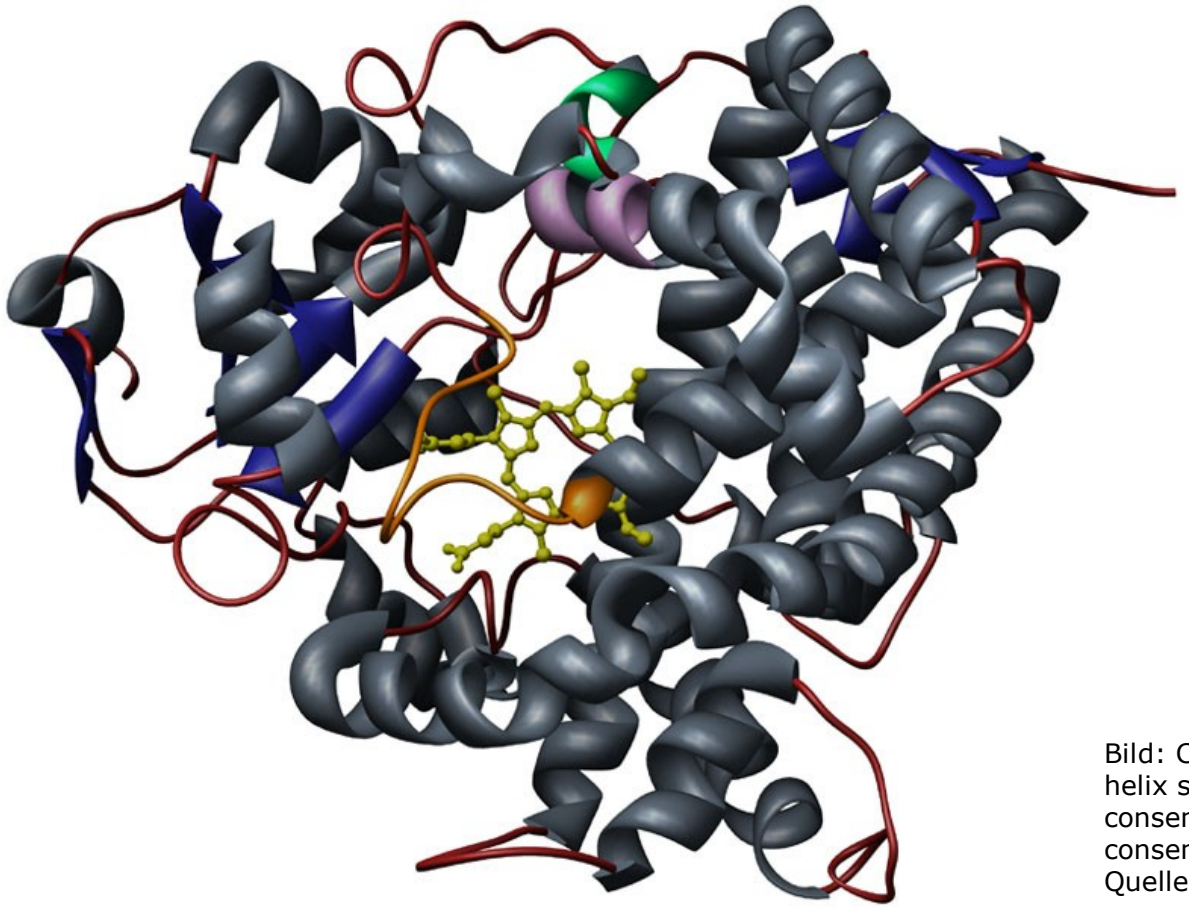
h_da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Alexander del Pino
Fachbereich Informatik
Sommersemester 2007

Genetische Algorithmen



3. Teil

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

Genetische Algorithmen

Die wichtigste Folie in dieser Vorlesung

Wir haben bisher folgendes herausgefunden:

- Viele wichtige Probleme in der Informatik sind Suchprobleme.
- Durch die Evolution bringt die Natur immer wieder Erstaunliches hervor.

Wie bringen wir nun beide Erkenntnisse zusammen ?

Genetische Algorithmen

Inspiration

Lösung von Informatikproblemen durch Aufgreifen von Analogien aus der Biologie:

- Die Suche nach einer guten Lösung .... Die Evolution als Prozess der die Anpassung von Individuen an eine Umgebung verbessert.
- Ein Lösungskandidat ... Ein Lebewesen.
- Mehrere Lösungskandidaten ... Eine Population.
- Die Güte eines Lösungskandidaten ... Die Fitness eines Lebewesens.
- Die Kodierung des Lösungskandidaten ... Der Genotyp eines Lebewesens.
- Kodierung einer Lösungskandidaten als Datenstruktur, z.B. als Bitstring ... Kodierung des Genotyps durch Gene, also durch Nukleotidsequenzen auf den Chromosomen.

Genetische Algorithmen

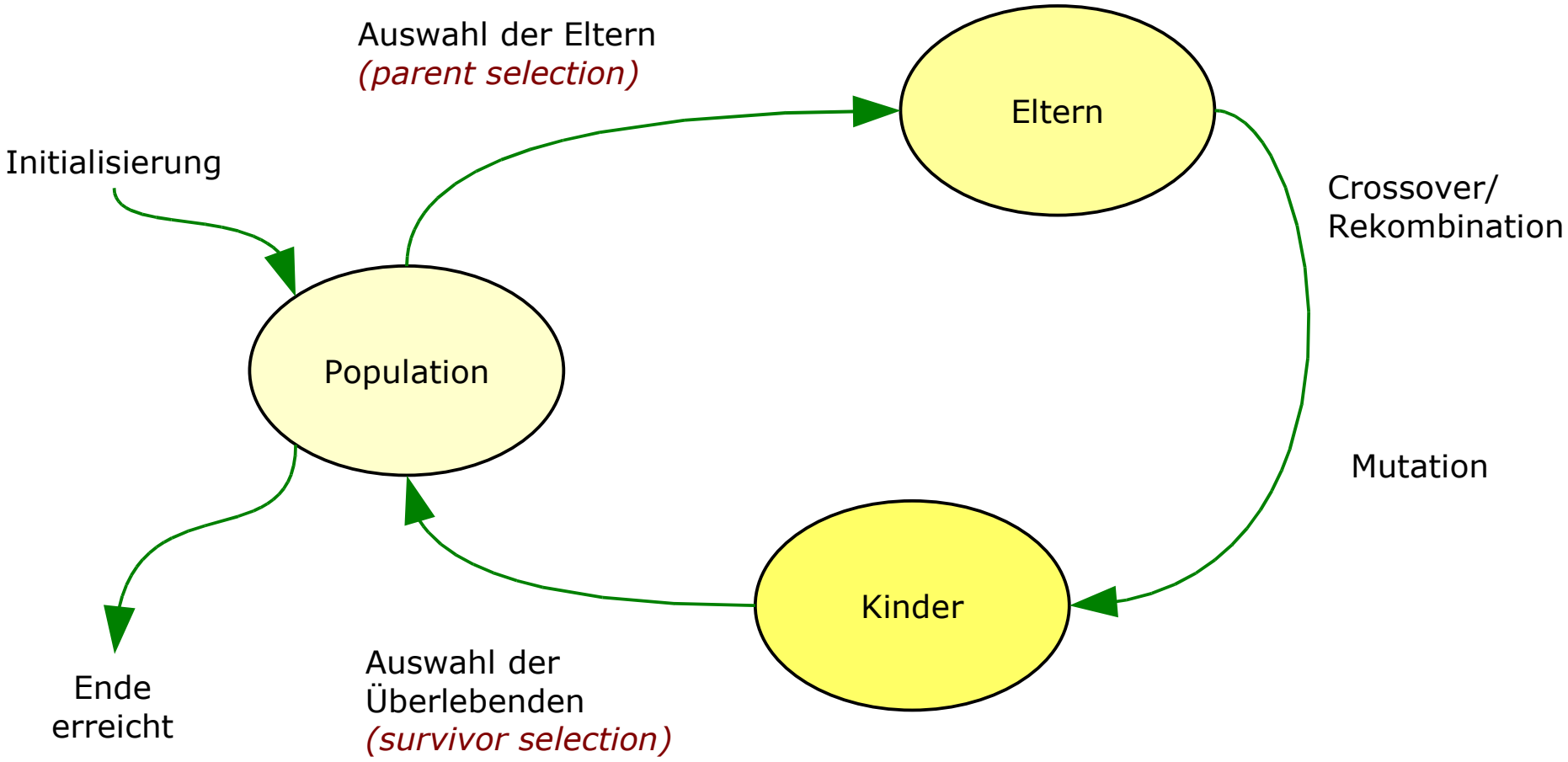
Inspiration

Lösung von Informatikproblemen durch Aufgreifen von Analogien aus der Biologie:

- Aus einer Anfangslösung wird eine bessere gefunden ... Ein Lebewesen hat Nachkommen mit einer höheren Fitness.
- Veränderung der Kodierung eines Lösungskandidaten ... Änderungen im Erbgut eines Lebewesen durch Mutation.
- Mischen von Eigenschaften zweier Lösungskandidaten ... Rekombination bzw. Crossover.

Genetische Algorithmen

Der grundsätzliche Ablauf



Genetische Algorithmen

Struktur eines genetischen Algorithmus

Aus diesem Ablauf ergibt sich eine mögliche Grundstruktur eines genetischen Algorithmus:

```
public void einfacherGenetischerAlgorithmus() {
    int generation = 0;
    Population p = erzeugeZufälligeAnfangspopulation();
    Fitness f = p.evaluation();
    while (f.istNochNichtAusreichend() &&
           generation < maxGeneration) {
        generation++;
        p = p.selection(); // age biased replacement
        p.crossover();
        p.mutation();
        f = p.evaluation();
    }
}
```

Genetische Algorithmen

Suchraum

Beispiel: Gegeben sei folgende Funktion:

$$f(x) = 150 + 42 * x * \sin(20 * x) * \cos(2 * x)$$

Das Optimum dieser Funktion im Bereich $x \in [2 \ldots 4]$ soll mit einer Genauigkeit von 5 Nachkommastellen gefunden werden.

