Genetische Algorithmen

Caroline Sporleder

Computational Linguistics
Universität des Saarlandes

Sommersemester 2011

16.06.2011

Evolutionäre Algorithmen

sind durch die Natur inspiriert (Nachahmung von Evolution und natürlicher Selektion).

Genetische Algorithmen

- sind eine Art von Evolutionärer Algorithmus
- Lösungen für ein Problem werden iterativ verbessert durch Selektion, Veränderung und Kombination von partiell guten Lösungen

Komponenten

- **genetische Repräsentation** (Chromosom oder Individuum) möglicher Lösungen
  $\Rightarrow$ typischerweise ein Bitvektor
- eine **Fitnessfunktion** zum Evaluieren von Chromosomen/Individuen

Die Gesamtzahl der Individuen/Chromosomen in einer Iteration nennt man Population (oder Generation).

Beispiel: Max und die Vorlesung (1)

Num	Doz	The	Wo	We	Max da?
1	B	D	D	R	J
2	B	A	D	R	J

Naive Repräsentation als Bitvektor

- **jedes Bit** repräsentiert ein binäres **Attribut**,
  z.B.: *Doz*, *The*, *Wo*, *We*
- ein Wert wird durch '0' dargestellt, der andere durch '1'
  z.B.: $Doz : A \Rightarrow 0$ und $Doz : B \Rightarrow 1$

Beispiel

Beispiel: Max und die Vorlesung (1)

Num	Doz	The	Wo	We	Max da?
1	B	D	D	R	J
2	B	A	D	R	J

Naive Repräsentation als Bitvektor

- **jedes Bit** repräsentiert ein binäres **Attribut**,
  z.B.: *Doz*, *The*, *Wo*, *We*
- ein Wert wird durch '0' dargestellt, der andere durch '1'
  z.B.: $Doz : A \Rightarrow 0$ und $Doz : B \Rightarrow 1$

Beispiel

1: <1010> 2: <1110>

Nachteile

Beispiel: Max und die Vorlesung (1)

Num	Doz	The	Wo	We	Max da?
1	B	D	D	R	J
2	B	A	D	R	J

Naive Repräsentation als Bitvektor

- **jedes Bit** repräsentiert ein binäres **Attribut**,
  z.B.: *Doz*, *The*, *Wo*, *We*
- ein Wert wird durch '0' dargestellt, der andere durch '1'
  z.B.: $Doz : A \Rightarrow 0$ und $Doz : B \Rightarrow 1$

Beispiel

1: <1010> 2: <1110>

Nachteile

- nicht-binären Attributen
- disjunktive Lösungen (d.h. mehrere Werte eines Attributs sind möglich)

Beispiel: Max und die Vorlesung (1)

Num	Doz	The	Wo	We	Max da?
1	B	D	D	R	J
2	B	A	D	R	J

Komplexere Repräsentation als Bitvektor

- **jedes Bit** repräsentiert ein **Attributwertpaar** (AWP), z.B.: $Doz : B$, $The : A$, $Wo : D$, $We : S$
- eine '1' bedeutet, dass dieses AWP gesetzt sein kann
- dadurch lassen sich auch disjunktive Lösungen darstellen

Beispiel

	Doz:A	Doz:B	The:A	The:D	Wo:D	Wo:F	We:S	We:R
1:								
2:								
1 $\vee$ 2:								

Beispiel: Max und die Vorlesung (1)

Num	Doz	The	Wo	We	Max da?
1	B	D	D	R	J
2	B	A	D	R	J

Komplexere Repräsentation als Bitvektor

- **jedes Bit** repräsentiert ein **Attributwertpaar** (AWP), z.B.: $Doz : B$, $The : A$, $Wo : D$, $We : S$
- eine '1' bedeutet, dass dieses AWP gesetzt sein kann
- dadurch lassen sich auch disjunktive Lösungen darstellen

Beispiel

	Doz:A	Doz:B	The:A	The:D	Wo:D	Wo:F	We:S	We:R
1:	< 0	1	0	1	1	0	0	$1 >$
2:	< 0	1	1	0	1	0	0	$1 >$
$1 \vee 2$:	< 0	1	1	1	1	0	0	$1 >$

Beispiel: Max und die Vorlesung (2)

Fitnessfunktion

Beispiel: Max und die Vorlesung (2)

Fitnessfunktion

- z.B. Accuracy auf dem Development-Set
- oder Accuracy plus Kostenfunktion
- oder F-Score
- etc.

