Georg-August-Universität
Göttingen
Institut für Informatik

Lars Runge
SS 2012
Matr.Nr.: 21027257

Hausarbeit

im Studiengang „Angewandte Informatik“

Gerichtete Evolution von Kommunikation und Kooperation in digitalen Organismen

Proseminar: „Artificial Life“
Dozent: Prof. Dr. Winfried Kurth

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Diese Arbeit beschäftigt sich mit dem Thema der zielgerichteten Evolution von digitalen Organismen zur Erschaffung von Kommunikation und kooperativen Verhalten. Dabei wird besonderer Wert auf die Anpassungsfähigkeit der Organismen auf veränderbare Umweltbedingungen gelegt, denn Interaktionen zwischen Computersysteme und der echten Welt werden immer häufiger und wichtiger. Deshalb ist es notwendig dass verschiedene Computersysteme mit unterschiedlichen Eigenschaften und Möglichkeiten zuverlässig miteinander kommunizieren können, auch wenn das Umfeld sich verändert.
Die Aufgabenstellung ist die Erschaffung von digitalen Organismen durch Evolution, die sich soweit entwickeln, dass sie in der Lage sind eine komplexe Aufgabe durch Zusammenarbeit der ganzen Bevölkerung zu lösen. Die spezielle Aufgabe ist die Bestimmung des größten Wertes, der durch die Individuen wahrgenommen wird. Dazu gehört die Wahrnehmung des individuellen Wertes und dessen Verbreitung, sowie der Vergleich mit den Werten anderer Individuen und die Bestimmung des maximalen Wertes. Dieses Verhalten könnte zum Beispiel dazu genutzt werden, um einen Anführer in der Bevölkerung zu wählen oder den größten Wert in einem Funknetzwerk zu bestimmen.
Für die Simulation wird die AVIDA Plattform genutzt mit der eine virtuelle Umgebung, in der digitale Organismen leben, erschaffen und nach Wünschen konfiguriert werden kann.
Die Ergebnisse werden zeigen, dass digitale Evolution eine mögliche Vorgehensweise ist solches Verhalten zu erschaffen und eventuell Lösungen hervorzubringen, die nicht intuitiv von Menschen entwickelt worden wären. Diese Ergebnisse mögen vielleicht nicht optimal sein, wenn man die Aufgabenstellung isoliert betrachtet, aber sie besitzen andere wichtige Eigenschaften wie z.B. Resistenz gegen Umweltänderungen und Angreifer oder Selbstheilung. Genauso wie lebende Organismen diese Fähigkeiten über die Zeit entwickelt haben. (vgl. [1], S. 385)

1.2 Aufbau

Die Ausarbeitung nimmt als Grundlage das Paper „Directed Evolution of Communication and Cooperation in Digital Organisms“ von David B. Knoester, Philip K. McKinley, Benjamin Beckmann und Charles Ofria. Es werden zusätzliche Informationen in manchen Bereichen hinzugefügt und versucht die Vorgehensweise zu verdeutlichen.
In Kapitel 2 werden die Grundlagen des AVIDA Systems, die essentiellen Abläufe und Einstellungen erläutert. Dabei wird auf die Konstruktion von Organismen und Umfeld sowie auf wichtige Mutationen bei der Fortpflanzung eingegangen. Ebenfalls wird die Aufteilung der

Rechenzeit auf die Organismen und deren Bewertung behandelt. Kapitel 3 beinhaltet die Darstellung der verschiedenen Versuche und erläutert dabei die Herangehensweise und Ergebnisse der Simulationen. Danach werden die Ergebnisse dargebracht, das Fazit gezogen sowie ein Ausblick auf die Verwendung in zukünftigen Experimenten gegeben. Weitere Möglichkeiten und Studien mit der AVIDA Plattform werden aufgezeigt aber nicht im Detail erklärt.

2 AVIDA

2.1 Grundlagen

AVIDA ist eine Software Plattform für Experimente mit selbst-reproduzierenden und -entwickelnden Computerprogrammen. Sie wurde 1993 von Charles Ofria, C. Titus Brown und Chris Adami entwickelt, um ein großes Hindernis bei der Erforschung von evolutionären Prozessen zu überwinden: die Zeit. Da Evolution sehr lange dauert, kann man in der Realität nur kontrollierte evolutionäre Experimente mit Organismen durchführen, die eine kurze Lebensspanne haben, damit die Bevölkerung hunderte von Generationen in nur wenigen Monaten oder Jahren durchlaufen kann. Die Plattform soll eine möglichst detaillierte Erforschung von digitalen Organismen ermöglichen, mit dem Vorteil, dass tausende von Generationen in einem Bruchteil der Zeit vorübergehen und die Daten leichter und genauer gemessen werden können. (vgl. [2], S. 191)

Dazu besteht AVIDA aus drei Hauptbestandteilen. Die erste ist der AVIDA Kern, der die Bevölkerung der digitalen Organismen, das simulierte Umgebung und die Ressourcen mit denen die Organismen interagieren können aufbaut und kontrolliert. Er enthält ebenso einen Scheduler, der für die Zuteilung der Rechenzeit an die einzelnen Organismen zuständig ist. Der zweite Bestandteil ist die graphische Oberfläche zum bedienen der Software und der dritte Bestandteil beinhaltet eine Vielzahl an analytischen und statistischen Werkzeugen, die Daten über die Bevölkerung sammeln und weitere Verwendungszwecke ermöglichen. (vgl. [2], S. 193-194)

