Ausarbeitung Seminar Künstliche Immunsysteme

Using Genetic Algorithms to Explore Pattern Recognition in the Immune System

Von Burkhard Menges

Inhalt:

Abbildungsverzeichnis

1.Einführung in Genetische Algorithmen

Genetische Algorithmen gehören zu den evolutionären Algorithmen. Ein evolutionärer Algorithmus arbeitet auf einer Population von Individuen, von denen jedes einen Punkt im Suchraum der möglichen Lösungen für ein gegebenes Problem darstellt. Der Algorithmus entwickelt diese Population mittels der Operationen Reproduktion, genetische Variation und Selektion.

1.1 Ein Standard-evolutionärer Algorithmus

Ein typischer evolutionärer Algorithmus verläuft in folgenden Schritten:

1. *Initialisierung*: Erzeuge eine initiale Population von Individuen (Eltern). Dies wird oft dadurch gemacht, dass man aus dem Raum der möglichen Lösungen zufällig eine Menge auswählt;
2. *Evaluation*: Evaluiere die Qualität des Verhaltens des Systems für alle Individuen der Population. Jedem Individuum wird ein Performanzindex zugeordnet, der seine Fitness in Bezug auf die Umgebung angibt;
3. *Selektion*: Bestimme mittels Auswahloperatoren, welche Individuen mit welcher Häufigkeit für die Reproduktion ausgewählt werden;
4. *Reproduktion und genetische Variation*: Erzeuge eine neue Population von Individuen (Nachkommen) durch Reproduzieren der Ausgewählten und zufälliges Variieren ihrer Nachkommen. Diese zufällige Variation wird mittels genetischer Operatoren wie Crossover und Mutation durchgeführt.
5. *Schleife*: Die in Schritt 3 ausgewählten Individuen bilden eine neue Population und durchlaufen Schritt 4. Der Algorithmus bricht ab, wenn eine vordefinierte Lösung erreicht wird oder eine bestimmte Anzahl von Generationen durchlaufen ist.

Jedes Mal wenn die Schritte 2 bis 4 durchlaufen sind, ist eine neue Generation entstanden. Dieser Prozess der simulierten Evolution entsteht also durch sukzessive Anwendung des natürlichen Auswahlmechanismus, der die Wahrscheinlichkeit für die besonders fitten Individuen zum Überleben über Generationen hinweg erhöht, gefolgt von der Anwendung der „künstlichen“ Reproduktion und genetischen Variation der ausgewählten Individuen. Ein evolutionärer Algorithmus versucht eine Lösung für ein Problem zu bestimmen, indem er eine Menge möglicher Lösungen zu Grunde legt, den so genannten Suchraum, und auf diesem operiert. Eine Fitness-Landschaft ist die Repräsentation einer Abbildung vom Raum aller möglichen Genotypen auf ihre jeweilige Fitness.

1.2 Genetische Algorithmen

Ein genetischer Algorithmus kann als stochastischer Algorithmus definiert werden, dessen Suchmethode die biologischen Phänomene der genetischen Vererbung und natürlichen Auslese zu modellieren versucht. Einfache genetische Algorithmen bilden abstrakte Modelle der natürlichen Evolution und arbeiten mit einer Population fester Größe und Individuen, die durch „genetische Strings“ fester Länge repräsentiert werden. Neue Populationen entstehen mittels einer probabilistischen Fitness-proportionalen Auswahl von Individuen, Crossover und Mutation, wodurch zu den Eltern ähnliche Nachkommen entstehen. Die Hauptmerkmale eines genetischen Algorithmus sind:

- Populations-basierte Suche;
- Benutzung einer Kostenfunktion (Fitness, Ziel oder Anpassungsfähigkeit) anstelle von Ableitungen oder anderen Arten von Wissen;
- Benutzung probabilistischer Übergangsregeln (Selektion und Reproduktionsmechanismen) anstelle deterministischer.

Um zu entscheiden, ob sich ein genetischer Algorithmus zur Lösung eines gegebenen Problems eignet, kann man einige Kriterien angeben:

- Wenn der Suchraum groß ist, keinen glatten Verlauf hat, nicht gleichförmig ist, unbekannt ist oder die Fitness-Funktion verrauscht ist, dann ist ein genetischer Algorithmus wahrscheinlich gut geeignet;
- Wenn der Suchraum glatt und einförmig ist, dann sind Gradientenverfolgung oder Bergsteigen besser als genetische Algorithmen;
- Wenn der Suchraum gut bekannt ist, dann kann man mit heuristischen Methoden arbeiten, die man in irgendwelche Suchalgorithmen einbaut, auch in genetische Algorithmen.

Soll ein Problem mit einem genetischen Algorithmus gelöst werden, dann muss zunächst eine geeignete genetische Repräsentation festgelegt werden. Dann kann ein Algorithmus nach der Struktur des in folgender Abbildung dargestellten Diagramms verwendet werden.

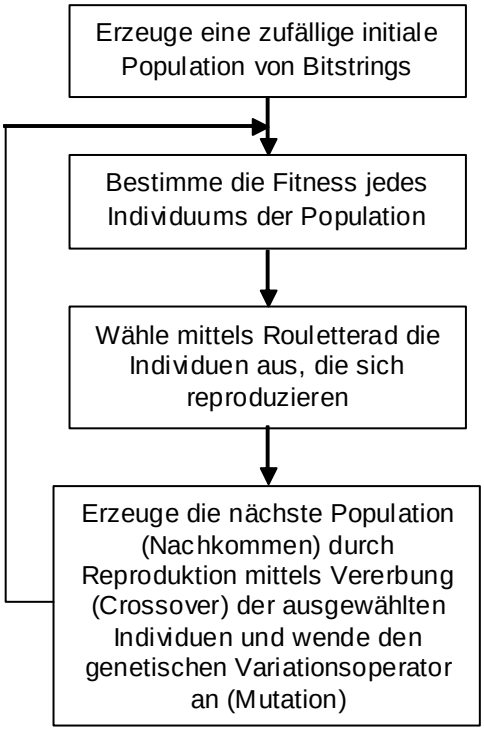

Abbildung 1: Struktur eines genetischen Algorithmus

Bei der Rouletterad-Methode ist die Wahrscheinlichkeit für die Auswahl eines Chromosoms (d.h. eines Individuums einer Population) proportional zu seinem Fitnesswert. Die nächste Abbildung veranschaulicht die Funktionsweise des Rouletterads.

