Genetische Algorithmen

Shawn Keen

Zusammenfassung

Eine weitere Herangehensweise an das maschinelle Lernen ist die Nachahmung evolutionärer Prozesse. Hier wollen wir uns mit den sogenannten *Genetischen Algorithmen* befassen.

1 Biologischer Hintergrund

Die Evolution hat (Darwins These folgend) äusserst effiziente Lebewesen hervorgebracht. Sie ist aber kein auf ein bestimmtes, festgesetztes Ziel gerichteter Prozess. Vielmehr handelt es sich dabei um die Optimierung von Individuen, die gegeneinander um Resourcen konkurrieren. Einige Indiviuen sind dafür besser geeignet, haben daher grössere Überlebenschancen und können ihre Eigenschaften an Nachkommen weitergeben, andere nicht. Diese Eigenschaften sind im Erbgut festgelegt. Einen einzelnen Parameter, der eine Eigenschaft beschreibt nennen wir Gen. Ein ganzer Satz solcher Gene ist ein Chromosom. Wichtig ist dabei eine Codierung der Information festzulegen, sodass Gene immer der jeweils selben Eigenschaft zugeordnet werden können.

Für die Weiterentwicklung gibt es drei Kriterien:

Kreuzung Das neue Erbgut setzt sich aus einem Teil des Elternteils A und aus einem Teil des Elternteils B zusammen. Die Erbinformation wird also *rekombiniert*.

Mutation Auftretende Fehler führen zu einer Veränderung der Erbinformation.

Selektion Nach bestimmten, festgelegten Kriterien werden die am besten geeignetsten Individuen zur Fortpflanzung ausgewählt.

Die Evolutionsvorgänge wollen wir nun durch einen konkreten Ablauf, einen Algorithmus, beschreiben.

20.Juni 2004

2 Genetische Algorithmen

Diese Art der Algorithmen geht auf John Holland und D.E.Goldberg zurück. Ein GA (*Genetischer Algorithmus*) besteht grob aus den folgenden Schritten:

1. Überführe das Problem in eine Zielfunktion und codiere dementsprechend die Variablen der Funktion als Chromosomen. Bestimme die Grösse N der Population und die Abbruchkriterien. Lege die Kreuzungs- und die Mutationswahrscheinlichkeit (p_k und p_m) fest.

2. Definiere eine Bewertungsfunktion, um die Güte (Fitness) eines Individuums bestimmen zu können. Von der Güte ist abhängig, ob das Individuum zur Kreuzung tauglich ist.

3. Generiere eine zufällig gewählte Anfangspopulation.

4. Berechne die Güte jedes einzelnen Individuums.

5. Wähle basierend auf ihrer Güte Paare zur Kreuzung aus.

6. Führe die genetischen Operatoren Kreuzung und Mutation aus.

7. Wiederhole ab Schritt 5, bis die Anzahl der Kinder N erreicht hat.

8. Ersetze die Elternpopulation durch die Kinderpopulation.

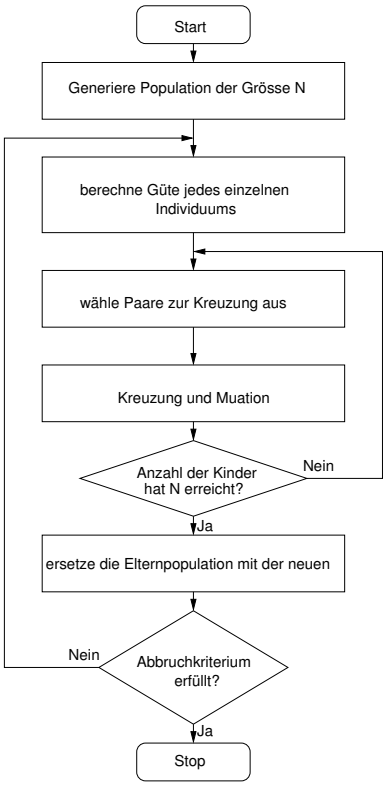
9. Wiederhole ab Schritt 4, bis ein Abbruchkriterium erfüllt ist.

