Projekt: Multiagent Wumpus World

Oliver Böttner, Darius Marzisch, Hannes Lackner,
Ronald Nguyen, Michael Gutbrod

January 2025

Git-Repository google docs präsi

1 Multiagenten-Wumpus-Welt

1.1 Charakterisierung

Unsere Implementierung basiert auf der urspünglichen Wumpus-Welt der Vorlesung. Das bedeutet, es gibt ein quadratisches Spielfeld mit Feldern, die von den Agenten betreten werden können. Es gilt die Von-Neumann-Nachbarschaft und die Agenten starten in den Eckfeldern. Das Verhalten der Wumpus-Welt ist in der environment.py beschrieben. Die Implementierung ist objekt-orientiert – die Interaktion der Welt mit den Agenten läuft über das Aufrufen der Agentenfunktionen.
Auf dem Feld werden Pits und große und kleine Wumpi verteilt. Wenn ein Agent so ein Feld betritt, stirbt er und verliert Gold. Auf den Nachbarfeldern erhält der Agent eine Beobachtung, BREEZE bei einem Pit, SMELLY bei einem kleien Wumpus und VERY_SMELLY bei einem großen Wumpus. Das Ziel der Agenten ist es, möglichst viel Gold zu sammeln. Dafür kann er zum einen auf dem Feld verteiltes Gold finden, Wumpi töten oder andere Agenten überfallen. Um einen Wumpus zu töten, muss ein Agent mit einem Pfeil auf einem nebenliegenden Feld in die Richtung des Wumpus schießen. Ein kleiner Wumpus stirbt bei einem Pfeil und hinterlässt ein kleines Goldvorkommen, ein großer stirbt, wenn ihn zwei Pfeile gleichzeitig treffen und hinterlässt ein großes Goldvorkommen. Ein Agent startet mit Pfeilen oder kann sich zu Beginn jeder Runde welche kaufen. Die Agenten haben die Möglichkeit, einen Funkspruch abzugeben. Dabei können sie den anderen Agenten mitteilen, wo sich Wumpi befinden, wo möglicherweise Pits sind und wann sie auf (große) Wumpi schießen werden. Somit wird eine Zusammenarbeit zwischen den Spielern ermöglicht. Sobald sie einen Funkspruch abgegeben haben, müssen sie 5 Runden warten, bis sie einen weiteren Funkspruch abgeben können. Wenn zwei Agenten auf dem gleichen Feld landen, hat jeder Agent die Möglichkeit, nichts zu tun oder den anderen zu überfallen und bekommt oder verliert entsprechend Gold. Wenn ein Agent aber eine Rüstung gefunden hat, ist er immun gegen einen Überfall.

Ein Spiel endet, wenn entweder alle Agenten oder alle Wumpi tot sind oder nach einer zufälligen Anzahl an Spielzügen. In dem folgenden Spiel starten die Agenten wieder in den gleichen Ecken und behalten ihr Gold vom vorigen Spiel. In der simulation.py kann die Gesamtanzahl an Spielen eingestellt werden.

In der simulation.py können die verschiedenen Eigenschaften für einen Spieldurchlauf festgelegt werden. Darunter die Feldgröße, Anzahl an Pits, kleiner und großer Wumpi und kleiner und großer Goldvorkommen, die Menge an Gold, die Agenten für das Finden der Goldvorkommen bekommen, den Kaufpreis und die Anfangsmenge der Pfeile. Außerdem kann die Ergebnismatrix des Aufeinandertreffens der Agenten verändert werden.
Wie diese Konzepte mit der Vorlesung zusammenhängen, wird im Folgenden beschrieben.

1.2 Ansätze aus der Vorlesung

1.2.1 Spieltheorie

Wenn zwei Agenten aufeinander treffen, haben sie jeweils die Möglichkeit, auszurauben oder nichts zu tun. Diese Situation ist ähnlich dem Gefangenendilemma. Die Ergebnismatrix kann dabei angepasst werden. Die Situation ist durch Rüstungen erweitert. Außerdem können sich zwei Agenten mehrmals begegnen und wissen, wem sie begegnen, so dass sie sich vorherige Aufeinandertreffen merken können. Sie wissen nicht, wann eine Runde beendet wird und wissen somit nicht, welches Aufeinandertreffen ihr letztes in einer Runde ist. Sie haben aber die Möglichkeit, durch die Wahl ihrer Strategien Einfluss auf die Treffen zu nehmen. Zum einen können sie auf dem Spielfeld herumwandern und versuche, Rüstungen zu finden, zum anderen können sie vermeiden, sich übermäßig viel herumzubewegen und so Treffen aus dem Weg gehen.