Der *Suchraum (search space)* ist die Menge aller *Lösungskandidaten (solution candidates)* für ein gegebenes Problem.

In unserem Beispiel ist der Suchraum also das Zahlenintervall [2 ... 4].

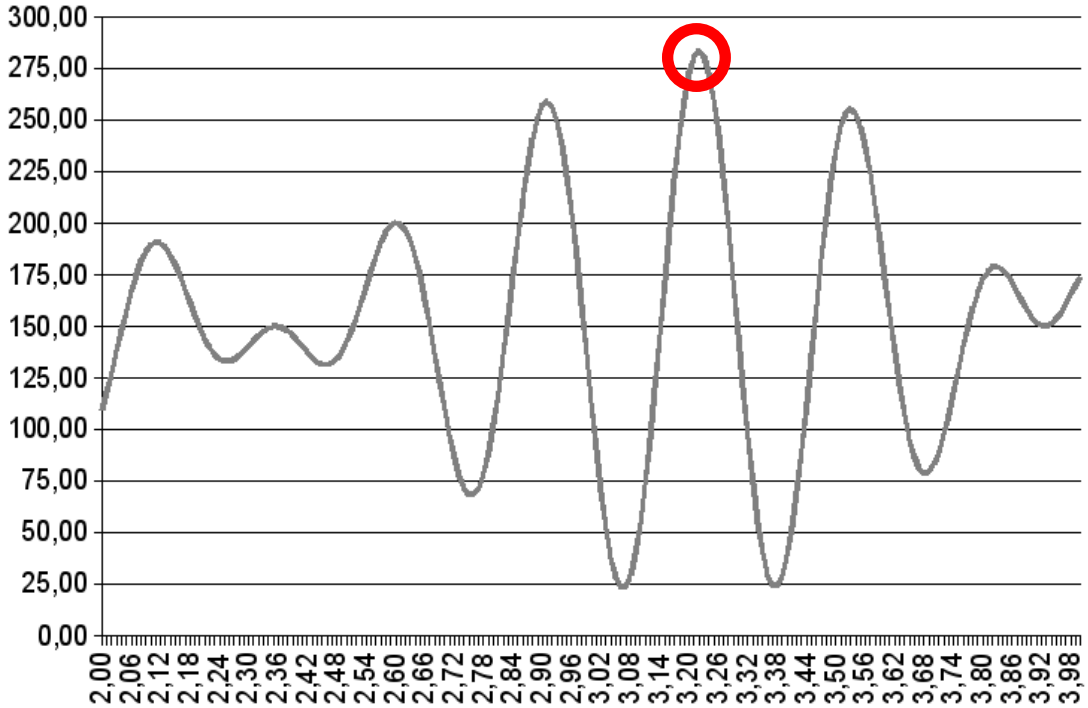
Xmin = 2

Xmax = 4

Genetische Algorithmen

Fitnessfunktion

Die *Fitnessfunktion (fitness function, fitness landscape)* ist ein Begriff welcher den Raum aller möglichen Genotypen mit ihrer Fitness bezeichnet.



*Die Fitnessfunktion in unserem Beispiel ist die Funktion f(x) = 150 + 42 * x * sin(20 *x) * cos(2*x) mit einer unabhängigen Variable.*
Sie hat in unserem Suchraum mehrere lokale Maxima und Minima, sowie ein globales Maximum.

Die Fitness zu dem Lösungskandidaten 3.22012 beträgt 283.58047

Genetische Algorithmen

Lösungskandidaten

Lösungskandidaten (solution candidates) sind Elemente aus dem Suchraum, in unserem Beispiel also reelle Zahlen. Die Lösung soll auf fünf Nachkommastellen genau sein. Wie gross ist der Suchraum ?

digits = 5

Xmin = 2

Xmax = 4

*Wertebereich = (Xmax – Xmin) * 10 ^ digits = 200000*

Es gibt also 200000 verschiedene Zahlen, oder Phänotypen in diesem Suchraum.

2.00000

2.00001

2.00002

...

4.00000

Genetische Algorithmen

Kodierung der Lösungskandidaten

Jeder Lösungskandidat wird als ein Bitstring kodiert.

Wie viele Bits benötigt man ?

bits = log (200000) / log (2) = 17.609

Diese Zahl wird aufgerundet auf 18 Bit. Beachten Sie dass mit 18 Bit tatsächlich

2 ^ 18 = 262144

Werte dargestellt werden, es ergibt sich somit eine etwas feinere Auflösung als für dieses Beispiel unbedingt notwendig wäre.

Genetische Algorithmen

Kodierung der Lösungskandidaten

Somit erhält man folgende Zuordnung:

Genotyp	***Genotyp (dezimal)***	***Phänotyp***
0000000000000000	*0*	*2.00000 = Xmin*
0000000000000001	*1*	*2.00001*
0000000000000010	*2*	*2.00002*
...		
0111111111111110	*131070*	*2.99999*
0111111111111111	*131071*	*3.00000*
1000000000000000	*131072*	*3.00000*
1000000000000001	*131073*	*3.00001*
...		
1111111111111101	*262141*	*3.99998*
1111111111111110	*262142*	*3.99999*
1111111111111111	*262143*	*4.00000 = Xmax*

Genetische Algorithmen

Dekodierfunktion: Vom Genotyp zum Phänotyp

Bei einem gegebenen Genotyp lässt sich in unserem Beispiel der zugehörige Phänotyp wie folgt leicht ausrechnen:

$$Phänotyp = Xmin + Genotyp * \frac{(Xmax - Xmin)}{2^{bits} - 1}$$

- Die Funktion welche zu einem Genotyp den zugehörigen Phänotyp zurückliefert dekodiert also den Genotyp.

Allgemein wird eine solche Funktion auch *Dekodierfunktion (decode function)* genannt.

- Umgekehrt kann der Genotyp zu einem gegebenen Phänotyp wie folgt leicht ermittelt werden:

$$Genotyp = \left\lfloor \frac{Phänotyp - Xmin}{Xmax - Xmin} * (2^{bits} - 1) \right\rfloor$$

Genetische Algorithmen

Initiale Population

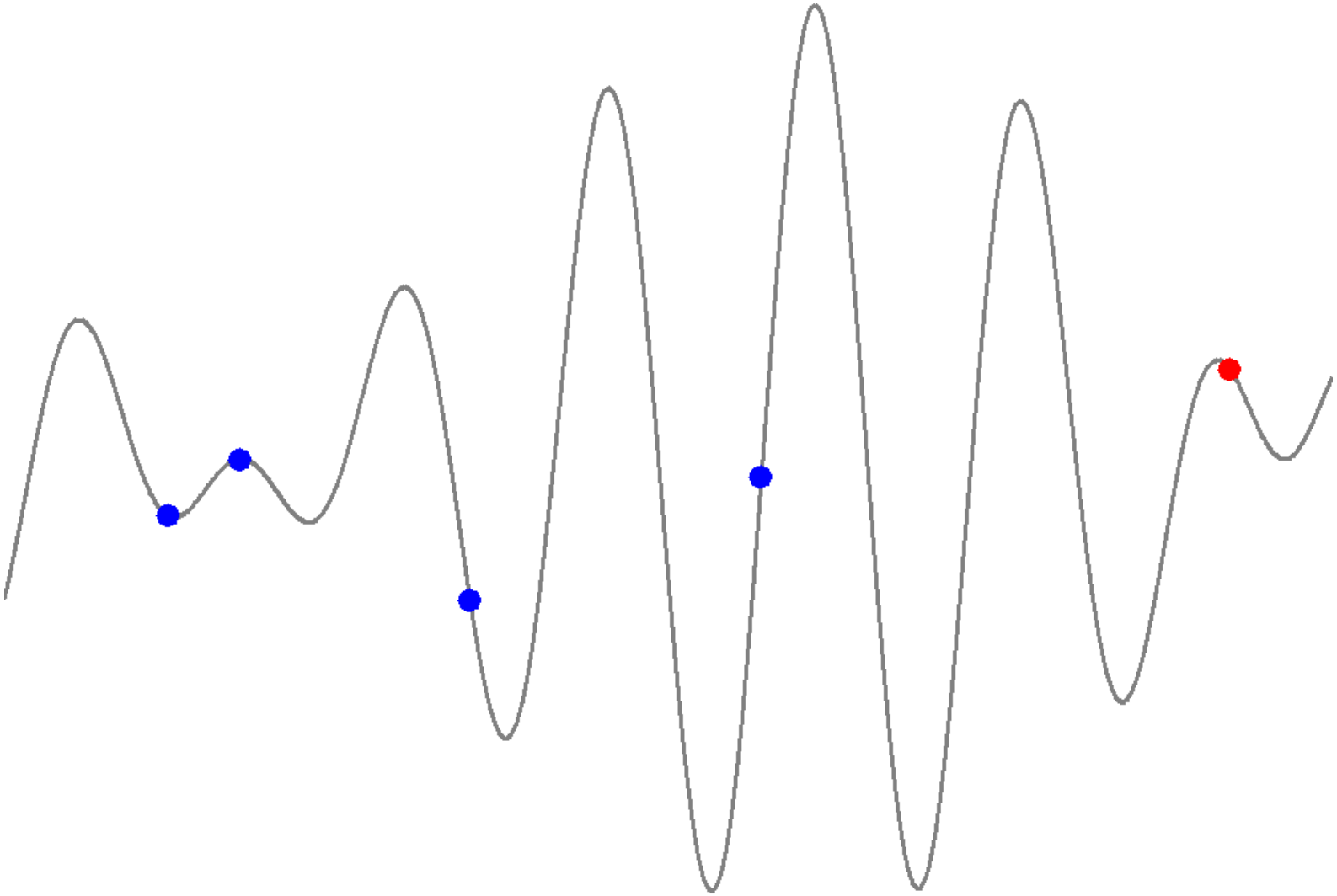
Im ersten Schritt des genetischen Algorithmus wird eine initiale Population erzeugt.