2.2 Aufbau von Organismen und Umgebung

Jeder einzelne digitale Organismus ist unabhängig von dem Rest der Bevölkerung und besteht aus einem virtuellen CPU, einer Befehlsliste und drei Registern, zwei Stacks und vier Heads, die die Funktion von Befehl- und Stackzeigern ausführen. Zusätzlich besitzt jeder Organismus einen Input- und Output-Buffer, die zur Interaktion mit dem Umfeld genutzt werden können. Die Instruktionsfolgen der Organismen, oder auch Genom genannt, sind in einer Sprache verfasst, die Assembler sehr nahe kommt. AVIDA unterstützt nicht nur eine Sprache sondern viele, die jeweils Vorteile in bestimmten Einsatzgebieten mit sich bringen. Die Befehlsliste eines Organismus ist, wie man in Abbildung 1 erkennt, zirkulär aufgebaut mit einem festgelegten Anfang und Ende, die bei der Fortpflanzung eine Rolle spielen.
Außerdem ist auf Abb. 1 zu erkennen, dass Organismen nur auf bestimmten Stellen auf der Umgebung leben, den sogenannten Zellen. In jeder Zelle kann zur gleichen Zeit nur ein Organismus leben, wird ein neuer Organismus in dieser Zelle erschaffen, so wird der alte überschrieben. Die Zellen sind netzartig untereinander verbunden und haben eine bestimmte

Ausrichtung zu einer anderen Zelle. Über diese Ausrichtung können dann von den Organismen Nachrichten an benachbarten Zellen gesendet werden. Beinhaltet die benachrichtigte Zelle einen Organismus so wird die Nachricht in dessen Input-Buffer hinterlegt, ansonsten geht die Nachricht verloren. Außerdem kann die Ausrichtung der Zelle von dem Organismus zum Beispiel mit den Instruktionen „GET-FACING“ bestimmt und mit „ROTATE-{L,R}“ verändert werden. Zum Schluss besitzt jede Zelle eine eindeutige zufällige 32-Bit Zahl, die Zell-ID, die ebenfalls mit der Instruktion „GET-ID“ bestimmten werden kann und zur eindeutigen Identifikation dienen kann. (vgl. [1], S.385)

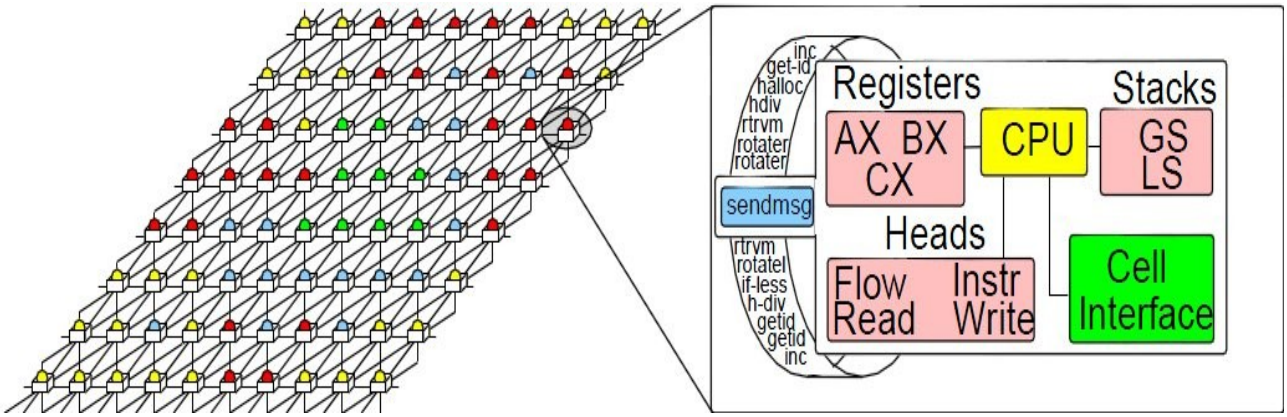

Abbildung 1: Struktureller Aufbau einer AVIDA-Simulation *Quelle: [1], S.386*

2.3 Reproduktion

Die Simulation startet mit einem einzigen Organismus, der nur in der Lage ist sich zu reproduzieren. Dieser Prozess wird durch die sogenannten „Copy-Loop“ ausgeführt, die mit der Instruktion „h-alloc“ beginnt und den Speicher des Organismus verdoppelt. Dieser neue Speicher wird zwischen der Start- und Endmarkierung angelegt und wird ebenfalls ausgeführt sofern Instruktionen enthalten sind. Damit dieser Speicher gefüllt wird, werden nun solange mit der Instruktion „h-copy“ Instruktionen hineinkopiert bis optimaler Weise das vollständige Genom exakt kopiert wurde. Danach wird mit der Instruktion „h-divide“ der Speicher wieder halbiert und das neue Genom an die AVIDA Welt übergeben, die es dann nach der eingestellten Ersetzungsstrategie in eine Zelle platziert. Abbildung 2 zeigt die Verwaltung des Speicherbereichs in der „Copy-Loop“.

