gleichen sind. Erwähnenswert ist hierbei, dass der Nice-Bot immer die gleichen Startzüge wählt.Tabelle 5.4 zeigt die Startzüge des Nice-Bots.

5.1.2.4 Priorisierung taktischer Zielsetzungen

In Diplomacy ist es von entscheidener Bedeutung, einen taktischen Vorteil über seinen Gegner zu erlangen, indem wichtige Provinzen gehalten oder erobert werden. Dabei kann es sein, dass Ländereien im Zuge der sich wandelnden Spielsituationen eine neue Wertigkeit erhalten, also zunächst wichtige Provinzen beim Vorrücken auf das Kernland des Feindes ihre Wertigkeit verlieren, oder in einer Schlacht einer ehemals nebensächlichen Provinz eine entscheidende Bedeutung zufällt. Die Spielsituation zerfällt also in eine Betrachtung von Provinzen mit zeitlich diverser Relevanz.

Die Zielsetzung einer Macht sollte also darin bestehen, Provinzen mit aktuell höherem taktischen Gewicht zu bevorzugen.

Das Taktikmodul adressiert genau diese Problemstellung, indem es ermöglicht, dass basistaktische Berechnungen des Diplominators modifiziert werden können. Dies bewerkstelligt es, indem jede Provinz gegen eine Tabelle geprüft wird, ob ein Modifizierungsfaktor $[0,\infty]$ vorliegt und, sollte dies der Fall sein, die Provinzwertigkeit mit diesem multipliziert wird. Damit dies in einem zeitlich begrenztem Rahmen bleibt, ist jeder Modifizierungsfaktor nur eine bestimmte Anzahl an Jahreszeiten gültig. Ideen was diese Faktoren bewirken können finden sich im weiteren Verlauf.

Im folgenden die geplanten, jedoch nie vollständig realisierten, Zielsetzungen für dieses Modul:

Allianzbruch: Die Rache des Geschmähten

Eigene Macht	Provinzen auf dem Einheiten stehen	Provinzen wohin Einheiten ziehen
Österreich/Ungarn	Armee - Budapest Armee - Wien Flotte - Triest	Serbien Galizien Albanien
England	Armee - Liverpool Flotte - Edinburgh Flotte - London	Edinburgh Norwegisches Meer Nördliches Meer
Frankreich	Armee - Marseille Armee - Paris Flotte - Brest	Burgund (Unterstützung) Burgund Mittlerer Atlantik
Deutschland	Armee - Berlin Armee - München Flotte - Kiel	Kiel Ruhr Holland
Italien	Armee - Rom Armee - Venedig Flotte - Neapel	Apulien Tirol Ionisches Meer
Russland	Armee - Moskau Armee - Warschau Flotte - Sebastopolis Flotte - St. Petersburg	Ukraine Galizien Rumänien Bottnischer Meerbusen
Türkei	Armee - Konstantinopel Armee - Smyrna Flotte - Ankara	Bulgarien Armenien Schwarzes Meer

Tabelle 5.4: Eröffnungszüge des Nice-Bots

Trotz gründlicher Prüfung von Allianzangeboten kann es passieren, dass ein Alliierter dem Nice-Bot in den Rücken fällt. Verhindern lässt sich dies nicht, doch ungesühnt sollte dies nicht bleiben. Das Taktikmodul hat deshalb eine weitere Komponente, die es ermöglicht sämtliche Provinzen einer Macht mit einem Modifikator zu belegen. Nach jeder Jahreszeit muss diese Provinztabelle aktualisiert werden, da Ländereien ihren Besitzer wechseln konnten. Aufgrund der Schwierigkeit einen Allianzbruch zu erkennen, wurde diese Komponente nie eingesetzt.

Charaktere

Nebst der taktischen Entscheidungsführung kann das Taktikmodul auch genutzt werden, um unterschiedliche Charaktere darzustellen. Soll beispielsweise ein defensiver Spieler erstellt werden, können die eigenen Provinzen mehr Relevanz erhalten, was den Nice-Bot dazu bringt, diese stärker zu verteidigen und weniger Angriffe auf fremdes Teritorium zu vollführen. Dem Streben nach eine einheitliche Spielstärke zu erlangen wurden keine Charaktere ins Spiel eingebaut.

Shoppinglisten

Sogenannte Shoppinglisten sind eine Menge von mindestens 18 Provinzen, die länderspezifische Präferenzen bezüglich der Supportcenter, die zum Sieg führen, bereitstellen und sind somit auf ein Solo ausgelegt. Diese Provinzen sind also der einfachen Erreichbarkeit und der guten Verteidigung halber in diese Liste aufzunehmen.

Beispielsweise ist München (Mun) für alle in der Nähe befindlichen Mächte ein lohnendes Ziel, dessen zentrale Lage und taktische Position in Zentraleuropa viele Möglichkeiten eröffnet. Dafür sind Länder, die weit ab der eigenen Region liegen, uninteressant,

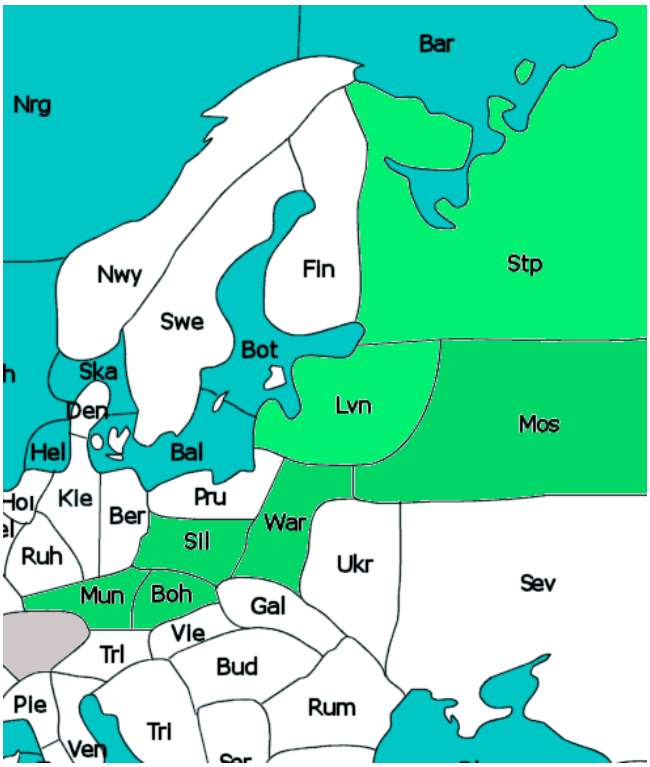


Abbildung 10: Die Ost-West Stalemate Line

beispielweise wäre München für das Osmanische Reich (TUR) nicht von allzu großer Bedeutung und London (Lon) ist noch weiter entfernt, was es erschwert die Provinz zu erobern und zu verteidigen.

Ein weiterer wichtiger Punkt in der Auswahl der Shoppinglistprovinzen sind sogenannte Stalemate Lines, zu dt. etwa Stillstandslinien (im Sinne eines Patts). Dies sind Linien, die, vorausgesetzt einer bestimmten Anzahl an Armeen, gegen beliebig viele Armeen von einer Seite verteidigt werden können. Dies dient zur sorgenfreien Sicherung der Ländereien gegen Angriffe.

Ein gutes Beispiel ist hier die Ost-West Stalemate Line, siehe Abbildung 10, mit München (Mun), Böhmen (Boh), Schlesien (Sil), Warschau (War) und Moskau (Mos), um sich gegen den Norden zu verteidigen, oder mit zusätzlich St. Petersburg (Stp) und Livland(Lvn), um eine sichere Position gegen Angriffe aus dem Süden zu errichten. Für nähere Informationen wird der Text 'Long-Term Planning' von Stephen Agar (vgl. [Aga92]) empfohlen.

Die Shoppingliste und die damit verbundene Notwendigkeit, spezielle Provinzen zu bevorzugen, wird durch das Taktikmodul ermöglicht, indem eine Liste mit Modifizierungsfaktoren zu Beginn bereitgestellt werden kann, diese Liste wurde jedoch nie erstellt.

Aufgrund zeitlicher Knappheit konnten notwendige Erweiterungsmaßnahmen im Nice-Code zur Nutzung des Allianzbruchs nicht mehr umgesetzt werden. Die Charakter- und Shoppinglisten-Umsetzungen blieben Planspiel. Somit ist das Tatikmodul implementiert, die geplanten Zielsetzungen wurden jedoch nicht erreicht.

5.1.3 Verhandlungsmodule

5.1.3.1 ALY: Das Schmieden von Allianzen

Das Allianz-Modul ist dafür zuständig, potenzielle Verbündete zu erkennen und diesen Vorschläge für ein Bündnis zu unterbreiten. Dabei enthält der Nice-Bot zwei verschiedene Allianz-Module, ein statisches und ein für den späteren Spielverlauf gedachtes dynamisches Allianz-Modul. Dies ist darin begründet, als dass die Machtverteilung und die Positionen zu Beginn fest, doch im Laufe des Spiels große Änderung erfahren. Es wird allerdings nur das statische Modul eingesetzt, da aus Zeitgründen das Dynamische nie abschliessend implementiert werden konnte. Das statische Allianz-Modul begründet sich darauf, dass die Ausgangsposition zu Beginn des Spiels nur von der gespielten Macht abhängt und somit immer gleich ist. So ist für Frankreich Deutschland kein guter Verbündeter, weil so die Möglichkeit seine Armeen sinnvoll einzusetzen stark eingeschränkt wird, solange Frankreich Deutschland nicht in den Rücken fällt. Das liegt vorallem daran, dass Frankreich nur zwei Nachbarn an Land hat, wobei Italien eine gute defensive Position gegenüber Frankreich inne hat. Italien hingegen kann für Frankreich zu einem starken Verbündeten werden, da der südwestliche Kartenrand als sichere Grenze dient und Deutschland, im Ballungsgebiet Europas, ein lohnendes Ziel darstellt. Weiterhin hat die Allianz im weiteren Spielverlauf (nach der Eroberung Deutschlands) viele Möglichkeiten, als das Osteuropa und Östereich/Ungarn als potentielle Angriffsziele bleiben, und es somit zu keinem Konflikt zwischen Italien und Frankreich kommen muss. Ausgehend von dieser Tatsache, kann man zu Beginn jedes Spiels von „besseren“ und „schlechteren“ Allianz-Partnern sprechen. Für den Nice-Bot sind mögliche Alliierte aus Tabelle 5.5 zu entnehmen. Dabei ist zu beachten, dass für jede Macht ein zweiter potentieller Alliierter zur Verfügung steht. Dieser wird jedoch nur Bündnispartner, wenn der erste Allianzfavorit nach einmaligem Anfragen kein Interesse an einer Allianz zeigt oder nicht verhandlungsbereit ist. Es ist zu erwähnen, dass wir die ausgewählten Bündnispartner der Internetplattform *The Diplomatic Pouch* [Han09] entnommen haben.

Eigene Macht	Erster Alliierter	Zweiter Alliierter
Österreich/Ungarn	Deutschland	Russland
England	Frankreich	Deutschland
Frankreich	Italien	Russland
Italien	Frankreich	Türkei
Deutschland	Österreich/Ungarn	England
Russland	Türkei	Österreich/Ungarn
Türkei	Russland	Italien

Tabelle 5.5: Bevorzugte Alliierte

Das dynamische Allianz-Modul versucht, Bündnisse mit der Macht einzugehen, die die zweitmeisten angrenzenden Provinzen zu der eigenen Macht besitzt. Es wird ein Bündnis vorgeschlagen, das sich gegen die Macht mit den meisten angrenzenden Provinzen zu unserer Macht richtet. Der Zweck dieses Vorgehens besteht darin, dass von der Macht, die die meisten adjazenten Provinzen zu unserer Macht hat, oftmals das größte Gefahrenpotential für mögliche Angriffe ausgeht, daher sollte diese Macht zusammen mit der Unterstützung des Alliierten zu allererst angegriffen werden. Wichtig zu erwähnen ist noch, dass der Nice-Bot in jeder Phase eines Spiels maximal eine Allianz mit einer Macht eingeht, weil es bei mehreren Alliierten zu Konflikten zwischen den Allianzpartnern kommen kann. Bei einem Allianzbruch einer Macht gegenüber dem Nice-Bot sollte ein anderes Modul aktiviert werden, welches untersucht, ob es sich lohnt, einen Gegen-

schlag an der ehemals verbündeten Macht durchzuführen. Dieses Modul wird jedoch nicht eingesetzt, weil es nicht fertig gestellt wurde.

5.1.3.2 XDO: Der Schrei nach Unterstützung

Ein wesentlicher Bestandteil der Level 20 Kommunikation ist es, Zugvorschläge an andere Mächte senden zu können und auf eingehende Zugvorschläge reagieren zu können. Im Folgenden wird zunächst auf die entworfenen Konzepte eingegangen und dann die Funktionsweise näher erläutert.

Für das Empfangen und Verarbeiten von eingehenden Zugvorschlägen war im *Diplominator* bereits eine rudimentäre Codebasis vorhanden. Die Gruppe entschied sich dafür, die bestehende Basis zu erweitern und für das Versenden ein komplett neues Verfahren zu entwickeln. Die Ansätze waren folgende:

- Einkommende Zugvorschläge: Die Codebasis des *Diplominators* sah vor, dass sämtliche zu behandelnden Zugvorschläge in einem Bankensystem zu verwalten sind. Immer wenn ein Mitspieler eine Unterstützung bei einem Zug wünscht, wird überprüft, ob er noch genügend Kredit auf seinem Bankkonto hat. Die Höhe des Kredites hängt von dem Wert ab, welchen der *Diplominator* für das Land in der Strategiephase bestimmt hat. Wenn der Kredit des Spielers ausreicht, wird der Zugvorschlag durchgeführt und der Spieler unterstützt. Ein Mitspieler hat selber die Möglichkeit, sein Bankguthaben zu erhöhen, indem er Vorschläge des Spielers akzeptiert und durchführt. Die Summe der Gutschrift berechnet sich analog zur Höhe der Lastschrift durch Bewertung des Ziellandes. Wichtig für die Gutschrift ist es, dass die Unterstützung von unserem Bot gewünscht wurde. Unterstützte Züge, welche vorher nicht angefragt wurden, resultieren nicht in einer Bankgutschrift.
- Ausgehende Zugvorschläge: Um entscheiden zu können, welche Unterstützungen angefragt werden sollen, muss überlegt werden, welche Züge eines Mitspielers dem Spieler in der aktuellen Spielsituation den größten Nutzen bringen. Hierfür wird eine normale Spielzugberechnung durchgeführt, bei welcher jedoch nicht nur für die eigenen Einheiten ein Zug berechnet wird, sondern so getan wird, als ob der Spieler ebenfalls frei über die Einheiten eines Mitspielers entscheiden darf. Hierbei ist zu beachten, dass für die Berechnung der Züge die bereits vorhandenen Diplominatormethoden genutzt werden. Die einfließenden Faktoren können im Abschnitt 4.5, welcher sich mit den bereits existierenden Bots beschäftigt, eingesehen werden. In der Zugphase werden nun die Befehle für die eigenen Einheiten wie gewohnt durchgeführt und die Züge für die Einheiten des Mitspielers als Zugvorschlag übersendet. Für die Zugvorschläge an die Mitspieler werden folgende drei Listen verwaltet.

    1. Versendete Anfragen: Sobald der Spieler einen Zugvorschlag sendet, wird dies gespeichert.
    2. Versendete und akzeptierte Anfragen: Sollte der Mitspieler den Zugvorschlag akzeptieren wird dieser in diese Liste verschoben.
    3. Akzeptierte und durchgeführte Anfragen: Der Mitspieler hat die Anfrage durchgeführt. Nun findet die Bankgutschrift statt.

An einem Beispiel (die Ausgangssituation ist veranschaulicht in Abbildung 11) wird die Vorgehensweise beim XDO verdeutlicht: In einem Spiel spielt ein Nice-Bot Russland, und es gelang dem Bot, Finnland zu erobern, wo er nun eine Armee besitzt. Deutschland hat eine Armee in Schweden, und das alliierte England hat eine Armee in Norwegen.

Nun möchte der Nice-Bot Schweden erobern. Die einzige Möglichkeit, Schweden zu besetzen, ist es, die Armee von Finnland nach Schweden zu bewegen.

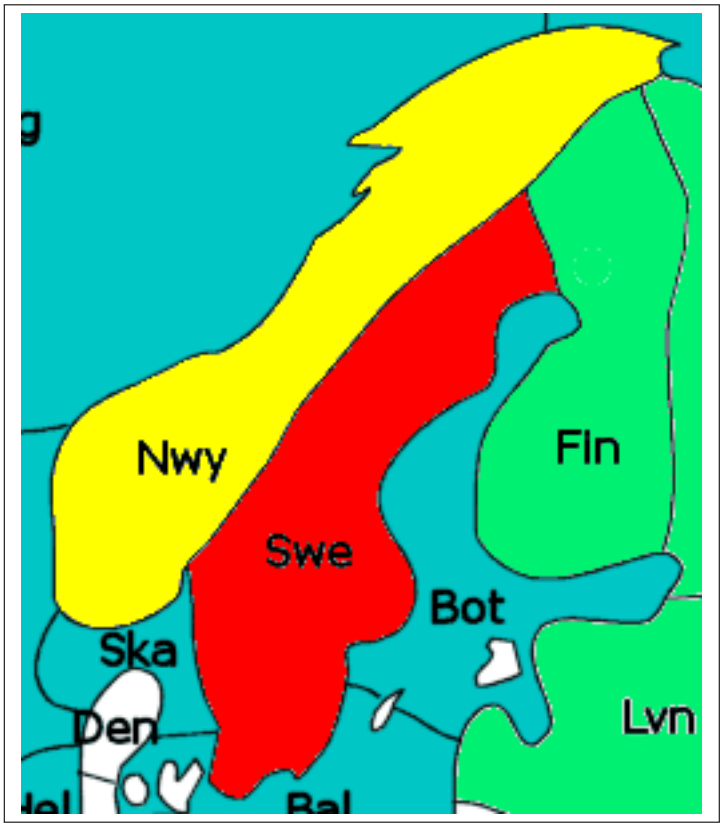


Abbildung 11: XDO: Beispielszenario

Während der Strategieberechnung erkennt der Nice-Bot, dass er nicht in der Lage ist Schweden erobern zu können, da eine Armee nicht ausreicht, um eine andere Armee zu schlagen. Man benötigt stets eine unterstützende Armee. Der Nice-Bot berechnet nun wie die Situation wäre, wenn ihm nicht nur die eigene Armee zur Verfügung stünde, sondern auch die vom alliierten England.
Die Strategieberechnung unter diesen neuen Voraussetzungen ergibt nun, dass die Armee in Finnland in der Lage ist, Schweden zu erobern, wenn die Armee in Norwegen sie dabei unterstützt. Da der Nice-Bot nur über die russischen Armeen verfügen kann, und nicht über die des englischen Spielers, wird eine Anfrage auf Unterstützung dahingehend gestellt, dass Norwegen Finnland beim Angriff auf Schweden unterstützt.

5.1.3.3 DRW: Das Angebot einer friedlichen Koexistenz

Dieses Modul ist dafür zuständig, den anderen Mächten ein Unentschieden (engl. draw, Abk. drw) anzubieten. Ein Unentschieden wird immer dann vorgeschlagen, wenn die eigene Macht weniger Versorgungszentren als jede andere Macht hat und somit sich im Vergleich zu den anderen Mächten in einer schlechteren Lage befindet. Wenn $\forall k_\delta$: $k_\alpha < k_\delta$ gilt, wobei k_δ die Anzahl der Versorgungszentren für eine Macht $\delta \in \Delta$, Δ die Menge aller Mächte und k_α die Anzahl der Versorgungszentren für die eigene Macht, dann unterbreitet der Nice-Bot für jedes $\delta \in \Delta$ ein Unentschieden. Es wird immer allen Mächten ein Angebot unterbreitet, wobei ein Unentschieden in *Diplomacy* erst dann feststeht, wenn alle Mächte ein Unentschieden akzeptiert haben, oder sofern die

Option aktiviert ist, Mächte mit insgesamt mehr als der Hälfte aller Versorgungszentren zustimmten (partial draw). Falls die Siegchancen des Nice-Bots gering sind, bietet sich das Unentschieden an, um das aktuelle Spiel wenigstens nicht zu verlieren.

5.1.3.4 DMZ: Die Befriedung von Provinzen

Wir sind der Meinung, das einige Mächte, nämlich jene, die ihre Startposition in Zentral- und Osteuropa haben, stärker eingeschränkt sind, weil diese Mächte von nahezu allen Seiten angreifbar sind. Daher versucht der Nice-Bot, die Stellung dieser benachteiligten Mächte mit Hilfe von demilitarisierten Zonen (DMZ) zu verbessern. Diese demilitarisierten Zonen besagen, dass Truppen der Macht, mit der ein DMZ vereinbart wurde, nicht in dieses Gebiet eindringen sollen. Wir erhoffen uns durch dieses Vorgehen, die unsichere Position in Zentral- und Osteuropa zu stabilisieren. Der Nice-Bot schlägt in jeder Bewegungsphase vor, seine demilitarisierten Zonen auf die Provinzen, die die heimischen Versorgungszentren bilden, zu setzen (engl. Home Supply Centre – HSC). HSC sind anfänglich schwer zu verteidigen, da zu Beginn des Spiels äußerst wenige Einheiten zur Verfügung stehen. So wird in Abhängigkeit von der Startmacht an bestimmte Mächte ein Angebot unterbreitet. Falls dieses Angebot von der anderen Macht angenommen wird, so sollte diese nicht in die HSC eindringen. Im Gegenzug wird ein Angebot gemacht, das garantiert, dass auch der Nice-Bot nicht in die HSC der Macht eindringt, mit der die DMZ gilt. Der Nice-Bot hält sich an diesen „Vertrag“ und greift die HSC des Vertragspartners nicht an, solange, abhängig von der Startmacht, eine bestimmte Anzahl von Versorgungszentren nicht überschritten wurde. Tabelle 5.6 zeigt, mit welchen Mächten der Nice-Bot ein DMZ eingehen will.

Eigene Macht	Macht mit der ein DMZ eingegangen wird
Deutschland	Österreich/Ungarn, Russland
Österreich/Ungarn	Deutschland, Russland, Türkei (nur wenn Deutschland und Russland ablehnen)
Russland	Österreich/Ungarn, Deutschland

Tabelle 5.6: Verhandlungspartner des Nice-Bots für demilitarisierte Zonen

5.1.4 Modultest

Aufgrund der Tatsache, auf einen schon existierenden Bot aufgebaut zu haben, war es möglich, die direkten Auswirkungen der erweiternden Implementierungen zu prüfen. Der Vergleich besteht daraus, dass der Nice-Bot gegen seinen Ursprungsbot, den Diplominator, in einem Diplomacyspiel antritt.

Zwei Bots in einem Spiel mit sieben Teilnehmern in fairer Art und Weise gegeneinander antreten zu lassen, war nun die Aufgabe, die es zu lösen galt. Das Ergebnis des Entscheidungsprozesses ist, drei Nice-Bots gegen vier Diplominatoren antreten zu lassen. Somit hat der Nice-Bot die Aufgabe, 42,8% (3/7) der Spiele zu gewinnen, um sich als gleichwertig zu beweisen, oder mehr zu erlangen, um den Diplominator zu überflügeln. Am erreichten Prozentwert kann dann die Verbesserung oder Verschlechterung gegenüber dem Diplominator abgelesen werden. Aufgrund der Möglichkeit mehr als nur eine binäre Entscheidung zu erhalten, wurde eine Erweiterung des Tests vorgenommen.

Um die Auswirkungen der einzelnen Implementierungen auf die Spielweise des Bots zu testen, wurden diese in spieltechnisch sinnhafte Gruppen zusammengefasst. Diese Module konnten nun durch Ein- und Ausschaltbarkeit in den Testspielen in verschiedenen

Kombinationen antreten. Dazu wurde der Nice-Bot um eine solche Konfigurierbarkeit erweitert.

Die Folge waren die fünf Module:

- ALY (inklusive DMZ)
- Cluster
- EA-Parameter
- Eröffnungsbuch
- XDO

ALY (siehe 5.1.3.1 auf S. 33) beschreibt hierbei die Bildung von Allianzen mit anderen Mächten, dies geht mit Verhandlungen zu demilitarisierten Zonen (DMZ) (siehe 5.1.3.4 auf S. 36) einher.

Beim Cluster (siehe 5.1.2.1 auf S. 28) handelt es sich um die Verwaltung der Truppen im Sinne einer Haufenbildung.

Die vom EA (siehe 5.1.2.2 auf S. 28) gewonnenen Parameter verändern die Aggressivität und das Defensivverhalten gegenüber anderen Mächten.

Das Eröffnungsbuch (siehe 5.1.2.3 auf S. 30) legt den ersten Zug auf statische Weise fest.

Das XDO (siehe 5.1.3.2 auf S. 34) lässt den Nice-Bot mit Mächten kooperieren, indem Zugvorschläge ausgetauscht werden.

Um einen ersten Eindruck zu gewinnen, wurde ein vollfaktorielles Design (vgl [Mon01]) entworfen, in welchem für jede Konstellation 30 Spiele zu vollführen waren. Die Größenordnung der Anzahl Spiele lag somit bei $2^5 \cdot 30 = 960$, wobei ein Spiel in etwa 40 Sekunden entschieden war. Um keinen Einfluss auf die Ergebnisse des Tests zu nehmen, wurde in jedem Spiel mit zufällig gleichverteilten Ausgangspositionen der Mächte gespielt.

Das ermittelte Ergebnis zeigte deutlich auf, dass 30 Spiele nicht genug waren, um ein geringes Signifikanzniveau zu gewährleisten.

Dies implizierte eine Abänderung des Testsetups. Da eine Veränderung des Verhältnisses von Nice zu Diplominator nicht erfolgversprechend aussah, wurde die Anzahl Kombinationen reduziert und ein teilfaktorielles Design mit 13 Kombinationen zu je 100 Spielen kreiert. Hierbei entspricht die Reihenfolge der der oben eingeführten Module ALY, Cluster, EA, Eröffnung, XDO und die Bezeichnung ‘t’ und ‘f’ ob die Module in dem Testsetup ‘aktiviert’(t) oder ‘deaktiviert’(f) sind.

- ttttt
- ttttf, tttft, ttftt, tfttt, ftttt
- ftttf
- fffft, ffftf, fftff, ftfff, tffff
- fffff

Die Idee dahinter bestand daraus, beobachten zu können, ob ein einzelnes Modul eine große Veränderung im Spielverhalten, sei es durch Abschalten im Nice-Bot, oder Hinzufügen zur Basisversion (‘fffff’) des Nice-Bots, welche den Diplominator darstellt, hervorruft. Eine Kombination (‘ftttf’) wurde aus persönlichem Interesse hinzugefügt.

Desweiteren wurde in einem weiteren Schritt die Möglichkeit der Festlegung der Mächte auf ein immer gleiches Setup geprüft: Die festgesetzten Mächte entsprachen Österreich-Ungarn (AUS), dem Vereinigten Königreich von Großbritannien und Irland (ENG) und dem Deutschen Reich (GER) für den Nice-Bot. Diese Auswahl wurde daran angelehnt, als das Österreich-Ungarn und das Deutsche Reich eine Allianz gründen sollen (siehe das Allianzmodul 5.1.3.1 auf S.33), um das Allianzmodul zu testen, als auch das das Setup eine einigermaßen faire Kräfteverteilung versprach. Die Ergebnisse sind in Tabelle 5.7 zu sehen.

	rand Powers(100)		**fixed Powers(100)**	
Komb	gew Spiele	σ	gew Spiele	σ
ttttt	43,0%	15,5	32,0%	15,3
ttttf	39,0%	17,0	41,0%	18,1
tttft	40,0%	13,4	33,0%	6,4
ttftt	46,0%	10,1	30,0%	13,4
tfttt	48,0%	19,8	38,0%	14,0
tffff	38,0%	13,5	38,0%	10,7
ftttt	47,0%	12,6	41,0%	15,7
fttft	44,0%	15,6	42,0%	17,2
ftfff	39,0%	20,7	42,0%	16,0
fftff	41,0%	10,4	51,0%	13,7
fffft	45,0%	8,0	51,0%	16,4
ffffft	44,0%	12,0	41,0%	12,2
fffff	51,0%	21,1	40,0%	19,4

Tabelle 5.7: Diplominator gegen Nice nach 100 Spielen

Aufgrund der großen Schwankungen der Gewinnraten zwischen den zufällig gewählten und den festgesetzten Mächten innerhalb der Kombinationen wurde eine Prüfung der Standardabweichung angestrebt. Um hierbei die Standardabweichung σ zu berechnen, wurden je 10 Spiele nach zeitlicher Ordnung zu einem Bündel kombiniert und via Stichprobenschätzer eine Varianz $Var(X)$, letztendlich die Standardabweichung, berechnet. Die hier benutzte Formel, mit den Bündeln X, x_i als Spiele des Bündels und n als Anzahl der Bündel, lautet:

$\sigma = \sqrt{Var(X)}$
mit $Var(X) = 1/n * \sum_{i=1}^{n}(x_i - \mu)^2$
mit dem Mittelwert $\mu = 1/n * \sum_{i=1}^{n}(x_i)$

Die Standardabweichung der Ergebnisse war sehr groß und ließ vermuten, dass eine {0,1}-Entscheidung nicht das probate Mittel war. Eine Umstellung auf die höchste Anzahl der Supplycenter der Nice-Bots in einem Spiel brachte eine deutlich geringere Standardabweichung.

Da der Versuch mit festen Mächten am erfolgversprechendsten erschien, wurden hier weitere 300 Spiele investiert und die Bündelgröße auf 20 angehoben.

Im Ergebnis in Tabelle 5.8 sind einige Unterschiede zwischen Modulkombinationen zu erkennen. Interessant ist beispielsweise, dass selbst die Basisversion des Nice-Bots, der Diplominator, hier mit 'fffff' gekennzeichnet, in der Konstellation der Mächte nicht an die geforderten 42,8% heranreicht. Eine simple Anpassung der Aggressivitätswerte ('fftff') bringt hier die größte Verbesserung, die Kombination in der alle Module aktiviert sind ('ttttt') schneidet am schlechtesten ab. Problematisch ist hier die Standardabweichung, die durchschnittlich bei 1,4 liegt, während der maximale Unterschied zwischen

	fixed Powers(400)		
Komb	gew Spiele	#sup cen	σ(#sup cen)
ttttt	32,5%	9,6	1,5
ttttf	37,0%	10,4	1,2
tttft	34,2%	10,1	1,7
ttftt	37,7%	10,5	1,5
tfttt	36,2%	10,4	1,1
tffff	35,7%	10,1	1,7
ftttt	33,2%	9,9	1,3
ftttf	37,2%	10,3	1,4
ftfff	37,7%	10,2	1,5
fftff	39,7%	10,7	1,6
ffftf	37,5%	10,5	1,5
fffft	33,2%	9,9	1,1
fffff	38,2%	10,6	1,7

Tabelle 5.8: Diplominator vs Nice nach 400 Spielen

den Werten mit 1,1 noch darunter fällt. Ein Test auf statistische Signifikanz wurde nicht durchgeführt.

5.1.5 Fazit

Obwohl nicht alle der entworfenen Module in dem geplanten Umfang realisiert werden konnten, was auf den teilweise sehr großen Aufwand für die Erarbeitung und Implementierung der größeren Module zurückzuführen ist, kann man die Arbeiten der Gruppe dennoch als erfolgreich bezeichnen. Durch die Erweiterung der Kommunikationsstärke von Presslevel 10 auf Presslevel 20 gelang es, die Interaktion zwischen dem Nice-Bot und den anderen Mitspielern deutlich zu erhöhen. Die Kommunikation besteht nicht länger nur aus der Vertragsschließung von Nichtangriffspakten, sondern es werden priorisiert Allianzpartner gesucht und mit diesen gezielt gegenseitige Unterstützungen mit Hilfe von versendeten und empfangenen Zugvorschlägen koordiniert und durchgeführt. Der modulare Aufbau hat sich als sinnvoll erwiesen, da er das separierte Entwickeln und Testen sowie die Überprüfung der Wechselwirkungen von eingebauten Features vereinfachte.

5.2 Stragotiator

Der folgende Abschnitt beschreibt den von den Projektgruppenmitgliedern Marc Antonius Gorzala, Markus Kemmerling, Sebastian Uellenbeck und Wolfgang Walz erstellten Diplomacy Bot „Stragotiator“.

Der Begriff „Stragotiator“ ist ein Neologismus, komponiert aus den Kernmodulen **Stra**tegie Kern (vgl. 5.2.2.2 auf Seite 59) und Ne**gotiator** (vgl. 5.2.2.1 auf Seite 48).

Der „Stragotiator“ besteht aus den vier Modulen *Negotiator*, *Strategie Kern*, *Gegnerbewertung* und *EA-Optimierung*. Geplant wurde die Aufteilung von allen Gruppenmitgliederung. Hauptverantworlich für den *Negotiator* war Markus Kemmerling, Sebastian Uellenbeck kümmerte sich um den *Strategie Kern*, Marc Antonius Gorzala realisierte

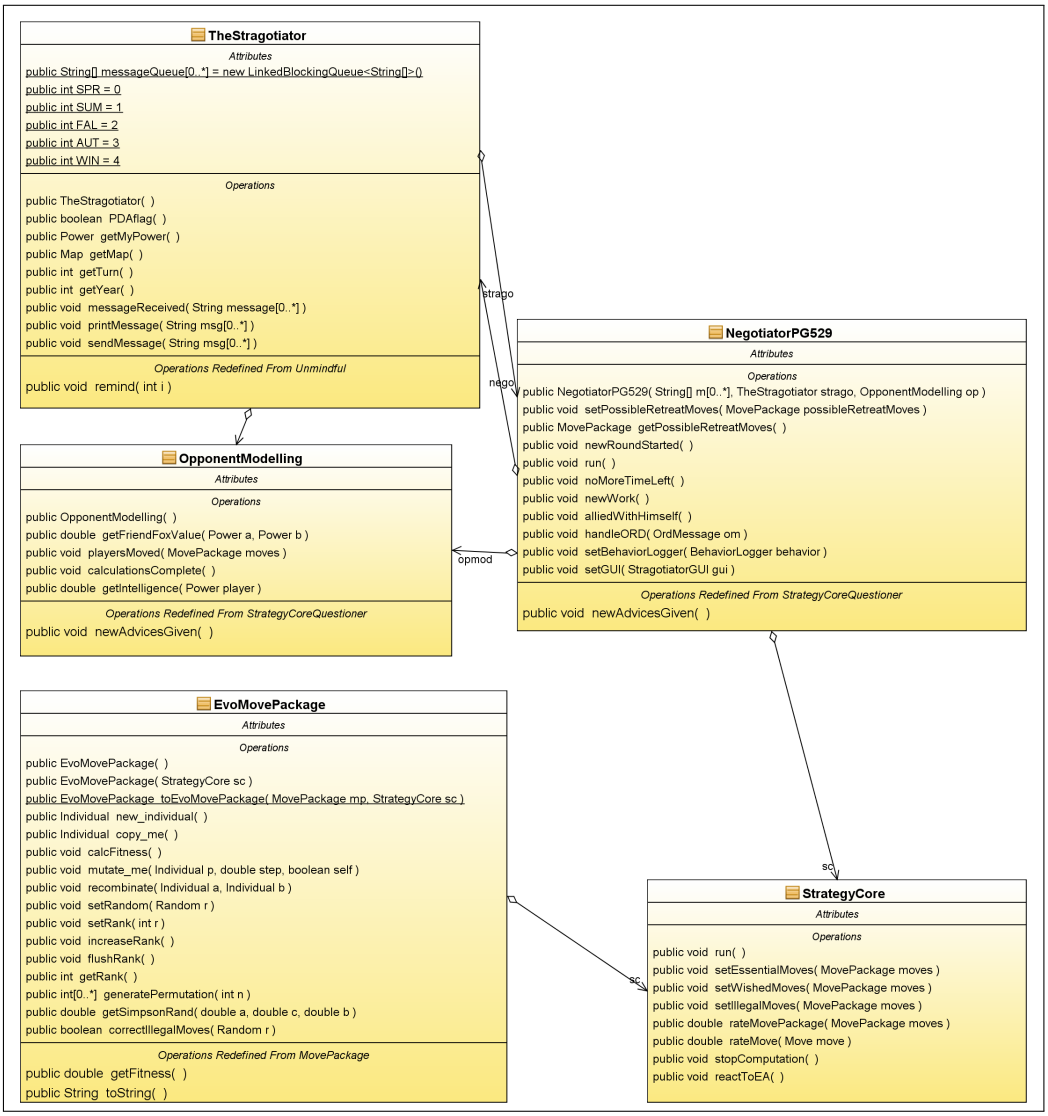


Abbildung 12: Klassendiagramm der Hauptklassen des *Stragotiators*

die *Gegnerbewertung* und Wolfgang Walz entwickelte die *EA-Optimierung*. Ein Klassendiagramm, das die Hauptklassen und deren Referezen zeigt, ist Abbildung 12 auf Seite 41.

Negotiator
Das Verhanldungsmodul übernimmt sämtliche Kommunikation mit anderen Mächten, wobei auf Informationen der *Gegnerbewertung* zurückgegriffen wird, und leitet die auszuführenden Züge, die von dem *Strategie Kern* berechnet werden, weiter.

Strategie Kern
Der *Strategie Kern* errechnet aus den vom *Negotiator* gegebenen Informationen, wie bspw. eigene Macht, Züge die ausgeführt werden müsse oder Züge, die nicht ausgeführt werden dürfen, Zugpakete und lässt diese durch die *EA-Optimierung* verbessern.

Gegnerbewertung
Die *Gegnerbewertung* übernimmt die Aufgabe, die Intelligenz der Gegner mit Hilfe des *Strategie Kerns* einzuschätzen und berechnet die Freund/Feind-Matrix. Die Matrix gibt für je zwei Mächte A und B an, ob Macht A sich gegenüber Macht B friedlich oder

feindlich verhält.

EA-Optimierung
Die *EA-Optimierung* verbessert Zugpakete für alle Einheiten bezüglich einer vom *Strategie Kern* vorgegebenen Zielfunktion.

5.2.1 Planung

In der Planungsphase wurde zunächst in verschiedenen Literaturquellen recherchiert, um gute Strategien im Spiele Diplomacy zu identifizieren. Auf Grundlage dieser Quellen wurden Grundideen und Ziele ausgearbeitet, welche sich hauptsächlich auf die Bereiche Taktik und Verhandlung konzentrierten. Die anschließende Modulplanung des Bots ermöglichte eine effiziente Aufgabenverteilung an die einzelnen Teammitglieder.

5.2.1.1 Paperinhalte

Weil sich die Gruppe vor dem Projekt weder mit theoretischen noch wissenschaftlichen Ansätzen bezüglich Diplomacy informiert hatte, beschäftigten wir uns in einer Literaturphase mit bereits vorhandenen Arbeiten, von denen wir die wichtigsten im Folgenden kurz beschreiben.

5.2.1.1.1 Paul D. Windsor - What's Your Point

In „What's Your Point“ (siehe [Win99]) beschreibt Paul D. Windsor vier grundlegende Spielertypen und ihr Verhalten.

Der Klassiker ist sehr auf Allianzen fixiert. Er versucht stets, mit mindestens einem anderen Spieler verbündet zu sein. Selber ist er in diesen Allianzen sehr loyal. Das bedeutet, dass er bei seinen Spielzügen immer das Wohl der ganzen Allianz im Blick hat und nicht nur sein eigenes. Dazu kommt, dass er seine Verbündeten nicht anlügt und ihnen vor allem nicht in den Rücken fällt. Zusätzlich denkt er bei Bündnissen immer an etwas langfristiges und strebt keinen alleinigen Sieg an. Er unterbreitet oft seinen Verbündeten das Angebot, mit ihnen zusammen zu gewinnen und akzeptiert solche Angebote auch gerne. Eine weitere Folge seines Verhalten ist, dass Allianzen zwischen zwei Klassikern so gut wie nicht zu zerstören sind. Wenn zwei Klassiker mit zwei benachbarten Mächten, die in einem Bündnis generell sehr stark sind (z.B. England und Frankreich), starten, dann dominiert dieses Bündnis in der Regel sehr schnell das Spiel. Dass diese beiden Mächte eine Allianz eingehen, ist höchst wahrscheinlich, weil ein Klassiker generell in der ersten Runde versucht Bündnisse mit seinen Nachbarn zu schließen. Spieltechnisch ist er sehr taktisch orientiert und geht ungerne Risiken ein. Weil er sehr viel Wert auf Allianzen legt, ist er selber auch ungerne die alles überragende Supermacht. Wird er in einem Bündnis einmal von seinem Partner hintergangen und angegriffen, dann sinnt er nicht auf Rache. Vielmehr versucht er, sich in eine Position zu manövrieren, in der er möglichst lange im Spiel bleiben kann. Wenn er am Anfang eines Spiels Pech hat und „an die Wand“ gedrückt wird, zieht er sich in eine gut zu verteidigende „Ecke“ zurück und versucht unter allen Umständen zu überleben und ein Unentschieden, einen so genannten Draw, zu erreichen.

Der Romantiker ist ein sehr risikofreudiger Spieler, der über die Folgen seiner Taten nicht lange nachdenkt. Er spielt egoistisch und strebt an, der alleinige Sieger zu sein. Ein Unentschieden ist in seinen Augen kein Sieg. Kommt er in eine Lage, in der er nicht mehr gewinnen kann, dann wird er ein „Shooting Star“, der sich

für den Vorteil eines Mitspielers aufopfert. Denn ein Sieg eines Gegners ist einem Unentschieden vorzuziehen. Er geht zwar gerne Bündnisse ein, betrachtet seine Bündnispartner allerdings auch als Gegner. Ein Bündnis dient ihm nur dazu, sich selber einen Vorteil zu verschaffen. Am Anfang des Spiels versucht er deswegen Allianzen gegen eine andere Macht zu schließen, um diese schnell aus dem Spiel zu werfen. Generell versucht er, eine schwache Macht so schnell wie möglich zu vernichten, um die Anzahl der verbleibenden Spieler möglichst gering zu halten. Weil er seine Bündnispartner als Gegner betrachtet, fällt er ihnen gerne in den Rücken. Für ihn ist ein Versorgungszentrum, welches zwar einem Bündnispartner gehört, dieser es aber nicht besetzt, ein freies Versorgungszentrum. Ebenso will er nicht, dass einer seiner Bündnispartner zu mächtig wird. Besitzt sein Partner also zu viele Versorgungszentren, dann stellt er in den Augen des Romantikers eine Gefahr für seinen Sieg dar und wird angegriffen. Weil er ein Bündnis aber auch nur eingeht, um sich selber einen Vorteil zu verschaffen, bricht er einen Vertrag auch sofort, wenn er darin keinen Nutzen für sich mehr sieht. Befindet er sich in einer Allianz und erhielt in den letzten zwei Runden kein neues Versorgungszentrum, versucht er, sich neue auf Kosten seines Alliierten zu holen. Kommt er selber in die Lage zehn oder mehr Versorgungszentren zu besitzen, trifft er nur noch Absprachen für eine Runde.

Der Clubspieler spielt um Spaß zu haben. Deswegen ist er auch völlig unberechenbar. Er geht gerne große Risiken ein und tut alles, was das Spiel aus seiner Sicht interessant macht. So kann er einem Verbündeten, weil das Spiel gerade langweilig ist, in den Rücken fallen. Kommt er in eine Lage, aus der er das Spiel nicht mehr gewinnen kann, ist es sowohl möglich, dass er ein Unentschieden anstrebt und akzeptiert, aber auch, dass er sich für einen anderen Spieler opfert. Der Spieler, für den er sich opfert, ist in der Regel derjenige, von dem er meint, dass er das Spiel besonders interessant gemacht hat. Gerät er in eine aussichtslose Position, dann strengt er sich noch mehr an und gibt unter keinen Umständen auf. Trotz allem ist der Clubspieler sehr sozial eingestellt und begeht Taten reiner Nächstenliebe. So akzeptiert er ein Unentschieden, obwohl er einen Alleinsieg erreichen könnte, oder beschützt einen Spieler vor dessen Auslöschung, ohne daraus einen strategischen Nutzen zu erzielen. Bei Verhandlungen neigt er dazu, Informationen über andere Spieler weiter zu geben.

Der Lügner ist häufig ein Anfänger. Jedoch gibt es auch erfahrene Spieler, die zu diesem Spielertyp zählen. Diese sind in der Regel sehr erfolgreich und spielstark. Der Lügner strebt einen Solosieg an. Seine Taktik besteht darin, andere Mächte gegeneinander auszuspielen um sich auf diese Weise das Leben leichter zu machen. Am Anfang des Spiels versucht er, das Vertrauen seiner Mitspieler zu gewinnen, um sie später besser gegeneinander aufzuhetzen. Dabei richtet er sich bevorzugt nach den Mächten, die die meisten Versorgungszentren besitzen, um Konfrontationen zu vermeiden. Im weiteren Verlauf versucht er, Misstrauen zwischen anderen Mächten zu säen und Bündnisse zu verhindern oder zu zerstören. Dazu gibt er Informationen über Vorhaben anderer Spieler an dritte weiter, wobei er sich nicht unbedingt an die Wahrheit hält. Über sich selber gibt er so wenig wie möglich preis. Er kümmert sich nicht darum, was andere über ihn denken, betrachtet Abkommen nicht als bindend und geht höchstens kurze Bündnisse ein. Um im Spiel für Verwirrung zu sorgen, löscht er schwache Mächte nicht aus. Sollte er in eine aussichtslose Situation kommen, fügt er sich seinem Schicksal. Denn er weiß, dass die anderen kein Bündnis mit ihm eingehen werden oder ihn an einem Unentschieden beteiligen.

5.2.1.1.2 Danny Loeb - Challenges in Multi-Player Gaming by Computer „Challenges in Multi-Player Gaming by Computer“ von Danny Loeb (siehe [Loe95a]) war das

Ausgangsmaterial, das unser Interesse am *Bordeaux Diplomaten* weckte. Loeb geht hier auf die Grundlagen von Diplomacy ein und stellt die Konzepte seines später entwickelten Bots vor. Als mathematische Grundlage beschreibt er die traditionelle Technik des Minmax-Algorithmus, der allerdings nur für Spiele mit zwei Spielern Verwendung findet. Schon auf Spiele mit drei Mitspielern lässt sich dieser Ansatz nicht mehr problemlos erweitern. Der Autor erörtert, warum die Verhandlung ein Kernelement von Diplomacy ist und warum ein NPC ein menschliches Verhalten simulieren sollte. Er beschreibt auch die Problem und Ergebnisse von Tests, die dem von uns später ausgeführten Turing Test (vgl. 6.4 auf Seite 116) ähneln. Bei seinen Tests stellte sich beispielsweise heraus, dass Menschen, die wissen, dass sie gegen einen Bot spielen, diesen aufgrund von Vorurteilen anders behandeln. Vor allem verhandeln sie mit dem Bot anders, als sie es mit einem Menschen machen würden. Weiter führt er an, dass Testspiele gegen Kopien des ursprünglichen Bots nicht sehr hilfreich sind, da weit entwickelte Bots durch bestimmte Nachrichten bereits zu Beginn des Spiels herausfinden können, dass es sich um eine Kopie handelt, um direkt ein Bündnis einzugehen. Der von uns entwickelte *Stragotiator* hätte schon nach den Zügen der ersten Runde feststellen können, dass es sich vermutlich um eine Kopie handelt, indem er die Eröffnungszüge mit seinen eigenen Eröffnungszügen verglichen hätte (vgl. 5.2.2.2.1 auf Seite 60). Weiter erläutert Loeb die strategischen Grundlagen des *Bordeaux Diplomaten*, die wir im späteren Verlauf versuchten zu übernehmen.

5.2.1.1.3 Danny Loeb - Opening By Computer: Spring 1901 Through The Eyes of A Computer Player Danny Loeb stellte in „Opening By Computer: Spring 1901 Through The Eyes of A Computer Player“ (siehe [Loe95f]) fest, dass ein Bot mit sehr großer Wahrscheinlichkeit für ein Land immer den gleichen Eröffnungszug machen würde, da die äußeren Gegebenheiten zu diesem Zeitpunkt immer die gleichen sind. Deshalb plante er für die finale Version des *Bordeaux Diplomaten* eine Datenbank ein, in der vordefinierte Eröffnungszüge stehen würden und aus denen der Bot einen Zug zufällig auswählen würde.

5.2.1.1.4 Danny Loeb - Computer Interpretation of Diplomacy Openings In „Computer Interpretation of Diplomacy Openings“ (siehe [Loe95d]) stellt Danny Loeb verschiedene Eröffnungsstrategien für alle Länder vor. Er erklärt, dass man aus rein taktischer Sicht anhand der Eröffnungszüge der Gegner erkennen kann und mit welchem Gegner eine Allianz sinnvoll ist. Durch diesen Umstand wird die später im *Stragotiator* realisierte Freund/Feind-Matrix motiviert. Weiter erklärt er wie die Freund/Feind-Matrix im *Bordeaux Diplomaten* initialisiert wurde und welche Parameter für ihre Berechnung relevant sind.