Elementarer Algorithmus

1. **Initialisiere** Population mit m Individuen
2. solange noch nicht **Terminiere**=true
   1. **Evaluiere** jedes Individuum
   2. **Selektiere** n Individuen für die neue Generation ($n < m$)
   3. **Generiere** $m - n$ neue Individuen aus den n Elternindividuen
3. Gebe das Individuum mit der höchsten Fitness zurück

Initialisierung

- zufällige Auswahl der Anfangspopulation (ggf. aus Trainingsinstanzen)
- möglichst diverse Population

Terminierung

Mögliche Abbruchkriterien

- das beste Individuum (d.h. die beste Lösung) erreicht eine bestimmte Performanz
- eine vorher festgelegte Anzahl von Generationen wurde durchlaufen
- ein Plateau wurde erreicht (in den letzten k Generationen wurde keine oder nur noch eine minimale Performanzverbesserung erreicht).

Evaluierung

Evaluierung anhand der Fitnessfunktion

Selektion

Naive Selektion

wähle die n fittesten Individuen

Roulette Wheel Selection

ein 'fitteres' Individuen, h_i, wird mit einer höheren Wahrscheinlichkeit ausgewählt, z.B.:

$$P(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{m} Fitness(h_j)}$$

Tournament Selection

Zwei Individuen, h_1 und h_2, werden zufällig ausgewählt (mit uniformer Wahrscheinlichkeit). Mit einer Wahrscheinlichkeit p ($p > 0.5$) wird das fittere von beiden für die nächste Generation ausgewählt.

Generierung

Generierung neuer Individuen erfolgt durch sogenannte genetische Operatoren (engl. genetic operators):

Crossover

Aus zwei Elternindividuen werden zwei neue Individuen generiert, die die Eigenschaften der Eltern unterschiedlich kombinieren.

- single point crossover
- two-point crossover
- uniform crossover

Mutation

Einzelne Bits im Bitvektor eines Individuums werden zufällig verändert.

Single Point Crossover

Eine Position im Bitvektor wird zufällig ausgewählt. An dieser Stelle werden Kopien der Elternvektoren geteilt und vertauscht und dadurch die Kinder generiert.