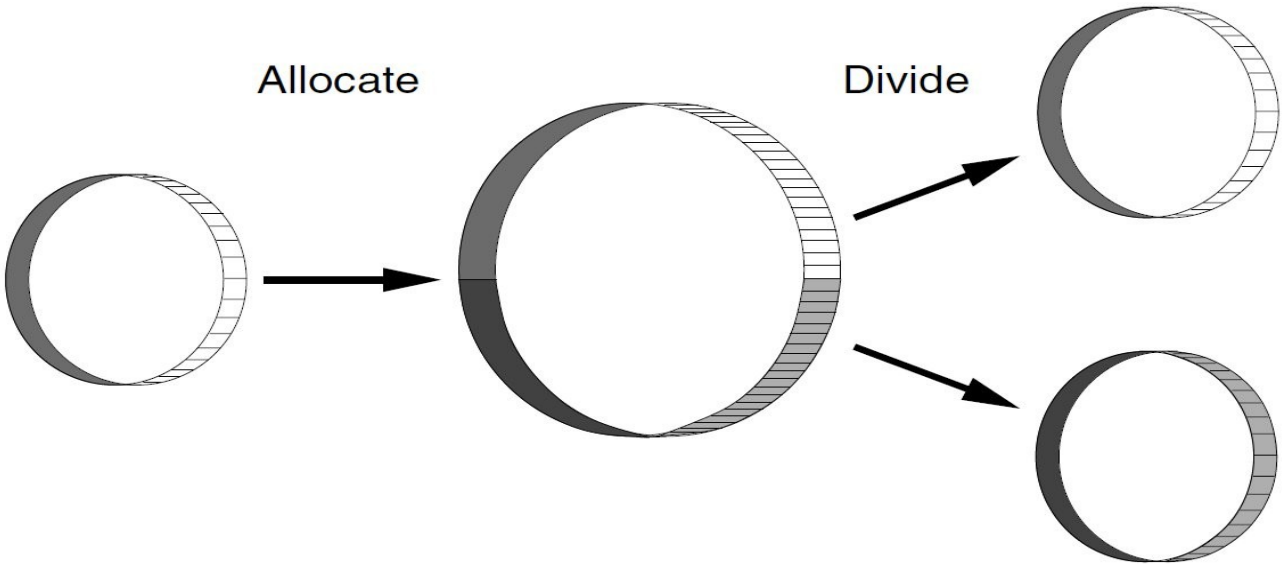

Abbildung 2: Aufbau und Zerteilung des Speichers bei Reproduktion *Quelle: [2], S. 200*

Es gibt verschiedene Ersetzungsstrategien wie zum Beispiel „MASS-ACTION“, die eine Zelle zufällig aus allen Zellen der Welt auswählt, oder „NEIGHBORHOOD“, die eine Zelle zufällig aus den benachbarten Zellen wählt. Ersetzung des ältesten Nachbarn oder andere komplexere Strategien sind ebenfalls möglich, werden aber in den folgenden Versuchen nicht benutzt.
Damit die Reproduktion möglichst fehlerfreie Nachkommen erzeugt, werden ein paar Voraussetzungen an „h-divide“ gestellt. Die wichtigsten sind, dass sowohl Eltern- als auch Nachkommen-Genom wenigstens 10 Instruktionen nach der Trennung beinhalten müssen und dass mehr als die Hälfte des Eltern-Genoms nach der „h-alloc“ Instruktion ausgeführt worden ist. Außerdem muss der neue Speicher mindestens zur Hälfte gefüllt worden sein. Dadurch wird verhindert, dass zu viele „Frühgeburten“ entstehen, die zu klein sind um richtig zu funktionieren und meistens bei der Trennung auch den Elternteil funktionsunfähig machen. Nach der Trennung werden die Stacks und Register des Nachkommen als auch des Eltern-Organismus gelöscht.

2.4 Mutationen

Damit bei der Fortpflanzung evolutionäre Entwicklungen entstehen, werden zufällige Mutationen an die „h-copy“ Instruktion angeknüpft. So kann es bei jeder Ausführung von „h-copy“ nach einer bestimmten Wahrscheinlichkeit geschehen, dass die kopierte Instruktion durch eine zufällige Instruktion aus der zuvor zusammengestellten Liste aller Befehle ersetzt wird. Zusätzlich besteht die Möglichkeit, dass nach der Ausführung von „h-divide“ eine zufällige Instruktion hinzukommt oder gelöscht wird. Diese Wahrscheinlichkeiten können zuvor vom Benutzer festgelegt werden.
Außerdem gibt es eher unkontrollierbare Mutationen wie zum Beispiel die Punktmutation, die eine zufällige Änderung im Speicher eines Organismus durchführt und immer wahrscheinlicher wird desto länger dieser läuft, sowie implizite Mutationen, die durch fehlerhafte „Copy-Loop“s entstehen

und zum Beispiel einen Bereich des Genoms doppelt kopieren oder auslassen. Letztere können zu Beginn der Simulation durch die „FAIL-IMPLICIT“-Funktion ausgeschlossen werden, die alle Nachkommen bei der Erstellung überprüft und bei Abweichungen vom Eltern-Genom, die ohne kontrollierbare Mutationen entstanden sind, löscht. (vgl. [2], S. 200-201)

2.5 Besonderheiten der Speicherverweisung

Eindeutige Verweise an Stellen im Genom sind durch das mutationsbedingte Hinzufügen und Entfernen von Instruktionen nur schwer von Eltern auf die Nachkommen zu übertragen. Damit Sprünge und Schleifen bei sich verändernden Genomen weiterhin funktionieren wird ein sogenanntes „template matching“ System verwendet, dass ähnlich wie Label in anderen Sprachen aufgebaut ist. Wichtig hierfür sind die drei „no operation“-Instruktionen, kurz nop-Instruktionen. Sie führen keine Berechnung im CPU durch, sondern sind in komplementären Paaren aufgebaut, die auf sich verweisen. Die Instruktion „nop-A“ ist das Komplement zu „nop-B“, während „nop-B“ zu „nop-C“ komplementär ist und „nop-C“ wiederum zu „nop-A“. Ein Template ist dann eine Aneinanderreihung von „nop“-Instruktionen. Soll gesprungen werden, so wird mit „h-search“ das Komplement des anschließenden Templates gesucht und falls dieser existiert so wird der „Flow Control“-Head auf diese Stelle gesetzt. Die Instruktion „mov-head“ setzt dann dem „Instruction“-Head an diese Stelle. Diese Technik kann zwar nur immer eine Stelle im Speicher als Label zur gleichen Zeit speichern, aber sie ist größtenteils immun gegen Mutationen, sofern kein Template durch eine zufällige Mutation zerstört wird. Außerdem können die nop-Instruktionen auch als Argumente verwendet werden, um die drei Stacks anzusprechen. Zum Beispiel kann mit „nop-A“ nach der „inc“-Instruktion der Register „AX“ angesprochen werden, der dann um 1 erhöht wird.