5.2.1.1.5 Danny Loeb - The Combinatorics Of Retreats Danny Loeb erörtert in „The Combinatorics Of Retreats“ (vgl. [Loe95c]) die Strategie des *Bordeaux Diplomaten* bezüglich Rückzügen und den mathematischen Hintergrund zu Rückzügen in Diplomacy. Im Gegensatz zu der Anzahl verschiedener Züge in den Bewegungsphasen ist die Anzahl der Rückzüge deutlich kleiner. Während es 4.430.690.040.914.420 verschiedene Eröffnungszüge gibt (vgl. [Loe95a]), lassen sich für die maximal elf möglichen Rückzüge überschaubare 71.680.000 Möglichkeiten finden. Nimmt man die Züge hinzu, bei denen die sich zurückziehende Einheit vernichtet werden, so erhöht sich die Zahl auf 524.880.000. Diese - im Vergleich zur Anzahl an Eröffnungszügen - geringe Zahl resultiert aus der Tatsache, dass es maximal 34 Versorgungszentren geben kann, wodurch sich auch maximal 34 Einheiten im Spiel befinden können. Da für einen Rückzug mindestens 3 Einheiten benötigt werden, kann es auch nur elf verschiedene gleichzeitige Rückzüge geben. Der *Bordeaux Diplomat* berechnet daraufhin bei einem Rückzug alle Möglich-

keiten. Er antizipiert jeden eigenen und gegnerischen Rückzug für Verhandlungen, um so den besten Rückzug zu finden.

5.2.1.1.6 Danny Loeb - The Combinatorics Of Adjustments Die Kernaussage von Danny Loebs „The Combinatorics Of Adjustments“ (vgl. [Loe95b]) ist, dass die Anzahl der Möglichkeiten, Einheiten zu erstellen, mit elf bis 45 Kombinationen noch deutlich unter der Anzahl der möglichen Rückzüge liegt. Aus diesem Grund kann die Berechnung beim Erstellen von neuen Einheiten vollständig alle Möglichkeiten überprüfen.

5.2.1.1.7 Vincent Mous - The Diplomacy Survival Guide Dieser Abschnitt enthält zehn Hinweise (siehe. [Mou95]), die einem Einsteiger beim Spielen von Diplomacy helfen sollen. Da einige dieser Hinweise stark auf menschliches Verhalten abzielen, werden wir sie hier kurz auflisten.

1. Paranoia ist eine hilfreiche Eigenschaft. Versuch dir klarzumachen, dass alle anderen gegen dich spielen, auch wenn es nicht so aussieht.
2. Zu viel Paranoia ist schlecht. Zwar sind alle anderen Mitspieler eigentlich deine Gegner, doch ohne eine Kooperation kannst du nicht gewinnen. Du musst also zumindest am Anfang kooperieren.
3. Verschaffe dir Sicherheit durch eine große Allianz. Wenn du Teil einer Allianz bist und noch einen anderen Spieler als Gegner hast, wirst du normalerweise nicht von deinen Alliierten angegriffen. Fällt dir doch ein Alliierter in den Rücken, so helfen die verbleibenden Alliierten oft bei einem Vergeltungsschlag.
4. Sprich mit jedem. Es ist immer gut, mit jedem anderen Mitspieler in Kontakt zu bleiben. Selbst wenn man im Moment keine Allianz mit einem Mitspieler möchte, so könnte sich das im Verlauf des Spiels ändern.
5. Höre nicht auf zu kommunizieren. Auf diese Weise vermittelst so ein Desinteresse und könntest damit unerwünschte Emotionen verursachen.
6. Intelligenz ist gefährlich. Zeig nicht, dass du klug bist. Wenn du das Spiel beherrscht, dann verheimliche dies gut, denn sobald die Anderen von dir denken, dass du ein guter Spieler bist, wirst du für sie zu einer potentiellen Gefahr.
7. Der frühe Wurm wird gefangen... vom frühen Vogel. Entscheide dich nicht zu früh für eine endgültige Strategie oder mit wem du eine Allianz eingehen willst. Die schlechteste Art ein Spiel zu beginnen, ist jemanden direkt zu Beginn anzugreifen. Man kann dann davon ausgehen, dass der Angegriffene sofort einen Vergeltungsschlag starten wird.
8. Ein Gegner ist gut, drei Gegner sind schlecht. Dein Nachbar wird es merkwürdig finden, wenn du gar keinen Gegner hast und dir vermutlich in den Rücken fallen. Hast du allerdings zu viele Gegner und nicht genug Alliierte, so wirst du nicht lange überleben.
9. Erkundige dich nach guten Eröffnungszügen. Allianzen sollten nicht zufällig gewählt werden, da es Großmächte gibt, für die eine Zusammenarbeit einfacher und effektiver ist.
10. Spiel England. Man hat zwar mit England kaum eine Chance zu gewinnen, man verliert aber auch nicht.

Einige der oben genannten Hinweise flossen direkt in die Entwicklung des *Stragotiators* mit ein. Andere konnten aufgrund der Komplexität nicht berücksichtigt werden.

5.2.1.1.8 Joseph Wheeler - On Diplomacy Joseph Wheeler beschreibt in „On Diplomacy“ (vgl. [Whe97]) die Kriegskunst nach dem preußischen General, Heeresreformer und Militärtheoretiker Carl Philipp Gottlieb von Clausewitz (* 1. Juli 1780, † 16. November 1831). Nach Wheeler ist Krieg ein Akt von Gewalt um den Gegner zu zwingen, das zu tun, was man will. Krieg wird niemals durch sich selbst motiviert. Das Ziel ist politischer Natur und der Krieg ist das Mittel um dies zu erreichen. Das Primärziel (18 Versorgungszentren zu erobern) kann erst im späteren Spielverlauf betrachtet werden. Weiter führt Wheeler aus, dass Kriege nur solange sinnvoll sind, wie sie den Spieler seinem Ziel näher bringen, danach sollten sie nicht weiter geführt werden. Zu Beginn des Spiels sollten die Ziele des Spielers wie folgt gekennzeichnet sein:

- das weitere Überleben sichern
- einen Weg für die zukünftige Expansion sichern

Dabei ist zu bedenken, dass Krieg keine Aktion einer lebendigen Kraft gegen eine leblose Masse ist, sondern immer eine Kollision von zwei lebendigen Kräften. Ein unerfahrener Spieler darf nicht vergessen, dass man andere angreifen und vernichten muss. Er muss sich dabei bewusst sein, dass der Gegner auf die Attacke mit einem Vergeltungsschlag reagieren wird. Außerdem muss beachtet werden, dass Krieg niemals völlig unerwartet ausbricht. Es ist sinnvoll, neben der logischen Motivation des Konflikts auch den Charakter des Gegners zu betrachten. Der Zweck, durch den der Krieg entsteht, muss darüber entscheiden, welche Opfer für ihn zu erbringen sind. Auch wenn Verhandlungen eine große Rolle spielen, darf trotzdem nicht vergessen werden, dass alle Verhandlungen auf der Möglichkeit von Konflikten basieren.

Eine Strategie entscheidet, wann, wo und womit trotz einer Verabredung gekämpft wird. Potentielle Ereignisse sollten dabei aufgrund ihrer Konsequenzen wie reale Ereignisse behandelt werden. Besser ist es allerdings immer, auf feindlichem Territorium zu kämpfen, als auf seinem eigenen.

Seine Stärke im entscheidenden Punkt zu erreichen hängt von der Stärke der Armee und der Geschicklichkeit, mit der die Stärke verteilt wurde, ab. Wheeler empfiehlt die größt mögliche Armee in die Schlacht zu schicken und den entscheidenden Zeitpunkt abzuwarten. Dabei gibt es zwei Faktoren, die für Überraschung sorgen: Geheimhaltung und Geschwindigkeit.

Weiter unterscheidet Wheeler zwischen „gutem“ und „schlechtem“ Verrat. Demnach ist der Bruch eines Abkommen unter einem der folgenden Gesichtspunkten sinnvoll:

- man hat danach 18 Versorgungszentren
- man löscht damit den ehemaligen Alliierten aus
- man schwächt einen ehemaligen Alliierten so stark, dass er keine Gefahr mehr darstellt

Zum Schluss gibt er noch folgende Hinweise:

- alle Einheiten, die zu einem strategischen Zweck zur Verfügung stehen, sollten gleichzeitig eingesetzt werden
- wenn du jemanden schwächen willst, dann tue es hart, schnell und mit allen möglichen Einheiten
- dein Ziel sollte in diesem Fall sein, den Krieg so schnell wie möglich zu beenden und ihn nicht zu lange andauern lassen
- je länger du dich im Krieg befindest, desto größer wird die Wahrscheinlichkeit, dass du von einer anderen Großmacht angegriffen wirst, da du durch den Krieg schon geschwächt und beschäftigt bist

Die von Wheeler genannten Ratschläge wurden im *Strategie Kern* umgesetzt, soweit es die Implementierung zuließ.

5.2.1.1.9 Tom Woodhouse - Diplomatic Tactics In „Diplomatic Tactics“ beschreibt Tom Woodhouse (vgl. [Woo99]) Richtlinien, resultierend aus seinen persönlichen Erfahrungen bezüglich taktischem Verhalten in Diplomacy für Einsteiger. Er empfiehlt, genau wie auch schon Vincent Mous in [Mou95] (vgl. 5.2.1.1.7 auf Seite 44) viel mit den Mitspielern zu kommunizieren, wobei Woodhouse das Ziel der Kommunikation primär im Spielespaß sieht, wogegen Mous eher den Sieg als Ziel favorisiert. Woodhouse führt weiter an, dass die ausgedehnte Kommunikation zu einer Allianz führen kann, während ein eher stilles und zögerliches Verhalten einen Spieler schnell zum Außenseiter macht. Bei allen Arten der Kommunikation soll der Spieler auf höfliche Umgangsformen achten, da die Gegner sich auch von Gefühlen leiten lassen und ein unfreundlicher Ton schnell zu Problemen führen kann. Bei der Wahl der Partner für eine Allianz soll man sich auf die anliegenden Mächte konzentrieren und selber nicht versuchen, zwischen zwei stärkere Mächte zu geraten. Wichtig ist auch, dass man sich nicht nur anguckt, wo der Gegner seine Einheiten stationiert hat, sondern auch, um welche Einheiten es sich handelt, da der Typ der Einheiten den nächsten Schritt stark beeinflusst. Weiter gibt Woodhouse ein paar einfache Regeln für das Schmieden von Allianzen:

- das Verschicken von detaillierten Vorschlägen zeigt dem potentiellen Partner, dass die Vorschläge ernst gemeint sind
- man soll vorsichtig aussprechen, welche Bedingungen an den Allianzpartner gestellt werden
- Strafen für Nichteinhaltung von Verabredungen müssen im Vorhinein geklärt werden
- die Entscheidung für eine Allianz soll möglichst schnell getroffen werden, da Allianzen zur Not auch noch später gebrochen werden können
- um eine Allianz zu schließen, muss dem potentiellen Partner vermittelt werden, warum eine Allianz vorteilhaft ist
- geht es nur um einen Nichtangriffspakt, so muss über vernünftige demilitarisierte Zonen verhandelt werden
- generell sind Allianzen nur solange gut, solange beide Partner dadurch profitieren
- der Koalitionspartner sollte nicht zu schnell und zu oft gewechselt werden, da andere dies merken und das Vertrauen in den Spieler dann sinkt

Zum Schluss gibt Woodhouse noch den Ratschlag, einen ehemaligen Verbündeten komplett und gründlich auszulöschen, sofern man ihm in den Rücken fällt, da dieser nie wieder Vertrauen in einen aufbauen wird.

5.2.1.2 Grundidee

Bevor wir uns um konkreten Ziele kümmerten, überlegten wir, in welche Richtung die Entwicklung des Bots gehen sollte. Wichtig für uns war ein modularer Aufbau, damit alle Mitglieder der Gruppe parallel, ohne die anderen bei ihrer Arbeit zu stören, an ihren Modulen arbeiten konnten. Die Wünsche der Teilnehmer gingen dabei grob in zwei Richtungen. Der Fokus lag zum einen auf dem Kernelement von Diplomacy, der Verhandlung, und zum anderen auf der taktischen Ebene, den Zügen. Auch wurde beschlossen, dass ein bereits vorhandener Bot erweitert werden sollte, da sich die Gruppe lieber um die beschriebenen Optimierungen als um Grundlagen kümmern wollte. Nachdem die Grundideen geklärt waren, folgten die konkreten Ziele.

5.2.1.3 Ziele

Nachdem wir uns zu Beginn des Projektes mit der Literatursuche beschäftigt hatten, stellten wir fest, dass es bereits einen Bot gab, der bezüglich des modularen Aufbaus und der taktischen Fokusierung viele unserer Ideen umsetzte. Der von Danny Loeb Mitte 1990 an der Universität Bodeaux entwickelte *Bordeaux Diplomat* vereint all die taktischen und strategischen Eigenschaften, die unseren Überlegungen vorausgegangen sind (siehe Abschnitte 5.2.1.1.2, 5.2.1.1.3, 5.2.1.1.4, 5.2.1.1.5 und 5.2.1.1.6 ab Seite 43). Durch Danny Loebs Aussage "In combination with an automatic negotiator which will temper the aggressivity of its strategic alter-ego, we should be able to compete at an equal level with humans in the game of Diplomacy" (vgl. [Loe95a]) wurde unser Interesse noch weiter geweckt. Wir bemühten uns darum, den nicht verfügbaren Quellcode des *Bordeaux Diplomaten* zu erlangen, da wir uns vorher entschieden hatten, auf einen bestehenden Bot aufzubauen. Uns waren nur theoretische Ansätze aus den Veröffentlichungen bekannt, jedoch keine praktischen Umsetzungen. Wir wussten daher nicht, in welcher Programmiersprache der *Bordeaux Diplomat* entwickelt wurde. Leider gelang es uns nicht direkt, Kontakt mit Danny Loeb herzustellen, weil er schon lange vor unserem Kontaktversuch die Universität verlassen hatte. Wir versuchten über den Gründer der Internetseite *The Diplomatic Pouch* [1], Manus Hand, Kontakt mit Danny Loeb herzustellen, da Loeb alle Paper über *Diplomacy* auf dieser Seite veröffentlicht hatte. Manus Hand teilte uns mit, dass er schon sehr lange nicht mehr von Danny Loeb gehört hatte, aber versuchen wollte, ihn zu kontaktieren, damit er sich bei uns meldet. Leider hörten wir nach der ersten Antwort nie wieder etwas von Manus Hand.

Da unsere Hauptaufgabe darin bestand, einen menschenähnlichen Bot zu realisieren, mussten wir uns darüber einig werden, welchen Charakter der Bot simulieren sollte. In "What's your Point" (siehe 5.2.1.1.1 auf Seite 42) stellt Winsor die vier Charaktere Klassiker, Lügner, Clubspieler und Romantiker vor. Wir entschieden uns nach einer genauen Betrachtung der Charaktere für den Klassiker, da dieser Charakter einfacher zu modellieren ist als die anderen. Unser Wunsch, den wesentlich spielstärkeren Lügner zu implementieren, musste zurück gestellt werden, weil dessen Realisierung leider ein höheres Verhandlunslevel erfordert, was mehr als die verfügbare Zeit beansprucht hätte.

5.2.2 Aufbau und Realisierung

Im Folgenden wird der konkrete Aufbau des *Stragotiators* beschrieben. Es wird ein Abstraktionsniveau gewählt, welches einen möglichst guten Einblick in die Realisierung der Module *Negotiator*, *Strategie Kern*, *EA-Optimierung* und *Gegnerbewertung* bietet, sich allerdings nicht in zu genauen Details verliert.

5.2.2.1 Negotiator

Das Verhandlungsmodul, bestehend aus der Klasse **NegotiatorPG529**, initiiert und verarbeitet eingehende Verhandlung mit anderen Mächten. Darüber hinaus verwaltet es, in welcher Beziehung der *Stragotiator* zu den anderen Spielern steht. Dies darf nicht mit der Gegnerbewertung (Freund/Feind-Matrix und Intelligenz - siehe Abschnitt 5.2.2.4 auf Seite 67) verwechselt werden. Das Verhandlungsmodul weiß, für welche Macht es spielt, kennt alle eigenen Verhandlungen und prüft, ob sich der jeweilige Verhandlungspartner an diese hält. Das sind Informationen, die der Gegnerbewertung nicht bekannt sind.

Das Verhandlungsmodul bildet auch die Schnittstelle zwischen dem Strategiekern und der Stragotiatorklasse, welche die Schnittstelle zum Netzwerk und somit zum Server

[1] `http://www.diplom.org`

und den anderen Spielern ist. Alle Informationen, die dem Strategiekern zur Verfügung gestellt werden, werden ihm vom Verhandlungsmodul gegeben. Alle Züge, die der Strategiekern berechnet, werden an das Verhandlungsmodul übergeben und von diesem in Verhandlungen umgesetzt bzw. ausgeführt.

Das Verhandlungsmodul beinhaltet verschiedene Parameter, die über eine Textdatei konfiguriert werden. Darunter sind Schwellenwerte für die Freund/Feind-Matrix, die Gegnerintelligenz und die Zugbewertung. Die Freund/Feind-Matrix (siehe Abschnitt 5.2.2.4 auf Seite 67) codiert mit Hilfe reelwertiger Zahlen das Verhältniss zwischen allen Mächten. Diese Darstellung ist sehr unscharf. Das Verhandlunsmodul benötigt allerdings eine scharfe Entscheidung, ob sich ein Spieler einem anderen gegenüber freundlich oder feindlich verhält. Zu diesem Zweck, das Verhalten der anderen Spieler zu bestimmen, existieren zwei Schwellenwerte. Mit Hilfe von diesen wird der Fuzzy Wert der Freund/Feind-Matrix defuzzyfiziert. Liegt ein Wert in der Freund/Feind-Matrix über dem Schwellenwert für freundliches Verhalten, nimmt das Verhandlungsmodul an, dass sich die Mächte freundlich gesonnen sind. Liegt der Wert unter dem Schwellenwert für feindliches Verhalten, nimmt es an, dass sich die Mächte feindlich gesinnt sind. Befindet sich der Wert zwischen beiden Schwellen, ist das Verhältniss der beiden Mächte neutral. Ein weiterer Schwellenwert wird für die Interpretierung der Intelligenz benutzt. Die Gegnerbewertung (siehe Abschnitt 5.2.2.4 auf Seite 67) versucht, die Intelligenz eines Spielers anhand seiner Spielweise zu ermitteln und codiert sie als reelle Zahl. Liegt die Intelligenz eines Mitspielers über diesem Schwellenwert, betrachtet das Verhandlungsmodul diesen Spieler als intelligent. Ab wann ein vorgeschlagener Zug angenommen wird, hängt von einem anderem Parameter ab. Im *Stragotiator* trägt der Strategiekern (siehe Abschnitt 5.2.2.2 auf Seite 59) die Verantwortung für die Spielbewegungen. Er berechnet u.A. Züge für alle eigenen Einheiten und fasst diese zu einem Zugpaket zusammen. Diese Zugpakete haben eine Fitness, die angibt, wie gut ein Zugpaket ist. Der Parameter für die Annahme vorgeschlagener Züge ist ein Prozentwert. Bei einem eingehenden Zugvorschlag wird die Fitness des bisher besten berechneten Zugpaketes mit diesem Prozentwert multipliziert. Daraus resultiert die eigentliche Schranke. Liegt die Bewertung des vorgeschlagenen Zuges über dieser Schranke, wird er als gut angesehen. Die Bewertung eines Zuges obliegt ebenfalls dem Strategiekern (siehe Abschnitt 5.2.2.2.5 auf Seite 65), welcher ebenfalls die Züge für die eigene Macht, die erwähnten Zugpakete, berechnet. Darüber hinaus kann über die Parameterdatei angeben werden, ab wann der Stragotiator von sich aus *Draws*(DRW) anbietet. Ein *Draw* ist ein Gemeinschaftssieg, bei dem nicht ein Spieler alleine sondern mehrere Spieler gemeinsam gewinnen. Der in der Parameterdatei angegebene Wert wird mit einer Zufallszahl $-10 \leq x \geq 10$ addiert. Der so erhaltene Wert gibt an, wie viele Phasen vergehen sollen, bis der *Stragotiator* das erste mal einen Gemeinschaftssieg anbietet. Vorher werden natürlich eingehende Angebote wie unten beschrieben behandelt. Weiterhin lässt sich über die Datei auch festlegen, wie das Vertrauen modelliert werden soll. Es gibt eine abstrakte Klasse **Trust**. Welche konkrete Instanz dieser Klasse verwendet wird, kann per Parameter festgelegt werden. Zur Zeit gibt es jedoch nur eine Klasse, die von **Trust** erbt, **FuzzyTrust**. **FuzzyTrust** modelliert das Vertrauen mittels einer *linguistischen Variable* (Details siehe Abschnitt 5.2.2.1.3 auf Seite 56).

Auch wenn das ursprüngliche Ziel war, Verhandlungslevel 40 (vgl. Abschnitt 4.4.1 auf Seite 21) zu erreichen, wurde in der zur Verfügung stehenden Zeit lediglich Level 20 teilweise implementiert. Zu den nicht unterstützen Befehlen zählen DMZ und SLO. Bei SLO war nicht klar, wie man dieses Kommunikationsmittel vorteilhaft einsetzten kann. Ebenso war nicht bestimmbar, wie das Modell eines Klassikers auf ein solches Angebot zu reagieren hat. Die Behandlung von demilitarisierten Zonen hingegen wäre sowohl hinsichtlich der Modellierung einer menschlichen Spielweise, wie auch aus taktischen Gründen vorteilhaft gewesen, allerdings in der zur Verfügung stehenden Zeit nicht realisierbar. Paul D. Windsor empfiehlt [Win97], nur einen Vertrag auszuhandeln, bei dem

man die Möglichkeit hat, dessen Einhaltung militärisch zu erzwingen. Über eine demilitarisierte Zone sollte nicht verhandelt werden, wenn man die eigenen Truppen aus den umliegenden Provinzen abzieht, weil ein Vertragsbruch durch den Verhandlungspartner nicht geahndet werden könnte. Windsor empfiehlt in einer solcher Situation, die Provinz dem Verhandlungspartner offiziell zu überlassen. Doch ein solches Verhalten ist aufwendig zu implementieren und würde eine Planung über mehrere Runden in die Zukunft voraussetzen.

Bei der Implementierung des Verhandlungsmoduls wurde besonders viel Wert auf die Zusammenarbeit mit anderen Spielern gelegt, wofür unserer Meinung nach XDOs maßgeblich sind. Anschließend stand dann die Implementierung des Verhandlungslevels zehn im Vordergrund.

Ein nur teilweise implementierter Befehl von Level 10 ist NOT. Weil mit ihm jede mögliche Aussage negiert werden kann, ist die Implementierung ebenfalls sehr aufwendig. Außerdem war nicht klar, wie dieses Kommunikationsmittel effektiv eingesetzt werden kann. Es könnte zwar zur Beendigung bestehender Verträge eingesetzt werden, doch der Klassiker, und somit der *Stragotiator*, hält sich an einmal getroffene Abkommen, womit dieser Einsatzzweck nutzlos ist. Weiter könnte, indem ein negiertes Friedensangebot gesendet wird, einer anderen Macht offen der Krieg erklärt werden. Dies erscheint auf den ersten Blick unsinnig, weil die andere Macht vorgewarnt wird, kann aber zumindest bei der so genannten *Chainsaw Diplomacy Taktik* (vgl. [Win98]) eingesetzt werden. Bei einer Chainsaw Nachricht versucht man, seinen Gegenüber einzuschüchtern und zu verunsichern. Er soll denken, dass er es mit einem Verrückten zu tun hat. Doch diese Taktik ist sehr aufwendig zu modellieren und erfordert ein höheres Verhandlungslevel als 20. Aus diesen Gründen wurde entschieden, dass der *Stragotiator* kein NOT versendet.

Bei der Reaktion auf eingehende NOT-Nachrichten wurde lediglich die Negierung von PCE und ALY berücksichtigt. Diese beiden Fälle sind einfach zu interpretieren und zu implementieren. Sie enthalten ein hohes Maß an Informationen, nämlich dass der Sender offensichtlich kriegerische Absichten hegt. Sämtliche weitere Fälle wurden wegen dem großen Aufwand und dem vermutlich seltenen Einsatz im Spiel nicht behandelt. Details zur Reaktion auf NOT befinden sich auf Seite 56.

5.2.2.1.1 Initialisierung von ausgehenden Verhandlungen

XDO
Zu Beginn jeder Runde wird eine neue Instanz des Strategiekerns erstellt. Dadurch soll sichergestellt werden, dass keine alten Informationen aus der vorherigen Runde versehentlich weiter benutzt werden. Der Stratiegern arbeitet nebenläufig, so dass das Verhandlungsmodul zeitnah auf eingehende Verhandlungen reagieren kann und diese bei der Zugberechnung direkt vom Strategiekern berücksichtigt werden können. Für seine Arbeit benötigt der Stategiekern folgende Informationen, welche weiter untern erläutert werden.

- Der Aufrufer (die Instanz des Verhandlungsmodules)
- Eine Prioritätswarteschlange, in der die berechneten Zugpakete gespeichert werden
- Die aktuelle Karte
- Die aktuelle Spielphase, bestehend aus Jahr und Jahreszeit
- Alle möglichen Rückzüge von allen Mächten im Falle einer Rückzugsphase
- Array mit Mächten, mit denen zusammen gearbeitet werden soll

- Array mit Mächten, die nicht angegriffen werden sollen
- Array mit neutralen Mächten
- Array mit Mächten, mit denen Krieg herrscht

Mit allen Mächten, mit denen ein Friedensvertrag oder eine Allianz besteht und denen nicht misstraut wird (siehe Abschnitt 5.2.2.1.3), soll zusammen gearbeitet werden. Sie bilden das erste Array. Ursprünglich sollte der *Stragotiator* nur mit den Mächten zusammen arbeiten, mit denen er alliiert ist. Dies wurde verworfen, weil ebenfalls ursprünglich nur mit Mächten eine Allianz eingegangen wurde, mit denen bereits ein Friedensvertrag bestand und denen vertraut wurde. Weil das Vertrauen in eine Macht nur steigt, wenn mit ihr zusammengearbeitet wird, konnte der *Stragotiator* so niemals mit einer anderen Macht zusammen arbeiten. Im weiteren Verlauf des Projektes wurden zwar die Kriterien, ab wann eine Allianz vorgeschlagen bzw. angenommen wird, leicht abgeschwächt, doch auch mit dieser Anpassung war es noch so, dass selten eine Allianz zustande kam. Deswegen wurde entschieden, dass der *Stragotiator* auch mit den Mächten, mit denen er einen Friedensvertrag geschlossen hat, zusammenarbeitet. Denn das Ziel, den Klassiker zu modellieren, impliziert eine Zusammenarbeit mit anderen Mächten. Außerdem werden Allianzen immer gegen mindestens eine Macht geschlossen. Es sollte jedoch auch die Möglichkeit geben, mit Mächten zusammen zu arbeiten, ohne einen gemeinsamen Gegner zu haben.

Durch das Herausfiltern von Mächten, denen misstraut wird, sollte erreicht werden, dass der *Stragotiator* sich auf die Zusagen seiner Mitspieler, denen er Zugvorschläge unterbreitet, verlassen kann. Hält ein Mitspieler sich öfters nicht an seine Absprachen, sinkt das Vertrauen in ihn (siehe Abschnitt 5.2.2.1.3) und der *Stragotiator* arbeitet in künftigen Spielphasen nicht mehr mit ihm zusammen.

Das Array von Mächten, die nicht angegriffen werden sollen, bestand ursprünglich aus allen Mächten, mit denen ein Friedensvertrag besteht. Um sicherzustellen, dass keine Alliierten angegriffen werden, denen misstraut wird (sie befinden sich nicht im ersten Array), werden alle Mächte, mit denen eine Allianz oder ein Friedensvertrag besteht, in diesem Array zusammengefasst. Die eigene Macht muss aus diesem Array jedoch herausgenommen werden, weil der Strategiekern sonst nicht auf Versorgungszentren zieht, die dieser Macht gehören. Es sollte aber möglich sein, dass der *Stragotiator* seine eigenen Versorgungszentren verteidigt.

Das Array mit Mächten, mit denen Krieg herrscht, besteht aus den Mächten, die den *Stragotiator* angegriffen, ein Friedensangebot abgelehnt, ein NOT PCE oder ein NOT ALY gesendet haben oder gegen die der *Stragotiator* eine Allianz akzeptiert hat.

Sämtliche anderen Mächte werden als neutral eingestuft und befinden sich in dem dafür vorgesehen Array.

Hat der Strategiekern ein neues Zugpaket berechnet, welches besser ist als die bisher berechneten, wird dem Verhandlungsmodul mitgeteilt, dass es neue Zugvorschläge gibt. Dieses betrachtet dann alle Züge in dem Paket. Wenn ein Zug für eine Einheit dabei ist, die nicht der eigenen Macht gehört, wird überprüft, ob er eine Fitness größer gleich null hat. Das ist notwendig, weil der Strategiekern häufig *HOLDs* berechnet, die nur mangels einer vernünftigen Zugmöglichkeit zustande kommen. Weil diese Notlösungen, die eine festgelegte Fitness von -100 haben, weder der eigenen noch der anderen Macht einen Vorteil bringen, werden diese auch nicht in eine Verhandlung umgesetzt. Sie hätten sogar den ungewollten Effekt, falls die andere Macht auch von einem *Stragotiator* gespielt wird, dieser dem Sender aufgrund der schlechten Zugvorschläge irgendwann misstraut. Die Fitnesswerte von Zügen sind reelle Zahlen. Ein negativer Wert deutet dabei an, dass mit diesem Zug etwas nicht stimmt - er z.B. eine Notlösung ist.

Alle Züge für fremde Einheiten mit einer Fitness größer gleich null werden zusätzlich

überprüft, ob sie schon einmal vorgeschlagen wurden. Das Verhandlungsmodul verwaltet drei Mengen von Zügen. Eine für alle Züge, die ausgeführt werden müssen (wurden früher verhandelt und angenommen), eine für die, die nicht ausgeführt werden dürfen (wurden früher verhandelt und abgelehnt), und eine für die, die gerade verhandelt werden (auf ein bereits gesendetes XDO hat der *Stragotiator* noch keine Antwort erhalten). Befindet sich der Zug in keiner dieser drei Mengen, wird er durch das Senden eines XDOs dem Mitspieler vorgeschlagen und der Menge der unter Verhandlung stehenden Züge hinzugefügt. Das keine Züge mehrfach vorgeschlagen werden, führt zu einem menschlicheren Verhalten. Denn ein Mensch würde einen Zug, den ein Mitspieler schon zugesichert hat, nicht mehrfach vorschlagen.

Alle Züge für eigene Einheiten aus dem Zugpaket werden ausgeführt. Um dies dem Server mitzuteilen, wird eine SUB-Nachricht gesendet.

Die vorgeschlagenen Züge bleiben so lange in der Menge der unter Verhandlung stehenden Züge, bis der *Stragotiator* eine Antwort erhält. Wurde der Vorschlag von der anderen Macht angenommen, wird der Zug aus der Verhandlungsmenge entfernt und in der Menge für Züge, die auf jeden Fall gemacht werden müssen, ergänzt. Diese aktualisierte Menge wird dem Strategiekern mitgeteilt, damit alle enthaltenen Züge bei weiteren Berechnungen berücksichtigt werden, denn der *Stragotiator* geht davon aus, dass zugesicherte Züge von den anderen Mächten ausgeführt werden.

Wenn die andere Macht den Vorschlag nicht annimmt, wird er ebenfalls aus der Verhandlungmenge gelöscht, jedoch anschließend der Menge von Zügen, die nicht gemacht werden dürfen, hinzugefügt. Auch diese Menge wird dem Strategiekern übergeben.

PCE
Für *Diplomacy* existiert eine Statistik , welche Bündnisse besonders spielstark sind (vgl. [Mou95]). Daraus leiten sich für eine Macht die folgenden Bündnispartner ab:

Macht	Bevorzugte Bündnispartner
AUS	ITA, RUS
ENG	GER, TUR
FRA	ENG, RUS
GER	ITA, FRA
ITA	TUR, GER
RUS	ENG, ITA
TUR	RUS, FRA

Tabelle 5.9: Bevorzugte Bündnispartner für eine bestimmte Macht

In der ersten Runde wird auf Grundlage dieser Tabelle, abhängig davon welche Macht der *Stragotiator* spielt, an zwei Mächte ein Friedensangebot gesendet. Spielt der *Stragotiator* zum Beispielt Österreich-Ungarn (AUS), dann wird im Frühling 1901 an Italien (ITA) und Russland (RUS) jeweils ein Friedensangebot gesendet. Es wird nicht direkt eine Allianz angeboten, weil man laut Tim Miller (vgl. [Mil98b]) niemals Verhandlungen mit einer Allianz anfangen sollte. Man sollte immer zuerst über kleinere Dinge verhandeln. Wenn die Verhandlungen gut laufen, dann wird sich später von alleine eine Allianz ergeben. Deswegen sendet der *Stragotiator* am Anfang Friedensangebote.

Im weiteren Verlauf des Spieles wird am Anfang jeder Bewegungsphase - Frühling und Herbst - für jede andere Macht überprüft, ob Krieg herrscht, kein Friedensvertrag besteht, der Wert der Freund/Feind-Matrix (siehe Abschnitt 5.2.2.4 auf Seite 67) über dem Schwellenwert für freundliches Verhalten liegt, sie intelligent ist (siehe Abschnitt 5.2.2.4 auf Seite 68) und ihr nicht misstraut wird. Wenn das der Fall ist, wird ihr ein Friedensangebot unterbreitet.

Die Überprüfung der Intelligenz beruht auf der Aussage von Danny Loeb (vgl. [Loe95a]), dass ein Mensch lieber mit einem Mitspieler zusammen spielt, den er für intelligent hält. Bei uns wurde auch die Idee diskutiert, dass ein Bündnis mit einem dummen Spieler vorteilhaft sein könnte, weil man ihn ausnutzen kann. Doch ein Klassiker, den wir modellieren wollten, hat immer das Wohl der gesamten Allianz im Auge und nicht nur sein eigenes. Ein Ausnutzen eines Mitspielers steht unserer Meinung nach im Widerspruch zu diesem Verhalten. Zum anderen ist das Ausnutzen eines dummen Spielers nach Paul D. Windsor (vgl. [Win97]) gefährlich, denn nicht jeder Spieler ist vollständig dumm. Irgendwann wird jeder erkennen, dass er ausgenutzt wird und denjenigen angreifen, der ihn übervorteilt.

Das Mächten, denen misstraut wird, kein Friedensangebot unterbreitet wird, soll bewirken, dass sich der *Stragotiator* auf einen eventuell zustande kommenden Friedensvertrag verlassen kann. Wenn einer Macht misstraut wird, kann das u.A. daran liegen, dass sie öfters Absprachen nicht eingehalten hat. Ein Friedensvertrag wäre in so einem Fall wertlos.

Mächte, mit denen bereits ein Friedensabkommen besteht, sollten nicht jede Runde sondern in unregelmäßigen Abständen an das Abkommen erinnert werden. Zu diesem Zweck gibt es eine Zählvariable. Sie wird in jeder Bewegungsphase dekrementiert. Wenn sie den Wert null erreicht hat, wird an alle Mächte, mit denen ein Friedensvertrag besteht, eine neue Friedensnachricht gesendet. Nach Paul D. Windsor (vgl. [Win97]) soll man vollständige und detaillierte Regeln aufstellen - was darf man alles tun, was nicht, welche Folgen gibt es und welchen Vorteil birgt das Abkommen -, damit die Bündnispartner nicht Zeit darin investieren, sich Gedanken zu machen, wie sie den bestehenden Vertrag möglichst geschickt brechen können. Durch die beschränkten Kommunikationsmöglichkeiten kann man für einen Friedensvertrag keine weiteren Regeln aufstellen. Weil aber durch das Vergessen eines Abkommens dieses zwangsläufig für diese Partei nicht mehr vollständig definiert ist, versuchen wir, durch Erinnerungen der empfohlenen Vollständigkeit über lange Sicht nahe zu kommen. Außerdem wird laut Tim Miller (vgl. [Mil98a]) die Wahrscheinlichkeit gesenkt, dass eine Macht einen Vertrag ohne Vorwarnung bricht, wenn man ihr signalisiert, dass man offen für Verhandlungen und Gespräche ist. Nach dem Senden der Nachricht wird die Zählvariable wieder zufällig zwischen null und fünf gesetzt. So wird erreicht, dass die Erinnerung an den Frieden in unregelmäßigen Abständen erfolgt.

Erhält das Verhandlungsmodul eine Antwort auf eine von ihm gesendete Friedensnachricht, wird unterschieden, ob es akzeptiert oder abgelehnt wurde. Wurde es angenommen, wird sich gemerkt, dass mit der entsprechenden Macht ab jetzt ein Friedensvertrag existiert. Wurde das Friedensangebot abgelehnt, muss das zwangsläufig bedeuten, dass die Macht dem *Stragotiator* feindlich gesinnt ist. Das Verhandlungsmodul merkt sich diesen Sachverhalt entsprechend und eventuell bestehende Allianzen oder Friedensverträge sind hinfällig.

ALY

Ähnlich wie bei Friedensangeboten wird am Anfang jeder Bewegungsphase - Frühling und Herbst - für jede andere Macht überprüft, ob mit ihr bereits ein Friedensvertrag besteht, sie sich laut Freund/Feind-Matrix (siehe Abschnitt 5.2.2.4 auf Seite 67) friedlich verhält, sie intelligent ist (siehe Abschnitt 5.2.2.4 auf Seite 68) und ihr nicht misstraut wird. Wenn das der Fall ist, dann ist die Macht ein potenzieller Allianzpartner.

Die Gründe für die Überprüfungen der Intelligenz und des Vertrauens sind dieselben wie bei Friedensangeboten. Dass nur Mächte, mit denen schon ein Friedensvertrag geschlossen wurde, als Allianzpartner in Frage kommen, beruht darauf, dass man sich in einer Allianz darauf verlassen muss, dass der Partner einen nicht angreift.

Ursprünglich war geplant, dass einer Macht vertraut werden muss, bevor der *Strago-*

tiator mit ihr eine Allianz eingeht. Das wurde so abgeschwächt, dass dieser Macht nur nicht misstraut werden darf, weil die Anzahl an eingegangen Allianzen bei Testspielen sehr gering war. Zudem gibt es in der aktuellen Implementierung keinen Unterschied, ob es mit einer Macht einen Friedensvertrag oder eine Allianz gibt (siehe 5.2.2.1.1), weswegen in diesem Punkt eine Unterscheidung der Voraussetzungen für Frieden und Allianz unsinnig wäre.

Wurde ein potenzieller Allianzpartner gefunden, wird überprüft, ob es eine Macht gibt, gegen die man eine Allianz eingehen kann. Dazu wird für jede Macht, mit der sich der *Stragotiator* schon im Krieg befindet, geprüft, ob dieser Feind eine Einheit besitzt, die maximal zwei Provinzen von einer Einheit des potenziellen Bündnispartners entfernt ist. Für jede solche Macht wird dem potentiellen Alliierten dann eine Allianz gegen den ermittelten Feind vorgeschlagen.

Wird eine vorgeschlagene Allianz angenommen, merkt sich das Verhandlungsmodul für die entsprechende Macht, dass mit ihr eine Allianz besteht. Bei einer Ablehnung wird nichts gemacht. Denn das Ablehnen einer Allianz kann mehrere Gründe haben. So kann die ablehnende Macht schon ein Bündnis mit der Macht, gegen die die Allianz vorgeschlagen wurde, haben. Die Ablehnung muss nichts mit dem *Stragotiator* zu tun haben.

DRW
Im Verhandlungsmodul gibt es eine Zählvariable, anhand der entschieden wird, ab welcher Runde Draws vorgeschlagen werden. Am Anfang wird der Wert aus der Parameterdatei ausgelesen und zufällig um maximal zehn erhöht oder verringert. So soll erreicht werden, dass der Bot nicht immer ab dem selben Jahr anfängt, Draws vorzuschlagen. Weil der Klassiker, der modelliert werden sollte, nicht auf einen Solosieg hinarbeitet, sondern gerne mit anderen Spielern zusammen gewinnt, stellt sich nicht die Frage, durch welchen Spielverlauf der *Stragotiator* zu einem Unentschieden bewegt wird. Lediglich die Mächte, mit denen er ein Unentschieden akzeptiert, sind vom Spielverlauf abhängig.

Die oben erwähnte Zählvariable wird am Anfang jeder Runde dekrementiert. Sobald sie den Wert Null erreicht hat, wird am Anfang jeder Bewegungsphase - Frühling und Herbst - die Bereitschaft zu einem Draw an den Server gesendet und anderen Mächten ein Draw vorgeschlagen. Dabei wird unterschieden, ob es sich bei dem Spiel um ein PDA-Spiel[2] handelt oder nicht. Für den Fall, dass es kein PDA-Spiel ist, wird dem Server einfach nur die Bereitschaft zu einem Unentschieden mitgeteilt und jeder noch im Spiel befindlichen Macht, mit der ein Friedensvertrag besteht, einzeln ein Unentschieden vorgeschlagen. Im Fall eines PDA-Spiels wird eine Liste mit allen verbleibenden Mächten, mit denen ein Friedensvertrag existiert, einschließlich der eigenen Macht, gebildet. Dem Server wird dann mitgeteilt, dass man ein Unentschieden mit diesen Mächten eingehen möchte. Anschließend wird dann eine DRW-Nachricht an alle Mächte aus dieser Liste exklusiv der eigenen Macht gesendet.

Auf die Antworten der anderen Mächte auf eine DRW-Nachricht reagiert das Verhandlungsmodul in der aktuellen Version nicht. Wegen fehlender Implementierungszeit werden diese nicht analysiert und weiter behandelt. Es wäre jedoch denkbar, dass das Modul sich merkt, wer eine bestimmte DRW-Nachricht ablehnt, um die Liste der Mächte, mit denen ein Unentschieden akzeptiert wird, in den folgenden Runden anders aufzubauen.

5.2.2.1.2 Behandlung von eingehenden Verhandlungen

[2] PDA steht für PPartial draws are allowedünd bedeutet, dass die Spieler genau festlegen können, mit wem sie einen Gemeinschaftssieg eingehen möchten. In einem nicht PDA-Spiel kann ein Spieler nur angeben, ob er an einem Gemeinschaftssieg teilnehmen möchte, nicht aber, mit welchen Mächten.

XDOs
Erhält das Verhandlungsmodul ein eingehendes Zugangebot, wird zuerst überprüft, ob der Zug eine eigene Einheit betrifft. Ist das nicht der Fall, dann wird eine BWX-Nachricht gesendet um dem Sender mitzuteilen, dass den *Stragotiator* die Nachricht nichts angeht. Ist der vorgeschlagene Zug ein Zug für eine eigene Einheit, wird zuerst geprüft, ob der Zug in der Menge der Züge, die auf jeden Fall ausgeführt werden müssen bzw. nicht ausgeführt werden dürfen, enthalten ist. Ist das der Fall, wird der Zug angenommen bzw. abgelehnt.

Danach wird überprüft, ob sich der vorgeschlagene Zug mit einem der Züge, die unter Verhandlung stehen bzw. die gemacht werden müssen, in Konflikt befindet. Ein Konflikt zwischen zwei Zügen besteht dann, wenn die selbe Einheit verschiedene Befehle ausführen soll oder wenn zwei Einheiten in die selbe Provinz geordert werden. Tritt ein solcher Konflikt auf, wird der Zugvorschlag abgelehnt. Diese Abfrage ist notwendig, weil ein angenommener Zugvorschlag in die Menge der auszuführenden Züge aufgenommen wird. Bevor diese Überprüfung eingefügt wurde, konnte es passieren, dass Züge ausgeführt wurden, die sich gegenseitig behinderten.

Anschließend wird überprüft, ob durch die Ausführung des Zuges ein Vertrag gebrochen würde. Dazu wird, falls es sich bei dem Zugvorschlag um eine Bewegung, eine Bewegungsunterstützung, einen Konvoi, eine Bewegung per Konvoi oder einen Rückzug handelt, der Zug vom Strategiekern bewertet. Ist die Bewertung negativ, ist das ein Indiz dafür, dass mit dem Zug ein Alliierter angegriffen oder ein Friedensvertrag gebrochen würde. Der Vorschlag wird abgelehnt.

Tritt keiner der eben genannten Fälle ein, kommt es zur eigentlichen Bewertung des Vorschlages. Hierfür wird zuerst das Vertrauen in den Sender überprüft. Wird ihm misstraut, wird der Vorschlag mit einer Wahrscheinlichkeit von $\frac{3}{4}$ abgelehnt. Hat der *Stragotiator* vollstes Vertrauen zu ihm, wird der Vorschlag mit einer Wahrscheinlichkeit von $\frac{3}{4}$ angenommen. Sonst wird der vorgeschlagene Zug vom Strategiekern bewertet. Damit ein Zug angenommen wird, muss seine Bewertung einen bestimmen Prozentanteil des bisher besten berechneten Zugpaketes übersteigen. Der Prozentanteil wird über eine Parameterdatei bestimmt. Wird ein Zug gut bewertet, steigt das Vertrauen in den Sender. Im anderen Fall sinkt es.

Das Einfließen des Vertrauens beruht auf einer Aussage von Danny Loeb (vgl. [Loe95a]). Er sagt, dass wenn das Vertrauen in einen Mitspieler groß genug ist, kann es sein, dass ein Spieler auch Vorschläge umsetzt, die weniger seinem, sondern mehr im Interesse des Beratenden sind. Im Gegensatz, wenn man kein Vertrauen mehr in einen Spieler hat, wird ein Mensch kaum auf Vorschläge hören. Im extremen Fall kann er seine normalen Präferenzen ignorieren und den Sieg einem dritten Spieler „schenken“.

Alle angenommen Zugvorschläge werden der Menge von Zügen, die gemacht werden müssen, hinzugefügt. So wird sichergestellt, dass der *Stragotiator* sein Wort hält. Das wird gemacht, weil der Klassiker seine Bündnispartner nicht anlügt. Es wäre jedoch möglich, das Modell so zu erweitern, dass Spieler, mit denen kein Friedensvertrag oder eine Allianz besteht, angelogen werden. Das würde insbesondere Sinn ergeben, wenn mit dem Spieler Krieg herrscht.

Die abgelehnten Zugvorschläge werden nicht der Menge von Zügen, die nicht gemacht werden dürfen, hinzugefügt. Sonst könnte es passieren, dass ein Zug abgelehnt wird, weil er schlecht bewertet wurde, obwohl es keinen besseren für die betreffende Einheit gibt. Würde die Ausführung des Zuges durch das Hinzufügen in die Menge verhindert, würde der Strategiekern gezwungen, schlechtere Züge als möglich wären zu berechnen. Im schlimmsten Fall würde es dazu führen, dass für eine Einheit kein Zug erlaubt würde.

PCE
Erhält das Verhandlungsmodul ein Friedensangebot von einer anderen Macht, wird

zuerst überprüft, ob der Sender in der Liste von Mächten, zwischen denen Frieden angeboten wird, enthalten ist. Diese Idee wurde vom Diplominator (vgl. [WCW+08]) (näheres siehe Abschnitt 4.5.7 auf Seite 25) übernommen. So soll verhindert werden, dass der *Stragotiator* einem böswillig inszenierten Friedensangebot aufsitzt. Schickt eine Macht ein Friedensangebot an dem sie selber nicht beteiligt ist, wird das Angebot, weil der *Stragotiator* ehrlich ist, abgelehnt und das Vertrauen in die betreffende Macht verringert.

Bei einem normalen Friedensangebot, bei dem der Sender in der Liste der Mächte, zwischen denen Frieden angeboten wird, enthalten ist, wird für jede Macht aus der Liste geprüft, ob eine der folgenden Bedingen zutrifft:

- Es existiert bereits ein Friedensvertrag oder eine Allianz
- Die Macht ist intelligent, verhält sich freundlich, ihr wird nicht misstraut und sie führt keinen Krieg gegen den *Stragotiator*

Erfüllt jede Macht aus der Liste eine der beiden Bedingungen, wird das Angebot angenommen und das Verhandlungsmodul merkt sich, dass mit den Mächten ein Friedensvertrag besteht. Gibt es mindestens eine Macht, die beide Bedingung nicht erfüllt, wird das Angebot abgelehnt.

ALY
Bei einem eingehenden Allianzangebot wird, wie bei eingehenden Friedensangeboten, zuerst überprüft, ob der Sender in der Liste der Mächte, zwischen denen eine Allianz angeboten wird, enthalten ist. Wie bei einem Friedensangebot wird die Allianz, wenn der Sender nicht in der Liste enthalten ist, abgelehnt und das Vertrauen in den Sender verringert.

Ist der Sender enthalten, werden zwei Sachverhalte geprüft

- Wird mit allen angebotenen Allianzpartner eine Allianz gewollt?
- Wird mit allen Mächten, gegen die die Allianz angeboten wird, Krieg gewollt?