1.2.2 Kommunikation

Bei der Kommunikation können die Agenten mit einer Kombination aus Performative Verb und Content kommunizieren, das heißt die Nachrichten sind ähnlich aufebaut wie in ACL von FIPA. Der Aufbau der Nachrichten wird in 2 beschrieben. Die möglichen performatives sind inform und agree. Diese sind ebenfalls in ACL enthalten.
Aus der Vorlesung: Inform: passing Information.
Bei uns: Informationen über den Standort des Wumpus.
Aus der Vorlesung: agree: performing action
Bei uns: Akzpetieren zum Wumpus zu laufen und zu schießen.

2 Simulation und Umsetzung

Die environment.py enthält die Spielsimulation. Zuerst wird das Spielfeld erstellt. Jedes Feld enthält in einem dictionary die Informationen für Agenten auf

dem Feld, Pits, Wumpus, Gold oder Rüstung. Dann werden die Agenten positioniert und anschließend die Pits, Wumpi, Goldvorkommen und Rüstung. Die Agenten werden dem Spiel übergeben, so dass dieses ihre jeweiligen Funktionen aufrufen kann. Beispielsweise wird im Anschluss die Kauf-Funktion der Agenten für Pfeile wird aufgerufen. Die Agenten geben eine Zahl an Pfeilen zurück, die sie kaufen wollen, dann wird ihnen dementsprechend Gold abgezogen.
Dann haben die Agenten die Möglichkeit Funksprüche zu senden, wenn für den jeweiligen Agenten die 5 Runden abgelaufen sind, die er warten muss, um eine Nachricht zu senden. Diese besteht aus einem eindimensionalen Array [inform“, w(x1,y1) p(x2,y2) s(m, x3, y3)]. Die Nachricht ist zu verstehen als: Wumpus bei x,y, position bei x2, y2 und ich schieße, auf der Position stehend x3, y3, in m Zügen. Diese Nachricht wird an alle anderen Agenten verschickt. Falls ein anderer Agent den Wumpus ebenfalls töten möchte, kann der Agent eine Nachricht mit dem perfomative Verb ägree“: [ägree“, player_id] senden. Dabei ist die player_id die ID des Agenten, der die ursprüngliche Nachricht geschickt hat.
Anschließend wird die Pfeil-Funktion der Agenten aufgerufen. Zum Schluss wird die Bewegen-Funktion der Agenten aufgerufen. Dann wird das Spielfeld geupdated. Wenn Agenten auf ein Feld mit einem Pit, Wumpus oder vom Spielfeld laufen, sterben sie, indem ihr Status auf dead gesetzt wird. Wenn sie ein Goldvorkommen finden, bekommen sie entsprechend Gold.
Wie bereits in 1.1 erwähnt, können 2 Agenten auf dem gleichen Feld landen und dementsprechend den anderen Agenten ausrauben oder nichts machen. Dabei ergibt sich beispielsweise die folgende Matrix:

agent1/agent2	kooperiert + keine Armor	raubt aus + keine Armor	kooperiert + Armor	raubt aus + Armor
kooperiert + keine Armor	3,3	-4,4	3,3	-4,4
raubt aus + keine Armor	4,-4	-2,-2	0,0	-4,4
kooperiert + Armor	3,3	0,0	3,3	0,0
raubt aus + Armor	4,-4	4,-4	0,0	-1,-1

Die Matrix kann je nach Agenten und nach Szenario angepasst werden.
Damit die Implemetierung funktioniert, muss gelten:
Für alle a_{ij} = (x,y) ∈ A und a_{ji} = (x',y') ∈ A: x = y' und y = x'.
(Daher auch für i = j: x = y)

