Proposal zur Masterarbeit

Kombination der neuroevolutionären Methoden EANT mit Q-Learning und CMA-ES

Tchando Kongue

Einleitung

Neuroevolutionäre Algorithmen sind Methoden, die durch die Benutzung von genetischen Algorithmen künstlichen Neuronalen Netze optimieren. Neuroevolution kann Reinforcement Learning für komplexe Aufgabe annähren, indem bei jeder Generation Individuen mit den meisten Belohnungen am wahrscheinlichsten sich entwickeln und überleben. Im Allgemein wird die Belohnung nicht sehr effizient benutzt, da die Evolution und die Optimierung jedes Individuums sich auf die gesamte Belohnung basiert, und nicht auf dem direkt Reward. Weil Lernen Evolution helfen kann([NF00]), ist eine Kombination von neuroevolutionären Algorithmen mit anderen Verfahren, wo Optimiereung auf direkte Belohnung basiert, wünschwert.

Die Neuroevolutionären Algorithmen sind unter zwei Klasse unterteilt: die Algorithmen, die nur die Gewichten von Netz evolvieren und die Algorithmen, die dazu noch die Topologie des Netzes evolvieren (TWEANNs[1]). TWEANNs Algorithmen sind am meisten benutzt.

In dieser Arbeit wird das neuroevolutionnäre Verfahren EANT erweitern, indem Lernen und Evolution kombinieren werden und/oder indem Spezielle

[1]Topology & Weight Evolving Artificial Neural Networks

Optimierungsverfahren zusammenintegriert werden. Das erweiterte Verfahren wird auf dem Spiel Brio getestet.

Stand der Technik

Es existiert schon mehr neuroevolutionäre Verfahren. Einigen haben die möglichkeiten sich mit Optimierungsverfahren kombinieren zu lassen. Lernen und Evolution ist eine Kombination, die schon bei manchen neuroevolutionären Algorithmen probiert würde. Folgende ist eine Liste von bekannten TWEANNs und ihre Erweiterungen.

- GNARL [SAP94]
  GNARL ist ein neuroevolutionäres Verfahren, wo die Gewichten und die Struktur evolviert werden. Die Anzahl von verstecken Neuronen ist zufällig initialisiert. Während die Mutation wird bei jedem Netz neue Knoten und Links hinzugefügt(bzw. gelöscht). Die Anzahl von Links und Knoten einzufügen(bzw. zu löschen) ist abhängig von der Fitness des Netzes. Eine parametrische Mutation findet auch statt, wobei die Gewichten abhängig von der Fitness zufällig gestört werden.

- EPNet [YL]
  Das Netz in EPnet ist feedforward, EPnet kombiniert Evolution und Lernen. Lernen (in zwei Schritte) wird durch Backpropagation und simulated annealing[YL] gemacht. Der Evolutionsschritt besteht aus mehreren Mutationen. Im gegensatz zu GNARL wird die Evolution nur auf das Netz mit der höchsten Fitness durchgeführt. Wie bei GNARL gibt es keine crossover-Operation. Die Knoten und Verbindungen werden während der Mutation hinzugefügt oder gelöscht.

- NEAT [SM]
  Im gegensatz zu EPnet und GNARL ist NEAT auf einer genetischen Kodierung gebaut. Die Kodiereung unterstützt die Crossover-Operation. Das Netz wächst immer. Bei der strukturellen Mutation werden neue Knoten und/oder Links Hinzugefügt (nicht gelöscht). Das Netz kann recurrente Verbindungen besitzen.

NEAT allein ist ein evolutionäres Verfahren ohne ein explizites Lernen. Die Kombination mit Q-Learning [WS06] ermöglicht das Training von Netz in jeder Generation, das Netz aproximiert die Q-Funktion. Die Korrektur wird mit der Backpropagation durchgeführt.