Trifft beides zu, wird die Allianz angenommen und das Verhandlungsmodul merkt sich für die entsprechenden Mächte, dass mit ihnen eine Allianz besteht bzw. dass wir mit ihr im Krieg sind. Sonst wird das Angebot abgelehnt.
Mit einer Macht wird genau dann eine Allianz gewollt, wenn mit ihr bereits eine Allianz oder ein Friedensvertrag besteht oder sie intelligent ist, sie sich laut Freund/Feind-Matrix uns gegenüber friedlich verhält, der *Stragotiator* mit ihr nicht im Krieg ist und ihr nicht misstraut.
Mit einer Macht wird Krieg gewollt, wenn mit ihr bereits Krieg herrscht oder mit ihr weder ein Friedensvertrag noch eine Allianz besteht und sie sich laut Freund/Feind-Matrix uns gegenüber feindlich verhält.

DRW
Wird dem *Stragotiator* ein Unentschieden angeboten, wird eine Liste erstellt, in die der Sender, alle Empfänger und gegebenenfalls alle weiteren Mächte, die an dem Draw teilnehmen sollen, aufgenommen werden. Anschließend wird geprüft, ob er sich mit einer dieser Mächte im Krieg befindet oder ihr misstraut. Sollte das für eine Macht der Fall sein, wird die Anfrage abgelehnt. Sonst wird das Angebot angenommen und dem Server mitgeteilt, dass der *Stragotiator* mit diesen Mächten ein Unentschieden eingehen möchte.

NOT
Aus Zeitgründen wurde die Behandlung von eingehenden NOT-Nachrichten nur teilweise implementiert. Lediglich auf NOT PCE und NOT ALY wird reagiert, indem das

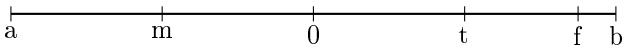


Abbildung 13: Bereichsunterteilung des Vertrauens

Verhandlungsmodul sich merkt, dass die betreffende Macht weder Frieden noch eine Allianz mit dem *Stragotiator* möchte und von nun an mit ihr Krieg herrscht. Der Grund dafür ist, sollte ein bestehender Vertrag aufgelöst werden, kann man davon ausgehen, dass der Initiator der Auflösung von nun an genau das Gegenteil möchte. In beiden Fällen ist das Krieg. Bestand vorher kein Friedensvertrag oder eine Allianz, dann kann solch eine Nachricht als offenkundige Kriegserklärung gewertet werden. Anschließend wird der Erhalt der Nachricht durch das Senden eines YES bestätigt.

Alle anderen NOT-Nachrichten sind nicht implementiert, und dem Sender wird dies durch Senden einer HUH-Nachricht mitgeteilt.

5.2.2.1.3 Vertrauensmodell

Das Vertrauensmodell ist ein wichtiger Bestandteil des Verhandlungsmodules und bildet das Vertrauen, dass ein menschlicher Spieler in seine Mitspieler hat, nach. Der Grund, dass es implementiert wurde, ist folgende Aussage von Danny Loeb (vgl. [Loe95a]) , wie das Vertrauen das Verhalten eines Menschen bei XDOs beeinflusst. Nach seiner Aussage werden Vorschläge von Spielern, denen ein Mensch sehr vertraut, berücksichtigt, ohne dass viel darüber nachgedacht wird. Während bei Vorschlägen von Spielern, die nicht vertrauensvoll sind, genau das Gegenteil gemacht wird.

Die erste Idee war es, das Vertrauen als Gleitkommazahl zu speichern. Diese sollte sich nur in einem definierten Bereich $[a, b]$ mit $a < 0$ und $b > 0$ bewegen dürfen. In diesem Bereich sollte es drei Schranken $m < 0$ (mistrust), $t > 0$ (trust) und $f > 0$ (fulltrust) geben (siehe Abbildung 13). Die so entstehenden Bereiche hatten folgende Bedeutungen:

a-m: Dem Spieler wird misstraut

m-t: Dieser Bereich ist neutral. Es wird jedoch zwischen positiven und negativen Werten unterschieden, um Tendenzen zu codieren. Der Bot wäre sich also nicht sicher gewesen, wie er den Gegner einschätzen soll, hätte aber zu Misstrauen bzw. Vertrauen tendiert.

t-f: Dem Spieler wird vertraut

f-b: Dem Spieler wird vollständig vertraut

In der späteren Implementierung wurde dieses Modell so nicht umgesetzt. Nach genauer Betrachtung wurde deutlich, dass man die verschiedenen Bereiche durch *Fuzzy Mengen* darstellen kann. Deswegen wurde das Vertrauen letztendlich als *linguistische Variable* implementiert. Mit einer *linguistischen Variablen* lässt sich im einfachsten Fall eine wie oben beschriebene Unterteilung von Bereichen darstellen. Sie ist jedoch wesentlich flexibler als die einfache Unterteilung und bietet die Möglichkeit eine Entscheidungsfindung durch einen *Fuzzycontroller* zu realisieren. Da die Implementierung eines *Fuzzycontrollers* in Java sehr aufwendig ist, wurde diese Möglichkeit nicht weiter verfolgt. Stattdessen wurde mit Hilfe von vier trapezförmigen *Fuzzy Mengen* eine Unterteilung eines zu definierenden Bereiches realisiert, wobei die Unterscheidung zwischen positiven und negativen Werten im neutralen Bereich zur Vereinfachung weggefallen ist (Beispiele siehe Abbildung 15-19 auf Seite 69ff). Der Bereich, in dem die Vertrauenswerte sich bewegen dürfen, und die vier *Fuzzy Mengen* sind über eine Parameterdatei einstellbar. Der Vertrauenswert, der für jeden Gegner separat gespeichert wird, kann während eines

Spieles steigen oder fallen. Dazu wird eine, in der Parameterdatei angegebene Konstante auf den Vertrauenswert addiert bzw. von ihm subtrahiert. Die Schrittweiten für das In- und Dekrementieren können unterschiedlich groß gewählt werden. Um zu entscheiden, ob einem Spieler vertraut wird oder nicht, wird die Zugehörigkeit des entsprechenden Vertrauenswertes zu allen *Fuzzy Mengen* bestimmt. Zu welcher *Fuzzy Menge* der Wert am meisten gehört, bestimmt, ob der Macht vertraut wird oder nicht. Steigt oder sinkt ein Vertrauenswert über eine Bereichsschranke der *linguistischen Variable* hinaus, wird er auf die entsprechende Schranke gesetzt.

Der Vertrauenswert für einen Mitspieler wird in folgenden Fällen inkrementiert:

- Wenn ein zugesagter Zug ausgeführt wurde.
- Wenn ein vorgeschlagener Zug vom Strategiekern gut bewertet wird.

Er wird dekrementiert, wenn einer der folgenden Fälle auftritt.:

- Wenn ein zugesagter Zug nicht ausgeführt wurde (dreimaliges dekrementierten, weil dies besonders stark bestraft werden soll).
- Wenn ein vorgeschlagener Zug vom Strategiekern schlecht bewertet wird.
- Wenn der Sender bei einem Allianzangebot nicht in der Liste der Mächte enthalten ist, zwischen denen die Allianz angeboten wird.
- Wenn der Sender bei einem Friedensangebot nicht in der Liste der Mächte enthalten ist, zwischen denen Frieden angeboten wird.

5.2.2.1.4 Kontrolle der ausgeführten Züge

Ein wichtiger Punkt ist die Kontrolle der ausgeführten Züge. Es muss überprüft werden, ob sich andere Mächte an Absprachen halten und ob man selbst von einer anderen Macht angegriffen wurde. Am Ende einer jeden Runde wird, damit die Spieler den Spielverlauf nachvollziehen können, für jeden Zug eine ORD-Nachricht gesendet. In dieser wird genau angegeben, welche Spieler welche Züge in der Runde ausgeführt haben und ob diese erfolgreich waren und, falls nicht, warum sie fehlschlugen.

Diese Nachrichten werden vom Verhandlungsmodul zuerst darauf überprüft, ob es sich um einen Zug handelt, der von einer anderen Macht zugesichert wurde. Wenn das der Fall ist, wird der Zug als ausgeführt markiert. Diese Markierung ist notwendig, weil zu Beginn der nächsten Runde die Liste mit den zugesicherten Zügen durchlaufen und überprüft wird, ob jeder Zug auch wirklich ausgeführt wurde. Sollte eine andere Macht einen Zug durch die Annahme eines XDOs zugesichert, sich aber nicht an diese Absprache gehalten haben, sinkt das Vertrauen in diese Macht um drei Schritte. Die Schrittweite wird, wie auf Seite 56 beschrieben, über eine Parameterdatei festgelegt. Dass bei einem nicht ausgeführten Zug das Vertrauen um drei Schritte sinkt liegt daran, dass ein Vertragsbruch stärker bestraft werden soll als ein schlechter Zugvorschlag. Weil es aber möglich ist, dass der andere Spieler lediglich vergessen hat den Zug anzugeben, soll ihm nicht gleich misstraut werden. Wenn ein zugesicherter Zug wie erwartet ausgeführt wurde, steigt das Vertrauen in die Macht. Weil der Klassiker modelliert wurde, der sich an seine Absprachen hält, ist davon auszugehen, dass er dieses Verhalten auch von seinen Verbündeten erwartet. Darum wird dieses Verhalten nicht, wie bei der Bestrafung nicht ausgeführter Züge, durch dreifaches inkrementieren des Vertrauenswertes belohnt.

Ein im Voraus abgesprochener Zug wird nicht auf einen Angriff überprüft, weil das Verhandlungsmodul diesem Zug vorher schon zugestimmt hat und somit eine Einheit, die von einer Bewegung angegriffen worden wäre, aus der Zielprovinz abgezogen werden konnte.

Wenn der Zug nicht vorher abgesprochen war und es sich um eine Bewegung, einen Konvoi, eine Bewegung via Konvoi oder eine Bewegungsunterstützung handelt, wird überprüft, ob er einen Angriff auf die eigene Macht darstellt. Alle anderen Züge, z.B. ein *HOLD*, können niemals ein Angriff sein. Wurde dem Strategiekern die Macht, die den Zug ausgeführt hat, so übergeben, dass mit ihr zusammen gearbeitet werden sollte, kann der Strategiekern am besten entscheiden, ob der Zug einen Angriff darstellt oder nicht, weil dieser alle ausgeführten Züge berücksichtigen kann. Weil der Strategiekern nicht weiß, für welche Macht er spielt, kann er nur feststellen, ob es ein Angriff auf eine der Mächte ist, für die er Züge berechnen sollte. Das umfasst die eigene Macht und die Mächte, mit denen zusammen gearbeitet werden sollte. Stellt der Strategiekern fest, dass ein solcher Angriff vorliegt, muss das Verhandlungsmodul noch überprüfen, ob die eigene Macht angegriffen wurde. Dazu wird geprüft, ob zu Beginn der Runde eine eigene Einheit auf der Zielprovinz stand oder ob die Zielprovinz ein Versorgungszentrum ist, welches der eigenen Macht gehört. Trifft einer dieser Fälle zu, wird der Zug als Angriff gewertet und sich gemerkt, dass mit dem Angreifer zukünftig Krieg herrscht. Sollte vorher ein Friedensvertrag oder eine Allianz mit dem Angreifer bestanden haben, wird zusätzlich dieser Macht (in Zukunft) misstraut.

Wenn mit der Macht, die den Zug ausgeführt hat, nicht zusammen gearbeitet werden sollte, wird nur überprüft, ob am Anfang der Runde eine eigene Einheit auf der Zielprovinz stand oder ob die Zielprovinz ein eigenes Versorgungszentrum ist. Bei einem dieser Fälle wird wie im Fall, dass mit der Macht zusammen gearbeitet werden sollte, sich gemerkt, dass mit dem Angreifer ab jetzt Krieg herrscht und ihm eventuell misstraut. Auch für den Fall, dass die Einheit, die auf der Zielprovinz stand, in eine andere Provinz geordert wurde, konnte die andere Macht das nicht wissen und musste somit einen Angriff geplant haben.

Am Anfang fand bei jedem ausgeführten Zug nur eine Überprüfung statt wie in dem Fall, dass mit der anderen Macht nicht zusammen gearbeitet werden sollte. Dies führte dazu, dass jeder Kettenzug, bei dem eine Einheit in eine Provinz zieht aus der eine andere Einheit weggezogen wird, als Angriff gewertet wurde. Dieses ungewollte Verhalten führte dazu, dass zwei *Stragotiatoren*, die eigentlich zusammen arbeiten sollten, sich sehr schnell gegenseitig bekriegt haben.

5.2.2.2 Strategie Kern

Diplomacy basiert auf den zwei Basiselementen Verhandlung und Taktik, wobei nicht abgestritten werden kann, dass beide Elemente für ein erfolgreiches Spiel gut zusammenarbeiten müssen. Der *Strategie Kern* ist dabei für die taktische Komponente zuständig. In seinen Papern geht Danny Loeb näher auf die Probleme in Diplomacy und den Versuch ein, einen spielstarken Bot zu erschaffen (siehe 5.2.1.1.2, 5.2.1.1.3, 5.2.1.1.4, 5.2.1.1.5 und 5.2.1.1.6). Mitgenommen haben wir aus diesen Informationen verschiedene Aspekte, die wir später realisiert haben. Wichtigster dieser Aspekte war die Separation der Strategie von der Verhandlung: Der *Strategie Kern* weiß nicht, für welche Nation er spielt. Er kann nicht beurteilen, welchem Gegner der Spieler vertraut oder wem er misstraut. Er kennt die Verhandlungen nicht und benötigt sie gar nicht, um möglichst gute Züge zu ermitteln. Er kennt lediglich die Karte und weiß, welche Einheiten sich wo auf der Karte befinden. Instantiiert wird der *Strategie Kern* zum einen vom *Negotiator*, der den *Strategie Kern* nutzt, um Zugpakete zu erhalten, da er sich selber um die Verhandlung kümmert, aber nicht um die Berechnung von Zügen. Zum anderen gibt es noch einen Aufruf durch die *Gegnerbewertung*, welche herausfinden möchte, wie gut ein gegnerischer Zug war, um die Intelligenz des Gegners einschätzen zu können.

Der *Strategie Kern* erhält über seinen Konstruktor vier verschiedene Arrays, die Großmächte enthalten.

Allianz-Array beinhaltet alle Großmächte, die mit dem Bot verbündet sind. Die Macht, für die der Bot spielt, befindet sich auch im Allianz-Array. Befindet sich nur eine Macht im Allianz-Array, so kann man davon ausgehen, das es genau die Macht ist, für die der Bot spielt.

Freunde-Array beinhaltet Großmächte, die nicht angegriffen werden dürfen.

Neutral-Array enthält Großmächte, zu denen es weder freundliche noch feindliche Beziehungen gibt.

Feind-Array besteht aus explizit verfeindeten Mächten. Im Gegensatz zum Allianz- und Freunde-Array dürfen und sollen die Mächte im Neutralen- und Feinde-Array angegriffen werden.

Im Laufe der Berechnungen und der Verhandlungen der anderen Module können vom *Negotiator* in drei weitere Datenstrukturen Informationen gespeichert werden. Es gibt drei Queues, die Zugpakete halten. Die erste Queue enthält Zugpakete, die gemacht werden müssen (*EssentialMoves*), die zweite solche, die nicht gemacht werden dürfen (*IllegalMoves*) und die letzten beinhaltet diejenigen, die erwünscht sind (*WishedMoves*). Sobald sich eine der Datenstrukturen ändert, werden mit den neuen Informationen neue Züge erstellt und die alten vorhandenen Zugpakete auf Inkonsistenzen überprüft.

Da es, wie in Abschnitt 4 auf Seite 19 beschrieben, fünf verschiedene Phasen im Spiel gibt, muss auch vom *Strategie Kern* auf jede Phase individuell reagiert werden. Dazu wird direkt nach dem Aufruf geprüft, in welcher Phase der Aufruf kam, um anschließend mit Hilfe von unterschiedlichen Methoden die Aufgabenstellungen der jeweiligen Phase zu bearbeiten.

Die Bewegungsphasen im Frühling und Herbst werden in 5.2.2.2.2 beschrieben, wobei an dieser Stelle auch auf Unterschiede zwischen den beiden ähnlichen Phasen eingegangen wird. Die beiden Rückzugphasen beschreibt Abschnitt 5.2.2.2.3. Abschnitt 5.2.2.2.4 behandelt die Phase, in der neue Einheiten erstellt werden können.

5.2.2.2.1 Eröffnungen Im Jahr 1995 veröffentlichte Danny Loeb ein Paper [Loe95f], in dem er sich mit Eröffnungszügen für Diplomacy beschäftigte (siehe 5.2.1.1.3 auf Seite 44). Die Eröffnungsdatenbank wurde nach den uns vorliegenden Informationen nie im *Bordeaux Diplomaten* verwendet. Wir entschieden uns allerdings dafür, das von Loeb geplante Vorhaben in unseren Bot zu implementieren. Der *Strategie Kern* prüft aus diesem Grund in Bewegungsphasen, ob es sich um den Frühling im Jahr 1901 handelt. Für diesen Spezialfall existiert eine Datenstruktur, welche für jedes Land drei bis vier verschiedene Eröffnungszüge aus „The Library of Diplomacy Openings“ (vgl. [Han95]) bereit hält. Das zufällig ausgewählte Zugpaket wird dann, ohne dass es durch den *EA* verändert werden kann, an den *Negotiator* zurück gegeben. Diese Tatsache beschränkt zwar die Anzahl der möglichen Eröffnungszüge, hilft aber dabei, dass ein Eröffnungszug nicht beliebig schlecht werden kann. Außerdem wird dadurch veranlasst, dass nicht immer die gleichen Eröffnungszüge gespielt werden.

5.2.2.2.2 Bewegungs-Phase Aus der Motivation heraus, dass die Berechnung von Zugpaketen länger dauern könnte als der *Strategie Kern* Zeit hat, wurde eine effiziente Methode implementiert, die zu Beginn einer Bewegungsphase in die Datenstruktur des Aufrufers ein Zugpaket schreibt, in dem sich ausschließlich Halte-Befehle für alle Einheiten befinden. Dies sollte die Fehleranfälligkeit deutlich senken. Das zweite und offenbar interessantere Vorgehen war dann die Berechnung von möglichst guten Zügen. Zur Verdeutlichung der einzelnen im Folgenden näher beschriebenen Arbeitsschritte des Algorithmus dient Abbildung 14 auf Seite 61.

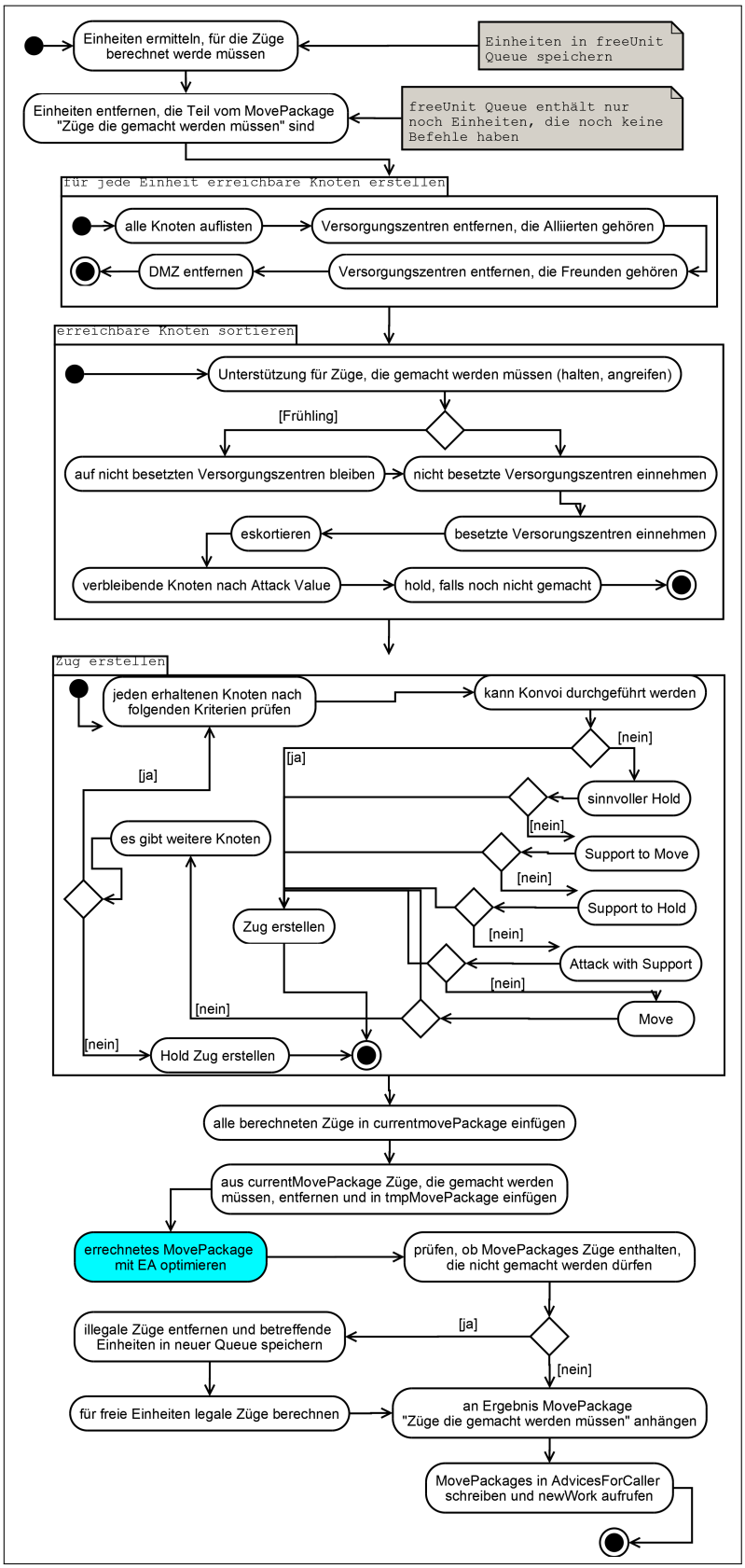


Abbildung 14: Strategie Kern: Aktivitätsdiagramm Bewegungs Phase

Einheiten suchen
Da der *Strategie Kern* nicht weiß, für welche Macht er spielt, muss er sich auf die Informationen berufen, die ihm übergeben werden. Um herauszufinden, für welche Einheiten Züge berechnet werden müssen, wird das Allianz-Array durchlaufen. Von jeder in dem Array enthaltenen Macht werden sämtliche Einheiten gespeichert und anschließend weitergereicht.

Einheiten filtern
Es besteht die Möglichkeit, dass Einheiten gefunden werden, die bereits einen Auftrag haben und für einen anderen Zug eingeplant wurden. Daher werden *EssentialMoves* und die Menge der Einheiten verglichen. Wird eine Übereinstimmung gefunden, dann wird die betreffende Einheit aus der Datenstruktur entfernt, da sie nicht mehr für weitere Befehle zu Verfügung steht. Das Resultat dieser Operation ist eine Liste von Einheiten, für die Züge berechnet werden müssen. Für jede dieser Einheiten werden die nachfolgenden Schritte durchgeführt.

Erreichbare Provinzen suchen
Nicht alle erreichbaren Provinzen eignen sich gleich gut für einen Zug. Manche habe sogar Eigenschaften, die sie als Ziel disqualifizieren. Befindet sich beispielsweise ein Versorgungszentrum einer alliierten oder befreundeten Macht auf der erreichbaren Provinz, so darf sich die Einheit nicht auf diese Provinz bewegen. Eine Bewegung der Einheit könnte als feindliches Verhalten klassifiziert werden. Aufgrund der Tatsache, dass wir uns für die Modellierung des Klassikers (vgl. 5.2.1.1.1 auf Seite 42) entschieden haben, ist ein solcher Zug untragbar. Beachtet werden muss hier, dass ohne eine Einschränkung der Funktionalität eine Einheit nie auf ihr eigenes Versorgungszentrum ziehen würde, da sich der Besitzer des Versorgungszentrum im Allianz-Array befindet. Deshalb muss hier geprüft werden, ob der Besitzer des Versorgungszentrums der Großmacht dieser Einheit entspricht, was wiederum einen Zug auf das Versorgungszentrum erlaubt. Weiterhin muss geprüft werden, ob die Zielprovinz zu einer demilitarisierten Zone gehört, was einen Zug dahin auch verbietet. Bevor es in der Berechnung der Züge fortgeführt wird, muss noch eine wesentliche Bedingung der Spielregeln erfüllt sein: Spielregel 6 (vgl. 4.1 auf Seite 19) besagt, dass zwei Einheiten nur mit Hilfe eines Geleitzuges, also eines Konvois, direkt ihre Plätze tauschen können. Eine Überprüfung ist also unumgänglich, ob auf der Zielprovinz eine Einheit steht, die in die Quellprovinz ziehen möchte.

Erreichbare Provinzen sortieren
Nachdem herausgefunden wurde, welche Provinzen von einer Einheit erreichbar sind, müssen den Provinzen anhand von verschiedenen Kriterien, wie zum Beispiel dem Angriffswert der Provinz, Prioritäten zugewiesen werden. Eine Sortierung der Provinzen ist sinnvoll, da im nächsten Schritt dann nur noch geprüft werden muss, ob ein Zug auf die Provinz mit der höchsten Priorität ausgeführt werden kann. Falls das nicht der Fall ist, kann die nächste Provinz in der Liste überprüft werden.

Die durch die Karte vorgegebenen Angriffswerte wurden wie folgt verändert:

- gehört die Provinz einem Feind: +50
- gehört die Provinz einer neutralen Macht: +50
- befindet sich auf der Zielprovinz ein Versorgungszentrum
  - und es ist noch unbesetzt: +100
  - es ist nicht von einer Einheit besetzt und gehört auch nicht der angreifenden Macht: +25
  - gehört nicht der angreifenden Macht und die Zielprovinz ist die Ausgangsprovinz: +200

  - gehört nicht der angreifenden Macht und die Zielprovinz ist nicht die Ausgangsprovinz: +100
  - und es befindet sich bereits im Besitz der angreifenden Macht: −500
- es gibt Einheiten, die die Zielprovinz angreifen oder halten wollen und Unterstützung benötigen, weil mehr als eine feindliche Einheit angreifen könnte: +100

Die erreichbaren Provinzen wurden nach den modifizierten Angriffswerten dann absteigend sortiert und weitergereicht.

Zug erstellen
Bei der Erstellung von Zügen wurden die nach Priorität sortierten Provinzen nach und nach untersucht, bis ein Zug erstellt werden konnte. Konnte für keine Zielprovinz ein Zug erstellt werden, dann wurde ein Halte-Befehl an die betreffende Einheit gegeben.

Die zur Erstellung von Zügen eingesetzte Methode arbeitet nach folgenden Prinzip: Zuerst wird geprüft, ob die Zielprovinz der Quelle entspricht und die Einheit eine Flotte ist. Bei dieser Konstellation ist zwar ein Halte-Befehl möglich, ein Konvoi aber interessanter und gegebenenfalls auch effektiver. Folglich wird geprüft, ob es eine Armee gibt, die sich in Reichweite der Flotte befindet und noch keine Befehle hat. Gibt es eine solche Einheit, dann wird überprüft, ob ein Konvoi möglich ist, es also auch eine erreichbare Küste für die Armee gibt. Sind diese Umstände gegeben, dann wird ein Konvoi durch die Flotte für die Armee erstellt. Als nächstes wurde auf einen sinnvollen Haltebefehl geprüft.

Ein Haltebefehl ist nach Wheeler [Whe97] beziehungsweise Woodhouse [Woo99] (vgl. 5.2.1.1.8 auf Seite 45 und 5.2.1.1.9 auf Seite 46) genau dann sinnvoll, wenn ein fremdes Versorgungszentrum damit erobert oder ein eigenes Versorgungszentrum verteidigt werden kann. Aus rein taktischer Sicht gibt es noch weitaus mehr Gründe für einen Halte-Befehl, die allerdings aufgrund der Komplexität der Gründe und der fehlenden Möglichkeit im Spiel, in die Vergangenheit zu sehen, nicht berücksichtigt wurden. Danach wird die Möglichkeit für einen Bewegungsbefehl überprüft.

Es gibt für die Bewegung einer Einheit vier verschiedene Typen. Der wichtigste, ohne den keine Einheit geschlagen werden könnte, ist die Unterstützung einer Bewegung. Bei der Berechnung von Zügen wird dies als erster Bewegungstyp abgehandelt. Gesucht wird dabei eine Einheit, die die gleiche Zielprovinz hat und für die es Gründe für eine Unterstützung gibt. Ein Grund für eine solche Unterstützung ist ohne Zweifel die Tatsache, dass sich eine feindliche oder neutrale Einheit auf der Zielprovinz befindet. Sofern sich die Armee auf der Zielprovinz nicht von dem Feld entfernt, würde der Angriff von nur einer Einheit einfach abgewehrt werden und wäre somit unsinnig. Erst durch die Unterstützung der zweiten Einheit könnte ein sinnvoller Angriff durchgeführt werden. Beachtet werden muss hierbei allerdings, dass die haltende feindliche oder neutrale Einheit auch eine Unterstützung von anderen Einheiten erhalten könnte, sofern andere Einheiten in der Nähe sind. Deshalb kann auch ein Angriff mit nur einer Unterstützung fehlschlagen. In diesem Fall müssten also mehrere Einheiten den Angriff unterstützen, damit er ausgeführt werden kann. Auf der anderen Seite besteht auch die Möglichkeit einer unsinnigen Unterstützung, da es auch schon genügend unterstützende Einheiten geben kann. Sind dann weniger feindliche oder neutrale Einheiten zum Halten der Zielprovinz in erreichbarer Nähe, dann muss die Summe der angreifenden Einheiten, inklusive der unterstützenden Einheiten, die verteidigenden beziehungsweise haltenden Einheiten nur um eine Einheit überschreiten. Alle anderen Einheiten können sich wichtigeren Aufgaben widmen. Folglich wird nicht nur geprüft, ob eine Unterstützung einer anderen Einheit sinnvoll ist, sondern auch, ob die Unterstützung vielleicht sogar unsinnig ist.

Der zweite Teil der Überprüfung einer Einheitenbewegung behandelt die Unterstützung

eines Haltebefehls. Ähnlich wie bei der Unterstützung eines Angriffs kann es eine Einheit geben, die alleine ihren Befehl nicht erfolgreich ausführen kann. Befindet sich nämlich eine Einheit auf der Zielprovinz, die bereits einen Haltebefehl bekommen hat, so ist es unumgänglich zu überprüfen, ob diese Einheit beim Halten der Provinz unterstützt werden sollte. Eine Unterstützung gliedert sich auch hier wieder in sinnvoll und unsinnig. Sinnvoll ist die Unterstützung genau dann, wenn die Anzahl der potentiellen Angreifer die der Verteidiger übersteigt. Im Gegensatz dazu ist sie unsinnig, wenn es weniger mögliche Angreifer als Verteidiger gibt. Beides wird hier überprüft und gegebenenfalls wird ein Befehl an die aktuelle Einheit gegeben.

Der dritte abgehandelte Punkt in der Phase der Zugerstellung ist der Bewegungsbefehl mit Unterstützung von anderen Einheiten. Ähnlich wie beim ersten Fall wird hier ein Angriff ausgeführt. Allerdings gibt es im Gegensatz zum ersten Fall noch keine Einheit, die unterstützt werden kann. Damit der Angriff nicht direkt abgewehrt wird, muss also nach mindestens einer Einheit gesucht werden, die den Angriff unterstützt. Geprüft werden muss auch, wie viele gegnerische Einheiten den Angriff unter Umständen vereiteln können. Folglich muss die richtige Anzahl von Einheiten gefunden werden, die den Angriff durchführen und unterstützen. Wird ein sinnvoller Angriff mit ausreichend vielen unterstützenden Einheiten ermittelt, so wird dieser Zug berechnet.

Die letzte Möglichkeit für einen Bewegungsbefehl ist die normale Bewegung. Diese wird dann ausgeführt, wenn sich keine andere Einheit auf der Zielprovinz befindet, und die aktuelle Einheit somit vermutlich auch nicht an ihrer Bewegung gehindert wird.

Konnte für keine Provinz in der sortierten Liste ein möglicher Zug gefunden werden, dann wird der aktuellen Einheit der Haltebefehl gegeben. Eigentlich sollte man davon ausgehen können, dass eine Einheit immer einen Zug machen kann, der sich von einem unsinnigen Zug unterscheidet. Allerdings wird jeder erstellte Zug, bevor er im nächsten Schritt zum Zugpaket hinzugefügt wird, mit den Zügen die Liste der *IllegalMoves* vergleichen. Dabei ist nicht sichergestellt, dass nicht jeder Zug aufgrund von Verhandlungen im *Negotiator* verboten ist. Um generelle Konflikte zu vermeiden, die entstehen, wenn eine Einheit gar keinen Befehl hat, wird der Einheit aus diesem Grund mindestens der Haltebefehl gegeben.

Zuletzt wird dann die Einheit, für die ein Zug berechnet wurde, aus der Liste der freien Einheiten entfernt, damit es nicht zu Konflikten kommt, bei denen eine Einheit verschiedene Befehle ausführen soll. Beachtet werden muss auch hier, dass mehr als nur eine Einheit aus der Liste der freien Einheiten entfernt werden kann, da in einem Schritt unter Umständen mehr als nur einer Einheit Befehle gegeben werden (Beispiel: Konvoi oder Angriff mit Unterstützung).

Züge zu einem Zugpaket zusammenfügen
Die durch die Zugerstellung berechneten Züge werden in einem Zugpaket gesammelt.

Bestimmte Züge aus einem Zugpaket entfernen
Auf der einen Seite gibt es Züge, die nicht gemacht werden dürfen. Diese Züge werden bei der Berechnung direkt verworfen. Auf der anderen Seite gibt der *Negotiator* allerdings auch Züge vor, die gemacht werden müssen. Diese Züge werden nun aus dem Zugpaket entfernt, damit der *EA* keine Möglichkeit hat, diese Züge zu verändern.

Zugpaket von *EA* optimieren lassen
Dem *EA* wird das aktuelle Zugpaket übergeben und dieser startet dann die Optimierung.

Illegale Züge entfernen
Da der *EA* die Züge zwar optimiert, selber aber nicht weiß, welche Züge nicht erlaubt sind, müssen die durch den *Negotiator* als illegal gekennzeichneten Züge heraus gefiltert werden. Dies wird an dieser Stelle durchgeführt. Um die dann frei werdenden Einheiten

wird sich im nächsten Schritt gekümmert.

Für freie Einheiten neue Züge berechnen
Die im vorherigen Schritt frei gewordenen Einheiten durchlaufen alle vorherigen Schritte ab **Erreichbare Provinzen suchen** erneut. Die resultierenden Züge werden allerdings nicht erneut durch den EA optimiert, da es sonst im schlimmsten Fall zu einer Endlosschleife kommen könnte.

Alle Züge in Zugpaket vereinen
Nachdem alle Einheiten ihre Befehle haben, werden dem aktuellen Zugpaket noch die Züge hinzugefügt, die der *Negotiator* als Pflichtzüge gekennzeichnet hat. Das gesamte Zugpaket wird dann an den *Negotiator* zurückgegeben.

5.2.2.2.3 Rückzug-Phase Wird eine eigene Einheit durch einen gegnerischen Angriff attackiert, und ist der Angriff erfolgreich, so wird die Einheit nicht direkt entfernt. Sie hat die Möglichkeit, sich durch einen Rückzug zu retten. Die vom *Negotiator* übergebenen möglichen Rückzüge werden dann analysiert. Es wird überprüft, ob es schon eigene Einheiten gibt, die sich in eine bestimmte Provinz zurückziehen wollen. Da alle Einheiten entfernt werden, die sich in die gleiche Provinz zurückziehen, wird überprüft, dass es nicht schon eine eigene Einheit gibt, die sich in die betreffende Provinz zurück zieht. Die Ergebnisse werden dann an den *Negotiator* übergeben.

5.2.2.2.4 Bau-Phase In der einmal pro Jahr im Spiel stattfindenden Bau-Phase wird zuerst geprüft, ob neue Einheiten erstellt werden sollen oder alte Einheiten entfernt werden müssen. Können neue Einheiten gebaut werden, so wird ein Heimversorgungszentrum gesucht, in dem sich keine andere Einheit befindet. Ist England die Macht, für die neue Einheiten erstellt werden können, werden nur Flotten gebaut, sonst sind es Armeen. Tritt der zweite Fall ein, dass es mehr Einheiten als Versorgungszentren gibt, dann müssen Einheiten entfernen werden. In diesem Fall wird, falls vorhanden, zuerst eine Armee, danach eine Flotte entfernt. Diese Strategie wird solange iteriert, bis genug Einheiten entfernt wurden.

5.2.2.2.5 Zug-Bewertung Da sich die Verantwortung des *Strategie Kerns* nicht nur über die Erstellung von Zügen erstreckt, sondern der *Strategie Kern* beantworten muss, wie gut ein eigener oder gegnerischer Zug ist, sind in ihm weitere Methoden für dieses Vorhaben realisiert. Es gibt eine öffentliche Methode, die ganze Zugpakete bewertet. Diese stützt sich allerdings auf eine weitere öffentliche Methode, die einzelne Züge bewertet. Bei der Bewertung der Züge wird zum einen unterschieden zwischen dem Typ des übergebenen Zuges und zum anderen zwischen der Relevanz des Zuges. Haltebefehle bekommen nur dann einen positiven Wert, wenn sie als sinnvoll eingestuft werden. Dann erhöht sich der Wert mit der Anzahl der Gegner, die versuchen, die verteidigte Provinz zu erobern. Unnötige Haltebefehle erhalten einen negativen Wert. Unterstützungsbefehle erhalten genau dann eine negative Bewertung, wenn es keine feindlichen Einheiten in der Nähe gibt, durch die die Unterstützung gerechtfertigt wäre. Ansonsten steigt auch ihr Wert mit der Anzahl der Gegner. Bewegungs- oder Angriffbefehle werden nach der Kategorie der Gegner bewertet. Angriffe auf Feinde sind dabei besser als Angriffe auf neutrale Mächte. Der Zug einer Einheit, die einen Konvoi ermöglicht und auch der Zug der Einheit, die durch den Konvoi überführt wird, werden beide positiv bewertet. Die konkrete Bewertung ist abhängig von den Angriffswerten der Provinzen, die je nach Zugtyp mit einem festen Faktor multipliziert werden.

5.2.2.3 EA-Optimierung

Der Evolutionäre Algorithmus (EA) ist ein externes Modul, welcher beliebige Individuen optimierieren kann. Das Modul übernimmt die Organisation der Evolution und Selektion ohne Berücksichtigung der individuumsspezifischen Mutation oder Rekombination.

Die zu optimierenden Individuen im Stragotiator sind Instanzen der Klasse *EvoMovePackage*, welche von der Klasse *MovePackage* erben. Ein *MovePackage* besteht aus einer Menge von Zügen (*Moves*). Jedes *EvoMovePackage* gilt nur für eine Runde/Jahreszeit. Danach werden neue *EvoMovePackages* berechnet. In dieser Klasse sind die Realisierung der Operationen Mutation und Rekombination kodiert.

Mutation
Die Mutation hat für einen EA die Aufgabe, ein Individuum so zu manipulieren, dass theoretisch jeder Punkt im gesamten Suchraum erreicht werden kann. Der EA hat in diesem Fall dabei nur die Entscheidungsgewalt über Züge, welche Einheiten bewegen und welche im Frühling und im Herbst stattfinden. Züge wie *Wave build*, *Build*, *retreat* (vgl. Abschnitt 4.2) entfallen daher. Syntaktisch haben alle diese Züge die Gemeinsamkeit, dass sie Start und Ziel der Bewegung beinhalten. Aus dem Festlegen des Ziels lässt sich der Typ bestimmen. Ein Zug in eine Provinz X mit einer Einheit E_2 wird beispielsweise als *Support_to_hold* interpretiert, sofern E_2 in X bleibt.

Die Mutation für ein Zugpacket erfolgt durch die sukzessive Mutation von Zügen. Für jeden Zug wird zunächst eine Liste aller möglichen Zielprovinzen erstellt, in die die Einheit ziehen kann. Die Liste wird anschließend um die Startprovinz als mögliches „Ziel“ erweitert, was dem EA die Möglichkeit gibt, die Einheit nicht zu bewegen. Aus dieser Liste wird dann randomisiert ein Ziel ausgewählt. Ist die neue Zielprovinz gleich der Startprovinz, wird der *Move* auf *Hold* gesetzt. Steht in der Zielprovinz eine Einheit, die der EA steuert, wird der Typ des Zuges auf *Move* gesetzt. Falls in der Zielprovinz eine Einheit einer alliierten Macht steht, wird gleichverteilt randomisiert entschieden, ob ein neues Ziel gewählt oder diese Einheit bei einem *Hold* unterstützt wird.

Rekombination
Die Rekombination zwischen zwei *EvoMovePackages* gestaltet sich sehr einfach, sofern eine Ordnung über den einzelnen *Moves* existiert. Die Ordnung der *Moves* ist durch die lexikographischen Ordnung ihrer Startprovinzen gegeben. Da niemals mehr als eine Einheit auf einer Provinz stehen kann, und weil die Provinzen alle unterschiedliche Bezeichnungen haben, ist diese Ordnung eindeutig.

Durch die gegebene Ordnung kann die Rekombination realisiert werden. Ein *Kind-EvoMovePackage* wird nun erzeugt, indem es für jede Einheit einen entsprechenden Zug aus dem einen *Eltern-EvoMovePackage* oder dem anderen erhält. Welches Elternindividuum nun den Zug bereitstellt, wird randomisiert zugewiesen. Tabelle 5.10 zeigt ein Beispiel einer Rekombination von zwei Zugpaketen zu einem neuen. Dabei repräsentiert das Element $M_{A,2}$ den zweiten Zug (gemäß der lexikographischen Ordnung) aus *EvoMovePackage A*. Das *Kind-EvoMovePackage* erhält nun die Züge aus beiden *EvoMovePackages* randomisiert zugeteilt. In diesem Fall den ersten und letzten aus *EvoMovePackage A* und den zweiten und dritten aus *B*. Die anderen Züge werden aus Übersichtlichkeitsgründen nicht aufgelistet.

Beispiel:

Kollisionsbehandlung
Bei der Evolution können Fehler innerhalb des *EvoMovePackages* auftreten, die während der Mutations- oder Rekombinations-Schritte noch nicht erkannt werden. Zum Beispiel kann es vorkommen, dass zwei Einheiten eine gegnerische Einheit angreifen

EvoMovePackage A	$M_{A,1}$	$M_{A,2}$	$M_{A,3}$	...	$M_{A,n}$
EvoMovePackage B	$M_{B,1}$	$M_{B,2}$	$M_{B,3}$	...	$M_{B,n}$
Kind-EvoMovePackage	$M_{A,1}$	$M_{B,2}$	$M_{B,3}$	...	$M_{A,n}$

Tabelle 5.10: Beispiel einer Rekombination von zwei Zugpaketen

wollen (oder, falls es keine solche gibt, in die selbe Provinz ziehen). Dann würden beide Züge kollidieren und gemäß den Regeln nicht ausgeführt. In dem Fall ist es so, als hätten diese Einheiten gar nicht gezogen. Die feindliche Einheit (sofern eine in der Zielprovinz stand) wäre unversehrt.

Um das zu vermeiden, müssen in jeder Generation vor der Bewertung der Zugpackete solche Konflikte gelöst werden.

Für die Behandlung des ersten Konflikttyps wird eine Hashtabelle angelegt, welche als Zellen lineare Listen enthält. Jeder *Move* wird mit der Zielprovinz als Schlüssel gehasht und in die Liste eingefügt. Am Ende wird für jede Liste, die mehr als einen Zug enthält (was genau bei der Situation entsteht, dass mehr als eine Einheit in eine Provinz ziehen möchten), randomisiert ein *Move* aus der Liste gewählt, der zieht. Alle anderen *Moves* werden so modifiziert, dass sie den Ausgewählten Zug unterstützen (*Support_to_move*).

Eine weitere ungewollte Situation stellt das Ziehen einer Einheit E_1 in eine Provinz Z, wo bereits eine Einheit E_2 der selben Macht steht, welche dort stehen bleibt. In dem Fall bleibt E_1 wo sie ist (*Hold*). Allein vom Spiel betrachtet stellt diese Situation kein gravierendes Problem dar, weil E_1 sich nicht bewegen wird. Allerdings wird u.U. ein Zugpaket besser bewertet, weil der Zug von E_1 die Bewertung vom Zugpaket erhöht, obwohl der Zug gar nicht erfolgreich sein kann.

Ein weiterer illegaler Zug ist das Wechseln von zwei benachbarten Einheiten. Zum Beispiel zieht Einheit E_1 von A nach B, und Einheit E_2 zieht von B nach A. Dies muss durch ein Durchsuchen des ganzen *EvoMovePackages* erkannt und unterbunden werden, indem für beide Einheiten der *Hold*-Befehl festgelegt wird.

Fitnessfunktion
Um die Qualität von Zugpaketen zu ermitteln, wird eine Fitnessfunktion benötigt, welche einem Zugpacket eine reelle Zahl zuweist. Die Fitnessfunktion wird zentral vom *Strategie Kern* festgelegt (vgl. 5.2.2.2.5 auf S. 65).

Terminierung
Der EA optimiert die *EvoMovePackages* über 500 Generationen hinweg. Falls er jedoch innerhalb 350 Generationen keine besseren *EvoMovePackages* berechnen kann, terminiert der EA frühzeitig.

5.2.2.4 Gegnerbewertung

Die Grundideen, sowohl für die Frage nach Freund und Feind, als auch nach der Frage der vermuteten Intelligenz aller Mächte, basieren ebenfalls auf einem Paper von Danny Loeb [Loe95e]. Das Verhandlungsmodul benötigt demnach Informationen darüber, in welcher Beziehung die einzelnen Mächte zueinander stehen. Sind sie befreundet, verfeindet oder irgendetwas dazwischen?

Diese Beziehungen werden durch eine Matrix mit sieben mal sieben Feldern von reellen Zahlen dargestellt. Hohe Werte sollen hier ein freundliches Verhältnis modellieren, niedrige entsprechend ein eher feindliches Verhältnis.

Zur Bestimmung der Werte der Matrix werden alle platzierten Züge pro Runde betrachtet.

Loeb machte nun 9 Vorschläge diese Züge zu betrachten, um herauszufinden, wie die Mächte zueinander stehen:

Verletzung von besonderen Vereinbarungen Darunter zählen unter anderem Vereinbarungen bzgl. demilitarisierter Zonen, Zugvorschläge usw. Werden solche Vereinbarungen gehalten, so gibt es einen Bonus auf den entsprechenden Wert der Matrix, ansonsten einen stärkeren Abzug.

Verletzung von allgemeinen Vereinbarungen Hierzu zählen in erster Linie Friedensangebote.

Koordinierte Bewegungen Wenn eine Macht mit einer anderen erkennbar bei Zügen kooperiert (Unterstützung, Konvoy,...), dann wird dies als starker Hinweis auf eine freundschaftliche Beziehung gewertet.

***Angriffe** Greift eine Macht eine andere an, dann wird dies als sicheres Zeichen dafür gewertet, dass man den entsprechenden Wert in der Matrix herabsetzen kann.

***Bewegung** Wenn eine Macht eine Einheit bewegt, wird für jede andere Macht das jeweils nächste Versorgungszentrum betrachtet und geschaut, ob man sich diesem genähert hat oder nicht. Im ersteren Fall wird der entsprechende Wert der Matrix verkleinert, sonst erhöht. Je größer allerdings die Distanzen sind, um so geringer hat eine solche Änderung auszufallen, da die Auswirkung auf die Macht dann auch geringer sind.

***Geschichte** Damit nicht immer nur die letzten Züge in die Matrix eingehen, sollen auch noch die vorletzten Züge zu einem gewissen Prozentsatz mit übernommen werden.

Größe Die Größe der eigenen Macht sollte ebenfalls Einfluss haben. Je größer wir sind, desto eher schüchtern wir potentiell ein.

***Symmetrie** Es wird davon ausgegangen, dass die Beziehung zwischen zwei Mächten immer symmetrisch ist: Macht A ist mit einer anderen Macht B genauso befreundet, wie diese wiederum mit Macht A befreundet ist.

Transitivität Freunde meiner Freunde, sollen auch wieder meine Freunde sein.

Leider konnte aus Zeitgründen nur ein kleiner Teil dieser Ideen umgesetzt werden. Die endsprechenden Ideen sind mit einem * markiert. Ebenfalls mussten bei den umgesetzten Punkten Entscheidungen bzgl. der Höhe der Strafwerte/Bonuswerte getroffen werden. Auch hier fehlte leider die Zeit, die Parameter hinreichend anzupassen.

Damit hatten wir allerdings zumindest eine rudimentäre Freund/Feind-Matrix. Bei der Gegenermodellierung war allerdings für das Verhandlungsmodul noch die Einschätzung der Mitspieler bzgl. deren Intelligenz wichtig. Wir sind bei der Frage, wie man Intelligenz von Mächten bestimmen können, sehr selbstbewusst vorgegangen:

Intelligent ist, wer so zieht, wie wir ziehen würden.

Das heißt im Falle des Stragotiators: Wer ähnlich zieht, wie es auch unser Strategiekern tun würde.

Wir mussten also nur die Züge jeder Macht nehmen und schauen, ob unser Strategiekern für diese Macht einen ähnlichen Zug vorgeschlagen hätte. Dazu musste auch nicht mehr viel entwickelt werden, denn der Strategiekern hat schon aus einem anderen Grund die Fähigkeit, Züge zu bewerten. Diese Bewertung wurde dann direkt für den Intelligenzwert verwendet.

