



Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Institut für Technische und Theoretische Informatik
Fachgebiet Neuroinformatik

STUDIENJAHRESARBEIT

Das Verhalten von Reinforcement-Agenten unter dem Einfluss von Problemen der Realwelt

Manuela Bischoff	Marco Saupe
27475	27241
INF98	INF98

Ilmenau, den 4. Juli 2003

betreut von Dr.-Ing. Volker Stephan

Inhaltsverzeichnis

1. Einleitung und Zielstellung	5
2. Grundlagen der Simulation	6
2.1. Simulation von Realwelteigenschaften	6
2.2. Die Simulationsumgebung	6
2.3. Reinforcement-Agenten	9
2.3.1. Clusterer	9
2.3.2. Lernverfahren	13
3. Benchmarking	16
3.1. Die Testumgebung	16
3.1.1. Die Parameter der Simulation und der Ballwelt	16
3.1.2. Die Parameter der Netzwerke	16
3.2. Versuchsplan	18
3.2.1. Wahl der Agenten	18
3.2.2. Beschreibung der Experimente	19
4. Testergebnisse und Auswertung	22
4.1. Standard-Szenario und das Clusterer-Problem	22
4.2. Verschiedene Reward-Funktionen	26
4.3. Verschiedene Gebirge	30
4.4. Verschiedenes Rauschen	40
4.5. Vorhandensein von Totzeiten	41
4.6. Veränderliche Zielstellung	44
4.7. Teilweise beobachtbare Umwelt - Störgröße Wind	52
4.8. Anzahl der möglichen Aktionen des Agenten	54
4.9. Kombination von Teilproblemen	56
4.10. Zusätzlicher Test zu den Q-Sarsa-Agenten	58
5. Zusammenfassung und Ausblick	60
Literaturverzeichnis	64
Abkürzungsverzeichnis	65

Abbildungsverzeichnis	66
A. Abbildungen	68

1. Einleitung und Zielstellung

Animals responses to environment in a given situation are adapted such that the probability for finding satisfaction if this situation occurs again is enhanced.

[Throndike, 1911]

Dieses Prinzip versucht der Mensch seit geraumer Zeit auf die eine oder andere Art und Weise nachzuahmen, indem er daran arbeitet lernende Maschinen zu entwickeln. Unter den Lernverfahren ist das *Reinforcement*-Lernen wohl das, welches dem biologischen Lernen am nächsten kommt; ausgelegt auf Belohnung (*reward*) bzw. Bestrafung nach Erfolg oder Misserfolg. Das Konzept des maschinellen Lernens wendet sich von der Vorstellung ab, dass eine Maschine nur das kann, was der Entwickler ihr einprogrammiert. Vielmehr soll ein lernendes System, ein *Agent*, über die Zeit Aktionen auswählen, um seine Umgebung in gewünschter Weise zu beeinflussen oder in einen Zielzustand zu führen. Durch die Bewertung seines Verhaltens über die *Reward-Funktion* soll er mittels lokaler Entscheidungen eine global optimale Strategie zur Lösung eines praktischen Problems finden.

Reinforcement-Lernen ist bei solchen Aufgaben sinnvoll, bei denen das Lernziel nur sehr unspezifisch beschrieben ist. So wird z. B. der amtierende Weltmeister im Backgammon-Spiel durch einen Reinforcement-Lernalgorithmus gestellt.

Ein Anwendungsgebiet ist der Einsatz von Lernverfahren unter Realweltbedingungen. Im Gegensatz zu Spielen wie Backgammon, welches einen diskreten Situations- und Aktionsraum bietet, ergeben sich in der realen Welt für einen Agenten neue Probleme, wie die Kontinuität des Eingabe- und Aktionsraums, das Vorherrschen von Totzeiten oder eine veränderliche Umwelt.

Helge Renkewitz untersuchte in seiner Diplomarbeit [Renkewitz, 2002] zum ersten Mal verschiedene Reinforcement-Learning-Agenten (RL-Agenten) auf ihre Realwelttauglichkeit. Dies geschah mit Hilfe einer simulierten Umgebung, da nur hier die Umweltbedingungen einfach reproduzierbar und steuerbar sind und somit ein Benchmarking erst möglich wird.

Aufbauend auf die Diplomarbeit und die dort implementierte Applikation sollen nun in dieser Arbeit Aspekte untersucht werden, die offen geblieben sind. Dazu zählen unter anderem Tests zu anderen Reward-Funktionen sowie Erweiterungen der getesteten Agenten. Ein wichtiger Teil ist hierbei der Einsatz von inkrementellen Clusterern und deren Vergleich mit den schon getesteten Verfahren.

2. Grundlagen der Simulation

2.1. Simulation von Realwelteigenschaften

Die Realwelt ist sehr komplex und lässt sich nicht formal beschreiben. Für den technischen Bereich und speziell für den Einsatz von RL-Agenten sind jedoch einige wichtige Aspekte zu erkennen. Folgende wurden in der Diplomarbeit von [Renkewitz, 2002] bereits umgesetzt und für die Experimente genutzt:

- kontinuierliche und/oder sehr große Zustands- und Aktionsräume
- nur teilweise beobachtbare Umwelt (POMDP)
- Veränderung der Umwelt oder der Zielstellung
- selten Reward
- Vorherrschen einer Totzeit
- beschränkte Energieresourcen

Die genannten Aspekte werden auch in dieser Arbeit zum Teil für Experimente herangezogen; eine genaue Beschreibung der Versuche folgt in Abschnitt 3.2. Um zu testen, ob RL-Agenten mit diesen Problemen umgehen können, wird auf die bereits in der Diplomarbeit ausgearbeitete Simulationsumgebung zurückgegriffen, die in Abschnitt 2.2 näher erläutert werden soll. Der Vorteil einer Simulation in diesem Zusammenhang ist die Reproduzierbarkeit der Umgebungsbedingungen und die Vergleichbarkeit der Untersuchungen. Jedoch kann die Simulation niemals die komplexen Zusammenhänge einer „echten“ Realweltanwendung exakt nachbilden und es kann von einem Erfolg eines Agenten in der Simulation nicht auf die Realwelt geschlossen werden. Lediglich die untersuchten Teilaspekte und deren Zusammenspiel können ausgewertet und interpretiert werden.

2.2. Die Simulationsumgebung

In der Diplomarbeit wurde eine einfach mathematisch beschreibbare Testumgebung umgesetzt. Dabei handelt es sich um eine eindimensionale Ballwelt (Abb. 2.1), wobei die Aufgabe des Agenten darin besteht, den Ball an eine bestimmte Stelle zu bewegen. Die Simulation der Ballwelt genügt physikalischen

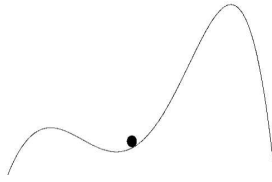


Abb. 2.1: Eindimensionale Ballwelt. Der Ball soll vom Agenten zu einem vorgegebenen Ziel bewegt werden.

Gesetzen, wobei gewisse Parameter variiert werden können, so z. B. die Masse des Balles und die Kraft, mit der ein Agent diesen bewegen kann. Die herrschende Schwerkraft zieht den Ball nach unten, die Masseträgheit verhindert sprunghafte Geschwindigkeitsänderungen. Auf die Umsetzung der Reibung wurde verzichtet, was bedeutet, dass der Ball z. B. auf einer Ebene nicht von alleine langsamer wird.

Für diese Arbeit wird die bereits vorliegende Implementierung in Form einer Java-Applikation genutzt und erweitert. Diese kann mit verschiedenen Umgebungen aufgerufen werden. Die Form der Ballwelt (Szenario) wird hierbei durch ein Polynom beschrieben und kann beliebig verändert werden. Start- und Zielposition können ebenfalls frei festgelegt werden. Im Folgenden sollen kurz die verwendeten Szenarien erläutert werden.

Die Ebene

Dieses Szenario stellt das einfachste dar, da der Agent hier nicht der Schwerkraft entgegenwirken muss, um den Ball an die Zielposition zu bringen und dort zu halten. Wie bereits erwähnt wird der Ball jedoch nicht durch Reibung abgebremst.

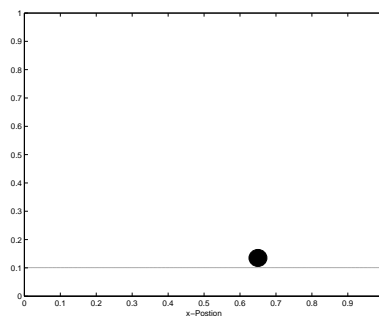


Abb. 2.2: Szenario Ebene mit $p = 0.1$

Die Wanne

Durch den Anstieg zu den Seiten muss der Agent in diesem Szenario lernen, der Schwerkraft entgegenzuwirken, jedoch ist für das Erreichen der Zielposition nicht

die Überwindung eines Gipfels nötig.

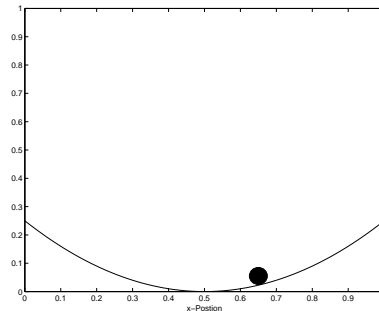


Abb. 2.3: Szenario Wanne mit $p = x^2 - x + 0.25$

Die Welle

Die Schwierigkeit dieses Szenarios liegt bei entsprechender Start- und Zielposition in der Überwindung des Gipfels, falls der Agent nicht genug Kraft hat, den Ball direkt darüber hinwegzubewegen. Dann ist hier nämlich ein „Schwung holen“ nötig.

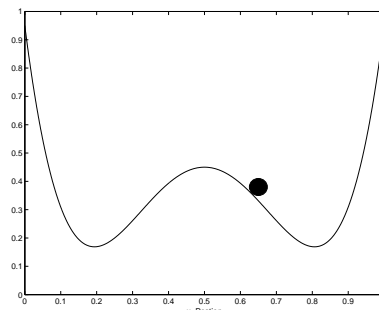


Abb. 2.4: Szenario Welle mit $p = 32x^4 - 64x^3 + 42x^2 - 10x + 0.95$

Der Berg

Dieses Szenario erwies sich in der Diplomarbeit als das schwierigste, da bei einer Startposition von 0.25 und einem Ziel von 0.8 der Agent vor allem einen „Verlust“ des Balles am linken Rand vermeiden muss und der Agent das Ziel nicht erreichen kann.

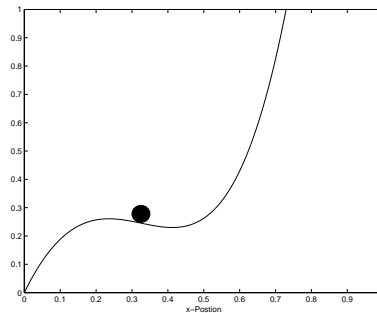


Abb. 2.5: Szenario Berg mit $p = 5x^4 + 4.6x^3 - 7.7x^2 + 2.6x$

2.3. Reinforcement-Agenten

Aufgabe der Agenten ist es zunächst den zweidimensionalen, kontinuierlichen Eingaberaum, bestehend aus aktueller Geschwindigkeit und Position des Balles zu diskretisieren, da die Reinforcement-Lernverfahren nicht mit einer kontinuierlichen Eingabe umgehen können. Diese Aufgabe wird vom sogenannten Clusterer übernommen. Im Folgenden sollen verwendete Cluster- und Lernverfahren kurz erläutert werden.

2.3.1. Clusterer

Cerebellar Model Arithmetic Computer (CMAC)

Mit dem CMAC wurde in [Albus, 1975] ein neuronales Netz entwickelt, welches auf dem Modell des menschlichen Kleinhirns basiert. Es handelt sich um ein nicht-inkrementelles, einschichtiges Netz, dessen Neuronen fest im Raum platziert und untereinander nicht verbunden sind (Abb. 2.6). Jedes Eingangsgewicht \underline{w}_i dieser Neuronen entspricht einem rezeptiven Feld im Eingaberaum, wobei die Summe der durch die Eingabe angesprochenen rezeptiven Felder die Ausgabe des CMAC darstellt.

Zur Berechnung der Aktivierung y_i des Neurons i bei Präsentation eines Eingabevektors \underline{x} wird in dieser Arbeit die folgende einfache Gleichung verwendet:

$$y_i = \begin{cases} 1 & \text{falls } i \text{ Best-Matching-Neuron} \\ 0 & \text{sonst} \end{cases}$$

wobei das Best-Matching-Neuron (BM-Neuron) durch den geringsten euklidischen Abstand seines Eingangsgewichtes \underline{w}_i zum Eingangsvektor \underline{x} gekennzeichnet ist. Diese stark vereinfachte Variante eines CMAC stellt die Tabellenimplementierung des in Abschnitt 2.3.2 vorgestellten Q-Lernens dar.

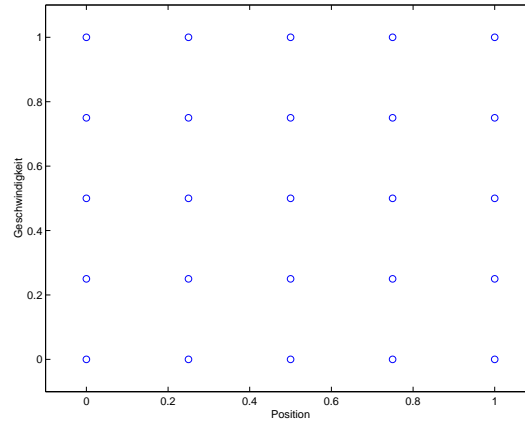


Abb. 2.6: Starre Clustering des Eingaberaumes durch den CMAC am Beispiel des Ballwelt-Problems, Eingaben sind Geschwindigkeit und Position des Balles. Die Anzahl der Neuronen beträgt 25.

Neural Gas (NG)

In [Martinetz und Schulten, 1991] wird ein adaptiver Vektorquantisierer beschrieben, dessen Neuronen eine topologie-erhaltende Clustering des Eingaberaumes ermöglichen. Während der Trainingsphase erlernt das NG die Struktur und Nachbarschaftsbeziehungen der Eingabedaten. Die Neuronen dieses Netzes haben keine Verbindungen untereinander, können also beim Lernen beliebig platziert werden.

Wird in der Lernphase dem NG ein Eingangsvektor $\underline{x}(t)$ präsentiert, gilt für das BM-Neuron r' , welches den geringsten euklidischen Abstand zum Eingangsvektor hat, folgendes:

$$\|\underline{w}_{r'}(t) - \underline{x}(t)\| = \min_{r \in W} \|\underline{w}_r(t) - \underline{x}(t)\|$$

Zur Berechnung der Aktivierungen $y_{rr'}(t)$ der Neuronen ist es nötig, diese aufsteigend nach ihrem Abstand zum BM-Neuron zu ordnen. Jedem Neuron wird dabei sein Index d_r in dieser Distanzreihenfolge zugordnet; für das BM-Neuron gilt $d_{r'} = 0$. Die Aktivierung der Neuronen berechnet sich dann nach folgender Gleichung:

$$y_{rr'}(t) = e^{-\frac{d_r}{b(t)}} \text{ mit } b(t) \text{ ist Lernradius}$$

Damit wird klar, dass das BM-Neuron eine Aktivierung von 1.0 besitzt und alle anderen Neuronen je nach Rang in der Distanzreihenfolge weniger aktiv sind.

Die Gewichte \underline{w}_r der Neuronen werden wie folgt adaptiert:

$$\underline{w}_r(t+1) = \underline{w}_r(t) + \eta(t) \cdot y_{rr'}(t) (\underline{x}(t) - \underline{w}_r(t))$$

wobei η die Lernrate des Neuronalen Gases ist.

Im Gegensatz zu dem Growing Neural Gas aus dem nächsten Abschnitt handelt es sich um einen nicht-inkrementellen Clusterer, d. h. die Anzahl der vorhandenen

Neuronen ist von vornherein fest und es werden weder Neuronen dazugefügt noch gelöscht.

Growing Neural Gas (GNG)

In [Fritzke, 1995] wurde das inkrementelle Netzwerk Growing Neural Gas (GNG) vorgestellt. Dieses Netzwerk ist zusätzlich zu den im Abschnitt des NG erwähnten Eigenschaften in der Lage, mittels Competitive Hebbian Learning [Martinetz, 1993] Topologien zu lernen. Dabei werden die beiden dem Eingangssignal naheliegendsten Neuronen durch eine Kante verbunden um Zusammengehörigkeit abzubilden. Da diese mit der Zeit durch das Wandern der Neuronen mittels Gewichtsadaption im Eingaberaum nicht mehr repräsentativ sein könnten, können die Kanten über einen Alterungsmechanismus wieder entfernt werden, was Verbindungen zwischen getrennten Merkmalsbereichen auflöst.

Ein anderes Problem von nicht-inkrementellen Netzwerken ist, dass im Voraus die Zahl der Neuronen im Netzwerk festgelegt werden muss, was besonders in veränderlichen oder hochdimensionalen Umwelten nicht immer möglich ist bzw. suboptimale Lösungen bewirkt. Im Gegensatz dazu beginnt das GNG seinen Lernvorgang mit nur zwei Initial-Neuronen und fügt im späteren Verlauf zusätzliche Neuronen und Verbindungen zwischen ihnen ein, um die Verteilung der Signalmuster im Eingaberaum möglichst gut abzubilden.

Der Algorithmus des GNG läuft folgendermaßen ab: Die Bestimmung des BM-Neurons erfolgt wiederum durch die Wahl des Neurons mit dem kleinsten euklidischen Abstand seines Gewichtsvektors zum Eingabemuster. Kanten zu den Nachbarn des BM-Neurons werden gealtert, außer der Kante zum Second-BM-Neuron, deren Alter auf 0 gesetzt wird. Wenn die Kante noch nicht existiert, wird sie eingefügt. Kanten, die ein bestimmtes Alter überschreiten, werden ebenso wie Neuronen, die keine Kanten mehr besitzen, entfernt, sofern mindestens zwei miteinander verbundene Neuronen im Netzwerk verbleiben. Der lokale Fehlerzähler des BM-Neurons wird abhängig vom Clusterfehler erhöht, der aller anderen Neuronen um einen Faktor d verringert. Die Gewichte dieses Neurons und seiner Nachbarn werden nach der Gleichung

$$\Delta w_{(bm/n)} = \epsilon_{(bm/n)} \cdot (x(t) - w_{(bm/n)})$$

adaptiert. Nach einer bestimmten Schrittzahl wird zwischen dem Neuron mit dem höchsten Fehlerzähler und demjenigen unter seinen Nachbarn, dessen Fehlerzähler den nächsthöchsten Wert besitzt, ein neues Neuron eingefügt. Dieses wird mit den beiden Neuronen, zwischen denen es entstand, verbunden, und die vorher vorhandene Kante wird gelöscht.

Lebenslang Lernende Zellstrukturen (LLCS)

Die Lebenslang Lernenden Zellstrukturen (LLCS) [Hamker, 2001] sind dem GNG vom prinzipiellen Aufbau und den Grundalgorithmen her sehr ähnlich, besitzen jedoch einige Erweiterungen, die im folgenden Abschnitt kurz erläutert werden sollen.

Ein Grund für diese Erweiterungen ist die Behandlung des Stabilitäts-Plastizitäts-Dilemmas [Grossberg, 1988] beim Lernen. Dieses Dilemma beschreibt das Problem, dass ein Netzwerk in der Lage sein soll veränderliche Muster zu erlernen, ohne dabei bereits erworbene Wissensstrukturen wieder komplett zu zerstören. Zu diesem Zweck besitzen die LLCS eine lokale, adaptive Lernrate, die vom Neuronenalter Y und dem lokalen Lernerfolg B^L abhängig ist. Jedes Neuron besitzt zwei Fehlerzähler τ_S bzw. τ_L , in denen der durchschnittliche Fehler über einen kürzeren bzw. längeren Zeitraum ermittelt wird. Diese werden jeweils für das BM-Neuron aktualisiert. Bleibt die Verteilung der Eingangssignale gleich und das Netzwerk hat dieses Muster erlernt, wird der Kurzzeitfehler kleiner als der Langzeitfehler sein. Tritt jedoch ein Signalmuster auf, das das Netzwerk noch nicht erlernt hat, wird sich der höhere aktuelle Fehler zuerst in einem erhöhten Kurzzeitfehler niederschlagen, wodurch sich der Wert von B^L erhöht.

$$B^L = \frac{\tau_S + 1}{\tau_L + 1}$$

Abhängig von diesem Lernerfolg, vom Alter des jeweiligen Neurons und einer Anpassungsschwelle errechnet sich ein Wert α^i , in dessen Abhängigkeit die globalen Lernraten für das BM-Neuron und seine Nachbarn lokal für das jeweilige Neuron adaptiert werden.

$$\alpha^i = \frac{B^L}{\vartheta_L^i + 1} + Y - 1$$

$$\eta^i = \begin{cases} 0 & \text{if } \alpha^i < 0 \\ \alpha^i & \text{if } \alpha^i > 1 \\ \alpha^i \eta & \text{else} \end{cases}$$

Ein Problem des GNG und verwandter inkrementeller Netzwerke ist es, dass in Gebieten mit hohem Fehler übermäßig viele Neuronen eingefügt werden, was den Rechenaufwand unnötig erhöht. Eine künstliche Festlegung der maximal erlaubten Neuronenzahl könnte die Flexibilität des Netzwerkes aber zu stark beschränken. Um diese Problematik zu behandeln, benutzen die LLCS einen veränderten Einfügemechanismus, der ebenfalls lokal und adaptiv arbeitet. Die Grundidee des Verfahrens ist folgende: Zunächst wird das Neuron mit dem höchsten Wert des Einfügekriteriums bestimmt, welches sich aus lokalem Fehler, der Einfügeschranke und der Einfügetoleranz berechnet. Soll nun dort ein neues Neuron eingefügt werden, wird zunächst überprüft, ob das vorherige Einfügen an dieser Stelle erfolgreich war. Dazu wird ein Vergleich zwischen dem aktuellen lokalen Langzeitfehler und dem durch die lokale Einfügeschwelle angepassten Langzeitfehler, der beim letzten Einfügen an dieser Stelle auftrat, durchgeführt. Hat sich der Fehler nicht verringert, kann angenommen werden, dass auch

weitere Neuronen in diesem Gebiet keine Verbesserung bewirken würden. Die Einfügeschwelle wird erhöht, was weiteres Einfügen erschwert.

Außerdem werden Regionen mit hoher Neuronendichte mit dem Ansatz der *Similarity Based Deletion* ausgedünnt. Dabei werden Neuronen gelöscht, die Nachbarn mit sehr ähnlichen Gewichten besitzen.

Ein problematischer Aspekt der LLCS ist, dass dieses Netzwerk zur Steuerung der adaptiven Lernraten, Einfügeschwellen und Löschkriterien eine große Zahl von Parametern besitzt. Voruntersuchungen in anderen Experimenten haben gezeigt, dass die Wahl dieser Parameter einen deutlichen Einfluss auf die Leistungsfähigkeit haben kann.

Da das Netzwerk in dieser Arbeit nur als Clusterer arbeiten soll, wird als Fehlerparameter im Algorithmus der Clusterfehler verarbeitet.

2.3.2. Lernverfahren

Bei allen verwendeten Lernverfahren handelt es sich um Reinforcement-Lernverfahren. Mit Hilfe von Reward (Belohnung) sollen die Agenten ein optimal an das Problem angepasstes Verhalten – die optimale Policy – lernen.