3 Basisinteraktionen der Agenten

Jeder Agent nimmt die Umgebung auf dem Feld wahr, auf dem er sich befindet, sowie die angrenzenden Felder. Wahrnehmbare Hinweise umfassen Breeze, ein Hinweis auf ein benachbartes Loch, sowie Smelly oder Very Smelly, die auf einen

kleinen oder großen Wumpus in der Nähe hindeuten. Basierend auf diesen Wahrnehmungen aktualisieren die Agenten ihre Wissensbasis, um gefährliche Felder wie Löcher oder Wumpi zu identifizieren und sichere Felder zu markieren. Dies hilft ihnen, informierte Entscheidungen zu treffen und Risiken zu vermeiden.
Die Agenten können sich innerhalb des Spielfeldes bewegen, dessen Größe sie bei der Initialisierung erhalten. Ihre Bewegungsentscheidungen basieren auf ihrer Wissensbasis und ihrer aktuellen Zielsetzung. Dabei gibt es drei grundlegende Bewegungsarten: Sicheres Erkunden, bei dem bekannte Gefahren vermieden und sichere Felder bevorzugt betreten werden, gezieltes Navigieren, bei dem Algorithmen wie A*-Suche genutzt werden, um effizient Ressourcen wie Gold oder Rüstung zu erreichen, sowie zufällige Bewegung, bei der sich die Agenten innerhalb sicherer Bereiche bewegen, falls keine spezifischen Ziele definiert sind.
Die Agenten interagieren auch mit den Elementen der Umgebung, um Ressourcen zu sammeln oder Gefahren zu vermeiden. Beim Betreten eines Feldes mit Gold wird dieses gesammelt, während Rüstungen aufgenommen werden, um Angriffe anderer Agenten abzuwehren. Pfeile können verwendet werden, um entfernte Wumpi zu eliminieren, während das Betreten von Feldern mit Löchern oder Wumpi zum sofortigen Tod des Agenten führt.
Jeder Agent besitzt eine Wissensbasis, die den Zustand jedes Feldes auf dem Spielfeld repräsentiert. Diese Wissensbasis enthält mögliche Zustände wie sicher, Loch, kleiner oder großer Wumpus, Gold und unbekannt. Zusätzlich gibt es Blöcke, die bestimmte Zustände ausschließen, wie zum Beispiel ein Feld ohne Breeze, das kein Loch haben kann. Die Wissensbasis wird kontinuierlich aktualisiert und bildet die Grundlage für alle Entscheidungen der Agenten. Je nach Plan, etwa Erkunden oder gezieltes Bewegen, wird die Wissensbasis verwendet, um sichere Routen oder potenzielle Gefahren zu identifizieren.
Bei Begegnungen auf demselben Feld können die Agenten sich überfallen oder nichts tun und sie können mit anderen Agenten zum beispiel über das Töten der Wumpi kommunizieren.

4 Lösungsstrategien der Agenten

4.1 Basislogik für Lösungsstrategien

Planung/Ziele: Jeder Agent nutzt in der Regel einen eigenen Plan bestehend aus einem *Typ* (RANDOM, GO_TO, WAIT, EXPLORE, COLLECT_GOLD), *Zielfeldern*, *Wumpusfelder* und *Geduld*. Ein Plan bestimmt, ob der Agent sich rein-zufällig über Felder bewegt, zu bestimmten Feldern reist, auf einem Feld untätig stehenbleibt oder gezielt alle ihm unbekannten Felder exploriert. Dabei läuft, insofern spezifiziert, eine numerische *Geduld* ab, was zum Abbrechen des Plans führt. Dies ist nützlich, um Pläne zu wechseln, wenn sie sich zu sehr in die Länge ziehen und daher weniger nützlich sind. Anfangs starten Agenten mit dem RANDOM/EXPLORE-Plan, um mehr über die unbekannte Welt zu erfahren. Der Plan kann sich während des Spiels unter Umständen ändern (siehe *Wumpus-töten*).