Vereinfachend betrachten wir in unserem Beispiel eine Population der Größe fünf, die durch Zufallszahlen ermittelt wird.

Nr.	Genotyp	dezimal	Phänotyp	Fitness	
0	1001000111110000000	149376	3.13965	144.88529	
1	0001111111011011110	32478	2.24779	133.20796	
2	0101100111011101000	91880	2.70099	108.49095	
3	1110110000001000001	241729	3.84425	176.50203	<-- peak
4	0010110110000000011	46595	2.35549	149.99805	

Genetische Algorithmen

Initiale Population



Genetische Algorithmen

Selektion

Als nächster Schritt werden nun Lösungskandidaten aus der aktuellen Population selektiert um die Population für die nächste Generation zu bestimmen.

- In diesem Beispiel nehme wir vereinfachend an, dass die Grösse der Population stets konstant bleibt.
- Die Kandidaten sollen proportional zu ihrer Fitness ausgewählt werden *(fitness proportional selection)*.
- Der gleiche Kandidat kann gegebenenfalls mehrmals selektiert werden.
- Der beste Kandidat muss nicht zwingend in die nächste Generation weitergegeben werden.

Genetische Algorithmen

Selektion

Um eine Selektion die proportional zu der Fitness der Kandidaten ist zu verstehen, benötigen wir den Begriff der durchschnittlichen Fitness.

Die *durchschnittliche Fitness einer Population (average fitness)* ist ihre *Gesamtfitness (total fitness)* geteilt durch ihre Grösse.

> *Durchschnittliche Fitness = Gesamtfitness / Populationsgrösse*
> *= (144.88529 + 133.20796 + 108.49095 + 176.50203 + 149.99805) / 5*
> *= 713.08428 / 5 = 142.61686*

Survival of the fittest: Es ist nun wünschenswert, dass:

- Kandidaten, deren individuelle Fitness *oberhalb* der durchschnittlichen Fitness der Population liegt, *überproportional* häufig selektiert werden.
- Kandidaten, deren individuelle Fitness *unterhalb* der durchschnittlichen Fitness der Population liegt, *unterproportional* häufig selektiert werden.

Genetische Algorithmen

Selektion

Die Wahrscheinlicheit selektiert zu werden, soll also dem proportionalen Anteil der individuellen Fitness an der Gesamtfitness der Population entsprechen.

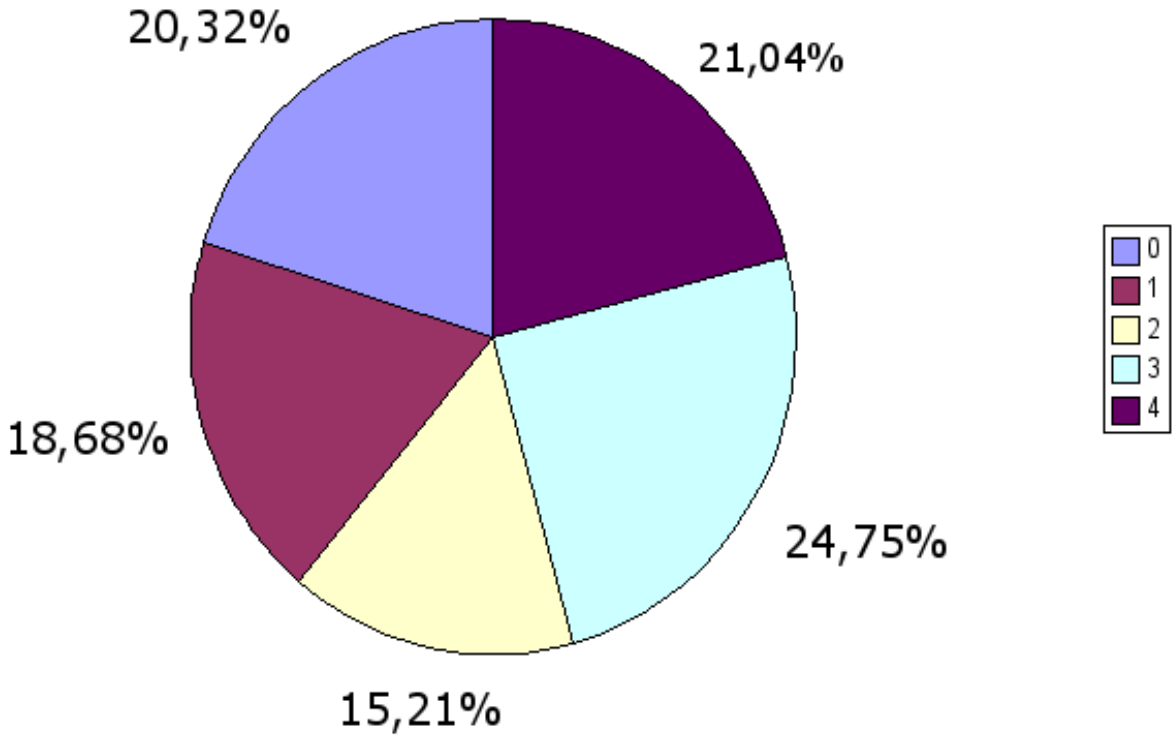
Nr.	***Fitness / Gesamtfitness***	***%***
0	*144.88529007781636 / 713.08428*	*20.32*
1	*133.20795740641046 / 713.08428*	*18.68*
2	*108.49095104736568 / 713.08428*	*15.21*
3	*176.5020339789287 / 713.084285*	*24.75*
4	*149.99805272631176 / 713.08428*	*21.04*

Genetische Algorithmen

Fitness-proportionale Selektion

Bei der *fitness-proportionalen Selektion* wird konzeptuell ein *Glücksrad (roulette wheel)* konstruiert, dessen Segmentanzahl der Populationsgrösse entspricht.

Im Gegensatz zu einem normalen Glücksrad sind die Segmente aber unterschiedlich breit und entsprechen dem proportionalen Anteil der Fitness des jeweiligen Kandidaten zur Gesamtfitness.



Genetische Algorithmen

Fitness-proportionale Selektion

Dieses Glücksrad wird nun gedreht um die diejenigen Kandidaten in der aktuellen Population auszuwählen, die in die nächste Generation übernommen werden sollen.

Wir erhalten in unserem Beispiel folgende Ergebnisse:

Nr.	***Auswahl***	***Genotyp***	***dezimal***
0	*0*	*1001000111110000000*	*149376*
1	*3*	*1110111000001000001*	*241729*
2	*3*	*1110111000001000001*	*241729*
3	*4*	*0010111011000000011*	*46595*
4	*3*	*1110111000001000001*	*241729*

In diesem Beispiel wurde der beste Kandidat dreimal ausgewählt, der schlechteste Kandidat wurde gar nicht ausgewählt.

Genetische Algorithmen

Mutation

Als nächstes werden in unserem Beispiel die ausgewählten Kandidaten mutiert.

- Die Mutationsrate wird meistens als Prozentzahl festgelegt. In unserem Beispiel nehmen wir eine Mutationsrate von 1% an.
- Mutationen finden am Genotyp, also auf dem Bitstring statt.
- Vereinfachend betrachten wir in diesem Beispiel nur Punktmutationen. Eine Punktmutation invertiert hier jeweils ein einzelnes Bit.

Genetische Algorithmen

Mutation

In diesem Beispiel finden zwei Mutationen statt.

- Mutation von Kandidat Nr. 0 an der Bitposition 0:

	Genotyp	***dezimal***
Vorher	*1001000111100000**0***	*149376*
Nachher	*1001000111100000**1***	*149377*

- Mutation von Kandidat Nr. 2 an der Bitposition 15:

	Genotyp	***dezimal***
Vorher	*11**1**011000001000001*	*241729*
Nachher	*11**0**011000001000001*	*208961*

- Beachten Sie, dass die Auswirkung einer Mutation in diesem Beispiel von der Stelle die mutiert wird abhängig ist, und von gering bis stark schwanken kann.