Jeder Durchlauf repräsentiert eine *Generation*.

2.1 Abbruchkriterien

Als Abbruchkriterium bietet sich natürlich zunächst einmal das Erreichen der Zielfunktion an. Dabei ist aber zu beachten, dass Genetische Algorithmen stochastische Suchalgorithmen sind. Anschlaulich heisst das: Der GA sucht bei der Ausführung den Raum der möglichen Werte seiner Chromosomen ab. Wobei jedes Individuum durch einen Punkt in diesem Raum repräsentiert wird. Jedem Gen kann eine Achse zugeordnet werden. Es ist klar, dass nicht jeder Punkt im Raum geprüft werden kann und damit das absolute Optimum nicht unbedingt erreicht wird. Also müssen wir einen Abbruch bei hinreichender Nähe zur Zielfunktion zulassen. Ausserdem sollte der GA, falls er sich

Abbildung 1. Diagramm eines Genetischen Algorithmus



dem Optimum nicht schnell genug nähert, nach einer maximalen Anzahl von Generationen aufhören.

2.2 Die Codierung

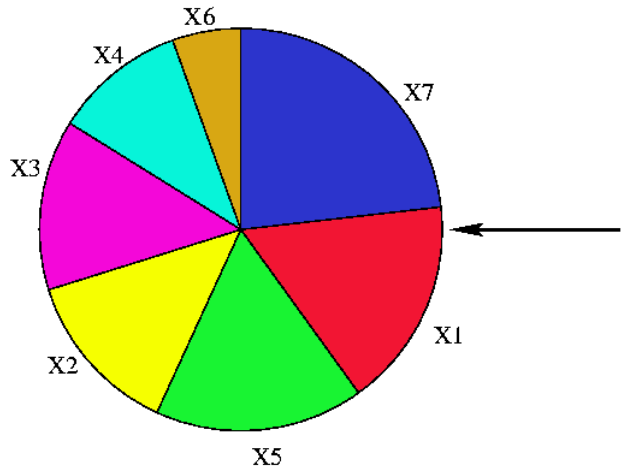
Jedes Individuum wird durch ein Chromosom mit entsprechenden Genen charakterisiert. Ein Gen besteht aus einem oder mehreren Bits. Abhängig vom Problem können die Gene Variablen einer Funktion, Synaptische Gewichte von neuronalen Netzen, Koeffizienten eines Polynoms etc. repräsentieren. Bestehen die Gene selber aus einem Bitmuster, werden sie zu einem String von Bits zusammengesetzt und bilden auf diese Weise das Chromosom. Die Gene g_1 bis g_4 mit $g_1 = 0$, $g_2 = \langle 1, 0\rangle$, $g_3 = 1$ und $g_4 = \langle 1, 1\rangle$ bilden das Chromosom $\langle g_1, g_2, g_3, g_4\rangle$ bzw. $\langle 0, \langle 1, 0\rangle, 1, \langle 1, 1\rangle\rangle$, also den String 010111. Wichtig ist, dass bei der Berechnung der Fitness, bei der Kreuzung und bei der Mutation darauf geachtet wird, welches Bit zu welchem Gen gehört. Wieviele Gene nötig sind, hängt von der Anzahl der Parameter ab. Der Wertebereich des Parameters bestimmt die Anzahl der Bits im Gen. Allerdings kann die Länge eines Bits auch dynamisch an den derzeitigen Wert des Parameters angepasst werden.
Im folgenden gehen wir der Einfachheit halber davon aus, dass es sich bei den betrachteten Genen um einzelne Bits handelt.