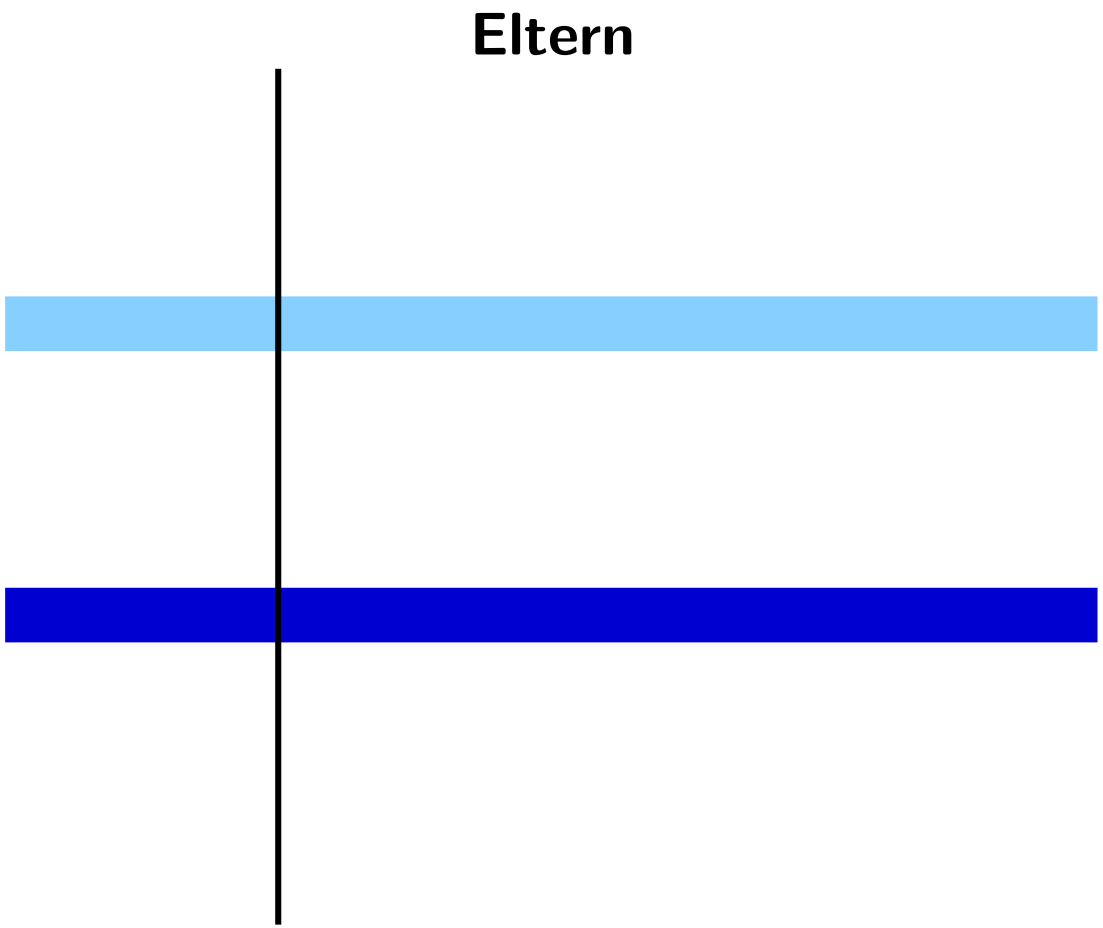
Single Point Crossover

Eine Position im Bitvektor wird zufällig ausgewählt. An dieser Stelle werden Kopien der Elternvektoren geteilt und vertauscht und dadurch die Kinder generiert.