Genetische Algorithmen

Die zweite Generation

In unserem Beispiel erhalten wir nun folgende zweite Generation:

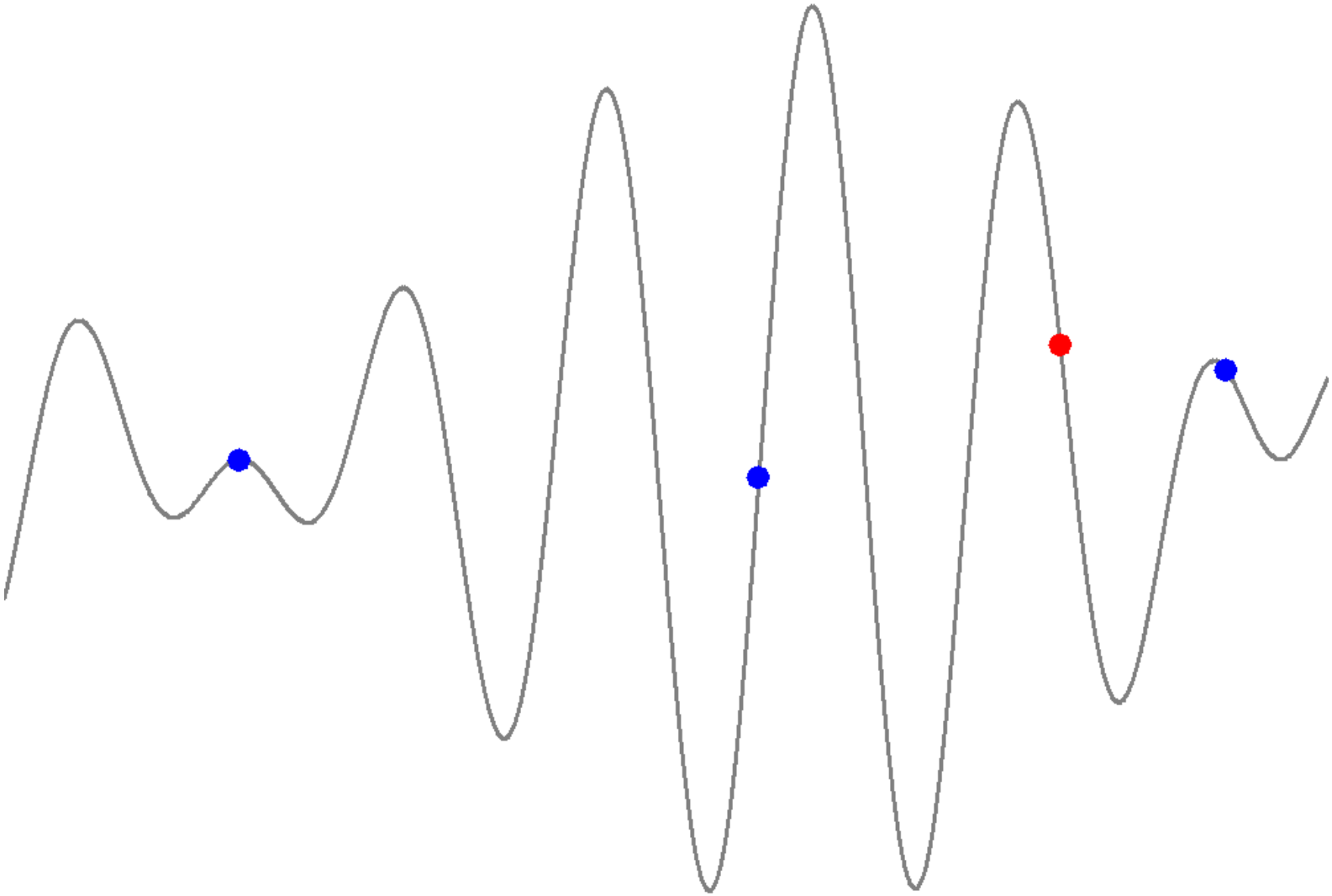
Nr.	Genotyp	dezimal	Phänotyp	Fitness	
0	*1001000111110000001*	*149377*	*3.13966*	*144.90538*	
1	*1110110000001000001*	*241729*	*3.84425*	*176.50203*	
2	*1100110000001000001*	*208961*	*3.59425*	*183.84336*	*<-- peak*
3	*0010110110000000011*	*46595*	*2.35549*	*149.99805*	
4	*1110110000001000001*	*241729*	*3.84425*	*176.50203*	

Vergleich mit der initialen Population:

- Die durchschnittliche Fitness hat sich von 142.61686 auf 166.35017 verbessert.
- Die Fitness des besten Kandidaten hat sich von 176.50203 auf 183.84336 verbessert.
- Die Diversität *(diversity)* in dem Genpool hat sich in der zweiten Population verkleinert, denn die Lösungskandidaten Nr. 1 und Nr. 4 sind vom Genotyp her identisch.

Genetische Algorithmen

Die zweite Generation

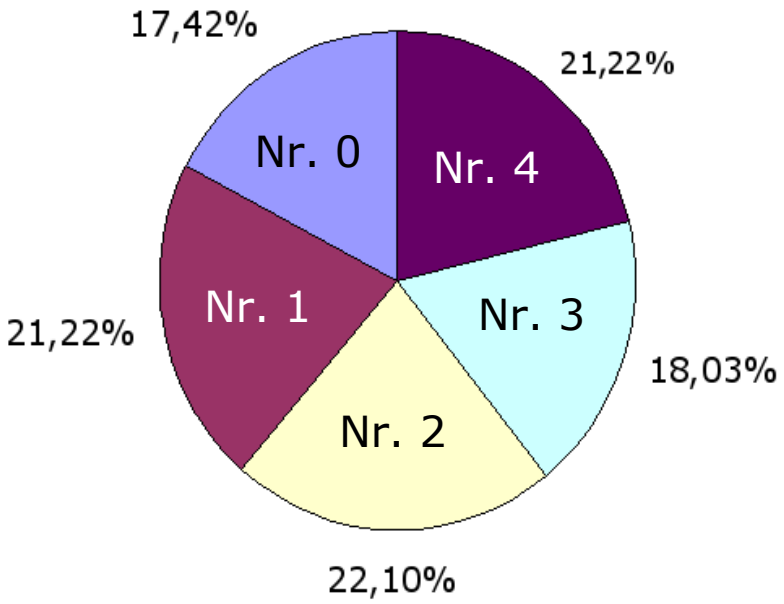


Genetische Algorithmen

Fitness

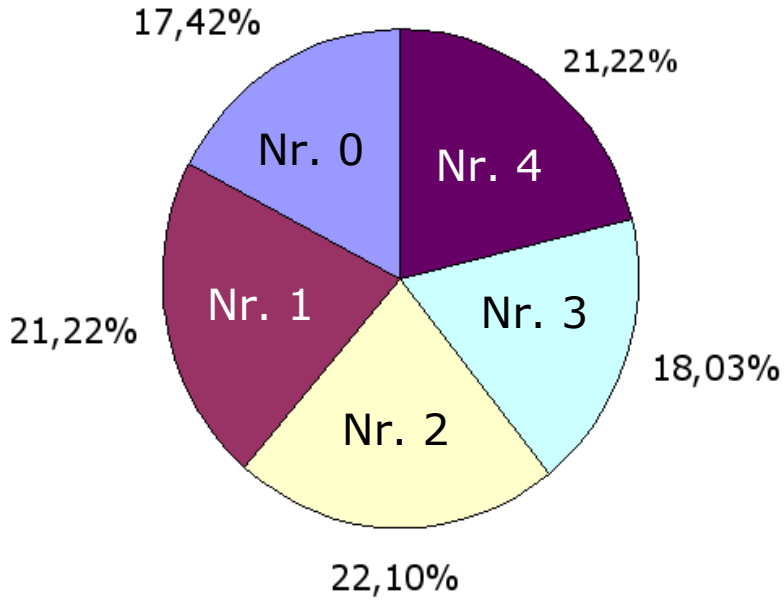
In der zweiten Generation ergibt sich in unserem Beispiel folgende Fitness:

Nr.	*Fitness / Gesamtfitness*	*%*
0	*144.90538 / 831.75086*	*17.42*
1	*176.50203 / 831.75086*	*21.22*
2	*183.84336 / 831.75086*	*22.10*
3	*149.99805 / 831.75086*	*18.03*
4	*176.50203 / 831.75086*	*21.22*



Genetische Algorithmen

Selektion



Die Selektion ergibt folgendes Ergebnis:

Nr.	*Auswahl*	*Genotyp*	*dezimal*
0	*1*	*1110110000010000001*	*241729*
1	*2*	*1100110000010000001*	*208961*
2	*0*	*1001000111100000001*	*149377*
3	*0*	*1001000111100000001*	*149377*
4	*3*	*0010110110000000011*	*46595*

Beachten Sie dass in dieser Runde der Kandidat mit der schlechtesten Fitness, Nr. 0, gleich zweimal selektiert wurde.

Genetische Algorithmen

Crossover

In diesem Beispiel findet in der zweiten Generation auch ein Crossover statt.

- Wie häufig findet Crossover statt ? Dies wird durch eine Rate festgelegt. In unserem Beispiel beträgt sie 25%.
- Vereinfachend betrachten wir hier ein Crossover, wo das Chromosom nur an einer einzigen Stelle, die durch Zufall ermittelt wird, aufgeschnitten wird *(one point crossover)*.
- Im Gegensatz zur Mutation verändert ein Crossover beide beteiligten Kandidaten.

Genetische Algorithmen

Crossover

Der Zufall ergibt, dass zwischen den Kandidaten Nr. 2 und Nr. 0 ein Crossover an der Bitposition 14 stattfindet:

Vorher

Nr.	*2*	*0*
Genotyp	*1001000111110000001*	*1110111000001000001*
dezimal	*149377*	*241729*
„Links"	*1001000000000000000*	*1110000000000000000*
„Rechts"	*0000000111110000001*	*0000111000001000001*

Nachher

Genotyp	*1110000111110000001*	*1001111000001000001*
dezimal	*231297*	*159809*

Genetische Algorithmen

Crossover

Nach dem Crossover sieht die neue Population wie folgt aus:

Nr.	*Genotyp*	*dezimal*
0	*1001110000001000001*	*159809*
1	*1100110000001000001*	*208961*
2	*1110000111110000001*	*231297*
3	*1001000111110000001*	*149377*
4	*0010110110000000011*	*46595*

Beachten Sie, dass durch den Crossover der Genpool durchgemischt wurde. Die Population besteht jetzt wieder aus fünf verschiedenen Genotypen.