2.3 Selektion

Üblicherweise werden aus zwei Elternteilen zwei Kinder erzeugt. Die fittesten Individuen bringen mit grosser Wahrscheinlichkeit auch die fittesten Nachkommen hervor. Ein *Heirats-Schema* wählt für ein Individuum gemäß seiner Güte einen Partner aus. Bei einem der am häufigsten verwendeten Schemata, die *Roulette-Wheel-Selektion*, wird erst die Gesamtfitness berechnet und danach für jedes Individuum das Verhältnis zwischen seiner Fitness und der Gesamtfitness ermittelt. Sein Fitnessverhältnis ist (als Prozentwert gesehen) die Wahrscheinlichkeit, dass ein Individuum zur Kreuzung ausgewählt wird. Auf die selbe Weise wird auch der Partner für das so ausgesuchte Elternteil ermittelt.

Veranschaulicht wird das durch eine Rouletteschеibe, auf der jedes Chromosom einen Bereich gemäss seines Fitnessverhältnisses erhält. Nachdem die Scheibe gedreht wurde, gilt das Individuum, bei dem der Zeiger stehengeblieben ist, als ausgewählt. Das ganze wird zwei mal durchgeführt. Dabei kann es auch vorkommen, dass Individuen zur Kreuzung mit sich selbst bestimmt werden.

Abbildung 2. Rouletteschеibe mit den Chromosomen X1 bis X7

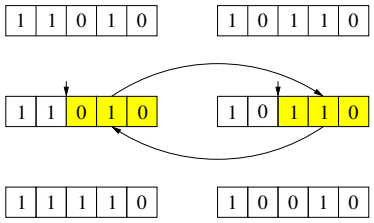


2.4 Kreuzung

Sind zwei Partner ausgewählt worden, wird das Erbgut gemischt. Der Kreuzungsoperator wählt zufällig eine Stelle im Chromosom aus. Alle Gene vor dieser Stelle bleiben erhalten, die Gene danach werden mit denen des Partners ab dieser Stelle getauscht. Es entstehen zwei Kinder.

Ob zwei Chromosome überhaupt gekreuzt werden, hängt ausserdem von der Kreuzungswahrscheinlichkeit p_k ab. Hiermit können wir den Prozeß feinjustie-

Abbildung 3. Kreuzung zweier Chromosome



ren. $p_k = 0,7$ liefert in der Regel recht gute Ergebnisse. Sollte ein Paar von der Kreuzung ausgeschlossen worden sein, werden die Chromosome einfach kopiert. Die Kinder sind also Klone ihrer Eltern.

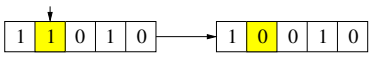
Nach der Kreuzung sind die Kinder nicht unbedingt genauso lang, wie ihre Eltern, wenn sich die Länge der Gene dynamisch an den derzeitigen Wert ihres Parameters anpasst.

2.5 Mutation

Mutation kommt in der Natur recht selten vor. Sie kann zwar zu einigen Verbesserungen führen, richtet meist aber eher Schaden an. Im Bereich der GA hat sich Mutation als recht sinnvoll erwiesen, da hiermit verhindert werden kann, dass der GA an einem lokalen Optimum ,,hängenbleibt". In einem solchen Fall stagniert die Weiterentwicklung, die Population wird zunehmend homogener und es werden nur noch identische Individuen miteinander gekreuzt, ohne dass das tatsächliche Optimum je erreicht wird. Die Mutation sorgt dafür, dass einzelne Chromosomen das lokale Optimum verlassen und eine neue Richtung eingeschlagen werden kann. Der Suchraum wird ,,breiter" durchschritten und der Verlust von Vielfältigkeit verhindert.

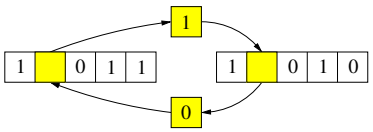
Bei der Anwendung kommt es vor allem auf die Codierung der Chromosome an. Mutiert wird jeweils nur ein Bit. Es wird gemäß der Mutationswahrscheinlichkeit p_m bestimmt, ob ein Chromosom mutiert wird, und dann zufällig ein Bit geändert.

Abbildung 4. Bitweise Mutation eines Chromosoms



Eine weitere Methode ist, bei zwei Chromosomen, die zur Mutation gewählt wurden, ein Gen zu tauschen und dadurch die Mutation durchzuführen. Wichtig ist natürlich, dass dabei gleichwertige Gene getauscht werden.

