h_da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Alexander del Pino
Fachbereich Informatik
Sommersemester 2007

Genetische Algorithmen

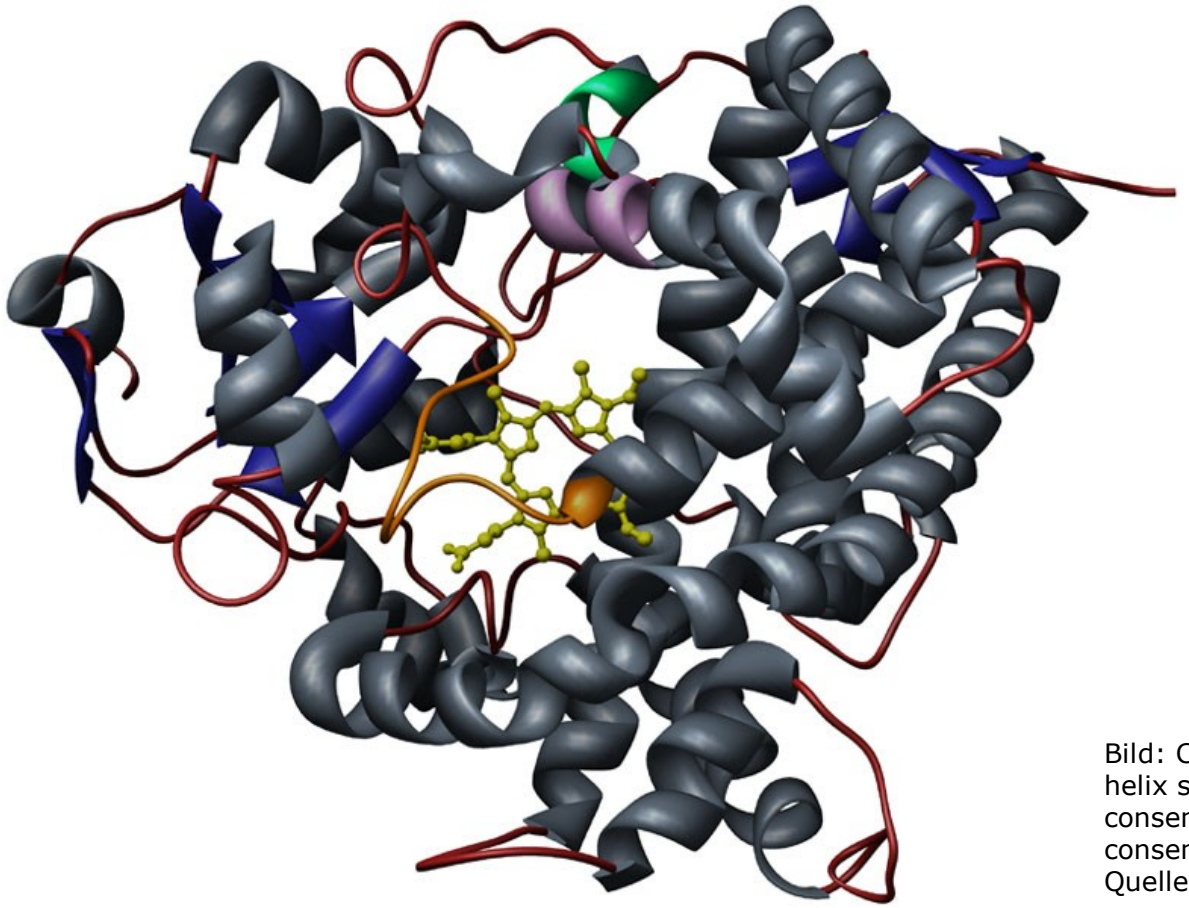

Bild: CYP3A4 [1TQN, Homo sapiens] ribbon detail. Color key: helix structures, gray; strand structures, blue; PERF consensus, green; K-helix concensus, purple; heme-binding consensus, orange; heme ligand, yellow
Quelle: http://p450.kvl.dk/gallery/CYP3A4.jpg

9. Teil

Das Schematheorem

Das Schematheorem

Worum geht es ?

Bisher haben wir uns hauptsächlich damit beschäftigt, zu verstehen wie ein genetischer Algorithmus funktioniert.

Dazu haben wir uns zuerst den grundsätzlichen Aufbau eines typischen genetischen Algorithmus angeschaut und anschließend Details betrachtet.