Knowledge-Base: Agenten speichern Wissen über die Welt in einem 2D-Grid mit möglichen/ausgeschlossenen Positionen von Pits, kleinen/großen Wumpi, usw. Vor jedem Zug wird diese Knowledge-Base genutzt, um basierend auf Perceptions das Wissen zu aktualisieren:
1. Falls keine Perception aufgenommen wurde, können Nachbar-Felder als sicher angenommen werden. Gefahren werden komplett ausgeschlossen.
2. Falls Perceptions aufgenommen wurden, werden auf allen nicht-sicheren Feldern die entsprechenden Gefahren angenommen. Restliche Gefahren werden analog ausgeschlossen.
Mit diesen Regeln können Agenten mit festgelegter Risikoaversion Hindernisse vermeiden.
Wumpus-töten: Jede Runde versucht ein Agent, insofern er nichts vorhat (RANDOM-Plan), eine Nachricht zu schicken bzw. empfangen. Zum Aufbau der Nachrichten siehe 2. Falls der Plan RANDOM ist und der Agent weiß wo sich ein großer Wumpus aufhält versendet er eine *inform*-Performative Nachricht. Beim Empfangen des *inform*-Performative entscheidet der (andere) Agent, aufgrund der Entfernung zum Ziel, ob er zum Wumpus laufen möchte und diesen zusammen mit dem (ersten) Agenten töten möchte. Falls er sich für den Plan entscheidet, schickt er eine *agree*-Performative Nachricht. Falls beim ersten Agenten keine *agree*-Performative Nachricht ankommt bricht er den Plan ab. Beide wählen ihre Geduld nach der Zahl, wie lange der erste Agent (der Nachrichtensender), vermutet zu brauchen, um beim Wumpus anzukommen. Als Suchalgorithmus wird A*-Search mit Manhattan-Distanz als Heuristik angewendet. Sollte es dazu kommen, dass einer der Agenten keinen Pfad zum Ziel findet, oder die Geduld abläuft, wird dessen Plan abgebrochen. Sobald einer der Agenten das Ziel gefunden hat, wartet er bis der andere Agent angekommen ist oder bis die Geduld abgelaufen ist. Falls sich die beiden Agenten auf dem Feld neben dem Wumpus treffen, schießen sie. Wenn die Perception nicht mehr VERY_SMELLY ist, sammeln sie das Gold ein.

4.2 Lösungsstrategie vom Cooperative Agent

Der CooperativeAgent verfolgt eine Strategie, die stark auf Interaktionen mit anderen Agenten basiert, um möglichst effizient Gold zu sammeln. Dabei setzt er die bewährte Tit-for-Tat-Strategie ein, die sich im wiederholten Gefangenendilemma als eine der erfolgreichsten Strategien erwiesen hat. Wenn ein anderer Agent zuvor kooperativ gehandelt hat, erwidert der CooperativeAgent dieses Verhalten ebenfalls mit Kooperation. Sollte er jedoch in der Vergangenheit beraubt worden sein, reagiert er bei zukünftigen Begegnungen mit Vergeltung, um sein eigenes Verhalten anzupassen und mögliche Verluste auszugleichen.
Neben der Interaktion mit anderen Agenten, kommuniziert der Cooperative Agent auch mit anderen Agenten um gezielt große Wumpi zu töten, dabei versendet er die Koordinaten von großen Wumpi und tötet diese anschließend mit einem anderen Agent. Außerdem geht der Cooperative Agent gezielt zu großen Wumpi, wenn er eine Nachricht erhält, wo sich einer befinden könnte.

4.3 Lösungsstrategie vom Defensive Agent

Der Defensive Agent fokussiert sich primär auf das Sammeln von Gold und vermeidung von Gefahren.
Bei Begegnungen mit anderen Agenten setzt der Defensive Agent seine Rüstung ein, um sich vor Raub zu schützen. Sollte er jedoch keine Rüstung besitzen, versucht er, selbst einen Raub durchzuführen, um potenzielle Verluste auszugleichen.
Das zentrale Konzept dieser Strategie liegt in der Balance zwischen defensivem Verhalten (Schutz vor Angriffen und Gefahren) und effektivem Ressourcenmanagement (insbesondere das Sammeln von Gold), um eine maximale Punktzahl zu erzielen.

4.4 Lösungsstrategie vom Aggressiven Agenten

Der AggressiveAgent verfolgt eine Strategie, die stark auf konfrontative Aktionen ausgerichtet ist. Er nutzt die Tit-for-Tat-Strategie, wobei er bei der ersten Begegnung immer ausraubt. Eine besondere Fähigkeit des AggressiveAgent ist das Verbreiten falscher Informationen über die Positionen von Wumpi, um Gegner zu verwirren. Außerdem kauft der Agent Pfeile, sobald er Gold hat. Dieses stellt sicher, dass der Agent immer ausreichend bewaffnet ist um gezielt auf Wumpis zu schießen. Der Agent aktualisiert kontinuierlich sein Wissen basierend auf Wahrnehmungen und markiert Positionen als sicher oder gefährlich. Er bewegt sich strategisch basierend auf seiner aktuellen Wahrnehmung und Zielsetzung.