Genetische Algorithmen

Mutation

In diesem Beispiel findet in der zweiten Generation lediglich eine Mutationen statt, nämlich von Kandidat Nr. 2, der ja schon durch den Crossover verändert wurde.

Mutation von Kandidat Nr. 2 an der Bitposition 15

	Genotyp	***dezimal***
Vorher	*1110000111110000001*	*231297*
Nachher	*1100000111110000001*	*198529*

Genetische Algorithmen

Die dritte Generation

In unserem Beispiel erhalten wir nun folgende dritte Generation:

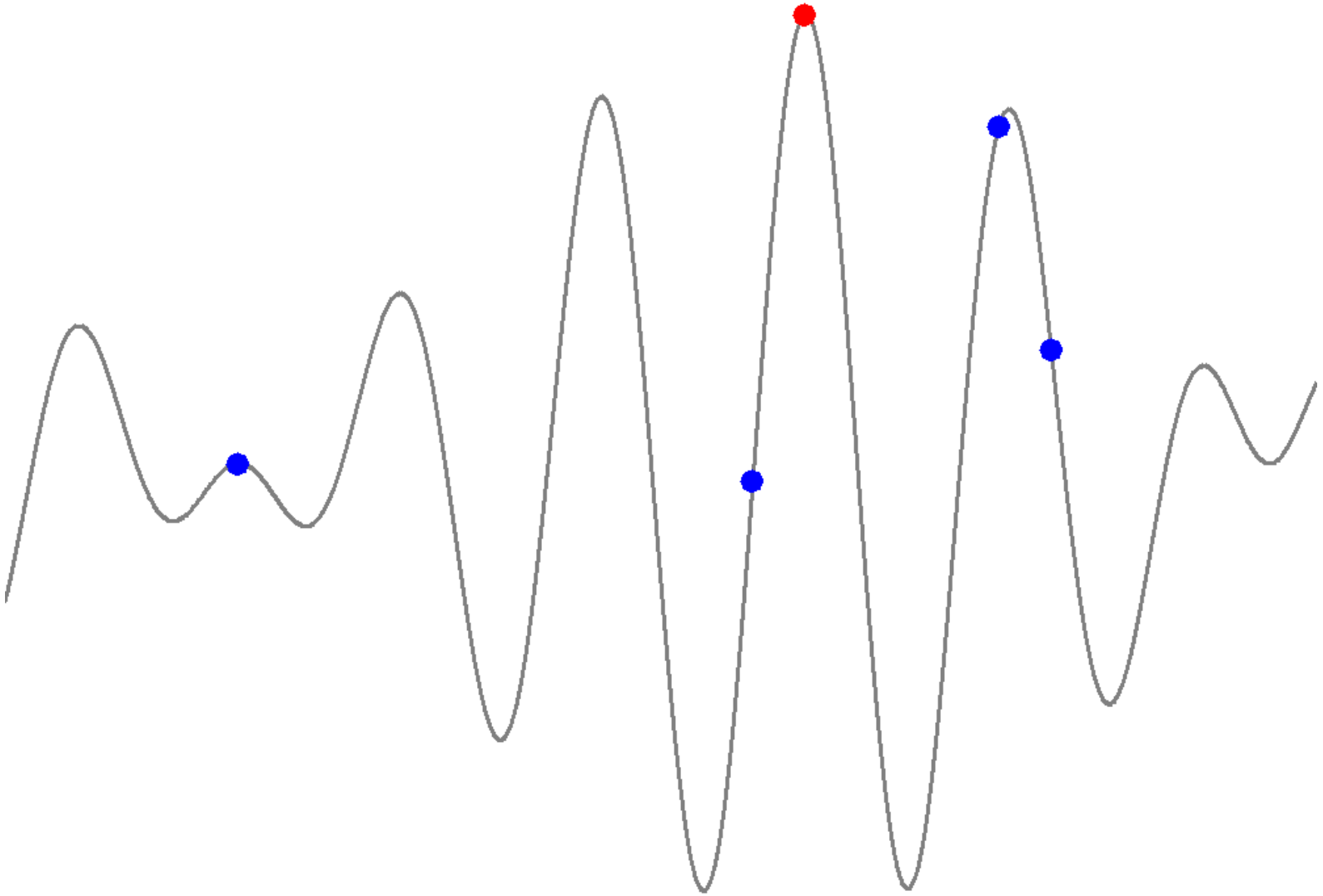
Nr.	Genotyp	dezimal	Phänotyp	Fitness	
0	*1001110000010000001*	*159809*	*3.21925*	*283.56020*	*<-- peak*
1	*1100110000010000001*	*208961*	*3.59425*	*183.84336*	
2	*1100000111100000001*	*198529*	*3.51466*	*250.14959*	
3	*1001000111100000001*	*149377*	*3.13966*	*144.90538*	
4	*0010110110000000011*	*46595*	*2.35549*	*149.99805*	

Vergleich mit der zweiten Generation:

- Die durchschnittliche Fitness hat sich nochmals verbessert, hier von 166.35017 auf 202.49132.
- Die Fitness des besten Kandidaten hat sich ebenfalls nochmal von 183.84336 auf 283.56020 verbessert.
- Die Gesamtfitness der Population beträgt jetzt 1012.45659

Genetische Algorithmen

Die dritte Generation



Genetische Algorithmen

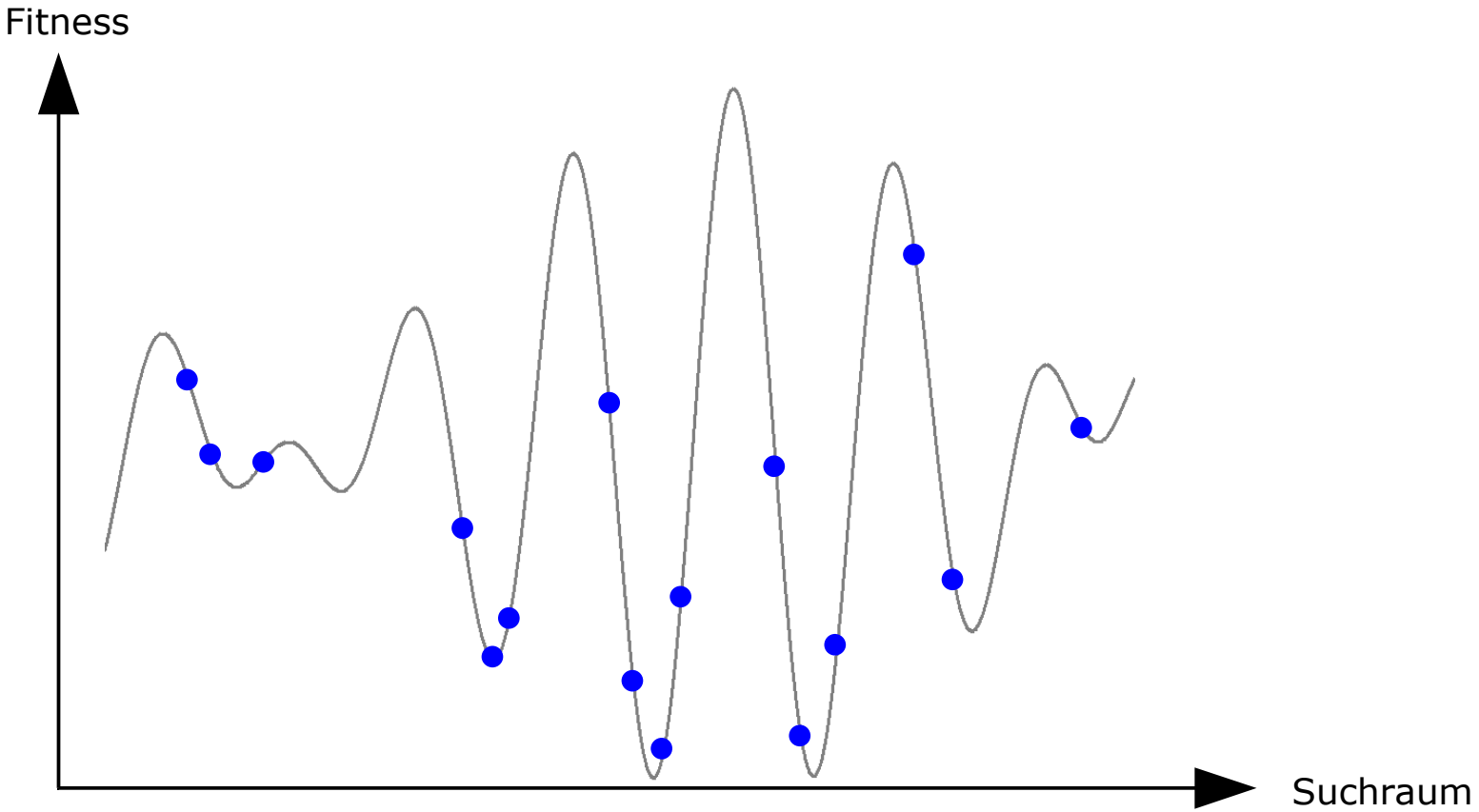
Beobachtungen

- Wir haben in drei Schritten eines genetischen Algorithmus eine gute Näherung für das Maximum einer Funktion gefunden.
- Der genetische Algorithmus operiert lediglich auf Bitstrings und hat keinerlei Kenntnis der zugrunde liegenden Funktion.
- Die Wirkung des genetischen Algorithmus beruht auf Selektion, Crossover und Mutation.

Welche weiteren allgemeinen Beobachtungen kann man machen ?