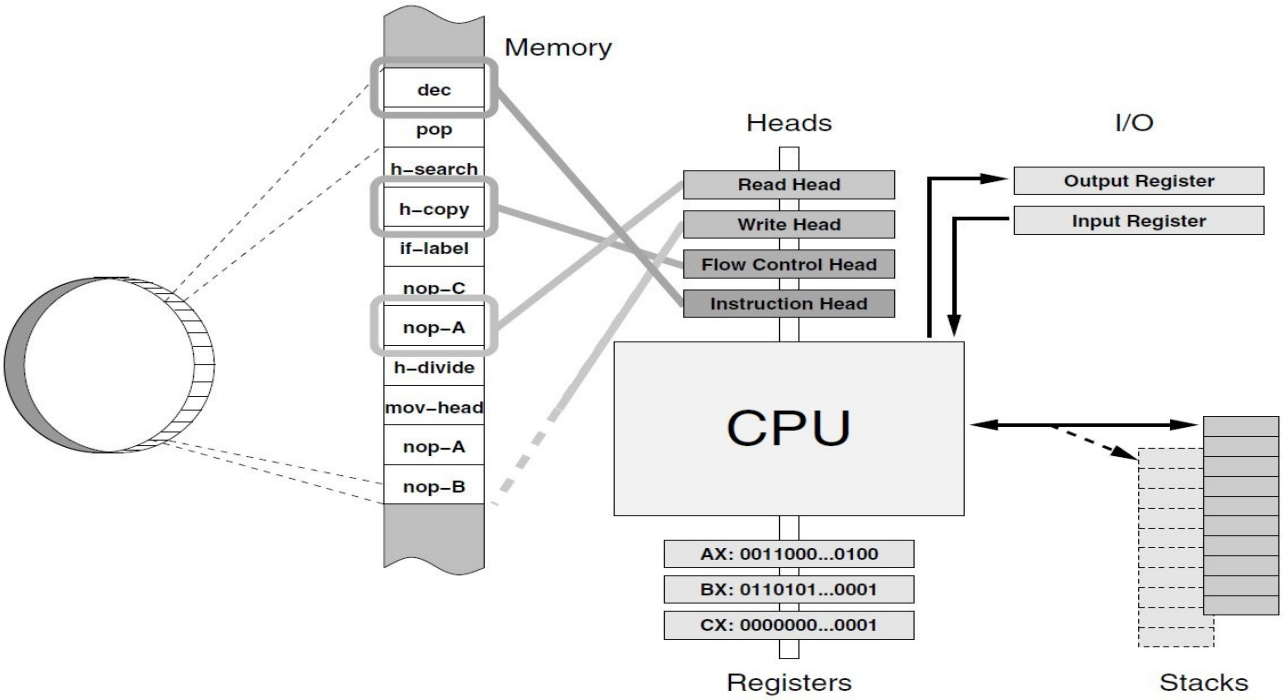

Abbildung 3: Detaillierter Aufbau eines digitalen Organismus *Quelle: [2], S. 195*

2.5 Tasks und Leistung von Organismen

Damit Organismen bewertet werden können ob sie sich in die gewünscht Richtung entwickeln, können in der AVIDA Umgebung globale Tasks an die mutierenden Organismen gestellt werden. Diese werden durch den Benutzer vor dem Versuch definiert und belohnen oder bestrafen Organismen individuell dafür wenn ihr Verhalten eine Task erfüllt. Mehrere Tasks können zu komplexeren Aufgaben kombiniert werden und das Lösen verschiedener Tasks wird multiplikativ belohnt und ist daher besonders erstrebenswert. Hierbei ist zu beachten, dass nur der Phänotyp eines Organismus bewertet wird, also nur Aktionen die von außerhalb des Organismus zu erkennen sind. Damit ist es zuerst irrelevant, wie der Organismus das Verhalten intern produziert und man erhält eine sehr große Gestaltungsfreiheit zur Lösung der Tasks. (vgl. [1], S. 386)
Die Leistung eines Organismus wird dann durch die Anzahl der erfüllten Aufgaben bestimmt und beeinflusst die Menge an Instruktionen, die dieser Organismus im Vergleich zu den übrigen berechnen darf. Die Übersetzung der Relation ist simpel, denn wenn Organismus A doppelt so viel Leistung hat wie Organismus B, so darf er doppelt so viele Instruktionen in der gleichen Zeit ausführen. Somit pflanzt sich ein Organismus mit einer höheren Leistung schneller fort als ein Organismus mit niedriger Leistung, wodurch letztendlich dieser Genotyp nach relativ kurzer Zeit das Simulationsfeld dominieren wird. Allerdings werden die Veränderungen in der Leistung nicht, wie bei Fitness-Funktionen, in diskreten Zeitschritten berechnet, sondern asynchron individuell nach der Erfüllung einer Task. Dies führt dazu, dass Organismen, die eine Aufgabe in weniger Rechenschritten lösen als Andere, mehr für diese belohnt werden. Somit wird Effizienz belohnt und dies führt zu einer mehr realistischen Simulation sowie einem leicht besserem Ergebnis, wie später noch verdeutlicht wird.