Q-Lernen

Das Q-Lernen [Watkins, 1989] ist die zur Zeit meistverwendete Methode des Reinforcement-Lernens. Dabei erlernt der Agent Q-Werte für alle Zustands-Aktions-Paare. In jedem Schritt wählt er im Standardfall die höchstbewertete, im Zustand s mögliche Aktion a , gelangt durch diese in den neuen Zustand s' und erhält den entsprechenden Reward r_t . Anschließend werden die Q-Werte nach folgenden Gleichungen adaptiert:

$$Q_{t+1}(s, a) = Q_t(s, a) + \beta \cdot \Delta Q_t(s, a)$$

$$\Delta Q_t(s, a) = r_t(s, s') + \gamma \cdot \max_{a \in A} Q_t(s', a) - Q_t(s, a)$$

Würde der Agent jedoch ausschließlich der Maximumauswahl folgen, wäre es möglich, dass nur ein kleiner Bruchteil der möglichen Aktionen ausgeführt wird; der Algorithmus würde nicht zu einer optimalen Policy konvergieren. Um dies zu vermeiden, wird das Verfahren durch eine Explorationskomponente erweitert. In dieser Arbeit wird dies durch das Addieren eines Rauschens auf den Q-Vektor des aktuellen Zustandes vor der Aktionsauswahl erreicht. Damit wird zu Beginn des Lernens ein ausgeprägtes Explorationsverhalten bewirkt. Die Amplitude des Rauschens wird mit der Zeit verringert, so dass der Agent der erlernten Policy folgt.

Q-Learning mit Activity Traces

Activity Traces können mit verschiedenen RL-Verfahren eingesetzt werden um den Lernvorgang zu beschleunigen. Das Prinzip dabei ist es, die aktuelle Änderung des Q-Wertes auch sofort auf die Aktionen aufzuschlagen, durch die man in

den aktuellen Zustand gelangt ist. Diese Werte werden zwar auch z. B. im normalen Q-Lernen auf alte Aktionen weitergereicht, aber immer nur um eine Stufe pro Trial, während mit Traces bei jedem Schritt die komplette Aktionskette rückwirkend adaptiert wird. Wie stark die momentane Änderung weitergegeben wird, lässt sich mittels des Parameters $\lambda \in [0, 1]$ steuern.

Implementiert wird das Verfahren folgendermaßen: Es wird eine Matrix eingeführt, die zu jedem Zustands-Aktions-Paar einen Activity-Wert speichert. Dem ausgewählten Zustands-Aktions-Paar im aktuellen Schritt wird ein Wert von 1 zugewiesen. Die Werte aller anderen Paare werden jeweils um den Faktor λ vermindert. Die aktuelle Änderung des Q-Wertes ΔQ wird dann für jedes Paar mit dem jeweiligen Activity-Wert multipliziert und zum Q-Wert addiert. Somit werden für $\lambda = 1$ alle verwendeten Zustands-Aktions-Paare voll adaptiert, während ein Wert von 0 dem jeweiligen Verfahren ohne Traces entspricht. Für alle anderen Werte von λ gilt, dass weiter zurückliegende Entscheidungen weniger stark adaptiert werden als kürzlich getroffene, da sie einen weniger starken Einfluss besitzen.

Wird vom Agenten eine Zufallsaktion zur Exploration ausgeführt, sind alle vorgegangenen Activity-Werte auf 0 zu setzen, da durch die Zufallsauswahl der kausale Zusammenhang der Aktionsfolge zerstört wird.

SARSA

Beim Sarsa-Lernalgorithmus nach [Rummery und Niranjan, 1994] handelt es sich um einen On-Policy-Lernalgorithmus. Während der Off-Policy-Algorithmus des Q-Lernens versucht gegen die Action-Value-Funktion der optimalen Policy zu konvergieren, folgt das Sarsa-Lernen der Action-Value-Funktion der aktuell benutzten, sich ändernden Policy.

Der wichtigste Unterschied von Sarsa- zum Q-Lernen ist Art und Weise der Adaption der Q-Werte: letzteres verwendet hier den Maximal-Wert der Q-Werte des nächsten Zustandes, wohingegen Sarsa-Lernen den Q-Wert der nächsten Aktion, die mit Hilfe der aktuellen Policy gewählt wurde, verwendet. Die Vorschrift für den neuen Q-Wert $Q(s, a)$ lautet beim Sarsa-Lernen wie folgt:

$$Q(s, a) = Q(s, a) (1 - \beta) + \beta (r + \gamma Q(s', a'))$$

Hieraus lässt sich auch der Name des Verfahrens ableiten, denn es wird für die Adaption das komplette Quintupel $Q(s, a, r, s', a')$ benötigt. Dabei sind s und a das aktuelle Zustands-Aktions-Paar und r der Reward für den Zustandsübergang zum neuen Zustands-Aktions-Paar s' und a' .

R-Lernen

Dieses Lernverfahren ist, wie das Q-Lernen, ein off-Policy-Verfahren, welches ebenfalls die Action-Value-Funktion nutzt. Der Unterschied besteht jedoch in dem, was ein Action-Value $R(s, a)$ repräsentiert, denn die R-Werte stehen für den durchschnittlichen Reward, falls in Zustand s die Aktion a gewählt und dann

der Policy gefolgt wird. Daraus ergibt sich, dass R-Lernen den durchschnittlichen Reward maximiert, wohingegen beim Q-Lernen der kumulative und mit Discount-Faktor versehene Reward maximiert wird. R-Lernen soll damit in der Lage sein, eine Policy mit langfristig besserem Durchschnittsreward zu erlernen. Nachdem zu Beginn die R-Werte auf Null gesetzt werden, wird im folgenden entweder die Aktion mit dem höchsten R-Wert oder eine Zufallsaktion zur Exploration ausgewählt. Die R-Werte werden wie folgt angepasst:

$$R_{t+1}(s, a) = R_t(s, a)(1 - \beta) + \beta (r_t(s, s') - p_t + \max_{a \in A} R_t(s', a))$$

Wie schon beim Q-Lernen ist β die Lernrate, p_t steht für den durchschnittlichen Reward der Policy π . Dieser ist für alle Zustände gleich und wird nach jeder nicht zufälligen Aktion neu geschätzt:

$$p_{t+1} = p_t (1 - \alpha) + \alpha (r_t(s, a) + \max_{a \in A} R_t(s', a) - \max_{a \in A} R_t(s, a))$$

wobei α die Lernrate ist.

Unter bestimmten Bedingungen weist dieses Verfahren laut [Mahadevan, 1994] gewisse Schwächen auf; auch ist eine Konvergenz des R-Lernens gegen die optimale Policy noch nicht bewiesen. So ist es möglich, dass dieses Verfahren aufgrund der von Mahadevan beschriebenen *Limit Cycles* vor allem in Umlernsituationen versagt. Diese *Limit Cycles* führen dazu, dass der R-Agent in Zusammenspiel mit der zufälligen Auswahl von Aktionen ein zyklisches Verhalten aufweist, welches zu Performanz-Verlust führen kann.

3. Benchmarking

3.1. Die Testumgebung

3.1.1. Die Parameter der Simulation und der Ballwelt

Die Simulationsumgebung ist durch einige Parameter gekennzeichnet, welche für alle Experimente gleich sein sollen und der Vergleichbarkeit halber mit denen aus der Diplomarbeit [Renkewitz, 2002] übereinstimmen. Diese Parameter sind die Anzahl der Schritte, der Trials und der Runs.

Ein *Schritt* besteht in der Simulationsumgebung aus einer Aktion, die der Agent in einer bestimmten Situation ausführt, und der sich anschließenden Bewertung dieser Aktion. Ein *Trial* stellt eine Sequenz von Schritten dar, deren Anzahl im konkreten Fall auf 500 begrenzt ist. Am Ende eines Trials wird der Agent mit seinem aktuellen Trainingsstand an die Startposition der Umgebung zurückgesetzt und ein neuer Trial gestartet. 100 Trials repräsentieren einen *Run*, welcher Aufschluss über die Lerngeschwindigkeit und die Güte des Agenten geben kann. Eine Mittelung über insgesamt 10 Runs soll eine allgemeine Aussage über das Verhalten der Agenten ermöglichen.

Die Ballwelt selbst ist ebenfalls von einigen Parametern gekennzeichnet, die im folgenden aufgelistet sind:

Gravitationskonstante	9.81 m/s^2
Masse des Balles	1 kg
Maximale Kraft des Agenten	4 N

3.1.2. Die Parameter der Netzwerke

Für die Größe der nicht-inkrementellen Clusterer wird eine Anzahl von 25 Neuronen festgelegt. Dieser Wert wird auch für die inkrementellen Clusterer als maximal erlaubte Neuronenzahl gewählt, um eine bessere Vergleichbarkeit der Verfahren zu erreichen. Diese Beschränkung ist akzeptabel, da 25 Neuronen für eine gute Diskretisierung des Eingaberaumes in der simulierten Umwelt ausreichen.

Die Netzwerke, die ihre Eingabegewichte anpassen (NG, GNG, LLCS) starten den Lernvorgang mit einer Lernrate von 0.5. Bei NG und GNG verringert sich dieser Wert über einen Zeitraum von 3000 Schritten auf 0.1. Die LLCS passen die Lernrate im Verlauf des Lernvorgangs lokal selbstständig an.

Für das NG wurden für alle Versuche folgende Parameter gewählt:

Lernraten-Startwert	$\eta = 0.5$
Lernraten-Endwert	$\eta = 0.1$
Intervall der Senkung der Lernrate	3000 Schritte
Lernradius-Startwert	18.75
Lernradius-Endwert	0.1
Intervall der Senkung des Lernradius	3000 Schritte
Discount-Faktor	$\gamma = 0.95$

Für die inkrementellen Netzwerke gilt, dass die Lernrate der Eingabegewichte der Nachbarn des BM-Neurons auf 1/10 der Lernrate des BM-Neurons gesetzt wird. Genau wie oben beschrieben wird auch dieser Wert beim GNG mit der Zeit verringert, während die LLCS ihn lokal anpassen.

Für das GNG wurden Netzwerkparameter festgelegt:

Lernrate Best-Matching-Neuron	$\epsilon_b = 0.5$
Lernrate Nachbarn des Best-Matching-Neurons	$\epsilon_n = 0.05$
Einfügeschrittzahl	$\lambda = 10$
Abschlagsfaktor der Fehlerzähler	$d = 0.9$
Abschlagsfaktor des Fehlerzählers beim Einfügen	$\alpha = 0.25$
Maximales Kantenalter	$\vartheta_{age} = 10$

Für den LLCS-Clusterer waren aufgrund der Parametervielfalt umfangreiche Vortests mit verschiedenen Parameterkombinationen notwendig. Diese wurden im Standard-Szenario *Welle mit Starttal ungleich Zieltal* (siehe Abschnitt 3.2.2) durchgeführt. Aus diesen Tests ergeben sich folgende Parameterwerte für die Hauptuntersuchung:

Lernrate Best-Matching-Neuron	$\eta_b = 0.5$
Lernrate Nachbarn des Best-Matching-Neurons	$\eta_n = 0.05$
Einfügeschrittzahl	$\lambda = 10$
Lernrate der Einfügeschranke	$\eta_v = 0.5$
Schrittzahl des Kurzzeitspeichers	$T_S = 10$
Schrittzahl des Langzeitspeichers	$T_L = 50$
Alterungsfaktor des Best-Matching-Neurons	$T_Y = 50$
Einfügeschranke	$T_\eta = 50$
Maximales Kantenalter	$\vartheta_{age} = 10$
Anpassungsschranke der Eingabegewichte	$\vartheta_L^i = 0.001$
Anpassungsschranke der Ausgabegewichte	$\vartheta_L^o = -0.05$
Einfügetoleranz	$\vartheta_{ins} = 0.01$
Löschschranke	$\vartheta_{del} = 0.01$
Minimales Knotenalter	$\vartheta_{delY} = 0.01$
Stabilisierungsvariable	$\vartheta_{delBL} = 0.01$

3.2. Versuchsplan

Die Versuche in dieser Arbeit sollen offene Probleme der Diplomarbeit [Renkewitz, 2002] klären, den Vergleich der Verfahren auf inkrementelle Clusterer ausdehnen und einen systematischen Benchmark durchführen. Wichtige Aspekte sind hierbei der Test einer anderen Reward-Funktion, die Kombination der Clusterverfahren mit allen verwendeten Lernverfahren und die Ermittlung des Einflusses der Traces beim Q-Lernen. Es soll getestet werden, welche Auswirkung die verschiedenen Clusterer auf das Lernen in verschiedenen Situationen haben und wie gut diese mit den unterschiedlichen Lernverfahren zusammenarbeiten.

3.2.1. Wahl der Agenten

Ein Agent zeichnet sich in dieser Arbeit durch die Kombination eines Lernverfahrens mit einem Clusterer aus. Als Lernverfahren stehen Q- und R-Lernen, so wie die Q-Lern-Variante Sarsa zur Verfügung; CMAC, NG, GNG und LLCS sind die gewählten Clusterer. An dieser Stelle spielen die Erweiterung der Versuche der Diplomarbeit und neue Problemstellungen eine Rolle. GNG und LLCS bringen eine Neuerung um den Aspekt der inkrementellen Clusterer und stehen für einen Vergleich der verschiedenen Clusterverfahren und deren Einfluss auf die Güte der Agenten zur Verfügung. Die Versuchsreihe mit variierendem Parameter λ soll über den Einfluss der Traces beim Q-Lernen Aufschluss geben. Aus diesen Gründen wurden folgende Kombinationen für die Agenten gewählt:

	R	Sarsa	Q-Learning					
			$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 0.9$
Ng	x	x	x	x	x	x	x	x
CMAC	x	x	x	x	x	x	x	x
GNG	x	x	x	x	x	x	x	x
LLCS	x	x	x	x	x	x	x	x

Für die verwendeten Lernverfahren wurden folgende Parameter festgelegt:

Initialisierung der Q-Werte 0.1
 Q/R-Lernrate $\beta = 0.1$
 p -Lernrate beim R-Lernen $\alpha = 0.1$

Des Weiteren werden alle Versuche zusätzlich mit einem Zufallsagenten durchgeführt, um die Komplexität der jeweiligen Aufgabe einzuschätzen. Damit ergeben sich 33 Agenten, mit denen die im folgenden Abschnitt beschriebenen Experimente durchzuführen sind.

3.2.2. Beschreibung der Experimente

Standard-Szenario und das Clusterer-Problem

Die Parameter, die die Umwelt und das Lernen der Agenten wesentlich bestimmt, werden in diesem Szenario wie folgt festgelegt und alle anderen Versuche ändern jeweils nur die dort explizit angegebenen Parameter.

Rewardfunktion $r = -|\text{aktuelle Position} - \text{Zielposition}|$
 Rauschen Startrauschen von 100%, Discount-Faktor 0.995
 Totzeit keine Totzeit
 Wind kein Wind
 Anzahl möglicher Aktionen 2

Als Standard- und Referenz-Szenario kommt die *Welle* zum Einsatz (Abschnitt 2.2, Abb. 2.4). Start- und Zielposition liegen im gleichen Tal bei 0.25 bzw. 0.3.

Um den Einfluss der Startposition und damit der eingesetzten Clusterer zu testen, wird das gleiche Experiment nochmals mit versetzter Zielposition (0.8) durchgeführt: der Start befindet sich im einen, das Ziel im anderen Tal der Umgebung.

Reward-Funktionen

Im Standard-Szenario mit Start- ungleich Zieltal werden verschiedene Reward-Funktionen und deren Auswirkung auf das Lern-Ergebnis getestet.

- Reward aus der aktuellen Entfernung zum Ziel
- Reward = -1 für alle Positionen außer der Umgebung ums Ziel, dort gleich 0

Die Umgebung ums Ziel wird variiert über 5, 10 und 20% der Gesamtumgebung. Alle folgenden Experimente werden mit der ersten Variante der Reward-Funktion durchgeführt, wie sie auch in der Diplomarbeit [Renkewitz, 2002] genutzt wurde.

Verschiedene Gebirge

Hier werden die in der Diplomarbeit verwendeten Umgebungen, *Ebene*, *Welle*, *Wanne* und *Berg* getestet. Ziel dieses Experimentes ist es, die schon gewonnenen Erkenntnisse um die neuen Clusterer und die erweiterten Lernverfahren zu ergänzen.

Verschiedenes Rauschen

Der Test zum Verhalten bei unterschiedlichen Eigenschaften des Rauschens wird im Standard-Szenario mit unterschiedlichem Start- und Zieltal durchgeführt. Rauschen zu Beginn der Trainingsphase erhöht die Explorationsfreudigkeit der Agenten und ist deshalb für ein effektives Lernen nötig. Folgendes Rauschverhalten wurde untersucht:

- 100% Rauschen zu Beginn, Discount-Faktor pro Lernschritt 0.995 (Standard-Agent)
- 100% Rauschen zu Beginn, Discount-Faktor pro Lernschritt 0.7
- 50% Rauschen zu Beginn, Discount-Faktor pro Lernschritt 0.995
- 50% Rauschen zu Beginn, Discount-Faktor pro Lernschritt 0.7
- 50% Rauschen während des gesamten Experimentes

Vorhandensein von Totzeiten

Bei Realwelt-Problemen können Totzeiten vorherrschen, die zu Verzögerungen seitens der Ein- oder Ausgabe führen. Die hier verwendete Simulation einer motorischen Totzeit hat zur Folge, dass es zu einer zeitlichen Verzögerung zwischen Auswahl einer Aktion und deren Ausführung kommt. Renkewitz hielt in seiner Diplomarbeit [Renkewitz, 2002] fest, dass die Agenten unabhängig von der Totzeit in der Lage sein müssten, die Abbildung von einer Situation auf die zu wählende Aktion zu lernen; ggf. benötigen die Agenten jedoch länger bis zur Konvergenz. Die Experimente in dieser Arbeit werden im Standard-Szenario mit Totzeiten von 0, 1, 2, 3, 4 und 5 Takten durchgeführt.

Veränderliche Zielstellung

Bei Problemen der Realwelt kann es vorkommen, dass sich die Zielstellung ändert. Deshalb soll mit diesem Experiment die Fähigkeit der Agenten zum Umlernen überprüft werden. Darüber hinaus ist es aber auch wichtig, bereits Gelerntes nicht wieder zu vergessen (Stabilitäts-Plastizitäts-Dilemma).

Wieder wird hier auf das Standard-Szenario *Welle* zurückgegriffen, wobei Start- und Zielposition über einen Run von 300 Trials wie folgt variiert werden:

- Beginn mit $Start = 0.25$ und $Ziel = 0.75$, Wechsel zu $Start = 0.8$ und $Ziel = 0.75$, Rückkehr zu $Start = 0.25$ und $Ziel = 0.75$
- Beginn mit $Start = 0.25$ und $Ziel = 0.3$, Wechsel zu $Start = 0.25$ und $Ziel = 0.75$, Rückkehr zu $Start = 0.25$ und $Ziel = 0.3$
- Beginn mit $Start = 0.25$ und $Ziel = 0.75$, Wechsel zu $Start = 0.75$ und $Ziel = 0.25$, Rückkehr zu $Start = 0.25$ und $Ziel = 0.75$ zurück.

Teilweise beobachtbare Umwelt - Störgröße Wind

Ein wichtiger Aspekt der Realwelt ist die nur teilweise Beobachtbarkeit (POMDP) der aktuellen Situation. Das bedeutet, dass nicht alle Einflüsse für den Agent bekannt oder erlernbar sind. Um dies zu simulieren, wird in die Umwelt ein zufälliger Wind eingebracht, dessen Stärke in Newton sich im Intervall $[-w, w]$ bewegt. Im Standard-Szenario werden Versuche mit Windstärken $w \in [0 N, 2 N, 4 N, 8 N]$ durchgeführt.

Anzahl der möglichen Aktionen des Agenten

Dem Agenten stehen für die Bewegung des Balls in der Umgebung verschiedene Aktionen zu Verfügung. Standardmäßig soll der Agent mit zwei Aktionen nur die Möglichkeit haben, den Ball entweder mit voller Kraft nach rechts oder links zu bewegen. Wird die Anzahl der Aktionen für den Agent erhöht, kann er die Kraft abgestuft einsetzen und bei einer ungeraden Anzahl von Aktionen steht auch eine Aktion zum „Nichts tun“ zur Verfügung. Die Experimente in dieser Arbeit finden mit 2, 3, 4, 5 und 8 Aktionen statt und sollen Aufschluss darüber geben, ob dem Agenten eine Verfeinerung der Aktionen hilft oder das Lernen erschwert bzw. verlangsamt.

Kombination von Teilproblemen

Der Versuch soll zeigen, ob die Fähigkeit der Agenten, ein Problem zu lösen, durch das Vorhandensein einer weiteren Störung beeinflusst wird. Dies wird im Standard-Szenario mit einer Totzeit von zwei Takten und einer Windstärke von 2 simuliert.

4. Testergebnisse und Auswertung

Dieses Kapitel befasst sich mit der Auswertung und Darstellung der Testergebnisse. Zu diesem Zweck werden verschiedene Grafiken verwendet. Der Reward ist immer kleiner eins und entspricht dem Abstand des Balles zum Ziel, je näher der Reward der Null kommt, desto besser ist der Agent. Es kommen im wesentlichen drei Typen von Grafiken zum Einsatz:

Durchschnittsreward Dieser Plot beinhaltet meist alle Agenten, gruppiert nach dem Clusterer, und stellt in einem Balkendiagramm stets den durchschnittlichen Reward jedes Agenten gemittelt über das gesamte Experiment dar. Damit wird klar, dass in der Höhe des Balkens sowohl Lerngeschwindigkeit, also auch die letztendliche Performanz kodiert ist. Ein kurzer Balken steht somit für einen Agenten, der sowohl schnell lernt, also auch das erlernte gute Verhalten über die Zeit aufrechterhält. Der vom Zufallsagenten (RND-Agent) erzielte Reward wird zum Vergleich und als Maß der relativen Schwierigkeit des Experiments zusätzlich erwähnt.

Entwicklung des mittleren Rewards Diese Grafik verdeutlicht den Lernverlauf der jeweiligen Agenten über einen Run. Hierzu wurde der Durchschnitt aus 10 Runs gebildet um zufällige Einflüsse einzuschränken. Dargestellt ist jeweils der mittlere Reward pro Trial.

Ball-Position Dieser Plot beschreibt die Bewegung des Balles in der Umgebung. In einigen Fällen kann dies nötig sein, um gewisse Aspekte zu veranschaulichen oder zu unterstützen. Es wird für diese Grafik jeweils der letzte Trial im letzten Run der Trainingsphase ausgewählt, so dass der Verlauf als Beispiel für einen trainierten Agenten steht.

4.1. Standard-Szenario und das Clusterer-Problem

Das Standard-Szenario stellt die einfachste von den Agenten zu erfüllende Aufgabe dar. Sie starten bereits in der Nähe des Ziels und könnten nur mit Schwung holen aus dem Zieltal entkommen. Allerdings befindet sich der Zielpunkt an der Steigung des Hanges, so dass für einen möglichst hohen Reward ein Ankämpfen gegen die Schwerkraft nötig ist.

Zwischen den verschiedenen Clusterern und Lernverfahren sind kaum Unterschiede in der Performanz zu erkennen (Abb. 4.1). Das bei den Traces-Agenten erwartete Verhalten eines schnelleren Lernens tritt hier nur bei der Kombination mit

dem NG als Clusterer deutlich zutage. Alle anderen Clusterer zeigen sich nur von einem hohen Wert des Trace-Parameters wie $\lambda \geq 0.8$ beeinflusst, hier jedoch negativ, was dafür spricht, dass der Aufwand der Traces bei diesem einfachen Problem zu einer längeren Lernphase führt, aber sonst keine Vorteile bringt. Im allgemeinen kann diese Aufgabe aber von allen Agenten hervorragend gelöst werden, wobei die R-Agenten eine größere Varianz des mittleren Rewards über eine Trial aufweisen und damit die schlechteste Performanz zeigen.

Die guten Ergebnisse aller Agenten lassen sich damit begründen, dass bereits im Zieltal gestartet wird und damit die Clusterung einfach ist. Das sehr gute Verhalten auch bei den CMAC-Agenten, der eine starre, vorgegebene Clusterung besitzt, spricht dafür, dass auch das Erlernen des richtigen Verhaltens keine schwierige Aufgabe darstellt und auch mit wenigen Neuronen im Zielgebiet sehr gut zu lösen ist.