Genetische Algorithmen

Phasen eines genetischen Algorithmus - I

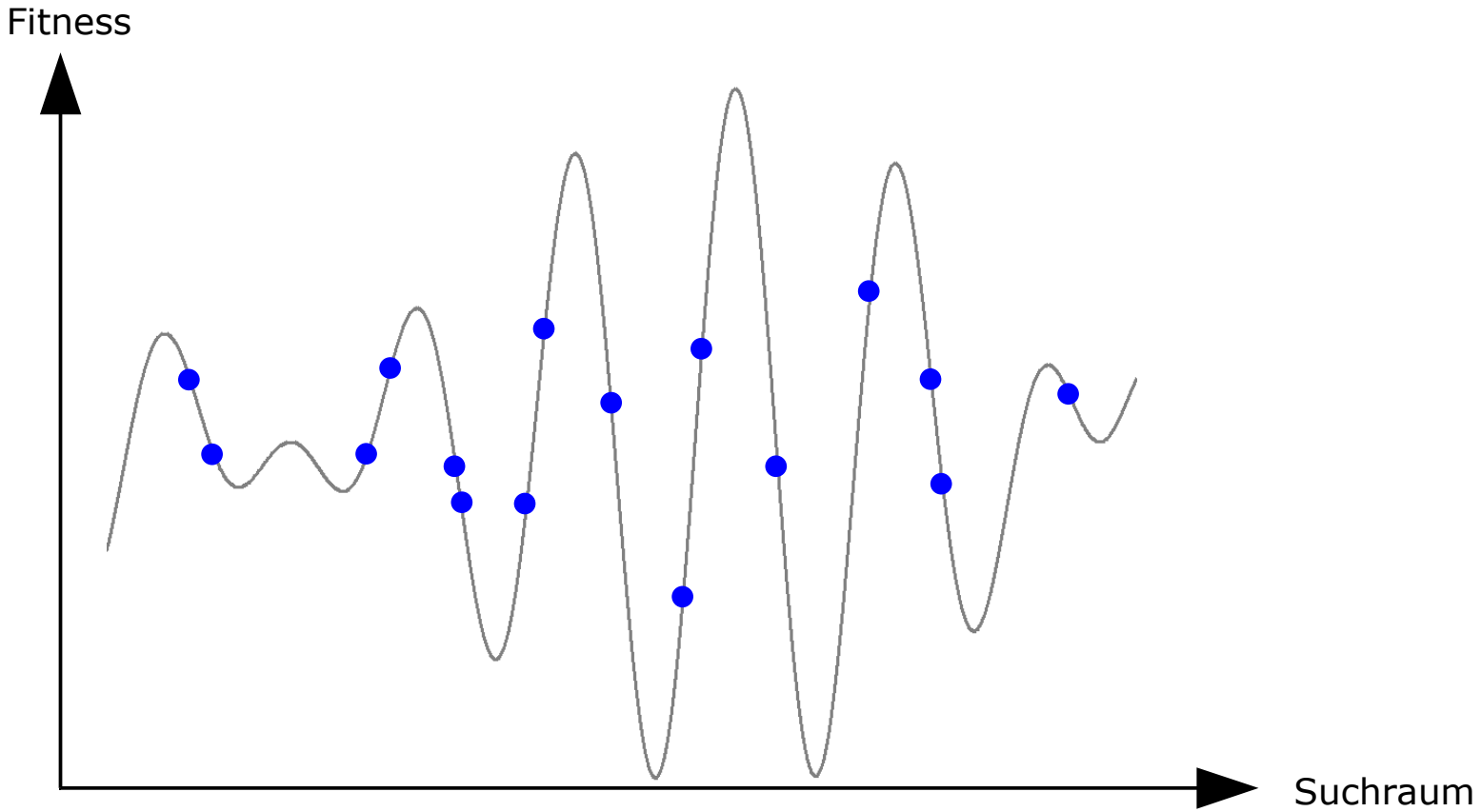


Man kann den Ablauf eines genetischen Algorithmus allgemein betrachtet in drei Phasen einteilen. Am Anfang sind die Lösungskandidaten zufällig über den ganzen Suchraum verteilt.

Warum ist das so ?

Genetische Algorithmen

Phasen eines genetischen Algorithmus - II

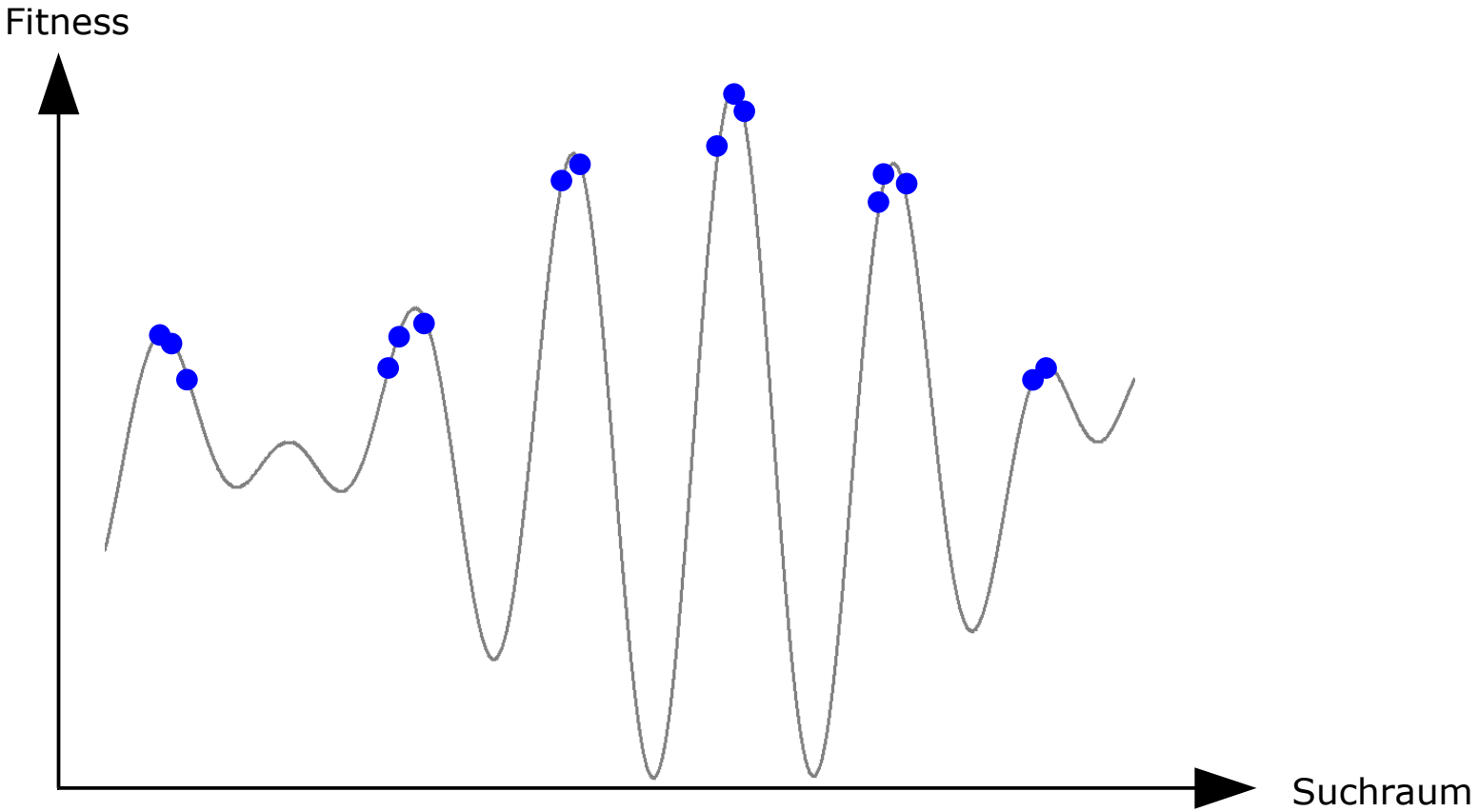


In der zweiten Phase, die bereits nach wenigen Generationen eintritt, verlässt die Population diejenigen Bereiche, welche eine niedrige Fitness aufweisen. Sie beginnt also – bildlich gesprochen - die lokalen Maxima hinaufzuklettern.

Warum ist das so ?