3 Experimente

3.1 Experimenteller Aufbau

Für jeden der folgenden Versuche wurden die selben grundsätzlichen Einstellungen für AVIDA benutzt. Es gibt 3600 Zellen in einem 60 mal 60 Feld und die Versuche laufen jeweils 100.000 Updates, wobei ein Update im Durchschnitt 30 Instruktionen pro Organismus entspricht. Die Mutationsrate für „h-copy“ liegt bei 0.75% und für „h-divide“ bei 5%, dies sind die Grundeinstellungen von AVIDA. Für die Ersetzungsstrategie wird zu Beginn „MASS-ACTION“ gewählt, dies wird aber später nochmal relevant. Jeder Versuchsaufbau wird 20 mal durchgeführt und der Durchschnitt der Ergebnisse betrachtet, um stochastische Abweichungen zu entfernen, die durch die zufälligen Mutationen entstehen. (vgl. [1], S.387)

Zur Lösung der Aufgabenstellung wurde eine Reihe an AVIDA Tasks entwickelt, die bei den Versuchen in unterschiedlichen Kombinationen zum Einsatz kommen. Alle Tasks sind zur Erzeugung des selben Verhaltens entwickelt worden, nämlich die Verbreitung der größten Zell-ID in der Bevölkerung. Dazu müssen die Organismen sich so weit entwickeln, dass sie Zell-IDs von normalen Zahlen unterscheiden, diese Versenden und Erhalten können und durch Vergleiche die Größte unter diesen bestimmen können. Da sie sich zu Beginn der Simulation nur fortpflanzen können, muss dieses Verhalten allein durch Evolution entwickelt werden. Dafür belohnt die Task „SEND-SELF“ Organismen, die ihre eigene Zell-ID versenden, wohingegen die Task „SEND-ID“ jene Organismen belohnt, die überhaupt eine Zell-ID versenden. Eine komplexere Task „MAX-KNOWN“ belohnt Organismen dafür die größte Zahl in ihrem Speicher zu versendet. Dazu zählt die eigene Zell-ID und alle bisher erhaltenden Nachrichten, wobei diese nicht unbedingt Zell-IDs enthalten müssen. Außerdem kann diese Aufgabe erst erfüllt werden nachdem mindestens eine Nachricht angenommen wurde. Die letzte Task „SEND-NON-ID“ bestraft Organismen dafür eine Nachricht zu senden, die keine Zell-ID enthält.

Zum Schluss wurde noch ein Event „RESET-ID“ definiert, das nach der Hälfte der Simulation die größte Zell-ID löscht und in einen zufälligen kleineren Wert abändert. Damit soll getestet werden, falls es eine Kombination von Tasks gibt die das gewünschte Verhalten erzeugt, ob sich die Bevölkerung der Organismen von dieser Umgebungsänderung erholen können.

3.2 Versuch 1: Filterung von Nachrichten

In der ersten Testreihe wird der Schwerpunkt darauf gelegt zu testen, ob die Organismen durch Evolution in der Lage sein werden Nachrichten effektiv zu filtern. Es werden die Tasks „SEND-ID“ und „MAX-KNOWN“ verwendet, in der Hoffnung, dass die Belohnung für Senden von Zell-IDs und das Senden von größeren Zahlen als die eigene Zell-ID am Ende dafür sorgt, dass nur noch die größte Zell-ID versendet wird. Abbildung 4 (a) zeigt die Ergebnisse von den 20 Versuchsläufen und (b) die Anzahl an Organismen, die die verwendeten Tasks erfüllen. „Total“ zeigt die Anzahl aller versendeten Nachrichten, während „Carry“ die Zell-ID tragenden Nachrichten und „ID“ die Menge an versendeten Nachrichten, die die Zell-ID des Versenders beinhalten, anzeigen. „>ID“ markiert die Anzahl an Nachrichten, die größere Zell-IDs als die ihrer Versender enthalten. (vgl. [1], S. 388)

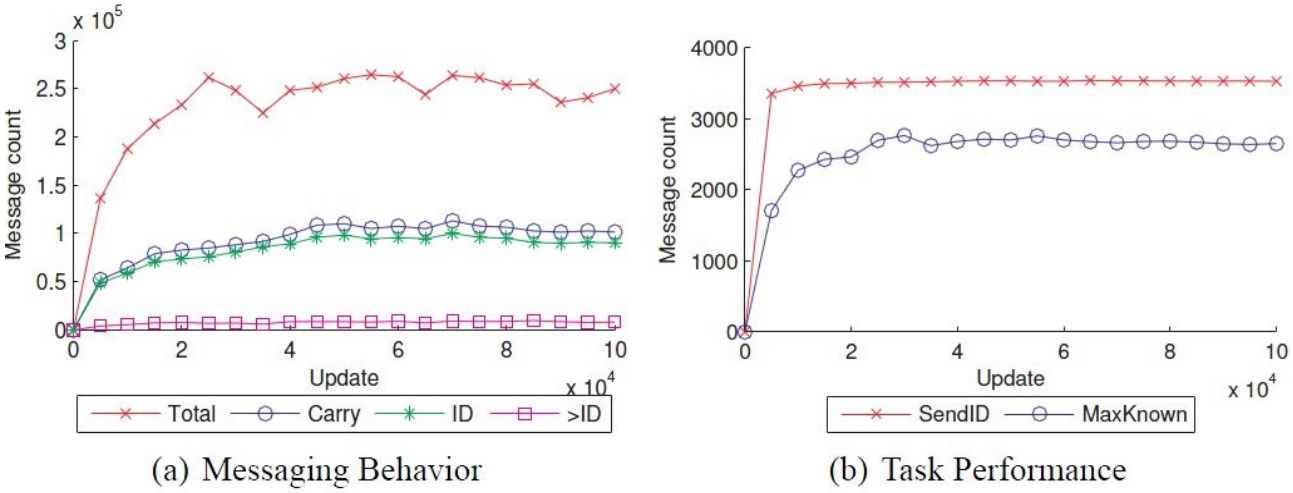

Abbildung 4: Versuchsergebnisse mit "Send-ID" und "Max-Known" *Quelle: [1], S. 388*

Wie Abbildung 4 (a) zeigt wird die Aufgabenstellung bei weitem nicht gelöst. Man erkennt, dass mehr als die Hälfte aller gesendeten Nachrichten keine Zell-ID enthalten und über 75% derjenigen Nachrichten die eine Zell-ID enthalten, die des Versenders entspricht. Gerade 1% der gesendeten Nachrichten enthalten Zell-IDs die größer als die des sendenden Organismus sind.