Individuum	Chromosom	Fitness	Grade
1	0001100101010	16	240
2	0101100101010	4	60
3	1011110100101	2	30
4	1010010101001	2	30

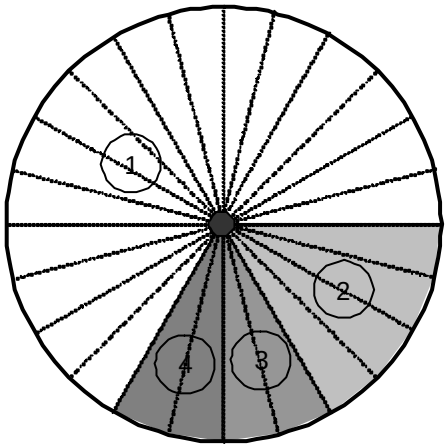

Abbildung 2: Rouletterad-Methode

Die Auswahl des Individuums erfolgt quasi durch „drehen“ des Rades. Es ist einzusehen, dass Individuum 1 somit die größte Wahrscheinlichkeit hat, ausgewählt zu werden.
Neben der Rouletterad-Methode gibt es beispielsweise auch noch die Verfahren Turnierselektion und Truncation Selection. Die Turnierselektion ist ähnlich der Rouletterad-Methode, nur dass dort die Individuen in einem "Turnier" gegeneinander antreten. Ihre Stärke entspricht dabei ihrer Fitness. Das Individuum, welches am Ende noch übrig ist, wird ausgewählt. Die Truncation Selection ist wesentlich simpler. Hier werden die Individuen nach ihrer Fitness sortiert und das an erster Stelle stehende wird ausgewählt.

Beim *Ein-Punkt-Crossover* wird eine Position in den beiden Elternchromosomen zufällig bestimmt, an dem die beiden Strings aufgetrennt werden. Die vier Teilstücke werden dann überkreuzt wieder zusammengesetzt. Jedes Chromosom wird mit der Wahrscheinlichkeit p_c für die Crossover-Operation ausgewählt. Bei der *Mutation* wird eine Position in einem Chromosom zufällig bestimmt und das Bit an dieser Position wird gekippt. Die Mutationswahrscheinlichkeit p_m legt die Rate fest, mit der jede Position des Chromosoms verändert werden kann. Die beiden Operationen sind in folgender Abbildung illustriert.

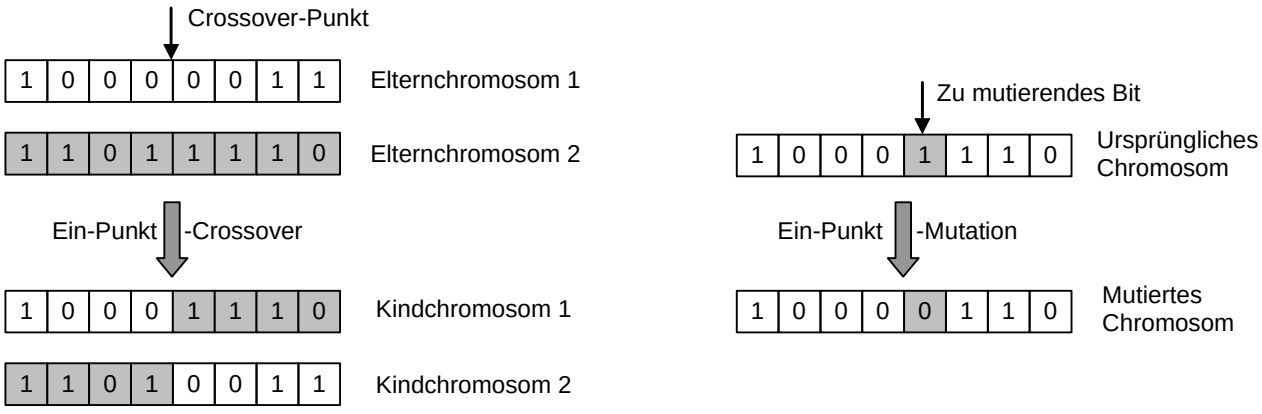

Abbildung 3: Crossover und Mutation

2. Das Immunsystemmodell

2.1 Natürliches Immunsystem

Das menschliche Immunsystem ist in der Lage etwa 10^{16} Fremdkörper zu erkennen, was praktisch heißt, dass das Immunsystem jedes körperfremde Molekül, das ihm präsentiert wird, als ein solches kategorisiert. Selbst im Labor gezüchtete körperfremde Moleküle, mit denen das Immunsystem noch nie vorher konfrontiert wurde, werden korrekt erkannt.
Die Erkennung der körperfremden Moleküle erfolgt im menschlichen Immunsystem durch B - und T – Lymphozyten deren Rezeptoren eine einzigartige Oberflächenbeschaffenheit besitzen. Wird ein Antigen von Beispielsweise einer B – Zelle erkannt, sondert diese Antikörper ab, welche die gleiche Form des B – Zellen Rezeptors haben um beispielsweise ein Antigen binden zu können.

2.2 Künstliches Immunsystem

Eine erste Abstraktion von natürlichen Immunsystemen wird dadurch gemacht, dass zwischen den verschiedenen Arten der erkennenden Leukozyten nicht unterschieden wird, ebenso nicht zwischen verschiedenen Arten von Elementen, die erkannt werden können. Im Folgenden wird nur von einer Art von Rezeptoren geredet, diese werden Antikörper genannt. Alles, was von ihnen erkannt werden kann, wird Antigen genannt. Antigene können fremd oder körpereigen sein. Die Stärke der Bindung zwischen Antikörper und Antigen wird Affinität oder Match-Grad genannt.
Sowohl Antigene als auch Antikörper werden hier durch binäre Strings mit einer Länge von 64 Bit repräsentiert. Antikörper werden direkt generiert und Selektion und Mutation werden durch Zufallsgeneratoren bestimmt. Dies ist eine extreme Vereinfachung zum realen Fall aber dennoch angemessen. Zum einen ist die Vielfalt der Antikörper und Antigene mindestens genauso groß wie im menschlichen Körper, da sich durch Strings der Länge 64 $2^{64} = 10^{19}$ verschiedene Möglichkeiten ergeben. Des Weiteren verfügt auch das Immunsystem über diverse komplexe Mechanismen um Antikörper bei der Mutation zu randomisieren.

Da es hier um die Möglichkeiten der Mustererkennung mithilfe des Immunsystems geht, behandelt das Modell nur die Erkennung der Antigene durch die Antikörper und nicht deren darauf folgende Vernichtung.