Eltern

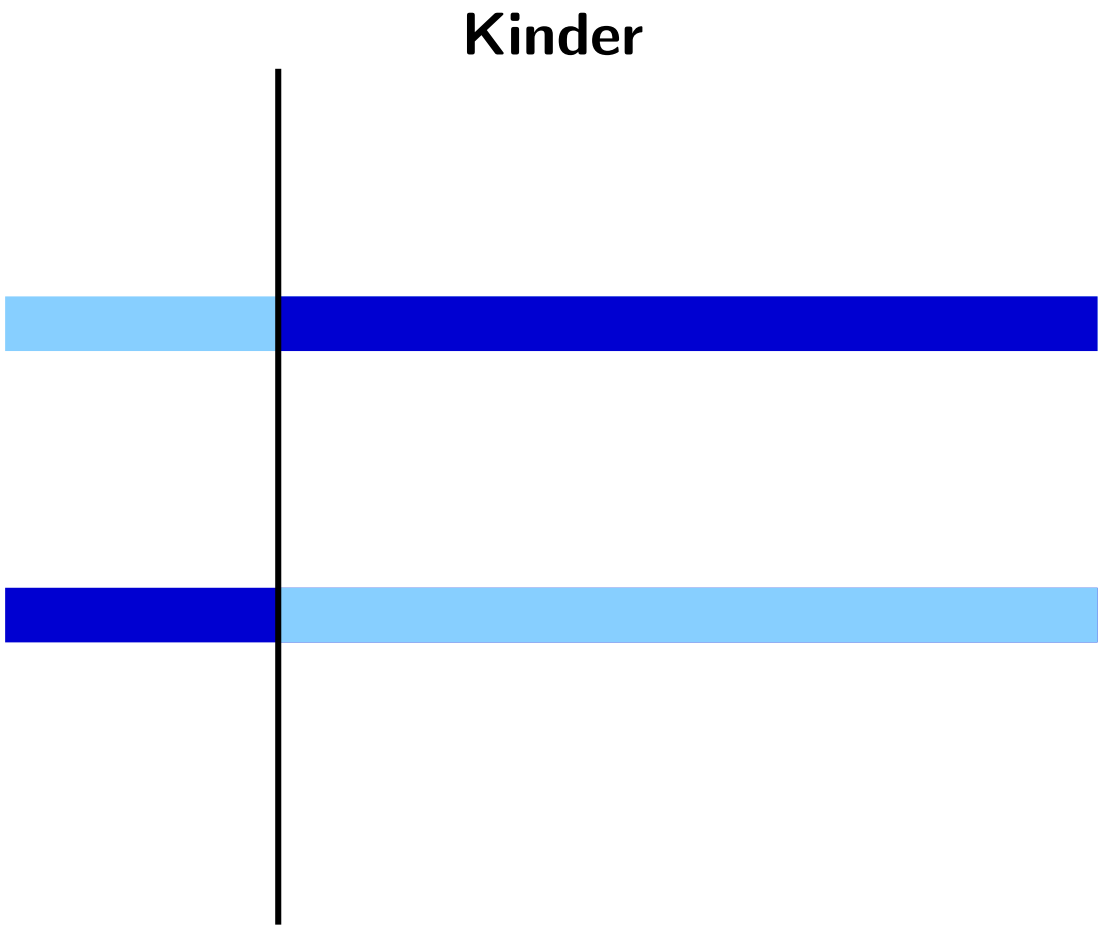
Single Point Crossover

Eine Position im Bitvektor wird zufällig ausgewählt. An dieser Stelle werden Kopien der Elternvektoren geteilt und vertauscht und dadurch die Kinder generiert.



Single Point Crossover

Eine Position im Bitvektor wird zufällig ausgewählt. An dieser Stelle werden Kopien der Elternvektoren geteilt und vertauscht und dadurch die Kinder generiert.



Two Point Crossover

Zwei Positionen im Bitvektor werden zufällig ausgewählt. Zwischen diesn beiden Stellen werden Kopien der Elternvektoren miteinander vertauscht und dadurch die Kinder generiert.