Nun stellt sich die Frage, ob sich die Organismen nicht so entwickeln wie gewünscht und nicht in der Lage sind diese Tasks auszuführen. Doch durch Abbildung 4 (b) wird deutlich, dass alle Organismen sehr schnell die „SEND-ID“ Task ausführen und nach kurzer Zeit auch, zu mindestens ein Großteil (72%), die „MAX-KNOWN“ Task. Der Fehler liegt also in der Kombination der Tasks, da Organismen, nachdem sie einen zufälligen kleineren Wert erhalten habe, durch das Versenden ihrer eigenen Zell-ID beide Tasks erfüllen. Wie die Abbildung noch zeigt kommt es häufig zu „Junk“-Nachrichten, die keine Zell-ID enthalten. Diese lassen sich zum Teil dadurch erklären, dass die Task „MAX-KNOWN“ Organismen auch dafür belohnt, zufällige große Zahlen zu versenden, die sie zuvor von einem anderen Organismus erhalten haben.

3.3 Versuch 2: Förderung von Zell-ID tragenden Nachrichten

Die Tasks werden größtenteils erfüllt, doch versenden die Organismen noch durch sehr viele zufällig mutierte „SEND-MSG“ Instruktionen willkürliche Werte, die die Kommunikation in der Bevölkerung behindert. Um der Erzeugung des erwünschten Verhaltens näher zu kommen wird nun versucht die Erzeugung von „Junk“-Nachrichten zu verringern oder am besten ganz zu unterdrücken.

Hierfür wurden zwei verschiedene Vorgehensweisen ausprobiert. Die eine benutzt eine weitere Task „SEND-NON-ID“, die nicht belohnt, sondern Organismen dafür bestraft Nachrichten, die keine Zell-ID enthalten, zu verschicken indem sie ihre Leistung um 75% senkt. Die andere Möglichkeit besteht darin die Kosten der „SEND-MSG“ Instruktion zu erhöhen. Dadurch verlieren Organismen die nebenbei „Junk“-Nachrichten produzieren einen größeren Teil ihrer wertvollen Rechenzeit an Instruktionen, die ihnen keine Leistung einbringt.

Beide bringen das gewünschte Ergebnis, wobei sich aber die Hinzunahme der „SEND-NON-ID“ Task als schneller herausstellt und deshalb weiterverwendet wird. Allerdings reicht diese Änderung noch nicht für eine deutliche Verbesserung. Erst bei der zusätzlichen Änderung der Ersetzungsstrategie von „MASS-ACTION“ zu „NEIGHBOORHOOD“ wird das Verhalten der Organismen deutlich besser. Dies ist auf Verwandtschaftsselektion zurückzuführen, da mit dieser Änderung verwandte Organismen nun benachbart sind und zusammenarbeiten können, um der Bestrafung durch „SEND-NON-ID“ zu entgehen. Die Änderung der Ersetzungsstrategie ohne Einführung der Bestrafung führt selbstverständlich zu dem gleichen Ergebnis wie in Versuch 1, da die Organismen trotz höherer Verwandtschaftsgrade in ihrer näheren Umgebung keinen Anreiz haben das Senden von „Junk“-Nachrichten zu unterlassen. (vgl. [1], S. 388-389)

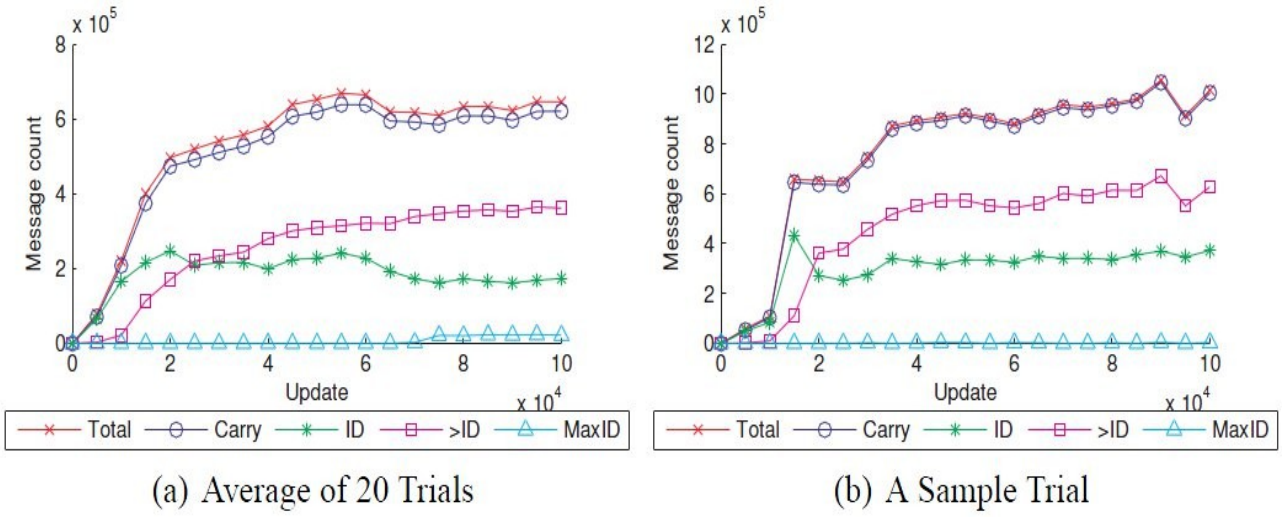

Abbildung 5: Versuchsergebnisse mit "Send-ID", "Max-Known", "Send-Non-ID" Quelle: [1], S.389

Abbildung 5 (a) zeigt die durchschnittlichen Ergebnisse der Simulationsabläufe wie oben erläutert, während 5 (b) eine besonders erfolgreiche Simulation zeigt. Hinzugenommen wurde die Markierung „MaxID“, die die Anzahl an Nachrichten mit der größten Zell-ID anzeigt. Man kann erkennen, dass die Menge an „Junk“-Nachrichten deutlich zurückgegangen ist und wie erwartet steigt damit auch langsam die Anzahl an Nachrichten die größere Zell-IDs enthalten als die ihrer Sender. Allerdings ist leicht zu sehen, dass die größte Zell-ID immer noch nicht verbreitet wird und das noch viele Organismen häufig ihre eigene Zell-ID versenden.