2.3 Matching

Ein Matching zwischen einem Antikörper und einem Antigen liegt genau dann vor, wenn ihre beiden Bitrepräsentationen komplementär sind. Da aber jeder Antikörper mehrere verschiedene Antigene matchen muss, wäre ein perfektes bitweises Matching nicht sonderlich geeignet. Vielmehr genügt es einen gewissen Match-Grad zu erreichen damit ein Antikörper ein Antigen erkennt.
Der Match-Grad wird mithilfe einer Matching-Funktion *M* bestimmt welche man auf verschieden Arten definieren kann. *M* kann beispielsweise einfach die Anzahl der komplementären Bits ausgeben oder aber auch die komplementären Regionen innerhalb eines

Bitstrings aufgrund ihrer Länge unterschiedlich gewichten und dementsprechend ein Resultat ausgeben.
In diesem Modell wird die einfache Variante benutzt:

$$M = \sum_i l_i$$

l_i ist hierbei die Länge einer komplementären Region.

Antibody: 11001001000100100001001010101010
Antigen: 01111100111001011110110101110100
Komplement: 10110101111101111111111111011110
Länge l: 1 2 1 5 13 4

Ziel bei der Generierung der Antikörper ist eine möglichst hohe Überdeckung der Antigenpopulation. Jedes Antigen soll von mindestens einem Antikörper erkannt werden. Je nach Match Bedingung kann dies ein triviales Problem sein. Fordert man beispielsweise ein Matching von nur einem Bit für die Erkennung eines Antigens, so würden zwei verschiedene Bitstrings (nämlich 1… und 0…) ausreichen um alle Antigene zu erkennen. Unter diesen Umständen würden die Antikörper jedoch auch alle körpereigenen Moleküle matchen.
Also muss zusätzlich gefordert werden, dass kein körpereigenes Molekül fälschlicherweise als körperfremd erkannt werden darf.
Um dies zu erreichen fordert man einfach eine hohe Fitness damit die Wahrscheinlichkeit für ein Self-Matching möglichst gering wird.

2.4 Allgemeine Problemstellung

Das Immunsystem ist in der Lage verschiedene Arten von Mustererkennungsproblemen zu lösen. Es stellt sich die Frage ob genetische Algorithmen ebenfalls in der Lage sind solche Problemstellungen zu bewältigen. In den folgenden Versuchen soll dies evaluiert werden.
Ziel der Recherche ist es das Immunsystem von einem Informationsverarbeitendem Standpunkt aus zu verstehen und die dadurch gewonnen Erkenntnisse und Ideen zur Entwicklung neuer effizienter Algorithmen zu verwenden.
Hierzu werden im Besonderen zwei verschiedene Arten der Mustererkennung untersucht. Zum einen das Erkennen gemeinsamer Schemata innerhalb der Antigen Bitstrings und zum anderen die Aufrechterhaltung der Unterschiedlichkeit (Diversität) unter den Antikörpern bei einer "gemischten" Antigenpopulation.

3. Erkennung gemeinsamer Schemata

Bei diesem ersten Mustererkennungsproblem soll getestet werden ob ein genetischer Algorithmus in der Lage ist bei gegebener Antigenpopulation Gemeinsamkeiten im Bitmuster (sprich Schemata) der Antigene zu erkennen und die Antikörper dementsprechend zu generieren.
In der Natur taucht diese Art der Mustererkennung zum Beispiel beim Erkennen der meisten Bakterien auf. So sind die Zellwände vieler Bakterienarten aus Polymeren, die bei menschlichen Zellen nicht vorkommen. Das Auffinden dieses Polymermusters bei den

Antigenen entspricht in unserem Fall also dem Auffinden eines bestimmten Bitmusters innerhalb des Strings.

3.1 Problemstellung

Die Bitmuster die hier verwendet werden sollen sind so genannte "length-schemas" also "Längen-Schemata" bei denen auf einer bestimmten Länge der Bitstrings ein einfaches Schema zu erkennen ist.

Half-length schema:

50 %	11 … 11** … **
50%	** … **11 … 11

Quarter-length schema:

25%	11******	25%	**11****
25%	****11**	25%	******11

Beim half-length schema besteht der String in der ersten bzw. in der zweiten Hälfte nur aus Einsen. Der Rest des Strings spielt keine Rolle und wird zufällig besetzt (Don't Cares). Beim quarter-length schema wird entsprechend eines der vier Viertel des Strings mit Einsen besetzt.

Für diese Art der Problemstellung wird die Fitness eines Antikörpers folgendermaßen bestimmt:

1. *Wähle aus einer festen Menge von Antigenen zufällig eine Teilmenge der Größe p aus*
2. *Berechne für jedes Antigen der Teilmenge den Match-Grad mithilfe der Matching-Funktion*
3. *Die Fitness eines Antikörpers ist der durchschnittliche Match-Grad über allen p Antigenen*

Der genetische Algorithmus sollte in der Lage sein einen Generalisten zu finden der alle vorkommenden Antigene zu einem gewissen Grad matcht. Der am besten passende Antikörper für die vorgestellten Schemata wäre: 000 … 000
Durch die große Anzahl an Don't Cares, die mit Abnahme der Schemalänge noch größer wird, ist das Antigenfeld stark verrauscht, was es für den genetischen Algorithmus schwerer macht einen passenden Antikörper zu finden.
Die durchschnittliche Fitness des optimalen Antikörpers sinkt bei Abnahme der Schemalänge. Sie errechnet sich zu:

$$[(s/l)(1) + (1 - s/l)(1/2)]l = \frac{l+s}{2}$$

Wobei *s* die Länge des Schemas darstellt und *l* die Länge der Bitstrings.
Im Falle des half-length schemas matcht der Antikörper die Hälfte der Bits eines Antigens mit einer Wahrscheinlichkeit von 1 und die restlichen Bits mit einer Wahrscheinlichkeit von ½. Also ist hier die durchschnittliche Fitness eines Antikörpers:

[(1/2)(1) + (1-1/2)(1/2)]64 = 48

Beim quarter-length schema stimmen ein Viertel der Bits mit Wahrscheinlichkeit 1 und entsprechend drei Viertel mit der Wahrscheinlichkeit ½. Die Fitness errechnet sich zu:

[(1/4)(1) + (1-1/4)(1/2)]64 = 40

Wie in der Natur könnten die Lösungen sowohl Generalisten als auch Spezialisten sein wobei ein Spezialist für die Lösung des gegebenen Problems eher ungeeignet wäre, da es eher unwahrscheinlich ist, dass ein Antigen auf das der Spezialist ausgelegt ist, dem gleichen Antikörper in einer der folgenden Generationen nochmals präsentiert wird.