Die Verlagerung des Zieles in das andere Tal der Welle bringt für die Agenten die Notwendigkeit mit sich, ein „Schwung holen“ zu erlernen, da die Kraft nicht ausreicht, den Ball direkt über den Berg zu rollen. Im Zieltal muss der Ball wieder wie im ersten Teil am Hang gehalten werden. Dieser Versuch soll deutlich machen, inwiefern die Wahl des Clusterers einen bedeutenden Einfluss auf den Lernerfolg hat. Ein wichtiger Unterschied zwischen den Clusterern ist hierbei die Variabilität: NG, GNG, LLCS können eine beliebige, der Aufgabe angepasste Clusterung des Inputraumes vornehmen, wohingegen dem CMAC bereits eine starre Clusterung vorgegeben ist (Abb. 2.6).

Das Erlernen von adäquatem Verhalten erfolgt hier wesentlich langsamer und der positive Einfluss der Traces zeigt sich deutlich; auch zwischen den verschiedenen Clusterern sind Unterschiede zu erkennen (Abb. 4.2). Bei den LLCS ist eine extrem starke Varianz in den Ergebnissen zu beobachten. Während einige Runs gute bis sehr gute Ergebnisse zeigen, können andere nur als erfolglos bezeichnet werden. Die CMAC-Agenten zeigen hingegen mit Q-Lernen und der Sarsa-Variante gute Ergebnisse, wobei aber ein Schwanken des mittleren Rewards über die Trials zu beobachten ist (Abb. 4.3(a)). Dies ist möglicherweise auf die starre Clusterung durch den CMAC zurückzuführen. Damit verbunden könnten auch die deutlichen Probleme der CMAC-Agenten bei niedrigen Traces sein; so ist ein den anderen Agenten qualitativ ebenwürdiges Verhalten erst bei Werten von $\lambda = 0.8$ oder 0.9 zu erkennen (Abb. 4.3(b)).

Die besten Ergebnisse dieses Versuches liefern die NG- und GNG-Agenten, wobei letztere ein etwas schlechteres Gesamtergebnis zeigen, was vermutlich dem zu Beginn stattfindenden Einfügen von neuen Neuronen geschuldet ist, während da NG schon am Anfang alle Neuronen zur Verfügung hat. Für beide Clusterer kann jedoch festgestellt werden, dass mit ihrer Hilfe die Agenten das nötige Verhalten stabil und sicher erlernen, was z.B. bei den LLCS ein Problem darstellt. Der Einfluss der Traces tritt bei den NG-Agenten (Abb. 4.4) deutlicher hervor als bei den GNG-Agenten, ist aber auch hier vorhanden und führt zu einem schnelleren Lernen. Auch die Sarsa-Agenten zeigen mit dem NG ein minimal besseres Ver-

4. Testergebnisse und Auswertung

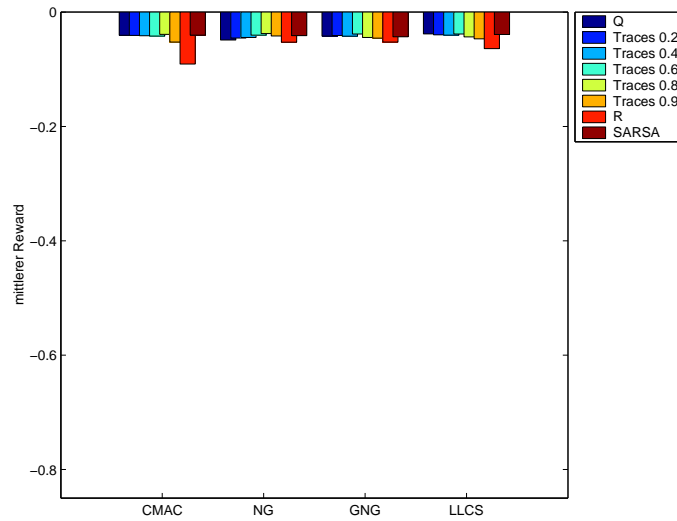


Abb. 4.1: Mittlerer Reward aller Agenten im Szenario Welle mit Starttal = Zieltal(Standard-Szenario) (RND-Agent: -0.2855)

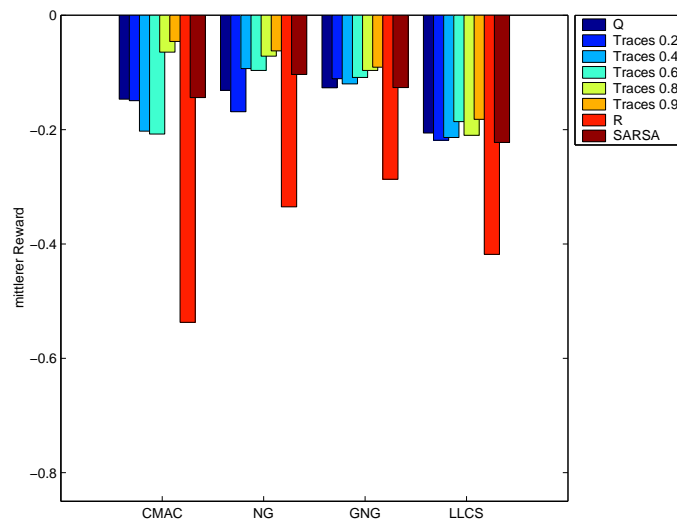


Abb. 4.2: Mittlerer Reward aller Agenten im Szenario Welle mit Starttal \neq Zieltal (RND-Agent: -0.4094)

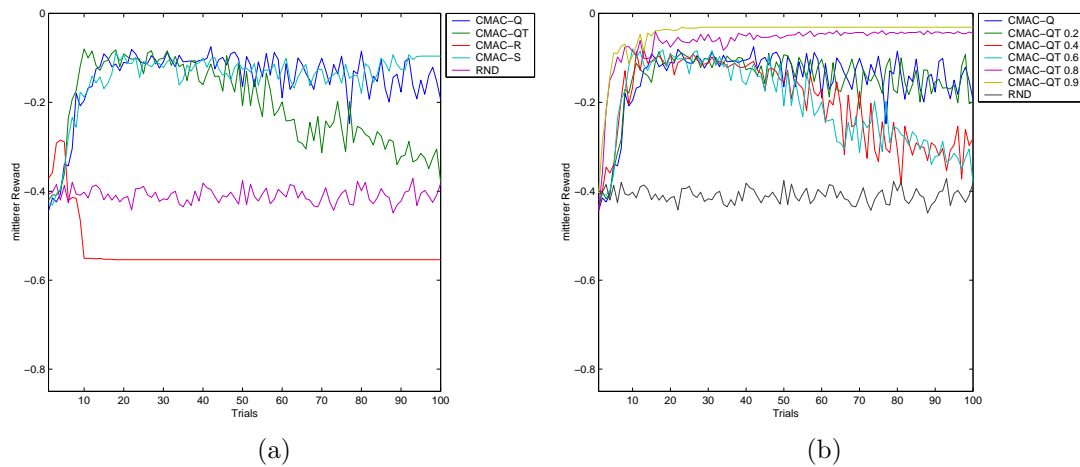


Abb. 4.3: CMAC-Agenten (a) und CMAC-Traces-Agenten (b) im Standard-Szenario mit Starttal \neq Zieltal, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

halten als bei CMAC und GNG. R-Agenten versagen hingegen in Kombination mit allen Clusterern und erreichen lediglich die Performanz eines Zufallsagenten. Da dieses schlechte Ergebnis bei jedem Clusterer auftritt, muss der Grund für das Versagen beim Lernverfahren selbst liegen; vermutlich verhindern die beim R-Lernen bekannten Limit Cycles das Erlernen des gewünschten Verhaltens. Eine variable Clusterung wie durch NG und GNG scheint für dieses Problem von Vorteil zu sein. Die CMAC-Agenten mit ihrer starren Clusterung benötigen einen großen Wert des Trace-Parameters um Leistungsschwankungen zu vermeiden. LLCS zeigen trotz einer variablen Clusterung nur eine mittelmäßige Performanz, was darauf hindeutet, dass dieser Clusterer in Kombination mit allen Lernverfahren deutlich Probleme zu haben scheint, wenn das Ziel des Balles nicht im Starttal liegt. Bezüglich dem Einsatz von inkrementellen oder nicht-inkrementellen Clusterern kann keine Aussage getroffen werden, da sowohl einmal die LLCS und einmal der CMAC schlechtere Ergebnisse zeigen. Die beste Wahl scheint jedoch ein Neuronales Gas in Form von einfachem NG oder GNG zu sein.

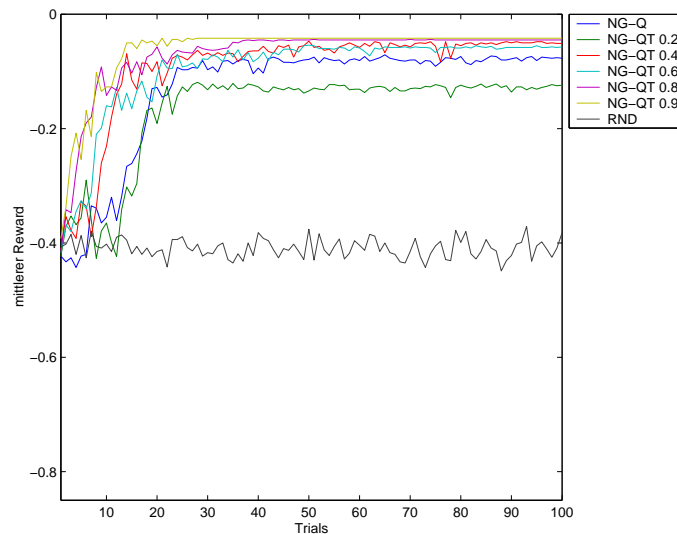


Abb. 4.4: NG-Traces-Agenten ($\lambda = 0.6$) im Standard-Szenario mit Starttal \neq Zieltal, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

4.2. Verschiedene Reward-Funktionen

Die Verwendung einer veränderten Rewardfunktion, bei der der Agent erst dann einen Reward ungleich -1 bekommt, wenn er das Zieltal erreicht hat, soll ebenfalls einen Aspekt realer Einsatzgebiete simulieren. Oftmals kann keine kontinuierlich abgestufte Rewardfunktion, wie es das sonst in dieser Arbeit verwendete Abstandsmaß ist, verwendet werden, sondern ein Reward wird nur selten oder nur einmalig erst nach dem erfolgreichen Erfüllen einer Aufgabe vergeben.

Bei dem simulierten Szenario gibt es damit zwei Schwierigkeiten für die Agenten: Zum einen bekommen sie erst dann eine Erfolgsmeldung, wenn sie den Ball zunächst einmal zufällig ins Zieltal bringen können. Zum anderen ist es für die Agenten, je schmaler der Rewardbereich wird, zunehmend unmöglich den Ball ständig in dieser Zone zu halten, da sie dem Anstieg der Welle am Zielpunkt nicht die exakt benötigte Kraft entgegengesetzen können um den Ball am Zielpunkt zu halten. Durch diese Probleme kann man auch den durchschnittlichen Reward der Agenten in diesem Experiment nicht direkt mit dem aus den anderen Szenarien vergleichen. Hier entspricht dieser eher dem prozentualen Anteil der Steps, die der Agent den Ball in der Zielzone halten konnte.

Die Performanz der Agenten bei einem verzögerten Reward und einem Ziel mit einer Breite von 0.2 ist mit Ausnahme der LLCS- und R-Agenten hervorragend; die Leistung der trainierten Agenten am Ende der Tests ist vergleichbar mit der der Standardrewardfunktion (Abb. 4.7).

Die R-Agenten versagen schon in dieser Variante mit dem breitesten Zielbereich

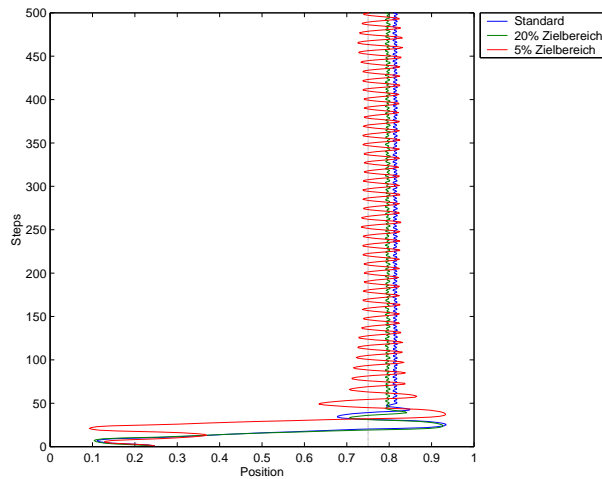


Abb. 4.5: Bewegung des Balles um die Zielposition bei Verwendung verschiedener Rewardfunktionen am Beispiel des GNG-Q-Traces-Agenten ($\lambda = 0.9$)

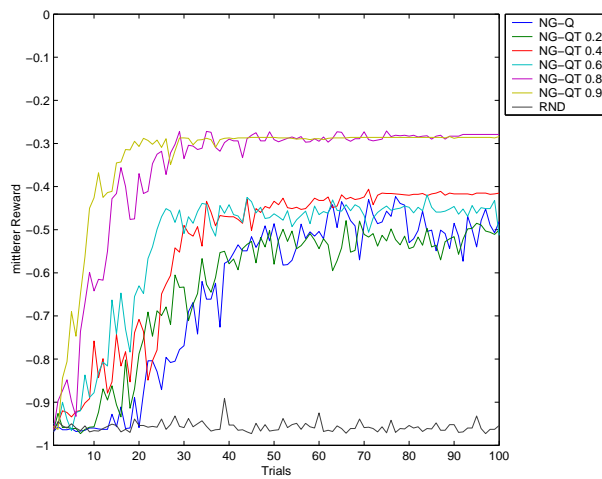


Abb. 4.6: Einfluss der Traces bei alternativer Rewardfunktion (10% Zielbereich) am Beispiel der NG-Q-Agenten, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

komplett und erreichen lediglich die Güte eines Zufallsagenten. Lediglich der GNG-R-Agent kann noch in Ansätzen zeigen, dass zumindest der Versuch des Lernens unternommen wird, die Lerngeschwindigkeit ist aber viel zu gering.

Auch die LLCs-Agenten erleben einen stärkeren Leistungseinbruch. Unregelmäßige Bewegungen des Balles durch die in der Anfangsphase weniger adaptierten Q-Werte stören den ohnehin instabileren Cluster noch zusätzlich. Wie in allen schwierigen Szenarien zeigen die LLCs-Agenten in einzelnen Runs, dass sie in der Lage sind das Problem zu lösen, aber die Schwankungen treten aus den genannten Gründen häufiger zutage. Dies gilt auch bei der Steuerung im Zieltal, bei dem die LLCs-Agenten teilweise größere Schwankungsbewegungen des Balles zeigen. Dies schlägt sich bei kleinen Rewardbereichen natürlich umso deutlicher in einem geringeren Reward nieder, da der Zielbereich häufiger verlassen wird.

Deutlich ist in diesem Szenario ein Vorteil der Q-Traces-Agenten zu erkennen, der umso größer wird, je höher der Wert des Traces-Parameter gewählt ist (Abb. 4.6). Die Adaption der Q-Werte dauert mit der alternativen Rewardfunktion deutlich länger, da nur selten nicht-negative Rewards vergeben werden. Durch die Verwendung von Activity Traces werden diese jedoch sofort auf die gesamte durchlaufene Kette von Zustands-Aktions-Paaren verteilt. Die Verringerung der Lerngeschwindigkeit kann damit fast komplett kompensiert werden.

Wie zu erwarten, gilt: Je schmaler das Zielgebiet wird, desto geringer fällt der durchschnittliche Reward aus (Abb. 4.8 und 4.9). Dies liegt aber fast ausschließlich am zweiten zu Beginn erwähnten Problem: den Ball in dem schmalen Zielbereich zu halten. Die erfolgreichen Agenten sind auch mit kleinerem Rewardfenster weiterhin zuverlässig in der Lage das Zieltal zu erreichen (Abb. 4.5). Man erkennt auch, dass die Agenten den Ball im Versuch mit 5% Rewardbereich weiter schwanken lassen. Dies ist aber zwingend nötig, um den Ball nah ans Ziel zu bringen. Der Zielbereich reicht nur noch von 0.725 bis 0.775 und nur durch das leichte Schwungholen kann dieser noch erreicht werden. Die erfolgreichen Agenten passen sich also sogar dieser Problemstellung gut an.

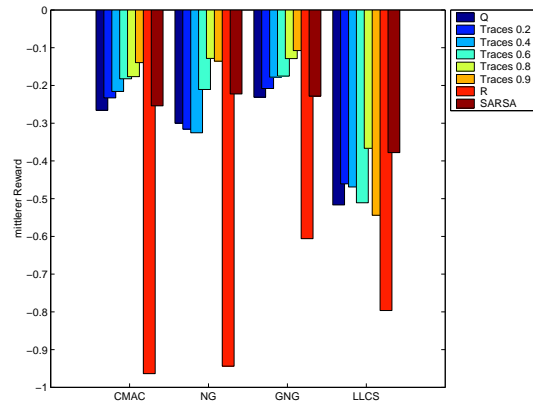


Abb. 4.7: Mittlerer Reward der Agenten bei einer Breite des Zielbereichs von 20% der Ausdehnung des Szenarios, (RND-Agent: -0.9079)

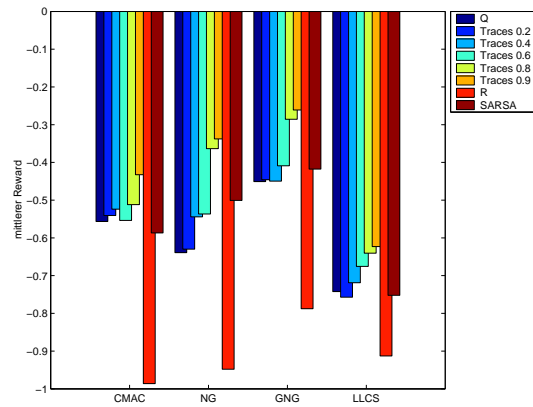


Abb. 4.8: Mittlerer Reward der Agenten bei einer Breite des Zielbereichs von 10% der Ausdehnung des Szenarios, (RND-Agent: -0.9574)

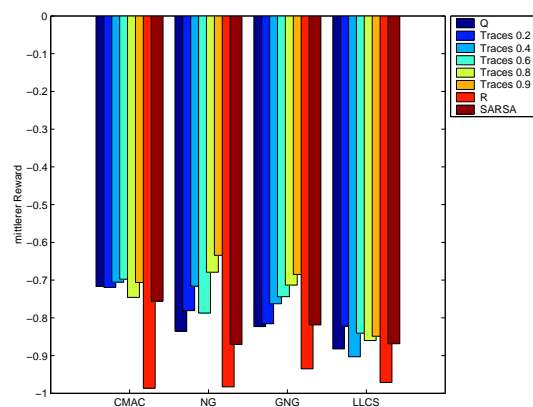


Abb. 4.9: Mittlerer Reward der Agenten bei einer Breite des Zielbereichs von 5% der Ausdehnung des Szenarios, (RND-Agent: -0.9778)

4.3. Verschiedene Gebirge

Ebene

Dieses einfache Szenario stellt an die Agenten nur den Anspruch, den Eingaberaum so zu clustern, dass der Ball möglichst nahe am Ziel bewegt werden kann.

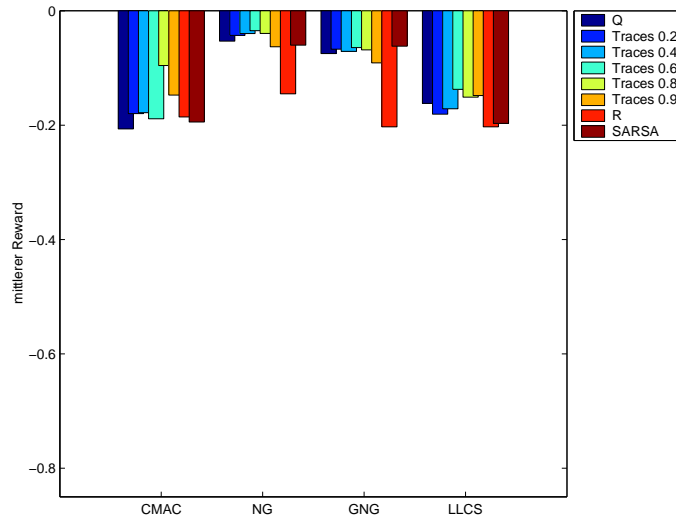


Abb. 4.10: Mittlerer Reward der Agenten in der Ebene, (RND-Agent: -0.3843)

In Abbildung 4.10 werden folgende Ergebnisse deutlich: GNG und NG liefern für die Agenten eine Clusterung, die zu sehr guten Ergebnissen führt, die zum Teil besser sind, als die im Standard-Szenario. Die LLCS-Agenten zeigen ein schlechteres Verhalten als die anderen Clusterer: einige Runs sind sehr erfolgreich, in anderen steuert der Agent den Ball gegen den rechten Rand, wo er verbleibt. Bei den CMAC-Agenten ist ein schlechteres Verhalten als bei NG oder GNG zu beobachten.

Der Grund für das schlechtere Verhalten der CMAC-Agenten liegt wohl bei der starren Clusterung durch den CMAC. Wie in Abb. 2.6 zu erkennen, liegen bei Position 0.75 und 1.0 Neuronen des CMAC, das Ziel jedoch liegt bei 0.8. Damit wird den CMAC-Agenten das halten des Balles an der Zielposition erschwert. Abbildung 4.11 verdeutlicht den Unterschied der Ballposition beim CMAC- und NG-Agenten mit Q-Traces-Lernverfahren: es ist zu erkennen, dass der NG-Agent viel näher am Ziel halten kann und damit einen besseren mittleren Reward für diesen Trial erhält. Ein weiterer Nachteil, den CMAC Agenten gegenüber einem Neuronalen Gas als Clusterer aufweisen, ist die Tatsache, dass es diesen möglich ist, mehrere Neuronen um das Zielgebiet zu platzieren und damit dieses besser mit adäquaten Aktionen auszustatten.

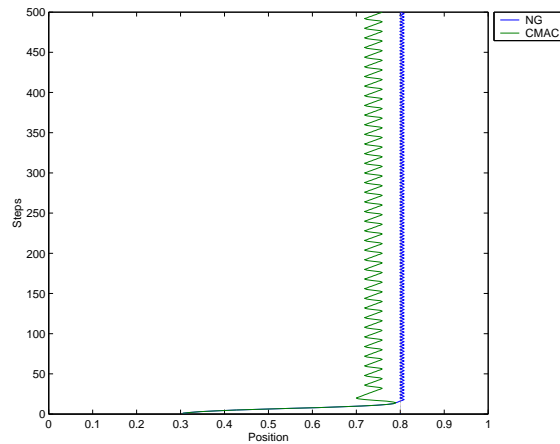


Abb. 4.11: Vergleich der Ballposition von CMAC- und NG-basierten Agenten in der Ebene, Q-Traces-Agenten mit $\lambda = 0.4$.

Beim R-Lernen zeigen sich deutlich Schwächen, wobei NG und GNG als Clusterer noch deutlich zur einer besseren Performanz beitragen. Das Q-Lernen erweist sich in diesem Szenario in all seinen Ausprägungen (Traces, Sarsa) als gute Wahl. Allgemein sind die Traces hier als wichtig einzuschätzen, da alle Clusterer mit höherem Trace-Parameter bessere Ergebnisse erreichen und lernen schneller. Bei den Sarsa-Agenten entstehen zwei Leistungsklassen: NG und GNG erreichen hervorragende Ergebnisse, LLCS und CMAC sind hier wesentlich schlechter (Abb. 4.13).

