Übung zu Organic Computing

„Survival of the Fittest“

Optimierung mittels Genetischer Algorithmen

Sabine Helwig

Lehrstuhl für Informatik 12 (Hardware-Software-Co-Design)

Universität Erlangen-Nürnberg

sabine.helwig@informatik.uni-erlangen.de

Naturinspirierte Optimierungsverfahren

Wir haben bereits kennengelernt:

- Partikelschwarmoptimierung
- Ameisenalgorithmus

Beide Verfahren optimieren durch Kooperation der Teilnehmer

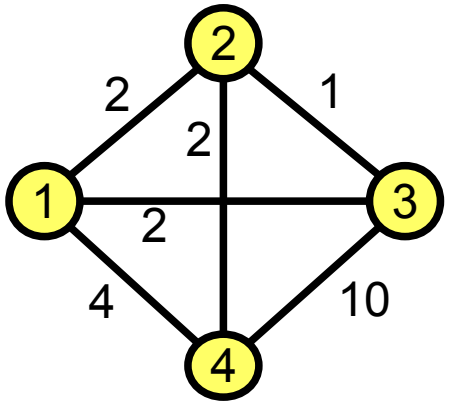
Weiteres naturinspiriertes Optimierungsverfahren:

- Genetische Algorithmen:
  - Nachahmung des Evolutionsprozesses in der Natur
  - Optimierung durch „Survival of the Fittest“

Ziel der heutigen Übung:
Kennenlernen eines genetischen Algorithmus
anhand des *Traveling Salesperson Problem (TSP)*

Traveling Salesperson Problem (TSP)

- Gegeben ist ein vollständig verbundener Graph, jede Kante hat ein Gewicht



- Gesucht ist eine kürzeste Rundreise auf dem Graphen, die jeden Knoten genau einmal besucht

- Codierung für den Genetischen Algorithmus (Genotyp):
  Jede Rundreise wird als Permutation repräsentiert, die die Besuchsreihenfolge der Knoten angibt, z.B. (1, 3, 4, 2).

Der evolutionäre Zyklus

1. Erstellung und Bewertung der Initialpopulation

2. Solange bis maximale Anzahl an Generationen erreicht:
   - **Selektion 1**: Elternselektion
   - **Crossover** (Rekombination): Generieren der Nachkommen
   - **Mutation**
   - Bewertung der Nachkommen
   - **Selektion 2:** Welche Individuen aus der Menge $\{Alte\ Population\} \cup \{Nachkommen\}$ bilden die nächste Generation?

Selektion

- Nur die besten Individuen sollen für das Crossover ausgewählt werden (Selektion 1)
- Nur die besten Individuen sollen für die nächste Generation ausgewählt werden (Selektion 2)

Selektion

- Nur die besten Individuen sollen für das Crossover ausgewählt werden (Selektion 1)
- Nur die besten Individuen sollen für die nächste Generation ausgewählt werden (Selektion 2)

Beispiele für Selektionsverfahren:

- **Turnierselektion**
- **Rangbasierte Selektion**

Zusätzlich kann Elitismus eingesetzt werden: Das beste Individuum wird automatisch in die nächste Generation übernommen.

Selektion

Turnierselektion

- Ziehe zufällig zwei oder mehr Individuen aus der Population.
- Der Bessere gewinnt (evtl. auch prohabilistisch, d.h. der Bessere gewinnt mit höherer Wahrscheinlichkeit).

Selektion

Turnierselektion

- Ziehe zufällig zwei oder mehr Individuen aus der Population.
- Der Bessere gewinnt (evtl. auch prohabilistisch, d.h. der Bessere gewinnt mit höherer Wahrscheinlichkeit).