3.2 Wie gut erkennt ein genetischer Algorithmus gemeinsame Schemata?

Für die nachfolgend beschriebenen Experimente wurde das Genesis Public Domain Software Package mit Veränderungen in der Fitnessfunktion benutzt. Die in den Diagrammen geplotteten Punkte sind Durchschnittswerte aus 10 Durchläufen in Abschnitt 3 und aus 30 Durchläufen in Abschnitt 4. Damit ein Antigen und ein Antikörper matchen müssen zwischen 95% (Abschnitt 4) und 100% (Abschnitt 3) der Bits komplementär sein.
Es soll nun also getestet werden ob sich ein genetischer Algorithmus bei der gegebenen Problemstellung bewähren kann. Dazu werden 200 Antikörper zufällig ausgewählt, welchen eine Antigenmenge der Größe p=5 präsentiert wird.
In Abbildung 4 kann man sehen, dass die durchschnittliche Fitness der Antikörper monoton bis zu einer Asymptote ansteigt. Man erkennt, dass die Asymptoten für half-, quarter- und eight-length schema genau bei dem jeweiligen errechneten Durchschnittswert liegen.
Abbildung 5 zeigt, dass der genetische Algorithmus das jeweilige Maximum schneller erreicht je länger das Schema ist. Das ist nicht allzu verwunderlich, da bei längerem Schema weniger zufällig verteilte Bits vorliegen und somit das Rauschen deutlich verringert wird.

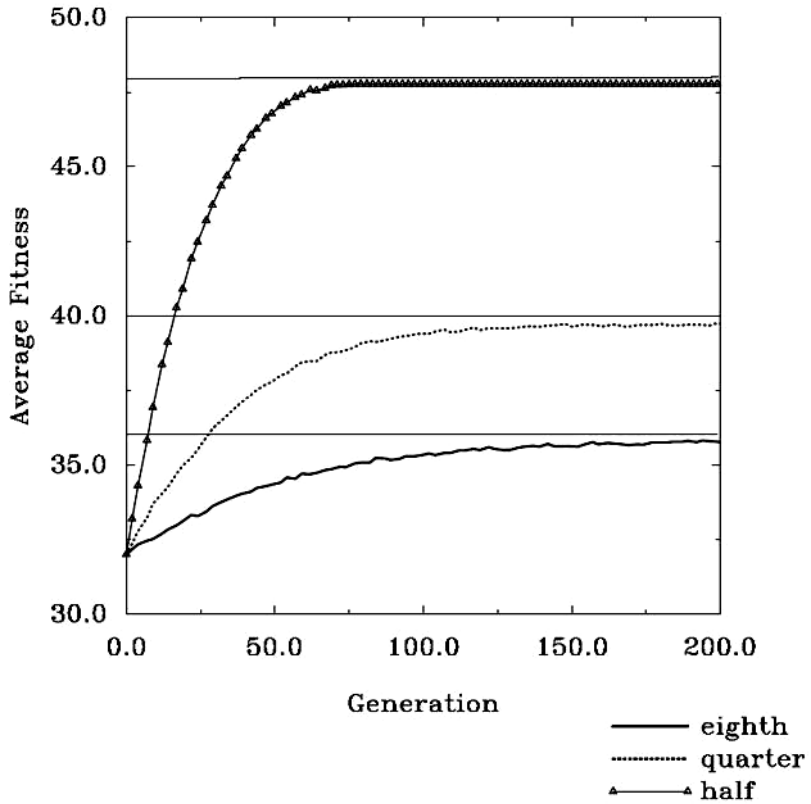

Abbildung 4: Durchschnittliche Fitness der Antikörper bei half-, quarter- und eight-length schema

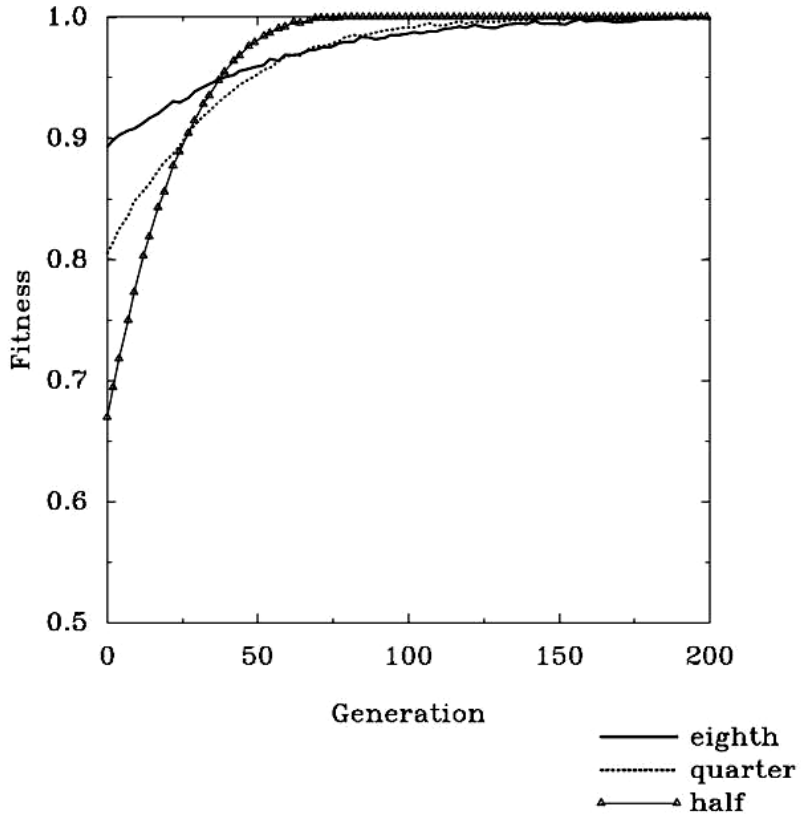

Abbildung 5: Konvergenzgeschwindigkeit der Schemata zu ihrer maximalen Fitness

Durch dieses Experiment wird deutlich, dass ein genetischer Algorithmus in der Lage ist gemeinsame Schemata auch zu finden, wobei bei kleineren Schemata das Rauschen und die Auffindungsdauer größer werden. Aufgrund dieser Tatsache liegt die Vermutung nahe, dass nur begrenzt kleine Schemata erkannt werden können.