4.5 Der Randommeeting Agent

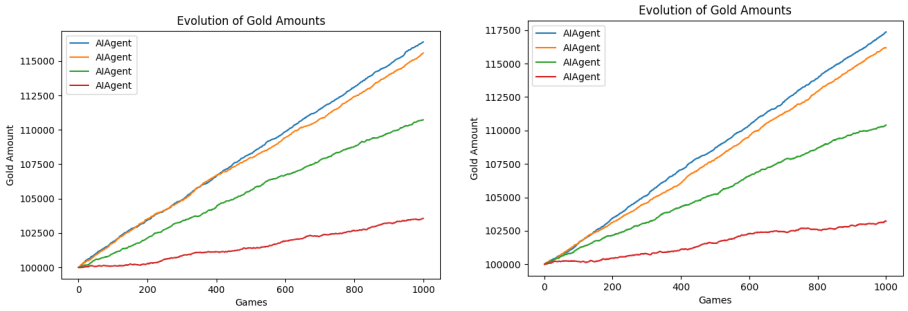
Der Random Meeting Agent ist ein AIAgent, der, beim Meeting, zufällig ausraubt oder nichts macht.

5 Evaluation der Lösungsstrategien

Um den Erfolg der Lösungsstrategien zu bewerten, können die unterschiedlichen Agenten mit unterschiedlichen Grid-Eigenschaften gegeneinander angetreten lassen werden. Anhand des Goldes der Agenten kann so nach einer großen Anzahl an Spielen festgestellt werden, welcher Agent mit der entsprechenden Strategie am besten abschneidet. Außerdem kann einem Agenten das Risikoverhalten in Form eines Parameters übergeben werden. Dieser Einfluss sollte auch Untersucht werden.
Um zuerst den Einfluss des Risikoverhaltens zu untersuchen, treten 4 Basis-Agenten ohne besondere Strategien und mit unterschiedlichen Risikoverhalten (Abb. 1) gegeneinander an. Dabei zeigt sich in Abb. 2 recht eindeutig, dass je geringer das Risiko ist, welches die Agenten eingehen, desto besser schneiden sie insgesamt ab.

```python
# pass the following setup for experimenting
testing_agents = [
    AIAgent(size=size, risk_aversion=1),
    AIAgent(size=size, risk_aversion=0.99),
    AIAgent(size=size, risk_aversion=0.95),
    AIAgent(size=size, risk_aversion=0.8),
]
```

Abbildung 1: Agenten mit unterschiedlichen Risikoverhalten



(a) Durchlauf 1

(b) Durchlauf 2

Abbildung 2: Perfomance AIAgenten

Um das Abschneiden der Agenten mit Ihren Strategien zu untersuchen, treten die Agents mit der Ergebnismatrix aus Abb. 4, die normales Gefangenendilemma darstellt, mit den Grid-Eigenschaften aus Abb. 3, gegeneinander an (Abb. 5). Der RandomMeeting Agent ist dabei der einzige Agent, der Gold verliert. Der Defensive Agent erhält wesentlich mehr Gold, als der Rest und der AggressiveAgent schlägt den CooperativeAgent nur knapp.

Wird die Ergebnismatrix in Abb. 6 so angepasst, dass die Agenten nicht mehr bestraft werden, wenn beide ausrauben, ändert sich das Ergebnis (Abb. 7). Nun verliert der RandomMeetingAgent kein Gold mehr und der CooperativeAgent schlägt jetzt den AggressiveAgent knapp.

Wenn die Ergebnismatrix wieder das normale Gefangendilemma darstellt (Abb. 8), die Agenten nun aber sehr viel Gold erhalten, wenn sie Kooperieren, dann verliert der DefensiveAgent und der RandomMeetingAgent gewinnt und der AggressiveAgent schlägt den CooperativeAgent wieder knapp.