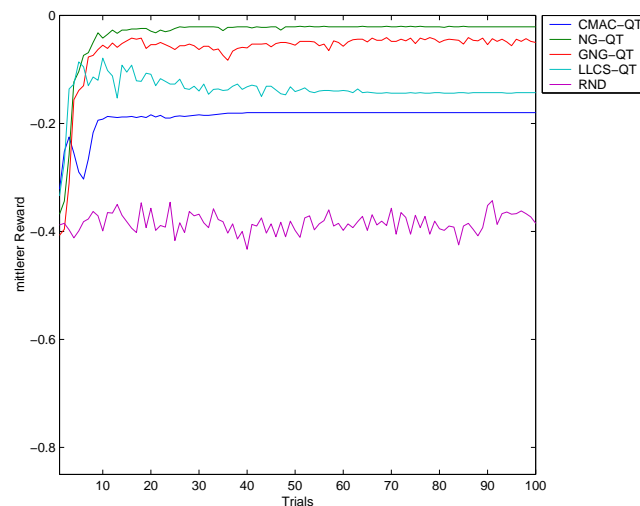


Abb. 4.12: Q-Traces-Agenten in der Ebene, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

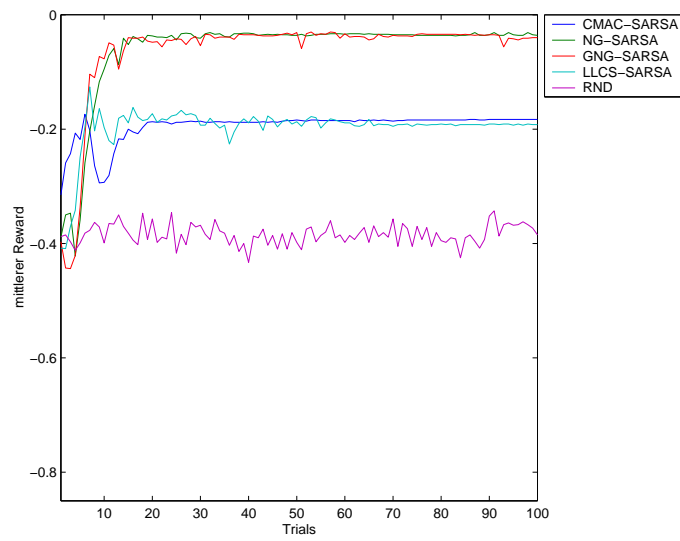


Abb. 4.13: Q-Sarsa-Agenten in der Ebene, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Wanne

Dieses Szenario ähnelt dem der Ebene, jedoch müssen die Agenten hier der Schwerkraft entgegen wirken und den Ball am Hang halten und sind deshalb etwas schlechter (bis auf die Ausnahme der CMAC-Agenten), da es ihnen nicht möglich ist, die Kraft für das Halten exakt zu dosieren.

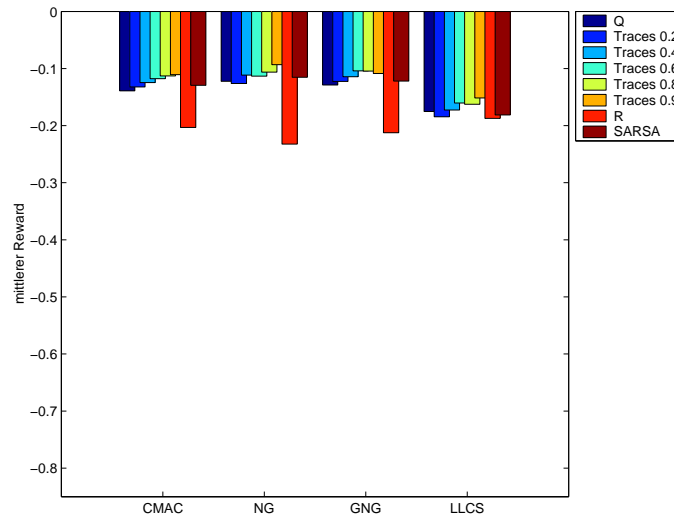


Abb. 4.14: Mittlerer Reward der Agenten im Szenario Wanne, (RND-Agent: -0.3379)

Auch hier zeigt sich, wie schon in der Ebene, eine leichte Überlegenheit der NG- und GNG-Agenten, wobei die CMAC-Agenten hier wesentlich besser mithalten als in der Ebene. (Abb. 4.15). Dies liegt vermutlich daran, dass die Platzierung der Neuronen des CMAC besser für die Wanne als für die Ebene geeignet ist.

Die Sarsa-Agenten lernen hier oftmals schneller als die einfachen Q-Agenten, wobei dies, abgesehen von dem schlechteren LLCS und dem schnellen NG, unabhängig vom Clusterer ist. Besser noch zeigen sich nur die Q-Traces-Agenten; hier ist ein größerer Trace-Parameter für ein schnelleres Lernen verantwortlich (Abb. 4.16), die End-Performanz ändert sich dabei aber kaum. Auf die GNG-Agenten scheinen die Traces hier einen leicht negativen Einfluss zu haben: die End-Performanz ist von hoher Varianz über das Experiment gekennzeichnet, kann jedoch dennoch als gut bezeichnet werden.

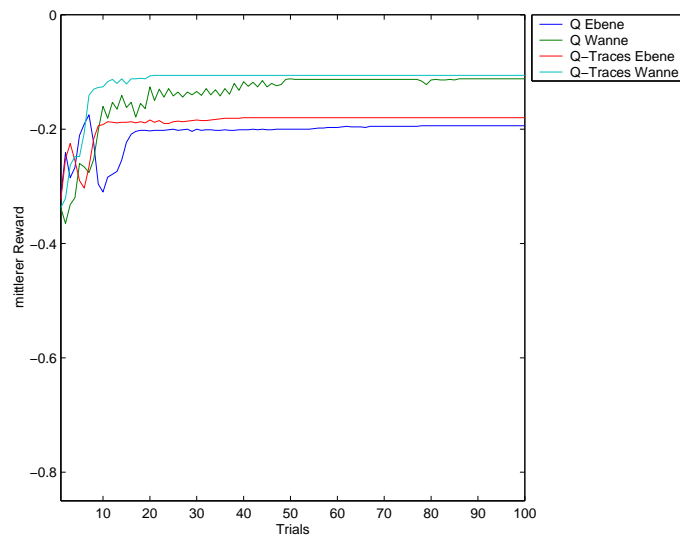


Abb. 4.15: CMAC-Agenten zum Vergleich in der Ebene und der Wanne, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments. Die Q-Traces-Agenten besitzen hier ein $\lambda = 0.6$.

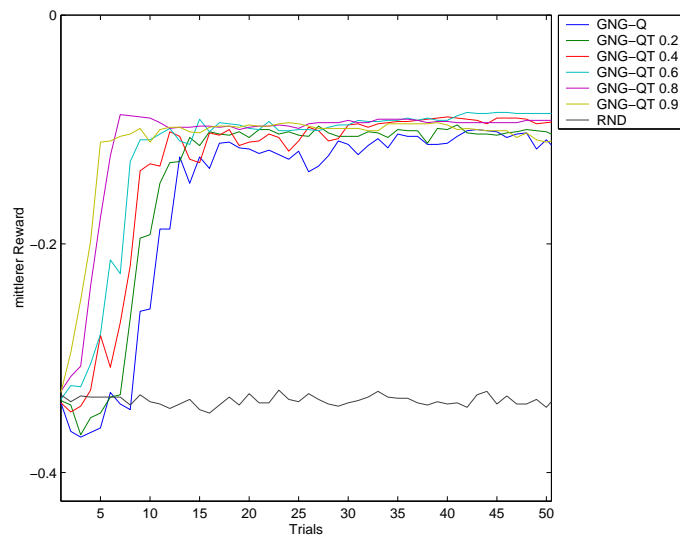


Abb. 4.16: GNG-Traces im Szenario Wanne, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments.

Die R-Agenten haben auch hier Probleme, sind aber in der Lage in Kombination mit einzelnen Clustern gute Ergebnisse zu erreichen. Der GNG-R-Agent z. B. lernt zwar langsam, erreicht aber die End-Performanz der anderen GNG-Agenten. Das gelernte Verhalten des NG-R-Agenten hingegen beschränkt sich darauf, den Ball von einer Seite zur andern zu rollen; dem CMAC-R-Agent ist es zumindest möglich, sich in der Zielgegend aufzuhalten.

Die LLCS-Agenten erreichen im Schnitt leicht bessere Ergebnisse als im Wellen-Experiment, hervorgerufen durch weniger erfolglose Runs. Dies lässt vermuten, dass der Clusterer Probleme mit dem Platzieren der Neuronen zum Schwungholen in der Welle hat; möglicherweise werden die nah beieinanderliegenden Neuronen durch den Similarity-Deletion-Ansatz zu oft gelöscht.

Berg

Das Szenario Berg ist die schwierigste Variante der untersuchten Gebirge. Die Kurve ist in der Nähe des Ziels so steil, dass der Agent diesen Punkt mangels Kraft nicht erreichen und auch im Falle des Abrutschens am linken Rand dieses Tal nicht mehr verlassen kann. Die minimal mögliche Annäherung zum Ziel ist ein Abstand von ca. 0.2, womit der vergebene Reward in diesem Experiment deutlich geringer als in den anderen Tests ist. Wenn der Agent versucht Schwung zu holen um dem Zielpunkt nahe zu kommen, wird dies in der Folge zum Abrutschen am linken Rand führen. Die Strategie für den höchsten durchschnittlichen Reward ist es also, im lokalen Minimum der Kurve im Bereich von 0.4 bis 0.5 zu bleiben.

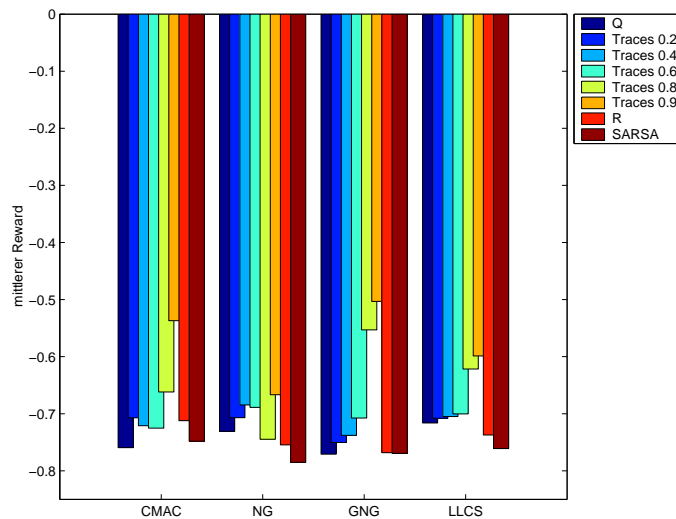


Abb. 4.17: Mittlerer Reward der Agenten im Szenario Berg, (RND-Agent: -0.7830)

Die Agenten haben große Schwierigkeiten diese Aufgabe zufriedenstellend zu lösen. Dies bedeutet vor allem, dass nicht sichergestellt ist, dass der Agent im größten Teil aller Runs die Strategie lernt. Es ist zu beobachten, dass es den Agenten zwar möglich ist, den Ball in der Nähe des Ziels zu halten, sie dieses Verhalten jedoch nur in einem Bruchteil der Fälle sicher erlernen. Bei der Schwierigkeit des Szenarios ist dies nicht überraschend, da eine falsche Zufallsentscheidung zur Exploration zu Beginn eines Trials, die in den linken Abgrund führt, schon nachhaltigen Einfluss auf die Leistung des Agenten haben kann.

Als teilweise erfolgreich kann man dabei nur die Q-Traces Agenten bezeichnen. Die CMAC- und GNG-Agenten mit diesem Lernverfahren erlernen das richtige Verhalten häufiger, aber erst bei großem Wert des Trace-Parameters ($\lambda = 0.9$) kann man sagen, dass sie das Problem lösen können, da sie ein deutliches Über-

gewicht an erfolgreichen Versuchen erreichen (Abb. 4.18). Dennoch können auch sie einzelne Fehlversuche nicht verhindern.

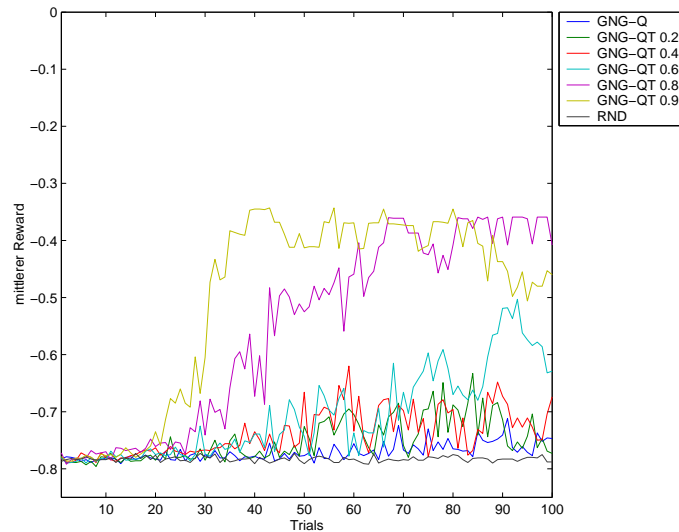


Abb. 4.18: Einfluss der Traces im Szenario Berg am Beispiel der GNG-Q-Agenten, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

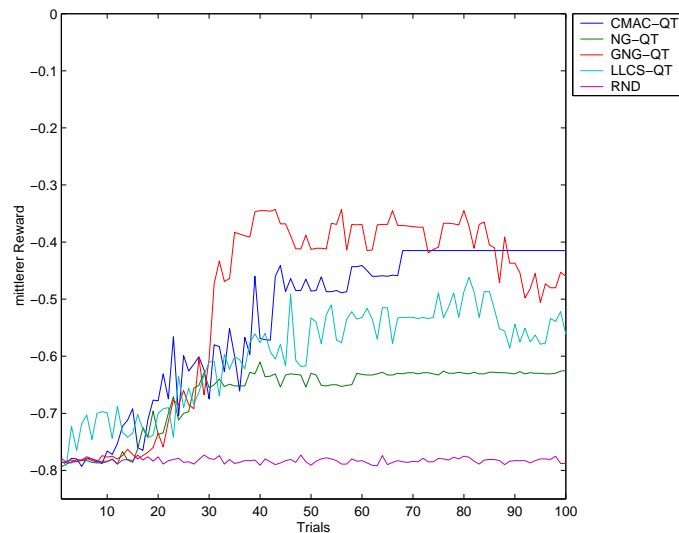


Abb. 4.19: Vergleich der Q-Traces-Agenten ($\lambda = 0.9$) mit den verschiedenen Clustern im Szenario Berg, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

NG- und LLCS-basierte Agenten zeigen ein schlechteres Verhalten. Auch sie schaffen einzelne erfolgreiche Runs, aber deutlich weniger häufig als die anderen Agenten. Die LLCS-Agenten sind dabei sogar noch ein wenig erfolgreicher. Dies könnte auch daran liegen, dass das praktische Ziel im lokalen Minimum der Kurve recht nah am Startpunkt liegt, und Agenten mit diesem Clusterer über die gesamte Testreihe bessere Ergebnisse zeigen, wenn Start und Ziel nahe beieinander liegen.

Das extrem schlechte Ergebnis der NG-Agenten könnte darauf zurückzuführen sein, dass, wenn der Ball zu Beginn des Tests ins linke Tal abrutscht, was während der Explorationsphase mit an Sicherheit grenzender Wahrscheinlichkeit passiert, alle Neuronen auf diese Position mitgezogen werden, da der Lernradius zu diesem Zeitpunkt noch groß ist. Dem Clusterer fällt es schwer später genügend Neuronen auf andere Positionen in der Umwelt zu verteilen. Es findet auch keine ordentliche Adaption der Q-Werte statt, da sich beim Festsitzen im Tal, egal welche Aktion ausgeführt wird, weder Position, noch Geschwindigkeit, noch der vergebene Reward ändern. Dadurch wäre auch das relativ dazu bessere Ergebnis der CMAC-Agenten erklärt, denn da dort keine Gewichte angepasst werden, können die Neuronen auch nicht in das Tal mitgezogen werden. Auch GNG- und LLCS-Agenten leiden unter diesem Effekt weniger als das NG, weil zum einen nicht alle, sondern nur einige miteinander verbundene Neuronen ins Tal mitgezogen werden, zum anderen können sie Neuronen an andere Positionen später neu einfügen.

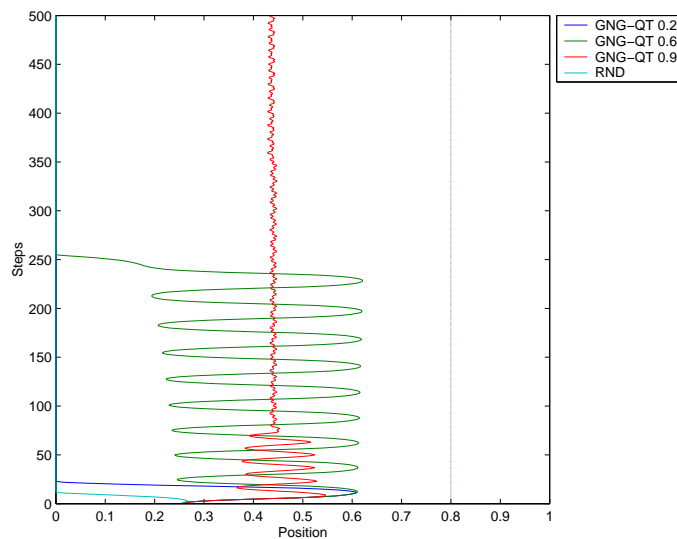


Abb. 4.20: Bewegung des Balles im Szenario Berg unter Verwendung verschiedener GNG-Q-Traces Agenten im Vergleich zum Zufallsagent

Die Gründe für den deutlichen Einfluss der Traces sind, dass es mit größerem Wert des Trace-Parameters den Agenten leichter fällt, den Zusammenhang zwischen zu viel Schwung in Richtung Ziel und dem danach folgenden Abrutschen links besser zu erkennen (Abb. 4.20), zum anderen das Erlernen der richtigen Strategie deutlich länger als in den anderen Szenarien dauert. Ein Grund dafür ist auch, dass während der Laufzeit der Tests nur relativ schlechte Rewards vergeben werden, da die Zielposition nicht erreicht werden kann. Insgesamt lässt sich erkennen, dass Activity-Traces nicht nur die Lerngeschwindigkeit erhöhen, sondern auch die Qualität der erlernten Policy verbessern, da sie helfen, zeitlich getrennte kausale Zusammenhänge besser zu erkennen.

R- und Sarsa-Lernverfahren versagen völlig, bei letzterem sollte aber eine Leistungsverbesserung durch Verwendung von Activity Traces zu erwarten sein, da auch Q-Learning ohne Traces kaum bessere Ergebnisse zeigt.

4.4. Verschiedenes Rauschen

In diesem Szenario müssen die Agenten bei verschiedenen Rauschfaktoren, womit unterschiedliches Explorationsverhalten erreicht werden soll, den Ball vom Start ins Zieltal bewegen.

Die verschiedenen Werte für das Rauschen bei der Aktionsauswahl haben nur relativ geringen Einfluss auf die Performanz der Agenten, die ähnliche Ergebnisse wie im Standardexperiment (Rauschen 1.0, Discount-Faktor 0.995) zeigen (Anhang A, Abb. A.2).

Wie zu erwarten ist der Gesamterward bei einem leichten, aber nicht im Verlauf des Experimentes reduzierten Rauschen geringer, da die Aktionsauswahl ständig gestört wird (Abb. 4.21).

Insgesamt lässt sich vermuten, dass also ein geringes initiales Rauschen, sowie die Vorbelegung der Q-Werte mit optimistischen Initialwerten ausreichend zur Exploration sind.

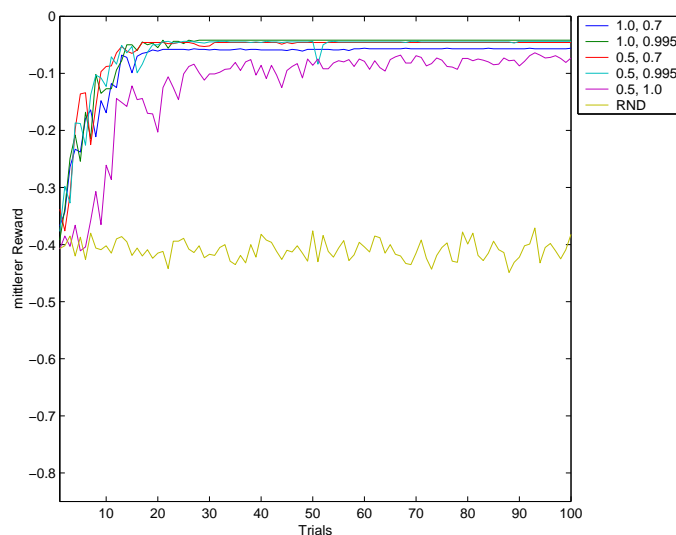


Abb. 4.21: Einfluss verschiedener Rauschwerte (1.0 oder 0.5 zu Beginn) und verschiedener Discount-Faktoren (0.7, 0.995 und 1.0 (kein Discount)) am Beispiel des NG-Q-Traces Agenten ($\lambda = 0.9$), Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

4.5. Vorhandensein von Totzeiten

Der Einfluss von Totzeiten wurde sowohl mit Start- und Zielpunkt im gleichen Tal, wie auch bei Zielpunkt im anderen Tal getestet. Zur Simulation dieses Effektes wurde zwischen Aktionsauswahl und dem Ausführen der Aktion eine Verzögerung von ein bis fünf Takten bewirkt. Dies entspricht einer motorischen Verzögerung eines realen Agenten und verhindert, dass eine korrekte Abbildung vom Zustands-Aktionspaar auf den zugehörigen Reward gelernt werden kann, da der kausale Zusammenhang zwischen einer Aktion und den daraus resultierenden Folgen durch die dazwischenliegende Verzögerung verwischt wird.

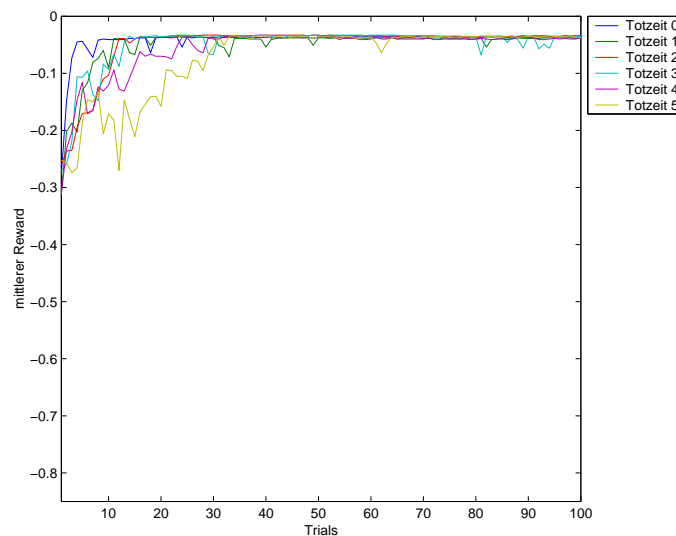


Abb. 4.22: Einfluss der Totzeiten auf die Lerngeschwindigkeit im Szenario Welle mit Starttal = Zieltal am Beispiel des GNG-Q-Agenten, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Im Standard-Szenario mit Start und Ziel im gleichen Tal zeigen die Agenten ohne Totzeit kaum Unterschiede in ihrer Performanz. Das Vorhandensein einer Totzeit, egal welcher Stärke, wirkt sich nicht auf die letztendliche Leistung der trainierten Agenten aus. Der mittlere Reward über das Experiment sinkt jedoch leicht ab. Dies ist eindeutig die Folge einer verringerten Lerngeschwindigkeit (Abb. 4.22). Wie erwartet können die Q-Traces Agenten dies meist kompensieren, wobei der Unterschied zwischen Q-Lernen mit und ohne Traces in der Gesamtp Performanz in diesem Szenario noch gering ist (Abb. 4.23 und Anhang A. Abb. A.3). Ausnahme sind wie meist die R-Agenten, die ein stärker schwankendes Verhalten zeigen.