4. Aufrechterhalten der Unterschiedlichkeit

Das zweite Problem beschäftigt sich mit der Frage, was passiert wenn die Antigenpopulation nicht von einem Generalisten erkannt werden kann. Dafür wird folgende Antigenpopulation betrachtet:

50% 000 … 000
50% 111 … 111

Um dieses Problem zu lösen, müsste ein genetischer Algorithmus zwei verschiedene Lösungen gleichzeitig aufrechterhalten. Diese Problemstellung wird Multiple Peaks Problem genannt und ist für nicht genetische Algorithmen schwer zu lösen, da diese ein starkes Konvergenzverhalten aufweisen. Typischerweise konvergieren Algorithmen zufällig gegen einen der Peaks, sprich eine mögliche Lösung.
Dennoch ist diese Problemstellung vom Immunologischen Aspekt gesehen vielleicht die Interessantere von beiden, da sie in der Biologie häufiger vorkommt.
Crossover ist hier als Reproduktionsmechanismus eher ungeeignet, da durch Vertauschung von Bitstringabschnitten guter Lösungen Hybride entstehen können, die als Lösung vollkommen ungeeignet sind. Daher wird die Crossover Rate in den folgenden Versuchen

möglichst klein gehalten. Das entspricht auch dem Vorgehen der Natur, in der somatische Mutation sogar überhaupt kein Crossover benutzt.
Die im vorherigen Abschnitt benutzte Vorgehensweise zur Berechnung der Fitness der einzelnen Antikörper ist für dieses Problem natürlich vollkommen ungeeignet, da ja keine Antikörperpopulation benötigt wird die alle vorkommenden Antigene gut matcht sondern eine, die verschiedene Formen (Lösungen) in sich aufrechterhalten kann. Folglich muss also das Fitnessschema angepasst werden. Zu diesem Zweck betrachtet man sich zunächst die Eigenschaften des Immunsystems bei der Fitnessberechnung:

- Antigene treten sequentiell auf
- Das Immunsystem reagiert nur mit einer Teilmenge seiner Lymphozyten
- Es gibt einen „Wettkampf“ um die Antigene. Antikörper mit größerer Affinität vermehren sich schneller
- Antikörper entwickeln sich durch Punkt Mutation (Somatische Mutation) => GA muss vermehrt mit Mutation anstelle von Crossover arbeiten

Daraus kann man folgendes Fitness Schema für die Antikörper entwickeln

1. Wähle ein Antigen zufällig aus
2. Eine Menge der Größe σ von Antikörpern wird zufällig gewählt
3. Jeder dieser Antikörper wird mit dem Antigen gematcht
4. Der Antikörper mit dem größtem Match Grad bekommt diesen auf seine aktuelle Fitness addiert
5. Wiederhole Vorgang für weitere Antigene (typischerweise 3*σ)

Dieses Schema entspricht der "best-match" Strategie, die von Klassifizierungssystemen benutzt wird.
Bezogen auf die gegebene Problemstellung ergeben sich einige Fragen an den genetischen Algorithmus, die mittels verschiedener Versuche evaluiert werden müssen.

- Kann ein konventioneller GA die verschiedenen Antikörper aufrechterhalten oder konvergiert er zu einer Lösung?
- Wie viele verschiedene Peaks kann der GA erhalten?
- Findet der GA die Peaks auch ohne vorgegebene Lösung?
- Spielt die Entfernung der Peaks voneinander eine Rolle (Hamming Distanz)
- Entwickeln sich Spezialisten oder Generalisten und unter welchen Umständen?

4.1 Kann Verschiedenheit erhalten werden?

Um diese Frage zu beantworten betrachtet man noch einmal die simple Antigenpopulation vom Anfang und eine Menge von Antikörpern der Größe σ = 30:

50% 000 … 000
50% 111 … 111

Bei Abbildung 6 wurde der genetische Algorithmus mit einer korrekten Lösung initialisiert um zu testen ob die Verschiedenheit der beiden Lösungen aufrechterhalten werden kann. Man erkennt, dass durch das Crossover zwar einige korrekte Antikörper verloren gehen aber die 50:50 Verteilung der beiden Lösungen erhalten bleibt.

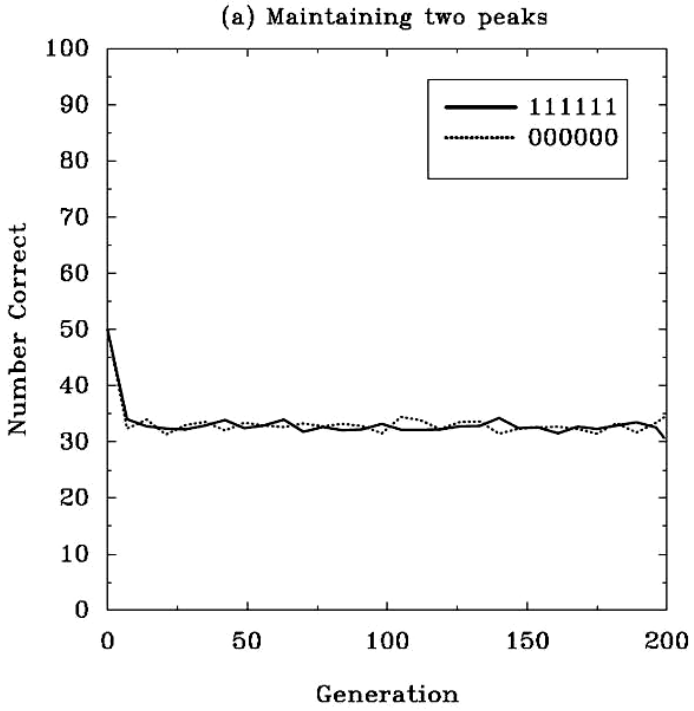

Abbildung 6: Aufrechterhaltung bei Vorgabe der korrekten Lösung

Es stellt sich die Frage ob der Algorithmus die 50:50 Verteilung der Lösungen auch selbstständig herstellen würde. Im zweiten Test wurde deshalb eine schlechte Anfangsverteilung der Lösungen vorgegeben. 99% der Antikörper bestanden aus 1er Strings und nur 1% aus 0er Strings. Abbildung 7 zeigt das Ergebnis dieses Experiments. Schon nach weniger als 10 Generationen ist das 50:50 Verhältnis der Lösungen hergestellt. Diese starke Tendenz zur Selbstregulierung lässt darauf schließen, dass der Algorithmus stabil gegen Störungen wie z.B. Rauschen ist.