Abbildung 5. Mutation durch Tauschen



Der Wert eines Bits ist abhängig von seiner Position im Gen. Das führt unter Umständen zu ganz erheblichen Veränderungen durch die Mutation. Die Binärdarstellung der Zahl 4 ist 0100. Durch Veränderung des letzten Bits erhalten wir 0101, also 5. Verändern wir aber das erste Bit, landen wir bei 12 (1100)! Wenn wir solche Effekte verhindern wollen, müssen wir zu anderen Codierungsverfahren greifen. Eine derartige Alternative ist die *Gray-Codierung*, bei der sich benachbarte Dezimalwerte nur um ein Bit unterscheiden.

Dezimal	Binärcode	Gray-Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Der Wert eines Bits ist hier aber immer noch Positionsabhängig. Wir können diesen Umstand umgehen, indem wir die Mutationswahrscheinlichkeit zusätzlich von der Bitposotion abhängig machen.

Fassen wir zusammen: Die Fitness jedes einzelnen Chromosoms wird ermittelt. Daraufhin wird im Heirats-Schema festgelegt, welche Chromosomen rekombiniert werden sollen. Die durch Kreuzung entstandenen Kinder werden dann der Mutation unterzogen und können nun ihre Eltern ersetzen.

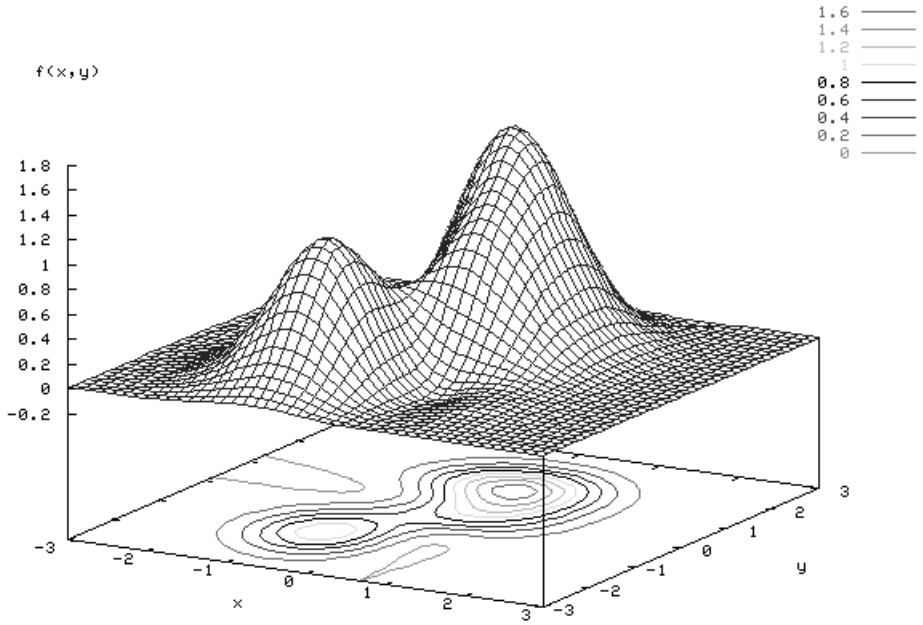
3 Ein Beispiel-GA

Nehmen wir an, wir wollten das Maximum der Funktion

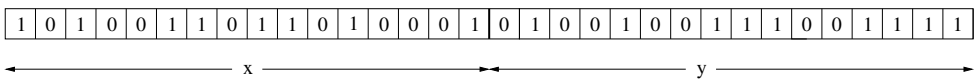
$$f(x,y) = (0,7-x)^2 e^{-x^2-(y+1,3)^2} - (x-(x+1)^2-y^3)e^{-x^2-(y-0,5)^2}$$

ermitteln. Zielfunktion und Bewertungsfunktion sind in unserem Beispiel identisch.