Folglich wird im nächsten Schritt versucht die Versendung von Nachrichten, die größere IDs enthalten als die des Senders, zu fördern. Da die Organismen nicht zwischen Zell-IDs und normalen Zahlen unterscheiden können, kann die Kombination aus „SEND-ID“ und „MAX-KNOWN“ nur sicher ausgeführt werden, wenn sie ihre eigene Zell-ID versenden. Die Gefahr durch „SEND-NON-ID“ bestraft zu werden ist bisher noch sehr groß.

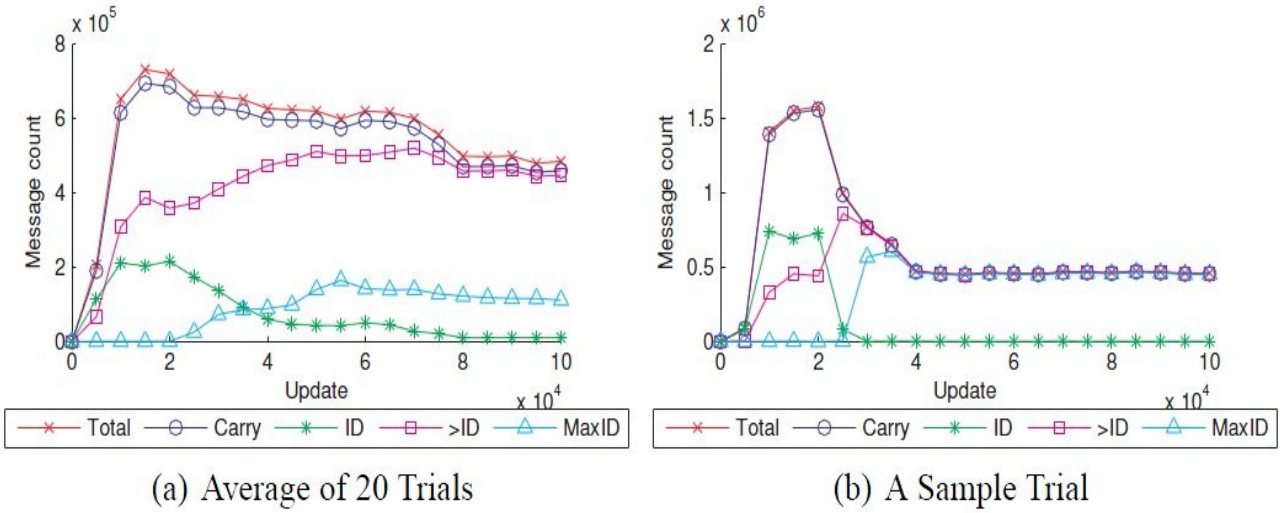

Abbildung 6: Versuchsergebnisse mit "Send-Self", "Max-Known", "Send-Non-ID"Quelle: [1], S. 390

Um dies zu ändern wird die Task „SEND-ID“ mit „SEND-SELF“ getauscht. Diese Änderung hat zur Folge, dass jeder Organismus am Anfang vermehrt seine eigene Zell-ID versendet, da er durch diese nicht bestraft wird. Folglich gibt es eine drastische Zunahme von Nachrichten zu Beginn der Simulation wie in Abbildung 6 deutlich zu erkennen ist. Doch wie speziell in Abbildung 6 (b) zu sehen ist nehmen die Nachrichten nach ungefähr 2000 Updates wieder ab. Daraus lässt sich folgern, dass die Organismen eine Filterungsmethode entwickeln, um nur noch Nachrichten zu versenden, wenn diese sicher eine größere Zell-IDs enthalten als ihre eigene. Folglich nähert sich die Anzahl der Nachrichten mit der größten Zell-ID immer mehr der insgesamt gesendet Nachrichten an. Abbildung 6 (b) zeigt einen Versuch, wo dieses Verhalten besonders schnell entwickelt wurde. Wie zu sehen ist enthalten nahezu alle gesendeten Nachrichten die größte Zell-ID somit wurde das gewünschte Verhalten erzeugt.

3.4 Versuch 3: Anpassungsfähigkeit an Umweltveränderungen

Da nun die Einstellungen gefunden wurden mit denen das gewünschte Verhalten erzeugt werden kann, wird als nächstes getestet, ob die Bevölkerung in der Lage ist sich an eine Änderung an der größten Zell-ID anzupassen. Hierfür wurde das bereits oben beschriebene Event „RESET-ID“ benutzt, das nach der Hälfte der Updates der größten Zell-ID einen zufälligen kleineren Wert zuordnet.