5.2.3 Parameteroptimierung

Nachdem wir uns mit der Realisierung unserer Vorhaben beschäftigt haben, werden wir im folgenden Abschnitt auf unsere Tests bezüglich der Güte der Parameter in den einzelnen Modulen eingehen. Für die Gegenerbewertung wurden aus Zeitgründen keine solchen Tests durchgeführt.

5.2.3.1 Negotiator

Wie schon in Abschnitt 5.2.2.1 auf Seite 48 beschrieben, befinden sich im Verhandlungsmodul Parameter um festzulegen, ab welcher Fitness ein Zug gut ist, ab welchem Intelligenzwert ein Mitspieler als intelligent gilt, welche Werte der Freund/Feind-Matrix das Verhalten eines Gegenspieler als freundlich bzw. feindlich klassifizieren, ab wann DRWs gesendet werden und wie das Vertrauen modelliert werden soll. Bei der Vertrauensmodellierung mit den *linguistischen Variablen* können der Bereich der Variablen, die vier *Fuzzy Mengen* und die Schrittweiten der Vertrauensveränderungen festgelegt werden.

Bei den Parametertests für das Verhandlungsmodul wurden die Auswirkungen der Parameter, ab wann ein eingehender Zugvorschlag angenommen wird, sowie die für die Vertrauensmodellierung, auf die Zusammenarbeit mit anderen Spielern, getestet. Ziel der Tests war es, Parameter zu finden, bei denen zwei *Stragotiatoren* besonders gut mit einander zusammenarbeiten. Die Testumgebung bestand aus einem Diplomacyspiel mit dem DAIDE-Server (siehe Abschnitt 4.4.2 auf Seite 22). Die Spiele wurden mit Verhandlungslevel 20, Movetimedeadline 5, Reatreattimedeadline 2 und Buildtimedeadline 2 gestartet. Um zu verhindern, dass ein Spiel zu lange dauert, sollte ein Unentschieden definiert werden, wenn sich die Zugehörigkeit der Versorgungszentren zu den Mächten binnen zehn Spieljahren nicht geändert hat. Die Mächte Deutschland und Österreich-Ungarn wurden von den beiden *Stragotiatoren* , die restlichen von Holdbots (siehe Abschnitt 4.5.8 aus Seite 26) gespielt. Die Holdbots als Gegner sollten die Spiele so deterministisch wie möglich machen und ein möglichst großes Maß an Reproduzierbarkeit gewährleisten.

Die ursprüngliche Idee war es, für den Schwellenwert der Zugfitness die Werte 0.0, 0.1, 0.2, ... 0.9, 1.0, und für die Schrittweite der Vertrauensveränderung, sowohl für das In-, wie auch das Dekrementieren die Werte 1, 2, 3, 4, 5, zu verwenden. Für das Vertrauen sollten die *linguistischen Variablen* TrustA (gleichmäßige Bereichsaufteilung, Bot soll nur sehr selten volles Vertrauen haben, siehe Abbildung 15), TrustB (gleichmäßige Bereichsaufteilung, Bot soll öfters volles Vertrauen haben, siehe Abbildung 16), TrustC (großer neutraler Bereich, Bot soll seinen Mitspielern möglichst lange neutral gegenüber stehen, siehe Abbildung 17), TrustD (großer Bereich für Misstrauen, Bot soll seinen Mitspielern eher misstrauen, siehe Abbildung 18) und TrustE (großer Bereich für Vertrauen, Bot soll seinen Mitspielern eher vertrauen, siehe Abbildung 19) getestet werden.

Von all diesen Parametern sollten mit jeder Kombination 5 Spiele gespielt werden. Bei den Spielen sollten für jeden der beiden *Stragotiatoren* folgende Informationen gespeichert werden:

- Wieviele XDOs wurden an den anderen *Stragotiator* gesendet
- Wieviele der gesendeten XDOs wurden vom anderen *Stragotiatoren* angenommen
- Wieviele der gesendeten XDOs wurden vom anderen *Stragotiatoren* abgelehnt
- Wieviele XDOs vom anderen *Stragotiator* sind eingegangen
- Wieviele der eingegangenen XDOs wurden angenommen

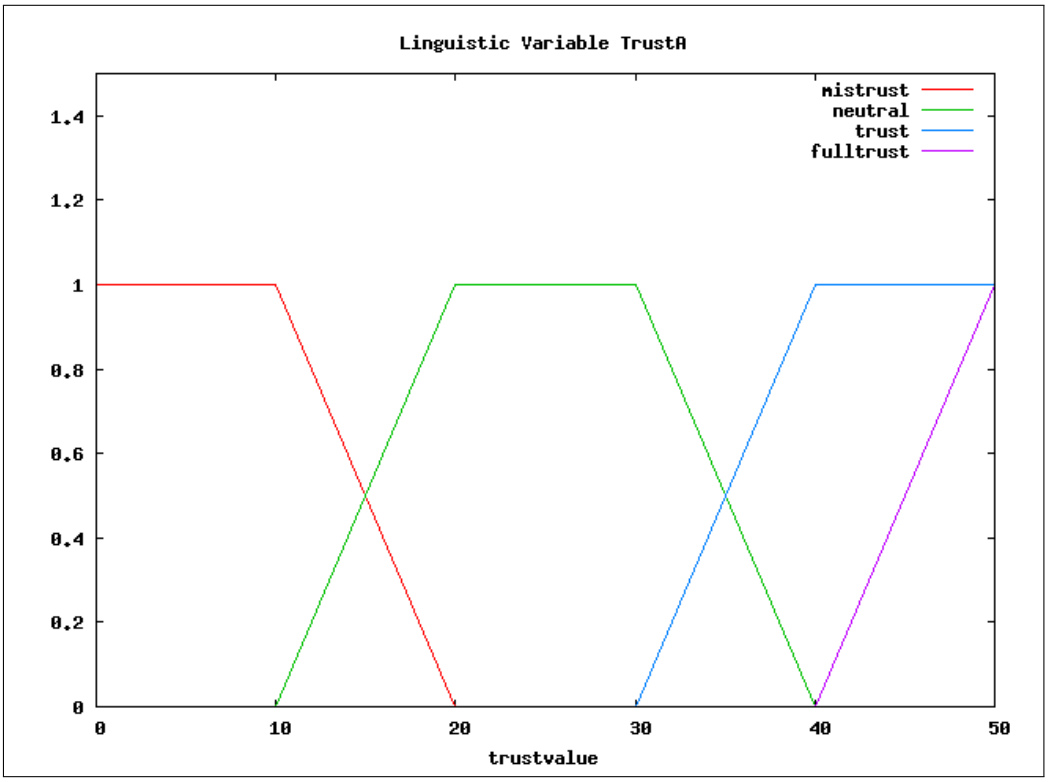


Abbildung 15: TrustA

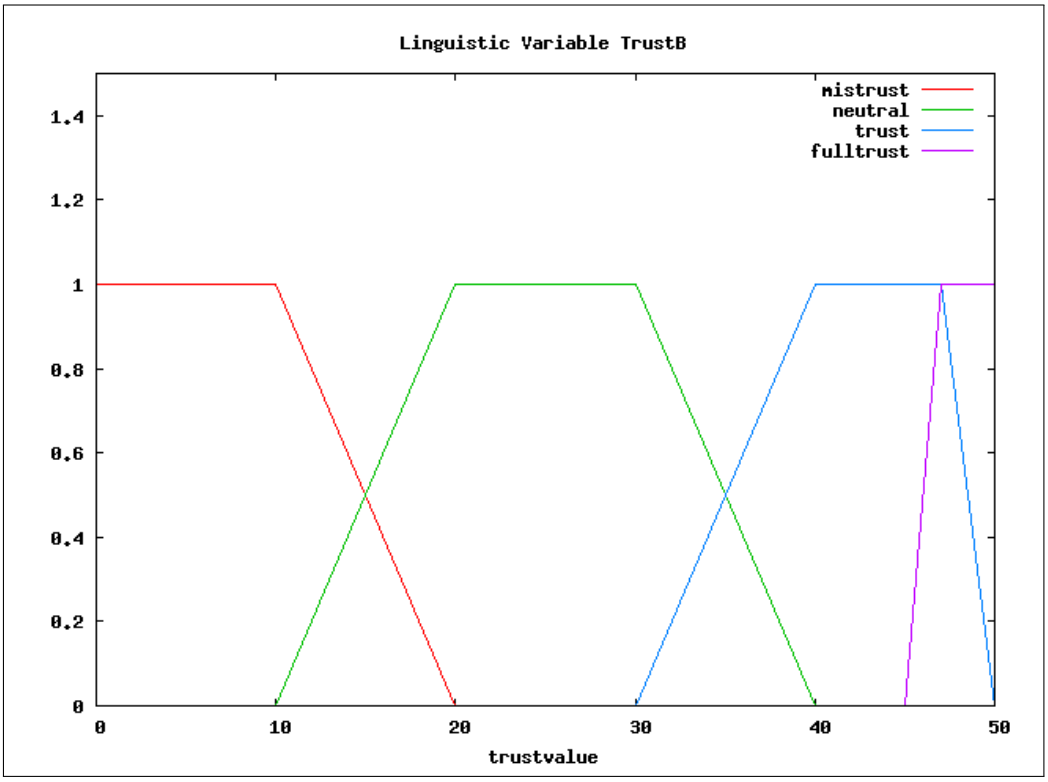


Abbildung 16: TrustB

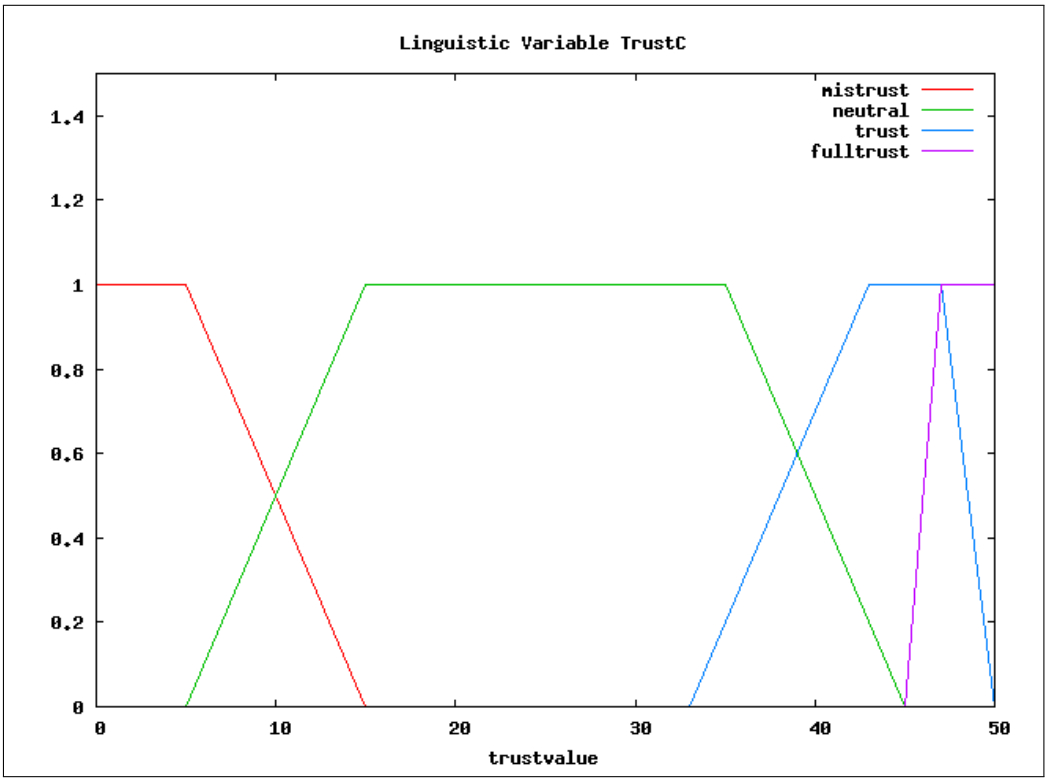


Abbildung 17: TrustC

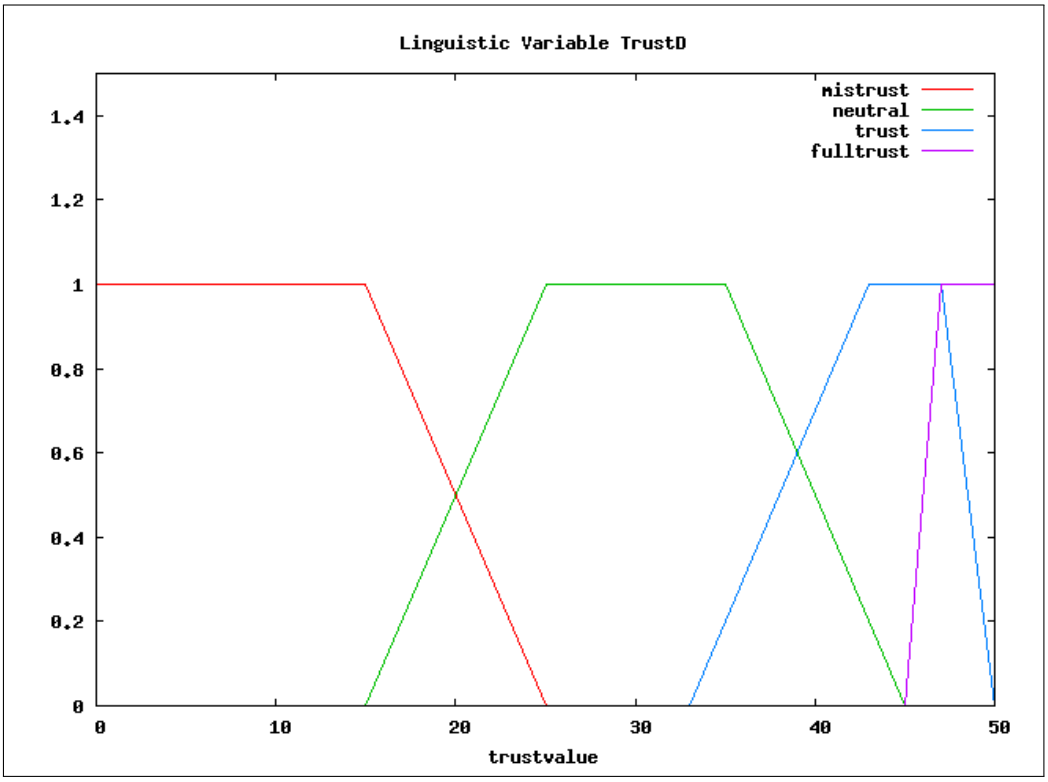


Abbildung 18: TrustD

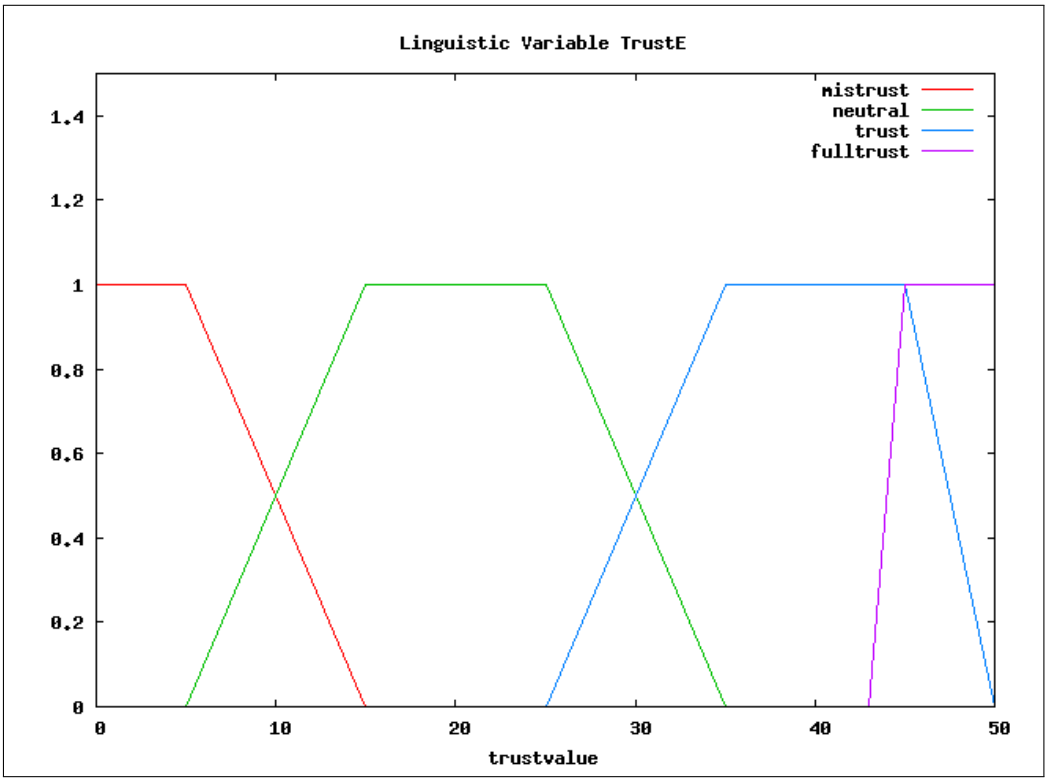


Abbildung 19: TrustE

- Wieviele der eingegangenen XDOs wurden abgelehnt
- Wieviele Supports wurden ausgeführt
- Wieviele davon haben den anderen *Stragotiator* unterstützt
- Wie war das Vertrauen am Ende des Spiels zu den anderen Mächten
- Wurde das Spiel durch ein Unentschieden oder einen Alleinsieg beendet

Weil die Arbeiten am *Stragotiator* zeitintensiver waren als erwartet, konnten diese Testspiele zeitlich nicht mehr durchgeführt werden. Von den fünf *linguistischen Variablen* wurde lediglich TrustB für die Parametertests verwendet. Außerdem wurde für das Dekrementieren des Vertrauenswertes eine Schrittweite von eins und für das Inkrementieren eine von zwei verwendet. Weil die Vermutung bestand, dass bei fünf Spielen die Streuung der Ergebnisse zu groß ist, wurden mit jedem der Schwellwerte für die Zugfitness zehn Spiele durchgeführt.

Zuerst wurden die Spiele ohne EA durchgeführt. Dabei stellte sich heraus, dass bei einem Schwellenwert zwischen null und eins jeder der beiden *Stragotiatoren* jeden Zugvorschlag annimmt. Also wurden für den Schwellenwert ebenfalls die Werte 1.5, 2.0, 2.5, und 3.0 getestet. Auch bei diesen haben beide *Stragotiatoren* bis auf maximal 2 XDOs pro Spiel alle angenommen. Bei der Suche nach dem Grund für dieses unerwartete Verhalten stellte sich heraus, dass der Strategiekern hauptsächlich Zugpakete berechnet, die eine negative Fitness besitzen. Der prozentuale Anteil dieser Fitness ist dann ebenfalls negativ, und ein vorgeschlagener Zug mit einer positiven Bewertung wird zwangsläufig angenommen. Aus diesem Grund wurde auf eine Aufbereitung der Daten und deren Vorstellung an dieser Stelle verzichtet.

Nach den Spielen ohne EA wurden die Schwellenwerte für die Zugannahme 0.0, 0.5, 1.0,

1.5, 2.0, 2.5, und 3.0 mit eingeschaltetem EA durchgeführt.

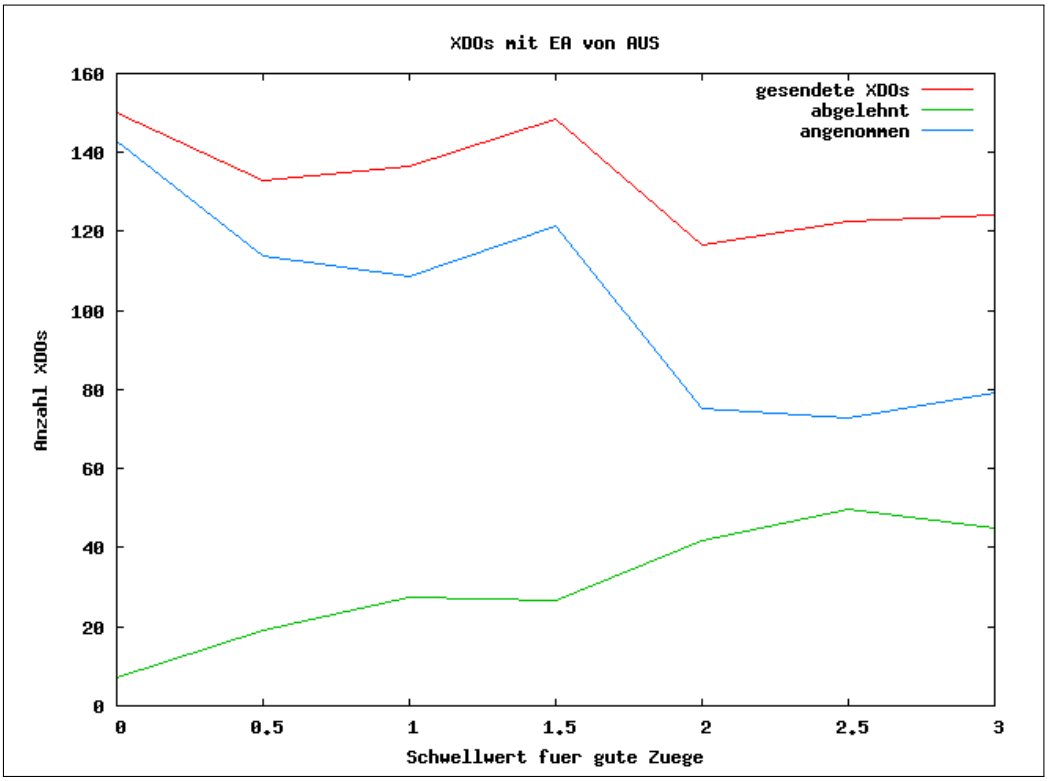


Abbildung 20: Von AUS gesendete Zugvorschläge

Schwellwert für gute Züge	Mistrauen	Neutral	Vertrauen	vollstes Vertrauen
0.0	0	0	0	10
0.5	1	0	0	10
1.0	0	0	1	9
1.5	1	0	0	9
2.0	2	0	1	7
2.5	1	1	1	7
3.0	2	0	1	7

Tabelle 5.11: Vertrauen von AUS zu GER

Die Abbildungen 20 und 21 zeigen, wieviele Zugvorschläge im Schnitt in den zehn Spielen mit einem Schwellenwert von AUS bzw. GER gesendet und wieviele davon von der jeweils anderen Macht angenommen bzw. abgelehnt wurden. Deutlich zu erkennen sind dabei die Erhöhungen für die gesendeten und angenommen XDOs bei einem Schwellenwert von 1.5. Zu beachten ist dabei, dass die Anzahl der Ablehnungen an dieser Stelle keine Erhöhung aufweist. Leider konnte keine Erklärung für dieses Verhalten gefunden werden.

In den Abbildungen 22 und 23 wird jeweils die durchschnittliche Anzahl an durchgeführten Unterstützungen von AUS bzw. GER bei zehn Spielen mit einem bestimmten Schwellenwert dargestellt. Auch hier weisen die Kurven für die gesamten Unterstützungen einen hohen Wert bei einem Schwellenwert von 1.5 auf. Ein eher unerfreuliches Ergebnis bildet die, unabhängig von den Schwellenwerten, geringe Anzahl an Unterstützungen von AUS an GER, und dass GER überhaupt keine Unterstützung für AUS durchführt.

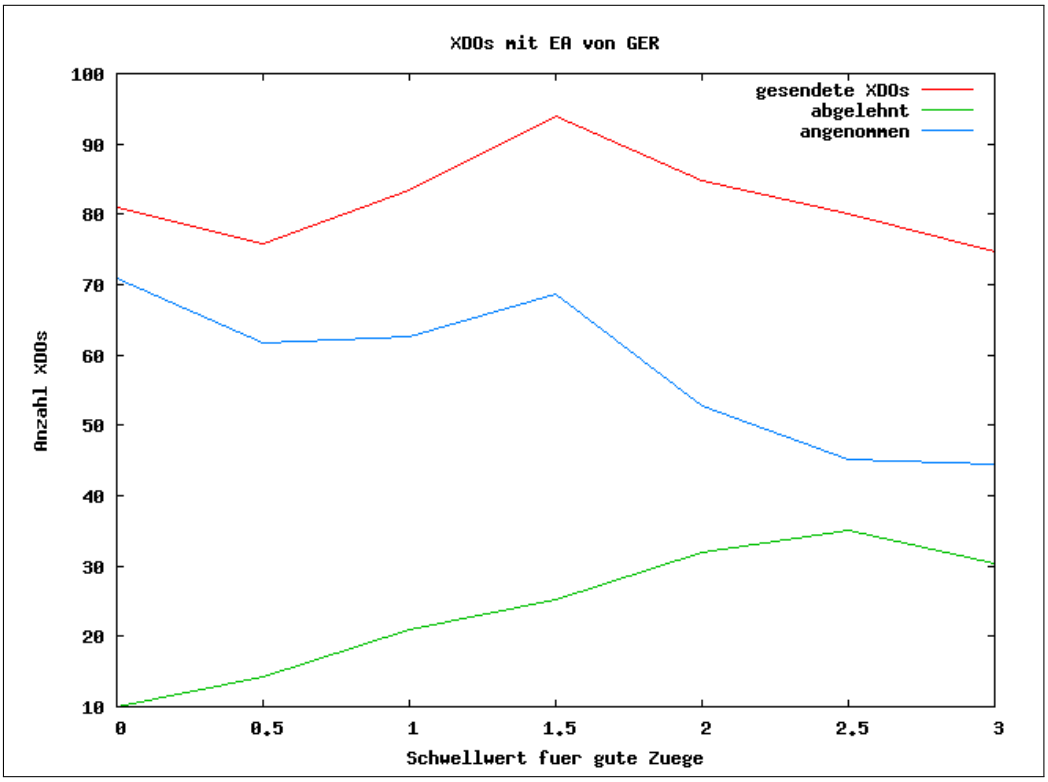


Abbildung 21: Von GER gesendete Zugvorschläge

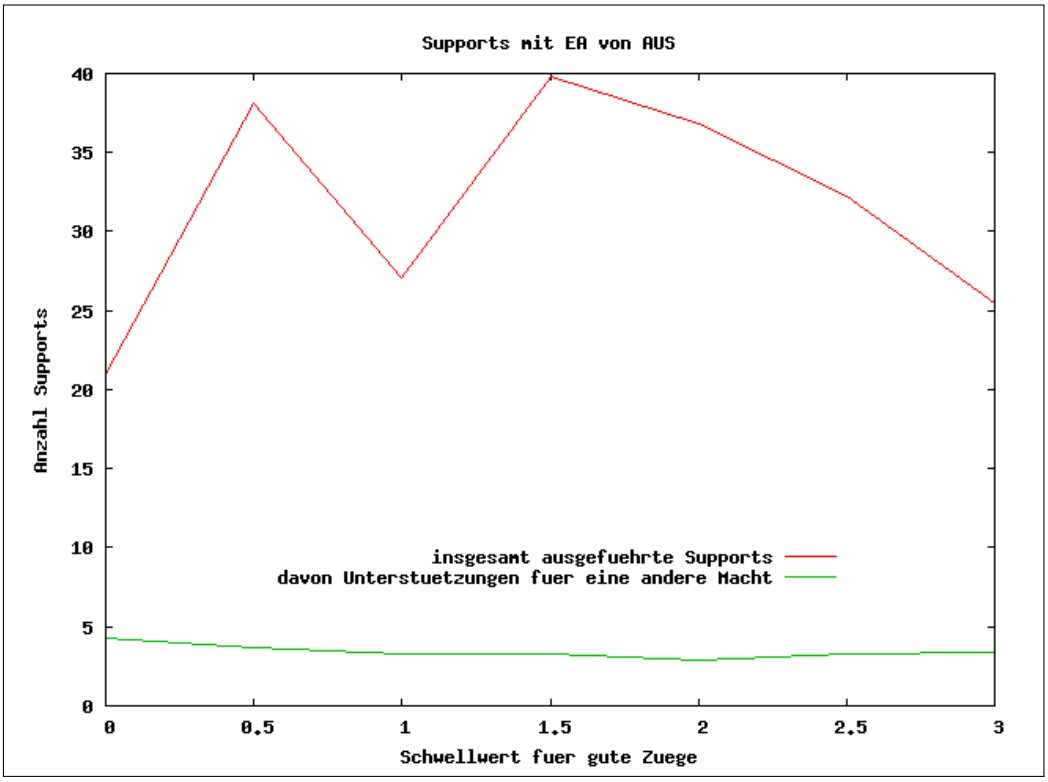


Abbildung 22: Von AUS ausgeführte Supports

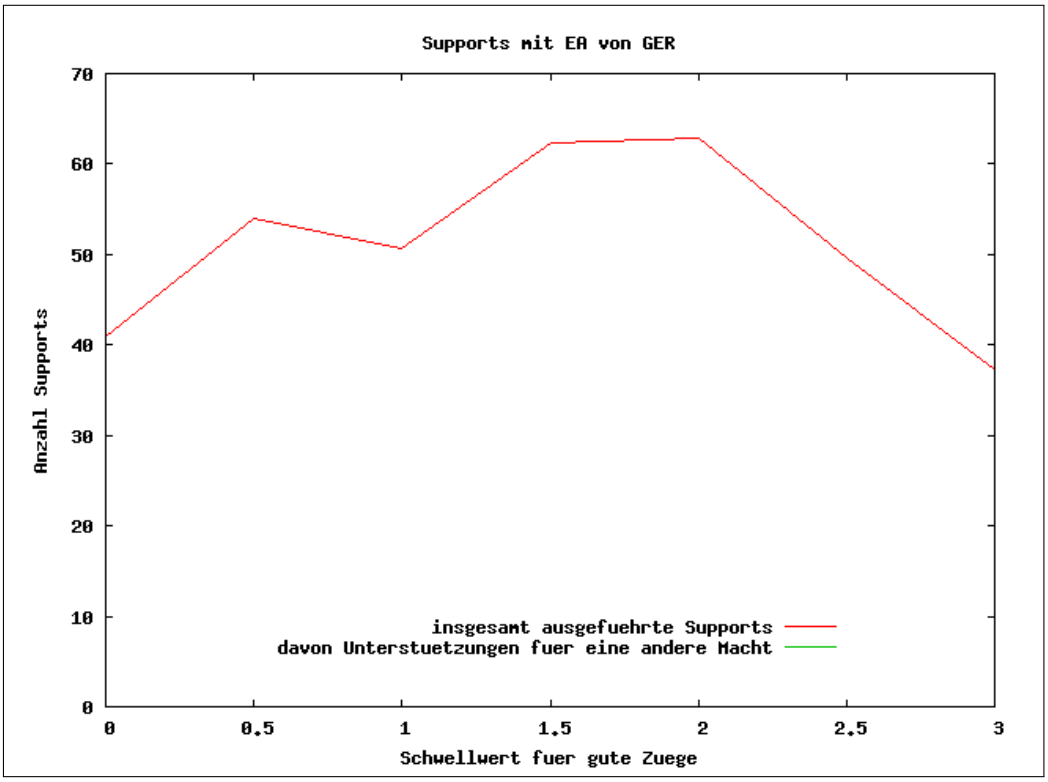


Abbildung 23: Von GER ausgeführte Supports

Schwellwert für gute Züge	Mistrauen	Neutral	Vertrauen	vollstes Vertrauen
0.0	0	0	0	10
0.5	1	0	0	10
1.0	0	0	1	9
1.5	0	0	0	10
2.0	2	0	3	5
2.5	3	1	0	6
3.0	0	3	0	7

Tabelle 5.12: Vertrauen von GER zu AUS

Die Tabelle 5.11 zeigt aus der Sicht von AUS, in wievielen Spielen bei einem bestimmten Schwellenwert GER misstraut wurde, GER neutral eingestuft wurde, GER vertraut wurde und GER vollkommen vertraut wurde. Die Haltung von AUS zu GER bezieht sich immer auf das Ende eines Spieles und berücksichtigt nicht den Spielverlauf. Tabelle 5.12 zeigt den selben Sachverhalt aus der Sicht von GER. Deutlich zu erkennen ist bei beiden, dass je höher der Schwellenwert für gute Züge ist, desto weniger wird der anderen Macht vollkommen vertraut. Je höher die Schwellenwerte sind, desto mehr wird dazu tendiert, der anderen Macht zu misstrauen. Dieses Verhalten ist nicht verwunderlich. Denn immer wenn ein *Stragotiator* einen Zugvorschlag erhält, den er als schlecht einstuft, sinkt das Vertrauen in den Sender. Weil ein hoher Schwellenwert somit bedeutet, dass ein Zug eine hohe Fitness benötigt, um als gut zu gelten, ist es nur verständlich, dass bei einem hohen Schwellenwert die Anzahl der Spiele zurückgeht, bei denen dem anderen *Stragotiator* vollkommen vertraut wird, und die Anzahl an Spielen steigt, bei denen ihm misstraut wird. Die Schwellenwerte haben keine Auswirkungen auf die Züge, die vom Strategiekern bzw. dem EA berechnet werden. Somit sind auch die Zugvorschläge, die ein *Stragotiator* einer anderen Macht unterbreitet, unabhängig von diesem Schwellwert. Er bestimmt lediglich, ab wann ein Zugvorschlag akzeptiert wird.

Für den anschließenden Contest und den Turingtest haben wir uns für einen Schwellenwert von 1.5 entschieden, weil bei diesem die Zusammenarbeit, gemessen an den gesendeten und angenommenen Zugvorschlägen, besonders intensiv ist. Wichtig an dieser Stelle ist auch, dass bei diesem Schwellenwert nicht alle Vorschläge angenommen wurden. Denn das Verhalten soll menschlich erscheinen, und ein Mensch nimmt nie alle Vorschläge an. Es wird im Laufe eines Spieles immer Situationen geben, wo man eine andere Meinung hat und daher andere Züge durchführen möchte. Ein weiterer Punkt für die Entscheidung zu diesem Schwellenwert war es, dass die Anzahl an Spielen, bei denen sich die *Stragotiatoren* gegenseitig vollkommen vertraut haben, recht hoch war. Ab einem Wert von 2.0 ist diese Anzahl, gerade aus Sicht von GER, sehr stark gesunken, weswegen ein höherer Schwellenwert nicht in Frage kam.

5.2.3.2 Strategie Kern

Die in Abschnitt 5.2.2.2 ab Seite 59 vorgestellte Realisierung des *Strategie Kerns* wäre eigentlich parameterlos. Alle benötigten Informationen wurden entweder durch den *Negotiator* oder durch die Karte gegeben. Es stellte sich allerdings heraus, dass die Knotensortierung, unter dem Gesichtspunkt, dass sie nur auf den durch die Karte gegeben Informationen basiert, kaum taktische Elemente berücksichtigt. Da die taktische Komponente allerdings durch den *Strategie Kern* des *Stragotiators* realisiert werden sollte, war eine Einführung von Parametern unumgänglich. Es stellte sich nun weiter die Frage, ob die Parameter der Karte angepasst werden sollten, oder ob eher jede Provinz einen individuellen Wert erhalten sollte. Als Zeitpunkt für die Vergabe von neuen Werten für die einzelnen Provinzen hat sich dann der Aufruf der Knotensortierung als geeignet herausgestellt. An dieser Stelle ist nämlich klar, welche Provinzen für eine bestimmte Einheit erreichbar sind. Hier sollte also auch entschieden werden, in welcher Reihenfolge die Provinzen auf mögliche Züge dorthin überprüft werden sollen. Um die Sortierung möglichst einfach zu halten, wurde zum Wert, den eine Provinz hat, nach verschiedenen Merkmalen eine bestimmte Konstante addiert (für genaue Werte: siehe 5.2.2.2.2 ab Seite 60).

Interessant war dann die Frage, inwieweit die gewählten Parameter überhaupt sinnvoll waren, oder ob es andere Parameter gibt, die den *Stragotiator* zu einem stärkeren oder vielleicht sogar menschlicheren Spieler machen. Aus diesem Grund wurde ein Parametertest durchgeführt. Da es acht verschiedene veränderliche Parameter gab, Zeit und Rechenleistung allerdings begrenzt waren, entschieden wir uns dazu, jeden Parameter nur mit zwei bis drei verschiedenen Ausprägungen zu testen. Daraus folgten also

$2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^8 = 256$ bis $3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 3^8 = 6561$ Möglichkeiten. Dazu kam allerdings noch die Überlegung, dass es einen großen Unterschied darstellen könnte, für welche Großmacht die Parameter getestet werden. Beide Grenzen würden sich also noch um den Faktor sieben vergrößern. Damit liegt die Anzahl möglicher Kombinationen zwischen 1792 und 45927. Unter der Annahme, dass ein einziges Testspiel für jede Kombination nicht ausreichend ist und die minimale Anzahl von Spielen für einen gängigen statistischen Test bei sechs liegt, vergrößerte sich die Gesamtanzahl der Spiele auf 10752 bis 275562. Schlussendlich wurden folgende Werte als realistisch testbar vereinbart:

1. gehört die Provinz einem Feind: $25, 50, 75$
2. gehört die Provinz einer neutralen Macht: $25, 50, 75$
3. befindet sich auf der Zielprovinz ein Versorgungszentrum
   a) und es ist noch unbesetzt: $100, 150$
   b) es ist nicht von einer Einheit besetzt und gehört auch nicht der angreifenden Macht: $15, 25$
   c) gehört nicht der angreifenden Macht und die Zielprovinz ist die Ausgangsprovinz: $15, 25$
   d) gehört nicht der angreifenden Macht und die Zielprovinz ist nicht die Ausgangsprovinz: $100, 200$
   e) und es befindet sich bereits im Besitz der angreifenden Macht: $-500, 0$
4. gibt es Einheiten, die die Zielprovinz angreifen oder halten wollen und Unterstützung benötigen, weil mehr als eine feindliche Einheit angreifen könnte: $50, 100, 150$

Es ergaben sich also $3 \cdot 3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 = 864$ verschiedene Kombinationen, die für jede der sieben Großmächte in fünf Partien getestet werden sollte (insgesamt 30240 Spiele). Da die Laufzeit der Parametertests stark beschränkt war und die Gegner auch nicht zu stark sein durften, entschieden wir uns für Spiele, die nur bis in das Jahr 1910 reichten und wählten als Gegner sechs Randbots, deren Handeln randomisiert ist. Mit Rundenzeiten von je einer Sekunde für Bewegung-, Rückzug- und Bauphasen dauerte somit eine Jahr im Spiel maximal fünf Sekunden und zehn Jahre ließen sich innerhalb von 50 Sekunden durchspielen. Ein Computer hätte also für die reine Simulation aller 30240 Spiele 17,5 Tage gebraucht. Da wir allerdings den Parametertest auf vielen Rechner gleichzeitig laufen lassen konnten, verringerte sich die Zeit auf weniger als zwei Tage.

Die erste Auswertung der Parameter war sehr ernüchternd. Ausgewertet wurde die Anzahl von Siegen und Niederlagen innerhalb von zehn Jahren. Ein Muster in den Ergebnissen war nicht erkennbar, da ein Sieg innerhalb von zehn Jahren in 30240 Spielen bei 6048 verschiedenen Kombinationen so selten auftrat, dass man nicht einzelne Parameter dafür verantwortlich machen konnte.

Bei der zweiten Auswertung der Tests erhofften wir uns ein besseres Ergebnis, da nun nicht anhand von Sieg oder Niederlage ausgewertet wurden, sondern anhand der Anzahl der Versorgungszentren nach zehn Jahren Spielzeit. In Tabelle 5.13 auf Seite 77 sind in einer Tabelle einige Kombinationen der Werte erfasst sowie die Anzahl der durchschnittlichen Versorgungszentren nach zehn Jahren.

In der Tabelle wird sowohl der beste Wert mit durchschnittlich 11 Versorgungszentren als auch der schlechteste mit durchschnittlich 9,8 abgebildet. Alle anderen Ergebnisse befinden sich in diesem Intervall. Da bei 6048 Kombinationen weder ein besserer, noch ein schlechterer Wert auftraten, wurden für den Contest (vgl. 6.2 ab Seite 115) die ursprünglichen Parameter wie in 5.2.2.2 auf Seite 59 verwendet. Letztendlich müssen

1.	2.	3.a)	3.b)	3.c)	3.d)	3.e)	4.	∅ # Vers.-Zentr.
25	25	100	15	15	100	0	50	11
25	50	150	15	25	100	-500	50	9,6
25	75	100	15	15	200	0	50	11
50	25	150	15	25	200	-500	50	9,8
50	50	100	15	15	100	0	100	10
50	75	150	25	25	100	-500	100	10,2
50	25	100	25	15	200	0	100	11
75	50	150	25	25	200	-500	150	11
75	75	100	25	15	100	0	150	9,8
75	50	150	25	25	200	-500	150	10,8

Tabelle 5.13: Parametertest *Stragotiator*: *Strategie Kern*

wir leider davon ausgehen, dass es sich bei den Ergebnissen um Rauschen handelt, da die Testmenge zu groß und die Anzahl der Wiederholungen zu klein war.

5.2.3.3 EA-Optimierungstests

Ein EA ist ein Black-Box-Optimierungsverfahren. Das heißt, dass es ohne Kenntnisse über die Strukur des Problems dieses optimieren kann. Um die Funktionsweise und Qualität des implementierten EA zu überprüfen, wurden klassische Benchmarkprobleme für Evolutionäre Algorithmen untersucht.

Ein solches Problem wird in der Literatur als *Sphere* [Rec73, DJ75] bezeichnet und hat die zu minimierende Zielfunktion.

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Die Individuen $x \in \mathbb{R}^n$ werden dabei oft auf den Bereich $-5,12 \leq x_i \leq 5,12$ für alle x_i beschränkt. Die Funktion besitzt nur ein lokales - und damit auch globales - Optimum im Punkt $(0,\ldots,0) \in \mathbb{R}^n$ mit $f(0,\ldots,0) = 0$. Zudem ist sie separierbar, was bedeutet, dass jede Komponente x_i unabhängig von den anderen optimiert werden kann.

Ein weiteres Benchmarkproblem ist *Schwefels Doppelsumme* [Sch95] oder auch *Schwefel 1.2* genannt. Dessen Zielfunktion ist zu minimieren.

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

Die Individuen $x \in \mathbb{R}^n$ werden durch einen Wertebereich $-65,536 \leq x_i \leq 65,536$ für alle x_i beschränkt. Die Funktion ist nicht separierbar und besitzt das einzige lokale Optimum im Koordinatenursprung mit Zielfunktionswert 0.

Als letztes untersuchtes Problem wird hier die *Rosenbrock*-Funktion vorgestellt, welche ebenfalls ein Minimierungsproblem ist.

$$f(x) = \sum_{i=1}^{n-1} \left(100 \cdot \left(x_i^2 - x_{i+1} \right)^2 + (1 - x_i)^2 \right)$$

ist nicht separierbar und besitzt ein globales Optimum im Punkt $(1,\ldots,1) \in \mathbb{R}^n$ mit Funktionswert 0. Alle Individuen für diese Funktion bewegen sich im Bereich $-2,048 \leq x_i \leq 2,048$ für alle x_i.

Zum Vergleich wurden Konfigurationen aus [Han06] entnommen. Darin wurden die verschiedene reellwertigen Benchmarkfunktionen jeweils mit 10 und 30 Dimensionen getestet. Für 10 Dimensionen war die Anzahl der Funktionsauswertungen auf 100.000 und für 30 Dimensionen auf 300.000 beschränkt. Jedes Problem wurde 25 mal mit einem Verfahren optimiert, um die Ergebnisse zu mitteln.

Bei allen Problemen wurden die Parameter des EA konstant gehalten. Die Anzahl der Elter- und der Kindindividuen beträgt immer $\mu = 100$ und $\lambda = 700$. Grundsätzlich wird eine Standardabweichung von $\sigma = 0.02$ und eine Rekombinationswahrscheinlichkeit von 0.3 verwendet. Mutation und Rekombination werden durch Gaußmutation bzw. arithmetischer Rekombination realisiert. Eine Anpassung der Schrittweite erfolgte in einigen Fällen durch Selbstadaption. Um die Begrenzung der Funktionsauswertungen einzuhalten wurde die Generationsanzahl parametrisiert. Für $n = 10$ Dimensionen wurde ein Maximalwert von 140 festgelegt. Für $n = 30$ Dimensionen lag die Grenze bei 425.

Das Problem *Sphere* lässt sich mit dem EA effizient lösen. Mit $n = 10$ Dimensionen wird bei jedem Ausführen des EA das globale Optimum mit 90 Generationen ($100+700 \cdot 90 = 63100$ Funktionsauswertungen) gefunden. Beschränkt man den EA auf 85 Generationen pro Durchlauf, wird das Optimum ausreichend genau gefunden (mit durchschnittlichen Funktionswert $1.33 \cdot 10^{-}315$ und einer Varianz von 0). Für $n = 30$ Dimensionen finden alle EA-Durchläufe nach 100 Generationen das globale Minimum.

Bei *Schwefels Doppelsumme* sieht man bei den Durchläufen mit $n = 10$ Dimensionen ähnliche Ergebnisse, da ebenfalls mit 90 Generationen das globale Optimum in jedem EA-Durchlauf gefunden wird. Mit 85 Generationen wird das Ergebnis leicht schlechter als bei *Sphere*, da durchschnittlich der Funktionswert bei $4.35 \cdot 10^{-293}$ mit einer Varianz von 0 liegt. Mit $n = 30$ Dimensionen wird bei 90 Generationen ein durchschnittlicher Funktionswert von $4.9 \cdot 10^{-324}$ ermittelt, der nah genug am Optimum liegt. Auch hier liegt die Varianz bei 0.

Anders sieht es bei der *Rosenbrock-Funktion* aus. Mit $n = 10$ Dimensionen ist das Ergebnis bei einer Begrenzung von 100 Generationen durchschnittlich $7,891$ mit einer Varianz von $0,9128$. Eine Erhöhung auf die maximale Generationszahl von 140 bringt nur eine leichte Verbesserung (durchschnittliche Fittness liegt bei $7,2275$ mit Varianz $1,0338$). Auch eine Schrittweitenanpassung mittels *Selbstadaption* (zusätzlich zu der Evolution mit 140 Generationen) erbringt keine deutliche Näherung an das Optimum. Die durchschnittliche Fitness beträgt 7.9777 mit Varianz 0.9848. Ähnlich schwierig ist das Problem mit $n = 30$ Dimensionen. Es lässt sich mit maximaler Generationszahl von 425 und einer adaptiven Schrittweitenanpassung nur eine durchschnittliche Fitness von $36,5931$ ermitteln. Die Varianz liegt bei $19,1685$.

Zwei von drei Testfunktionen wurden bei den EA-Tests zufriedenstellend gelöst. Jedoch konnte bei allen Tests ein Verhalten, welches den Zielfunktionswert verbessert, beobachtet werden.

5.2.4 Fazit

Die für den *Stragotiator* verantwortliche Teilgruppe kann mit ihren Ergebnissen sehr zufrieden sein. Obwohl wir keine Ergebnisse des Contests (vgl. 6.3 auf Seite 116) oder des Turing Test (vgl. 6.4 auf Seite 116) vorweg nehmen wollen, möchten wir trotzdem auf die wesentliche Aufgabestellung dieser Projektgruppe eingehen. Titel und gleichwohl Hauptaugenmerk der Projektgruppe ist „Spielercharaktere - Modellierung menschenähnlicher Gegenspieler in Strategiespielen mit Techniken der Computational Intelligence“. Bei der Entwicklung des *Stragotiators* war somit die Modellierung eines menschenähnlichen Gegenspielers ein wesentlicher Aspekt. Im Contest mussten wir leider feststellen, dass unser Bot nicht sehr spielstark ist. Die Tatsache dafür liegt aber

nicht darin, dass wir einen falschen Ansatz gewählt haben, sondern in der kurz nach der Literaturphase getroffenen Entscheidung, den Klassiker (siehe „Der Klassiker“ in 5.2.1.1.1 auf Seite 42) zu implementieren. Somit war uns schon zu Anfang bewusst, dass unser Bot anderen Ansätzen unterlegen wäre, was das Streben nach einem Sieg anging. Im anschließenden Turing Test zog der *Stragotiator* allerdings alle Register und verblüffte sämtliche Beobachter und Spieler. Er war der einzige Bot, von dem Testspieler glaubten, dass sie gegen einen Menschen spielen würden. Von zehn Testspielern hatten drei Spieler das Gefühl, ihr Gegner sei ein Mensch. Und auch die Beobachter täuschte er am besten. Elf mal glaubten verschiedene Beobachter, dass eine bestimmte Macht von einem Menschen gespielt wurde, obwohl der *Stragotiator* die Entscheidungen traf. Nach Aussagen der Beobachter wurde hinter den anderen beiden Bots insgesamt nur sieben mal ein Mensch vermutet. Aus diesem Grund hebt sich der *Stragotiator*, was die Modellierung des menschenähnlichen Gegenspielers angeht, weit von den anderen entwickelten Bots ab und die Teilgruppe sieht ihre Aufgabe als erfüllt an.