Abbildung 6. Beispielfunktion



Unser Problem hat zwei Parameter x und y mit dem Wertebereich $[-3 \ldots 3]$. Wir repräsentieren x und y als einen zusammengesetzten Binärstring:

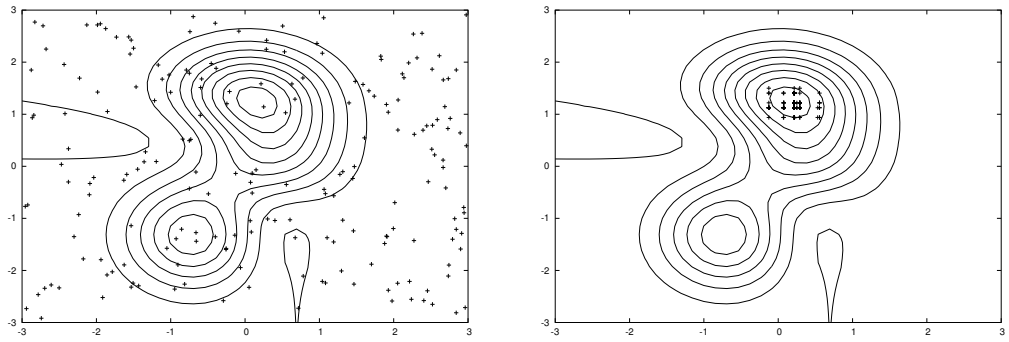


Jeder Parameter ist 16 Bit lang. Die Populationgrösse soll $N = 200$ sein.

Um die Güte der Chromosomen zu berechnen, wird der String aufgespalten und die Binärzahlen in reelle Zahlen decodiert. Dann werden x und y in $f(x,y)$ eingesetzt und der Wert der Funktion berechnet. Zur Selektion wird die bekannte Roulette-Wheel-Methode angewandt. Mutation geschieht durch Verändern einer Stelle im Bitstring.

In unserem ersten Versuch ist die Kreuzungswahrscheinlichkeit 0,7 und die Mutationswahrscheinlichkeit 0,001. Die Populationgrösse soll $N = 200$ sein.

Abbildung 7. Die Anfangsopulation (links) und nach 20 Generationen (rechts)



Wie wir in Abbildung 7 sehen, steuern die Werte bereits nach 20 Generationen gut auf das Maximum der Funktion zu.

Bei 100 Generationen haben wir den optimalen Wert schon fast erreicht. Alle Individuen sind fast identisch und so ist auch nach 500 Generationen keine Veränderung im Erbgut abzusehen.

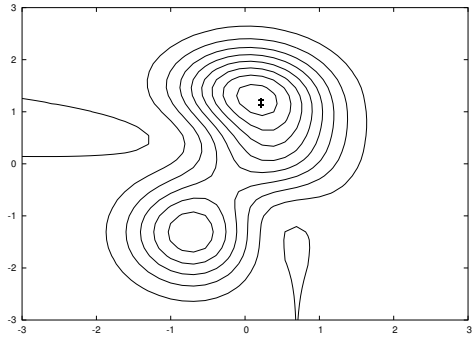


Abbildung 8. 100 Generationen

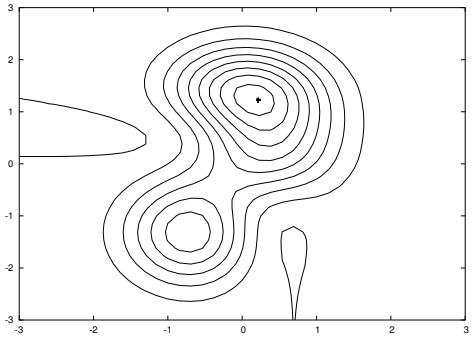


Abbildung 9. 500 Generationen

Die Individuen, die schon bei der Initialisierung dicht am globalen Maximum lagen, konnten schnell das Übergewicht bei der Selektion erhalten. Was passiert aber, wenn sich die Anfangspopulation garnicht mit dem globalen Maximum überschneidet? Probieren wir das mit 20 Individuen:

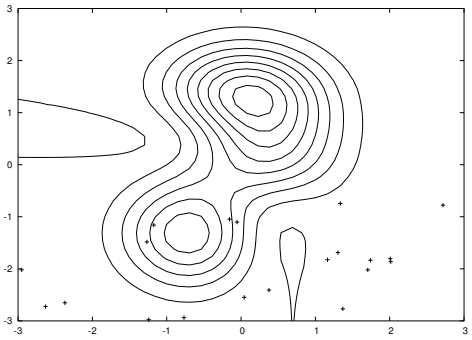


Abbildung 10. Anfangspopulation

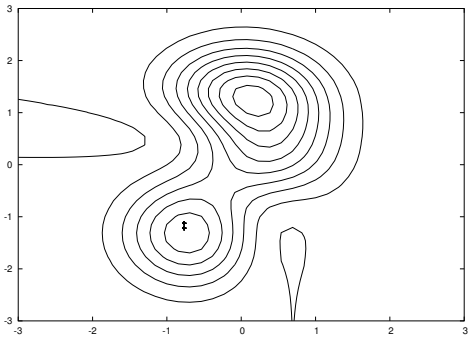


Abbildung 11. 20 Generationen

Es wird nur das lokale Maximum erreicht, da nur selten mutiert wird. Wir sollten also die Mutationswahrscheinlichkeit hoch setzen. Mit $p_m = 0,9$ erhalten wir:

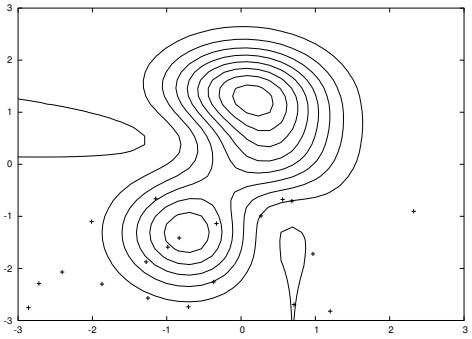


Abbildung 12. Anfangspopulation

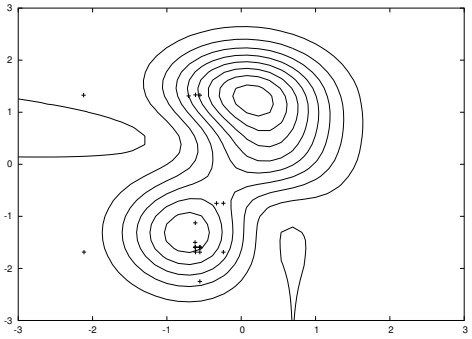


Abbildung 13. 20 Generationen

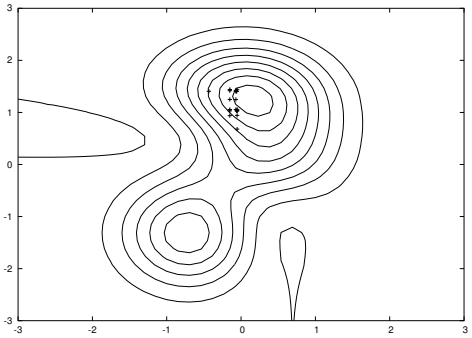


Abbildung 14. 50 Generationen

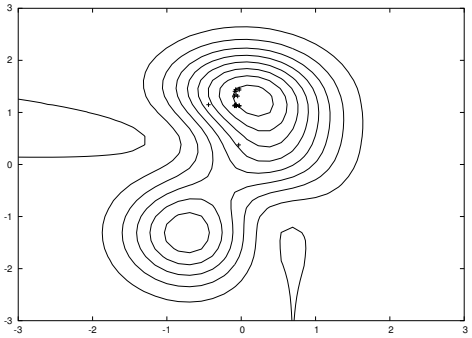


Abbildung 15. 100 Generationen

Die verstreuten Individuen in Abbildung 13 reichen aus, um den GA noch auf das globale Maximum ,,umzuschwenken".