- EANT [KME08]
  Wie NEAT benutzt EANT eine genetische Kodierung. Die Kodierung -die CGE Kodierung[KEM$^+$07]- hat eine besondere Eigenschaft, sie erlaubt Bewertung von Netzen ohne Dekodierung. Eine andere Eigenschaft von EANT ist der gesamtes Evolutionsschritt, der ist von der Nature inspirieren (Der Explorationsschritt dauert länger als der Exploitationsschritt).

  EANT2 [NTSS] ist eine kombination von EANT und CMA-ES(Abb. 1). CMA-ES wird benutzt um die Parameter zu optimieren (Structural Exploitation).

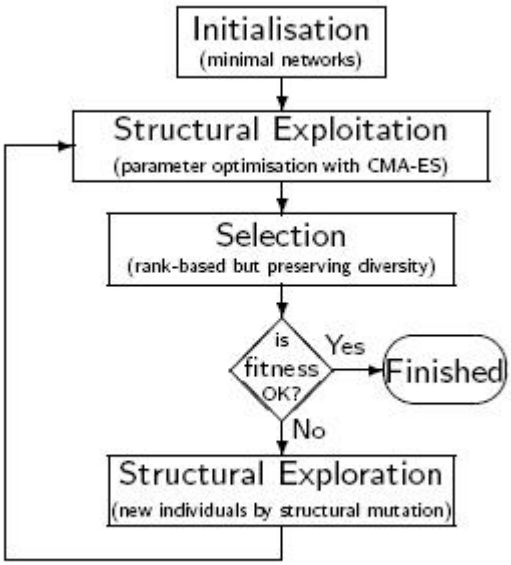

Abbildung 1: EANT+CMA-ES

geplante Arbeit

In der Arbeit geht es darum, ein bestehendes neuroevolutionäres Algorithmus für eine bestimte Aufgabe(BRIO Labyrinth) zu optimieren. durch eine mögliche effiziente Nutzung den Gewichten von Netze.

EANT+Q

Wir werden zuerst EANT und Q-learning versuchen zu kombinieren. Die Kombination wird die Nutzung des direkten Belohnung ermöglichen. Ein EANT-netz kann normalerweisse rekurrente Verbindungen enthalten, aber damit Q-learning genaue von einem Netz abgebildet wird, musst das Netz am besten Feedforward sein, da die Q-Funktion ein Markow-Prozess bildet. Die Backpropagation Algorithmus wird benutzt um das Netz zu trainieren.

EANT+CMA-ES

Im zweiten Versuch werden wir CMA-ES und EANT kombiniert[NTSS]. Dabei soll sich die Optimierung auf die gesamte Belohnung basieren. Das originale EANT passt die Parameter zufällig an. Die CMA(Covariance Matrix Adaptation) ist eine Methode, die die Kovarianzmatrix von multivariater Verteilung adaptiert. In der Kombination wird CMA-ES die Kovarianzmatrix der parametrischen Mutationsverteillung adaptiert, sodass die zufällige Anpassung von Parameter in eine bestimme Richtung durchgeführt wird[Han].

Testbed:BRIO

Um die ganze Implementierung auf das BRIO-Simulator zu testen werden wir einige Parameter wiederdefinieren. Die von Abdenebaoui[AKKK07] definierte Zustand und Aktionsraum des Simulator werden erweitert.

* Man könnte Ein Pfad (Linie zum Beispiel) durch das Labyrinth bauen lassen. Ein Subgoal des Spiels wäre auf dieses Pfad zu bleiben und sich

gleichzeitig nach vorne zu bewegen (bzw. sich von dem Anfangspunkt zu entfernen). Das Reward wäre umgekehrt proportionnal zu Abstand zu dem Weg, und zu dem Winkel zwischen Wegsrichtung und Bewegungsrichtung (Geschwindigkeitsvektor).

* Man könnte auch einfach versuchen sich das Ziel zu nahen und sich vom Anfangspunkt zu entfernen (Entfernung als Euklidische Abstand). Die Belohnung bei jeder Zustand wäre dann abhängig von der Position des Ball, relativ zum Ziel und Anfangspunkt. Die Fortbewegung wird natürlich als Belohnung erzwungen.