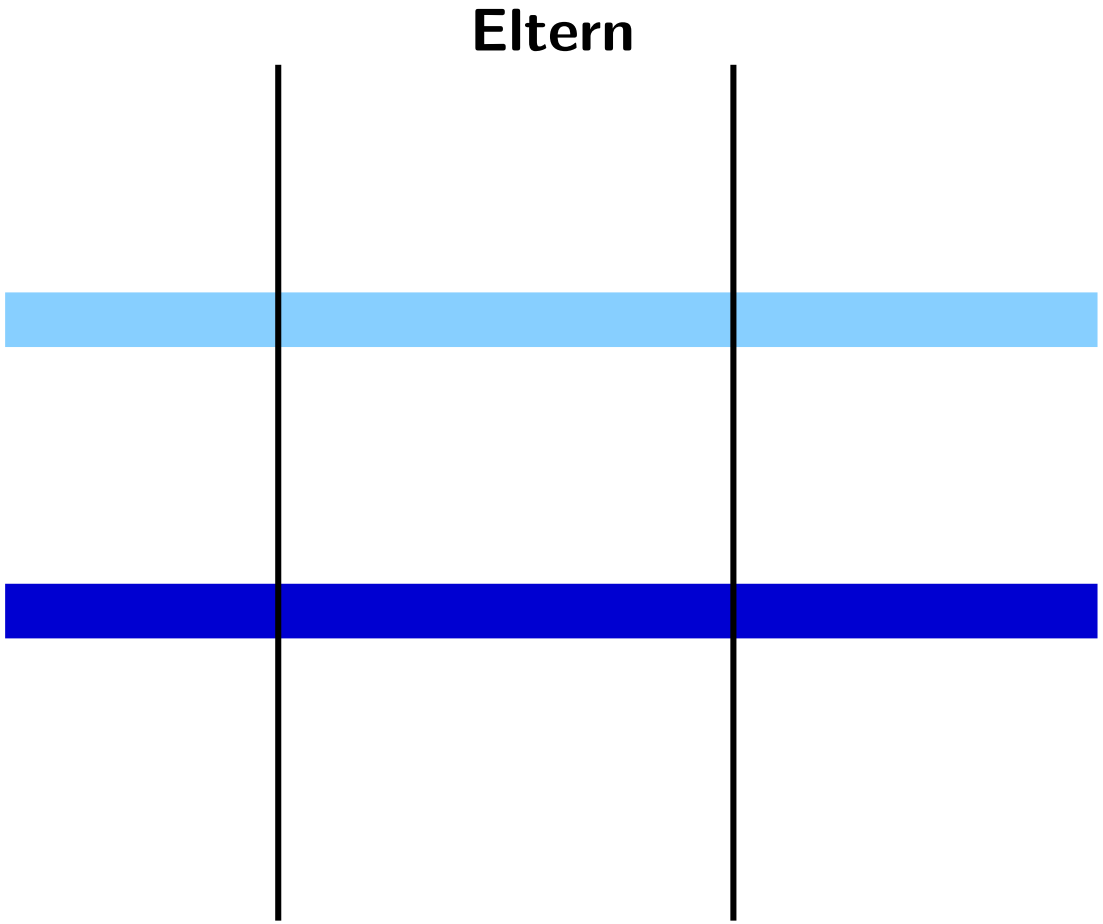
Two Point Crossover

Zwei Positionen im Bitvektor werden zufällig ausgewählt. Zwischen diesn beiden Stellen werden Kopien der Elternvektoren miteinander vertauscht und dadurch die Kinder generiert.