Genetische Algorithmen

Phasen eines genetischen Algorithmus - III

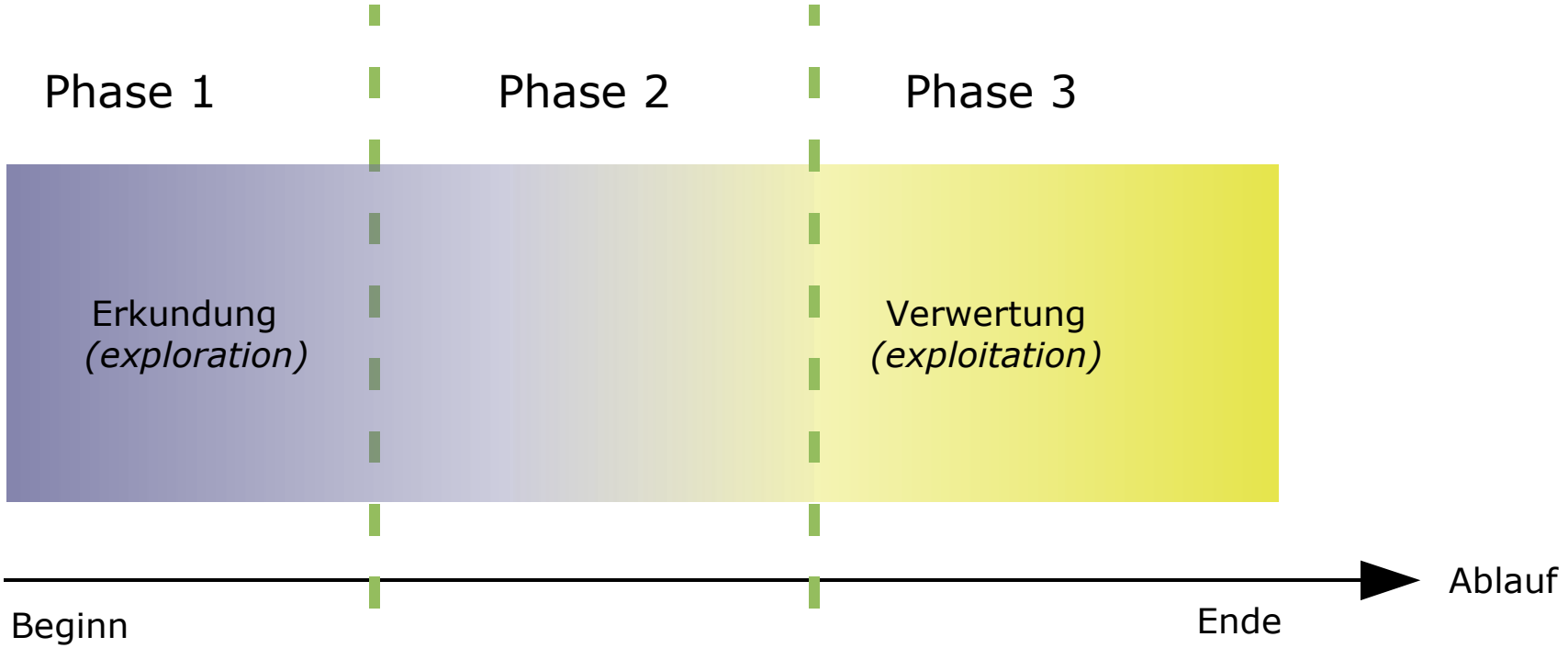


Sofern die Abbruchkriterien richtig gesetzt sind, hat sich die Population in der dritten Phase gegen Ende des Ablaufs bei den Gipfeln versammelt. Man kann dabei nicht ausschließen, dass dies nur suboptimale lokale Maxima sind.

Warum ist das so ?

Genetische Algorithmen

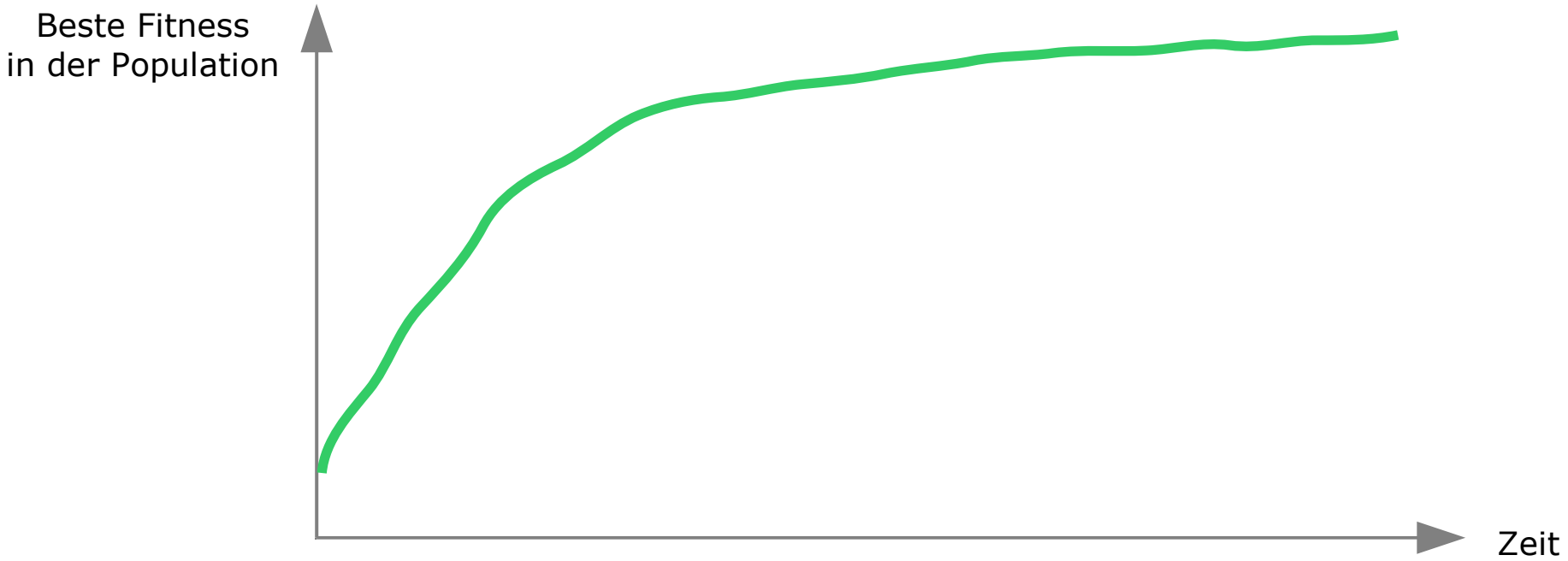
Phasen vs. Exploration / Exploitation



Am Anfang des Ablaufs steht steht eher die *Erkundung (exploration)* des Suchraums im Vordergrund. Gegen Ende des Ablaufs dagegen die *Verwertung (exploitation)* der besten bisher aufgefunden Stellen im Suchraum.

Genetische Algorithmen

Die derzeit beste Lösung

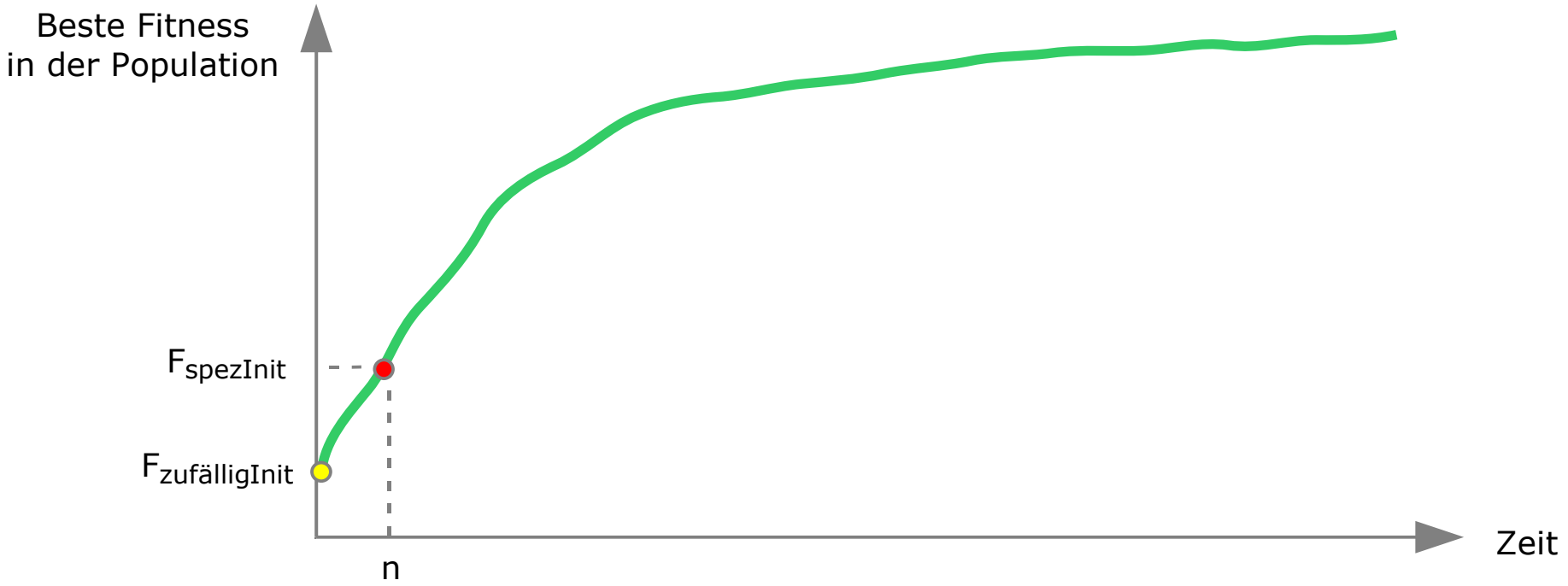


Ein genetischer Algorithmus kann jederzeit abgebrochen werden, und man hat einen Lösungskandidaten mit der besten Fitness, die bisher gefunden wurde.

Die obige Kurve ist vom Verlauf her typisch für genetische Algorithmen.