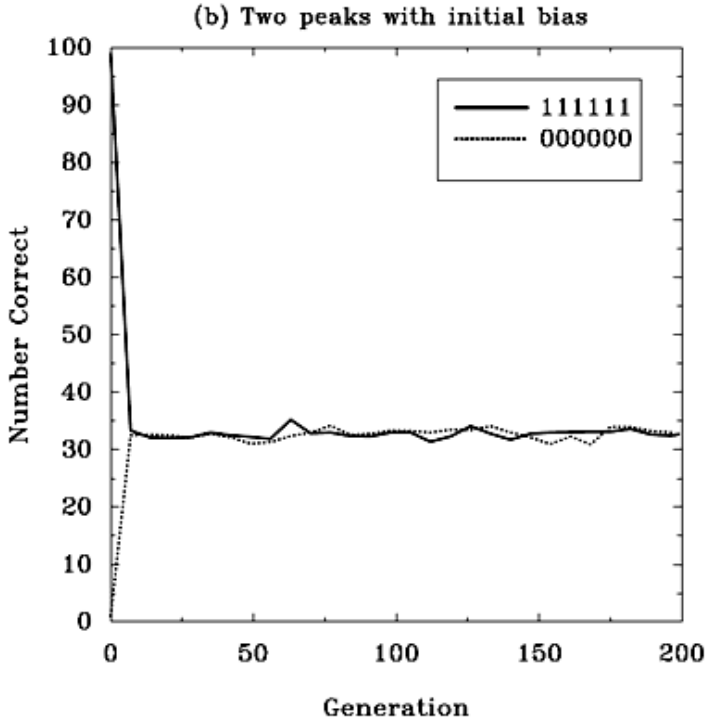

Abbildung 7: Initialisierung mit schlechter Verteilung der Antikörper

4.2 Werden Lösungen selbstständig gefunden?

Nachdem dem Algorithmus bisher die korrekten Lösungen vorgegeben wurden, soll jetzt getestet werden, ob er diese auch selbstständig finden kann. Für diesen Versuch gelten die gleichen Vorraussetzungen wie zuvor, nur dass jetzt die Antikörper zufällig der Gesamtpopulation entnommen werden.
Abbildung 8 zeigt, dass auch für diesen Fall Lösungen gefunden und aufrechterhalten werden.

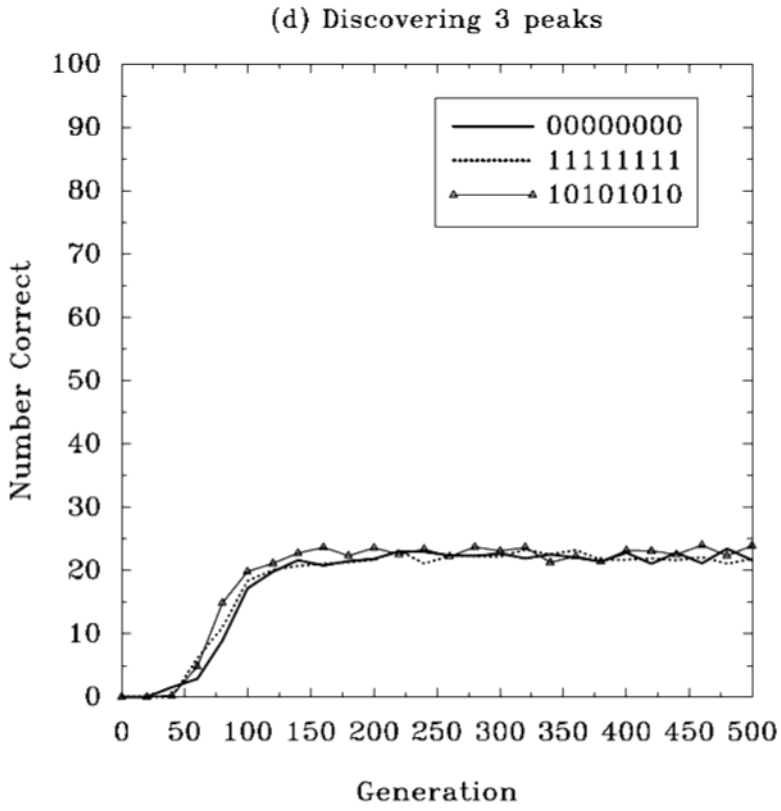

Abbildung 8: Selbstständiges Erkennen dreier unterschiedlicher Peaks

Als nächstes soll der Algorithmus mit 3 Antigenpopulationen konfrontiert werden, die analog zu den beiden vorherigen gleichverteilt sind und eine maximale Hamming Distanz untereinander aufweisen. Auch hier zeigt sich ein ähnliches Bild wie in Abbildung 8. Die drei Peaks werden selbstständig gefunden und aufrechterhalten.
Was passiert, wenn die drei Peaks nicht gleichverteilt sind, ist in Abbildung 9 zu sehen. Die Antigenpopulation, die dem Algorithmus präsentiert wird setzt sich wie folgt zusammen:

50% 000 … 000
30% 111 … 111
20% 101 … 010

Offensichtlich gelingt es dem genetischen Algorithmus auch hier die korrekten Lösungen zu finden. Darüber hinaus ist zu beobachten, dass die entstandene Antikörperpopulation die im gleichen Mengenverhältnis zu einander stehen wie die zugehörigen Antigene.

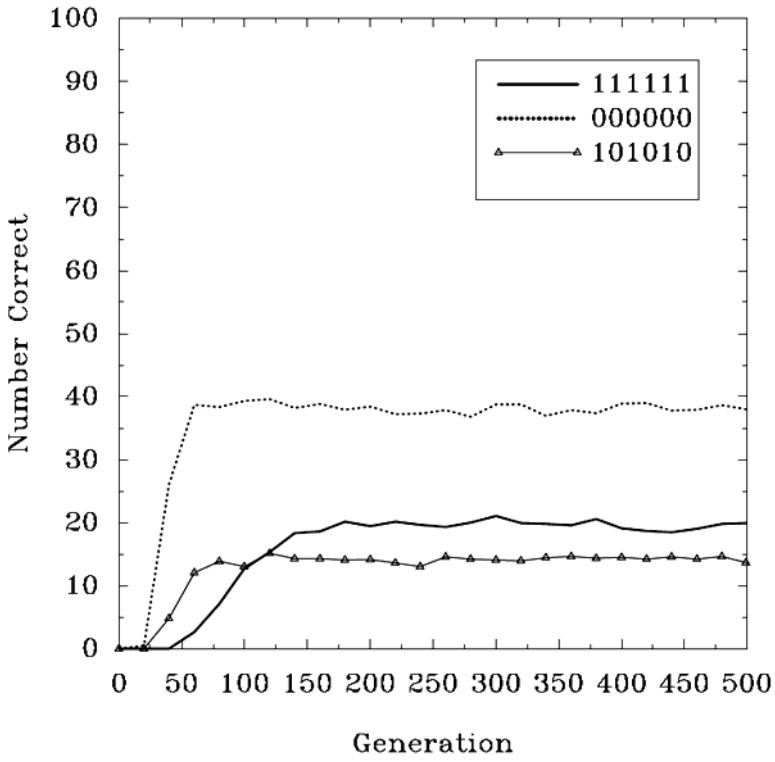

Abbildung 9: Erkennen von 3 Peaks mit ungleicher Mengenverteilung

4.3 Welche Rolle spielt σ?

Die vorherigen Versuche wurden alle mit einer Antikörpermenge der Größe σ=30 durchgeführt. Würde ein genetischer Algorithmus die verschiedenen Peaks auch mit einer geringeren Anzahl von Antikörpern finden oder gibt es so etwas wie eine maximale Erkennungskapazität?
Die Tabelle in Abbildung 10 beantwortet diese Frage. Etwa 15 Antikörper werden pro Peak benötigt, um dieses korrekt zu erkennen, wobei 10-15% der Antikörper keinem der Peaks zugeordnet werden können.