Jetzt geht es um die Frage:

> Warum funktioniert ein genetischer Algorithmus überhaupt ?

Dabei müssen wir folgendes beachten:

- Genetische Algorithmen sind große, komplexe Systeme und eignen sich zum Finden von Näherungslösungen für gerade solche Probleme, die oftmals selbst auch nur schwer zu beschreiben sind.
- Wegen der Verwendung von Zufallszahlen kann man allenfalls versuchen, mit Hilfe der Statistik das durchschnittliche Verhalten eines genetischen Algorithmus besser zu verstehen.

Das Schematheorem

Warum funktioniert ein genetischer Algorithmus ?

Es ist also eher *unwahrscheinlich*, für diese Frage ein exaktes Modell zu finden, welches das Verhalten eines genetischen Algorithmus für ein beliebiges Optimierungsproblem hinreichend genau beschreibt.

- *A. Eiben* zieht hierzu einen interessanten Vergleich:

> *Moreoever, while the field of EAs is fairly young, it is worth noting that the field of population genetics and evolutionary theory has a head start of more than a hundred years, and is still battling against the barrier of complexity.*
>
> Quelle: A. E. Eiben, J. E. Smith: *Introduction to Evolutionary Computing*. Springer-Verlag, 2003

- *J. Holland* schlug 1975 einen Erklärungsansatz für einen einfachen genetischen Algorithmus vor, der unter dem Begriff *Schematheorem* bekannt wurde.

Das Schematheorem

Ein einfacher genetischer Algorithmus

Betrachten wir einen einfachen genetischen Algorithmus.

- Die Population P besteht aus n Lösungskandidaten.

$$P = \{k_1, k_2, k_3, \ldots, k_n\}$$

- Die Lösungskandidaten werden als einfache Bitstrings der Länge l kodiert.
- Die Wahrscheinlichkeit, dass ein bestimmter Lösungskandidat k_i selektiert wird, ist proportional zu seiner Fitness.

$$p(k_i) = f(k_i) / \sum_{i=1}^{n} f(k_i)$$

- Die Rekombination erfolgt durch 1-Punkt-Crossover mit einer Wahrscheinlichkeit p_c.
- Die Mutation erfolgt bitweise unabhängig mit einer Wahrscheinlichkeit p_m.

Das Schematheorem

Was ist ein Schema ?

Ein *Schema* ist die Beschreibung eines Bitmusters, wobei neben den Symbolen 0 und 1 auch noch das **-Symbol* für solche Bitstellen verwendet wird, die einen beliebigen Wert annehmen dürfen *(don't care)*.

- Eine *Instanz eines Schemas* ist ein Bitstring, der auf dieses Schema passt, wobei anstelle der *-Symbole beliebig 0 und 1 verwendet werden darf.
- Beispiel

  Zu dem Schema **10* gibt es genau acht verschiedene Instanzen.

  Die rot markierten Bits sind hier durch das Schema vorgegeben, die grün markierten Bits ersetzen die *-Symbole:

  00100, 00101, 01100, 01101, 10100, 10101, 11100 und 11101.

-  Der Bitstring 01010 passt nicht zu diesem Schema. Warum ?

Das Schematheorem

Beschreibung von Schemen

Es gibt zwei Merkmale, die man bei einem Schema S betrachten kann.

- *Die Ordnung o(S)* des Schemas *S* ist die Anzahl der 0 und 1 Symbole im Schema *(order of the schema)*.
- *Die Distanz d(S)* des Schemas *S* ist der Index des am weitesten rechts stehenden 0 oder 1 Symbols minus dem Index des am weitesten links stehenden 0 oder 1 Symbols *(defining length of the schema)*.

Beispiel

S = *1**110*0**

o(S) = 5

d(S) = 9 – 2 = 7

Das Schematheorem

Aussagen und Beobachtungen zu Schemen

Wie würden Sie folgende Aussagen zu Schemen plausibel begründen ?

- Ein Bitstring der Länge k ist immer eine Instanz von 2^k Schemen.
- Mit einem Bitstring der Länge k können 3^k Schemen definiert werden.
- Eine Population von n Bitstrings der Länge k enthält mindestens 2^k Instanzen von Schemen.
- Eine Population von n Bitstrings der Länge k enthält höchstens $n * 2^k$ Instanzen von Schemen.
- Zu einem Schema S sind bei einem Bitstring der Länge k genau $2^{k-o(S)}$ verschiedene Instanzen denkbar.