Im Szenario mit Start- und Zielpunkt in verschiedenen Tälern zeigt sich ein größerer Einfluss der Totzeit auf die Performanz der Agenten. Die offensichtliche Begründung ist, dass für das Schwungholen um den Berg in der Mitte zu überqueren

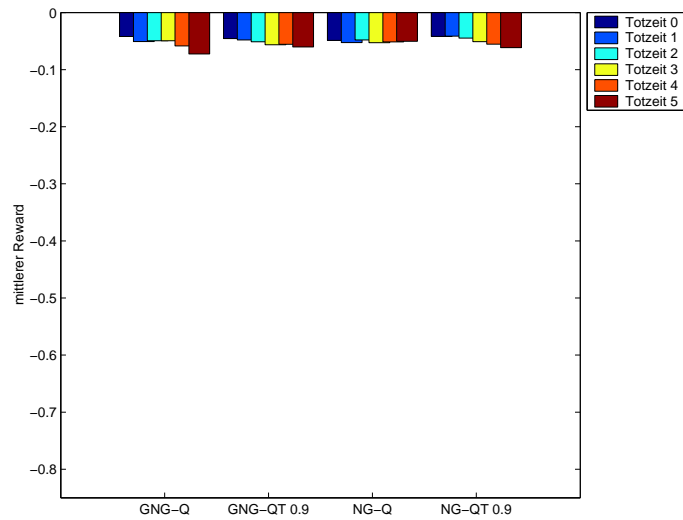


Abb. 4.23: Mittlerer Reward ausgewählter Agenten unter dem Einfluss verschiedener Totzeiten beim Experiment mit Start und Ziel im gleichen Tal der Welle

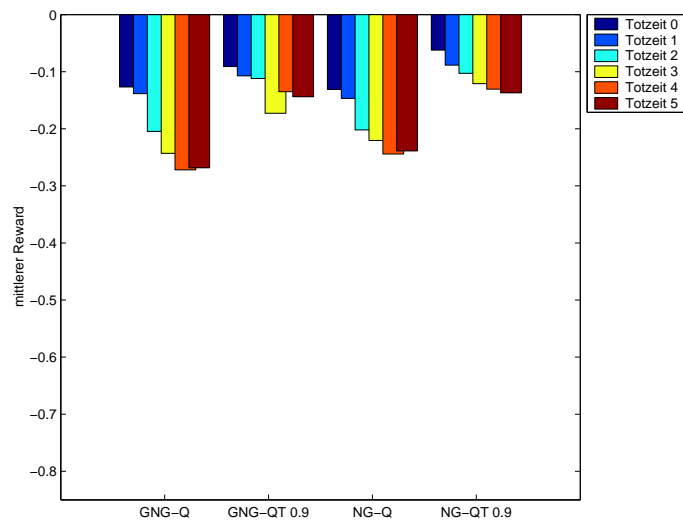


Abb. 4.24: Mittlerer Reward ausgewählter Agenten unter dem Einfluss verschiedener Totzeiten beim Experiment mit Start und Ziel in unterschiedlichen Tälern der Welle

ein gewisses Maß an Timing notwendig ist. Diese zeitliche Abstimmung der Aktionen fällt natürlich mit zunehmender Verzögerung immer schwerer (Abb. 4.24 und Anhang A Abb. A.4). Die CMAC-Agenten zeigen sich noch am wenigsten beeinflusst. Dies ist vor allem der beim CMAC meist höheren Lerngeschwindigkeit zu verdanken. Bei den NG- und GNG-Agenten vergrößert sich mit zunehmender Totzeit der Leistungsunterschied zwischen den Q-Lernverfahren mit und ohne Activity Traces. Nur Traces-Agenten mit großem Wert des Trace-Parameters kann man bei sehr großen Totzeiten noch als sehr gut bezeichnen, sowohl Sarsa- als auch Q-Agenten ohne Traces zeigen doch erhebliche Einbußen, die sich vor allem in einer deutlichen Schwankung der Ballposition um das Ziel zeigen (Abb 4.25). Bei den LLCs Agenten verschlechtern sich die schon im Vorhinein nicht besonders guten Ergebnisse weiter. Der Einfluss der Traces lässt sich hier, obwohl vorhanden, nicht auf den ersten Blick erkennen, da die Schwankungen des Clusterers dies überlagern. R-Lernen kann schon ohne Totzeiten das Problem kaum lösen. Mit zunehmender Totzeit werden auch diese geringen Leistungen nicht mehr erreicht. Insgesamt lässt sich also erkennen, dass die Totzeiten nicht nur den Lernvorgang verlängern, sondern auch die Qualität der erlernten Handlungsvorschrift negativ beeinflussen. Dies kann jedoch durch die Verwendung von Activity-Traces teilweise kompensiert werden. Es zeigt sich erneut, dass dieses Verfahren in Szenarien mit zeitlich getrennten Zusammenhängen Vorteile auch bei der Güte der erlernten Policy liefert. Beim Vergleich der Clusterer zeigen sich die CMAC-Agenten am wenigsten beeinflusst. Bei ihnen führt die Tatsache, dass die Clusterung nicht parallel zu den Q-Werten erlernt werden muss, zu einer nur geringen Leistungsver schlechterung.

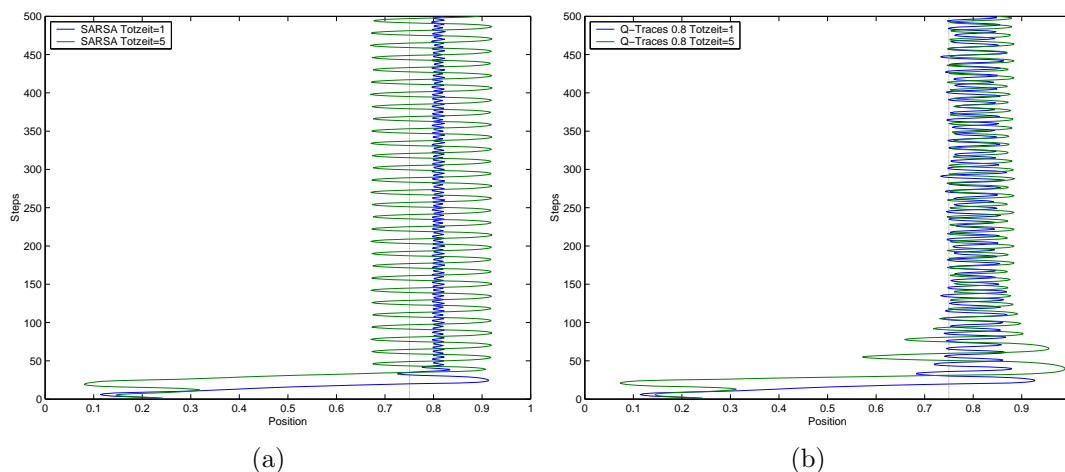


Abb. 4.25: Bewegung des Balles unter dem Einfluss verschiedener Totzeiten bei den Agenten GNG-SARSA (a) und GNG-Q-Traces ($\lambda = 0.8$) (b)

4.6. Veränderliche Zielstellung

In den folgenden Experimenten durchlaufen die Agenten jeweils drei Phasen zu 100 Trials. Nach 100 und nach 200 ändert sich stets die Umgebung und die Agenten müssen lernen, mit der neuen Situation umzugehen, wobei die dritte Phase immer wieder der ersten entspricht.

Veränderter Startpunkt - $0.25 \rightsquigarrow 0.75 \implies 0.8 \rightsquigarrow 0.75$

In diesem Szenario bleibt der Zielpunkt gleich, der Start befindet sich zu Beginn im anderen Tal, in der 2. Phase im Zieltal und in der letzten Phase wieder im anderen Tal. Für die Agenten sollte das kein Problem sein, da die Aufgabe „am Zielpunkt halten“ in allen drei Phasen identisch ist. Die Fragestellung in dieser Aufgabe ist es vielmehr, ob die Clusterer das Wissen um das Schwungholen vergessen.

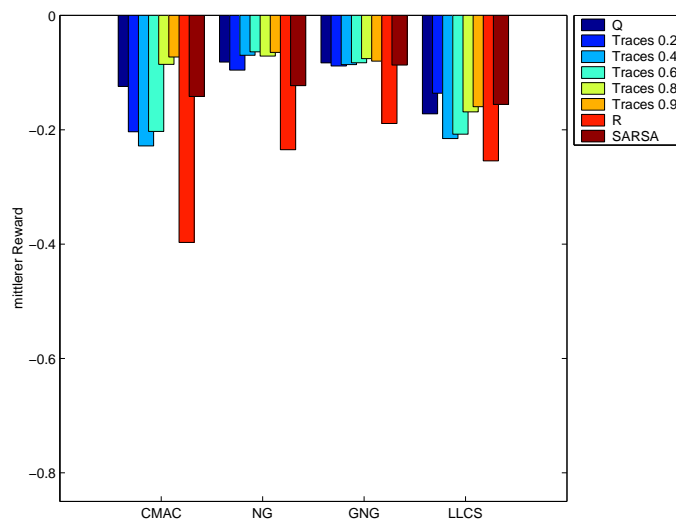


Abb. 4.26: Mittlerer Reward aller Agenten im Experiment mit konstantem Zielpunkt, Wechsel des Startpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials. (RND-Agent: -0.3676)

Die erste Phase verläuft bei allen Agenten entsprechend dem Experiment im Standard-Szenario und auch die zweite Phase stellt für den Großteil kein Problem dar, der mittlere Reward kann in der zweiten Phase sogar noch verbessert werden. Dies ist darauf zurückzuführen, dass bereits im Zieltal gestartet wird. Das schlechteste Verhalten zeigen wie immer die R-Agenten; in Kombination mit dem CMAC als Clusterer ist bei diesem Lernverfahren in der 2. Phase das schlechteste Verhalten zu beobachten.

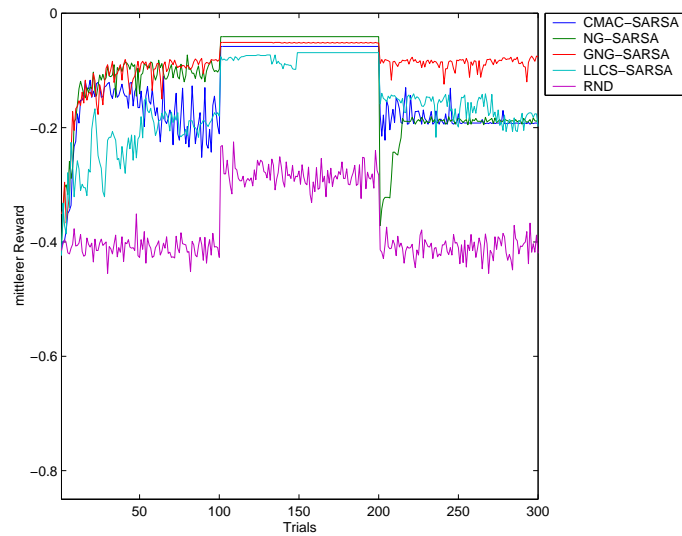


Abb. 4.27: Sarsa-Agenten im Experiment mit konstantem Zielpunkt, Wechsel des Startpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

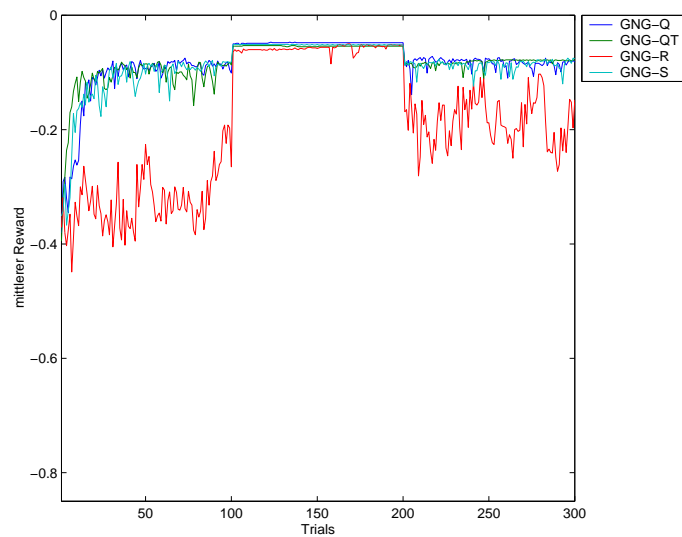


Abb. 4.28: GNG-Agenten im Experiment mit konstantem Zielpunkt, Wechsel des Startpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Erst in der dritten Phase treten deutliche Unterschiede zwischen den Clusterern und den Lernverfahren hervor (Beispiel Sarsa-Agenten in Abb. 4.27). So liefern die CMAC-basierten Agenten nur in Kombination mit Q- oder Q-Sarsa-Lernen auch in der dritten Phase zufriedenstellende Ergebnisse, wohingegen das GNG als Clusterer mit allen Lernverfahren sehr gute Leistungen zeigt (Abb. 4.28). Dies ist vermutlich der schnelleren und besseren Clusterung durch Hinzufügen und Entfernen von Neuronen zu verdanken. Den GNG-Agenten ist es sogar möglich ihre Leistung aus der ersten Phase noch zu verbessern, v. a. der GNG-R-Agent profitiert von den zusätzlichen Lernschritten über die drei Phasen und zeigt ein etwas besseres Verhalten als in anderen Versuchen.

Die nicht-inkrementellen Clusterer haben in der dritten Phase teilweise Probleme: der NG-Sarsa-Agent muss z. B. mit Beginn der dritten Phase sein altes Verhalten wieder neu erlernen und erreicht hier aber auch nicht die Performanz aus der ersten Phase (Abb. 4.27). Sowohl CMAC als auch NG liefern erst in Kombination mit sehr hohen Traces beim Q-Lernen gute Ergebnisse.

Auf der Seite der inkrementellen Clusterer ist bei den LLCs-Agenten ein schlechtes Verhalten zu beobachten. Sie verlieren in der dritten Phase an Wissen, der Reward sinkt, der Clusterer scheint also nicht stabil genug zu sein um dieses Problem sinnvoll zu lösen.

Veränderter Zielpunkt - $0.25 \curvearrowright 0.3 \implies 0.25 \curvearrowright 0.75$

In diesem Experiment bleibt der Startpunkt in allen drei Phasen gleich. Der Zielpunkt ist in der ersten und letzten Phase im Starttal, wandert aber in Phase zwei in das andere Tal. Darum muss der Agent in dieser Phase lernen, mit Schwung aus dem Starttal zu entkommen. Dies dürfte erschwert werden, da die Q-Werte aus der ersten Phase schon darauf adaptiert sein sollten, den Ball im Starttal zu halten. Darüber hinaus muss das dann erworbene Wissen zum Teil wieder vergessen werden, um in der dritten Phase das Verlassen des Starttals wieder zu vermeiden.

Die Agenten zeigen in diesem Szenario ein unterschiedliches Verhalten. Die R-Agenten mit CMAC, NG und LLCS Clusterer können überhaupt nicht umlernen und versagen in Phase 2 deutlich, sie vergessen jedoch nichts, bzw. wenig im Fall des LLCS-R-Agenten, und erreichen in Phase 3 problemlos die Ergebnisse aus Phase 1. Der GNG-R-Agent versucht ein Umlernen und schafft dies teilweise, die Zeit reicht aber nicht aus um völlig erfolgreich zu sein. In Phase 3 benötigt er dann ebenfalls geraume Zeit um die neue, alte Strategie wieder zu erlernen und erreicht den alten Wissensstand gerade so (Abb. 4.29).

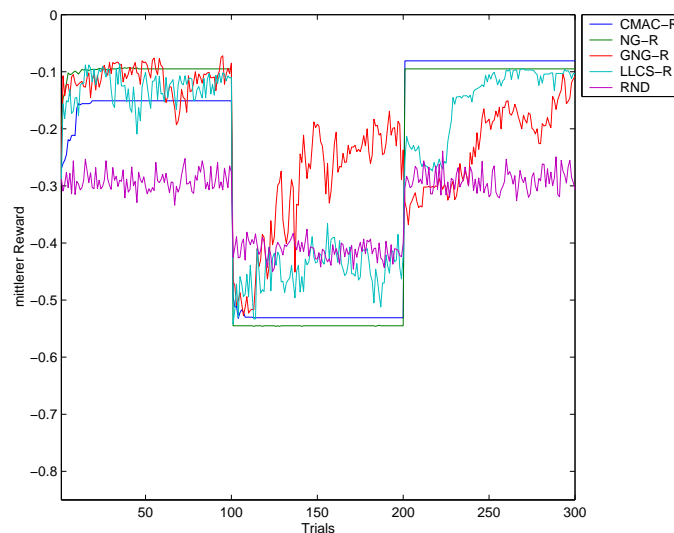


Abb. 4.29: Verhalten der R-Agenten im Experiment mit konstantem Startpunkt, Wechsel des Zielpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Die anderen CMAC-Agenten zeigen ein unbefriedigendes Verhalten, nur unter Verwendung von Traces mit großem Trace-Parameter ist ein Umlernen möglich. Doch je besser dies in Phase 2 gelang, desto schlechter war dann die Leistung in Phase 3, ein erneutes Umlernen ist diesen Agenten also nicht möglich (Abb. 4.30). Die NG- und GNG-Agenten lösen das Problem deutlich besser, abgesehen vom NG-Sarsa-Agenten, der unfähig zum Umlernen ist. Das Lernen der Strategie in Phase 2 erfolgt problemlos, wobei wie meist ein Vorteil der Traces-Agenten bei der Lerngeschwindigkeit, und in diesem Fall auch beim Lernerfolg zu bemerken ist. Die Anpassung für Phase 3 erfolgt dann auch wieder schnell, allerdings mit unterschiedlicher Qualität in den einzelnen Runs, von denen einige sehr gut, andere nur mittelmäßig zu bewerten sind. Deshalb sinkt der Reward pro Trial auch leicht ab, wobei die Performanz der GNG-Agenten am Ende des Experiments durchgehend minimal besser als die der NG-Agenten ist (Abb. 4.31).

Die LLCs-Agenten zeigen eine große Streuung in ihren Leistungen. Meist ist es so, dass sie nicht umlernfähig sind, aber das Wissen aus Phase 1 für Phase 3 bewahren können. Einzelne Runs zeigen aber ein völlig abweichendes Verhalten. So schaffen es die Agenten in diesen Fällen manchmal sehr gut umzulernen, dafür versagen sie dann in Phase 3. In anderen Runs zeigen sie Leistungen, vergleichbar mit den besten Runs der NG- und GNG-Agenten. Die inkonsistente Leistung des Clusterers scheint durch dieses Szenario noch verstärkt zu werden.

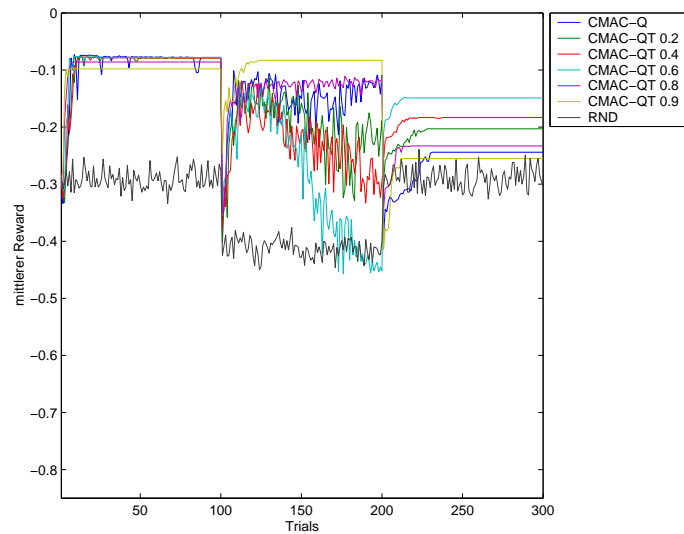


Abb. 4.30: Verhalten der CMAC-Agenten im Experiment mit konstantem Startpunkt, Wechsel des Zielpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

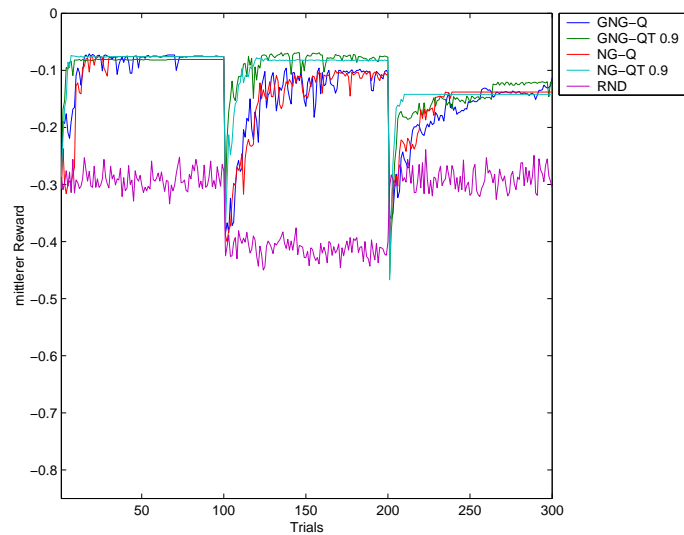


Abb. 4.31: Verhalten der NG- und GNG-Agenten im Experiment mit konstantem Startpunkt, Wechsel des Zielpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Veränderter Start- und Zielpunkt - $0.25 \curvearrowright 0.75 \implies 0.75 \curvearrowright 0.25$

In diesem Experiment werden Start- und Zielposition zweimal komplett getauscht. Das Problem für die Agenten besteht darin, dass in der zweiten Phase das „Schwung holen“ genau gegenläufig dem aus der ersten ist.

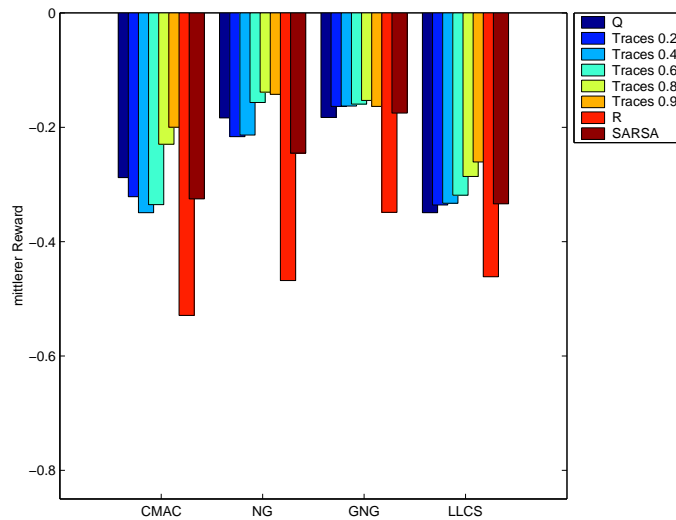


Abb. 4.32: Mittlerer Reward aller Agenten beim Tausch von Start- und Zielpunkt nach 100 Trials und Zurücktauschen nach 200 Trials, (RND-Agent: -0.4102)

Bei keinem der Agenten kann Wissen der ersten Phase für die zweite genutzt werden; entweder es findet ein Neulernen statt (Beispiel GNG vgl. Abb. 4.33) oder der Agent versagt in der zweiten Phase. Generell verlieren alle Agenten nach dem zweiten Wechsel an Leistungsfähigkeit.

Das Erlernen der neuen Zielstellung fällt den NG- und GNG-Agenten am leichtesten, wobei einfaches Q-Lernen hier in der zweiten und dritten Phase zur besten Performanz führt (vgl. Abb. 4.34 NG-Q und NG-QT); die Traces-Agenten verlieren hier hingegen relativ deutlich, vermutlich sind die Q-Werte bei ihnen schon stärker ans ursprüngliche Problem angepasst. Beim NG-Sarsa-Agenten kann in der zweiten Phase keinerlei Anpassung beobachtet werden, in dritten kann jedoch einwandfrei das Verhalten der ersten Phase erzeugt werden (Abb. 4.34 NG-S). Damit ist klar, dass die Kombination von NG und Q-Sarsa-Lernen nicht für Umlernprozesse zu gebrauchen ist, was sich auch schon im vorhergehenden Versuch andeutete.

Zu den CMAC-Agenten genügt es an dieser Stelle zu erwähnen, dass diesen das Umlernen hier sehr schwer fällt; auch wenn hohe Traces das Verhalten verbessern, reicht die Performanz jedoch nicht an die der anderen guten Agenten heran.