5.2.5 Ausblick

Bereits während der Planung vermuteten wir, dass nicht alle unsere Ideen in der zur Verfügung stehenden Zeit umgesetzt werden konnten. Zudem stellte sich bei der Realisierung und den verschiedenen Tests noch weiteres Verbesserungspotential heraus. Im Folgenden werden wir auf die verschiedenen Ansätze zur Optimierung näher eingehen.

5.2.5.1 Negotiator

Neben den nicht implementierten Verhandlungsmöglichkeiten (SLO, NOT, DMZ und alle Befehle höher Level 20 fehlen) bietet vor allem das Vertrauensmodell ein großes Potential. Die jetzige Implementierung partitioniert lediglich einen festzulegenden Bereich der reellen Zahlen mit *Fuzzy Mengen*. Diese *linguistische Variable* könnte in einem Fuzzycontroller ausgewertet werdet.

Für den modellierten Klassiker war es wichtig, dass Absprachen eingehalten und Verbündete nicht belogen werden. Diese Einschränkung gilt jedoch nicht für seine Feinde. Das Belügen von feindlichen Mächten könnte bei XDOs eingesetzt werden. In der aktuellen Implementierung wird zuerst überprüft, ob schon entschieden wurde, dass der Zug ausgeführt wird oder nicht. Wenn die Entscheidung bereits getroffen wurde, wird dem Verhandlungspartner wahrheitsgemäß geantwortet. Mächte, mit denen kein Bündnis besteht, könnten an dieser Stelle taktisch belogen werden.

Bei eingegangenen Allianzen wird nur gespeichert, dass mit einer Macht eine Allianz besteht. Es wird nicht gespeichert, welche Mächte noch an einer speziellen Allianz beteiligt sind oder gegen welche Mächte die Allianz geschlossen wurde. Diese Informationen könnten bei der Zugplanung und bei Zugvorschlägen berücksichtigt werden, um einem Allianzpartner nur Angriffe gegen einen gemeinsamen Feind vorzuschlagen.

Lehnt eine andere Macht eine Allianz oder ein Unentschieden ab, wird darauf nicht reagiert, weil der Grund für die Ablehnung nicht direkt bestimmbar ist. Es wäre eine Verbesserung, wenn der Grund für die Antwort durch weitere Verhandlung herausgefunden würde. Bei einer Allianz würde das zu Informationen führen, ob eine Macht mit einer anderen nicht zusammenarbeiten oder sie nicht bekämpfen möchte. Diese Informationen könnten sowohl für spätere Verhandlungen nützlich sein, wie auch bei der Zugberechnung berücksichtigt werden. So müssten keine Verhandlungen geführt werden, bei denen eine Ablehnung wahrscheinlich ist. Die Taktik könnte auf die Verhältnisse der anderen Mächte untereinander abgestimmt werden, um zum Beispiel Kriege zwischen zwei Mächten auszunutzen.

Die Behandlung von BWX und HUH Antworten sind ebenfalls nicht ausreichend implementiert. Diese Antworten werden behandelt wie Absagen. Das liegt daran, dass es im Falle eines Zugvorschlages keinen Unterschied darstellt, weil der vorgeschlagene Zug von der anderen Macht in jedem Fall nicht zugesichert wurde. Auch bei einem Friedensangebot birgt die aktuelle Implementierung kein Problem. Es werden nur Nachrichten gesendet, die Frieden zwischen dem Stragotiator und dem Empfänger anbieten. Somit kann eine Antwort, dass es den Empfänger nichts anginge, auch als Ablehnung interpretiert werden. Und mit einem Bot, der keine Verhandlung führen kann und ein HUH zurück sendet, kann nie ein Friedensvertrag zustande kommen. Tritt bei der Zugberechnung der Fall auf, dass sowohl eine neutrale wie auch eine feindliche Macht angegriffen werden kann, entscheidet sich der Strategiekern für den Angriff auf die feindliche. Dies sollte aber, bezogen auf die Behandlung der Ablehnung, nicht zu einem unmenschlicherem Verhalten führen.

5.2.5.2 Strategie Kern

Die Frage, wann ein Angriff Erfolg haben wird und wann nicht, ist nicht einfach zu entscheiden. Wenn eine gegnerische Einheit, die keine Unterstützung erhalten kann, von mehr als einer Einheit angegriffen wird - eine Einheit führt eine Attacke aus und wird von anderen unterstützt - ist die Frage leicht zu beantworten. Doch wie sieht es aus, wenn sich an einer Front je drei Einheiten von jeder Macht gegenüberstehen. Oder wenn es mehr als zwei Mächte sind, die Allianzen geschlossen haben. Der Erfolg eines Angriffes hängt nicht nur von der Anzahl an beteiligten Einheiten ab. Auch die Psychologie spielt eine Rolle. Wenn ein Gegner denkt, dass er von vielen Einheiten angegriffen wird, könnte er sich zurückziehen, um in eine besser zu verteidigende Position zu gelangen. In so einem Szenario könnte der Angriff mit wenigen Einheiten - vielleicht mit nur einer - ausgeführt werden, während die restlichen andere Ziele verfolgen.

Der Erfolg oder Misserfolg eines Angriffes hängt von folgenden Faktoren ab:

Anzahl der angreifenden Einheiten
: Desto mehr Einheiten einen Angriff ausführen, desto höher ist die Erfolgswahrscheinlichkeit.

Anzahl der angreifenden Einheiten unter eigenem Kommando
: Es muss zwischen eigenen und alliierten Einheiten unterschieden werden. Denn ein Alliierter könnte seine Zusage nicht einhalten. Nur bei den Einheiten, die man selber befehligt, kann man sicher sein, was sie tun.

Maximale Anzahl von Einheiten die angreifen könnten
: Wieviele Einheiten den Angriff tatsächlich ausführen, kann der Gegner nicht wissen. Sicher ist, wenn er einer Übermacht gegenübersteht, liegt der Ausgang nicht in seinen Händen. Er hängt hauptsächlich davon ab, mit wievielen Einheiten der Angriff ausgeführt wird. Der Gegner muss damit rechnen, dass mit allen zur Verfügung stehenden Mitteln angegriffen wird.

Maximale Anzahl von Einheiten unter eigenem Kommando die angreifen könnten
: Auch der Gegner weiß, dass Allianzen in *Diplomacy* nicht immer eingehalten werden. Somit ist eine große Anzahl von Einheiten einer Macht besonders imposant.

Anzahl potenzieller Verteidiger
: Auch als Angreifer weiß man nicht, wieviele Einheiten tatsächlich verteidigen. Man muss damit rechnen, dass der Gegner alle verfügbaren Einheiten zur Verteidigung benutzt.

Befindet sich in der Zielprovinz bereit eine Einheit
Es macht einen Unterschied, ob die Zielprovinz bereits von einer gegnerischen Einheit annektiert wurde oder ob sie an der Grenze zu einer gegnerischen Provinz liegt. In letzterem Fall müsste der Gegner seine Einheiten erst in diese Provinz ordern.

Anzahl möglicher gegnerischer Angriffe auf unterstützende Einheiten
Wird eine unterstützende Einheit angegriffen, wird die Unterstützung gemäß dem *Diplomacy* Regelwerk unterbrochen. Das muss berücksichtigt werden, weil der Gegner mit Einheiten, die nicht in der Nähe der Zielprovinz stehen, diese nicht verteidigen kann. Er kann aber, wenn er genug Unterstützungen verhindert, den Angriff schwächen.

Es ist nicht bekannt, wie stark der Einfluss der oben genannten Faktoren auf den Ausgang eines Angriffs ist, insbesondere konnten wir keine Literatur darüber finden. Deswegen kam die Idee auf, ein *Neuronales Netz* zu benutzen. Als Eingabe für das Netz sollten die sieben oben genannten Faktoren dienen. Als Ausgabe wäre sowohl eine binäre Antwort (1: angreifen - 0: nicht angreifen) wie auch eine Erfolgswahrscheinlichkeit denkbar. Weil das Netz aus Zeitgründen im *Stragotiator* nicht zum Einsatz kam, können wir leider keine Aussage über eine gute Struktur geben. Lediglich für das Anlernen des *Neuronalen Netzes* mit einer binären Ausgabe gab es eine Idee.

Es sollte online lernen um das Verhalten eines Menschen nachzuahmen, der erst mit der Zeit über mehrere Spiele hinweg lernt, was ein gutes Verhalten ist. Weil ein Zug, für den das Netz eine Null ausgibt, nicht ausgeführt wird, kann die Ausgabe nicht evaluiert werden. Also sollten die Kantengewichte zu Beginn so gewählt werden, dass immer eine eins berechnet wird. Bei einer fälschlich berechneten eins sollte das Netz mit *Backpropagation* lernen. Um nachzubilden, dass ein Mensch sich irgendwann seine Meinung gebildet und in seinem Verhalten festgelegt ist, sollte eine Lernrate benutzt werden, die mit jedem Lernenvorgang verkleinert wird. Zusätzlich wäre eine maximale Rundenzahl denkbar, die bestimmt, wieviele Runden maximal gelernt werden soll. Durch diese beiden Maßnahmen könnte verhindert werden, dass das Netz so lange lernt, bis es immer eine Null berechnet. Leider wäre es aber möglich, dass der Lernvorgang abgebrochen würde, bevor gute Kantengewichte gefunden wurden.

Bei Testspielen ist aufgefallen, dass der *Stragotiator* teilweise einen Angriff immer wieder ausführt, der nie erfolgreich ist. Das kann zu einem Stillstand im Spiel führen. Der Grund für dieses Verhalten ist, dass der Strategiekern kein Gedächtnis hat. Er weiß nicht, welchen Zug er schon in der vorhergegangenen Runde erfolglos ausgeführt hat. Dieses Verhalten könnte durch den Einsatz des oben beschriebenen *Neuronalen Netzes* behoben werden. Es würde lernen, dass der Angriff nicht erfolgreich ist und der Strategiekern würde auf Grund dieser Prognose einen anderen Zug berechnen.

Außerdem stellte sich heraus, dass gerade in der Bewertung von Zügen ein großes Potential steckt. Da der *EA* die Zugbewertung als Zielfunktion nutzt, wäre die Optimierung der Zugbewertung gleichzeitig ein möglicher Schritt zu einer Verbesserung der Resultate des *EAs*. Nicht nur der *EA* würde von einer Verbesserung profitieren, auch die Bewertung der gegnerischen Spieler nutzt diese Funktion und könnte somit die Stärke und Menschlichkeit der Mitspieler besser einschätzen. Konkret ist die Modellierung der Zugbewertung in der Realisierung eher grob gestaltet worden. In 5.2.2.2.5 auf Seite 65 wird beschrieben, wie die Züge anhand des Typs und mit Hilfe der umliegenden Einheiten bewertet werden. Da die Zugbewertung für den *Strategie Kern* selber eher eine untergeordnete Rolle darstellte, wurde sie mit wenig Aufwand realisiert.

Eine deutliche Verbesserung könnte auch die Änderung der Bewertung von Provinzen bringen. Die realisierte Implementierung betrachtet immer nur die betreffende Provinz nämlich, ob die Provinz ein Versorgungszentrum enthält und die sich um und in der

Provinz aufhaltenden Einheiten. Würde die Bewertung auch weiter entfernte Provinzen betrachten, dann wäre eine Globalstrategie realisierbar gewesen.

Eine Schwäche des *Strategie Kerns* ist die Tatsache, dass er vollständig deterministisch spielt und nicht die Vergangenheit kennt. Er kann sich nicht mehr daran erinnern, wenn er in einer vorherigen Runde eine Fehler gemacht hat, weil ein Zug aufgrund von gegnerischen Einheiten nicht ausgeführt werden konnte. Unter den gleichen Umständen würde er den gleichen Fehler erneut machen. Somit kann es bei einem hartnäckigen Gegner auf dem Spielfeld zu einem Patt kommen. In einem gewissen Bereich kann sich keine Einheit mehr bewegen, weil sie immer den gleichen Zug ausführen will, der aber durch andere Einheiten verhindert wird. Das Problem ließe sich nur dadurch beheben, dass die Karte, auf der alle Module arbeiten, geklont werden würde und der *Strategie Kern* alle vergangenen Karten und die gemachten Züge überblicken könnte. Dabei ist aber auch zu beachten, dass damit sämtliche Funktionen im *Strategie Kern* deutlich komplexer werden würden.

5.2.5.3 EA-Optimierung

Um die Zugoptimierung mittels EA noch weiter zu verbessern, gibt es die Möglichkeit, Convoy-Züge (*Move_by_convoy*) explizit zu untersuchen. Dazu müsste man bei jeder Zugoptimierung einmalig untersuchen, ob eine Armee mittels Convoy ziehen kann. Dafür müsste überprüft werden, ob die Armee in einer Provinz mit einer Küste steht, um anschließend mittels Tiefensuche (DFS) von dieser Küste adjazente Schiffseinheiten zu suchen. In dem DFS-Baum ließen sich dann die per Convoy erreichbaren Küstenprovinzen finden. Zusätzlich müsste noch die Zugmutation für Schiffe dahingehend verändert werden, dass bei allen Flotten, die in den DFS-Bäumen vorkommen (plural, da ein DFS-Baum immer nur die Convoy-Möglichkeiten einer Armee anzeigt) beispielsweise mittels eines Flags kenntlich gemacht werden, dass sie Einheiten einen Convoy ermöglichen. Für alle anderen Flotten würde die Mutation den Zug *Convoy* nicht ermöglichen.

Weiteres Verbesserungspotential liegt in der Bewertung der Provinzen. Provinzen werden in der *Map*-Klasse als Knoten gespeichert. Dazu wird noch ein Gewicht für jeden Knoten gespeichert, welcher die Wichtigkeit dieser Provinz widerspiegelt. Der EA optimiert die Züge, so dass die Einheiten möglichst viele Provincen einnehmen, welche ein hohes Gewicht haben (die Gewichte werden zusätzlich noch mit Faktoren multipliziert, die davon abhängen, wie die Einheit auf besagte Felder kommt - einfacher Zug, mit Unterstützung, *Hold*. Siehe dazu Abschnitt 5.2.2.2.5). Ein Problem bei diesem Vorgehen ist das Ignorieren von weit entfernten wertvollen Provinzen (beispielsweise nicht besetzte Versorgungszentren, welche in zwei Runden erreichbar wären). Um diesem Problem zu begegnen würde sich ein Breitensuche-Baum (BFS-Baum) einer Tiefe k anbieten, welcher in den Baum-Knoten die Knotengewichte (durch die Karte festgelegt) seiner Kinder summiert. Die Motivation dabei ist das Finden von stark gewichteten Knoten. Je stärker ein Knoten gewichtet ist, desto interessanter ist dieser. Wenn ein Knoten in dem BFS-Baum stark gewichtete Kinder hat, erhält er zu seinem eigenen Knotengewicht noch das Gewicht seiner Kinder.

Als Beispiel stehe eine Arme in der Provinz Bre, und es wird nun ein BFS-Baum der Tiefe $k = 2$ berechnet. Die Knoten (BRE, GAS, PAR, PIC, SPA, MAR, BUR, BEL) haben die Knotengewichte (5, 0, 5, 0, 15, 5, 0,15). Einen normalen BFS-Baum mit den Knotengewichten der Karte sieht man in Abbildung 24.

Nun wird zu jedem Baumknoten, der kein Blatt und nicht die Wurzel ist, das Gewicht seiner Kinder addiert.

Man sieht in Abbildung 25, dass der Knotenwert von GAS sich von 0 auf 20 erhöht hat. Die Idee ist, dass Provinzen, über die man interessante (stark gewichtete) Provinzen

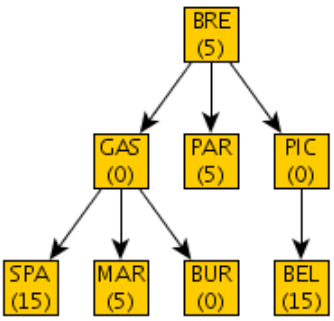


Abbildung 24: normaler BFS-Baum mit Knotengewichten

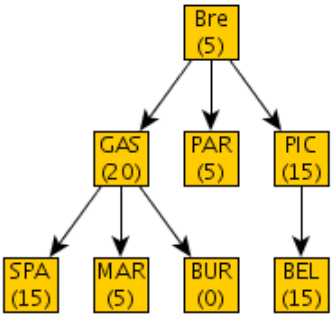


Abbildung 25: BFS-Baum mit kummulierten Knotengewichten für alle nicht-Wurzelknoten

erreichen kann, wiederrum durch ein stärkeres Gewicht interessanter werden. In diesem Beispiel wird GAS interessanter, da über diese Provinz SPA und MAR erreichbar sind. Da eine Einheit bereits auf der Wurzel steht, soll diese nicht durch die Gewichte der Kindknoten profitieren. Ansonsten würde in der Wurzel die Summe über alle Knotengewichte gebildet werden, welches offensichtlich den höchsten Wert im Baum bildet. Das hätte zur Folge, dass die Einheit, die im Wurzelknoten steht, dort bleibt. Durch das Erstellen eines BFS-Baumes wird garantiert, dass nur die Provinzen interessant werden, welche sich auf einem kürzesten Weg zu einem stark gewichteten Knoten befinden. Somit würden Mutationen, welche die Einheit von der Province Bre nach Gas ziehen lassen, vor anderen Zugmutationen bevorzugt werden.

5.2.5.4 Gegnerbewertung

Die Gegnerbewertung kann leicht durch folgende Maßnahmen verbessert werden:

Umsetzung der verbliebenen Ideen aus dem entsprechenden Paper von Loeb. Damit sollte es schwerer werden, Freundschaft oder Feindschaft vorzutäuschen. Ebenfalls hat die Freund/Feind-Matrix in der aktuellen Implementierung die Tendenz, in unnatürlich starker Weise Feindschaften zu diagnostizieren. Auch dies sollte sich durch Umsetzung der verbliebenen Punkte verbessern, da sich die umgesetzten Punkte eher dazu eigenen, feindliches Verhalten zu erkennen (z.B. das Erkennen von Angriffen oder das Verletzen von Vereinbarungen), als friedliches.

Ebenfalls ist davon auszugehen, dass die gewählten Parameter (wie stark wird ein An-

griff, eine zugesicherte Unterstützung, usw. gewertet) einen großen Einfluss auf die Ergebnisse haben. Aus Zeitgründen wurden hier Parametertests, wie sie z.B. für das Verhandlungsmodul durchgeführt wurden, unterlassen und die Werte konnten so nur nach Gefühl gewählt werden.

5.3 FrySpi

Der FrySpi-Bot wurde von den Projektgruppenmitgliedern Daniel Spierling und Thomas Harweg entwickelt. Der Name ist eine Wortschöpfung aus den Spitznamen der Entwickler im Instant-Messaging-Protokoll Jabber, das während der PG intensiv zur einfachen und schnellen Kommunikation genutzt wurde. Da bei der Version mit menschlichen Emotionen spielerische Einbußen befürchtet wurden, hat dieser bei der statistischen Auswertung den Namen CrySpi in Anlehnung an das englische Wort für Weinen erhalten.

5.3.1 Ziele des NPC

Der FrySpi-Bot wurde als ein von Grund auf neu und selbstständig entwickelter Diplomacy-Bot konzipiert. Es wurde sowohl auf die Benutzung von Code bestehender Bots, als auch auf vorhandene Kommunikations-Frameworks verzichtet. Das bestehende DAIDE-Kommunikationsprotokoll sollte zwar verwendet werden, aber auch hier durch eine eigene Implementation. Gründe für diese Entscheidung gab es verschiedene. So entfiel zunächst einmal die Einarbeitung in fremden Programmcode, man konnte also direkt mit der Implementierung beginnen, was sich positiv auf die Motivation auswirkte. Weiterhin bedeutete diese Herangehensweise die größtmögliche Kontrolle, da die Codestruktur des Bots nach eigenen Wünschen gestaltet werden konnte. Der Bot konnte also gezielt für die Einbindung der geplanten KI-Ansätze konzipiert werden, bzw. eine größtmögliche Flexibilität für mehrere potentielle Ansätze bieten. Nicht zuletzt wurde auch der Lerneffekt, den dieser Ansatz mit sich bringt, als am größten angesehen. So bedeutete es doch, ein Softwareprojekt von nicht ganz unerheblichem Umfang selber zu planen und umzusetzen. Als Programmiersprache sollte, wie bereits zuvor durch eine Abstimmung innerhalb der PG beschlossen wurde, Java zum Einsatz kommen. Auf spieltechnischer Seite waren ein gutes (und nachvollziehbares) taktisches Verhalten, gepaart mit einem möglichst menschenähnlichen Verhalten, die erklärten Ziele des FrySpi-Bots. Dieses menschenähnliche Verhalten beinhaltet einerseits eine Kommunikation mit den anderen Mitspielern auf Press-Level 20 des DAIDE-Kommunikationsprotokolls (siehe Abschnitt 4.4.1 auf Seite 21), also mit anderen Spielern Frieden schließen, demilitarisierte Zonen vereinbaren, Zugvorschläge unterbreiten oder auch Allianzen gegen Dritte bilden. Andererseits sollten relevante menschliche Emotionen wie Wut oder Vertrauen modelliert werden, die sich wiederum auf die Handlungen des FrySpi-Bots auswirken würden.

5.3.2 Ursprüngliche Planung

Der Aufbau des FrySpi-Bot sollte von vornherein modular gestaltet sein, inbesondere bezüglich der KI. Hier sollten sich verschiedene Module um einzelne Aufgaben kümmern, deren Ergebnisse schließlich von einer übergeordneten Instanz koordiniert werden sollten. Die Planung umfasste zunächst die folgenden acht Module: *Movement*, *Attacking*, *Defending*, *SupportDetector*, *BlockSupport*, *DynamicLocationWeight*, *Negotiation* und *Emotion*.

5.3.2.1 Movement - Truppenbewegung

Das Movement-Modul ist zuständig für die Bewegung der Truppen auf taktischer Ebene. Zugvorschläge werden unter Berücksichtigung der (Himmels-)Richtung, in die sich die Truppen bewegen sollen, berechnet und bewertet. Dies geschieht in Hinblick auf ein bestimmtes Land, das erobert werden soll, oder eine Front, die errichtet werden soll. Auch Faktoren wie Frieden oder (geplante) Allianzen mit anderen Mächten würden hier eine Rolle spielen.

5.3.2.2 Attacking - Angriff

Ähnlich wie das Movement-Modul macht das Attacking-Modul Zugvorschläge für die Einheiten, jedoch auf lokale, kurze Sicht. Für eine gegebene Einheit werden die möglichen direkten Angriffsziele ermittelt. Der Gesamtnutzen eines solchen Zuges setzt sich aus verschiedenen Punkten zusammen. Ein wichtiger Faktor ist, ob das Ziel ein Versorgungszentrum ist. Ist dies der Fall, erhöht sich der Nutzen – besonders in der Herbst-Phase des Spiels – deutlich. Jedoch muss auch die Erfolgsaussicht des Zuges mit einfließen. Diese sinkt, wenn das Angriffsziel von einer feindlichen Macht besetzt ist. Aber auch Einheiten aus anderen angrenzenden Ländern müssen betrachtet werden. So könnten sie eine verteidigende Einheit unterstützen oder, falls das betrachtete Land unbesetzt ist, dieses selber einnehmen wollen und wiederum dabei von einer anderen Einheit unterstützt werden.

5.3.2.3 Defending - Verteidigung

Das Defending-Modul ist das Gegenstück zum Attacking-Modul. Hier wird überprüft, ob eigene Provinzen potentiell bedroht werden. Analog zum Angriffsszenario haben auch hier Versorgungszentren erhöhte Priorität. Auf Basis dieser Bewertungen gibt dieses Modul Empfehlungen aus, wie sinnvoll es für eine Einheit ist, ihre aktuelle Position zu verteidigen (*hold*).

5.3.2.4 SupportDetector - Unterstützung

Da Züge bei Diplomacy nur Aussicht auf Erfolg haben, wenn man in diesen eine Überzahl gegenüber dem Gegner aufbaut, ist für einen Angriff oder die Verteidung einer Provinz eine Unterstützung (*Support*) häufig unabdingbar. Das SupportDetector-Modul liefert für eine gegebene Provinz die Einheiten, die bei der Übernahme oder der Verteidigung einer Provinz Hilfe leisten können. Der Nutzen eines solchen Supports ist abhängig davon, wie entbehrlich die unterstützende Einheit bezüglich ihrer aktuellen Position zu diesem Zeitpunkt des Spiels ist.

5.3.2.5 BlockSupport - Zusammenhalt

Um anderen Einheiten Unterstüzung zur Verfügung stellen zu können, müssen sich diese in umittelbarer Nähe auf dem Spielfeld befinden. Zugvorschläge des *BlockSupport-Moduls* sollen also verhindern, dass Einheiten sich zu weit voneinander entfernen. Es muss also sicherstellen, dass Blöcke aus eigenen Truppen bestehen bleiben, und sich nicht etwa eine Einheit aus dem Truppenverband löst, und den Gegnern in den folgenden Runden schutzlos ausgeliefert ist. Das Modul liefert je Einheit einen Zugvorschlag, inklusive eines Nutzwertes.

5.3.2.6 DynamicLocationWeight - Bewertung der Provinzen

Um leichter Aussagen über die strategische Bedeutung der Provinzen machen zu können, soll dieses Modul eine der Spielsituation angepasste Bewertung selbiger liefern. Einerseits werden statische Faktoren wie die Frage, ob es sich bei der betrachteten Provinz um ein Versorgungszentrum handelt, einbezogen. Andererseits werden veränderliche, spielbezogene Faktoren wie die Truppenverteilung auf einem und um ein Land herum, sowie das eigene Verhältnis der zugehörigen Macht (Frieden, Allianz), betrachtet. Für die Verwaltung solcher Werte sind für Mächte, Provinzen, Adjazenzen und Küsten Containerklassen mit entsprechenden spielbezogenen Informationen vorgesehen (*PowerInfo*, *LocationInfo*, *AdjacencyInfo* und *CoastInfo*). Ursprünglich war sowohl eine dynamische, als auch eine statische Bewertung der Provinzen angedacht, wobei letztere offline durch einen EA berechnet werden sollte. Da aber bereits die Stragotiator-Gruppe einen ähnlichen Ansatz verfolgte, wurde dieser Plan verworfen. Somit beschränkte sich die FrySpi-Gruppe auf die dynamische Bewertung während des Spiels.

5.3.2.7 Negotiation - Verhandlung

Da Kommunikation und Verhandlungen mit den anderen Parteien ein wesentlicher Bestandteil von Diplomacy ist, darf ein dediziertes Modul hierfür nicht fehlen. Diesem Umstand trägt das Negotiation-Modul Rechnung. Es ist zuständig für das Aushandeln von Frieden, dem Bilden von Allianzen und dem Vereinbaren von demilitarisierten Zonen. Außerdem soll es alliierten Mächten Zugvorschläge zum (kooperativen) Verhalten ihrer Truppen unterbreiten.

5.3.2.8 Emotion - Modellierung der Emotionen

Eines der erklärten Ziele der PG war es, Computergegenspieler mit möglichst menschenähnlichem Verhalten zu entwickeln. Die hierfür notwendige Modellierung der Emotionen wird vom Emotion-Modul übernommen. Aufgabe des Emotion-Moduls ist es, auf Ereignisse während des Spiels zu reagieren und die Emotionen entsprechend anzupassen. Als besonders relevant betrachtet wurden hierbei Hass, Vertrauen bzw. Einschätzung der Loyalität der Gegner, sowie ein gutes oder schlechtes Gewissen den Mitspielern gegenüber. Das Modul sollte somit auch verfolgen, wie ausgeglichen das Verhältnis zwischen geleisteter und empfangener Unterstützung ist.

5.3.3 Entwicklungsframeworks

Da der FrySpi-Bot von Grund auf neu entwickelt werden sollte, mussten, bevor mit der Entwicklung der künstlichen Intelligenz begonnen werden konnte, zunächst einige Basis-Frameworks erstellt werden. Auf unterste Ebene war eine Kommunikation per TCP/IP mit dem Diplomacy-Server notwendig. Darauf aufbauend sollte sich das Game-Framwork um die Verwaltung des Spielgeschehens kümmern, also um das Spielfeld, die teilnehmenden Mächte und deren Einheiten usw. Schließlich sollte ein flexibles KI-Framework das Einbinden auch mehrerer verschiedener KI-Ansätze ohne großen Aufwand ermöglichen.

5.3.3.1 Kommunikation

Von vornherein stand fest, dass der FrySpi-Bot mit den vorhandenen Servern (DAIDE AI-Server bzw. Parlance, Abschnitt 4.4.2 auf Seite 22) über das zugehörige Protokoll[3] kommunizieren soll. Obwohl für Java mit der Yule-API bereits ein Framework für diesen Zweck vorliegt, wurde auf dessen Einbindung verzichtet und ein eigenes, den Anforderungen und Designprinzipien des FrySpi-Bots ensprechendes Framework, entwickelt. Ziel war es, eine übersichtlich strukturierte Implementation des Protokolls zu erstellen. Hierfür wurde ein konsequent objektorientierter Ansatz verfolgt. Somit wurde einerseits sichergestellt, dass das Framework leicht zu erweitern ist, aber dass andererseits nicht mehr implementiert werden musste, als für die geplanten Anforderungen des Bots nötig war. Da der FrySpi-Bot, wie auch der Nice-Bot und der Stragotiator, für Kommunikation auf Press Level 20 angelegt war, umfasst die Kommunikations-API genau diese Funktionen. Jedoch ist es problemlos möglich, höhere Press-Level durch Hinzufügen weniger Klassen, die von den existierenden Klassen abgeleitet werden, zu unterstützen.

Klassen, die eine Nachricht repräsentieren, implementieren das Interface *IMessage*. Die Nachrichten sind in drei Kategorien unterteilt: Client-Nachrichten, Server-Nachrichten und bidirektionale Nachrichten. Client-Nachrichten sind Nachrichten, die der Client (also der Bot) an den Server schickt. Dies umfasst Anweisungen (*Orders*) wie *MoveTo*, *Hold*, *Build* und weitere, Verhandlungen (Press, z. B. *Peace*, *XDO*, ...) und Antworten (*Replies*) auf Verhandlungen, also etwa das Akzeptieren oder Verweigern einer Anfrage. Nachrichten dieser Art implementieren das Interface *IClientMessage*, welches wiederum von *IMessage* erbt. Für die umgekehrte Richtung, also für Nachrichten, die der Bot vom Server empfängt, steht analog das Interface *IServerMessage* zur Verfügung. Nachrichten dieser Kategorie betreffen hauptsächlich die Spielverwaltung, also etwa die Mitteilung, dass ein neues Spiel gestartet wird, oder die Angabe der Situation auf dem Spielfeld. Schließlich gibt es noch Nachrichten, die in beide Richtungen, also bidirektional, verschickt werden. Auch diese Kategorie beinhaltet weitgehend verwaltungsspezifische Nachrichten. Der Versand und Empfang der Nachrichten wird nun von den Klassen *MessageInputStream* und *MessageOutputStream* übernommen. *MessageInputStream* nimmt eingehende Nachrichten entgegen und analysiert diese (Parsing). Ist dies erfolgreich, wird ein Objekt der entsprechenden Klasse zurückgegeben, andernfalls wird eine *HuhException* geworfen. Diese Exception veranlasst das Senden einer *Huh*-Nachricht an den Server, die signalisiert, dass eine eingehende Nachricht nicht verstanden wurde, so wie es im DAIDE-Protokoll (s. o.) vorgesehen ist . Für den Versand hingegen ist die Klasse *MessageOutputStream* zuständig. Dies geschieht indirekt, durch einen Aufruf der Methode *serializeMessage()*, die von jeder Nachrichten-Klasse implementiert wird.

Um eingehende Nachrichten nun programmintern an die einzelnen Module zu verteilen, entschied sich die FrySpi-Gruppe für ein Listener/Observer-Modell, welches bereits im Projekt des ersten Semesters zum Einsatz gekommen war. Hierfür können im *CommunicationManager* neue Listener registriert (und ggf. auch wieder entfernt) werden. Eine Klasse, die registriert werden soll, muss lediglich das Interface *IMessageListener* implementieren. Dieses enhält Methoden, die bei einem bestimmten Ereignis aufgerufen werden, z. B. wenn eine neue Runde beginnt oder wenn eine Einheit auf dem Spielfeld zieht. Das Interface kann leicht erweitert werden, beispielsweise für die Implementierung höherer Press-Level, falls erwünscht.

[3] `http://www.daide.org.uk/external/comproto.html`

5.3.3.2 Game

Neben der Kommunikation mit dem Server musste natürlich auch der Verwaltung des Spiels an sich Sorge getragen werden. Es mussten also die teilnehmenden Mächte, das Spielfeld inklusive Provinzen, Adjazenzen und Einheiten, sowie die aktuelle Spielphase (Frühling, Sommer, Herbst, Winter) verwaltet werden. Diese Aufgaben werden von dem Game-Framework übernommen. Es umfasst die hierfür relvanten Klassen *Power*, *Location*, *Province*, *Coast*, *Adjacency*, *Phase*, sowie das Interface *IUnit*, das von den Klassen *Army* und *Fleet* implementiert wird. Anzumerken ist hier, dass Programmintern zwischen *Location* und *Province* unterschieden wird. Dies hat den Grund, dass eine Region (im Text allgemein „Provinz“ genannt) mehrere Häfen haben kann. In einer solchen Situation repräsentieren die *Provinces* die beiden Häfen. Legt eine Flotte nun an einem Hafen an, muss gewährleistet sein, dass sie in einem zukünftigen Zug in das Meer zieht, das zu eben diesem Hafen gehört. Der Zug in das Meer des anderen Hafens käme einem Durchqueren des Landes gleich, was für eine Flotte einen unerlaubten Zug darstellt.

5.3.3.3 KI

Da der Aufbau des FrySpi-Bots so flexibel wie möglich gestaltet werden sollte, und auch der (sogar parallele) Einsatz mehrerer KI-Ansätze ermöglicht werden sollte, wurde unter Anwendung generischer Programmierung ein recht komplexes Konstrukt erschaffen, das diesen Anforderungen genügte. Aufgrund des Umfangs soll an dieser Stelle nur grob darauf eingegangen werden. Die von einer KI benötigten Spielverwaltungsklassen *Power*, *Location*, *Adjacency* und *Coast* müssen von der jeweiligen abstrakten Variante der Klasse geerbt werden. Die so erstellten Klassen können nun nach den Anforderungen der KI angepasst werden und müssen bei der Erzeugung von Objekten bestimmter anderer Klassen angegeben werden. Zur Verdeutlichung ein kurzes Beispiel: Seien die für die KI erstellten (abgeleiteten) Klassen *AIPower*, *AILocation*, *AIAdjacency* und *AICoast*. Möchte man nun etwa einen Angriffsbefehl erstellen, müssen einem Objekt der hierfür zuständigen Klasse *MoveToOrder* die KI-Klassen übergeben werden: *MoveToOrder<AIPower, AILocation, AIAdjacency, AICoast>*. Tatsächlich bietet dieser Ansatz eine Flexibilität, die rückblickend im Rahmen des PG-Projektes nicht nötig gewesen wäre. Für eventuelle zukünftige Projekte, die auf dem Code aufbauen, könnte diese Flexibilität aber sehr nützlich sein.

5.3.4 Entwicklungsstufen KI

Die Entwicklung des Bots unterteilte sich in zwei Phasen. In der ersten Phase wurde eine individuelle Spielstärke ohne Kooperation mit anderen Mächten angestrebt. Diese Fähigkeiten des Bots dienten als Basis für einen um ein emotionales Modell erweiterten kooperierenden Bot um die Spielweise an einen menschlichen Spieler anzuhähern.

5.3.4.1 Spielstärke

Zu Spielbeginn liegt aufgrund der geringen Anzahl der Einheiten und der vergleichsweise hohen Anzahl unbesetzter Versorgungszentren im Vergleich zum späteren Spielverlauf eine stark abweichende Spielsituation vor. Es bieten sich nun verschiedene Ansätze zur Bewältigung dieser Herausforderung.

- statische Eröffnungsbibliothek
- unterschiedliche Zugstrategien für verschiedene Spielphasen

- dynamischer Ansatz

Die Möglichkeit eine statische Eröffnungsbibliothek für die verschiedenen Nationen zu verwirklichen wiederstrebte unseren Zielen einen über den gesamten Spielverlauf intelligenten Bot zu erschaffen. Die statische Unterteilung des Spiels in verschiedene Spielphasen hätte einen regelbasierten Charakter gehabt, so dass wir uns für einen dynamischen Ansatz entschieden haben, der implizit anhand der vorhandenen Situation seine Verhaltensweise anpaßt. Natürlich sind bei dieser Vorgehensweise die meisten Komplikationen zu befürchten, jedoch wollten wir in diesem Projekt Erfahrungen für die Zukunft sammeln und Herausforderungen lösen.

Spielsituation mit wenigen Einheiten
Zumal das Positionsspiel zu Beginn primär durch unbesetze Versorgungszentren bestimmt ist, haben wir uns entschlossen den Truppen einen Drang zu unbesiedelten Regionen zu geben. Nachdem wir verschiedene klassische Methoden der Computional Intelligence nach ihrer Anwendungstauglichkeit untersucht haben und keine unsere Anfordergen erfüllte, fiel unser Augenmerk auf Influence Maps, die insbesondere bei der Entwicklung von KIs für Spiele ihre Anwendung finden. Im Gegensatz zur klassischen Aufteilung der Spielkarte in Gitternetzlinien, liegen die Länder in Diplomacy nicht so gleichmäßig verteilt, wodurch wir gezwungen waren, unseren Fluss über die Kanten fließen zu lassen ohne über geographische Informationen der Orte zu verfügen. Um den Rückfluß von Werten zu verhindern, wurden die Flüsse über ein Verfahren ähnlich der Breitensuche verbreitet. Allerdings grenzen in Diplomacy nicht alle Länder an die gleiche Anzahl von Ländern, wodurch auf Ländern mit mehr Landesgrenzen häufig höhere Werte sind, so dass zur Korrektur eine Division der Werte um die angrenzen Länder notwendig war. Zumal insbesondere zu Beginn eines Spiels auch über mehrere Züge hinweg geplant werden kann, haben wir uns zu einer exponentiellen Fortpflanzungsstrategie mit einem Faktor von 0,5 entschieden, so dass immer die Hälfte des Flußes über eine maximale Distanz von drei Kanten an die angrezenden Länder fortgepflanzt wird. Dieser Faktor erscheint im Vergleich zu anderen Anwendungen relativ klein, doch verfügt das Spielfeld in Diplomacy über relativ wenige Zellen.

Bewegungsdrang mit vielen Einheiten
Nachdem die Aktionen des *MovementModule* bisher allein durch den Fluß zu unbesiedelten Regionen bestimmt war um somit Chancen wahrzunehmen, müssen im nachfolgenden Spielverlauf auch Gefahren erkannt und gebannt werden. Hierzu wird ein analoger Ansatz verwendet, wo allerdings der Fluss von gegnerischen Einheiten ausgeht. Jedes Gebiet ist nun mit einem Wert pro Spieler belegt, der die Bedrohung für dieses Gebiet beschreibt. Dieser Wert alleine bringt natürlich noch keinen Nutzen für das *MovementModule*, da die Gebiete für den Bot nicht alle den gleichen Nutzen haben. Ein besonderes Interesse gilt trivialerweise der Verteidigung eigener Versorgungszentren. Hierzu wird ebenfalls ein Fluss eingeführt, der von eigenen Versorgungszentren ausgeht. Ein wichtiges Kriterium für die Wahl einer linearen Fortpflanzungstrategie waren die Beobachtungen, dass einer einzelnen Einheit bei exponentialer Fortpflanzung aufgrund höherer Abschwächungen zu wenig Beachtung geschenkt wurde. Diese Fortpflanzungsstrategie ist der Schlüssel zur Gefahrenerkennung, die aus bisher als sicher eingestuften Regionen z.B. den Flanken anrückt. An der Stelle, wo sich die Gefahren- und Verteidigungsflüsse treffen, herrscht besonderes Konfliktpotential. Dieses Gebiet gehört nun also zu einem gewissen Grad zur Fuzzy-Menge der b̈edrohten Gebieteünd der v̈erteidigungswürdigen Gebiete:̈ Die Schnittmenge läßt sich durch das Minimum bestimmen. Dieser Konflikt-Koeffizient produziert analog zum Fluß zu unbesiedelten Regionen einen Verteidigungsfluß zu diesen Gebieten. Zumal zu Beginn des Spiels kaum Konflikt-Potential vorhanden ist, greift in der Regel zu Beginn der Drang zu den unbesetzten Versorgungszentren. Eine besonders schwierige Aufgabe war die Bestimmung geeigneter Koeefizienten, die den zu bevorzugenden Drang beeinflussen. Aufgrund einer Spiellänge (ohne Verzögerungen) von mehr als einer Minute und des Rauschen in den

relativ wenigen Spielergebnissen, waren Anpassungen diverser Koeffizienten eher spekulativ. Die gefahrenorientierte Strategie fühte dazu, dass die Truppen eine Art Mauer um die eigenen Versorgungszentren bildeten, so dass das *MovementModule* die Aufgabe des eigentlich geplanten *BlockSupportModule* übernahm.

Angriffsverhalten
Aus dieser defensiven Grundhaltung heraus, wurde es nun angestrebt Chancen zur Vergrößerung des Imperiums zu erkennen und diese zu nutzen. Hierzu liefert das *PowerDistributionModule* Kräfteverhältnisse auf den einzelnen Gebieten. Dieses Kräfteverhältnis basiert auf den Möglichkeiten einen Angriff zu unterstützen gegenüber der Verteidigung eines Angriffs. Das *PowerDistributionModule* ersetzt in Kombination mit Koeffizienten, der Versorgungszentren besonders im Herbst stärker gewichtet, das ursprünglich geplante *DynamicLocationWeightModule*, das bestimmten Provinzen bewerten sollte. Bei einem Kräfteverhältnis zu Gunsten des FrySpi werden alle Kräfte der angrenzenden Einheiten gebündelt um eine hohe Erfolgschance zu erhalten. Dieses Verhalten wurde dadurch erreicht, dass ein besonders gutes Kräfteverhältnis einen hohen Empfehlungswert für einen Zug hervorruft. Sobald ein Zug dann fixiert wurde, liefert das *SupportDetectorModule* für alle Unterstützungszüge einen besonders hohen Wert, so dass diese gewählt werden und keine Angriffskolisionen entstehen können. Die Züge werden also alle nacheinander selektiert und nicht als Zugpaket, wie beispielsweise beim *Stragotiator*. Diese Vorgehensweise vereinfacht die Bewertung erheblich, allerdings werden durch diese statische Auswahl mögliche Abhängigkeiten der Züge untereinander nicht berücksichtigt.

5.3.4.2 Analyzer

Um die von der KI durchgeführten Züge bewerten und optimieren zu können, war eine Analyse der selbigen notwendig. Der existierende DAIDE-Mapper (Kapitel 4.4.3, Seite 22) erschien der FrySpi-Gruppe für diesen Zweck aus meheren Gründen ungeeignet. Zwar zeigt dieser alle durchgeführten Züge an, inklusive gescheiterter Züge und Supports, für eine fundierte Analyse wären aber weitere, programminterne Informationen nötig, die der DAIDE-Mapper natürlich nicht liefern kann. Eine einfache Alternative wäre eine konsolenbasierte Ausgabe der Züge inklusive der relevanten Werte gewesen. Doch während diese Methode aus Sicht des Implementationsaufwands zwar günstig ist, wäre eine detailierte Zuganalyse hiermit doch ungleich aufwändiger. Somit fiel die Entscheidung, den *FrySpi-Analyzer*, eine eigene, programminterne grafische Oberfläche (Graphical User Interface, GUI) zur Analyse zu implementieren (siehe Abbildung 26).

Der Analyzer zeigt das Diplomacy-Spielfeld inklusive der Provinzen (Kreise für Versorgungszentren, Quadrate für die restlichen Provinzen). Die Farbgebung der Provinzen entspricht den Großmächten:

- **Rot:** England
- **Blau:** Frankreich
- **Türkis:** Deutschland
- **Grün:** Italien
- **Orange:** Österreich-Ungarn
- **Rosa:** Russland
- **Gelb:** Türkei

Ist eine Provinz noch nicht von einer Macht besetzt, so ist sie im Analyzer weiß dargestellt. Befindet sich eine Einheit in einer Provinz, wird dies durch ein kleines Quadrat innerhalb der eigentlichen Markierung signalisiert: Ein schwarzes Quadrat steht hierbei

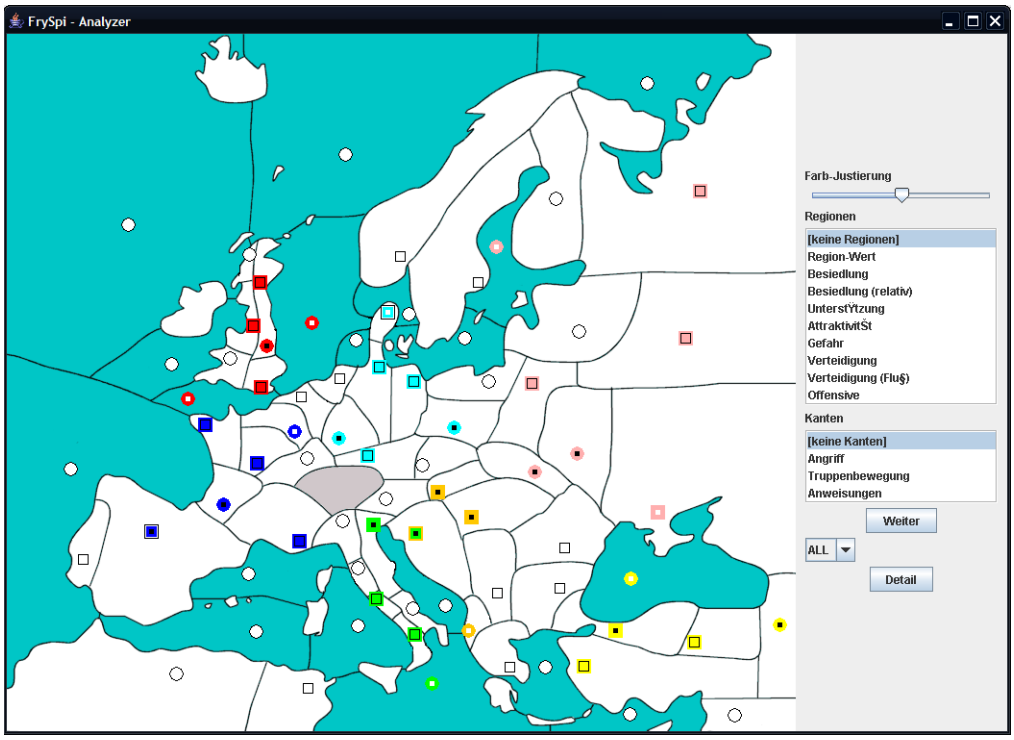


Abbildung 26: Das Analyzer-Fenster kurz nach dem Start des Spiels

für eine Truppe (*army*), während ein weißes für eine Flotte (*fleet*) steht. Die eigentliche Funktionalität der GUI kann nun durch die Bedienelemente an der rechten Seite des Fensters hinzugeschaltet werden. Die Auswahlliste „Regionen“ erlaubt es, die internen Bewertungen der Locations durch den Bot graphisch darzustellen. Die intern als Zahlenwerte vorliegenden Bewertungen werden durch Einfärbung der Regionen in einem Grauton dargestellt, wobei ein dunklerer Grauton einem höheren Zahlenwert entspricht. Der modulare Aufbau ermöglichte es, während der Entwicklung unabhängig von anderen Anpassungen einzelne Module zu analysieren. Nach Abschluss des Projektes stehen hier die folgenden Bewertungsmerkmale zur Auswahl:

- **Region-Wert:** (statischer Provinz-Wert, wurde gestrichen)
- **Besiedlung:** Bewertung bisher unbesetzter Provinzen
- **Besiedlung (relativ):** der Besiedlungs-Wert relativ zur Anzahl der angrenzenden Provinzen
- **Unterstützung:** die Wahrscheinlichkeit, für eine Kampfhandlung einer Provinz Unterstützung zu bekommen
- **Attraktivität:** (Attraktivitätswert einer Provinz, wurde gestrichen)
- **Gefahr:** die potentielle Bedrohung einer Provinz durch Gegner
- **Verteidigung:** der geschätzte Aufwand, um eine Provinz zu verteidigen
- **Verteidigung (Fluss):** wie *Verteidigung*, jedoch berechnet mittels eines simulierten Flusses über die Provinzen
- **Offensive:** der geschätzte Aufwand, um eine Provinz einzunehmen

Die zweite Auswahlliste, „Kanten“, ermöglicht es, Informationen über die möglichen Züge der eigenen Einheiten darzustellen. Die Optionen hier sind:

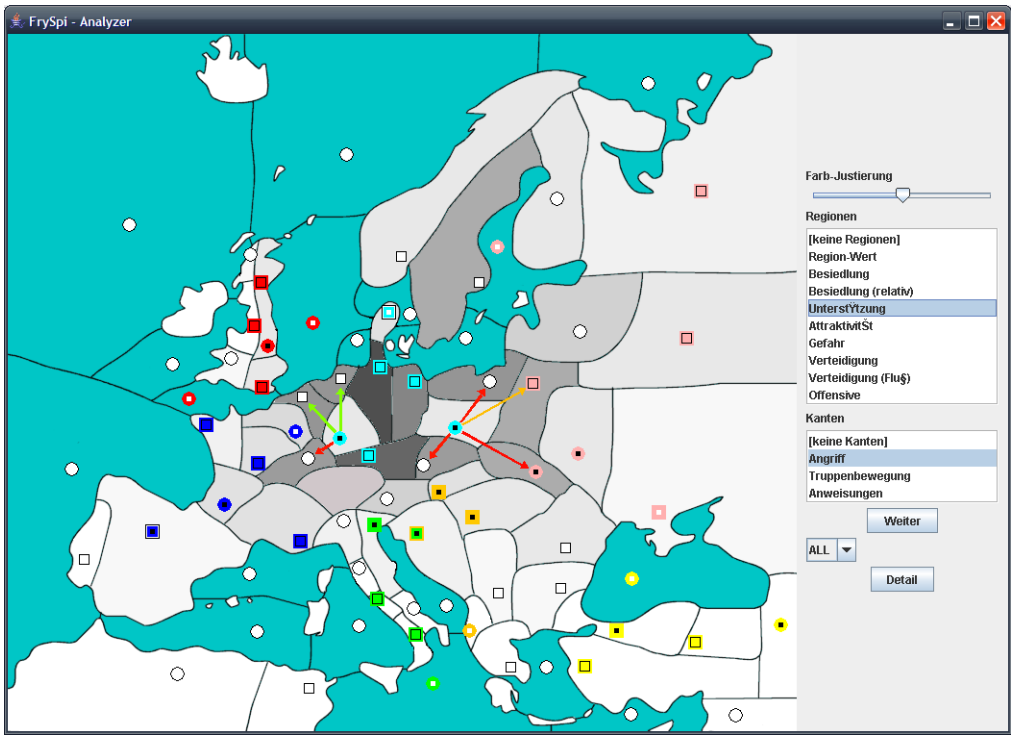


Abbildung 27: Das Analyzer-Fenster mit möglichen Angriffszügen und Unterstützungswerten der Regionen

- Angriff
- Truppenbewegungen
- Anweisungen

Die Darstellung der Züge geschieht in Form von Pfeilen, die, ausgehend von dem Aufenthaltsort der jeweiligen Einheit, auf ein potentielles Ziel zeigen (Abbildung 27). Wichtig ist hierbei die Farbe der einzelnen Pfeile, da diese die berechnete Güte des Zuges widerspiegelt. Mit steigender Güte steigt der Grünanteil und sinkt der Rotanteil und anders herum. Wird also ein Zug von der KI als gut angesehen, ist der entsprechende Pfeil in einem Grünton dargestellt, wird er als schlecht angesehen, in einem Rotton. Mittelmäßige Züge erhalten einen gelben Pfeil. Da geringe Farbunterschiede für das menschliche Auge sehr schwer auszumachen sein können, befindet sich am oberen rechten Rand des Analyzer-Fensters der Schieberegler „Farb-Justierung“. Dieser ist als eine Art Konstrastregler zu verstehen, mit dem sich geringe Farbunterschiede deutlicher darstellen lassen. Da die Rundenzeiten in einem Diplomacy-Spiel unter der ausschließlichen Beteiligung von Bots sehr kurz sein können, wird die Runde im Analyzer-Modus angehalten, d.h. die Zuganweisungen des eigenen Bots werden zurückgehalten, und erst beendet, wenn der Button „Weiter“ gedrückt wird.