Eltern

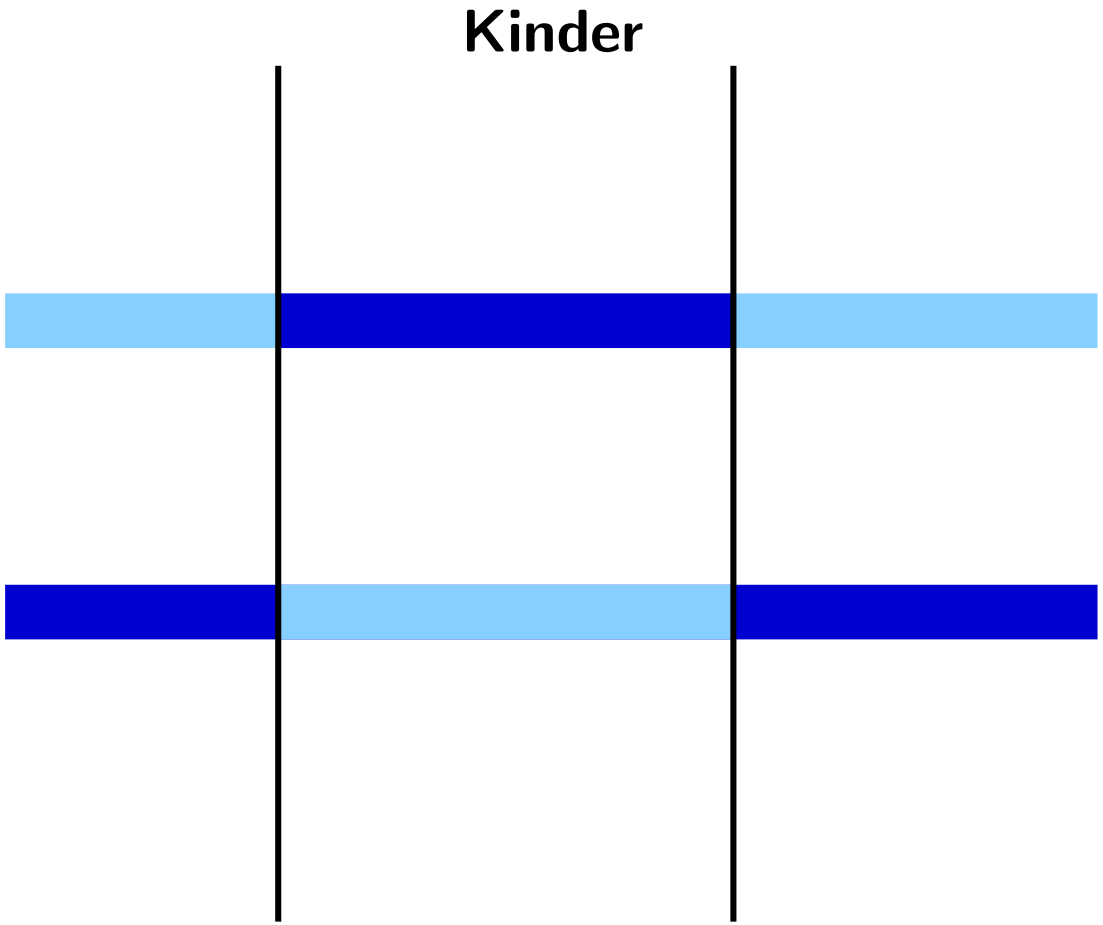
Two Point Crossover

Zwei Positionen im Bitvektor werden zufällig ausgewählt. Zwischen diesn beiden Stellen werden Kopien der Elternvektoren miteinander vertauscht und dadurch die Kinder generiert.



Two Point Crossover

Zwei Positionen im Bitvektor werden zufällig ausgewählt. Zwischen diesn beiden Stellen werden Kopien der Elternvektoren miteinander vertauscht und dadurch die Kinder generiert.



Uniform Crossover

Für jede Position im Bitvektor wird separat entschieden, von welchem Elternteil das Bit kommen soll.