Antigen	Population Size		
Types	50	100	200
2	21.6 (3.2)	46.6 (2.6)	93.0 (4.6)
3	14.1 (7.2)	30.5 (2.3)	60.7 (5.3)
4	0.0	23.0 (0.9)	45.4 (4.6)
5		18.0 (0.6)	36.1 (3.3)
6		0.0	30.0 (3.0)
7		0.0	25.3 (3.3)
8		0.0	22.6 (0.4)
9		0.0	20.0 (0.4)
10		0.0	17.6 (0.7)
11		0.0	15.4 (0.7)
12		0.0	0.0

Abbildung 10: Benötigte Antikörper zur Erkennung eines Antigens

Diese Zahl deckt sich nicht ganz mit den Beobachtungen die man in der Natur gemacht hat. Bei einem Tier reicht bereits ein Antikörper (sprich eine B-Zelle) um ein Antigen zu erkennen. Wenn allerdings immer nur ein Antikörper benutzt würde um eine Antigen zu erkennen, würden kleine Veränderungen des Antigens, die z.B. durch Mutationen hervorgerufen werden könnten, ausreichen um eine Erkennung durch den Antikörper zu verhindern. In der Tat benutzt das Immunsystem verschiedenartige Antikörper um ein Antigen auf verschiedene Art zu erkennen. Die Anzahl dieser verschiedenartigen Antikörper variiert sehr stark und bewegt sich im 2-3 stelligen Bereich. Im Schnitt kann man aber sagen, dass das Immunsystem etwa 100 Antikörper pro Antigen benutzt.
Verglichen mit dieser Zahl erscheinen 15 Antikörper pro Antigen nicht mehr sehr hoch und ließe sogar beinahe die Behauptung zu, dass ein genetischer Algorithmus effizienter arbeitet als das reale Immunsystem, wenn das Immunsystem nicht auch mit weniger Antikörpern auskäme als es tatsächlich benutzt.

4.4 Ähnlichkeit von Peaks

Bisher wurden Antikörperpopulationen ausgewählt die eine möglichst große Hamming Distanz zueinander aufgewiesen haben. Interessant wäre es also zu wissen, ob die Hamming Distanz zwischen den Peaks eine Rolle bei der Erkennung spielt.
Um das zu testen werden Antigen-Peaks gleicher Größe mit einer variierender Hamming Distanz zwischen 64 und 1 untersucht. Die Größe der Antikörpermenge beträgt 15 und diese werden wieder zufällig der Population entnommen. Ein Antigen muss einen Antikörper zu 100% matchen und die gefundenen Peaks müssen über 500 Generationen aufrechterhalten werden um als erkannt zu gelten.
Überraschenderweise zeigt der Versuch, dass die Hamming Distanz keinen Einfluss auf das Auffinden der Peaks hat, wie man an Abbildung 11 sehen kann. Probleme treten nur auf, wenn σ zu klein gewählt wird, da dann nicht mehr alle Peaks gefunden werden. Weiterhin kann man sehen, dass Peaks die ähnlich zueinander sind nahezu gleichzeitig erkannt werden. Je ähnlicher sich Peaks sind, desto zeitnaher werden sie also erkannt.

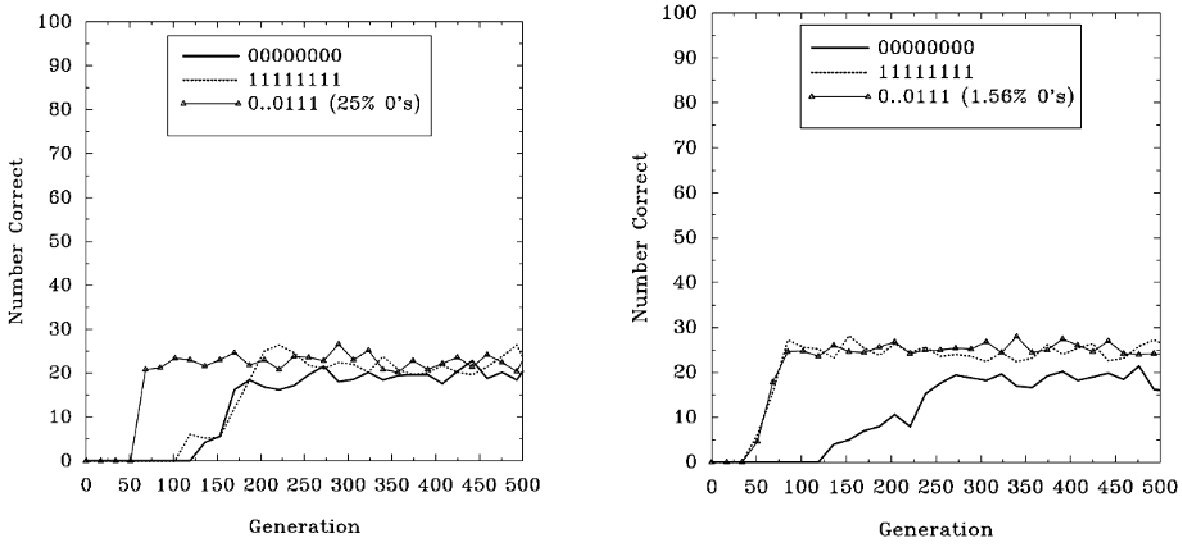

Abbildung 11: Erkennung von zueinander ähnlichen Peaks

4.5 Generalisierung

In Abschnitt 4.3 wurde bereits auf die Rolle des Parameters σ bei dem Erkennen der Antigene eingegangen. Nun soll in Abhängigkeit von σ die Generalisierung der Antikörper untersucht werden.
Die Idee ist, drei Antigen Peaks zu generieren, so dass ein Antikörper jedes Peak zu einem gewissen Grad matchen kann.
Betrachtet man z.B. die Antigene 000, 011 und 110, so wäre der Optimale Generalist für diesen Fall ein Antikörper der Form 101, da er jedes Antigen an zwei Stellen matchen würde.
Um herauszufinden inwiefern σ die Ausbildung von Generalisten und Spezialisten beeinflusst wurden drei Antigen-Klassen generiert die analog zum obigen Beispiel aufgebaut sind.