Die LLCS-Agenten zeigen ein recht unberechenbares Verhalten. Manchmal erlernen sie die erste Phase gut, können nicht umlernen, aber bewahren das Wissen für die dritte Phase. In anderen Runs vergessen sie dies. Dann gibt es Runs, in denen ein erstes, aber kein weiteres Umlernen möglich ist. Und schließlich gibt es auch wenige Einzelruns, die man als erfolgreich bezeichnen kann. Dies gilt für alle LLCS-Agenten, außer dem R-Agent, der hier nun völlig versagt.

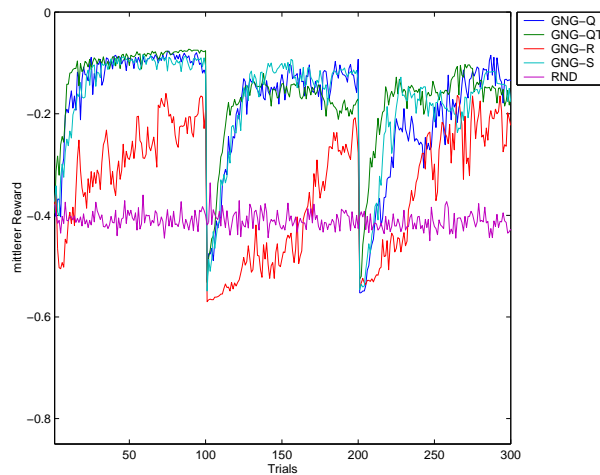


Abb. 4.33: GNG-Agenten beim Tausch von Start- und Zielpunkt nach 100 Trials und Zurücktauschen nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

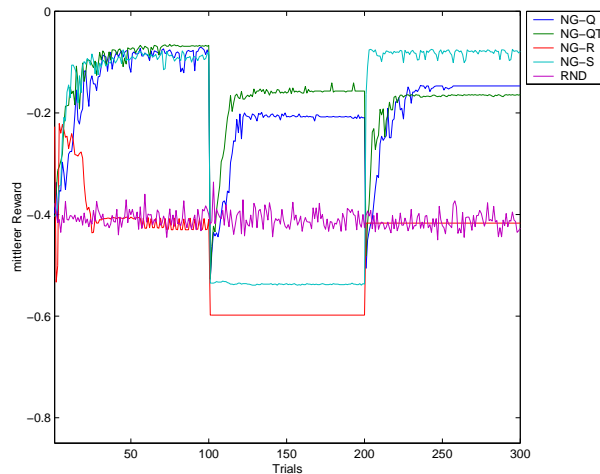


Abb. 4.34: NG-Agenten beim Tausch von Start- und Zielpunkt nach 100 Trials und Zurücktauschen nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

4.7. Teilweise beobachtbare Umwelt - Störgröße Wind

In diesem Experiment sollen die Auswirkungen eines nicht bekannten Störfaktors für den Agenten beobachtet werden. Der hierzu verwendete Wind in der Umgebung ist dem Agenten nicht bekannt und wird zufällig in jedem Schritt aus einem bestimmten Intervall gewählt. Dieses Intervall gibt die maximale Windstärke in Newton an. Für diese maximale Windstärke werden Werten von $0 N$, $2 N$, $4 N$ und $8 N$ getestet. Es ist zu erwarten, dass sich der durchschnittliche Reward verringert je stärker die unbeobachtbare Kraft des Windes ist.

Da dieses Experiment im Standard-Szenario stattfindet, ist der Einfluss des Windes auf alle Agenten im allgemeinen gering (Anhang A Abb. A.5). Die ohnehin schwachen Agenten zeigen sich auch von der Störung mehr beeinflusst. So geht die Performanz der verschiedenen Clusterer beim R-Lernen mit zunehmendem Wind immer weiter auseinander, wobei sich GNG als der beste und NG als der schlechteste Clusterer erweist.

Bei den GNG-Agenten kann eine höhere Varianz des mittleren Rewards über die Trials beobachtet werden; auf der Seite der nicht-inkrementellen Clusterer reagieren die NG-Agenten empfindlicher auf die Störung als die CMAC-Agenten. Dies ist möglicherweise darauf zurückzuführen, dass der CMAC schon eine feste Clusterung des Inputraumes hat und diese nicht durch die Störung beeinflusst werden kann. Bei den LLCS-basierten Agenten ist ein deutlicher Performanz-Einbruch erkennbar, der wiederum auf einen größeren Unterschied zwischen den einzelnen Runs zurückzuführen ist.

Unter den Lernverfahren wirkt sich die Störung am stärksten auf das R-Lernen aus, dessen Reward-Verlauf über einen Run vielen Schwankungen unterliegt. Q-Sarsa-Lernen, welches ohne Wind dem einfachen Q-Lernen ebenwürdig war, zeigt mit zunehmendem Wind etwas stärkere Schwankungen im mittleren Reward. Die Abbildungen 4.35 und 4.36 zeigen Q-Traces-Agenten unter dem Einfluss verschiedener Windstärken. Beim den GNG-basierten Agenten ist eine deutliche Abnahme der Performanz mit zunehmender Windstärke zu erkennen; der NG-Q-Traces-Agent hingegen verschlechtert sich erst bei einer Windstärke von acht. Dies lässt vermuten, dass die Störung auf den inkrementellen Clusterer größeren Einfluss hat.

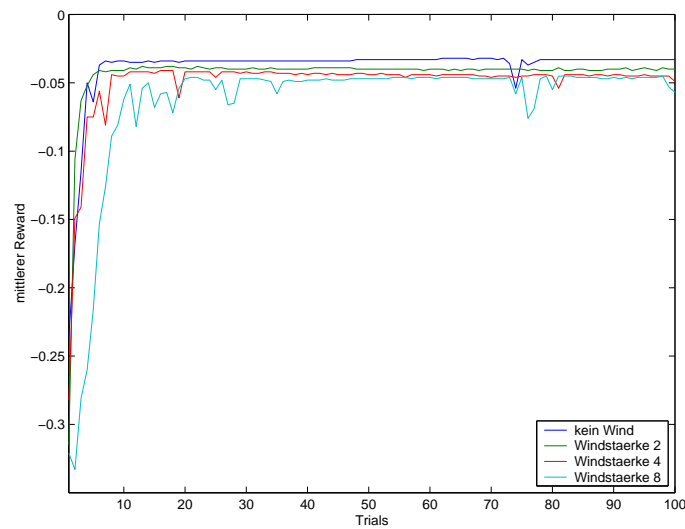


Abb. 4.35: GNG-Q-Traces-Agent mit $\lambda = 0.6$ bei verschiedenen Windstärken, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

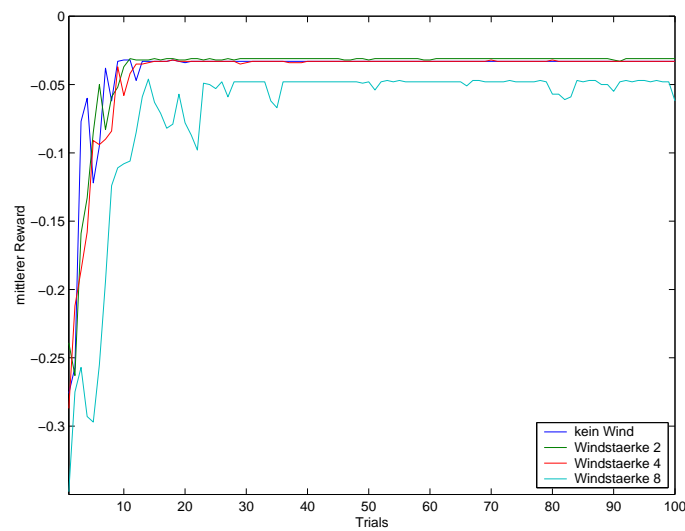


Abb. 4.36: NG-Q-Traces-Agent mit $\lambda = 0.6$ bei verschiedenen Windstärken, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

4.8. Anzahl der möglichen Aktionen des Agenten

In diesem Experiment wird die Anzahl der Aktionen, die den Agenten zur Verfügung stehen, variiert. Standardmäßig mit zwei Aktionen ausgestattet, erhalten die Agenten hier jetzt 3, 4, 5 oder 8 Aktionen. Mit einer ungeraden Aktionenanzahl wäre es den Agenten prinzipiell auch möglich den Ball nicht zu bewegen, eine höhere Anzahl ermöglicht es dem Agenten seine Krafteinwirkung beim Bewegen des Balles zu dosieren.

Die Verfügbarkeit von mehreren Aktionen hat keinen wesentlichen Einfluss auf die Gesamtperformanz (Anhang A, Abbildung A.6), was aber vermutlich auch dem Einsatz im Standard-Szenario geschuldet ist. Nur bei einigen Agenten lässt sich ein mehr oder weniger starker Einfluss erkennen. So verschlechtert sich z. B. der NG-Q-Sarsa-Agent mit zunehmender Aktionenanzahl, ein ebenso schlechter Einfluss ist bei den R-Agenten zu erkennen. Die CMAC-Agenten zeigen sich kaum beeinflusst, schaffen es aber den Ball näher am Ziel zu halten (Abb. 4.37).

Allgemein ist festzustellen, dass das erwartete Verhalten, dass die Agenten mit Activity Traces schneller auf die Policy konvergieren, umso deutlicher hervortritt, je mehr Aktionen verwendet werden (GNG als Beispiel in Abb. 4.38), da bei gleicher Zahl von Schritten mehr Zustands-Aktions-Paare zur Verfügung stehen und so jedes einzelne weniger oft gewählt und adaptiert wird. Das bedeutet, dass die Lerngeschwindigkeit mit zunehmender Anzahl von Aktionen zwar sinkt, dies jedoch durch die Traces kompensiert werden kann.

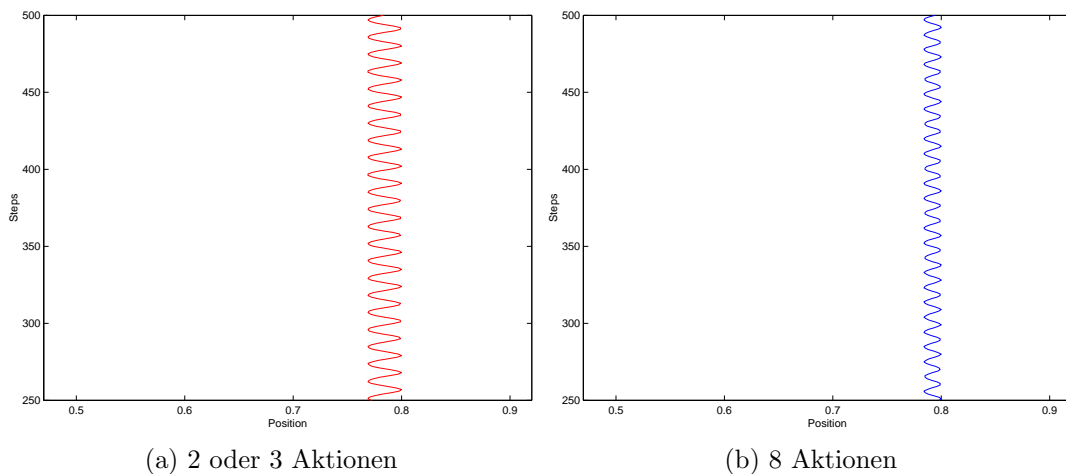


Abb. 4.37: Bewegung des Balles durch den CMAC-Q-Traces Agenten mit $\lambda = 0.4$, die Anzahl der möglichen Aktionen für den Agenten entspricht zwei und acht.

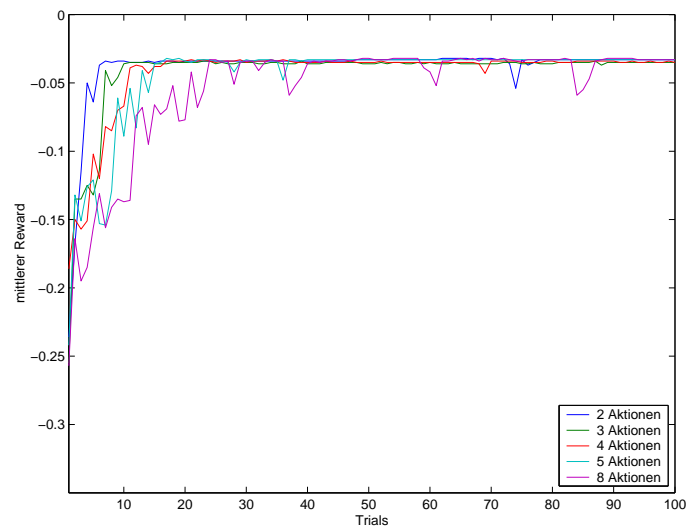


Abb. 4.38: Vergleich von GNG-Traces-Agenten mit $\lambda = 0.6$ und unterschiedlicher Anzahl von möglichen Aktionen, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

4.9. Kombination von Teilproblemen

Im kombinierten Szenario haben die Agenten die Schwierigkeit, mit mehreren störenden Einflüssen, Wind und Totzeit, zurechtzukommen. Die Einflüsse der Störgrößen allein hatten, mit den für diesen Test gewählten kleinen Werten, jeweils nur geringe Beeinträchtigungen der Leistung zur Folge, wobei Totzeiten auf die Leistung von Agenten ohne Traces stärkere Auswirkungen, speziell auf die Lerngeschwindigkeit, als der Wind hatte.

Im ersten Teil des Szenarios mit Start- und Zielpunkt im gleichen Tal sind die Ergebnisse fast aller Agenten gleich zu denen der Standardumwelt ohne störende Einflüsse. Lediglich die CMAC-Agenten verschlechtern sich minimal. Die Ausnahme davon sind die Agenten mit R-Learning, die das Problem deutlich schlechter lösen können (Abb. 4.40).

Beim Experiment mit Start und Ziel in verschiedenen Tälern zeigt sich, in wie weit die Agenten trotz störender Einflüsse das Schwungholen lernen können (Abb. 4.41). Die CMAC-, NG- und GNG-Agenten zeigen allgemein gute Ergebnisse und erreichen am Ende der Tests fast das Ergebnis ohne Störungen, mit leichten Verlusten, die vor allem durch leichte Schwankungen verursacht werden (Abb. 4.39). Allerdings verringert sich die Lerngeschwindigkeit durch diese Einflüsse, was für fast alle Agenten eine geringere Durchschnittsperformanz über das Experiment bewirkt. Q-Traces Agenten können dies, je größer der Trace-Parameter λ gewählt wird, immer besser kompensieren (Abb. 4.39).

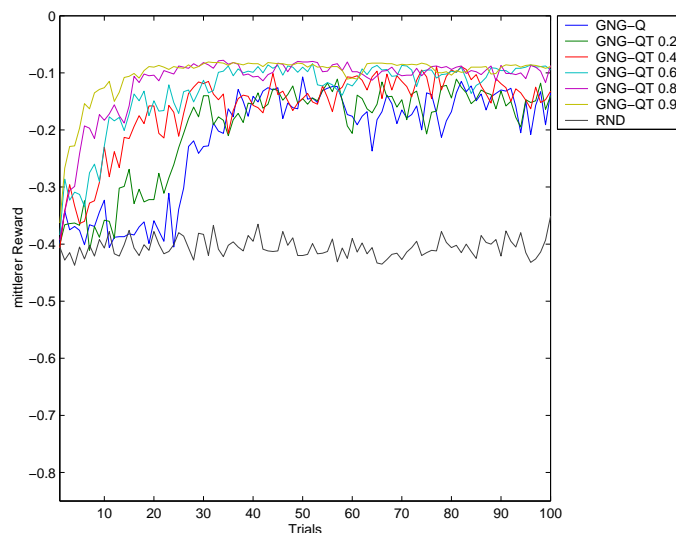


Abb. 4.39: Verminderte Lerngeschwindigkeit und Einfluss der Traces im kombinierten Szenario mit Starttal ungleich Zieltal, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

Die LLCS-Agenten und die Agenten, die R-Lernen verwenden, liefern schon im Standardfall eine schlechte Leistung, und können nach dem zusätzlichen Leistungsverlust nicht mehr als erfolgreich betrachtet werden.

Diese Ergebnisse entsprechen denen der Tests, in denen nur Totzeit, aber kein Wind vorherrschte. Der störende Einfluss, den die Totzeit auf die Leistung der Agenten hat, wird durch die nicht beobachtbare Kraft also nicht weiter vergrößert.

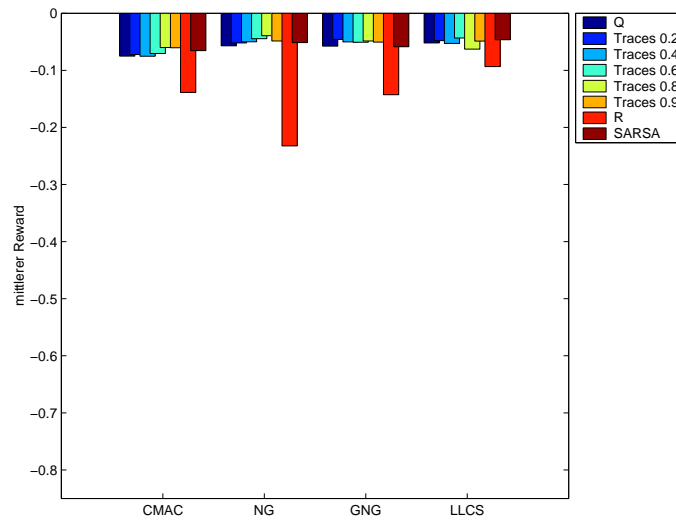


Abb. 4.40: Mittlerer Reward der Agenten im kombinierten Szenario mit Starttal gleich Zielal, (RND-Agent: -0.2820)

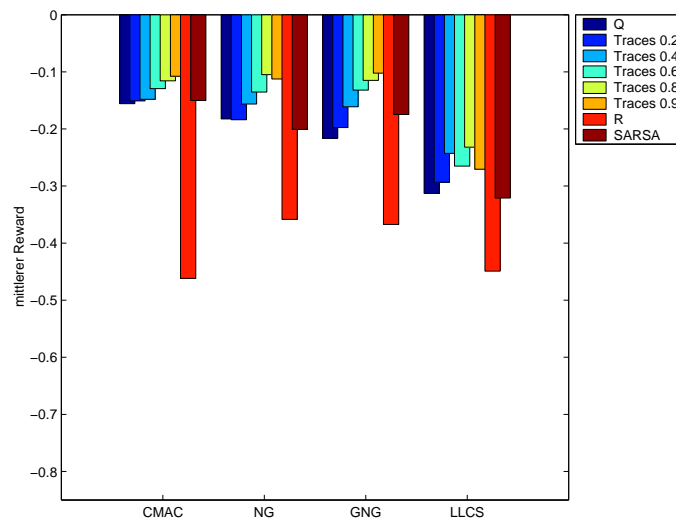


Abb. 4.41: Mittlerer Reward der Agenten im kombinierten Szenario mit Starttal ungleich Zielal, (RND-Agent: -0.4057)

4.10. Zusätzlicher Test zu den Q-Sarsa-Agenten

Während aller Tests in dieser Arbeit zeigen die Q-Agenten und die Sarsa-Agenten eine vergleichbare Performanz. Erst durch die Verwendung von Activity Traces erarbeitet sich das Q-Lernen einen Leistungsvorsprung. Aus diesem Grund soll mit einigen zusätzlichen Tests untersucht werden, inwieweit auch Sarsa-Lernen durch den Einsatz von Traces profitiert. Zu diesem Zweck werden die Agenten GNG-Q und GNG-Sarsa sowohl ohne, als auch mit Traces ($\lambda = 0.9$) in interessanten Szenarien verglichen.

In allen Tests zeigt sich, dass die Leistung der Sarsa-Traces-Agenten der Leistung der Q-Traces Agenten entspricht. Die Lerngeschwindigkeiten erhöhen sich in gleichem Maße, wie man besonders im Totzeit-Experiment gut erkennen kann. Das Verhalten beim Zielwechsel ist identisch und auch die Leistung bei der Verwendung der schwierigsten Umgebung, dem Berg, ist nun gut, mit nur noch einem misslungenem Versuch.

Auch wenn diese Tests nur einen Teil der Versuchsreihe umfassen, können sie doch aufgrund der relativen Schwierigkeit durchaus als repräsentativ gelten, und erlauben den Schluss, dass die Verwendung von Traces für Sarsa-Lernen identische Auswirkungen wie beim Einsatz für Q-Lernen hat.

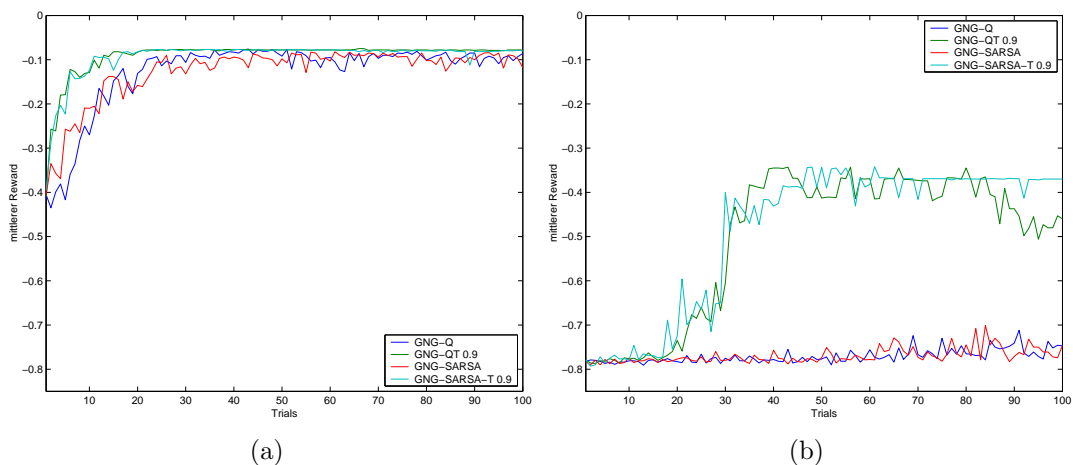


Abb. 4.42: Vergleich von Q- und Sarsa-Lernen ohne und mit Activity Traces ($\lambda = 0.9$) am Beispiel der Agenten mit GNG Clusterer in (a) Standardumgebung mit Startposition 0.25 und Zielposition 0.75 und (b) Szenario Berg

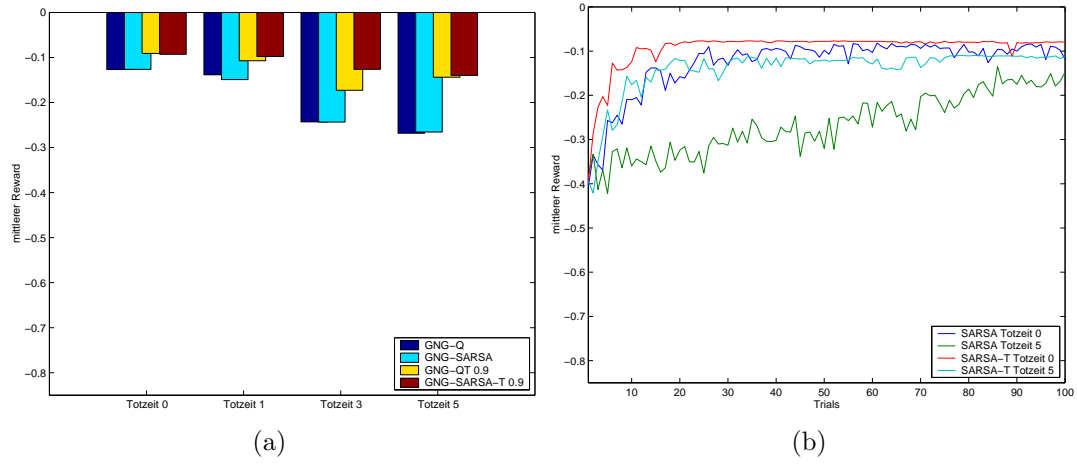


Abb. 4.43: Vergleich von GNG-Q- und GNG-Sarsa-Agenten ohne und mit Activity Traces ($\lambda = 0.9$) unter dem Einfluss von Totzeiten

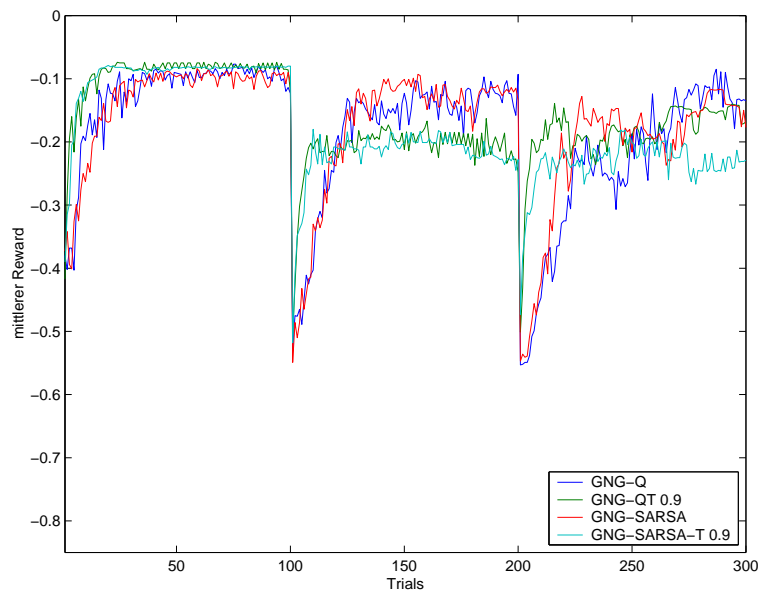


Abb. 4.44: Vergleich von GNG-Q- und GNG-Sarsa-Agenten ohne und mit Activity Traces ($\lambda = 0.9$) im Zielwechsel-Experiment mit Tausch von Start- und Zielposition in Phase 2

5. Zusammenfassung und Ausblick

Lediglich im einfachsten Szenario, Welle mit Start- und Zielpunkt im gleichen Tal, zeigen alle Agenten zufriedenstellende Ergebnisse. Sobald ein schwierigeres Szenario zu bewältigen ist oder störende Realweltbedingungen herrschen, zeigen sich deutliche Unterschiede in der Leistung der Agenten, wobei dafür sowohl Einflüsse des Clusterers, wie auch das verwendete Lernverfahren verantwortlich sind.