4 Effizienz und Optimierung

Wir kennen jetzt den Ablauf eines Genetischen Algorithmus und haben an einem Beispiel gesehen, dass Genetische Algorithmen tatsächlich gute Ergebnisse liefern können.
Ein GA nimmt N (Populationsgrösse) Stichproben und bewertet sie, er untersucht also mit minimalem Aufwand parallel die Güte von $O(N^3)$ Unterräumen. Man spricht daher auch von *implizierter Parallelität*.
Doch wie kann er dabei effizient sein?

Für die Weiterentwicklung sind die Selektion, Kreuzung und Mutation die Hauptfaktoren. Zentral ist hierbei die Codierung der Werte im Chromosom.

4.1 Das Schema-Theorem

Um die Auswirkungen der genetischen Operatoren genauer analysieren zu können, führte John Holland den Begriff des *Schemas* ein. Ein Schema ist ein Chromosom, das an verschiedenen Stellen nicht feste Werte, sondern eine Variable hat. Mit dem Variablensymbol '#' ("don't care") ist ein mögliches Beispiel für ein Schema:

$$h_1 = \langle 1, \#, 1, \#, 0, 0 \rangle$$

'#' ist ein Platzhalter für die Werte 0 oder 1. Ein Schema der Länge n definiert damit einen Unterraum von M^n ($M = \{0,1\} \Leftrightarrow M^n = \{0,1\}^n$).
Ein Chromosom $x = \langle x_1, \ldots, x_n \rangle \in M^n$ ist eine *Instanz* des Schemas $h \in \{0, 1, \#\}$, falls die Stellen, an denen im Schema keine '#' stehen, übereinstimmen. Für M^n gibt es 3^n Schemata. h_1 wird von vier Chromosomen instanziert:

$\langle 1, 1, 1, 1, 0, 0 \rangle$
$\langle 1, 0, 1, 1, 0, 0 \rangle$
$\langle 1, 1, 1, 0, 0, 0 \rangle$
$\langle 1, 0, 1, 0, 0, 0 \rangle$

Die *definierende Länge* $L(h)$ eines Schemas h ist der Abstand zwischen der ersten und der letzten von '#' verschiedenen Stelle im Chromosom, die *Ordnung* $o(h)$ die Anzahl der Positionen, die nicht '#' enthalten.
Beispiele:
$L(\langle 0, 1, 1, \#, 1, \#, \# \rangle) = 4$, $L(\langle 1, \#, \#, \# \rangle) = 0$, $L(\langle 1, \#, \#, \#, 1) = 4$
$o(\langle 1, 1, \#, \#, 1 \rangle) = 3$, $o(\langle 1, 0, 1, 1 \rangle) = 4$, $o(\langle \#, \#, \# \rangle) = 0$

Um herauszufinden, ob sich die Güte einer Population wirklich durch die genetischen Operatoren verbessern lässt, reduzieren wir das Problem auf die Schemata.

Es sei $m_h(i)$ die Häufigkeit von Instanzen eines Schemas h in der i-ten Generation der Polulation und $\hat{f}_h(i)$ die durchschnittliche Fitness dieser Instanzen. Uns interessiert, wieviele Instanzen eines Schemas in der nächsten Generation $i+1$ vorhanden sein werden. Da die Fortpflanzungswahrscheinlichkeit proportional zur jeweiligen Fitness ist, haben wir

$$m_x(i+1) = \frac{f_x(i)}{\hat{f}(i)}$$

Nachkommen eines Chromosoms x. Damit ergibt sich für alle Instanzen von h eine Nachkommenanzahl von

$$m_h(i+1) = \frac{\sum_{x=1}^{m_h(i)} f_x(i)}{\hat{f}(i)}, \qquad x \in h.$$

Da die Durchschnittsgüte der ganzen Population in der i-ten Generation

$$\hat{f}_h(i) = \frac{1}{m_h(i)} \sum_{x=1}^{m_h(i)} f_x(i), \qquad x \in \{1, \ldots, N\}$$