0…00…00…0
0…01…11…1
1…11…10…0

Damit der Bitstring gleichmäßig in drei Teile unterteilt werden kann wurde hier eine Stringlänge von 66 statt 64 Bits benutzt. Der passende Generalist hätte entsprechend die Form:

1…10…01…1

Abbildung 12 zeigt die Ergebnisse der Versuche mit σ = 2; 6; 7 und 10 und einer Antigenpopulation der Größe 100. Ein Antikörper musste zu 97% ein Antigen matchen um als korrekt klassifiziert zu werden und über 500 Generationen aufrechterhalten werden.
Man erkennt, dass bei σ = 2 nur der Generalist auftaucht, sich bei σ = 6 schon Spezialisten ausbilden und ab σ = 7 praktisch nur noch Spezialisten auftauchen. Wenn man die Antikörperpopulation von 100 auf 1000 erhöht treten wieder nur Generalisten auf (nicht abgebildet).

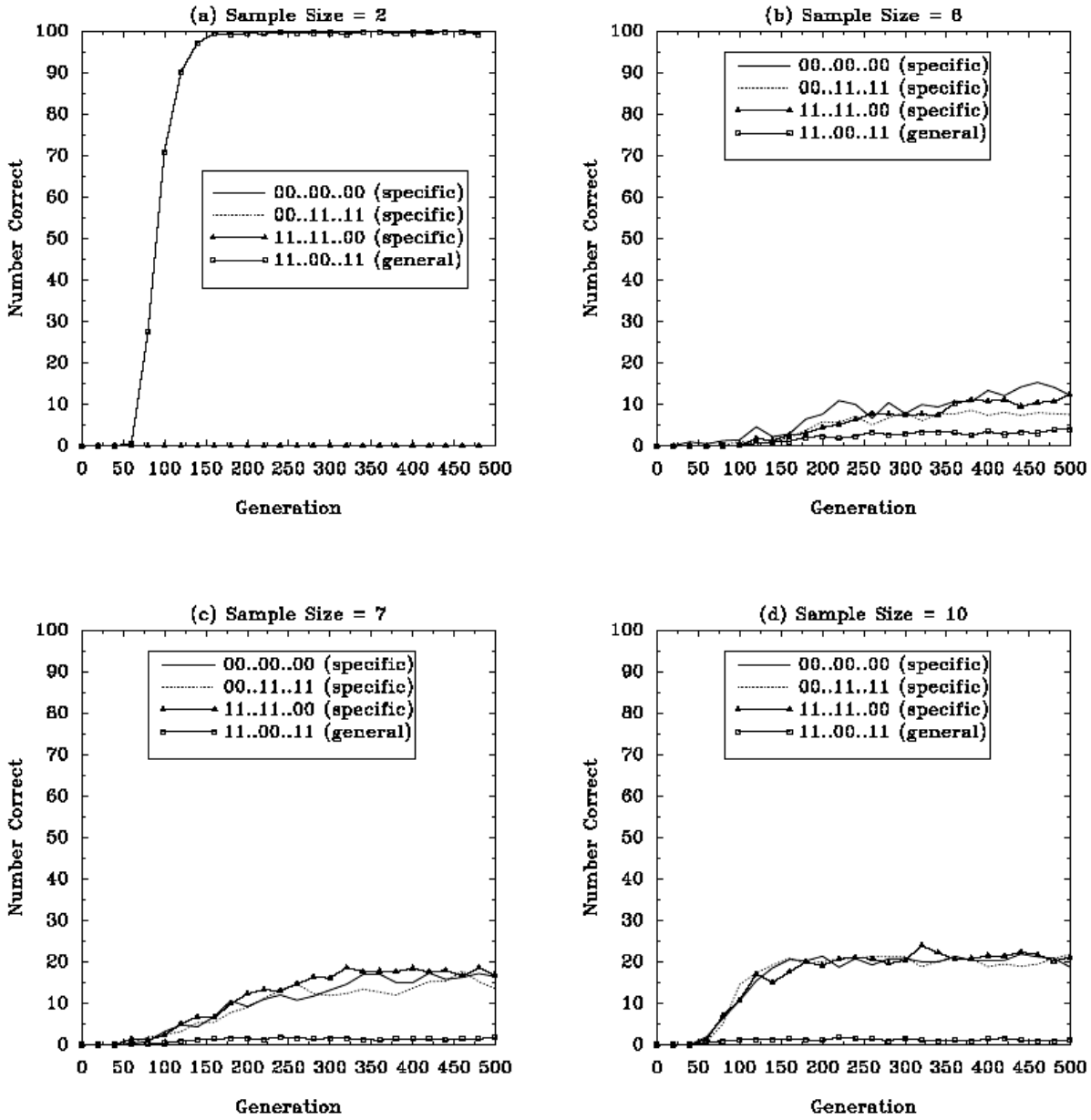

Abbildung 12: Ausbildung von Generalisten und Spezialisten bei unterschiedlichem σ

5. Zusammenfassung und Fazit

Wir haben ein Immunsystem-Modell kennen gelernt, das mit einer Matching-Funktion arbeitet, die spezifische Lösungen mehr belohnt als Unspezifische. Die Matching-Funktion spielt eine sehr wichtige Rolle in dem Modell, da sie dafür sorgt, dass nur fremde Individuen erkannt werden, also kein "Self-recognition" auftritt.
Der genetische Algorithmus ist sowohl in der Lage, gemeinsame Muster bei Antikörpern zu erkennen, als auch verschiedene Lösungen für ein Antigenproblem gleichzeitig aufrecht zu erhalten. Die Ähnlichkeiten der Antigen-Peaks beeinflussen dabei den genetischen Algorithmus nicht. Er benötigt etwa 15 Antikörper pro Peak um stabil zu bleiben.
Alles in allem ist ein genetischer Algorithmus also eine recht gute Approximation des reales Immunsystems, der unter anderem auch dabei hilft, biologische Abläufe nachvollziehbarer zu machen. Es sollte aber noch erwähnt werden, dass der vorgestellte Algorithmus nicht der

Einzige ist der die gegebenen Problemstellungen erfolgreich lösen kann. Der Fitness-Sharing Algorithmus arbeitet ähnlich und mit vergleichbarer Effektivität.

Literatur:

[1] Hartmut Pohlheim. *Evolutionäre Algorithmen* . Springer(1999)

[2] Stephanie Forrest, Brenda Javornik, Robert E. Smith, Alan S. Perelson. *Using Genetic Algorithms to Explore Pattern Recognition in the Immune System.* (1993)

[3] Leandro Nunes de Castro, Fernando J. Von Zuben. *Artificial Immune Systems: Part 1- Basic Theory and Applications.* (1999)

[4] John H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press (1975)

[5] L. Ivantysynova, L. Dölle. *Genetische Algorithmen.*