Genetische Algorithmen

Problemspezifische Initialisierung

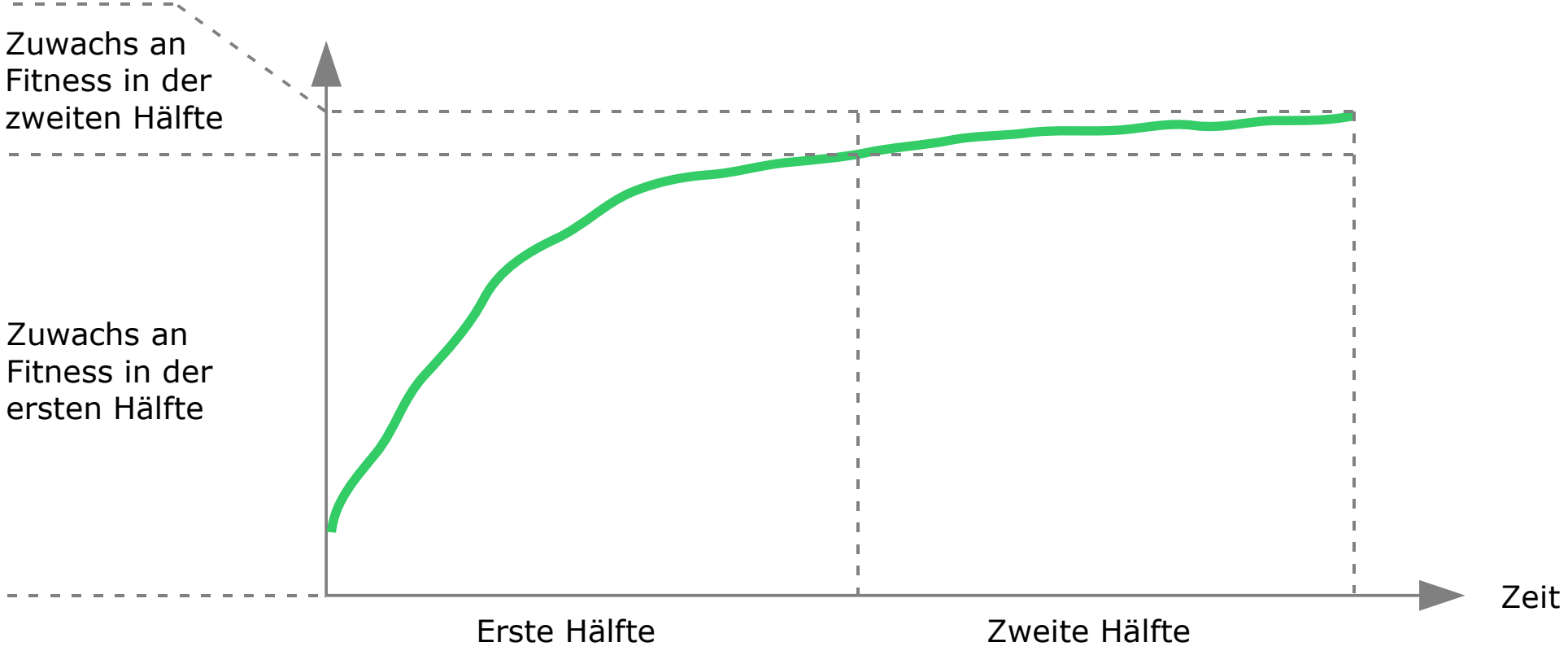


Lohnt es sich, die Anfangspopulation nicht zufällig, sondern problemspezifisch zu initialisieren, um den genetischen Algorithmus mit besonders vielversprechenden Kandidaten zu starten ?

Nein. Die dabei erreichbare Fitness wird typischerweise schon innerhalb weniger Generationen erreicht, so dass sich der zusätzliche Programmier- und Rechenaufwand nicht lohnt.

Genetische Algorithmen

Laufzeitverhalten

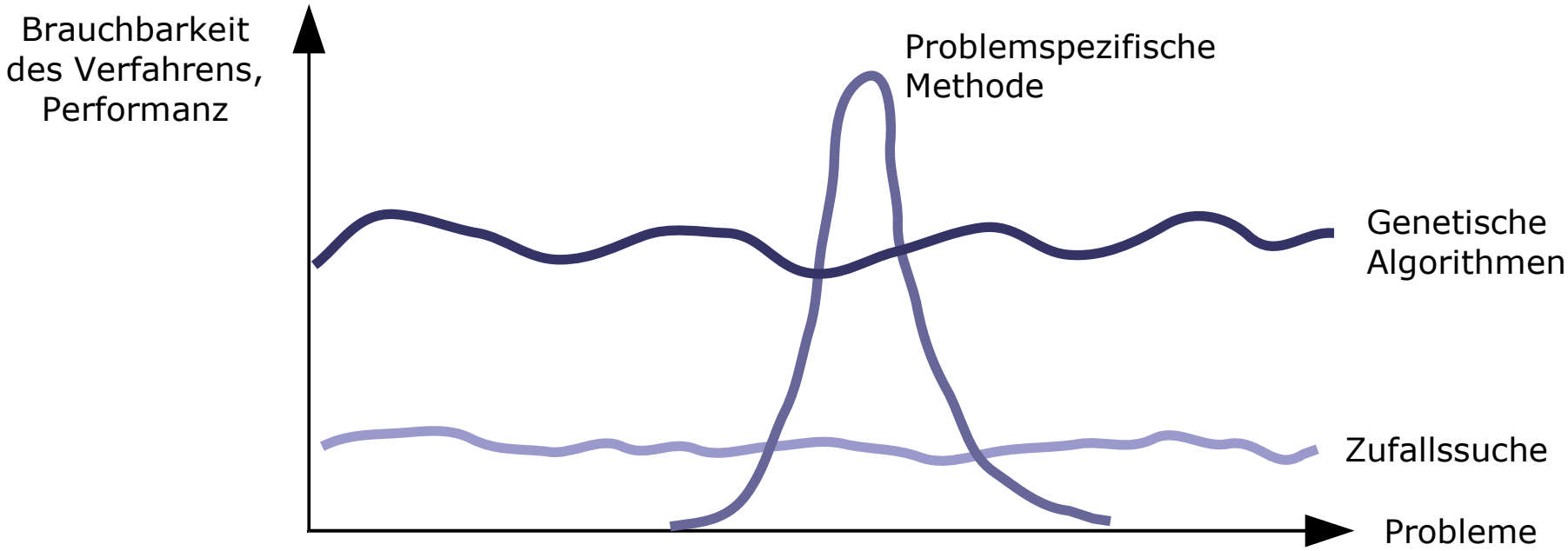


Soll man einen genetischen Algorithmus besonders lange laufen lassen, um eine gute Lösung zu finden ?

Eher nicht. Bei Verdopplung der bereitgestellten Rechenzeit erhält man typischerweise nur eine geringfügig bessere Lösung.

Genetische Algorithmen

Genetische Algorithmen als allgemeine Problemlöser

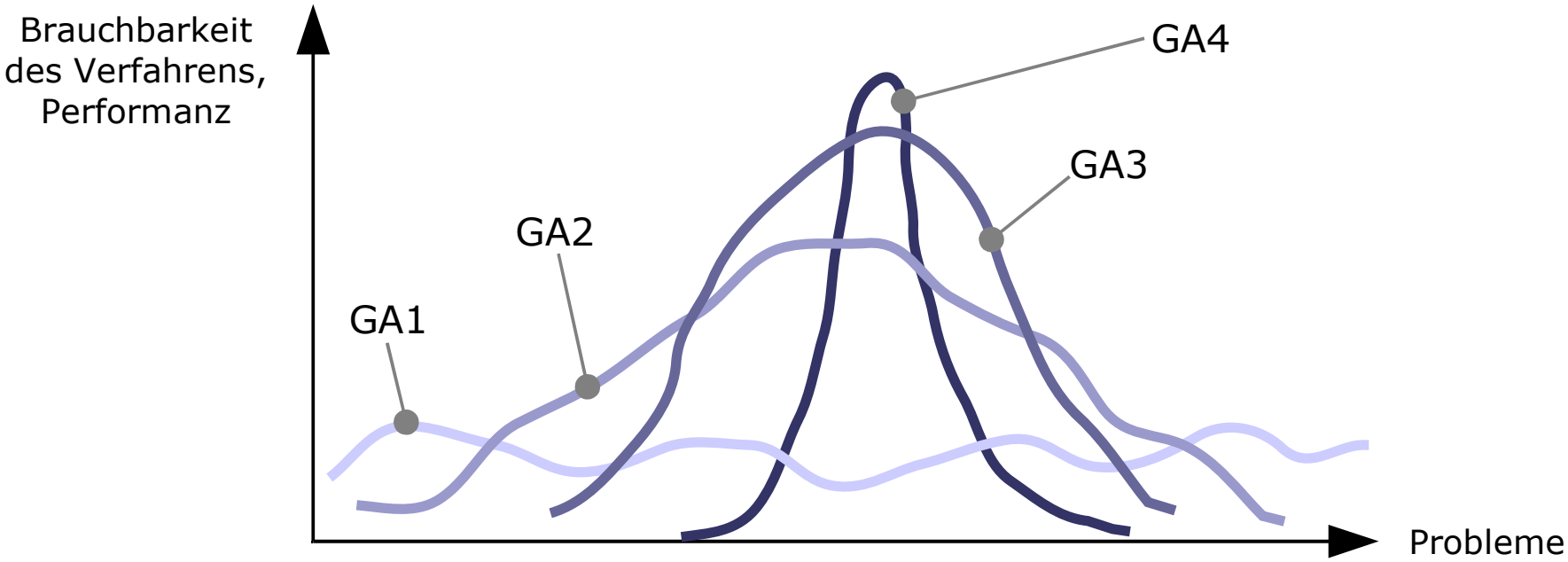


Sichtweise nach *D. Goldberg*, 1980er Jahre:

- Genetische Algorithmen benötigen kein Wissen über das zu Lösende Problem. Sie sind daher vielseitig einsetzbar.
- Im Gegensatz zu problemspezifischen Methoden gibt es bei genetischen Algorithmen keine Garantie dass die optimale Lösung gefunden wird.

Genetische Algorithmen

Spezialisierung von genetischen Algorithmen



Sichtweise nach *Z. Michalewicz*, 1990er Jahre:

- Durch unterschiedlich starke Anreicherung mit problemspezifischem Wissen wird die Brauchbarkeit von genetischen Algorithmen für spezielle Problemklassen verbessert, dadurch aber auch für alle anderen Probleme tendenziell eher verschlechtert.