Der einfache genetische Algorithmus verarbeitet *explizit* zwar nur die n Bitstrings der aktuellen Population, aber *implizit* eine weitaus größere Anzahl an Schemen.

Das Schematheorem

Die echte durchschnittliche Fitness eines Schemas

Die echte durchschnittliche Fitness eines Schemas ist die durchschnittliche Fitness aller möglichen Instanzen des Schemas.

- Beispiel

  Bei Bitstrings der Länge k=40 gibt es 2^{32} verschiedene Instanzen eines Schemas S mit $o(S)$=8.

  Um die echte durchschnittliche Fitness eines solchen Schemas zu berechnen müsste man also 2^{32} mal die Fitnessfunktion aufrufen.

- Wenn man berücksichtigt, dass es in diesem Beispiel $3^{k} \approx 1.2 * 10^{19}$ Schemen der Ordnung 8 gibt, wird klar, dass man die echte durchschnittliche Fitness der Schemen aus praktischen Gründen nicht explizit berechnen kann.

Das Schematheorem

Die durchschnittliche Fitness eines Schemas

Stattdessen betrachtet man als Näherungswert die durchschnittliche Fitness derjenigen Lösungskandidaten, die sich zu dem Zeitpunkt *t* tatsächlich in der Population befinden und Instanzen des betrachteten Schemas sind.

- Beispiel

Kandidat	**Fitness**
011001	45
100101	50
010100	15
101100	25
001110	60

Welche durchschnittliche Fitness besitzt das Schema ***10* in dieser Population ?

- Das Schema ***10* besitzt drei Instanzen in dieser Population, nämlich 100101, 010100 und 101100. Die durchschnittliche Fitness dieses Schemas beträgt also (50 + 15 + 25) / 3 = 90 / 3 = 30.

Das Schematheorem

Wirkung der Selektion

Sei *n(S,t)* die Anzahl der vorhandenen Instanzen des Schemas *S* zum Zeitpunkt *t*, und $x \in S$ eine Instanz von *S*. Die durchschnittliche Fitness von *S* beträgt dann wie in dem vorigen Beispiel gezeigt:

$$f(S,t) = \frac{\sum_{x \in S} f(x)}{n(S,t)} \quad (1)$$

Die zu erwartende Anzahl der Instanzen von *S* in der nächsten Generation *t*+1 ist abhängig von der Fitnessproportionalität der aktuellen Instanzen von *S*:

$$E(n(S,t+1)) = \sum_{x \in S} \frac{f(x)}{F(t)} \quad (2)$$

wobei *F(t)* die durchschnittliche Fitness in der aktuellen Population ist:

$$F(t) = \frac{1}{n} * \sum_{x=1}^{n} f(x) \quad (3)$$

Das Schematheorem

Wirkung der Selektion

(2) lässt sich wie folgt umschreiben:

$$E(n(S,t+1)) = \sum_{x \in S} \frac{f(x)}{F(t)} = \frac{1}{F(t)} * \sum_{x \in S} f(x) \quad (2')$$

(1) kann durch Multiplikation von *n(S,t)* auf beide Seiten wie folgt umgestellt werden:

$$f(S,t) = \frac{\sum_{x \in S} f(x)}{n(S,t)} \quad \Leftrightarrow \quad \sum_{x \in S} f(x) = n(S,t) * f(S,t) \quad (4)$$

Einsetzen von (4) in (2') liefert:

$$E(n(S,t+1)) = n(S,t) * \frac{f(S,t)}{F(t)} \quad (5)$$

Die erwartete Anzahl der Instanzen eines Schemas wächst also proportional zur durchschnittlichen Fitness des Schemas.

Das Schematheorem

Wirkung der Rekombination

Bei Bitstrings der Länge *l* kann der Crossover-Operator an *l*-1 Stellen wirken.