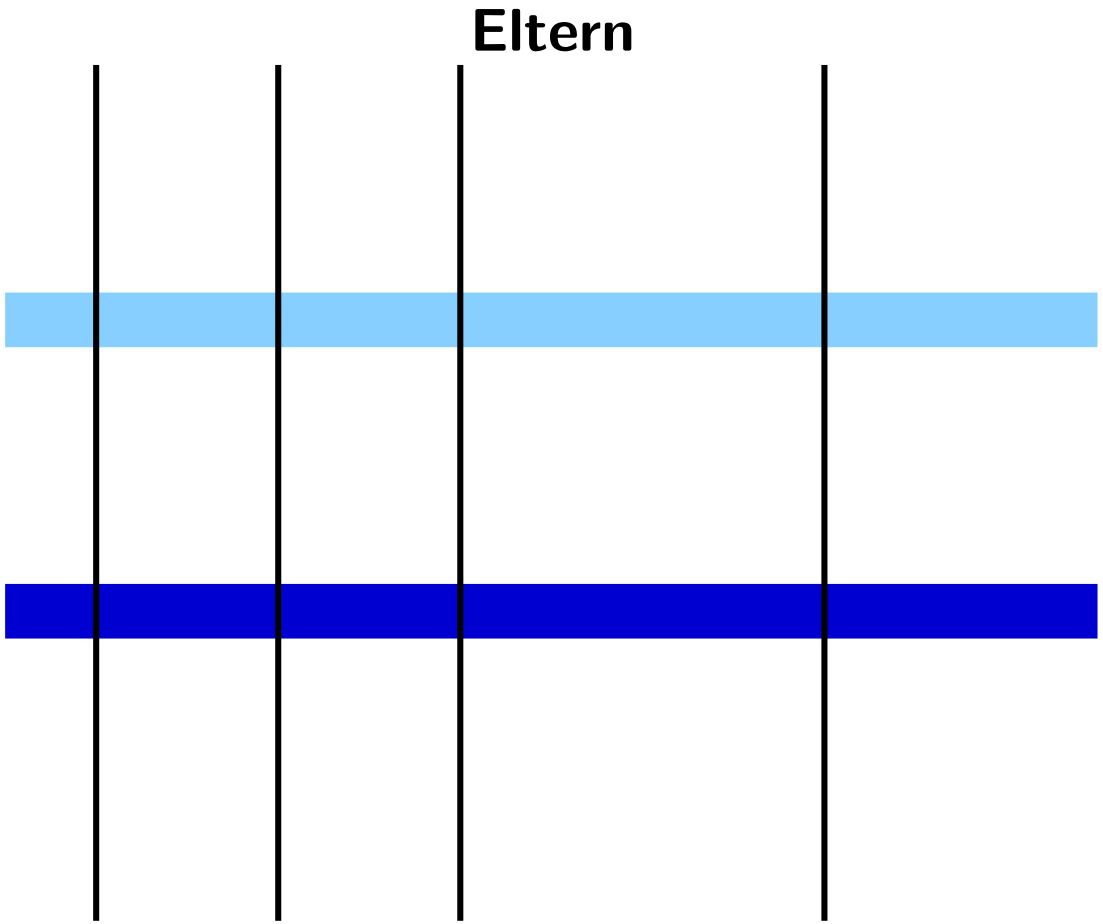
Uniform Crossover

Für jede Position im Bitvektor wird separat entschieden, von welchem Elternteil das Bit kommen soll.