Die R-Agenten zeigen stets die schlechtesten Ergebnisse und sind in ein einem großen Teil der Szenarien völlig erfolglos. Das Versagen dieser Agenten ist höchstwahrscheinlich auf das Problem der Limit Cycles zurückzuführen und macht dieses Verfahren für derartige Probleme unbenutzbar. Lediglich in Verbindung mit dem GNG lässt sich noch in einigen Fällen ein Lernvorgang erkennen, jedoch ist die Lerngeschwindigkeit viel zu gering und die Ergebnisse nicht zufriedenstellend. Q- und SARSA-Lernen zeigen meist gleichwertige Ergebnisse und sind beide für fast alle der untersuchten Probleme geeignet. Natürlich schlagen sich auch hier besonders schwierige Szenarien negativ auf die Leistung der Agenten nieder, doch sind sie in der Lage Lösungen für die meisten Problemfälle zu finden. Weiter verbessert wird die Leistung dieser Verfahren durch die Verwendung von Activity Traces, mit denen dann auch die komplexesten Szenarien wie der Berg gut bewältigt werden können. Wie zu erwarten kann durch Verwendung dieses Ansatzes die Lerngeschwindigkeit erhöht werden, so dass der störende Einfluss verschiedener Realwelteigenschaften auf die benötigte Zeit bis zur Konvergenz gut kompensiert werden kann. Außerdem zeigt sich, dass Activity-Traces auch die Qualität der erlernten Policy verbessern können, da sie den Agenten helfen, zeitlich getrennte kausale Zusammenhänge zu erkennen.

Die CMAC-basierten Agenten zeigen in vielen Experimenten gute Ergebnisse, welche sich v. a. auch in einem zügigem Lernen äußern. Das ist darauf zurückzuführen, dass diese Agenten nur die Q-Werte anpassen müssen ohne dabei die Clusterung des Eingaberaums zu erlernen. In anderen Szenarien jedoch wird bei den CMAC-Agenten die starre Clusterung zum Problem: so ist es z. B. in der Ebene dem Agenten nicht möglich den Ball fein genug um das Ziel zu bewegen, da für die dort bestehende Kombination aus Position und Geschwindigkeit des Balles dem CMAC nicht die passende Clusterung zur Verfügung steht. Auch scheint die starre Clusterung ein Umlernen der Agenten zu erschweren, was die schlechteren Ergebnisse in einigen Experimenten zur veränderlichen Zielstellung andeuten.

Auf der Seite der nicht-inkrementellen Clusterer zeigen die NG-Agenten hervor-

ragende Leistungen in nahezu allen Experimenten und arbeiten allgemein ohne größere Auffälligkeiten. Die große Ausnahme stellt das Szenario Berg da, bei der der Clusterer völlig versagt.

Die Leistungen der Agenten mit GNG als Clusterer sind durchweg als gut zu bezeichnen, und liegen etwa auf dem Niveau der NG-Agenten, sie können allerdings mit problematischeren Szenarien wie der Berg-Umwelt meist noch etwas besser umgehen. Die Lerngeschwindigkeit und damit die Dauer bis zur Konvergenz sind ebenfalls vergleichbar mit dem NG, jedoch geringer als beim CMAC, was zeigt, dass das gleichzeitige Lernen von Clusterpositionen und Q-Werten die Lerngeschwindigkeit doch leicht verringert, da die Anfangs starke Bewegung der Neuronen den Zusammenhang zwischen Position im Eingaberaum und erlerntem Q-Wert verwischt. Außerdem ist die Verzögerung des Lernens auch damit begründet, dass in der Anfangsphase noch Neuronen eingefügt werden müssen. Kleinere Probleme der GNG-Agenten gibt es jedoch dadurch, dass manchmal einzelne Neuronen nach Entfernung ihrer Kanten durch den Algorithmus gelöscht werden und damit die diesen Neuronen assoziierten Q-Werte verloren gehen. In fast allen Fällen führt dies zu einem kurzfristigen Performanzeinbruch der unterschiedlich stark ausfällt, aber nach durchschnittlich zwei bis vier Trials wieder kompensiert werden kann. Nur in sehr seltenen, einzelnen Fällen kann es passieren, dass der Agent das verlorene Wissen nicht wieder erlernt.

Die LLCS-Agenten zeigen die schlechtesten Leistungen aller Clusterer. Dies liegt daran, dass die Ergebnisse sehr schwankend sind. In einzelnen Runs sind die Ergebnisse absolut vergleichbar mit den besten Runs der anderen Agenten. Allerdings gibt es auch deutlich häufiger schlechte Runs und sogar bei einfacheren Aufgaben einige die nur als erfolglos bezeichnet werden können. Nur in den Szenarien, bei denen die Zielposition relativ nahe der Startposition ist, fiel die Performanz gegenüber den anderen Clusterern weniger ab, was deutlich macht, dass die Platzierung der Neuronen in vielen Fällen suboptimal ist. Die möglichen Gründe für dieses Verhalten sind vielfältig. Zum Einen kann es auch bei den LLCS vorkommen, dass Neuronen und damit auch die zugehörigen Q-Werte aus dem Netzwerk entfernt werden. Allerdings geschieht dies nicht nur wie beim GNG durch Löschen von Neuronen die keine Kanten mehr besitzen, sondern zusätzlich noch durch die Similarity-Based-Deletion von Neuronen in Regionen mit hoher Neuronendichte. Da z. B. besonders in der Nähe des Zielpunktes diese Dichte oftmals recht hoch sein kann, kann es manchmal passieren, dass gerade da wichtige Information durch das Löschen verloren geht. Weiterhin wurde in vorherigen Untersuchungen zu diesem Clusterer festgestellt, dass es manchmal zu zufälligen Abweichungen in der Leistung des Netzwerkes kommt. Die vermutete Ursache dafür könnte in der zufälligen Platzierung der beiden Initialneuronen liegen. In diesem Zusammenhang ist auch das verzögerte Einfügen neuer Neuronen zu erwähnen, das durch die Einfügeschwellwerte verlangsamt wird. Dadurch wird das Lernen der Q-Werte verzögert, da besonders in der Anfangsphase die wenigen Neuronen stark bewegt werden, und man nicht von einer brauchbaren Adaption

sprechen kann. Wie schnell eingefügt wird, hängt, außer vom Schwellwertparameter, auch vom Fehlerzähler der Neuronen ab, und dessen Wert wird, zumindest zu Beginn, auch durch die Initialposition beeinflusst. Außerdem kann nicht ausgeschlossen werden, dass die Wahl der Parameter trotz Voruntersuchung nicht für alle Szenarien optimal ist. Der letzte Grund für die unbefriedigende Performanz könnte auch in der Art der Implementierung des Netzwerkes für diese Arbeit sein. In [Hamker, 2001] wurde das Netzwerk für den überwachten Einsatz vorgestellt, die Fehlerzähler anhand des Ausgabefehlers adaptiert. In dieser Arbeit wurde es unüberwacht als reiner Clusterer benutzt, wobei die lokalen Fehlerzähler mittels des Clusterfehlers verändert werden. Es ist möglich, dass dies weniger gute Ergebnisse bei der Steuerung des Netzwerkes liefert.

Bei der Einschätzung der inkrementellen Clusterer muss allerdings gesagt werden, dass eine wichtige Eigenschaft, die Anpassung der Netzwerkgröße durch Einfügen von Neuronen, in dieser Arbeit nicht zum tragen kam, da die aus Gründen der Vergleichbarkeit gewählte Zahl von maximal 25 Neuronen für dieses Problem, völlig ausreichend war. In realen, hochdimensionalen Zustandsräumen, bei denen man die Zahl der nötigen Neuronen schlecht im Voraus bestimmen kann, sollte besonders das GNG im Vergleich zu den anderen Netzwerken besser abschneiden. Insgesamt lässt sich als Fazit dieser Untersuchung sagen, dass die Agenten mit der Kombination aus Q-Traces oder Sarsa-Traces als Reinforcement-Learning-Verfahren und NG oder GNG als Clusterer über alle Tests gesehen die besten Leistungen bei der Lösung der gestellten Probleme zeigen. Von diesen sind die GNG-Agenten universeller einsetzbar da sie mit allen Problemklassen zurechtkommen, während die NG-Varianten in manchen Szenarien wie dem Berg Probleme haben, dafür in anderen Szenarien leicht besser als das GNG performen und auch nicht an dem teilweisen Gedächtnisverlust durch Löschung von Neuronen leiden können, und damit in den erfolgreich bewältigten Szenarien die etwas sicherere Variante sind. Auch sollte der Wert des Trace-Parameters relativ hoch gewählt werden, da dies in den meisten Fällen bessere Ergebnisse bei der Lerngeschwindigkeit, wie auch in manchen Fällen bei der Qualität der erlernten Policy bewirkt.

Die vorliegende Arbeit ist eine Erweiterung der Diplomarbeit von Helge Renkewitz [Renkewitz, 2002] besonders in Hinsicht auf die Kombinationen von verschiedenen Clusterern und Lernverfahren, mit denen ein umfangreicher Bereich an Agenten abgedeckt wurde. Für zukünftige Arbeiten wären, neben alternativen Lernverfahren, beispielsweise auch weitere Kombinationen der hier untersuchten Problemklassen interessant. Außerdem könnte die verwendete Testumgebung zur Simulation weiterer Probleme verändert werden. Dazu gehört z. B. die Implementierung von Reibung, oder auch die Erweiterung auf zusätzliche Dimensionen wie die Verwendung eines 2-D Gebirges statt einer 1-D Kurve und entsprechenden Aktionsmöglichkeiten des Agenten.

Auch wenn abschließende Tests für die Verwendbarkeit der RL-Verfahren in Realwelten natürlich in solchen stattfinden müssen, sollten die in dieser Arbeit vorge-

stellten Ergebnisse der Untersuchungen der Agenten in den simulierten Szenarien, die wichtige Teilaspekte der Realwelt berücksichtigen, auf jeden Fall deutliche Hinweise auf die Tauglichkeit der einzelnen Verfahren liefern.

Literaturverzeichnis

- [Albus 1975] ALBUS, J. S.: A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC). In: *Transaction ASME* Bd. 97, 1975, S. 220–227
- [Fritzke 1995] FRITZKE, Bernd: A growing neural gas network learns topologies. In: *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, S. 625–632
- [Grossberg 1988] GROSSBERG, S.: Nonlinear neural networks: principles, mechanisms, and architectures. In: *Neural Networks 1* (1988), S. 17–61
- [Hamker 2001] HAMKER, Fred H.: Life-long Learning Cell Structures - continuously learning without catastrophic interference. In: *Neural Networks 14* (2001), S. 551–573
- [Mahadevan 1994] MAHADEVAN, Sridhar: To Discount or Not to Discount in Reinforcement Learning: A Case Study Comparing R-Learning and Q-Learning. In: *International Conference on Machine Learning*, 1994, S. 164–172
- [Martinetz 1993] MARTINETZ, T. M.: Competitive Hebbian learning rule forms perfectly topology preserving maps. In: *ICANN'93, International Conference on Artificial Neural Networks*, Springer, 1993, S. 427–434
- [Martinetz und Schulten 1991] MARTINETZ, T. M. ; SCHULTEN, K.: A „Neural-Gas“ Network Learns Topologies. In: *Artificial Neural Networks*, 1991, S. 397–402
- [Renkewitz 2002] RENKEWITZ, Helge: *Untersuchung verschiedener Reinforcement-Lernverfahren auf ihre Realwelttauglichkeit*, Technische Universität Ilmenau, Diplomarbeit, 2002
- [Rummery und Niranjan 1994] RUMMERY, G. ; NIRANJAN, M.: *On-line Q-learning using connectionist systems*. 1994
- [Throndike 1911] THRONDIKE, Edward: *Animal Intelligence*. Macmillan, 1911
- [Watkins 1989] WATKINS, C. J. C. H.: *Learning from Delayed Reward*, Cambridge University, Dissertation, 1989

Abkürzungsverzeichnis

BM-Neuron	Best-Matching-Neuron
CMAC	Cerebellar Model Arithmetic Computer
CMAC-Agenten	alle Agenten mit CMAC als Clusterer
GNG	Growing Neural Gass
GNG-Agenten	alle Agenten mit GNG als Clusterer
LLCS	Life Long Learning Cell Structures
LLCS-Agenten	alle Agenten mit LLCS als Clusterer
NG	Neural Gas
NG-Agenten	alle Agenten mit NG als Clusterer
Q-Agenten	alle Agenten mit Q-Lernen
Q-Traces-Agenten	alle Agenten mit Q-Lernen mit Traces
R-Agenten	alle Agenten mit R-Lernen
RL	Reinforcement Learning
Sarsa-Agenten	alle Agenten mit Q-Sarsa-Lernen

Abbildungsverzeichnis

2.1. Eindimensionale Ballwelt	7
2.2. Szenario Ebene mit $p = 0.1$	7
2.3. Szenario Wanne mit $p = x^2 - x + 0.25$	8
2.4. Szenario Welle mit $p = 32x^4 - 64x^3 + 42x^2 - 10x + 0.95$	8
2.5. Szenario Berg mit $p = 5x^4 + 4.6x^3 - 7.7x^2 + 2.6x$	9
2.6. Clusterung des Eingaberaumes durch den CMAC	10
4.1. Agenten im Szenario Welle, Starttal = Zieltal	24
4.2. Agenten im Szenario Welle, Starttal \neq Zieltal	24
4.3. CMAC-Agenten im Standard-Szenario, Starttal \neq Zieltal	25
4.4. NG-Traces-Agenten im Standard-Szenario, Starttal \neq Zieltal	26
4.5. Bewegung des Balles bei alternativen Rewardfunktionen	27
4.6. Einfluss der Traces bei alternativer Rewardfunktion	27
4.7. Mittlerer Reward bei alternativer Rewardfunktion, Zielbereich 20%	29
4.8. Mittlerer Reward bei alternativer Rewardfunktion, Zielbereich 10%	29
4.9. Mittlerer Reward bei alternativer Rewardfunktion, Zielbereich 5%	29
4.10. Agenten in der Ebene	30
4.11. Vergleich der Ballposition von CMAC und NG in der Ebene	31
4.12. Q-Traces-Agenten in der Ebene	31
4.13. Q-Sarsa-Agenten in der Ebene	32
4.14. Agenten im Szenario Wanne	33
4.15. CMAC-Agenten zum Vergleich in der Ebene und der Wanne	34
4.16. GNG-Traces im Szenario Wanne	34
4.17. Mittlerer Reward im Szenario Berg	36
4.18. Einfluss der Traces im Szenario Berg	37
4.19. Vergleich der Q-Traces-Agenten im Szenario Berg	37
4.20. Bewegung des Balles im Szenario Berg	38
4.21. Einfluss des Rauschens	40
4.22. Einfluss der Totzeiten auf die Lerngeschwindigkeit	41
4.23. Mittlerer Reward bei Einfluss von Totzeiten, Starttal = Zieltal	42
4.24. Mittlerer Reward bei Einfluss von Totzeiten, Starttal \neq Zieltal	42
4.25. Bewegung des Balles bei verschiedenen Totzeiten	43
4.26. Mittlerer Reward bei sich änderndem Startpunkt	44

4.27. Sarsa-Agenten bei sich änderndem Startpunkt	45
4.28. GNG-Agenten bei sich änderndem Startpunkt	45
4.29. R-Learning bei sich änderndem Zielpunkt	47
4.30. CMAC-Clusterer bei sich änderndem Zielpunkt	49
4.31. NG- und GNG-Agenten bei sich änderndem Zielpunkt	49
4.32. Mittlerer Reward bei wechselndem Start- und Zielpunkt	50
4.33. GNG-Agenten bei wechselndem Start- und Zielpunkt	51
4.34. NG-Agenten bei wechselndem Start- und Zielpunkt	51
4.35. GNG-Q-Traces-Agent bei verschiedenen Windstärken	53
4.36. NG-Q-Traces-Agent bei verschiedenen Windstärken	53
4.37. Bewegung des Balles bei 2 und 8 möglichen Aktionen	54
4.38. GNG-Traces-Agent mit unterschiedlicher Anzahl von Aktionen	55
4.39. Einfluss der Traces im kombinierten Szenario	56
4.40. Mittlerer Reward im kombinierten Szenario, Starttal = Zieltal	57
4.41. Mittlerer Reward im kombinierten Szenario, Starttal \neq Zieltal	57
4.42. Vergleich von Q- und Sarsa-Lernen mit und ohne Traces	58
4.43. Sarsa-Lernen mit Traces beim Einfluss von Totzeiten	59
4.44. Sarsa-Lernen mit Traces beim Zielwechsel-Experiment	59
A.1. Verhalten aller Agenten im Standard-Szenario	68
A.2. Mittlerer Reward bei verschiedenen Rauscheigenschaften	69
A.3. Mittlerer Reward bei Totzeiten in der Welle	70
A.4. Mittlerer Reward bei Totzeiten in der Welle mit Start \neq Ziel	71
A.5. Mittlerer Reward bei verschiedener Windstärke	72
A.6. Mittlerer Reward bei verschiedener Anzahl von möglichen Aktionen	73
A.7. Verhalten aller Agenten bei sich änderndem Zielpunkt	74
A.8. Verhalten aller Agenten bei wechselndem Start- und Zielpunkt	75

A. Abbildungen

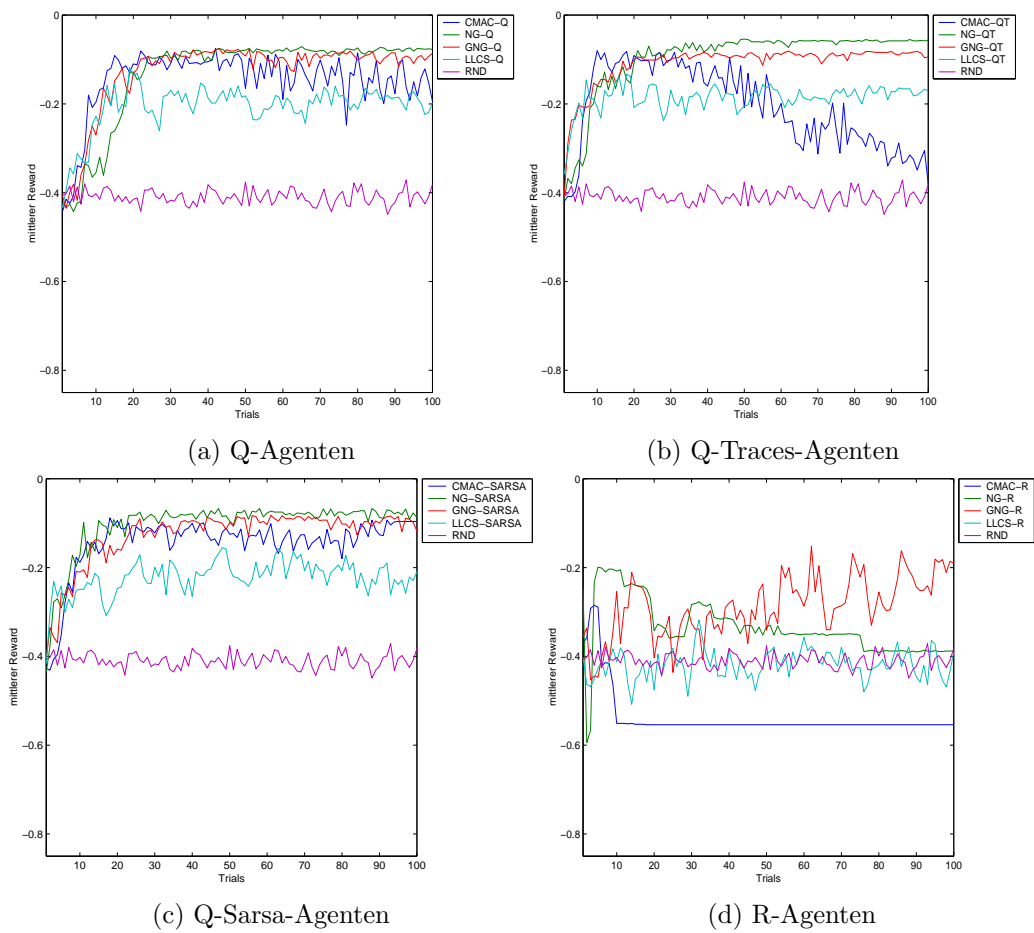


Abb. A.1: Verhalten aller Agenten im Standard-Szenario, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