5.3.4.3 Menschlichkeit

Verhandlungen

Da Kommunikation im Allgemeinen und Verhandlungen im Speziellen wichtige Elemente in Diplomacy einerseits und ein erklärtes Ziel der PG andererseits waren, war der nächste Schritt in der Entwicklung des FrySpi-Bots, dies zu implementieren.

Für die Verarbeitung fremder bzw. den Versand eigener Friedensangebote, sowie für die Verarbeitung von Demilitarisierungsanfragen, ist das *PeaceModule* zuständig. Eingehende Friedensnachrichten werden von dem Modul ungeprüft kategorisch akzeptiert, jedoch im weiteren Verlauf nicht unbedingt eingehalten, der FrySpi-Bot handelt hier egoistisch. Für das Versenden von Friedensangeboten ist die ist die Methode *SmokeCalumet()* („Friedenspfeife rauchen“) zuständig. Für jede eigene Provinz wird das Kräfteverhältnis zwischen den eigenen Truppen, die diese direkt erreichen können und denen jedes Mitspielers, errechnet. Hierzu werden die zuvor durch das *ThreatModule* berechneten Werte zur Hilfe gezogen. Ist das Kräfteverhältnis gleich oder größer 1.0, so wird der zugehörigen Macht ein Friedensangebot unterbreitet. Ob einer Bitte einer anderen Partei, eine Region zur demilitarisierten zu erklären, Folge geleistet wird, ist abhängig von einem Zufallswert und dem Hass gegenüber dem Mitspieler. Dieser Wert wird vom Emotionsmodul berechnet, das im nächsten Abschnitt genauer beschrieben wird. Eine Verweigerung einer Demilitarisierungsanfrage vermittelt dem Anfragenden immer eine Warnung. Aus diesem Grund wird für jegliche Angriffsziele diese Anfrage angenommen. Falls das Gebiet im kommenden Zug nicht eingenommen werden soll, wird die Anfrage nur bei dem Spieler, gegen den der FrySpi den meisten Hass hat, mit Wahrscheinlichkeit 0.5 abgelehnt. Die Ablehnung des Angebots führt bei dem Feind zu einer stärkeren Konzentration auf dieses Gebiet und andere Regionen werden dadurch anfälliger. Die Kooperation des FrySpi mit anderen Bots basiert auf der Differenz zwischen geleisteten und erhaltenen Unterstützungen. Sofern die Einheit nicht für den Angriff mit der höchsten Priorität eingeplant und die Differenz zwischen geleisteten und erhaltenen nicht größer als eins ist, wird die Unterstützung zugesagt. Bei Anfragen des Spielers gegen den der meiste Hass besteht, wird mit einer Wahrscheinlichkeit von 0.5 zugesagt und nicht eingehalten, die restlichen werden abgelehnt. Wie sich durch die statistische Auswertung herausstellte, wurden die Zuganweisungen meistens schon abgeschickt, bevor diese Anfragen integriert wurden, so dass ein Großteil der Zusagen nicht eingehalten werden konnte.

Emotionen

Nachdem das strategische Verhalten des Bots einen guten Standard erreicht hatte, und auch die Fahigkeit zur Kommunikation mit den Mitspielern gegeben war, sollte er schließlich noch um emotionales Verhalten erweitert werden. Zuständig hierfür ist das *EmotionModule*. Während des Spiels werden Daten über jede gegnerische Partei gesammelt. Diese Daten umfassen Informationen darüber, ob eine Macht uns gegenüber Unterstützung geleistet, verweigert oder versprochen und nicht eingehalten hat. Außerdem wird Buch darüber geführt, wie oft ein Gegner unsere Stellungen angegriffen hat, aufgeteilt in Versorgungszentren und Nicht-Versorgungszentren. Umgekehrt werden genau diese Daten auch in umgekehrter Richtung gespeichert, also das Verhalten des FrySpi-Bots gegenüber einer anderen Macht. Ausgehend von diesen Daten sollten nun die Emotionen Hass und Vertrauen (Loyalität des Gegners), sowie ein Gewissen gegenüber den Mitspielern modelliert werden. Die zugrunde liegenden Formeln sind nach eigenem Ermessen gewählt und basieren nicht auf einem existierenden Modell. Da das Emotionsmodul erst kurz vor Ende der PG-internen Implementierungsphase eingebaut wurde, war es aus Zeitmangel nicht mehr möglich, aussagekräftige Tests über die Berechnungsergebnisse des Moduls durchzuführen. Aus diesem Grund ist das Emotionsmodul nicht angepasst oder optimiert worden. Der auf die aktuelle Runde bezogene Hass-Wert einem anderen Spieler gegenüber berechnet sich durch die folgende Formel.

$$Hass = attackedNSC \cdot 0.1 + attackedSC \cdot 0.2 + fakeSupport \cdot 0.2 + deniedSupport \cdot 0.1$$

AttackedNSC ist hier die Anzahl der Angriffe auf die eigenen Stellung, die keine Versorgungszentren sind (non-supply-centre), durch den Gegner, *attackedSC* bezeichnet analog den Wert für die Versorgungszentren. Die Variablen *fakeSupport* und *deniedSupport* geben an, wie oft der betreffende Mitspieler Unterstützung zugesichert, aber

nicht eingehalten hat, bzw. wie oft er eine Anfrage auf Unterstützung direkt abgelehnt hat. Für den Fall, dass eine Allianz mit dem Spieler bestand, die dieser, etwa durch einen Angriff, gebrochen hat, erhöht sich der berechnete Hass-Wert um 0.3.

Die Loyalität eines Mitspielers setzt sich hauptsächlich aus dessen Verhalten bezüglich Truppenunterstützungen zusammen:

$Loyalitaet = givenSupport \cdot 0.1 - deniedSupport \cdot 0.1 - fakedSupport \cdot 0.15$

GivenSupport bezeichnet hier die tatsächliche gewährte Unterstüzung uns gegenüber, die anderen beiden Variablen entsprechen denen aus der Berechnung des Hass-Wertes. Auch der Loyalitätswert wird durch einen Allianzbruch zusätzlich beeinflusst. Ein solcher Bruch führt zu einer Senkung um 0.2, andernfalls erhöht sich der Wert automatisch um 0.05 pro Runde.

Das Gewissen gegenüber einem anderen Spieler setzt sich anteilig aus den Differenzen zwischen den Unterstützungen für und vom Mitspieler zusammen. *SupportDiff* bezeichnet die Differenz zwischen der Anzahl der von uns für den Mitspieler geleisteten Unterstüzung und der Anzahl der von ihm an uns geleisteten Unterstüzung, *fakeSupportDiff* ist analog die Differenz der gegenseitig versprochen, aber nicht eingehaltenen Unterstützungen.

$Gewissen = 0.6 \cdot (supportDiff/10.0) + 0.4 \cdot (fakeSupportDiff/10.0)$

Um definiert mit den Werten *Hass*, *Loyalität* und *Gewissen* rechnen zu können, werden sie jeweils auf das Intervall $[0, 1]$ beschränkt. Die pro Runde berechneten Werte fließen schließlich in je einen Gesamtwert ein, der eine Art Gedächtnis darstellt. Die Verrechnung geschieht nach einer Formel, die, in leicht abgewandelter Form, bereits in den Opponent-Modeling-Klassen des ersten Projekts in der PG Verwendung fand:

$Gesamtwertneu = \frac{2}{3} \cdot Gesamtwertalt + \frac{1}{3} \cdot Rundenwert$

Für jeden der drei Werte *Hass*, *Loyalität* und *Gewissen* wird also der jeweils in der abgelaufenen Runde berechnete Wert (*Rundenwert*) mit dem bisherigen Gesamtwert (*Gesamtwertalt*) zu einem neuen Gesamtwert (*Gesamtwertneu*) verrechnet.

5.3.5 Erkenntnisse aus der Durchführung

Die Entwicklung des FrySpi-Bots stellte eine nicht unerhebliche Herausforderung an die Gruppenmitglieder dar. So erforderte die Entwicklung eines kompletten Diplomacy-Bots sowohl einen deutlich erhöhten Programmier- als auch Planungsaufwand. Bevor überhaupt mit der Entwicklung der KI begonnen werden konnte, musste ein voll funktionsfähiges Grundgerüst konzipiert und entwickelt werden. Und allein der Entwicklung eines guten strategischen Verhaltens hinsichtlich der Züge, das dem Niveau der existierenden Bots, wie dem *DumbBot* oder dem *Diplominator*, bzw. den darauf aufbauenden Bots der beiden anderen PG-Untergruppen (*Nice* und *Stragotiator*), entsprach, erwies sich als nicht triviale Aufgabe. Hinzu kam, dass die FrySpi-Gruppe lediglich aus zwei Teilnehmern bestand, was allerdings von vornherein klar war und in Kauf genommen wurde. Dies konnte jedoch durch effizientes Teamwork, Kommunikation und eine gut organisierte, stark praxisbezogene Arbeitsweise kompensiert werden. Hier zeigte sich, dass eine kleine Gruppe auch durchaus Vorteile mit sich bringt, da sich der Mehraufwand an Verwaltung auf ein Minimum reduziert. Hilfreich war zudem die Tatsache, dass die Gruppenmitglieder bereits über einige Programmiererfahrung verfügten.

5.3.6 Fazit

Ingsgesamt betrachtet kann die Entwicklung des FrySpi-Bots mit Sicherheit als erfolgreich bezeichnet werden. Das gesteckte Ziel, einen vollständigen Diplomacy-Bot, der auf Press-Level 20 kommuniziert und Emotionen modelliert, von Grund auf neu zu entwickeln, wurde erreicht. Zudem war der FrySpi-Bot in strategischer Hinsicht im Vergleich zu den Bots der anderen beiden Gruppen am Ende der Implementierungsphase dominierend. Dies zeigte sich bei den ständig parallel zur Implementierung unter den Bots ausgetragenen Wettbewerben. Verbesserungs- und Erweiterungspotential steckt sicherlich in den Verhandlungs- und Emotionsmodulen, da diese aufgrund des zeitlich gesteckten Rahmens nicht zur vollen Zufriedenheit fertiggestellt werden konnten. Die Probleme basierten wie sich nachher herausstellte primär auf Timing-Problemen innerhalb der Kommunikation mit dem Server.

6 Statistische Auswertung

Die statistische Auswertung untergliedert sich in zwei Teile. Auf der einen Seite wurde das Spielgeschehen durch Auswertung der Log-Dateien analysiert. Auf der anderen Seite wurde ein Turing-Test durchgeführt, der die Menschlichkeit der Bots verifiziert.

6.1 Computergestütze Anayse

Für die Statistikanalyse wurden pro Konstellation ca. 2000 Spiele mit unterschiedlichen Botbelegungen durchgeführt. Während der Spiele wurden Informationen wie zum Beispiel die Anzahl der Siege eines Bots, die der Angriffe usw. mitgelogt. Jedes Spiel wurde bis zum Sieg einer Macht gespielt, maximal jedoch bis zum 100. Spieljahr. Dies war notwendig, da die Performanz des genutzten Servers danach dramatisch einbricht. In diesem Fall wurde das Spiel abgebrochen und erneut gestartet. Außerdem wurde eine Contestsoftware eingesetzt, um auf Fehler im Spielablauf zu reagieren. In diesem Kapitel werden zunächst die Ergebnisse einiger ausgewählter besonders interessanter Konstellationen dargestellt und danach eine allgemeinere Gesamtauswertung durchgeführt. Abschliessend werde die einzelnen Bots bezüglich Spielstärke in Abhängigkeit von der gespielten Macht untersucht.

6.1.1 die unterschiedlichen Spielkonstellationen

Das Ziel der Gruppe war es, die entworfenen Bots möglichst gut vergleichen zu können. Aus diesem Grund ist jeder der entworfenen Bots gegen jeden anderen angetreten. Tabelle 6.1 zeigt welche der Bots in den unterschiedlichen Konstellationen gegeneinander spielten.

Konstellation	Bottyp 1	Bottyp 2
1	CrySpi	FrySpi
2	CrySpi	Nice
3	CrySpi	Strago
4	CrySpi	StragoEA
5	FrySpi	Nice
6	FrySpi	Strago
7	FrySpi	StragoEA
8	Nice	Strago
9	Nice	StragoEA
10	Strago	StragoEA

Tabelle 6.1: Statistikauswertung: die unterschiedlichen Konstellationen

Da für eine Party Diplomacy sieben Spieler benötigt wurden, wurde festgelegt, dass jeweils drei Bots des Bottyps 1, drei Bots des Bottyps 2 sowie ein Diplominator teilnahmen. Der Diplominator wurde eingesetzt, damit es kein Ungleichgewicht zwischen den beiden Bottypen gab und somit von keinem Bottypen mehr Mächte kontrolliert wurden als von den anderen. Zudem wurde eine weitere Konstellation eingeführt. In dieser Konstellation 0 war jeder Bottyp einmal vertreten und der Diplominator zweimal. Im folgenden werden die Ergebnisse der Konstellation 0 dargestellt sowie die Ergebnisse der Konstellationen 1 (CrySpi vs. FrySpi) und 10 (Strago vs. Strago mit EA). Außerdem wird der Nice-Bot in den Konstellationen 2 und 8 mit den jeweils leicht spielstärkeren Varianten der anderen Bots dem CrySpi und dem Stragotiator ohne EA verglichen.

6.1.1.1 Jeder gegen jeden

In den Spielen in der Konstellation 0 sind sämtliche von der PG entworfenen Bots einmal vertreten. Der Diplominator ist zweimal vertreten.

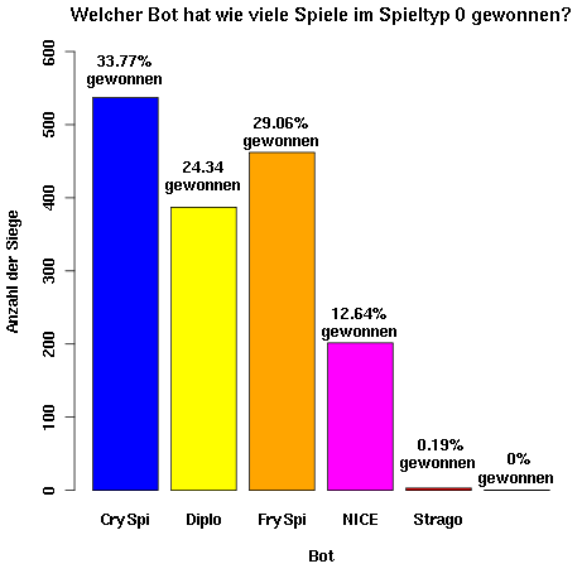


Abbildung 28: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 0

Die Auswertung der Anzahl der Siege (Abbildung 28) zeigt deutlich, dass die beiden von der FrySpi-Gruppe entworfenen Bots FrySpi und CrySpi die spielstärksten Bots sind, wobei der CrySpi den FrySpi um einige Prozentpunkte übertrifft. Der Diplominator kann 25 Prozent der Spiele gewinnen und der Nice-Bot, welcher auf dem Diplominator aufbaut 12 Prozent. Bei dem Vergleich zwischen Nice-Bot und Diplominator ist allerdings zu beachten, dass der Diplominator in dieser Konstellation mit zwei Bots vertreten ist und der Nice-Bot hingegen nur mit einem. Die beiden Bots der Stragotiator-Gruppe sind in dieser Konstellation mit 0, 19 Prozent (ohne EA) und 0 Prozent (mit EA) die schwächsten Bots.

Untersucht man die Anzahl der versandten und akzeptierten Friedensangebote (Abbildung 29 links) in der Konstellation 0 fällt auf, dass der Nice-Bot die meisten Friedensangebote versendet, die Anzahl der akzeptierten Friedensnachrichten jedoch eher gering ausfällt. CrySpi und FrySpi versenden im Vergleich deutlich weniger Friedensangebote und weisen eine höhere Quote akzeptierter Angebote auf. Die Anzahl der versandten Friedensangebote des Stragotiators ist sehr gering, die Quote akzeptierter Angebote ist jedoch vergleichweise positiv. Untersucht man die Anzahl der vorgeschlagenen Allianzen (Abbildung 29 rechts) fällt besonders auf, dass weder FrySpi, CrySpi oder der Diplominator Allianzangebote versenden. Ähnlich wie bei den Friedensangeboten versendet der Nice-Bot deutlich mehr Angebote als der Stragotiator. Auffällig ist, dass die Anzahl der akzeptierten Allianzvorschläge der Bots mit ungefähr 75 Prozent deutlich höher ist als die der akzeptierten Friedensangebote. Die absolute Anzahl der Allianzgebote fällt im Vergleich zu der absoluten Anzahl der Friedensangebote deutlich geringer aus.

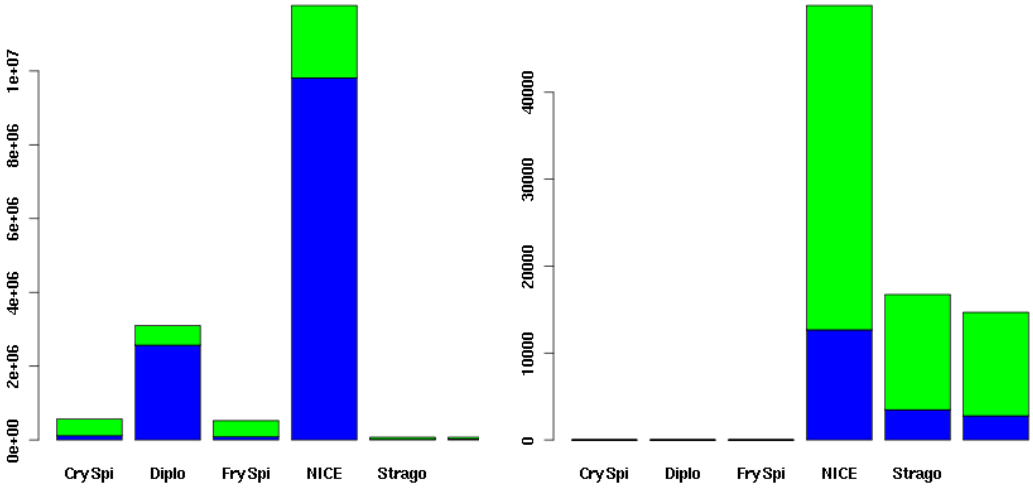


Abbildung 29: Statistikauswertung: Verhandlungen in Konstellation 0

6.1.1.2 CrySpi vs. FrySpi

In der Konstellation 1 (Abbildung 30) spielen drei FrySpi gegen drei CrySpi und einen Diplominator. Wie bereits in den Spielen der Konstellation 0 kann festgestellt werden, dass der CrySpi leicht spielstärker als der FrySpi ist. Die Anzahl der vom CrySpi gewonnenen Spiele ist um ungefähr 5 Prozentpunkte höher als die des FrySpi. Dem Diplominator gelingt es lediglich unter 2 Prozent der Spiele zu gewinnen.

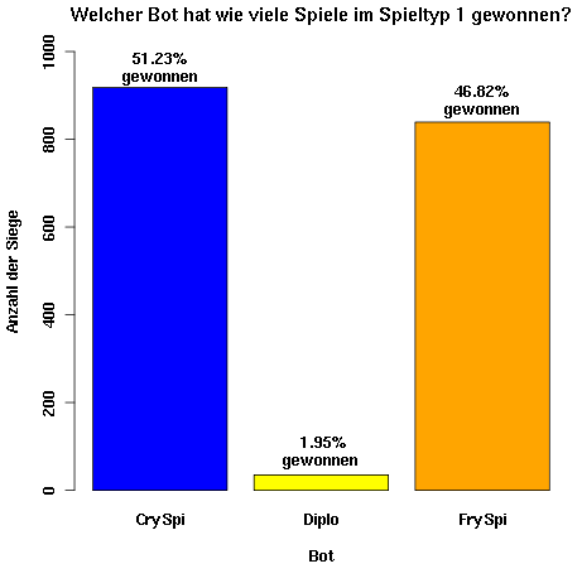


Abbildung 30: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 1

Der CrySpi versendet in dieser Konstellation (Abbildung 31 links) die meisten Friedensangebote und ungefähr 40 Prozent mehr als der FrySpi. Der Diplominator versendet am

wenigsten Friedensangebote und von diesen wird ebenfalls nur ein geringer Anteil von unter zehn Prozent akzeptiert. Die Akzeptierungsquote der Allianzvorschläge (Abbildung 31 rechts) von CrySpi und FrySpi liegen mit 40 Prozent auf vergleichbarem Niveau. Die absolute Anzahl der akzeptierten Vorschläge fällt für CrySpi auf Grund der ebenfalls erhöhten Anzahl an Nachrichten besser aus. Da alle drei Bots keine Allianzvorschläge versenden kann die Güte der Vorschläge nicht untersucht werden.

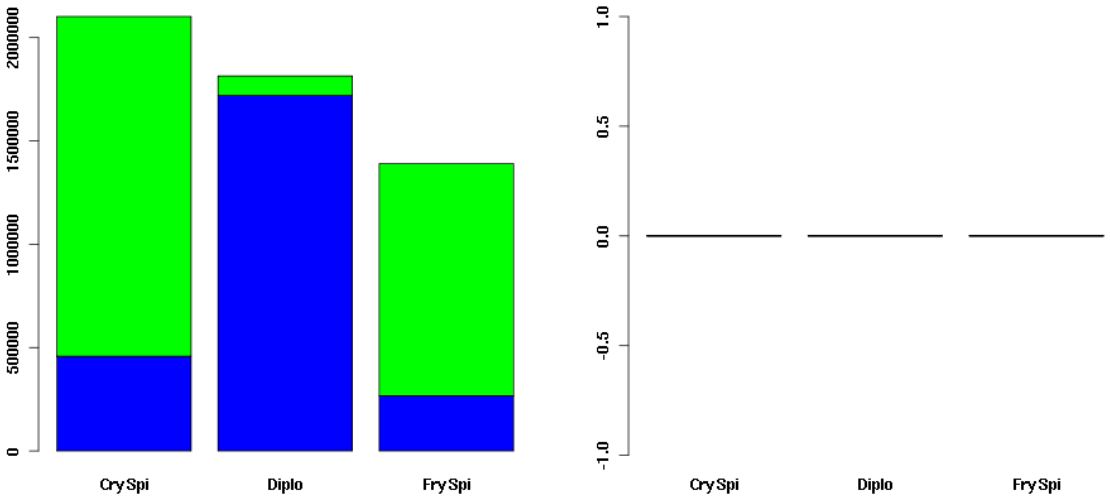


Abbildung 31: Statistikauswertung: Verhandlungen in Konstellation 1

6.1.1.3 Strago vs. Strago (mit EA)

In der Konstellation 10 (Abbildung 32) spielen drei Stragotiatoren ohne EA, drei Stragotiatoren mit EA und ein Diplominator. Der Diplominator gewinnt über 97 Prozent aller Spiele. Der Stragotiator ohne EA ist mit ungefähr 2,3 Prozent noch deutlich spielstärker als der Stragotiator mit EA.

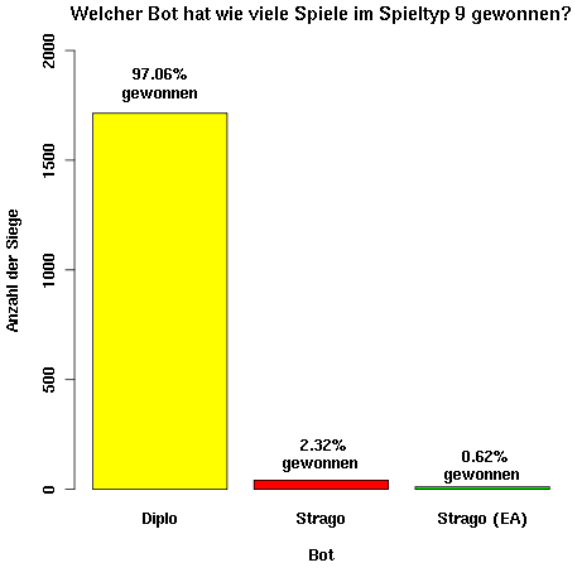


Abbildung 32: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 10

Der Diplominator, welcher die meisten Friedensangebote verschickt, hat eine Akzeptanzquote von ungefähr 60 Prozenten und die Friedensangebote der Stragotiatoren, sowohl mit als auch ohne EA, liegt bei annähernd 100 Prozent (Abbildung 33 links). Der Stragotiator ohne EA versendet deutlich mehr Allianzangebote als der Stragotiator ohne EA und die Akzeptanzquote fällt leicht höher aus (Abbildung 33 rechts). Das schwache Abschneiden des Stragotiators kann dadurch begründet sein, dass bei der Entwicklung der Fokus auf Vertrauen und Kooperation gelegt wurde. Der realisierte *Klassiker* ist zwar, wie die Statistik zeigt, ein guter Verhandlungspartner, kann jedoch gegen den, nur in sehr geringem Maße verhandelnden, aggressiven Diplominator nicht bestehen.

6.1.1.4 CrySpi vs. Nice-Bot

In der Konstellation 2 (Abbildung 34) spielen drei CrySpis, drei Nice-Bots und ein Diplominator. In dieser Konstellation gelingt es dem CrySpi, über 75 Prozent der Spiele zu gewinnen. Die Nice-Bots gewinnen ungefähr 16 Prozent der Spiele und der Diplominator acht Prozent.
Der Nicebot versendet die meisten Friedensnachrichten, gefolgt vom CrySpi und dem Diplominator (Abbildung 35 links). Auffällig ist erneut die deutlich höhere Akzeptanzquote der Anfragen des CrySpi im Vergleich zu denen des Diplominators und des Nice-Bots. Von den vom Nice-Bot versandten Allianzanfragen wurden ungefähr zwei Drittel akzeptiert (Abbildung 35 rechts). CrySpi und Diplominator versandten keine solchen Anfragen.

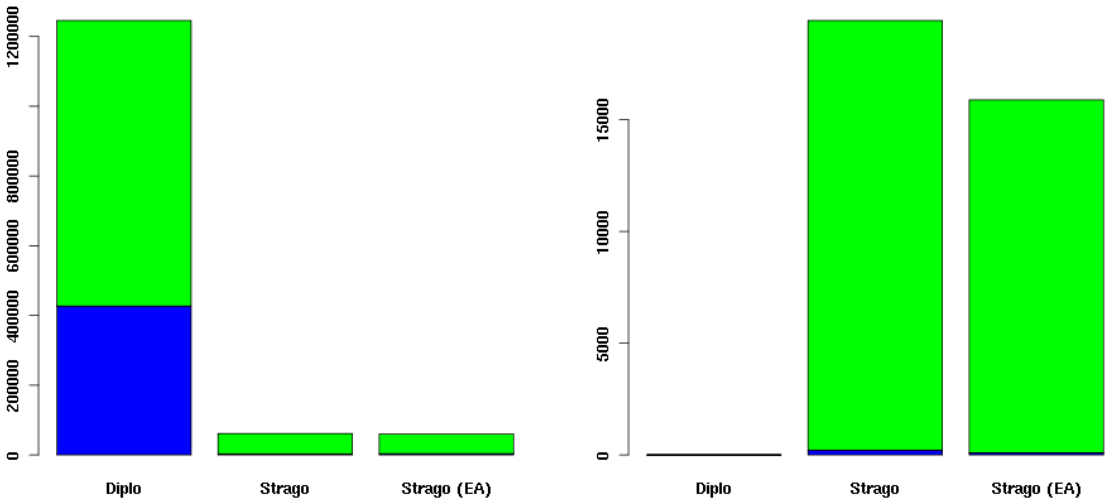


Abbildung 33: Statistikauswertung: Verhandlungen in Konstellation 10

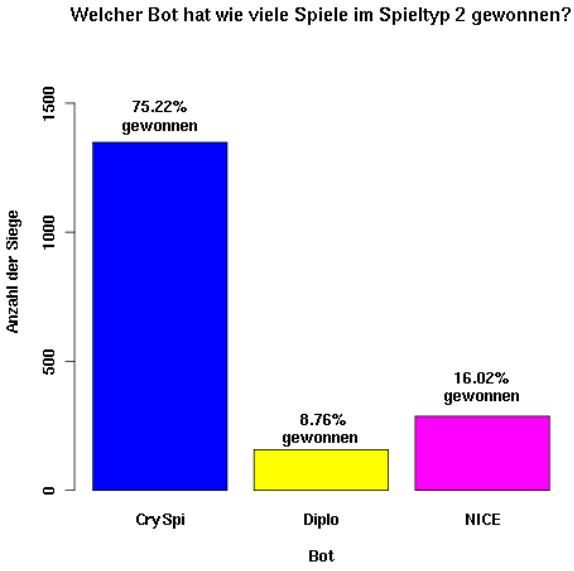


Abbildung 34: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 2

6.1.1.5 CrySpi vs. Strago

In der Konstellation 3 (Abbildung 36) spielen drei CrySpi, drei Stragotiatoren ohne EA und ein Diplominator. Der CrySpi gewinnt mit über 85 Prozent gewonnener Spiele sehr deutlich. Der Diplominator gewinnt ungefähr 14 Prozent der Spiele und der Stragotiator gewinnt $0,11$ Prozent der Spiele.
Wie bereits in den vorhergehenden Konstellationen festgestellt werden konnte, versendet der Diplominator eine sehr große Anzahl an Friedensangeboten (Abbildung 37 links). Im Vergleich hierzu versendet der CrySpi eine deutlich geringere Anzahl an Friedensangeboten und die Anzahl der Angebote des Stragotiators ist minimal. Die vom Stragotiator

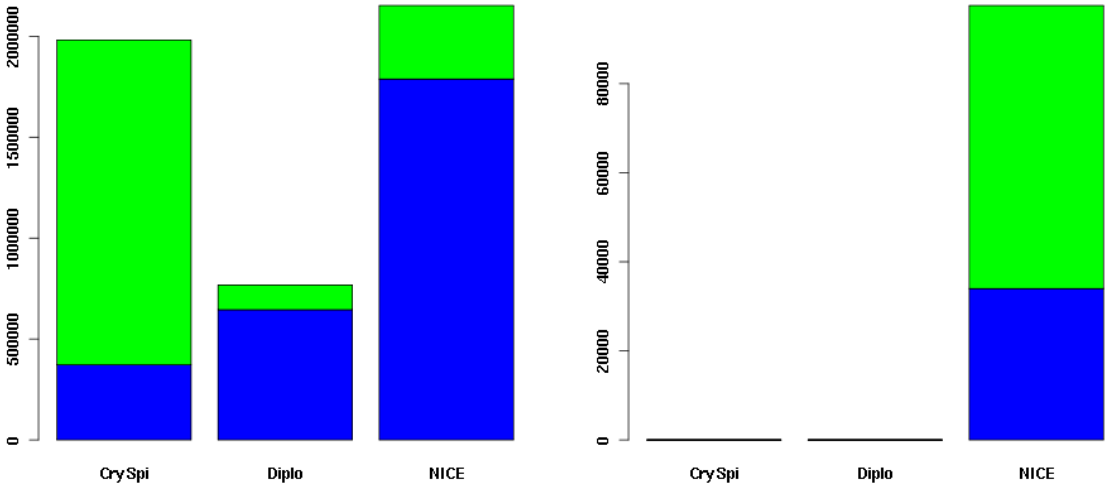


Abbildung 35: Statistikauswertung: Verhandlungen in Konstellation 2

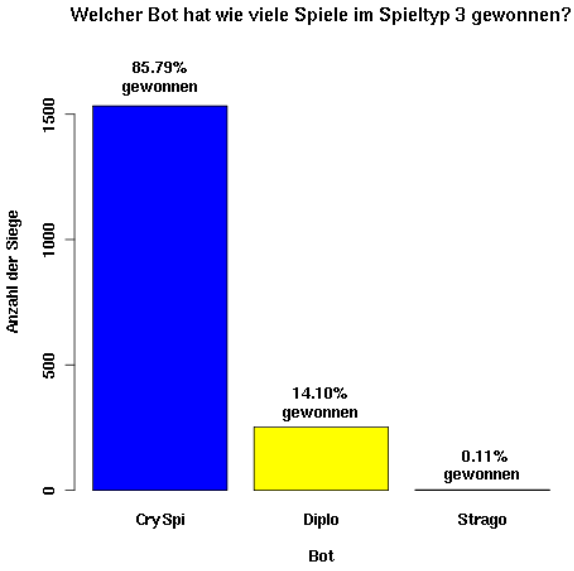


Abbildung 36: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 3

versandten Allianzangebote (Abbildung 37 rechts) werden mit einer sehr hohen Quote von ungefähr 80 Prozent akzeptiert. Kommt statt dem Stragotiator ohne EA der Stragotiator mit EA (Konstellation 4) zum Einsatz können sehr ähnliche Ergebnisse beobachtet werden. Die Anzahl der gewonnenen Spiele des CrySpi ist mit 83 Prozent leicht geringer und die des Diplominator mit 16,5 Prozent leicht höher als in der Konstellation 3. Der Stragotiator mit EA gewinnt mit 0,06 Prozent der Spiele weniger als der Stragotiator ohne EA mit 0,11 Prozent.

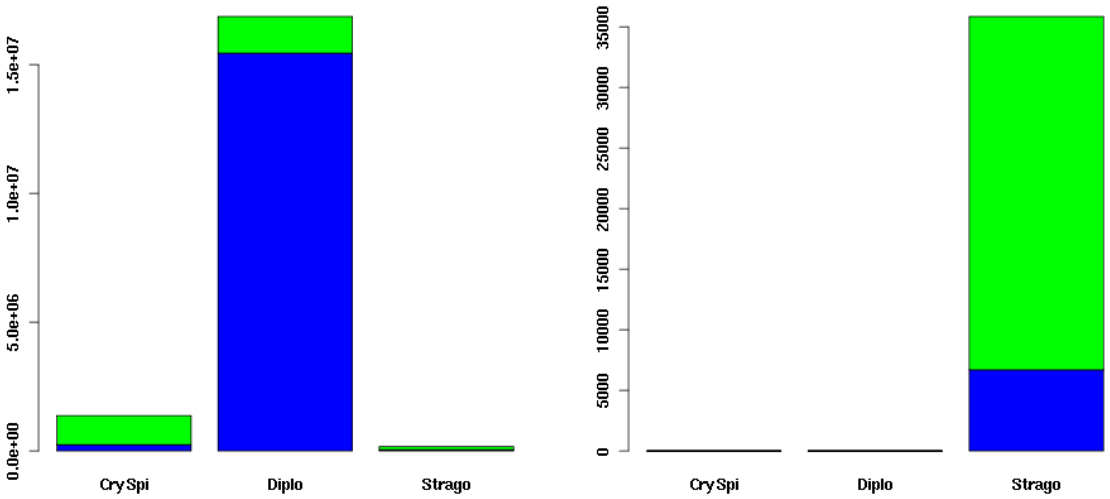


Abbildung 37: Statistikauswertung: Verhandlungen in Konstellation 3

6.1.1.6 Nice-Bot vs. Strago

In der Konstellation 8 (Abbildung 38) spielen ein Diplominator, drei Nice-Bots und drei Stragotiatoren. Die Nice-Bots gewinnen ungefähr drei Viertel aller Spiele und der Diplominator ein Viertel. Dem Stragotiator gelingt es lediglich sehr geringe 0.22 Prozent aller Spiele zu gewinnen. Dieser Wert ist allerdings höher als bei den Spielen des Stragotiators gegen den CrySpi und FrySpi.

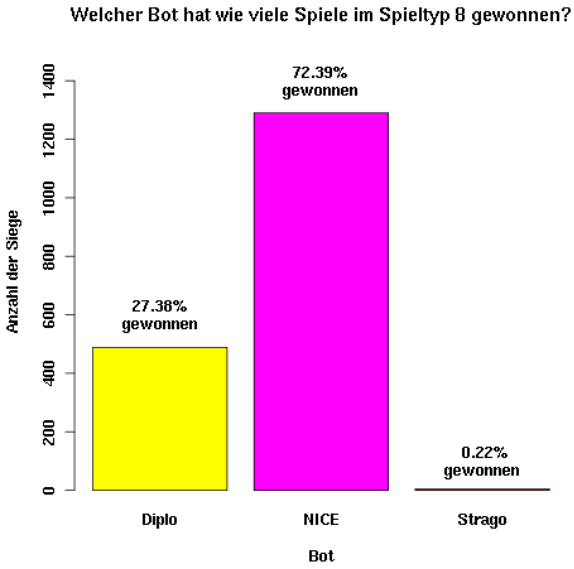


Abbildung 38: Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 8

Der Nicebot versendt im Vergleich zum Diplominator mehr als die doppelte Anzahl Friedensangebote (Abbildung 39 links). Der Stragotiator hingegen versendet nur eine

sehr geringe Anzahl Friedensangebote. Vergleicht man die Anzahl der versandten und akzeptierten Allianzangebote (Abbildung 39 rechts) so stellt man fest, dass diese in einem ähnlichen Umfang versandt wurden. Der Nicebot kann hierbei eine leicht stärkere Anzahl akzeptierter Vorschläge erreichen.

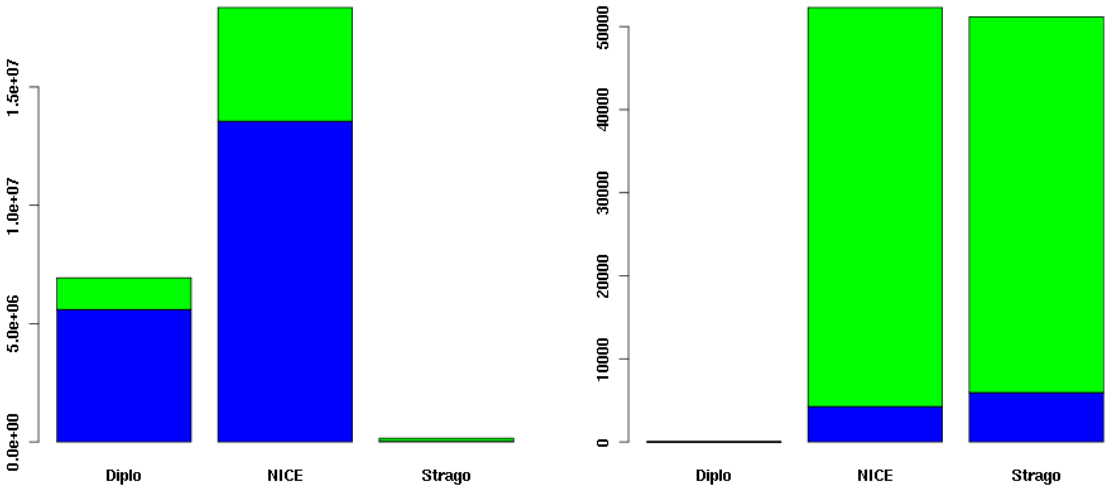


Abbildung 39: Statistikauswertung: Verhandlungen in Konstellation 8

6.1.2 Gesamtergebnis

In diesem Abschnitt werden die Ergebnisse der Spiele der unterschiedlichen Konstellationen zusammengeführt. Wie bereits bei den einzelnen Konstellation werden an dieser Stelle zunächst die Spielstärken der einzelnen Bots verglichen und dann die versandten Anfragen bezüglich Frieden, Allianz und Zugvorschlägen. Anschliessend werden die durchschnittliche Anzahl Supportcenter über 20 und 50 Jahre untersucht.
Wie bereits in den einzelnen Konstellationen deutlich wurde ist der CrySpi der spielstärkste Bot. Es gelang ihm die Leistung des FrySpi in jedem der Contests um einige Prozentpunkte zu übertreffen. Diplominator und Nice-Bot belegen den dritten und den vierten Platz. Der Stragotiator ohne EA und der Stragotiator mit EA konnten nur eine sehr geringe Anzahl an Spielen gewinnen und sind die schwächsten Bots bezüglich der Spielstärke. Hierbei ist zu beachten, dass der Stragotiator ohne EA spielstärker ist als der Stragotiator mit EA.

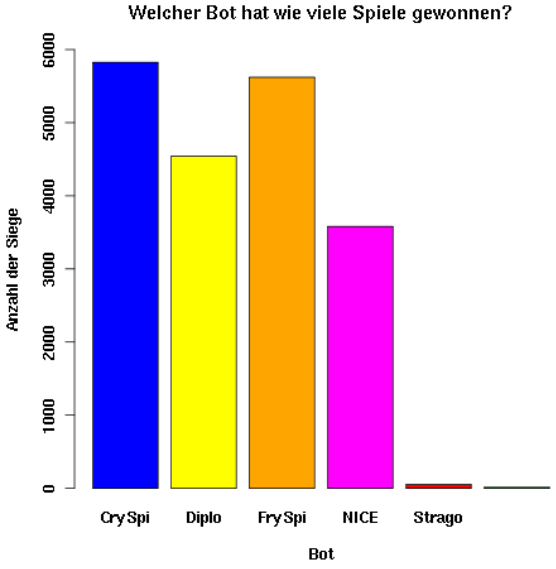


Abbildung 40: Statistikauswertung: Anzahl der gewonnenen Spiele

Betrachtet man die Anzahl der versandten Friedensangebote so fällt auf, dass Diplominator und der aufbauende Nice-Bot die meisten Friedensnachrichten versenden. CrySpi und FrySpi versenden deutlich weniger Friedensangebote und der Stragotiator mit und ohne EA die geringste Anzahl. Untersucht man die Anzahl der akzeptierten Friedensangebote erreichen der CrySpi und der FrySpi mit über 80 Prozent Akzeptanz die besten Ergebnisse. Der Stragotiator mit und ohne EA erzielen mit 76 bzw. 78 Prozent ein geringfügig schlechteres Ergebnis. Diplominator und Nice-Bot erzielen mit 14 und 26 Prozent das schwächste Testergebnis. Auffällig ist zudem, dass der Nice-Bot im Vergleich zum Diplominator trotz deutlich geringerer Anzahl an Angeboten eine höhere Akzeptanzquote erreicht.

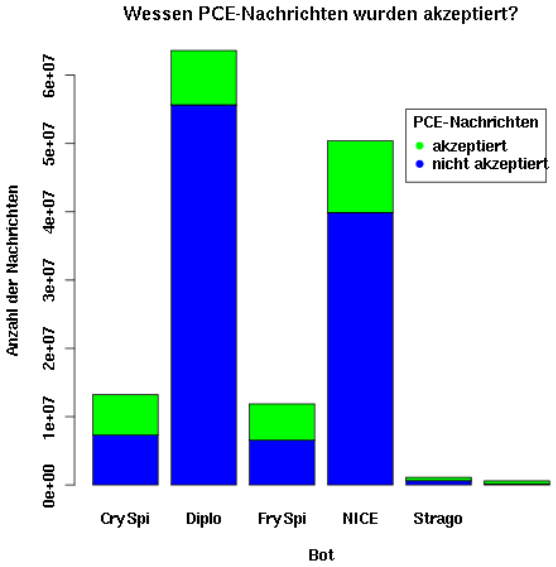


Abbildung 41: Statistikauswertung: Anzahl der durchschnittlichen Peace-Nachrichten

Untersucht man die Allianzangebote fällt auf, dass der Nice-Bot im Vergleich zum Stragotiator mehr als die doppelte Anzahl Allianzangebote versendet. Der Stragotiator ohne EA versendet einige Allianzangebote mehr als der Stragotiator mit EA und erreicht, ähnlich wie bei den Friedensangeboten, eine um ungefähr zwei Prozentpunkte bessere Akzeptanzquote. Drei von vier Allianzangebote des Nice-Bots werden akzeptiert, was für den Nice-Bot ein deutlich besseres Ergebnis darstellt als beim Versand von Friedensangeboten. CrySpi, FrySpi und Diplominator nutzen die Möglichkeit, Allianzangebote zu versenden nicht.

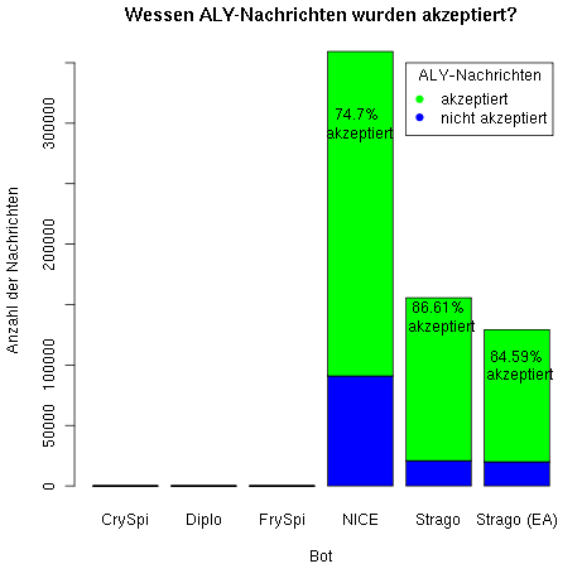


Abbildung 42: Statistikauswertung: Anzahl der durchschnittlichen Aly-Vorschläge

Nice-Bot und Diplominator versprechen am häufigsten Unterstützung führen diese jedoch in lediglich 2,9 Prozent (Nice) bzw 2.51 Prozent (Diplominator) durch. CrySpi und FrySpi nehmen ebenfalls verhältnismäßig viele Zugvorschläge an führen jedoch keinen der versprochenen Züge aus. Stragotiator mit EA und ohne EA nehmen zwar die geringste Anzahl an Zugvorschlägen an jedoch führen sie von diesen auch 25 Prozent (mit EA) und 27 Prozent (ohne EA) aus.