ist, erhalten wir

$$m_h(i+1) = \frac{\hat{f}_h(i) m_h(i)}{\hat{f}(i)}.$$

Zusätzlich überlegen wir uns, wie die Art eines Schemas sich auf die Fortpflanzungswahrscheinlichkeit auswirkt. Es ist recht klar, dass Schemata mit grosser definierender Länge und hoher Ordnung von den genetischen Operatoren eher ,,aufgebrochen" werden, also weniger Chancen zu überleben haben.
Auf dieser Basis entwickelte Holland sein *Schema-Theorem*:

$$m_h(i+1) \geqslant m(i) \frac{f_h(i)}{\hat{f}(i)} \Big(1 - p_k \frac{L(h)}{N-1} - p_m o(h)\Big)$$

Wobei h das Schema, $f_h(i)$ die Fitenss von h und $\hat{f}(i)$ die Durchschnittsfitness in der i-ten Generation ist. $L(h)$ ist die definierende Länge und $o(h)$ die Ordnung des Schemas h. N bezeichnet wieder die Populationsgrösse und p_k, p_m die Kreuzungs- bzw. Mutationswahrscheinlichkeit.

Bei der Codierung ist also immer darauf zu achten, dass zusammengehörige genetische Informationen auch zusammenhängend codiert werden. Besonders wichtige Parameter sollten möglichst kompakt codiert werden, damit sie nach der Kreuzung weiter intakt sind.

Wie wir an unserem Beispiel gesehen haben, nähert sich der Algorithmus ziemlich schnell an ein Optimum an. Das macht es recht schwierig GAs auf Daten anzuwenden, die sich schnell ändern. Lösen könnte man dieses Problem, indem man die Mutationswahrscheinlichkeit stark erhöht, sobald keine befriedigenden Ergebnisse mehr erreicht werden oder durch Einführen zufällig gewählter neuer Individuen in die Population.
Genetische Algorithmen eignen sich weniger dazu, ein Optimum genau zu treffen. Man kann mit ihnen aber das Optimum schnell einkreisen.
Der Hauptnachteil Genetischer Algorithmen ist der zu betreibende Rechenaufwand. Es muss jedes einzelne Individuum pro Generation neu evaluiert, selektiert, gekreuzt und ggf. mutiert werden.

5 Einsatz und Weiterführendes

Da es sich bei den Genetischen Algorithmen um ein (für Informatikverhältnisse) bereits recht altes Konzept handelt, ist hier schon viel geforscht und ausprobiert worden. Daher kann hier nur ein kleiner Ausschnitt der breiten Palette von Einsatzmöglichkeiten gezeigt werden.

- Traveling-Salesman-Problem
- Erstellen von Zeitplänen
- Entwickeln von Zellularautomaten
- Einstellen Synaptischer Gewichte neuronaler Netzwerke

Ausserdem entwickelte John Koza das *Genetische Programmieren*, das sich mit der automatischen Generierung von Programmen beschäftigt. Hier werden nicht Parameter optimiert, sondern Programme bzw. Programmteile.

Die Kreuzung ist bei GA von grosser Bedeutung. Die Parameter werden extra codiert und müssen wieder decodiert werden, was, wie wir gesehen haben, mit einigen Nachteilen verbunden ist. Wie wäre es, wenn wir die Parameter als reelle Zahlen hinterlegen? Diese und andere Überlegungen finden sich bei den *Evolutionsstrategien* wieder, die allerdings nicht Teil dieser Arbeit sind.

Literatur

[1] Jochen Heistermann: Genetische Algorithmen; Teubner, Stuttgart 1994

[2] Michael Negnevitsky: Artificial Intelligence, A Guide to Intelligent Systems; Addison-Wesley, Harlow 2002

[3] Rainer Hellmich: Einführung in intelligente Softwaretechniken; Prentice Hall, München 1997

[4] Eberhard Schöneburg: Genetische Algorithmen und Evolutionsstrategien; Addison-Wesley, Bonn 1994