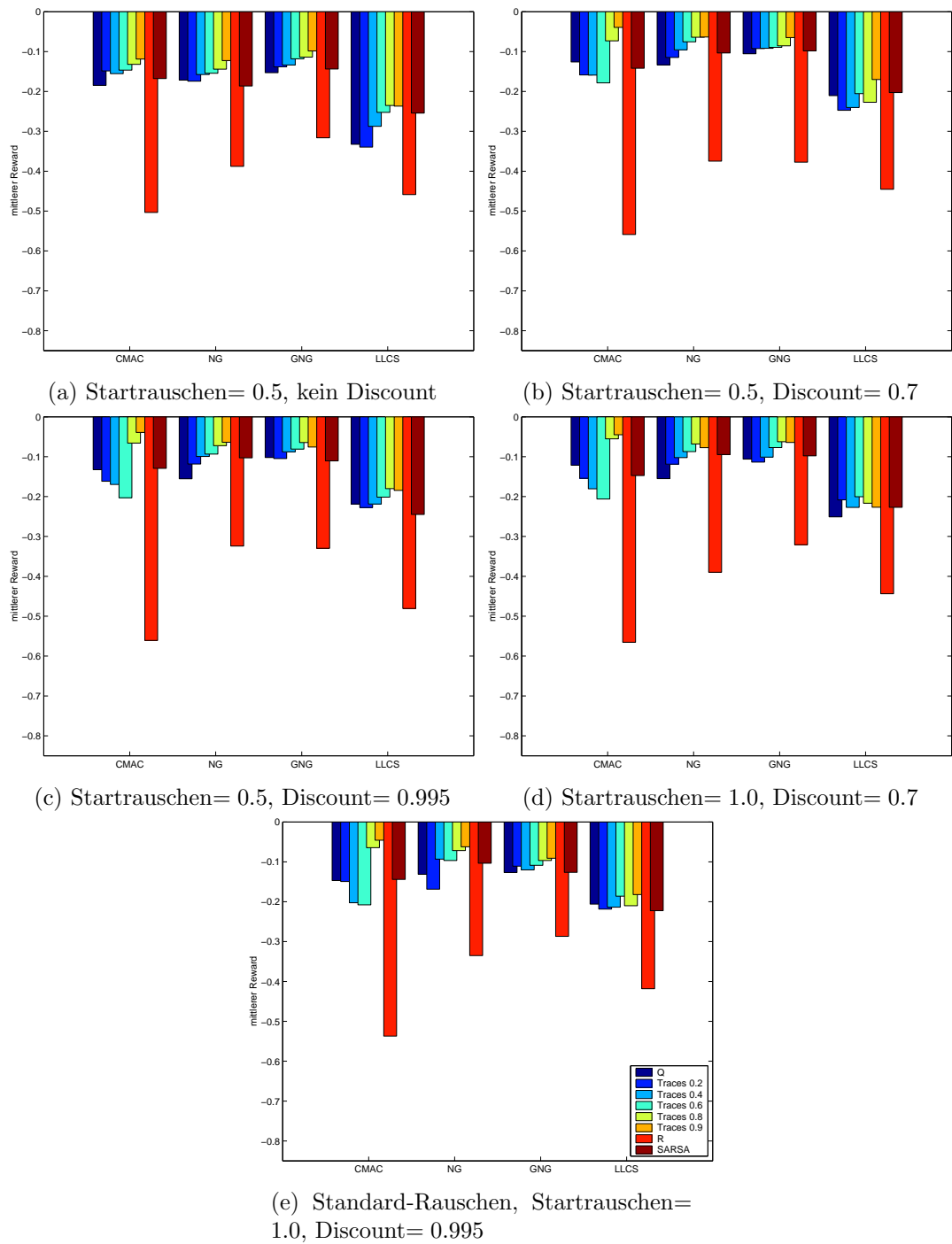


Abb. A.2: Mittlerer Reward aller Agenten bei verschiedenen Rauscheigenschaften im Standard-Szenario mit unterschiedlichem Start- und Zieltal

A. Abbildungen

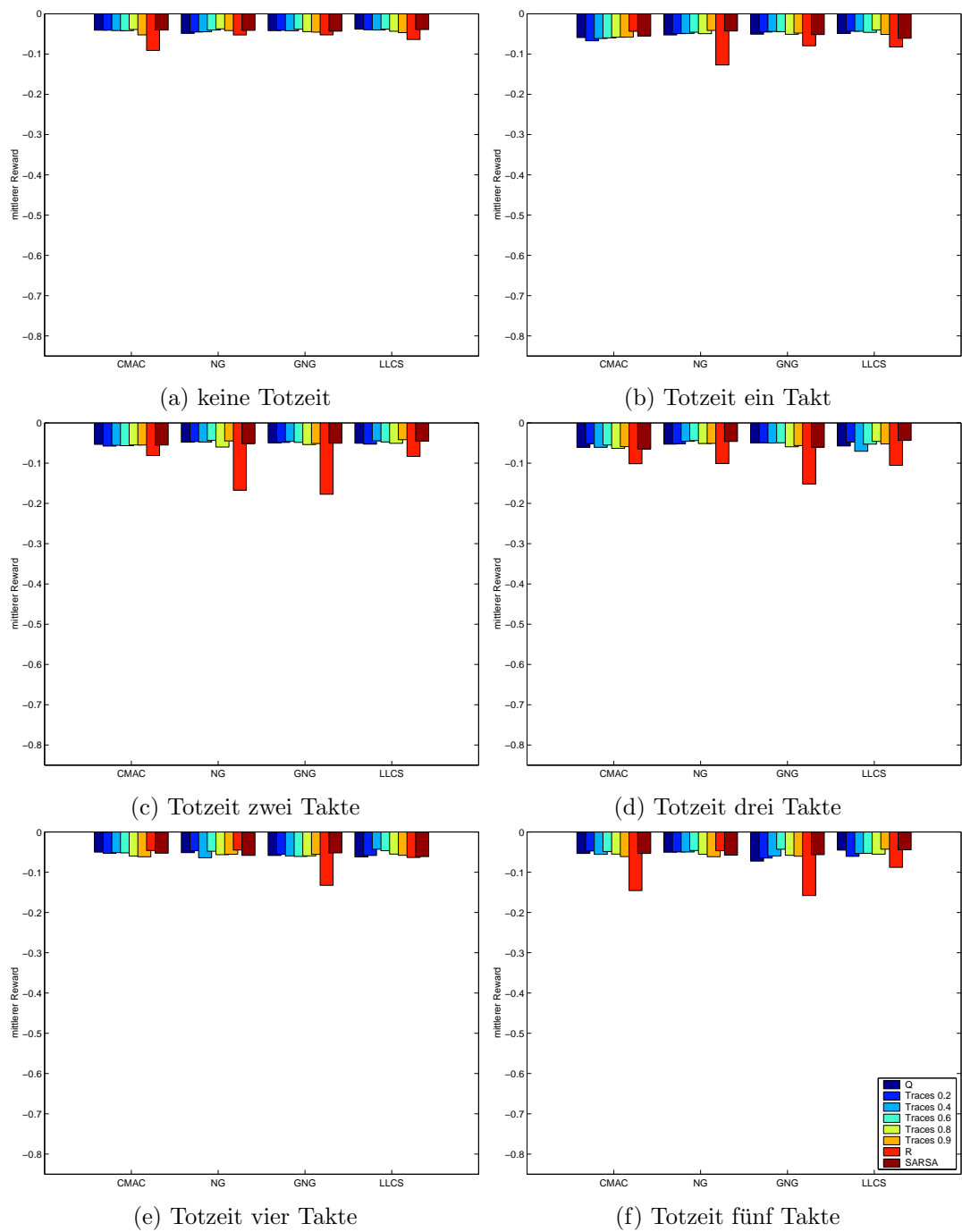


Abb.A.3: Mittlerer Reward aller Agenten bei verschiedenen Totzeiten im Standard-Szenario

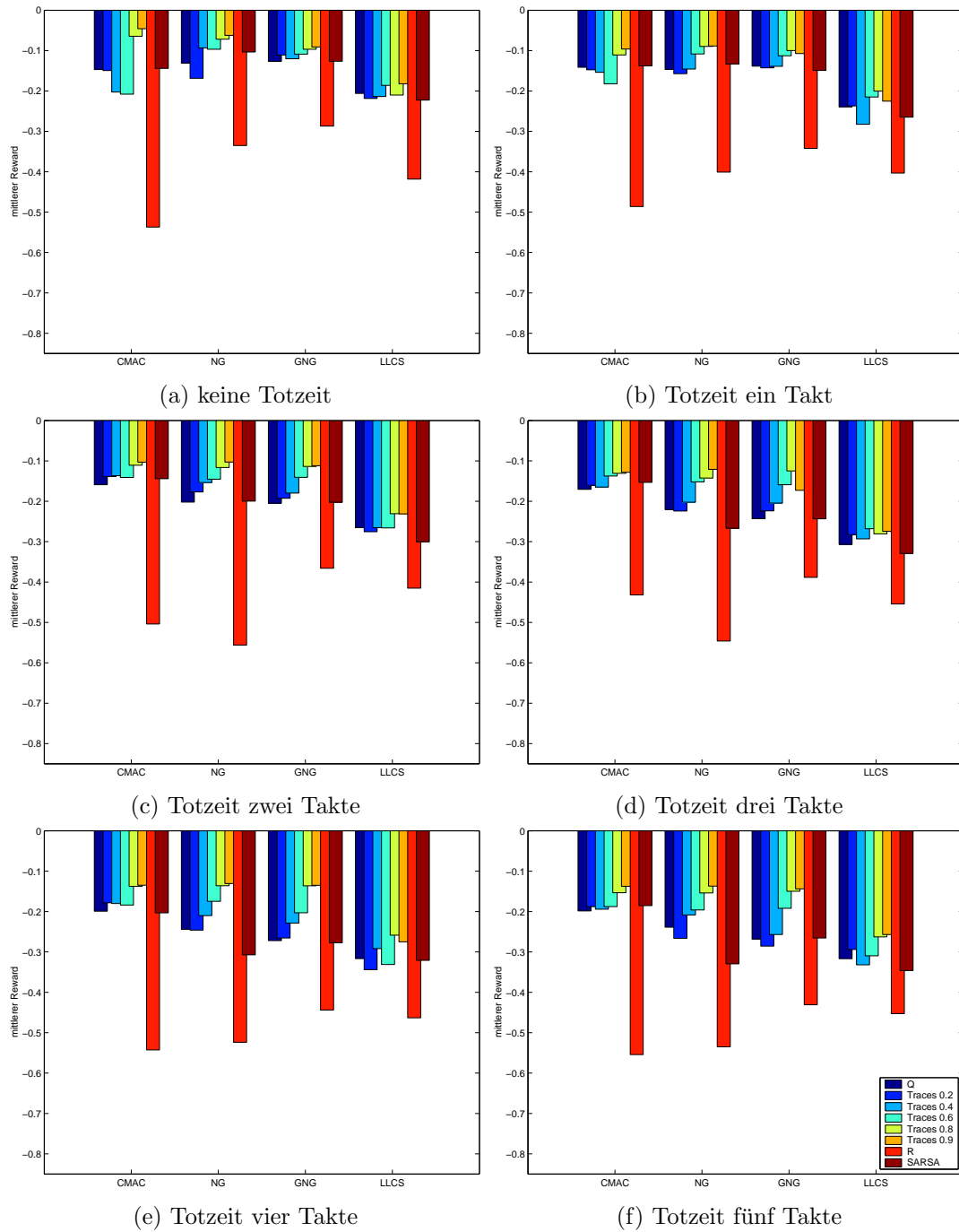


Abb. A.4: Mittlerer Reward aller Agenten bei verschiedenen Totzeiten im Standard-Szenario mit unterschiedlichem Start- und Zieltal

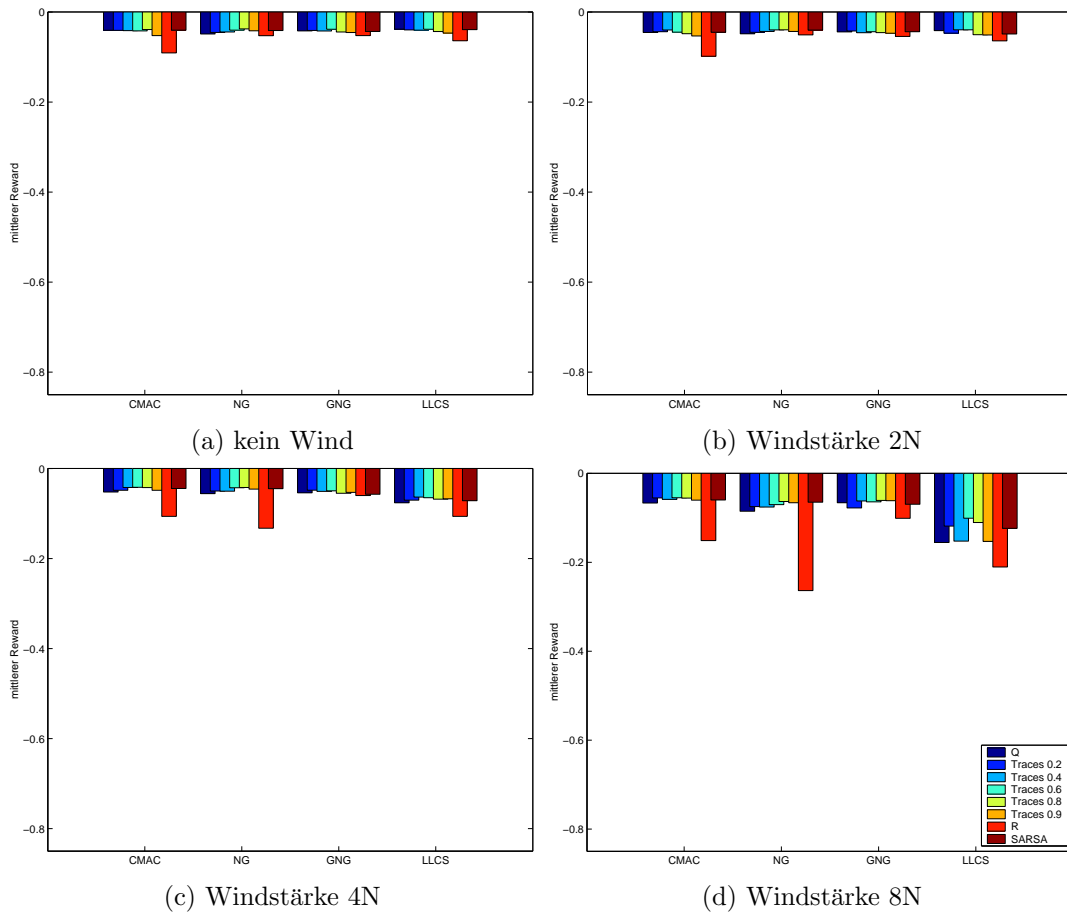


Abb. A.5: Mittlerer Reward aller Agenten bei verschiedener Windstärke im Standard-Szenario, wobei der Wind für den Agenten nicht beobachtbar ist.

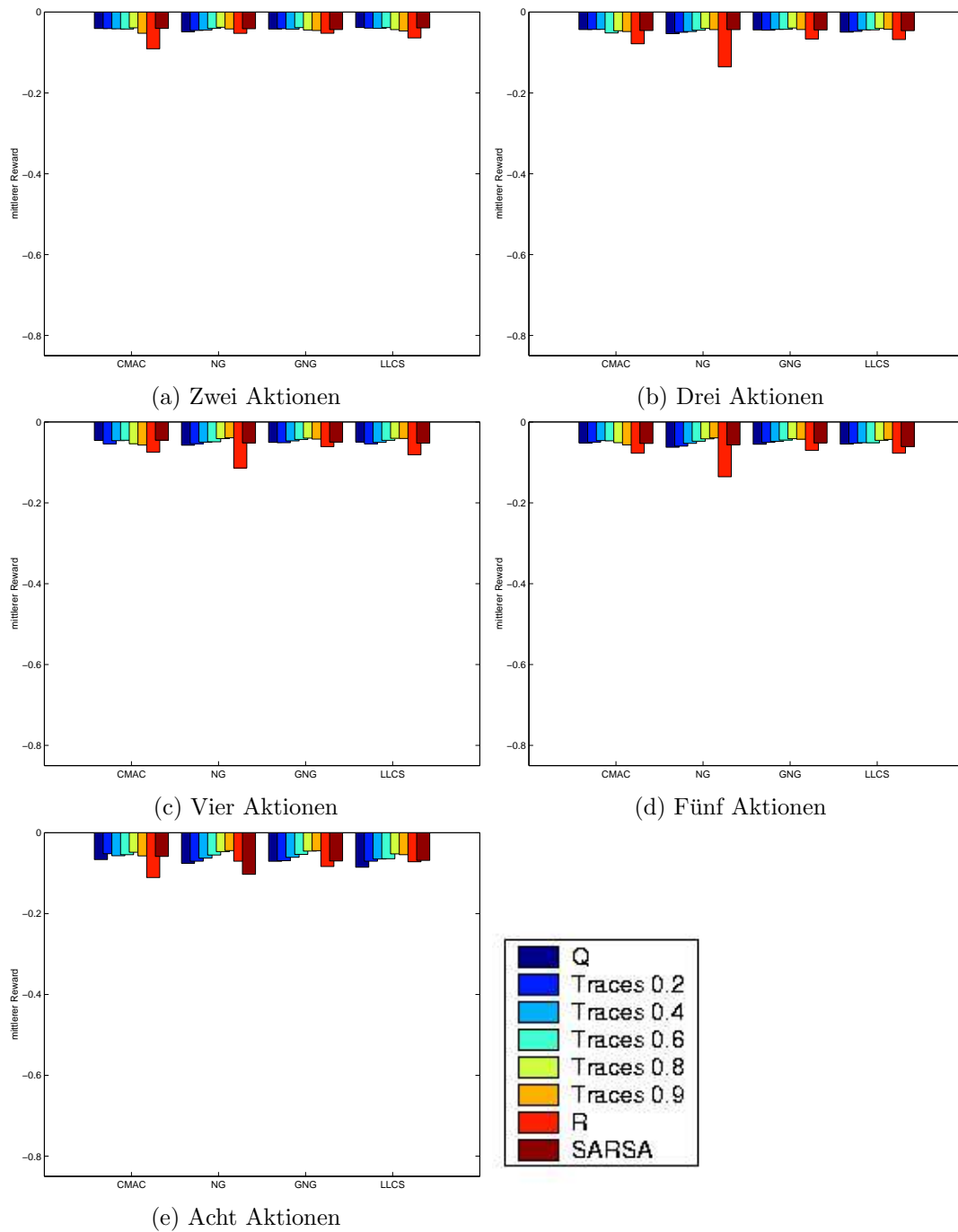


Abb. A.6: Mittlerer Reward aller Agenten mit verschiedener Anzahl von möglichen Aktionen, Standard-Szenario.

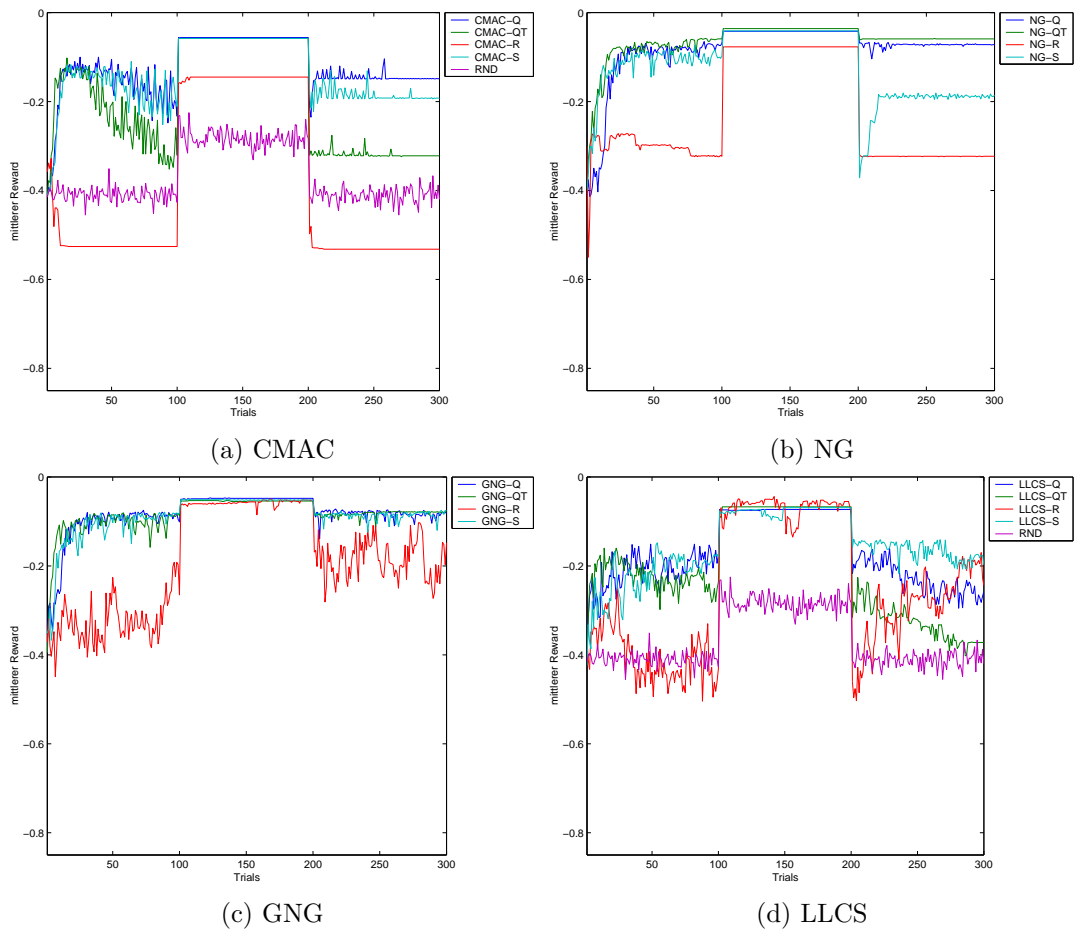


Abb. A.7: Verhalten alle Agenten im Experiment mit konstantem Startpunkt, Wechsel des Zielpunktes nach 100 Trials ins andere Tal und Zurückwechseln nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments

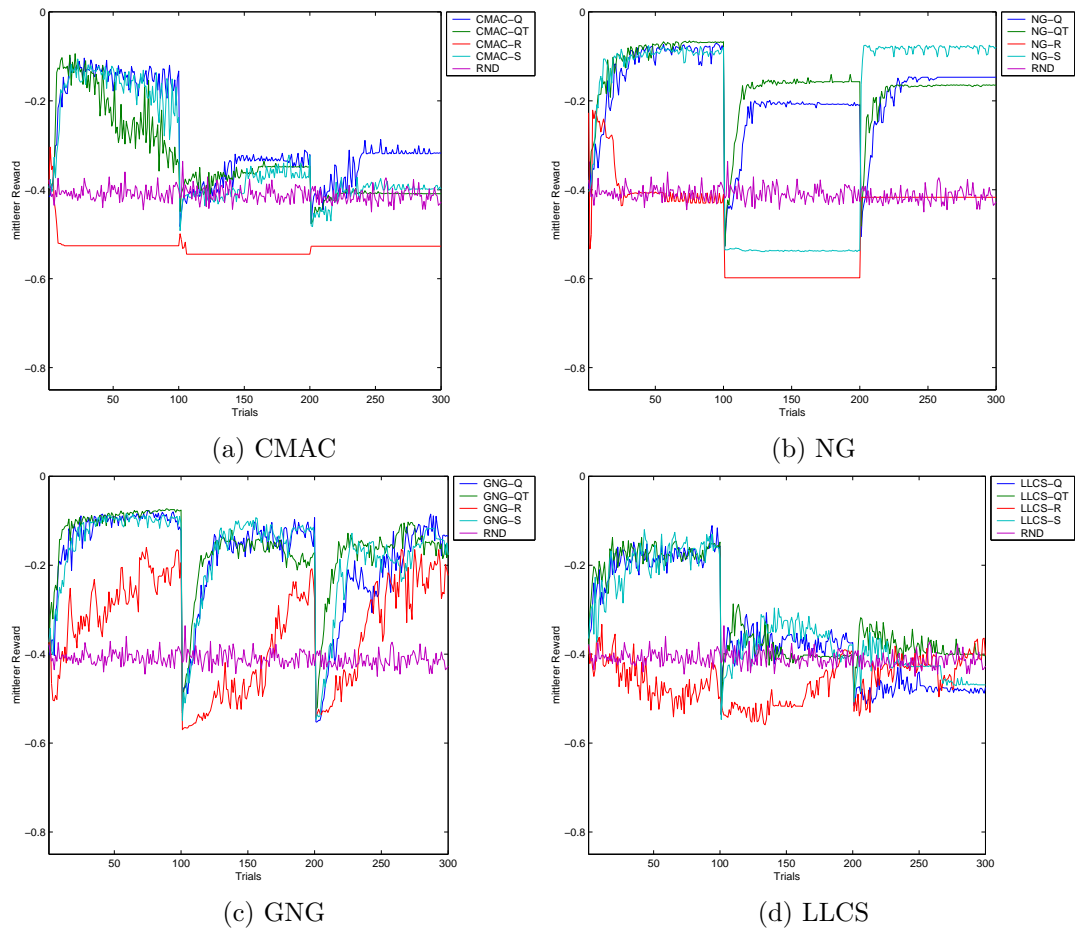


Abb. A.8: Verhalten aller Agenten beim Tausch von Start- und Zielpunkt nach 100 Trials und Zurücktuschen nach 200 Trials, Entwicklung des mittleren Rewards pro Trial im Verlauf des Experiments