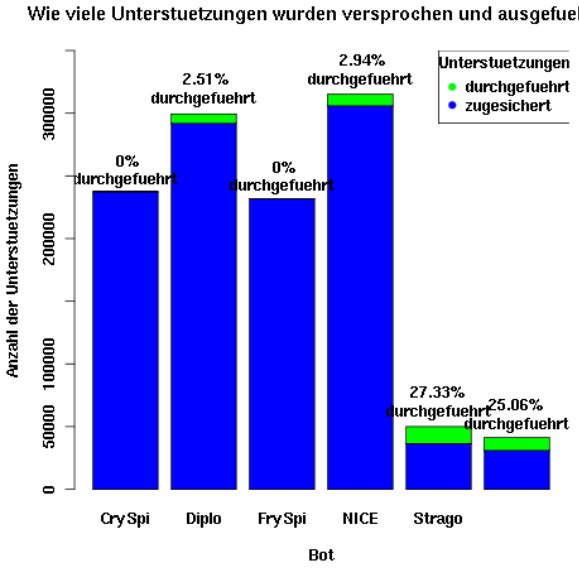


Abbildung 43: Statistikauswertung: Anzahl der durchschnittlichen akzeptierten und durchgeführten Zugvorschläge

Die Verläufe der durchschnittlichen Anzahl der Supplycenter der unterschiedlichen Bots über 20 Spieljahre und gemittelt über sämtliche Konfigurationen (Abbildung 44) bestätigt die vorhergehenden Analysen und zeigt die Ähnlichkeiten der Bots auf. So kann man sehr gut erkennen, dass FrySpi und CrySpi einen sehr ähnlichen Kurvenverlauf haben, Diplominator und Nice-Bot sowie Stragotiator mit und ohne EA. Außerdem erkennt man die leicht besseren Werte von CrySpi zu FrySpi und die etwas größeren Unterschiede zwischen Diplominator und Nice-Bot sowie FrySpi und CrySpi.

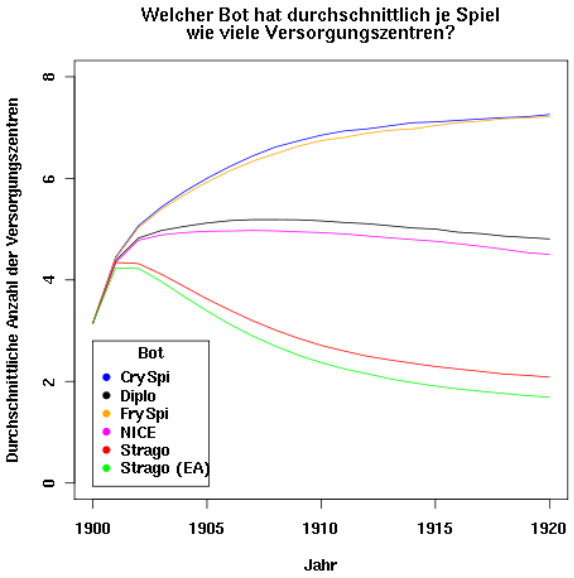


Abbildung 44: Statistikauswertung: Entwicklung Supplycenter in 20 Jahren

Untersucht man den Verlauf über 50 statt 20 Spieljahre (Abbildung 45) kann man erkennen, dass sich die Funktionswerte von Stragotiator mit und ohne EA zunehmend - wenn auch leicht - auseinander bewegen. Bei den Werten des Nice-Bot und des Diplominators ist ähnliches zu beobachten. Auffällig ist, dass sich die Funktionswerte des CrySpi und des FrySpi im Zeitraum um das Jahr 1930 schneiden und der FrySpi-Wert von da an leicht bessere Werte erzielt. Da viele der Spiele bereits nach ungefähr 20 Jahren beendet sind nimmt die Anzahl der Messpunkte mit steigendem Jahr ab, weshalb die Schwankungen in den Kurven größer werden.

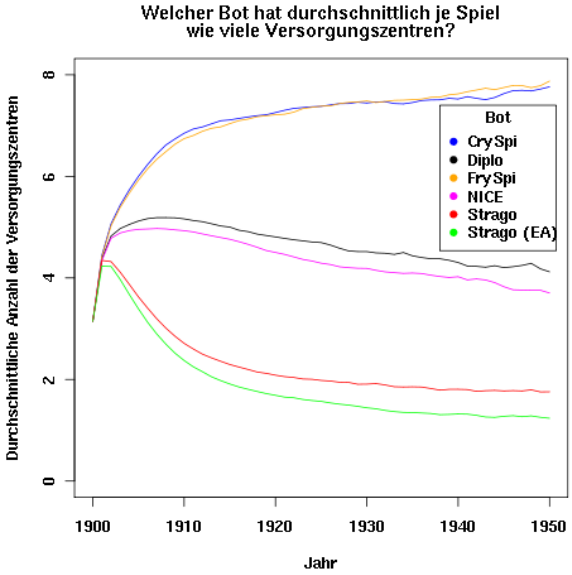


Abbildung 45: Statistikauswertung: Entwicklung Supplycenter in 50 Jahren

6.1.3 Angriffs- und Erfolgsanalyse der Bots

Ein wichtiges Kriterium zur Beurteilung der Spielstärke eines Spielers ist die Effektivität, mit derer Angriffe durchgeführt. Anhand dieser Auswertung kann abgeschätzt werden, wie gut die Spielaufteilung und Zugauswahl der Spieler ist.

6.1.3.1 CrySpi

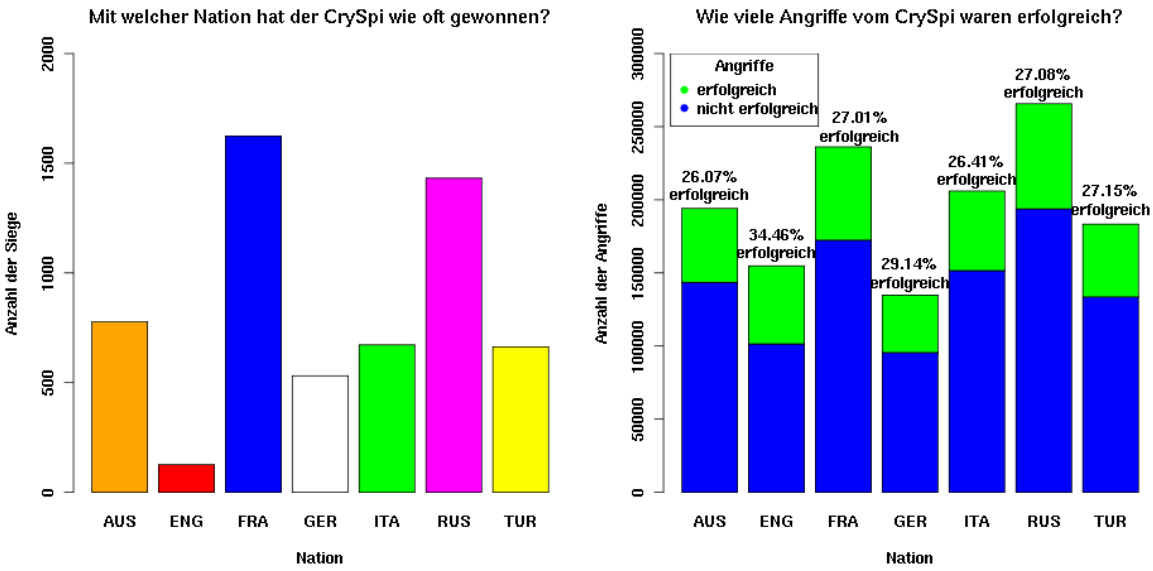


Abbildung 46: Statistikauswertung: Angriffe und Erfolge des CrySpi

Der CrySpi war besonders erfolgreich in den Spielen, in welchen er Frankreich oder Russland spielte (Abbildung 46). Mittelmäßige Leistungen erbrachte er mit den Mächten Österreich, Italien und Türkei, wobei er mit Österreich eine kleine Mehrleistung erzielen konnte. Im Vergleich zu der Anzahl der Siege mit den beiden starken Mächten gelingt es dem CrySpi mit diesen Mächten lediglich die halbe Anzahl Spiele, also ungefähr 750 statt 1500 Spiele zu gewinnen. Die Leistung mit Deutschland fällt mit 500 Spielen noch weiter ab und deutlich am spielschwächsten ist der CrySpi mit der Macht England. Dies kann unter Umständen durch die Schwierigkeit begründet sein Convoys durchzuführen. Untersucht man die Angriffe des CrySpi, kann man eine Korrelation zwischen der Spielstärke mit einer Macht und der Anzahl der durchgeführten Angriffe erkennen. Mit Frankreich und Russland wurden die meisten Angriffe durchgeführt, das Mittelfeld bilden Österreich, Italien und die Türkei und mit England und Deutschland wurden die wenigsten Angriffe durchgeführt. Die Erfolgsquoten der Angriffe liegen bei ungefähr 26 bis 27 Prozent. Die beiden schwachen Mächte Deutschland und England erzielen die höchsten Erfolgsquoten. Deutschland erzielt eine Erfolgsquote von 29 Prozent und England von 34 Prozent.

6.1.3.2 FrySpi

Der FrySpi besitzt ein sehr ähnliches Machtverhältnis wie der CrySpi (Abbildung 47). Er ist stark mit Frankreich und Russland, mittelmäßig mit Österreich, Italien und der Türkei und am schwächsten mit Deutschland und England. Das beim CrySpi bestehende Mächtegleichgewicht zwischen Italien und der Türkei besteht in geringerem Ausmaß:

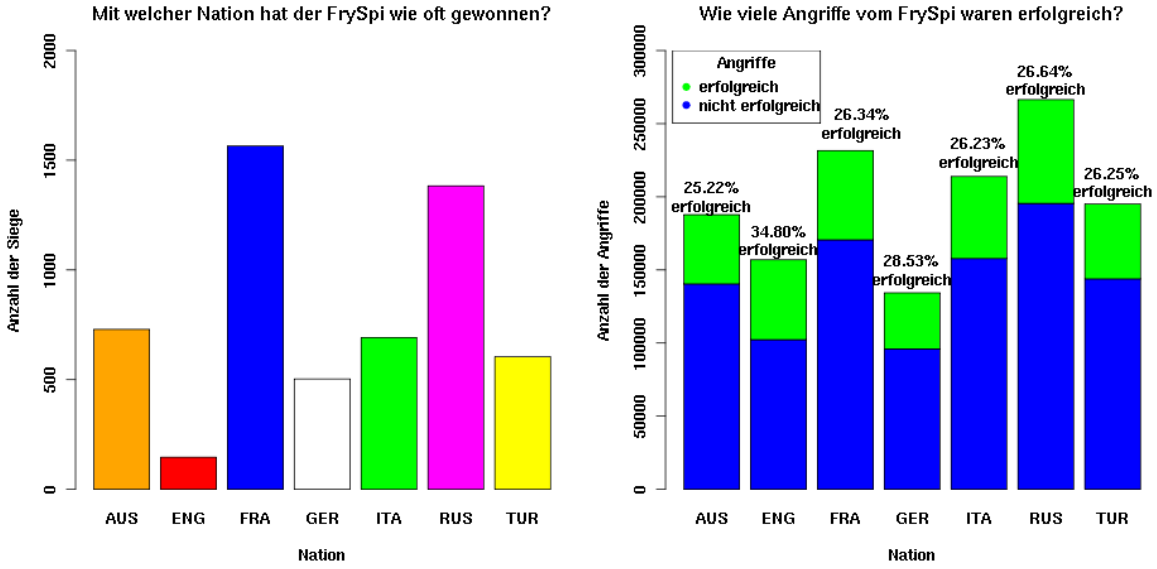


Abbildung 47: Statistikauswertung: Angriffe und Erfolge des FrySpi

die Anzahl der Siege mit der Türkei fällt denen mit Italien gegenüber leicht ab. Erneut ist eine Korrelation festzustellen zwischen der Anzahl der durchgeführten Angriffe und der Anzahl der Siege mit den unterschiedlichen Mächten. Die Erfolgsquote der Angriffe liegt zwischen 25 und 27 Prozent. Ausreißer sind Deutschland mit 28,5 Prozent und England mit fast 35 Prozent. Die enorme Korrelation zwischen Angriffen und gewonnen Spielen zeigt deutlich, dass ein strategischer aggressiver Bot, in Spielen mit geringem Presslevel, eine große Chance auf Erfolg hat.

6.1.3.3 Diplominator

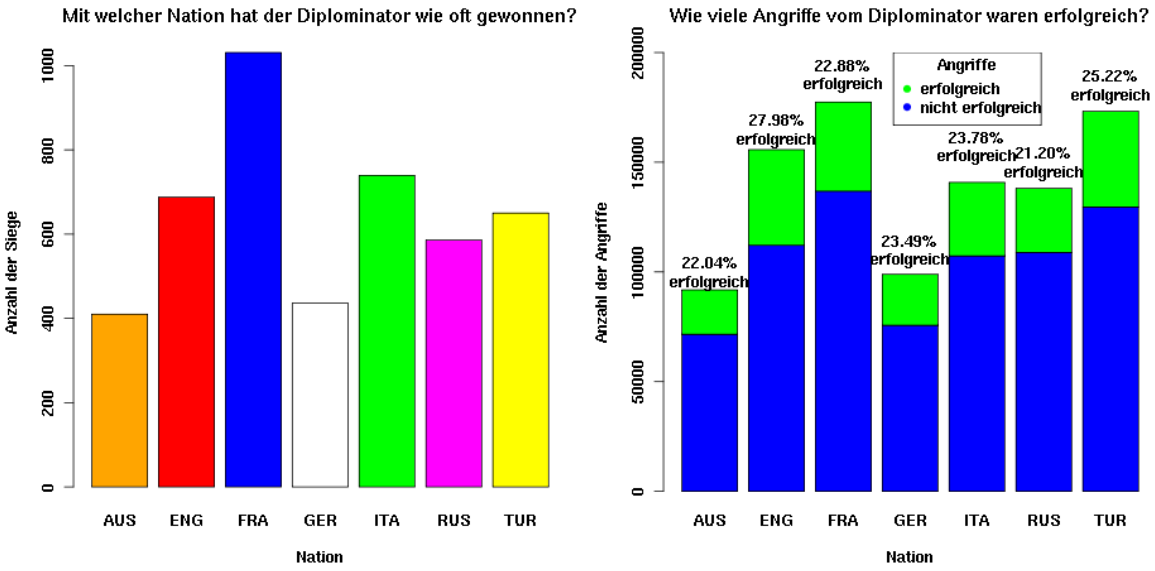


Abbildung 48: Statistikauswertung: Angriffe und Erfolge des Diplominator

Der Diplominator kann beim Spielen mit Frankreich deutlich mehr Spiele gewinnen als mit den anderen Ländern (Abbildung 48). Mittelstarke Ergebnisse erzielt er mit England, Italien, Russland und der Türkei, wobei die Werte für Russland und Türkei leicht abfallen. Mit den Ländern Österreich und Deutschland kann er hingegen deutlich weniger Spiele für sich entscheiden. Im Vergleich zu den Ergebnissen beim CrySpi und FrySpi fällt besonders auf, dass der Diplominator, was die Verteilung der gewonnen Spiele angeht, deutlich ausgeglichener ist. Der FrySpi ist mit Russland deutlich spielstärker als mit der Türkei oder Italien. Beim Diplominator hingegen kann man ein relativ ähnliches Gleichgewicht dieser Mächte feststellen. Außerdem gehört England, welches beim CrySpi die schwächste Macht ist, hier zu den eher spielstarken Mächten. Auch beim Diplominator kann man eine starke Korrelation zwischen der Anzahl der Angriffe und der Spielstärke mit den diversen Ländern feststellen. Die Anzahl der gewonnenen Angriffe schwankt zwischen 21 Prozent bei Russland und 24 Prozent bei Italien. Ausreißer sind die Türkei mit 25 Prozent gewonnener Spiele und England mit 28 Prozent.

6.1.3.4 Nice

Der Nice-Bot bietet ein wesentlich unausgewogeneres Bild als der Diplominator (Abbildung 49). Zwar erzielt er gute Werte mit Frankreich und Italien, besitzt jedoch im Vergleich zum Diplominator schwächere Werte in Bezug auf Russland und die Türkei und sehr schwache Werte bezüglich Österreich, England und Deutschland. Die Anzahl der erfolgreichen Angriffe schwanken zwischen $18,7$ und $26,5$ Prozent. Die schwachen Ergebnisse im Spiel mit Österreich, England und Deutschland, können unter anderem durch zwei Umstände begründet sein. Die schwache Performanz mit England kann auf die Nichtnutzung der Convoy-Funktion zurückgeführt werden. Ohne die Convoy-Funktion, ist es für England erheblich schwieriger einen Sieg zu erzielen. Beim Spielen der Länder Österreich und Deutschland, welche sehr zentral auf dem Spielfeld liegen, ist die Verhandlung von besonderer Wichtigkeit. Ein Alleingang ist mit diesen Startländern, besonders am Anfang des Spiels, fast unmöglich. Der Nice-Bot wurde dahingehend

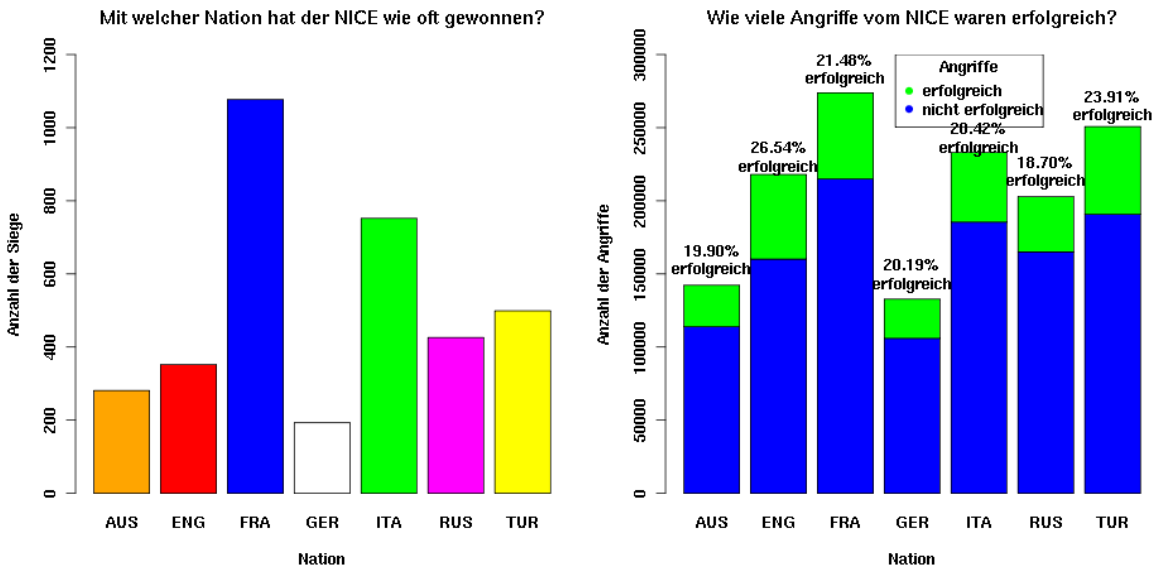


Abbildung 49: Statistikauswertung: Angriffe und Erfolge des Nice

entwickelt, dass er am Anfang besondere Bündnispartner sucht, zu welchen er ein besonderes Vertrauensverhältnis hat. Zu großes Vertrauen in den Bündnispartner kann schnell dazu führen, die Überwachung strategisch wichtiger Punkte zu vernachlässigen oder sogar ganz dem Bündnispartner zu überlassen. Je interessanter und wichtiger der Punkt ist, desto höher ist auch die Gefahr, dass der Bündnispartner jenes Bündnis bricht um mit dem leicht eroberten Land eigene Interessen zu verfolgen. Ein unausgewogenes Verhältnis zwischen Miss- und Vertrauen den anderen Spielern gegenüber, kann sich, insbesondere am Anfang eines jeden Spiels, in sehr hohem Maße auf die Erfolgsaussichten auswirken. Ein zu großes Vertrauen an die Alliierten kann an dieser Stelle als möglicher Grund für ein schwächeres Abschneiden angeführt werden.

6.1.3.5 Strago mit EA

Der Stragotiator konnte im gesamten Contest jeweils drei Spiele mit Deutschland, Italien und Russland, zwei Spiele mit Österreich und Frankreich und keins mit England für sich entscheiden (Abbildung 50). In wie weit diese Zahlen statistisch belastbar sind ist fraglich, da die absoluten Zahlwerte sehr gering ausfallen. Die Erfolgsquoten der Angriffe des Stragotiators mit EA sind stark abhängig von der gespielten Macht. mit Österreich erreicht der Stragotiator zum Beispiel lediglich eine Erfolgsquote von 13.65 Prozent und mit England eine Erfolgsquote von 26 Prozent, was einer doppelt so hohen Erfolgsquote entspricht.

6.1.3.6 Strago ohne EA

Der Stragotiator ohne EA kann deutlich mehr Spiele für sich entscheiden als der Stragotiator mit EA. Die deutlich stärkste Macht ist Österreich, gefolgt von Frankreich, Deutschland, Italien und der Türkei (Abbildung 51). Die Leistung mit Russland fällt deutlich ab und mit England konnten keine Gewinne erzielt werden. Im Vergleich zum Stragotiator mit EA kann der Stragotiator bessere Ergebnisse bezüglich der Erfolgsquote der Angriffe verzeichnen. Absehend von der Türkei liegen die Erfolgsquoten teils

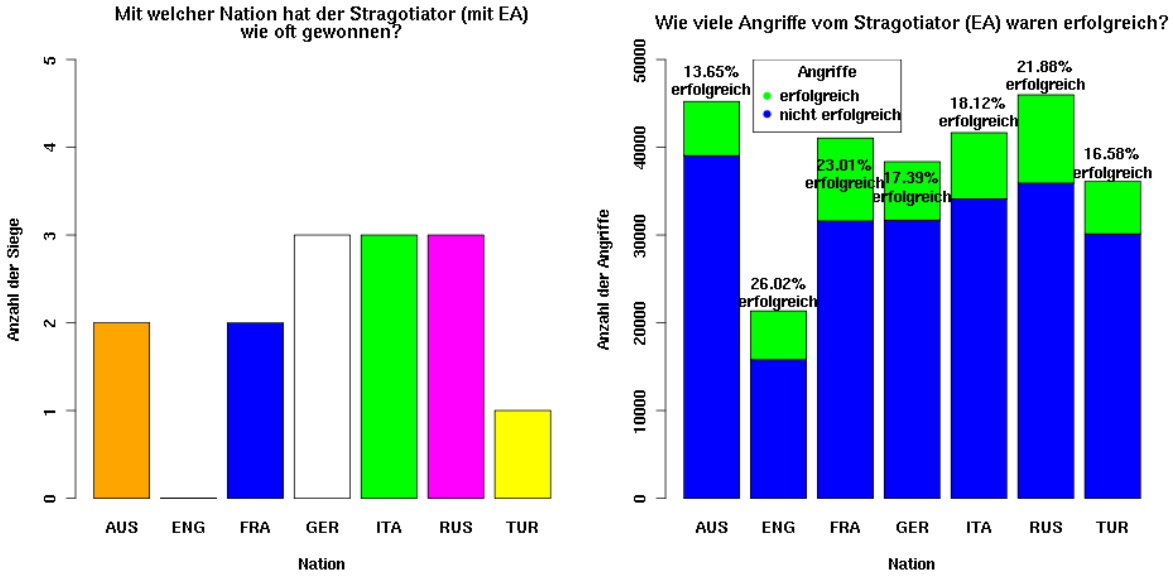


Abbildung 50: Statistikauswertung: Angriffe und Erfolge des Stragotiators mit EA

deutlich über 20 Prozent. Deutlich werden die Unterschiede insbesonders, wenn man die Werte der stärksten gespielten Macht Österreich vergleicht. Hier erzielt der Stragotiator ohne EA eine um fast zehn Prozentpunkte bessere Erfolgsquote. Der Stragotiator, sowohl mit EA als auch ohne EA, hatte durch die Wahl der Implementierung des *Klassikers* eine besondere Herausforderung zu meistern. Die meisten bestehenden Diplomacy-Bots haben ihren Fokus auf strategisch günstiger Entscheidungsfindung und unterstützen lediglich ein sehr geringes Presslevel. Zudem verfolgen sehr viele Bots eine aggressive Expansionsstrategie, welche sich nur sehr schwierig mit der defensiven wohlwollenden Art des *Klassikers* vereinbaren lässt. Die in vielen Spielen hervorragenden Ergebnisse bezüglich Kommunikation und Verhandlungskunst attestieren dem Stragotiator erhebliches diplomatisches Geschick. Letztendlich ist es die zu gutmütige Art des Stragotiators, welche das Siegen für ihn so schwierig macht. Das angestrebte Ziel des gemeinsamen Sieges mit einem Bündnispartner wird viel zu selten erreicht. Die Nutzung eines EA bei der Optimierung der Züge des Stragotiators macht durchaus Sinn, jedoch wurden bereits beim Entwurf und der Realisierung des EA klar, dass es sich hierbei um ein sehr komplexes Unterfangen handelt. Eine mögliche Anpassung dahingehend, die Wertigkeit entfernter aber sehr wichtiger Ziele anzupassen, scheint gut dafür geeignet die Leistung des EA erheblich zu verbessern.

6.1.4 Fazit

Untersucht man die Spielstärke an Hand der Anzahl der gewonnenen Spiele in den unterschiedlichen Konstellationen, zeichnet sich ein deutliches Bild ab. Die von der FrySpi-Gruppe entworfenen Bots FrySpi und CrySpi gewinnen teils deutlich sämtliche Wettbewerbe. Da es jedoch nicht das primäre Ziel der Projektgruppe war, einen möglichst spielstarken Computerspieler zu kreieren, ist die Menschenähnlichkeit mindestens von ebenbürtiger Bedeutung. FrySpi, CrySpi und Diplominator beschränken sich bei den Verhandlungen, welche als Maßstaß für die Menschenähnlichkeit dienen sollen, auf die Nutzung von Friedensangeboten. Nice-Bot und Stragotiator nutzen darüber hinaus die Möglichkeit Allianzangebote und Zugvorschläge zu senden und im Fall der Zugvorschläge ebenfalls auch durchzuführen. Dem Stragotiator gelang es hierbei besser als

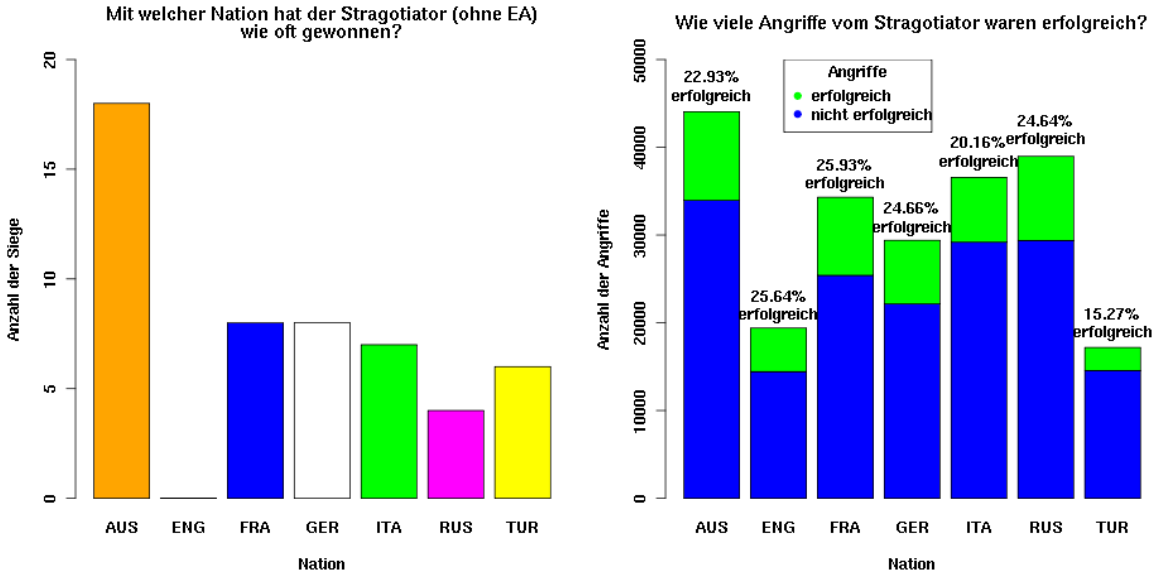


Abbildung 51: Statistikauswertung: Angriffe und Erfolge des Stragotiators ohne EA

dem Nice-Bot trotz einer geringen Anzahl an versandten Nachrichten stets eine hohe Akzeptanzquote zu erzielen. Der Stragotiator kann somit als der Bot mit dem höchsten Verhandlungsgeschick betrachtet werden. Leider ist die Spielstärke des Stragotiators eher gering und es besteht ein erhebliches Ungleichgewicht zwischen Verhandlungsstärke und Spielstärke, was relativ ungewöhnlich ist. Der Nice-Bot hingegen versendet sehr viele Friedens- Allianz- und Zugvorschläge allerdings liegt die Akzeptierungsquote meistens deutlich unter der des Stragotiators. Er besitzt dafür im Gegenzug eine Spielstärke, welche die des Stragotiators deutlich übertrifft. Das Fazit der Statistikauswertung ist, dass der FrySpi der spielstärkste und verhandlungsschwächste Bot ist, der Nice-Bot der nach dem FrySpi der zweitstärkste und nach dem Stragotiator der zweitverhandlungsstärkste ist und der Stragotiator der spielschwächste und verhandlungsstärkste der Bots.

6.2 Contest Software

Zu Anfang des Diplomacy-Projekts gab es drei Gruppen, die jeweils einen eigenen Bot entwerfen wollten. Auch wenn sie bei der Entwicklung dieser Bots die Menschenähnlichkeit in den Vordergrund stellten, so kam doch auch schnell der Wunsch auf, die Bots gegeneinander antreten zu lassen. Es sollten in kurzen Abständen die aktuellen Entwicklungsstände der Bots gegeneinander antreten, um zu schauen, welche am Spielstärksten seien. Der überwiegende Teil der Teilnehmer sah den Wettkampf als eine zusätzliche Motivation bei der Entwicklung an.

Mit der Umsetzung wurde eine Person beauftragt. Umgesetzt wurde die Fähigkeit, die Bots aus dem Versionsverwaltungssystem herunterzuladen, in ausführbaren Code zu übersetzen, diesen gegeneinander antreten zu lassen und die Ergebnisse zu protokollieren.

Leider stellte sich ein stabiles Ausführen dieses Wettbewerbs – aus verschiedenen Gründen – als schwierig heraus. Ein Grund war die Verwendung des Parlance-Servers. Er wurde anstelle des DAIDE-Servers aus folgenden Gründen verwendet: Einmal lief er ohne Emulation unter Linux. Da der Rechner, der uns zum Ausführen der Wettbewerbe

vom Lehrstuhl zur Verfügung gestellt wurde, unter Linux lief, lag es nahe, den Parlance-Server zu verwenden. Ein weiterer Grund war die Ausführbarkeit ohne weitere grafische Oberfläche. Da die Software im Hintergrund laufen sollte, war auch dies ein Argument für den Parlance-Server. Leider sollte diese Wahl auch für einige Probleme sorgen. So lief der Server sehr instabil. Bei Spielen, die länger als 100 Spieljahre liefen, wurde er so langsam, dass man die Spiele beenden musste. Zeitgrenzen für Züge unter 10 Sekunden brachten ihn ebenfalls direkt zum Erliegen. Außerdem stellte sich gerade bei den Bots, welche die Kommunikations-API von Henrik Bylund benutzten, ein Problem heraus: Sie konnten sich sehr häufig nicht mit dem Server verbinden. Ohne weiteren Aufwand war es nicht zuverlässig möglich, herauszufinden, wann sich ein Bot verbunden hatte und wann nicht. Der notwendige Aufwand, um diese Probleme anzugehen, konnte nicht mehr investiert werden, da der betroffene Entwickler sich wieder der Bot-Gruppe anschließen sollte.

Zum Ende des Projekts gab es dann aber nochmalig Grund für die automatisierte Durchführung von Spielen. Die entwickelten Bots sollten diesmal gegeneinander antreten, um herauszufinden, ob das erzielte Spielverhalten mit dem Geplanten in Übereinkunft gebracht wurde. Wurden beispielsweise wirklich Bots erschaffen, die Allianzen schmieden, sich gegenseitig unterstützen und Zugvorschläge unterbreiten? Dazu sollten die Bots in noch zu bestimmenden Konstellationen gegeneinander antreten und die dabei anfallenden Logdateien von einer Statistiksoftware entsprechend analysiert werden. Bei der Entwicklung wurde die bestehende Wettbewerbs-Software mit Blick auf deren Probleme erweitert. Die Probleme mit dem Parlance-Server wurden gelöst, indem nunmehr der DAIDE-Server genutzt wurde und die Software entsprechend angepasst wurde. Die Verbindungsprobleme der Bots wurden dadurch gelöst, dass eine Softwarekomponente entwickelt wurde, die sich mit dem Server als Beobachter (Observer) verbindet und immer benachrichtigt wird, wenn sich ein Bot verbindet. Der Verbindungsaufbau wurde solange wiederholt bis er erfolgreich zustande kam. Hierdurch konnten ausreichend stabile Wettbewerbe durchgeführt werden.

6.3 Durchführung der Contests

Nun existierte eine Software, die in verlässlicher Art und Weise Spiele anhand von Beschreibungsdateien durchführen konnte. In diesen Beschreibungsdateien stand die Konfiguration des Spiels: die verwendeten Bots, deren Parameter und die des Servers.

Ein durchgeführtes Spiel führte zu einer *Diplomacy Archiv Datei*, der sogenannten DAR-Datei. In dieser befinden sich alle Informationen, die während des Spiels erzeugt wurden. Also alle Log-Dateien des Spiels, die verwendeten Bots, deren Parameter und die des Servers.

Was noch für die Durchführung des Wettbewerbs fehlte, waren die konkreten Spielbeschreibungsdateien. Mithilfe eines kleinen Tools wurden diese, für die einzelnen Wettbewerbe erstellt.

6.4 Turing Test

Zuzüglich zur Durchführung der Statistikwertung wurde zudem ein Turingtest durchgeführt. Es sollte untersucht werden, wie die entwickelten Bots sowohl von menschlichen Mitspielern im Spiel als auch von externen Beobachtern, welche lediglich die durchgeführten Züge und nicht jedoch die Verhandlungen sehen konnten, bezüglich Spielstärke und Menschenähnlichkeit bewertet wurden. Zudem hatten die Spieler und Beobachter die Möglichkeit, Teilnehmer des Spiels als Mensch beziehungsweise Bot einzustufen.

6.4.1 Durchführung

Vor der Durchführung des Turingtest wurden zwei PG-Teilnehmer ausgewählt, welche sich um die Organisation während des Test kümmerten. Sie hatten die Aufgabe, in mehreren Runden jeweils drei PG-Teilnehmer randomisiert auszuwählen, welche dann als potentielle Mitspieler in einem Diplomacyspiel in separate Räume geführt wurden. Für jeden der potentiellen Mitspieler wurde wiederum randomisiert bestimmt, ob dieser aktiv am Spiel teilnehmen oder nur als Beobachter fungieren sollte. Die restlichen PG-Teilnehmer befanden sich in einem Raum, in welchem über ein Mappingtool der Spielverlauf nachvollzogen werden konnte. An die Spieler und Beobachter wurden unterschiedliche Fragebögen verteilt und die Organisatoren starteten ein Diplomacyspiel, wobei bis zu drei der sieben Spielslots von den potentiellen Spielern belegt und die restlichen mit den entworfenen Bots aufgefüllt wurden. Hierbei wurde darauf geachtet, dass jeder der entworfenen Bots zumindest einmal vertreten war. Es wurden jeweils die ersten fünf Spieljahre eines Spiels untersucht und bewertet. Da die Organisatoren Kenntnis über die Belegung der menschlichen und computergesteuerten Mitspieler hatten nahmen sie nicht aktiv am Bewertungsprozess teil. Insgesamt wurden vier Spiele durchgeführt und bewertet. Im ersten Spiel gab es sieben Beobachter und in den folgenden drei Spielen jeweils vier Beobachter. Am ersten und vierten Spiel nahmen zwei menschliche Spieler aktiv teil, im zweiten und dritten Spiel waren es zwei beziehungsweise ein menschlicher Spieler.

6.4.2 Ergebnisse des Turingtest

Die Spielstärke und Menschenähnlichkeit der Bots wurde jeweils auf einer Skala von eins bis fünf bewertet. Hierbei repräsentierte der Wert eins einen besonders spielschwachen beziehungsweise unmenschlichen Bot und der Wert fünf dementsprechend einen sehr spielstarken beziehungsweise sehr menschenähnlich wirkenden Spieler. Außerdem wird die Bewertung der menschlichen Spieler durch die Beobachter und durch die anderen menschlichen Spieler dargestellt. Durch eine schwarze horizontale Linie wird in jedem Balken der Median markiert und das rote Karo markiert den entsprechenden Mittelwert.

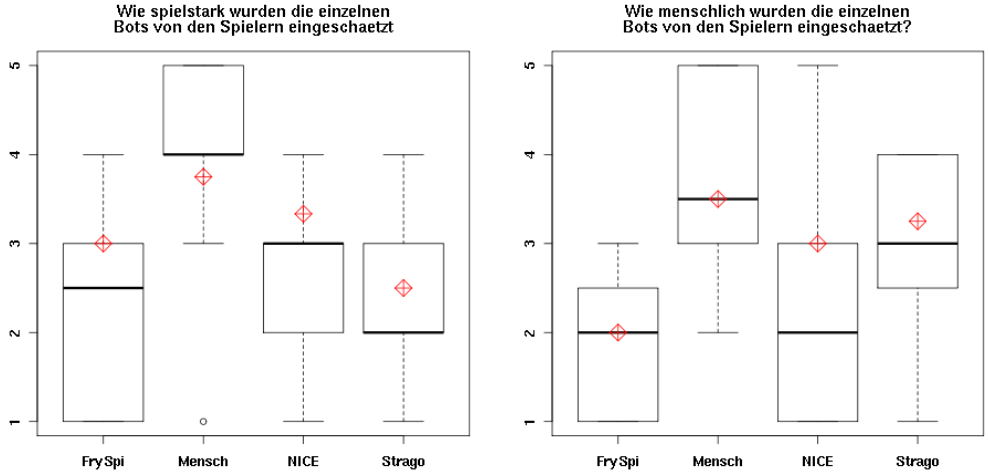


Abbildung 52: Turingtest: Spielerbewertung der Spielstärke und Menschenähnlichkeit

Untersucht man die Bewertung der Spielstärke durch die Spieler (Abbildung 52), stellt man fest, dass die menschlichen Spieler als die spielstärkste Einheit bewertet wurden.

Der Median besitzt hier einen sehr starken Wert von vier und der zugehörige Mittelwert ist nur unwesentlich geringer. Die drei Bots werden hingegen von den Spielern als spielschwächer eingeschätzt. Die beste Bewertung erhielt der Nice-Bot, welcher mit einem Wert von 3.3 im Durchschnitt etwas besser bewertet wurde als der FrySpi, welcher im Schnitt drei Punkte erzielen konnte. Der Stragotiator schließt mit einem Mittelwert von 2.5 Punkten am schwächsten ab. Auffällig ist zudem, dass die Bewertungen der drei Bots ein breites Wertespektrum abdecken. Dieses umfasst spielschwache, durchschnittliche und spielstarke Bewertungen. Ein im Vergleich zur Bewertung der menschlichen Spieler breites Spektrum kann ein Indikator dafür sein, dass die Qualität der Entscheidungen der Bots, je nach Spielsituation, Schwankungen unterlegen ist.
Betrachtet man die Einschätzung der Menschenähnlichkeit durch die Spieler, kann festgestellt werden, dass der Stragotiator, welcher von diesen als spielschwächster Bot empfunden wurde, ein großes Maß an empfundener Menschenähnlichkeit besitzt. Die durchschnittliche Bewertung von 3.3 ist zwar für sich alleinestehend noch nicht überragend, vergleicht man diesen Wert jedoch mit dem Wert 3.5, jenem, welcher von den menschlichen Spielern erzielt wurde, beträgt die Differenz zwischen dem als am menschenähnlichsten empfundenen Bot und einem menschlichen Spieler geringe 0.2 Punkte. Der Nice-Bot erzielt mit drei Punkten ein leicht schwächeres aber dennoch solides Ergebnis. Als am wenigsten menschlich wurde der FrySpi empfunden, dessen Verhalten als eher unmenschlich gewertet wurde.

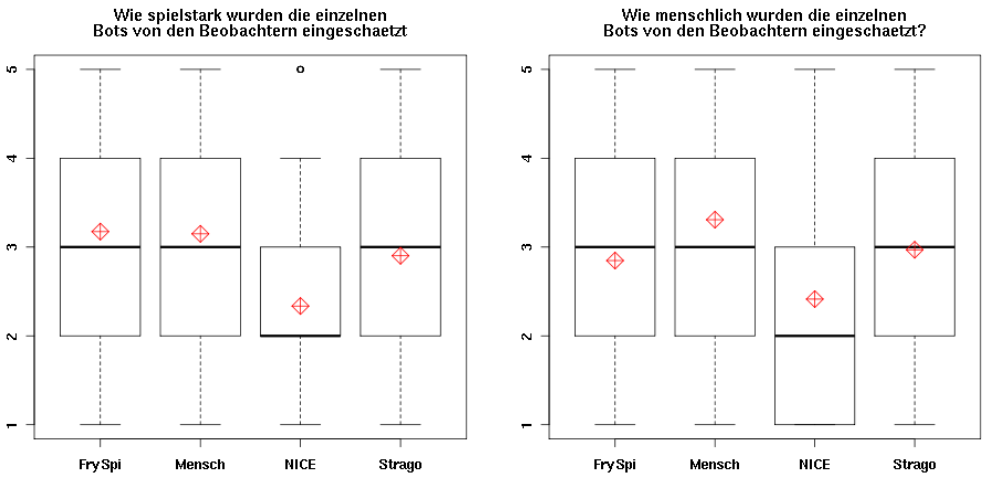


Abbildung 53: Turingtest: Beobachterbewertung der Spielstärke und Menschenähnlichkeit

Bei der Untersuchung der Bewertungen durch die Beobachter (Abbildung 53) fällt eine extreme Korrelation zwischen der Bewertung der Spielstärke und jener der Menschenähnlichkeit auf. Dies kann darauf zurückzuführen sein, dass die Beobachtergruppe die Verhandlungen zwischen den einzelnen Spielern nicht sehen konnte und eine menschliche Spielweise mit einer besonders klugen, erfolgreichen Spielweise gleichsetzten. Zudem konnte es nicht nicht vermieden werden, dass durch gegenseitige Absprachen zwischen den Beobachtern die Meinungsfindung beeinflusst wurde. Bei der Spielstärke fällt besonders auf, dass die menschlichen Spieler von den Beobachtern als deutlich spielschwächer eingestuft wurden, als dies bei der Bewertung durch die Mitspieler der Fall war. Zudem ist interessant, dass der Nice-Bot, welcher von den Spielern als spielstärkster Bot eingeschätzt wurde, nun mehr als am spielschwächsten bewertet wurde. Dieses kann vielleicht dadurch bedingt sein, dass die Spielergruppe auch die Verhandlungen in ihre Bewertung der Bots mit einfliessen lassen konnte. Die Bewertung der Menschenähnlichkeit

war auf Grund der genannten fehlenden Kenntnis der Verhandlungen sowie der kurzen Spieldauer von wenigen Jahren sehr schwierig. Wie bereits dargestellt wird anscheinend eine spielstarke Spielweise einer menschenähnlichen gleichgesetzt.

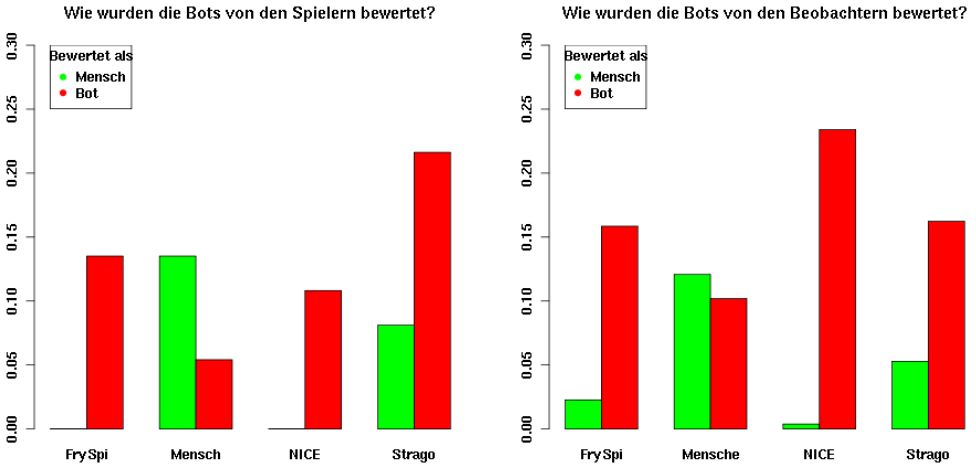


Abbildung 54: Turingtest: Einschätzung Mensch oder Bot

Von den Menschen wird der Stragotiator in fast 25 Prozent der Fälle als Bot bewertet (Abbildung 54). Der FrySpi wird in 14 Prozent der Fälle als Bot eingeschätzt und der Nice-Bot erzielt mit elf Prozent den besten Wert der drei Bots. Die menschlichen Mitspieler werden in fünf Prozent der Fälle als Bot identifiziert, können jedoch durch ihre Spielweise in 14 Prozent der Fälle als Mensch erkannt werden. FrySpi und Nice-Bot können nach Ansicht der Spieler nicht als Mensch überzeugen und werden nicht als Menschen eingeschätzt. Der Stragotiator, welcher am häufigsten als Bot eingeschätzt wird, erzielt zudem interessanterweise mit acht Prozent ein relativ gutes Ergebnis und kann als einziger Bot die menschlichen Mitspieler teilweise von seiner menschlichen Spielweise überzeugen. Die Beobachtung durch die Gruppe der menschlichen Spieler steht, bezogen auf die Einschätzung der Menschlichkeit, in Korrelation zu der durch die beobachtende Gruppe. Menschen und Stragotiatoren werden etwas seltener als solche eingeschätzt und FrySpi und Nice-Bot gelingt es geringfügig Beobachter von ihrer Menschlichkeit zu überzeugen. Dies kann, ähnlich wie bei den anderen Bewertungen durch die Beobachter, dadurch begründet sein, dass Verhandlungen nicht gesehen werden und somit auch nicht als potentiell unmenschlich wirken können. Die Klassifizierungsquote als Bot nimmt bei FrySpi, Mensch und Nice-Bot zu und beim Stragotiator ab. Es kommt beim Nice-Bot zu einem erheblichen Ansprung der Einschätzung als Bot. Dies kann durch Kenntnis des statischen Eröffnungsbuches und Absprache unter den Beobachtern begründet sein.

6.4.3 Fazit

In diesem Abschnitt wird zunächst über die Durchführung des Turingtest reflektiert und dann die wesentlichen Ergebnisse aus den Statistiken zusammengefasst. Bei der Auswertung der Turingtestdaten fielen insbesondere zwei Punkte auf, welche sich maßgeblich auf die Qualität und den Aufwand zur Datenerhebung auswirkten. Durch Aufteilung der Gesamtgruppe in zwei Teilgruppen hätte der Verwaltungsaufwand reduziert werden können und es hätte mehr Spielerdatensätze erspielt werden können. Die Qualität der Beobachterdatensätze ist im Vorfeld überschätzt worden, denn für eine qualitative Aussage über den Spielverlauf und das Spielverhalten der unterschiedlichen Mitspieler

benötigt man, insbesondere beim Spielen von Diplomacy, Wissen über die laufenden Verhandlungen. Durch das Fehlen dieser Informationen kam es zu der extremen Korrelation zwischen empfundener Spielstärke und Menschenähnlichkeit. Schöner wäre es hingegen gewesen mehr darüber herauszufinden, welche Aspekte der Verhandlungen als besonders menschlich wahrgenommen wurden. Der zweite Punkt ist, dass ein Turingtest in einer möglichst objektiven Umgebung stattfinden sollte. Durch die Vorkenntnisse der Beobachter, welche zudem auch die Entwickler der Bots waren, verzerrte das Bild bei der Bewertung. Bei der Durchführung eines solchen Test sollte man in Zukunft probieren möglichst vorurteilsfrei in die Gesamtsituation hineinzugehen und vor allem probieren seine Meinung für sich zu notieren und die anderen nicht zu beeinflussen. Durch öffentliche Diskussion beim Test kann es schnell passieren, dass die Meinung vieler teils stark beeinflusst wird.
Nun soll an dieser Stelle noch einmal kurz auf die Resultate des Test eingegangen werden. Untersucht man die Bewertung der Spielstärke durch die Spieler und Beobachter fällt auf, dass die Ergebnisse sich teilweise sehr stark von denen der statistischen Auswertung unterscheiden. Der FrySpi, welcher einer der spielstärksten Bots ist, wird von den Spielern teilweise als spielschwach und der Stragotiator im Vergleich zum Nice-Bot als gleichwertig beziehungsweise als überlegen eingestuft und dies obwohl die erspielten Statistikergebnisse deutlich andere Resultate vermuten lassen. Dies kann dadurch begründet sein, dass die kurze Spieldauer von wenigen Spieljahren nicht ausreicht, um die Spielstärke hinreichend zu analysieren und bewerten zu können.
Die Untersuchung der Ergebnisse der Menschenähnlichkeit ergibt, dass die Beobachter im Vergleich zu den Spielern das Verhalten der Bots als menschenähnlicher einstufen. Dies kann darauf zurückgeführt werden, dass die Beobachter die Verhandlungen der Spieler mit den Bots nicht mitverfolgen können und somit die Bewertung der Kommunikation nicht in die Bewertung mit einfließt. Zu häufiges Versenden von Friedens- oder ähnlichen Angeboten kann schnell unmenschlich wirken und sich somit negativ auf die Bewertung auswirken. Die Auswirkungen der Verhandlungen werden ebenfalls bei der Einschätzung deutlich, bei welcher Beobachter und Spieler festlegen sollten ob es sich bei einem Mitspieler um einen Menschen oder einen Bot handelt: Wurde in der Spielerbewertung lediglich der Stragotiator teilweise als Mensch eingeschätzt, so wurden von den Beobachtern auch FrySpi und Nice-Bot als menschlich empfunden, wenn auch in wesentlich geringerem Umfang.
Zusammenfassend kann man sagen, dass es den drei Bots gelungen ist, durch die Auswahl der Züge plausibles Verhalten darstellen zu können. Die Qualität und Quantität der Verhandlungen ist es letztendlich jedoch, welche die Menschenähnlichkeit in großem Maße beeinflußt und der Stragotiator, als Gewinner in dem Vergleich, konnte durch geschickte gezielte Kommunikation sowohl die Spieler als auch die Beobachter überzeugen.