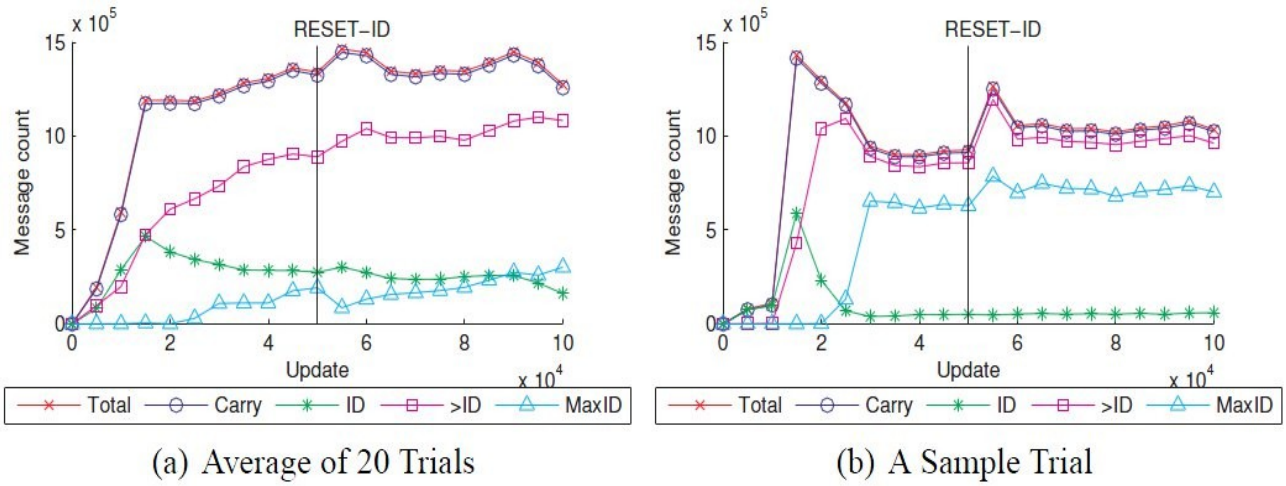

Abbildung 7: Versuchsergebnisse mit Reset *Quelle: [1], S. 390*

Wie in der Abbildung 7 (a) gut zu sehen ist gibt es einen kleinen Einbruch bei der Verbreitung der größten Zell-ID, aber innerhalb von 10.000 Updates erholt sich die Bevölkerung nicht nur sondern überholt die Rate vor dem Reset. Im Fall des Resets werden alle Organismen, die die vorherige größte Zell-ID versendet haben, durch die „SEND-NON-ID“ bestraft, wodurch werden sie schnell ersetzt werden.

Der evolutionäre Druck der durch die Bestrafung von „SEND-NON-ID“ ausgeht zwingt also die Bevölkerung anpassungsfähig zu bleiben. Man könnte ihn sich als Raubtier in biologischen System deuten oder als unbekannter Angreifer in virtuellen System. (vgl. [1], S. 390)

4. Zusammenfassung und Fazit

Die Ergebnisse der Testläufe haben deutlich gezeigt, dass man mit Hilfe von AVIDA und der Wahl der richtigen Tasks eine Bevölkerung von digitalen Organismen erschaffen kann, die in der Lage sind eine komplexe Aufgabe durch Zusammenarbeit zu lösen. Obwohl die Organismen keine zuvor eingebaute Funktion zur Kommunikation besitzen, wird dieses Verhalten allein durch zielgerichtete Evolution erschaffen. Durch die natürliche Selektion der „SEND-NON-ID“ Task wird nicht nur die wichtige Filterung bestimmter Nachrichten möglich, sondern sie verleiht der Bevölkerung eine gewisse Resistenz gegenüber Umweltveränderungen. Bei der Analyse einer der dominierenden Genotypen am Ende der Simulationen stellt man fest, dass die Organismen sich nur noch fortpflanzen wenn sie eine Nachricht mit größerer Zell-ID als ihre eigene erhalten. Sie haben sich also so weit entwickelt, dass die einzelnen, vorher unabhängigen, Organismen eine Abhängigkeit zueinander aufbauen, mit dem Ziel gemeinsam zu überleben. (vgl. [1], S. 392)
Damit erkennt man auch schon den großen Nutzen, den diese Entwicklungstechnik bietet. Man erhält Programme oder System, die zwar eine Aufgabe nicht am effektivsten lösen, aber dafür eine gewisse Variabilität mit sich bringen. Ebenso wie biologische Organismen haben sich die digitalen Organismen soweit entwickelt, dass sie mögliche Bedrohungen ausweichen oder sich von Angriffen erholen können.
Dabei wird versucht die Simulationen so realistisch wie möglich zu gestalten. AVIDA wurde seitdem kontinuierlich weiterentwickelt und es wurde eine Vielzahl an Details und Einstellungsmöglichkeiten hinzugefügt. Gerade der hohe Grad an präzisen und vielfältigen Beobachtungs- und Messwerkzeugen, die AVIDA mit sich bringt, stellt einen klaren Vorteil gegenüber den langwierigen und sehr schwierigen traditionellen Methoden dar. Für die Evolution der Organismen ist allerdings sehr wichtig, welche verschiedenen Instruktionen ihnen zur Verfügung stehen, da sie die Grundlage für ihre möglichen Entwicklungen darstellen. Daher muss bei den Forschungen besonders auf die Entwicklung des Instruktion-Sets geachtet werden.
Dafür ist es möglich durch neue Instruktionen immer mehr Forschungsgebiete mit AVIDA zu erschließen, so soll es möglich sein Organismen zu entwickeln, die Hinderniskurse durchlaufen oder bewegliche Objekte fangen. (vgl. [1], S. 392-393)

Literatur

[1] Knoester, D., McKinley, P.K., Beckmann, B., Ofria, C.: Directed Evolution of Communication and Cooperation in digital Organisms. Lecture Notes in Computer Science. Bd. 4648, Seite 384-394,2007.
[2] Ofria, C., Wilke, C.O.: Avida: A software platform for research in computational evolutionary biology. Journal of Artificial Life. Bd. 10, Seite 191-229,2004.