6 Ergebnisdiskussion

6.1 AiAgent

Es ist zu sehen, dass eine niedrigere riskaversion auch zu weniger Gold führt (2). Das liegt daran, dass der Agent häufiger stirbt. Für das Steben bekommt der Agent Gold abgezogen und kann den Rest der Runde kein weiteres Gold

```python
grid_properties = {
    "size": 7,          # Grid size: must be odd
    "num_pits": 1,      # Pits: this should be pretty low
    "num_s_wumpi": 2,   # Smol wumpi: this should be pretty low
    "num_l_wumpi": 1,   # Large wumpi: this should be pretty low
    "num_small_gold": 2,# Gold spawns
    "num_large_gold": 2,# Gold spawns
    "num_armor": 2,     # Armor spawns
    "small_gold": 5,    # Amount of gold for small gold
    "large_gold": 10,   # Amount of gold for large gold
    "arrow_price": 2,   # Price for arrows
    "amount_arrows_start": 2, # Amount of arrows at the start
```

Abbildung 3: Grid-Eigeschaften

agent1/agent2	kooperiert + keine Armor	raubt aus + keine Armor	kooperiert + Armor	raubt aus + Armor
kooperiert + keine Armor	3,3	-4,4	3,3	-4,4
raubt aus + keine Armor	4,-4	-2,-2	0,0	-4,4
kooperiert + Armor	3,3	0,0	3,3	0,0
raubt aus + Armor	4,-4	4,-4	0,0	-1,-1

Abbildung 4: normales Gefangenendilemma

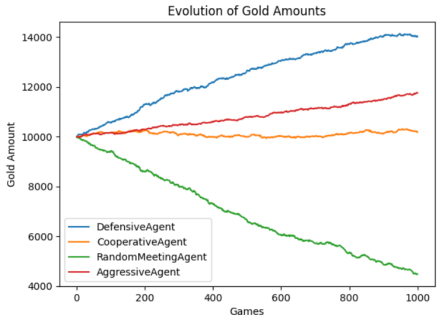


Abbildung 5: Ergebnis mit Matrix aus Abb. 4

agent1/agent2	kooperiert + keine Armor	raubt aus + keine Armor	kooperiert + Armor	raubt aus + Armor
kooperiert + keine Armor	3,3	-4,4	3,3	-4,4
raubt aus + keine Armor	4,-4	0,0	0,0	-4,4
kooperiert + Armor	3,3	0,0	3,3	0,0
raubt aus + Armor	4,-4	4,-4	0,0	0,0

Abbildung 6: Gefangenendilemma, keine Bestrafung beim Ausrauben

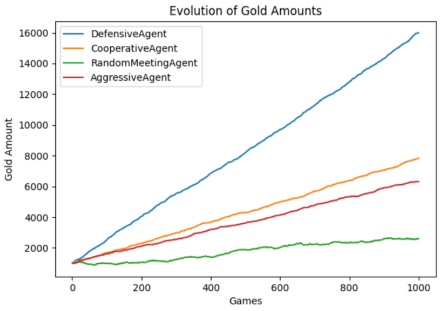


Abbildung 7: Ergebnis mit Matrix aus Abb. 6

agent1/agent2	kooperiert + keine Armor	raubt aus + keine Armor	kooperiert + Armor	raubt aus + Armor
kooperiert + keine Armor	100,100	-4,4	100,100	-4,4
raubt aus + keine Armor	4,-4	-2,-2	0,0	-4,4
kooperiert + Armor	100,100	0,0	100,100	0,0
raubt aus + Armor	4,-4	4,-4	0,0	-1,-1

Abbildung 8: Gefangenendilemma, hohe Belohnung beim Kooperieren

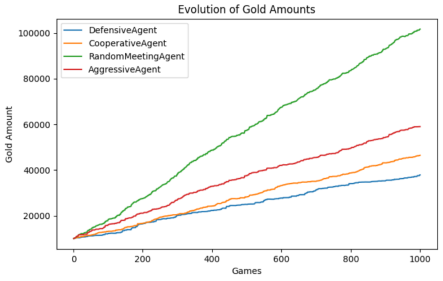


Abbildung 9: Ergebnis mit Matrix aus Abb. 8

einsammeln.