7 Fazit

Die Projektgruppe 529 der Fakultät für Informatik der Technischen Universität Dortmund hat im ersten Projektsemester einen Bot für das Kartenspiel Poker entwickelt, der mit Hilfe der Methoden der *Computational Intelligence* ein menschenähnliches Verhalten zeigt. Im zweiten Semester wurden drei verschiedene Bots für das Brettspiel Diplomacy entwickelt. Beide Projekte verbindet, dass sie versuchen, durch den Gebrauch von CI-Methoden einen Computergegner zu erschaffen, der sich menschenähnlich verhält. Die Planung, Realisierung und die anschließenden Tests liefen nicht immer ohne Zwischenfälle, trotzdem hat die Projektgruppe beide Projekte zu einem erfolgreichen Abschluss gebracht.

In diesem Kapitel werden wir uns mit den Problemen beschäftigen, die in der Projektgruppe auftraten, und versuchen aufzuzeigen, wie diese hätten vermieden werden können. Besonderes Augenmerk soll dabei auf den Lösungsansätzen liegen.

7.1 Verbessungspotential und Erkenntnisse für andere Projekte

Nachdem im ersten Projekt einige Probleme der Projektgruppe auftraten, hat sich die Gruppe darum bemüht, aus den Fehlern zu lernen.

Die Planung des zweiten Projekts verlief besser. Ein Grund dafür war allerdings nicht nur die Motivation der Teilnehmer, sondern auch die Tatsache, dass nicht mehr alle parallel an der gleichen Software arbeiteten und sich gegenseitig behinderten. Die Kommunikationswege innerhalb der Untergruppen waren nun deutlich kürzer und es entstand in gewisser Weise eine Hierarchie, welche den Gruppensprechern der einzelnen Untergruppen mehr Verantwortung gegenüber dem Gesamtprojekt gab. Bedingt durch die Tatsache, dass die Kommunikation nur noch innerhalb der Untergruppen das Planen und Entwickeln beeinflusste und ausschließlich in den Gruppensitzungen die Vorhaben kritisiert oder bestätigt wurden, fanden weniger unwichtige Diskussionen statt, und die zu erledigende Arbeit konnten schneller fertiggestellt werden. Die meisten Teilnehmer zeigten mehr Eigeninitiative und übernahmen mehr Verantwortung. Trotz der offensichtlichen Vorteile dieser Aufgabenteilung brachte die Zersplitterung der Teilnehmer in feste Untergruppen auch Nachteile mit sich. Kompetenzen, die nicht alle Teilnehmer hatten, fehlten in machen Gruppen.

Die Bereitschaft, eigene Schwächen einzugestehen, also einzuräumen, dass man eine Aufgabe nicht alleine lösen kann, war bis zum Ende der Projektgruppe nicht bei jedem Teilnehmer vorhanden.

Das Ticket-System und die Versionsverwaltung wurde von manchen Teilnehmern deutlich stärker genutzt als von anderen. Vereinbarungen, die eigentlich für die gesamte Gruppe gelten sollten, wurden somit durch den Wunsch von Teilnehmern anderer Untergruppen außer Kraft gesetzt. Der im ersten Semester nach anfänglichen Schwierigkeiten gewählte Projektleiter war nun nach der Meinung einiger nicht mehr nötig, und auch die Betreuer hielten eine Prüfung der Ergebnisse und Leistung durch einen gewählten Teilnehmer nicht mehr für sinnvoll. In einer Abstimmung entschied sich die Projektgruppe dann gegen einen Projektleiter. Daraus resultierte, dass niemand nachvollziehen konnte, an welcher Aufgabe ein Teilnehmer arbeitete, es sei denn, man arbeitete direkt in der Untergruppe mit ihm zusammen. Der fehlende Wille eines Teilnehmers, die Versionsverwaltung zu nutzen, erhöhte diese Schwierigkeit.

Durch die fehlende Arbeitsleistung und offensichtlichen Falschaussagen bezüglich der Arbeitsergebnisse eines Teilnehmers wurde die Arbeitsmoral aller Teilnehmer immer

weiter geschwächt. Letztendlich wurden die Betreuer in einem Gespräch auf den Missstand aufmerksam gemacht. Bis zu diesem Zeitpunkt hatten alle Teilnehmer das offene Gespräch mit den Betreuern und den Umgang mit Maßnahmen, die zum Ausschluss Einzelner aus der Projektgruppe führen konnten, vermieden. Der Grund dafür lag darin, dass die Betreuer bis Mitte des zweiten Semesters den Teilnehmern nicht das Gefühl vermittelten, dass ihre Gesamtarbeitsleistung für den Erhalt des Schein ausreichend war. Und auch nach der offenen Aussprache in der Gruppe gab es immer noch Teilnehmer, die die offene Aussprache und den konstruktiven Umgang mit dem dargestellten Problem als Fehler ansahen.

In einer Gruppensitzung wurde über die Leistung eines Einzelnen zuerst ohne dessen Anwesenheit gesprochen. Alle Gruppenmitglieder waren sich einig, dass die Leistung des Einzelnen bei weitem nicht an den Gruppendurchschnitt heran reichte. Aus diesem Grund wurde eine Sonderaufgabe vergeben, die das Leistungsdefizit ausgleichen sollte.

Die Sonderaufgabe bestand aus zwei Teilen: Zum einen sollte ein Observer für die Contest-Software (vgl. 6.2 auf Seite 115) geschrieben werden, der die Aufgabe hatte, das Starten des Servers und die Anmeldungen der Bots beim Server zu überwachen. Diese Aufgabe wurde nicht gelöst und nach Wochen von einem anderen Teilnehmer übernommen und dann von diesem innerhalb von wenigen Tagen fertig gestellt. Der zweite Teil der Aufgabe bestand darin, eine Software zu entwickeln, die aus den Logdateien der Contest-Software die Ergebnisse der Contests filtert und für eine weitere Auswertung aufbereitet. Auch diese Aufgabe wurde nicht gelöst und später von einem anderen Teilnehmer erfolgreich bearbeitet.

Insgesamt betrachtet brachte die Projektgruppe interessante wissenschaftliche Erkenntnisse hervor, die auf international anerkannten Konferenzen öffentlich gemacht wurden. So gab es zu den Ergebnissen der Arbeiten aus dem ersten Semester einen Vortrag auf der PPSN X (siehe [PPS08]). Und zum Zeitpunkt der Erstellung dieses Berichts war geplant, dass ein Paper für die CIG2009 (siehe [CIG09]) mit den Ergebnissen des Turing-Tests der Diplomacy-Bots geschrieben und eingereicht werden sollte.

Trotz der offensichtlichen Erfolge bleibt zu sagen, was anders hätte gemacht werden können, und vor allem, welches Wissen und welche Kompetenzen die Teilnehmer aus der Projektgruppe mitnehmen sollten. Durch den angesprochenen Problemfall haben die Teilnehmer gelernt, dass es Probleme im Arbeitsumfeld mit anderen Mitarbeiter geben kann, die man möglichst schnell lösen sollte. Nicht immer ist einem Mitarbeiter sein Fehlverhalten bekannt und so kann ein kurzes Gespräch schnell Abhilfe schaffen. Auch sollte klar geworden sein, dass es Menschen gibt, bei denen ein Gespräch aussichtslos ist, da diese ihr Fehlverhalten entweder nicht einsehen oder es ihnen bekannt ist, sie es aber nicht ändern wollen.

Das offene Gespräch mit der Leitung einer Arbeitsgruppe darf nicht gescheut werden. Auch müssen Teilnehmern oder Arbeitnehmern die Fehler bekannt sein, die zum Ausschluss aus der Gruppe oder dem Arbeitsverhältnis führen.

Die Bereitschaft, neue Fertigkeiten zu erlernen, Systeme zu nutzen, die die Arbeit begleiten und erleichtern, das Einhalten von Konventionen sowie das Reden über eigenen Probleme sind gerade im Umfeld der Informatik Schlüsselqualifikationen.

Wenn all diese Punkte Beachtung finden, dann steht einem baldigen und erfolgreichen Start ins Berufsleben nichts mehr im Weg.

A Anhang

A.1 Diplomacy Regeln

1. Alle Einheiten haben die gleiche Stärke.
2. In einem Gebiet kann sich immer nur eine Einheit aufhalten.
3. Wenn zwei oder mehr Einheiten gleicher Stärke versuchen, in dasselbe Gebiet zu ziehen, bleiben alle diese Einheiten in ihren Ausgangsgebieten.
4. Eine Pattsituation führt nicht dazu, dass sich eine Einheit zurückziehen muss, die sich zu Beginn des Spielzuges in dem Gebiet aufgehalten hat, in dem es zur Pattsituation gekommen ist.
5. Eine Einheit, die sich nicht bewegt oder nicht bewegen kann, kann dazu führen, dass sich mehrere Einheiten nicht bewegen.
6. Zwei Einheiten können ohne die Verwendung eines Geleitzuges nicht miteinander Plätze tauschen.
7. Drei oder mehr Einheiten können innerhalb eines Zuges ihre Plätze tauschen. Dabei dürfen aber keine zwei Einheiten direkt miteinander Plätze tauschen.
8. Wenn man einen Unterstützungsbefehl für eine Einheit erteilt, die sich nicht bewegt, reicht es, wenn man beim Unterstützungsbefehl das Gebiet angibt, welches unterstützt werden soll.
9. Wenn sich eine Einheit zu bewegen versucht, sind nur jene Unterstützungsbefehle gültig, bei denen die Bewegung der unterstützten Einheit korrekt angeführt wurde.
10. Eine Einheit, die zum Rückzug gezwungen wird, kann zu einer Pattsituation führen: Allerdings nicht in dem Gebiet, von dem aus sie zum Rückzug gezwungen wurde.
11. Eine Einheit, die zum Rückzug gezwungen wird, hat keine Auswirkung auf die Einheiten, die sie zum Rückzug gezwungen haben. Unterstützung durch andere Einheiten spielt keine Rolle.
12. Ein Land kann niemals seine eigenen Einheiten zum Rückzug zwingen. Es kann auch keinen Angriff unterstützen, der zu einem Rückzug seiner eigenen Einheiten führen würde. Dies gilt auch dann, wenn der entsprechende Befehl versehentlich oder unabsichtlich erteilt wurde.
13. Eine Unterstützung wird unterbunden, wenn die unterstützende Einheit von einem Gebiet aus angegriffen wird, bei dem es sich nicht um das Gebiet handelt, das mit Hilfe der unterstützenden Einheit angegriffen wird.
14. Eine Unterstützung wird unterbunden, wenn die unterstützende Einheit zum Rückzug gezwungen wird.
15. Eine Einheit, die zu einem Rückzug gezwungen wird, kann in diesem Spielzug dennoch eine Unterstützung unterbinden. Dies gilt jedoch nur, wenn der Angriff, der die Einheit zum Rückzug zwingt, nicht aus dem Gebiet kommt, in das die Einheit einen Angriff unterstützt.
16. Ein Angriff durch eigene Einheiten unterbindet keine Unterstützung.
17. Wenn eine der Flotten, die eine Armee geleitet, zum Rückzug gezwungen wird, scheitert der ganze Geleitzug.

18. Wenn eine Armee, die durch eine Flotte geleitet wird, sich in ein Gebiet bewegen würde, in dem es zu einer Pattsituation kommt, wird der Bewegungsbefehl der Armee nicht ausgeführt, das heißt, die Armee bewegt sich nicht, sie verbleibt in ihrem Ausgangsgebiet.
19. Einheiten können miteinander Plätze tauschen, wenn sich eine oder beide Einheiten mit Hilfe eines Geleitzuges bewegen. Regel 19 stellt eine Ausnahme zur Regel 6 dar.
20. Wenn mehrere Möglichkeiten für den Transport einer Armee per Geleitzug zur Verfügung stehen, führt diese Armee ihre Bewegung aus, wenn nicht alle Geleitzüge unterbunden werden.
21. Eine Armee, die sich mittels Geleitzug bewegt, kann nicht die Unterstützung für einen Angriff unterbinden, der eine der Flotten zum Ziel hat, die für den Geleitzug notwendig sind. Im Zweifelsfall setzt diese Regel die Regel 13 außer Kraft.
22. Eine Armee, der zumindest eine Möglichkeit zur Verfügung steht, einen Geleitzug zu benutzen, unterbindet die Unterstützung für einen Angriff, der auf eine Flotte entlang einer alternativen Geleitzugroute stattfindet. Im Zweifelsfall setzt diese Regel die Regel 21 außer Kraft.

(vgl. `http://www.hasbro.de/manuals/41307.pdf`)

A.2 Diplomacy Abkürzungen

A.2.1 Großmächte

AUS	Österreich-Ungarn	ENG	England	FRA	Frankreich
GER	Deutschland	ITA	Italien	RUS	Russland
TUR	Türkei				

A.2.2 Provinzen

ADR	Adriatic Sea	AEG	Aegean Sea
ALB	Albania	ANK	Ankara
APU	Apulia	ARM	Armenia
BAL	Baltic Sea	BAR	Barents Sea
BEL	Belgium	BER	Berlin
BLA	Black Sea	BOH	Bohemia
BRE	Brest	BUD	Budapest
BUL	Bulgaria	BUR	Burgundy
CLY	Clyde	CON	Constantinople
DEN	Denmark	EAS	Eastern Mediterranean
EDI	Edinburgh	ENG	English Channel
FIN	Finland	GAL	Galicia
GAS	Gascony	BOT	Gulf of Bothnia
LYO	Gulf of Lyon	HEL	Helgoland Bight
HOL	Holland	ION	Ionian Sea
IRI	Irish Sea	KIE	Kiel
LVP	Liverpool	LVN	Livonia
LON	London	MAR	Marseilles
MAO	Mid-Atlantic Ocean	MOS	Moscow
MUN	Munich	NAP	Naples
NAF	North Africa	NAO	North Atlantic Ocean
NTH	North Sea	NWY	Norway
NWG	Norwegian Sea	PAR	Paris
PIC	Picardy	PIE	Piedmont
POR	Portugal	PRU	Prussia
ROM	Rome	RUH	Ruhr
RUM	Rumania	SER	Serbia
SEV	Sevastopol	SIL	Silesia
SKA	Skagerrak	SMY	Smyrna
SPA	Spain	STP	St. Petersburg
SWE	Sweden	SYR	Syria
TRI	Trieste	TUN	Tunis
TUS	Tuscany	TYR	Tyrolia
TYS	Tyrrhenian Sea	UKR	Ukraine
VEN	Venice	VIE	Vienna
WAL	Wales	WAR	Warsaw
WES	Western Mediterranean	YOR	Yorkshire

B Seminararbeiten

B.1 Avatare und Kommunikation mit Computerspielern

Avatare besitzen eine große Ausdruckskraft. Durch die geschickte Auswahl von ihnen können Spieler mit einander kommunizieren. Darüber hinaus geben ihre Besitzer - bewusst oder unbewusst - Informationen, zum Beispiel was sie mögen oder welche Ängste sie haben, über sich Preis. In modernen Spielen können Spieler sogar über die Gesichtsmimik ihrer Avatare Emotionen ausdrücken.

Neben Avataren gibt es in Computerspielen eine Menge weiterer Kommunikationsmöglichkeiten. Am weitesten verbreitet sind Textchats. Sie dienen der Übermittlung von natürlicher Sprache. Bei Internetspielen unterhalten sich Spieler besonders über Geschehnisse im Spiel, Aktionen anderer Mitspieler und das Spiel an sich. Bei Diskussionen über Verhalten treten zwei Extreme auf. Teilweise werden besonders schöne oder gewinnbringende Taten gelobt. Häufiger jedoch werden Mitspieler beschimpft. Diese Beschimpfungen sind allerdings meistens nicht als verbaler Angriff sonder vielmehr als Zeichen des Unmuts aufzunehmen.

B.2 Emotionen in der klassischen KI

Einstweilen haben auch die Vertreter der Künstlichen Intelligenz erkannt, wie wichtig Emotionen für einen intelligenten Computer sind. In der Seminararbeit, siehe [AGG+09], werden Emotionen in der klassischen KI als regelbasierte Zustände definiert. Mit Hilfe dieser Regeln kann eine Formel zur Herleitung der Emotion bestimmt werden. Diese Regelsysteme sind Situationsgebunden und führen zum Schluss, dass Emotionen in der klassischen KI nicht erlebt werden, sondern in typischen Situationen ein typisches Verhalten zeigen.

B.3 Evolutionäre Algoithmen

Evolutionäre Algorithmen sind Zufallsheuristiken, welche die biologische Evolution als Optimierungsverfahren zum Vorbild haben.
Die Seminararbeit, siehe [AGG+09] Anhang C, behandelt den formalen Hintergrund und stellt den strukturellen Aufbau von Evolutionären Algorithmen und ihrer einzelnen Module vor. Im vorletzten Kapitel wird ein Bezug zur Projektarbeit erstellt indem eine mögliche Anwendung in einem Strategiespiel vorgestellt wird.
Zum Abschluss werden noch allgemeine Anmerkungen zu Evolutionären Algorithmen gemacht - z.B. wann sich ihr Einsatz lohnt - sowie ihre Vor- und Nachteile aufgelistet.

B.4 Fuzzy-Systeme

Diese Ausarbeitung dient der Einführung in die Grundlagen der Fuzzylogik und der Fuzzy-Systeme. Zusammen mit neuronalen Netzen werden den Fuzzy-Systemen, aufgrund zahlreicher erfolgreicher Verwirklichungen in der Praxis, auch zukünftig hohes Potenzial für eine Erweiterung auf neue Anwendungsbereiche zugeordnet. Insbesondere der Bereich der Neuro-Fuzzy-Systeme, der versucht die Vorteile der beiden Verfahren

zu vereinigen, erlangt in der Wissenschaft immer mehr Aufmerksamkeit und Popularität. Die Verwendung von Fuzzy-Systemen bewährt sich besonders, falls das System aufgrund seiner Komplexität nicht exakt beschrieben werden kann.

B.5 Künstliche Intelligenz in den Spielen Poker, Diplomacy und Junta

Die Arbeit „KI in den drei Spielen Poker, Diplomacy und Junta“ beschäftigt sich mit dem aktuellen Stand der Forschung zu den drei Spielen. Es wird untersucht welche Ansätze für den Entwurf der Computerspieler zur Zeit benutzt werden und die jeweiligen spielspezifischen Besonderheiten wie etwa Nichtdeterminismus in Form von zufälliger Kartenverteilung untersucht. Abschliessend werden die Spiele mit ihren Vor- und Nachteilen gegenübergestellt.

B.6 Ludologie

Die Ausarbeitung Ludologie (vgl. [AGG+09]) beschäftigt sich mit einem wissenschaftlichen Forschungszweig, der sich mit kulturellen, strukturellen, kommunikativen und technischen Aspekten von Spielen, insbesondere digitalen Videospielen, auseinandersetzt.

B.7 Maschinelles Lernen/Regelbasierte Steuerung

Es wird vorerst definiert, was hier unter Lernen im Allgemeinen und „maschinellem Lernen“ im Speziellen zu verstehen ist.

Maschinelle Lernverfahren können grob in überwachte und unüberwachte Lernverfahren unterteilt werden. Es werden für beide Klassen Vertreter besprochen, wobei ein Schwerpunkt auf den überwachten Lernverfahren liegt.

Zum Abschluss werden Ideen skizziert, wie man mit Hilfe der vorgestellten Verfahren Nicht-Spieler-Charaktere (NSC) entwickeln könnte.

B.8 Künstliche Neuronale Netze

Vergleicht man die Arbeitsweise eines Computers mit der des menschlichen Gehirns, so lassen sich, ungeachtet der Unterschiede in der generellen Leistungsfähigkeit, deutliche Unterschiede feststellen. So ist das menschliche Gehirn beispielsweise tolerant sowohl gegenüber dem Ausfall einzelner „Rechenelemente“ als auch gegenüber unvollständigen Informationen. Beides sind Grundlagen für eine hohe Anpassungsfähigkeit. Künstliche Neuronale Netze versuchen nun, angelehnt an den Aufbau des menschlichen Gehirns, solche Fähigkeiten zumindest in grundlegender Form auf Computer zu übertragen. Hierbei gibt es eine Vielzahl verschiedener Modelle, die sich für viele unterschiedliche Zwecke einsetzen lassen.

B.9 Soziologie/Psychologie: Modelle für Emotionen/Verhalten von Spielern

Emotionen sind eine grundlegende Notwendigkeit für Spiele, da Spiele Reize auslösen, um Spiel-Verhalten und -Motivation zu beeinflussen. Ebenso sind Emotionen ein essentieller Bestandteil des menschlichen Charakters, sowie zwischenmenschliche Beziehungen für Interaktionen miteinander.

In der Seminararbeit „Psychologie/Soziologie: Modelle für Emotionen/Verhalten von Spielern“ (vgl. [AGG+09]) werden die Einwirkungen dieser Faktoren auf das Denken und Handeln, sowie in diesem Kontext die Konnektivität zwischen Emotion und Verhalten, näher beleuchtet.

C PG-Ordnung

C.1 PG-Vereinbarungen

In der PG-Ordnung sind verbindliche Pflichten für alle PG-Teilnehmer enthalten.

C.1.1 Abstimmungsmodus

- Satzungsänderungen:
  Können nur beschlossen werden wenn mindestens zwei Drittel der Teilnehmer für die Änderung stimmen (qualifizierte Mehrheit).
- Alle anderen Änderungen:
  Änderungen werden per Mehrheitsentscheid (einfache Mehrheit) getroffen.
- Eine Abstimmung findet nur dann statt, wenn bezüglich eines Themas Kontroversen bestehen.
- Alle durchgeführten Abstimmungen sind offen.

C.1.2 Organisatorisches

- Anwesenheit:
  Die Teilnahme an den PG-Sitzungen ist Pflicht. Im Protokoll wird vermerkt, wenn ein PG-Teilnehmer oder ein Betreuer verspätet zur PG-Sitzung eintrifft.
- Beginn der PG-Sitzungen:
  PG-Sitzungen starten grundsätzlich pünktlich, auch wenn PG-Teilnehmer oder PG-Betreuer fehlen.

C.2 Etikette

In der Etikette sind alle wünschenswerten Aufgaben enthalten.

- Sowohl E-Mails als auch das Wiki müssen einmal am Werktag überprüft werden.
- Der Protokollant soll das Protokoll einen Tag nach der Sitzung bereitstellen.
- Der Moderator soll die Tagesordnung einen Tag vor der Sitzung bereitstellen.
- Ist ein PG-Teilnehmer verhindert, so muss dies den Betreuern und den restlichen PG-Teilnehmern per E-Mail und mit Angabe eines triftigen Grunds mitgeteilt werden.
- E-Mails sollen mit dem Ausdruck „[PG529]“ beginnen.
- Eine förmliche Anrede ist nicht nötig.
- Eine Änderung der Etikette bedarf einer einfachen Mehrheit.

Abbildungsverzeichnis

1 McCullon-Pitts Neuron . . . 4
2 Perceptron . . . 5
3 Ein in Schichten unterteilt Neuronales Netz . . . 5
4 Fuzzy Sets: Zuordnung eines Alters in Fuzzy-Sets . . . 8
5 Influence Maps: die Ausgangssituation des Beispiels . . . 9
6 Influence Maps: die Rasterung der Spielkarte . . . 9
7 Influence Maps: die Stärke der Spieler . . . 10
8 Die Projektplanung zu PokerTH. Die Grafik zeigt unseren ersten Entwurf eines Zeitplans des Projekts . . . 13
9 PokerTH PG529 Edition (mit Emoticons und Textbausteinen) . . . 15
10 Die Ost-West Stalemate Line . . . 32
11 XDO: Beispielszenario . . . 35
12 Klassendiagramm der Hauptklassen des *Stragotiators* . . . 41
13 Bereichsunterteilung des Vertrauens . . . 57
14 Strategie Kern: Aktivitätsdiagramm Bewegungs Phase . . . 61
15 TrustA . . . 69
16 TrustB . . . 70
17 TrustC . . . 70
18 TrustD . . . 71
19 TrustE . . . 71
20 Von AUS gesendete Zugvorschläge . . . 73
21 Von GER gesendete Zugvorschläge . . . 73
22 Von AUS ausgeführte Supports . . . 74
23 Von GER ausgeführte Supports . . . 74
24 normaler BFS-Baum mit Knotengewichten . . . 83
25 BFS-Baum mit kummulierten Knotengewichten für alle nicht-Wurzelknoten 83
26 Das Analyzer-Fenster kurz nach dem Start des Spiels . . . 91
27 Das Analyzer-Fenster mit möglichen Angriffszügen und Unterstützungswerten der Regionen . . . 92
28 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 0 . 97
29 Statistikauswertung: Verhandlungen in Konstellation 0 . . . 98
30 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 1 . 98
31 Statistikauswertung: Verhandlungen in Konstellation 1 . . . 99
32 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 10 . 100
33 Statistikauswertung: Verhandlungen in Konstellation 10 . . . 101
34 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 2 . 101
35 Statistikauswertung: Verhandlungen in Konstellation 2 . . . 102
36 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 3 . 102
37 Statistikauswertung: Verhandlungen in Konstellation 3 . . . 103
38 Statistikauswertung: Anzahl der gewonnenen Spiele in Konstellation 8 . 103
39 Statistikauswertung: Verhandlungen in Konstellation 8 . . . 104
40 Statistikauswertung: Anzahl der gewonnenen Spiele . . . 105
41 Statistikauswertung: Anzahl der durchschnittlichen Peace-Nachrichten . 106
42 Statistikauswertung: Anzahl der durchschnittlichen Aly-Vorschläge . . . 106
43 Statistikauswertung: Anzahl der durchschnittlichen akzeptierten und durchgeführten Zugvorschläge . . . 107
44 Statistikauswertung: Entwicklung Supplycenter in 20 Jahren . . . 108
45 Statistikauswertung: Entwicklung Supplycenter in 50 Jahren . . . 109
46 Statistikauswertung: Angriffe und Erfolge des CrySpi . . . 110
47 Statistikauswertung: Angriffe und Erfolge des FrySpi . . . 111
48 Statistikauswertung: Angriffe und Erfolge des Diplominator . . . 112

49 Statistikauswertung: Angriffe und Erfolge des Nice 113
50 Statistikauswertung: Angriffe und Erfolge des Stragotiators mit EA . . . 114
51 Statistikauswertung: Angriffe und Erfolge des Stragotiators ohne EA . . 115
52 Turingtest: Spielerbewertung der Spielstärke und Menschenähnlichkeit . 117
53 Turingtest: Beobachterbewertung der Spielstärke und Menschenähnlichkeit 118
54 Turingtest: Einschätzung Mensch oder Bot 119

Literaturverzeichnis

[Aar97] E. J. Aarseth. *Cybertext : Perspectives on Ergodic Literature*. Johns Hopkins University Press, Baltimore, 1997.

[Aga92] Stephen Agar. Long-term planning. `http://www.diplomacy-archive.com/resources/strategy/articles/long-term.htm`, 8 1992. (Verfügbar 30.01.09).

[AGG+09] N. Ackermann, A. Gholaman, M. Gorzala, M. Grochowski, T. Harweg, M. Kemmerling, D. Spierling, S. Uellenbeck und W. Walz. Spielcharaktere - Modellierung menschenähnlicher Gegenspieler in Strategiespielen mit Techniken der CI. Zwischenbericht zur PG 529, TU Dortmund, Dortmund, Februar 2009.

[AKA91] D. W. Aha, D. F. Kibler und M. K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991. `http://dblp.uni-trier.de`.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen und A. I. Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, page 307 ff. AAAI/MIT Press, 1996. `http://dblp.uni-trier.de`.

[Are04] M. Aretz. *Emotionen in der Künstlichen Intelligenz*. Diplomarbeit, FH Gießen-Friedberg, März 2004.

[Bae96] T. Baeck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, NY, USA, 1996.

[Bak87] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, New Jersey, USA, 1987. Lawrence Erlbaum Associates, Inc.

[Bar03] R. A. Bartle. *Design of Virtual Worlds*. New Riders, Indianapolis, 2003.

[BBD+03] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg und D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker, 2003. `http://www.cs.ualberta.ca/~darse/Papers/IJCAI03.pdf`.

[BDSS02] D. Billings, A. Davidson, J. Schaeffer und D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1-2):201–240, 2002. `http://www.cs.ualberta.ca/~jonathan/Grad/Papers/aij02.pdf`.

[Bet80] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, University of Michigan, Ann Arbor, 1980.

[BKKN95] C. Borgelt, F. Klawonn, R. Kruse und D. Nauck. *Neuro Fuzzy Systeme - Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen*. Vieweg, Wiesbaden, 1995.

[Bot95] H.-H. Bothe. *Fuzzy Logic - Einführung in die Theorie mit Anwendungen*. Springer, Berlin, Heidelberg, New York, 1995.

[Bra91] R. Brause. *Neuronale Netze - Eine Einführung in die Neuroinformatik*. B.G. Teubner, 1991.

[Bre08] C. Breazeal. Kismet. `http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html`, 2008. (Verfügbar 19.03.08).

[BS02] H. G. Beyer und H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[Bul08] F. Bulka. Dialog-KI in IT-basiertenSpielen Kommunikation zwischen Mensch und NPC. `http://bis.informatik.uni-leipzig.de/de/Lehre/0607/WS/GAMES/index`, 2008. (Verfügbar 19.03.08).

[Bur66] E. Burger. *Einführung in die Theorie der Spiele.* Walter de Gruyter & Co, Berlin, 1966.

[Cal74] Allan Calhamer. The invention of diplomacy. `http://www.diplomacy-archive.com/resources/calhamer/invention.htm`, 1 1974. (Verfügbar 31.01.09).

[CIG09] CIG2009 IEEE Symposium on Computational Intelligence and Games. `http://www.ieee-cig.org/`, 2009. (Verfügbar 17.02.09).

[Cle98] J. Cleve. Wissensextraktion / data mining, 1998. `http://www.wi.hs-wismar.de/cleve/vorl/dm2160207807/skript/node1.html`.

[Csi05] M. Csikszentmihalyi. *Das flow - Erlebnis. Jenseits von Angst und Langeweile: im Tun aufgehen.* Klett-Cotta, Stuttgart, 2005.

[Dav91] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, NY, USA, 1991.

[DJ75] K.-D. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, 1975.

[DMW04] R. De Maria und J. L. Wilson. *High Score!: The Illustrated History of Electronic Games.* McGraw-Hill Osborne, New York, 2004.

[Dre61] M. Dresher. *Strategische Spiele - Theorie und Praxis.* Verlag Industrieller Organisation, Zürich, 1961.

[EF78] P. Ekman und W. V. Friesen. Facial action coding system: A technique for the measurement of facial movement. *Consulting Psychologists Press*, 1978.

[Ekm05] P. Ekman, editor. *Gefühle lesen*, Heidelberg, 2005. Elsevier, Spektrum Akademischer Verlag.

[FJS06] J. Feindt, C. Janßen und S. Sölbrandt. Gesprächskompetenz im I-can-EIB-System. In Claus Möbus, editor, *Web-Kommunikation mit OpenSource*, chapter 13, pages 147–212. Springer, Heidelberg, 2006.

[FK93] J. C. Fodor und T. Keresztfalvi. *Non-conventional conjunctions and implications in fuzzy logic.* Springer-Verlag, 1993.

[Flu06] M. Fludernik. *Einführung in die Erzähltheorie.* WBG Wissenschaftliche Buchgesellschaft, Darmstadt, 2006.

[Fog95] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, NY, USA, 1995.

[FP63] R. Fletcher und M. J. D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

[GHJV04] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Entwurfsmuster.* Addison-Wesley, 2004.

[GR00] G. Goerz und C.-R. Rollinger. *Handbuch der künstlichen Intelligenz*. Oldenburg, 2000.

[Gra95] A. Grauel. *Fuzzy-Logik - Einführung in die Grundlagen mit Anwendungen*. Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich, 1995.

[Gra04] W. Grass. *Technische Anwendungen von Fuzzy-Systemen*. Universität Passau - Lehrstuhl für Rechnerstrukturen, Passau, 2004.

[GRS03] G. Goerz, C.-R. Rollinger und J. Schneeberger, editors. *Handbuch der Künstlichen Intelligenz*. Oldenbourg, 2003.

[Han95] Manus Hand. The diplomatic pouch: The library of diplomacy openings. `http://www.diplom.org/Online/Openings/textversion/`, 1995. (Verfügbar 04.02.09).

[Han06] N Hansen. Compilation of results on the 2005 cec - benchmark function set, 2006.

[Han07] N. Hansen. The CMA Evolution Strategy: A Tutorial. 2007.

[Han09] Manus Hand. The diplomatic pouch. `http://www.diplom.org/index.py`, 2009. (Verfügbar 30.01.09).

[Heb49] D. Hebb. *Organization of Behavior*. Wiley, New York, 1949.

[Hof93] N. Hoffmann. *Kleines Handbuch Neuronale Netze*. vieweg, 1993.

[Hol71] R. B. Hollistien. *Artificial Genetic Adaption in Computer Control Systems*. PhD thesis, University of Michigan, Ann Arbor, 1971.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975. (Verfügbar 06.04.08).

[Hop82] J. J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *PNAS USA*, 79:2554–2558, 1982.

[Hor79] R. Horst. *Nichtlineare Optimierung*. Carl Hanser, München, 1979.

[Hui54] J. Huizinga. *Geschichte und Kultur*. Alfred Kröner Verlag, Stuttgart, 1954.

[Hui94] J. Huizinga. *Homo ludens. Vom Ursprung der Kultur im Spiel (1939)*. Rowohlt Verlag, Hamburg, 1994.

[Jan05] T. Jansen. Evolutionäre Algorithmen. Vorlsungsskript WS 04/05 `http://ls2-www.cs.uni-dortmund.de/lehre/winter200405/evo-alg/skript-evoAlg.pdf`, 2005.

[Jue53] F. G. Juenger. *Die Spiele*. Klostermann Verlag, Frankfurt am Main, 1953.

[KDP83] S. Kirkpatrick, Gelatt C. D. und Vecchi M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Ken01] S. L. Kent. *The Ultimate History of Video Games: The Story behind the Craze that Touched Our Lives and Changed the World*. Three Rivers Press, New York, 2001.

[KMH+04] S. Kern, S. D. Mueller, N. Hansen, D. Bueche, J. Ocenasek und P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[Koh43] T. Kohonen. A logical calculus of the ideas immanent in nervous activity. *Ball Math Biophys*, 5:115–133, 1943.

[Kra95] Sarit Kraus. Desgning and Building a Negotiating Automated Agent. `http://www.cs.biu.ac.il/~sarit/Articles/13.ps`, 1995. (Verfügbar 20.02.09).

[KW08] A. Koch und K. Woog. Emotionstheorie von Ortony, Clore & Collins und affektives Priming. `http://www.allpsych.uni-giessen.de/vf/WS-2006-seminar-emotion/PDF/11-woog-koch-ortony-priming.pdf`, 2008. (Verfügbar 19.03.08).

[KWDD98] M. Kirsten, S. Wrobel, F.-W. Dahmen und H.-C. Damen. Einsatz von Data-Mining-Techniken zur Analyse Ökologischer Standort- und Pflanzendaten. *Künstliche Intelligenz*, 12(2):39–42, 1998.

[Lam01] J. Lampel. Joebot - Einsatz von Neuronalen Netzen in einem Bot für Counterstrike. `http://johannes.lampel.net/bll137pub.pdf`, 2001.

[Lec06] M. Lechner. Automatische Textgenerierung in Computerspielen. `http://bis.informatik.uni-leipzig.de/de/Lehre/0607/WS/GAMES/index/\files?get=gsc_Marcus_Lechner.pdf`, 2006. (Verfügbar 20.02.08).

[LeD96] J. LeDoux, editor. *The Emotional Brain: The Mysterious Underpinnings of Emotional Life*, 274, New York, 1996. Simon & Schuster.

[Lis02] K. Lischka. *Spielplatz Computer. Kultur, Geschichte und Ästhetik des Computerspiels.* Heise Verlag, Hannover, 2002.

[Loe95a] Danny Loeb. Challenges in multi-player gaming by computer. `http://devel.diplom.org/DipPouch/Zine/S1995M/Loeb/Project.html`, 1995. (Verfügbar 04.02.09).

[Loe95b] Danny Loeb. The combinatorics of adjustments. `http://www.diplom.org/Zine/W1995A/Loeb/Removes.html`, 1995. (Verfügbar 15.02.09).

[Loe95c] Danny Loeb. The combinatorics of retreats. `http://www.diplom.org/Zine/F1995R/Loeb/ret.html`, 1995. (Verfügbar 15.02.09).

[Loe95d] Danny Loeb. Computer interpretation of diplomacy openings. `http://www.diplom.org/Zine/S1995R/Loeb/Bourse.html`, 1995. (Verfügbar 15.02.09).

[Loe95e] Danny Loeb. The observation module. `http://devel.diplom.org/DipPouch/Zine/S1995R/Loeb/Observe.html`, 1995. (Verfügbar 15.02.09).

[Loe95f] Danny Loeb. Opening by computer: Spring 1901 through the eyes of a computer player. `http://www.diplom.org/Zine/S1995M/Loeb/DPP_Open.html`, 1995. (Verfügbar 04.02.09).

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th berkeley Symp. on Mathematical Statistics and Probability. 1967.

[Mai99] K. Mainzer, editor. *Computernetze und virtuelle Realität: Leben in der Wissensgesellschaft.* Springer, 1999.

[Man03] T. Manninen. Interaction Forms and Communicative Actions in Multiplayer Games. *the international journal of computer game research*, 3(1), 2003. `http://www.gamestudies.org/0301/manninen/`, (Verfügbar 20.02.08).

[MBC+06] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley und C. H. Yong. *Computational Intelligence in Games Computational Intelligence: Principles and Practice*. IEEE Computational Intelligence Society, 2006.

[MdWS91] B. Manderick, M de Weger und P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150, San Mateo, 1991. Morgan Kaufmann.

[Mic86] R. Michalski. Understanding the Nature of Learning. In T. M. Mitchell, editor, *Machine Learning – An Artificial Approach*. Morgan Kaufmann, Los Alto, CA, 1986.

[Mic92] Z. Michalewicz. *Genetic Algorithms + Data Structures =Evolution Programs*. Springer, Berlin, 1992.

[Mil98a] Tim Miller. White partial press. `http://devel.diplom.org/DipPouch/Zine/S1998R/Miller/Press.html`, 1998. (Verfügbar 04.02.09).

[Mil98b] Tim Miller. White partial press part 2. `http://devel.diplom.org/DipPouch/Zine/F1998M/Miller/press.html`, 1998. (Verfügbar 04.02.09).

[MK05] T. Manninen und T. Kujanpää. The Hunt for Collaborative War Gaming - CASE: Battlefield 1942. *the international journal of computer game research*, 5(1), 2005. `http://www.gamestudies.org/0501/manninen_kujanpaa/`, (Verfügbar 20.02.08).

[MM92] R. Maennde und B. Manderick. Parallel Problem Solving from Nature. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, North Holland, 1992.

[Mon01] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, New York, 5th edition, 2001.

[Mou95] Vincent Mous. The diplomacy survival guide. `http://www.diplom.org/Zine/S1995M/Mous/Survival.html`, 1995. (Verfügbar 04.02.09).

[Mue92] H. Muehlenbein. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *PPSN*, pages 15–26, 1992.

[Mue01] A. Muenkel, editor. *Computer.Gehirn; Was kann der Mensch? Was können die Computer?*, Paderborn, 2001. Schöningh.

[Mur98] S. K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Min. Knowl. Discov.*, 2(4):345–389, 1998. `http://dblp.uni-trier.de`.

[New08] John Newbury. Diplomacy blabbot. `http://uk.geocities.com/johnnewbury1/diplomacy/blabbot/index.htm`, 2008. (Verfügbar 16.02.09).

[Nis97] V. Nissen. *Einführung in Evolutionäre Algorithmen*. Vieweg, Wiesbaden, 1997.

[Oli05] F. Oliehoek. Game theory and ai: a unified approach to poker games, 2005. `staff.science.uva.nl/~faolieho/docs/Oliehoek05MScThesis.pdf`.

[PMAPA06] M. Ponsen, H. Munoz-Avila, Spronk P. und D. W. Aha. Automatically Generating Game Tactics with Evolutionary Learning. *AI Maga-*

zine, 27(3):75–84, 2006. http://www.cs.unimaas.nl/p.spronck/Pubs/ponsen_aimag.pdf.

[PMASA05] M. J. V. Ponsen, H. Muñoz-Avila, P. Spronck und D. W. Aha. Automatically Acquiring Adaptive Real-Time Strategy Game Opponents Using Evolutionary Learning. In *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 1535–1540, Menlo Park, California, 2005. AAAI Press. http://www.cs.unimaas.nl/p.spronck/Pubs/IAAI052PonsenM.pdf.

[Pon04] M. Ponsen. *Improving Adaptive Game AI with Evolutionary Learning.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 2004. http://www.kbs.twi.tudelft.nl/docs/MSc/2004/Ponsen_Marc/thesis.pdf.

[PPS08] PPSN 2008, 10th International Conference on Parallel Problem Solving From Nature. http://ls11-www.cs.uni-dortmund.de/ppsn/ppsn10/, 2008. (Verfügbar 17.02.09).

[PS04] M. Ponsen und P. Spronck. Improving Adaptive Game AI with Evolutionary Learning. In Q. Mehdi, N. Gough, S. Natkin und D. Al-Dabass, editors, *Computer Games: Artificial Intelligence, Design and Education*, pages 389–396, Wolverhampton, 2004. University of Wolverhampton. http://www.cs.unimaas.nl/p.spronck/Pubs/PonsenCGAIDE.pdf.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fommann-Holzboog, Stuttgart, 1973.

[Rei08] D. Reichardt. Emotional IT Computer mit Gefühl. http://www.ba-stuttgart.de/fileadmin/ba/Dokumente/Akademietag_2004/Emotional_IT-Computer_mit_Gef_hl.pdf, 2008. (Verfügbar 19.03.08).

[RHW86] D. E. Rumelhart, G. E. Hinton und R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323:533–536, 1986.

[RMS91] H. Ritter, T. Martinetz und K. Schulten. *Neuronale Netze - Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke.* Addison-Wesley, 1991.

[RN04] S. Russell und P. Norvig, editors. *Künstliche Intelligenz: Ein moderner Ansatz.* Pearson Studium, 2004.

[RSZ79] B. Rauhut, N. Schmitz und E.-W. Zachow. *Spieletheorie - Eine Einführung in die mathematische Theorie strategischer Spiele.* B.G. Teubner, Stuttgart, 1979.

[Rue08] G. Ruebenstrunk. Künstliche Gefühle. http://www.ruebenstrunk.de/emeocomp/2.htm, 2008. (Verfügbar 19.03.08).

[Sch68] H.-P. Schwefel. Experimentelle Optimierung einer Zweiphasendüse Teil I. Bericht Nr. 35 zum Projekt MHD–Staustrahlrohr 11.034/68, AEG Forschungsinstitut, Berlin, Oktober 1968.

[Sch77] H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research.* Birkhäuser, Basel, 1977.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking.* John Wiley & Sons, 1995.

[Sim67] H. A. Simon. Motivational and emotional controls of cognition. *Psychological Review*, 74:29–39, 1967.

[Sim83] H. A. Simon. Why Should Machines Learn? In T. M. Mitchell, editor, *Machine Learning – An Artificial Approach*, volume 1, chapter 3, pages 25–39. Morgan Kaufmann, Palo Alto, CA, 1983.

[SR95] H.-P. Schwefel und G. Rudolph. Contemporary evolution strategies. In *European Conference on Artificial Life*, pages 893–907, 1995.

[SS97] B. Sutton-Smith. *The Ambiguity of Play.* Harvard University Press, London, 1997.

[Sul01] J. Suler. The Psychology of Avatars and Graphical Space in Multimedia Chat Communities. In Michael Beiswenger, editor, *Chat Communication*, pages 305–344. Ibidem, Stuttgart, 2001. `http://www-usr.rider.edu/~suler/psycyber/psyav.html`, (Verfügbar 20.02.08).

[Sul07] J. Suler. Second Life, Second Chance. `http://www-usr.rider.edu/%7Esuler/psycyber/secondlife.html`, Januar 2007. (Verfügbar 20.02.08).

[Svo06] W. Svojanovsky. *Darstellung von Emotionen als Repräsentation von Dialogerfolg.* Diplomarbeit, Universität Karlsruhe (TH), November 2006.

[Tho07] J.-N. Thon. Kommunikation im Computerspiel. In Simone Kimpeler, Michael Mangold und Wolfgang Schweiger, editors, *Die digitale Herausforderung*, pages 171–180. VS, Wiesbaden, 2007.

[Tur50] A. M. Turing. Computing Machinery and Intelligence. *Mind*, 59:433–460, October 1950. One of the most influential papers in the history of the cognitive sciences: `http://cogsci.umn.edu/millennium/final.html`.

[Vog63] R. Vogelsang. *Die mathematische Theorie der Spiele.* Fred Dümmlers Verlag, Bonn, 1963.

[Wal06] C. Walter. Künstliche Emotionen. `http://www.dfki.de/~kipp/seminar/folien/Christine_Emotionen.pdf`, 2006. (Verfügbar 21.06.06).

[WBB02] T. Wright, E. Boria und P. Breidenbach. Creative Player Actions in FPS Online Video Games. *the international journal of computer game research*, 2(2), 2002. `http://www.gamestudies.org/0202/wright/`, (Verfügbar 20.02.08).

[WCW+08] Adam Webb, Jason Chin, Thomas Wilkins, John Payce und Vincent Dedoyard. Automated negotiation in the game of diplomacy. `http://www.doc.ic.ac.uk/project/2007/362/g0736203/TheDiplominator/TheDiplominator/finalreport.pdf`, 1 2008. (Verfügbar 05.02.09).

[Wei66] J. Weizenbaum. ELIZA–A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 1966. `http://i5.nyu.edu/%7Emm64/x52.9265/january1966.html`, (Verfügbar 01.03.08).

[Wei94] J. Weizenbaum, editor. *Die Macht der Computer und die Ohnmacht der Vernunft*, 274, Frankfurt am Main, 1994. Suhrkamp.

[Wei07] K. Weicker. *Evolutionäre Algorithmen*. Teubner, Wiesbaden, 2007.

[Wer74] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, 1974.

[Whe97] Joseph Wheeler. On diplomacy. `http://www.diplom.org/Zine/S1997M/Wheeler/Clausewitz.html`, 1997. (Verfügbar 04.02.09).

[Win97] Paul D. Windsor. Lawyer/diplomat. `http://devel.diplom.org/DipPouch/Zine/F1997R/Windsor/lawdip.html`, 1997. (Verfügbar 04.02.09).

[Win98] Paul D. Windsor. Chainsaw diplomacy. `http://devel.diplom.org/DipPouch/Zine/W1998A/Windsor/Chainsaw.html`, 1998. (Verfügbar 04.02.09).

[Win99] Paul D. Windsor. What's your point. `http://devel.diplom.org/DipPouch/Zine/S1999M/Windsor/point.html`, 1999. (Verfügbar 04.02.09).

[Woo99] Tom Woodhouse. Diplomatic tactics. `http://www.diplom.org/Zine/S1999M/Woodhouse/tactics.html`, 1999. (Verfügbar 04.02.09).

[Wri91] A. H. Wright. Genetic Algorithms for Real Parameter Optimization. In Gregory J. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 205–218, San Mateo, CA, 1991. Morgan Kaufmann. `citeseer.ist.psu.edu/wright91genetic.html`.

[Yee02] N. Yee. Adolescent identities: Assemling the self. `http://www.nickyee.com/mosaic/adolescence.html`, 2002. (Verfügbar 20.02.08).

[Yee03a] N. Yee. Communication / Relationship Skills. `http://www.nickyee.com/daedalus/archives/000339.php`, 2003. (Verfügbar 20.02.08).

[Yee03b] N. Yee. Identity Projection. `http://www.nickyee.com/daedalus/archives/000431.php?page=1`, 2003. (Verfügbar 20.02.08).

[Yee06] N. Yee. VoIP Usage. `http://www.nickyee.com/daedalus/archives/001519.php?page=1`, 2006. (Verfügbar 20.02.08).

[Zad65] L. Zadeh. *Fuzzy Sets - Information and Control*. Academic Press, New York, 1965.