Eltern

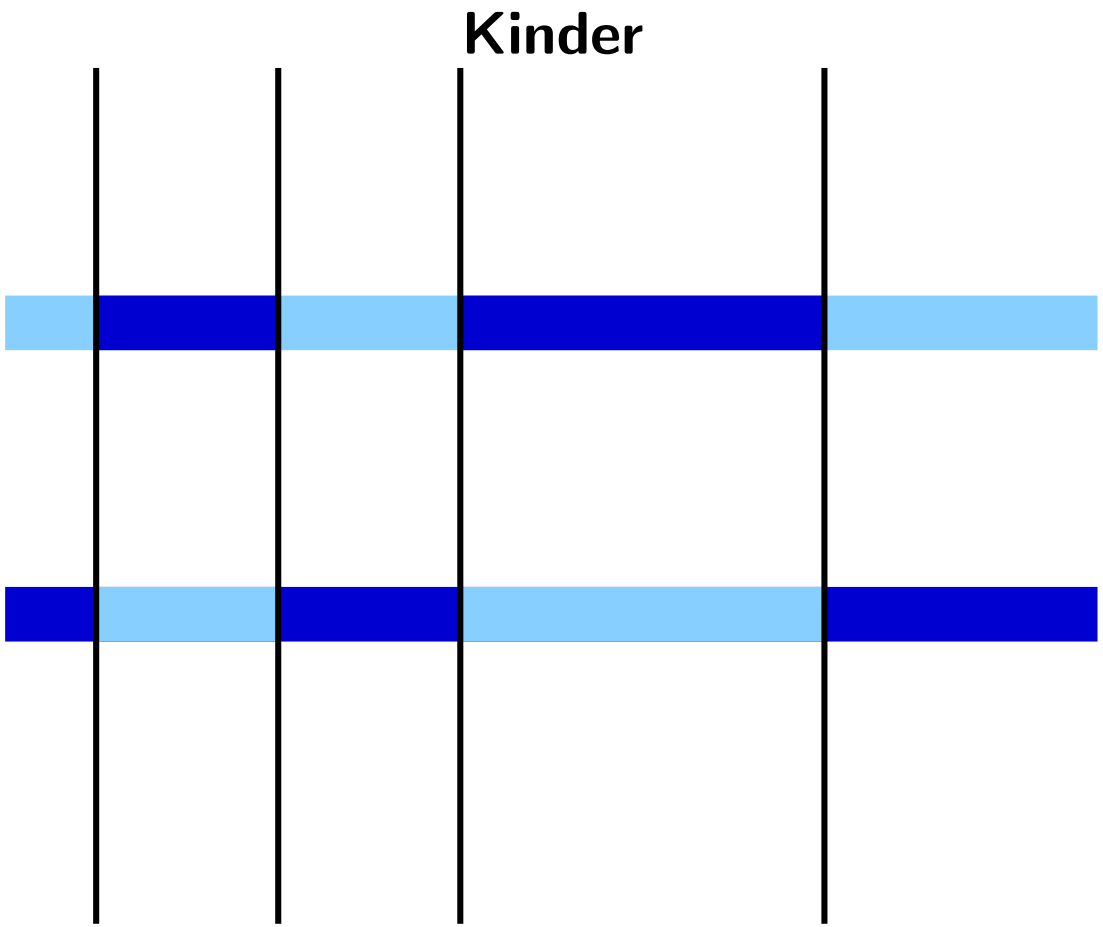
Uniform Crossover

Für jede Position im Bitvektor wird separat entschieden, von welchem Elternteil das Bit kommen soll.



Uniform Crossover

Für jede Position im Bitvektor wird separat entschieden, von welchem Elternteil das Bit kommen soll.



Welchen Effekt haben die einzelen Operatoren

Was ist der Effekt der einzelnen Crossover-Techniken?

Warum ist Mutation sinnvoll?

Welchen Effekt haben die einzelen Operatoren

Was ist der Effekt der einzelnen Crossover-Techniken?

- Single Point Crossover ist potenziell am wenigsten zerstörerisch, lange Ketten von Teillösungen bleiben erhalten
- Two Point und Uniform Crossover haben einen größeren Effekt
- Uniform Crossover kein Längen-Bias
- welche Technik in einem bestimmten Fall an besten funktioniert läßt sich schwer vorhersagen; in der Praxis funktioniert Uniform Crossover oft am besten

Warum ist Mutation sinnvoll?

Welchen Effekt haben die einzelen Operatoren

Was ist der Effekt der einzelnen Crossover-Techniken?

- Single Point Crossover ist potenziell am wenigsten zerstörerisch, lange Ketten von Teillösungen bleiben erhalten
- Two Point und Uniform Crossover haben einen größeren Effekt
- Uniform Crossover kein Längen-Bias
- welche Technik in einem bestimmten Fall an besten funktioniert läßt sich schwer vorhersagen; in der Praxis funktioniert Uniform Crossover oft am besten

Warum ist Mutation sinnvoll?

- ein Element des Zufalls, das die Suche in eine neue Richtung lenken kann
- verhindert, dass einzelne Bits vollständig und für immer verloren gehen

Parameter von GAs

Viele Wahlmöglichkeiten

- Genetische Repräsentation?
- Wieviel der Population wird in jeder Generation ausgetauscht?
- Wie werden die Individuen ausgewählt?
- Welche Operatoren werden verwendet? Mit welcher Wahrscheinlichkeit?
- Wann terminiert das Verfahren?
- etc.

$\Rightarrow$ es gibt viele Parameter zu optimieren und viele Designmöglichkeiten, das ist nicht immer gut!

Vor- und Nachteile von GAs

Vorteile

Nachteile

Vor- und Nachteile von GAs

Vorteile

- oft gut bei **komplexen Problemen**
- **parallelisierbar**

Nachteile

Vor- und Nachteile von GAs

Vorteile

- oft gut bei **komplexen Problemen**
- **parallelisierbar**

Nachteile

- langsam, wenn das Berechnen der Fitnessfunktion aufwendig ist
- Gefahr von **lokalen Optima**
- GAs lassen sich nur anwenden, wenn man **Teillösungen** identifizieren kann

Varianten von GAs

- aufgabenspezifische Operatoren
- Repräsentationen jenseits von Bitvektoren
  - numerische
  - nominale oder
  - strukturierte Repräsentationen (z.B. Bäume)

Zusammenfassung

Was ihr gelernt haben solltet

- Was sind GAs?
- Wie funktioniert ein einfacher GA Algorithmus?
- Wie kann man Lernprobleme als Bitvektoren repräsentieren?
- Was ist eine Fitnessfunktion?
- Welche genetischen Operatoren gibt es? Welchen Effekt haben sie?
- Was sind die Vor- und Nachteile von GAs?