Die Wahrscheinlichkeit $p_z(S)$ dass eine Instanz des Schemas *S* durch Crossover zerstört wird, hängt dabei maßgeblich von der Distanz *d(S)* des Schemas ab:

$$p_z(S) = \frac{d(S)}{l-1} \quad (6)$$

Allerdings wird der Crossover-Operator ja nicht auf jeden Bitstring angewendet, sondern der Bitstring wird mit einer Wahrscheinlichkeit p_c dafür ausgewählt:

$$p_z(S) = p_c * \frac{d(S)}{l-1} \quad (7)$$

Die Wahrscheinlichkeit $p_{üc}(S)$ dass eine Instanz des Schemas *S* den Crossover-Operator überlebt, also nicht auseinander geschnitten wird, beträgt somit:

$$p_{üc}(S) = 1 - p_c * \frac{d(S)}{l-1} \quad (8)$$

Das Schematheorem

Wirkung der Rekombination

Weiterhin besteht eine - wenn auch kleine - Chance, dass die Schemainstanz nicht durch Crossover zerstört wird, weil beide Elternteile Instanzen dieses Schemas sind. Somit gilt also:

$$p_{üc}(S) \geq 1 - p_c * \frac{d(S)}{l-1} \qquad (9)$$

Aus (5) und (9) können wir die erwartete Anzahl der Instanzen eines Schemas in Abhängigkeit von dessen aktueller Anzahl, dessen Fitness und dessen Distanz wie folgt ausdrücken:

$$E(n(S,t+1)) \quad \geq \quad n(S,t) * \frac{f(S,t)}{F(t)} * \left(1 - p_c * \frac{d(S)}{l-1}\right) \qquad (10)$$

(10) zeigt, dass Instanzen von Schemen mit überdurchschnittlicher Fitness und mit kurzer Distanz bevorzugt weiterkommen. Umgekehrt tendieren Schemen mit unterdurchschnittlicher Fitness und/oder einer hohen Distanz dazu, auszusterben.

Das Schematheorem

Wirkung der Mutation

Die Wahrscheinlichkeit, dass ein Bit durch Mutation verändert wird, beträgt p_m.

Somit beträgt die Wahrscheinlichkeit $p_{üm}(S)$ dass keines der $o(S)$ Bits der Instanz des Schemas S durch die Mutation verändert wird:

$$p_{üm}(S) = (1 - p_m)^{o(S)} \tag{11}$$

Wenn wir also den Einfluß der Selektion, Rekombination und Mutation zusammen betrachten, erhalten wir:

$$E(n(S,t+1)) \geq n(S,t) * \frac{f(S,t)}{F(t)} * (1 - p_c * \frac{d(S)}{l-1}) * (1 - p_m)^{o(S)} \tag{12}$$

Dieser Ausdruck ist nun das Schematheorem von *John Holland*. Es besagt, dass sich Schemen mit überdurchschnittlicher Fitness, kleiner Ordnung und kurzer Distanz besonders gut weiterentwickeln.

Das Schematheorem

Die Baustein-Hypothese

Aus dem Schematheorem lässt sich die Baustein-Hypothese (building block hypothesis) ableiten.

- Die Baustein-Hypothese besagt, dass ein genetischer Algorithmus versucht, Schemen mit überdurchschnittlicher Fitness, kleiner Ordnung und kurzer Distanz - sogenannte Bausteine (building blocks) - so zusammenzusetzen, dass damit eine gute Lösung gefunden wird.
- *David Goldberg* beschreibt dies sehr treffend:

> Because highly fit schemata of low defining length and low order play such an important role in the action of genetic algorithms, we have already given them a special name: building blocks.
>
> Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemata, or building blocks.
>
> Quelle: D. Goldberg: *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989, S. 41

Das Schematheorem

Täuschungsprobleme

Wenn sich die Lösung eines genetischen Algorithmus tatsächlich durch Anordnen von Bausteinen ergibt, dann stellt sich folgende Frage:

> Wie reagiert ein genetischer Algorithmus, wenn die Lösung sich nicht aus aus solchen Bausteinen zusammensetzt ?

Beispiel

> Der Bitstring 00000000 repräsentiert die beste Lösung, besitzt also die höchste Fitness: f(00000000) = max.
>
> Die Fitness der anderen Bitstrings wird so definiert, dass sie für die Schemen mit 0 kleiner ist als für solche Schemen, wo die 0 durch 1 ersetzt wurde, z.B:
>
> f(*******0) < f(*******1)
>
> f(**0***0*) < f(**1***1*)

Solche Probleme nennt man Täuschungsprobleme (deception problems).

Das Schematheorem

Übungsaufgabe

Konstruieren Sie ein Beispiel für ein 4-Bit Täuschungsproblem.

Wie hoch ist die *echte durchschnittliche Fitness* der Schemen *00* und *01* ?

Betrachen Sie nun eine Population mit den folgenden fünf Lösungskandidaten:

1010

1001

0101

1010

0001

Wie sieht die *durchschnittliche Fitness* der beiden Schemen *00* und *01* in dieser Population aus ?