6.2 Cooperative Agent

Der Cooperative Agent wurde so konzipiert, dass er eine ausgewogene Strategie verfolgt: Er profitiert stark von Kooperation, ist jedoch gleichzeitig nicht vollständig von anderen Agenten abhängig. Die bisherigen Simulationsergebnisse zeigen, dass der Cooperative Agent mit der aktuellen Ergebnismatrix (Abb. 4) in Meetings der drittbeste Agent ist (Abb. 5). Dies liegt daran, dass er sich flexibel an das Verhalten der anderen Agenten anpasst und dadurch eine konstante Leistung erbringt, aber auch viel Gold verlieren kann, wenn er mit einem unberechenbaren Agenten interagiert. Interessanterweise ist er mit einer neuen Ergebnismatrix (Abb. 6), welche das Ausrauben nicht mehr bestrafen lässt, besser als der Aggressive Agent(Abb. 7). Da die Probleme aus dem vorherigen Durchlauf (Abb. 5) nicht mehr vorhanden sind und davon profitiert der Cooperative Agent sehr. Wenn nun aus einer Kooperation eine sehr hohe Belohnung erfolgt (Abb. 8), so schneidet der Cooperative Agent auch durchschnittlich ab, wird aber wieder schlechter als der Aggressive Agent. Das passiert, weil er mit Agenten wie dem Aggressive Agent oder dem Defensive Agent in eine Negativspirale kommen kann, wodurch er nicht so oft kooperiert wie zum Beispiel der RandomMeetingAgent. Das heißt, dass der Cooperative Agent sich bei Veränderungen anpassen kann, aber nur wenn andere Agenten nicht immer das Gleiche machen, an diesem Punkt verfliegt der Vorteil vom Cooperative Agent und deshalb ist er immer im Mittelfeld und nie der schlechteste oder beste Agent.

6.3 Defensive Agent

Der Defensive Agent sollte ein Agent sein, der viel von der Armor profitiert und bei den meisten Meetings ausraubt. So sieht man in Abb. 5, dass er der beste Agent ist, da er durch viel ausrauben sein Gold erwirtschaftet. Mit der neuen Ergebnismatrix (Abb. 6) erkennt man in Abb. 7, dass sich der Abstand

zu den anderen Agenten vergrößert hat. So profitiert der Defensive Agent viel mehr als die anderen Agenten davon, dass beim Ausrauben kein Gold verloren werden kann. Das liegt daran, dass der Defensive Agent viel mehr ausraubt als die anderen Agenten. Wird die Ergebnismatrix (Abb. 8) so angepasst, dass aus einer Kooperation eine Belohnung erfolgt, so ist der Defensive Agent am schlechtesten (Abb. 9), da er nur mit Armor kooperiert und die Belohnung der Kooperation mit Armor und ohne ist gleich, somit entfällt dieser Vorteil.

6.4 Aggressive Agent

Mit der Ergebnismatrix aus Abb. 4, die das normale Gefangenendilemma darstellt, bekommt der AggressiveAgent Gold durch das Ausrauben, verliert aber auch Gold, wenn der andere beteiligte Agent ausraubt. Durch die Bestrafung fällt der Goldzuwachs in Abb. 5 insgesamt nur schwach aus, liegt aber im Positiven. Bei stärkerer Bestrafung ist zu erwarten, dass der Agent insgesamt Gold verlieren wird. Ohne Bestrafung (Abb. 6), ist der Goldzuwachs deshalb auch deutlich größer. Da der AggressiveAgent nicht immer ausraubt, sondern seine Strategie auch Kooperationen zulässt, ist der Goldgewinn bei großer Belohnung für Kooperation (Abb. 8) deutlich größer (Abb. 9).

6.5 Randomeeting Agent

Der Randommeeting Agent dient als Vergleichsagent zu den anderen Agenten. Er gewinnt das Spiel, wenn beide Agenten bei einem Meeting viel Gold bekommen, wenn beide nichts machen, siehe 9. Das liegt daran, dass er 50% der Zeit nichts macht. Da er dann möglicherweise sehr viel Gold bekommt, lohnt es sich, ohne Taktik, häufig nichts zu machen. Dahingegen ist er bei 5 und bei 7 letzter Platz, da er ohne Taktik zufällig entscheidet, ob er ausraubt und so von den anderen Agenten ausgenutzt wird.