Zeitplan

Die Abbildung 2 zeigt den Zeitplan. Das grosse Teil der Literaturrecherche ist schon gemacht. Das Schreiben des Bericht wird parallel mit fast alle andere Teile durchgeführt.

- Literaturrecherche (2 Wochen)
- Optimierung von EANT untersuchen (5 Wochen)
- Optimierung implementieren und auf standarten Systems testen(single Pole balancing ...) (8 Wochen)
- Parameter/Umgebung (Input, output, reward ...) definition für den Test auf das BRIO-Labyrinth (5 Wochen)
- Bericht schreiben (18 Wochen)

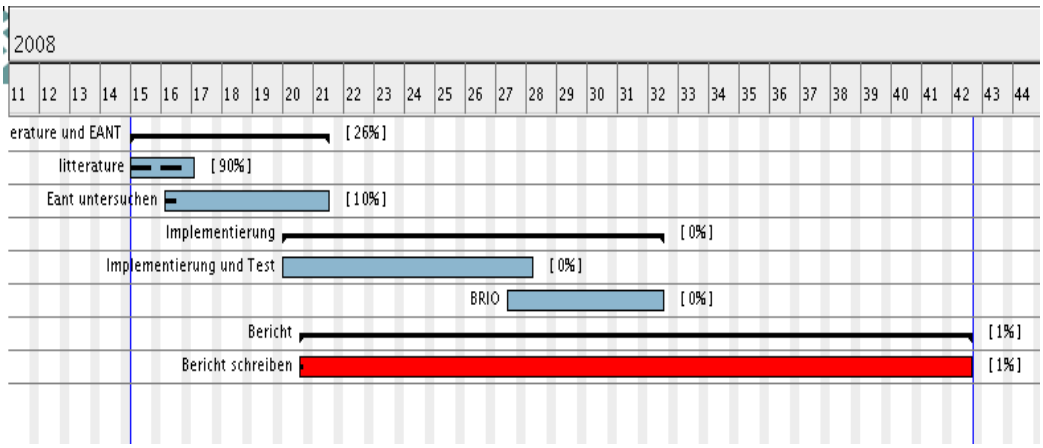

Abbildung 2: Plan

Literatur

[AKKK07] Larbi Abdenebaoui, Elsa Kirchner, Y. Kassahun, and Frank Kirchner. A connectionist architecture for learning to play a simulated brio labyrinth game. In *Proceedings of the 30th Annual German Conference on Artificial Intelligence (KI07)*, volume 4667/2007 of *Computer Science*, pages 427–430, Osnabrck, Germany, 9 2007. Springer.

[Han] Nikolaus Hansen. The cma evolution strategy: A tutorial.

[KEM+07] Y. Kassahun, M. Edgington, Jan Hendrik Metzen, G. Sommer, and Frank Kirchner. A common genetic encoding for both direct and indirect encodings of networks. In *Genetic and Evolutionary Computation Conference (GECCO–2007)*, pages 1029–1036, 0 2007.

[KME08] Y. Kassahun, J. H. Metzen, and M. Edgington. *Computational Intelligence in Autonomous Robotic Systems*, chapter Incremental acquisition of neural structures through evolution, page n/a. Studies in Computational Intelligence. Springer Verlag, 2008. Accepted.

[NF00] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, USA, 2000.

[NTSS] Jochen Krause Nils T Siebel and Gerald Sommer. Efficient learning of neural networks with evolutionary algorithms.

[SAP94] Gregory M. Saunders, Peter J. Angeline, and Jordan B. Pollack. Structural and behavioral evolution of recurrent networks. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 88–95. Morgan Kaufmann Publishers, Inc., 1994.

[SM] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies.

[WS06] Shimon Whiteson and Peter Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, May 2006.

[YL] Xin Yao and Yong Liu. Evolving artificial neural networks through evolutionary programming.