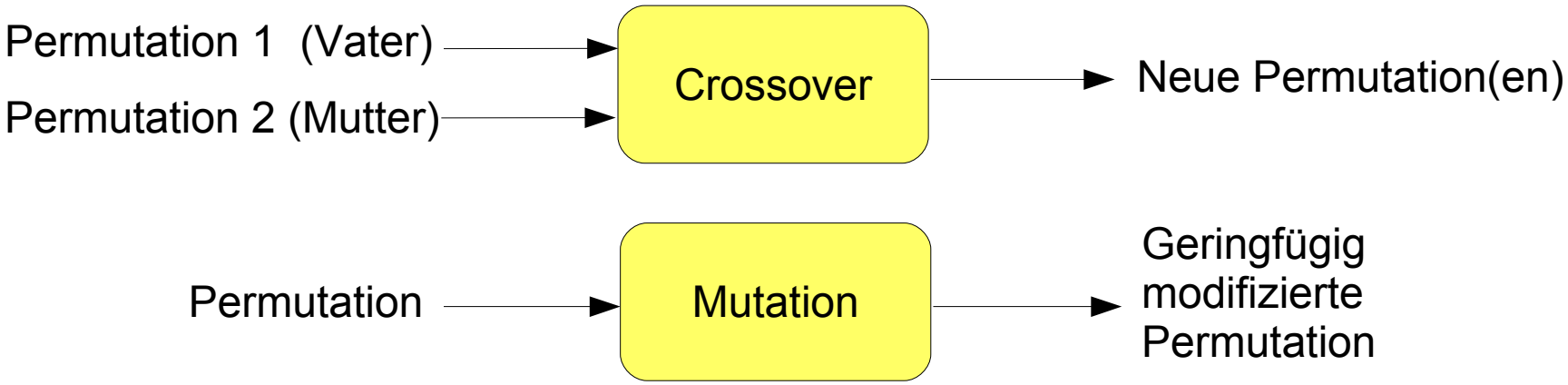
Rangbasierte Selektion

- Erstelle eine Rangliste der Individuen bzgl. ihrer Güte
- Sei I_1 das beste und I_N das schlechteste Individuum
- Wähle I_k mit Wahrscheinlichkeit $Pr[I_k] = \frac{2}{N} \cdot \left(1 - \frac{k-1}{N-1}\right)$
- Auch andere Wahrscheinlichkeitsverteilungen möglich
- Beispiel N=10:

$$Pr[I_1] = \frac{9}{45}, \quad Pr[I_2] = \frac{8}{45}, \quad Pr[I_3] = \frac{7}{45}, \quad \ldots, \quad Pr[I_{N-1}] = \frac{1}{45}, \quad Pr[I_N] = 0$$

Crossover und Mutation

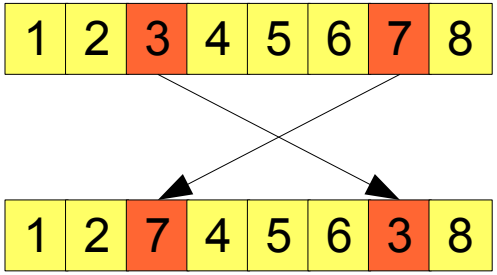
- Verantwortlich für das Generieren neuer Lösungen
- Verschiedene Rollen:
  - Crossover: Die neuen Individuen sollen möglichst viele Eigenschaften der Eltern „erben“
  - Mutation: Geringfügige Modifikation eines Individuums
- Da wir das TSP lösen wollen, müssen beide Operatoren auf Permutationen arbeiten:



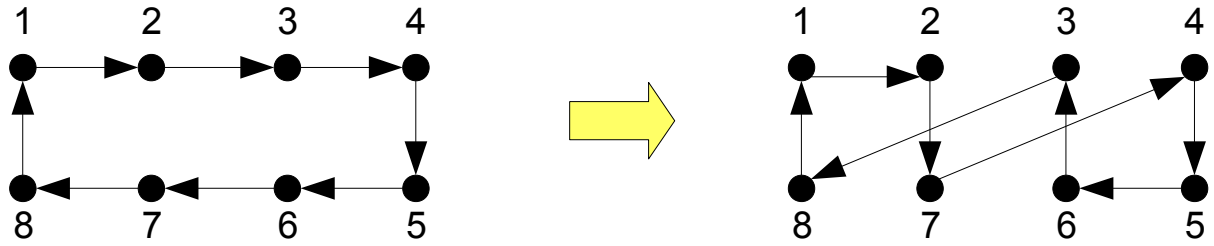
Mutation

Swap mutation operator

- Wähle zufällig zwei Zahlen der Permutation und vertausche sie:



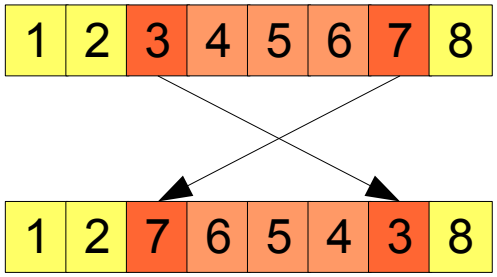
- Auswirkungen auf die Rundreise (Phänotyp):



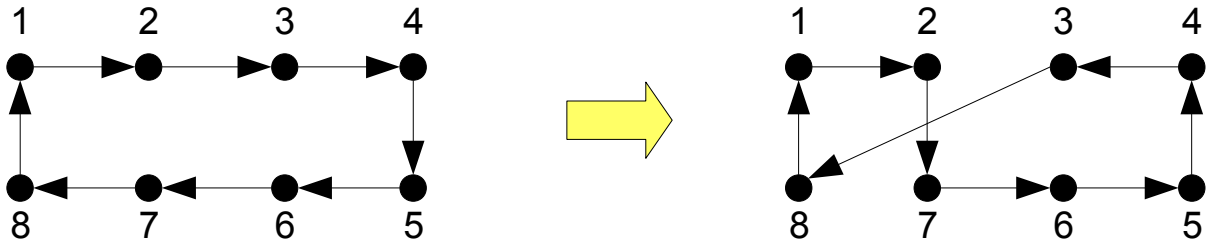
Mutation

Inversion mutation operator

- Wähle zufällig zwei Zahlen der Permutation und spiegle die dazwischenliegende Sequenz:



- Auswirkungen auf die Rundreise (Phänotyp):

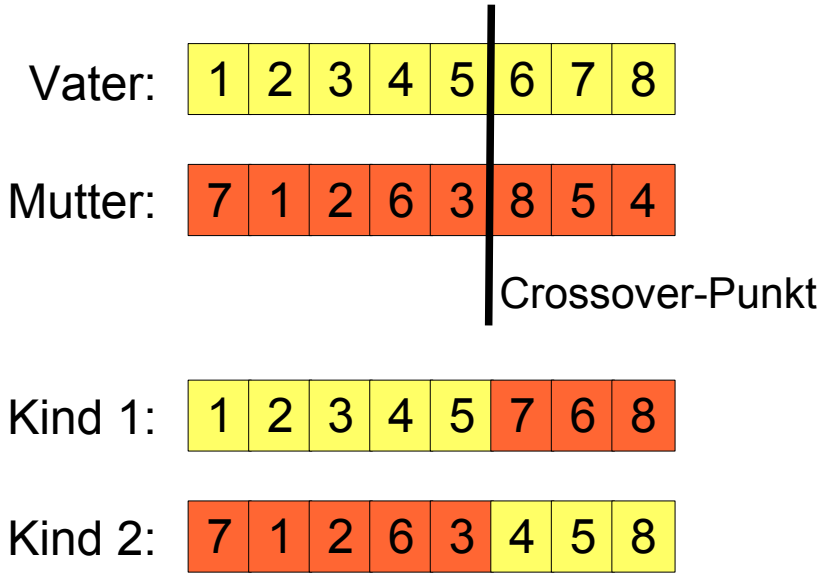


Geringere Auswirkungen als Swap mutation operator.

Crossover

One-point crossover

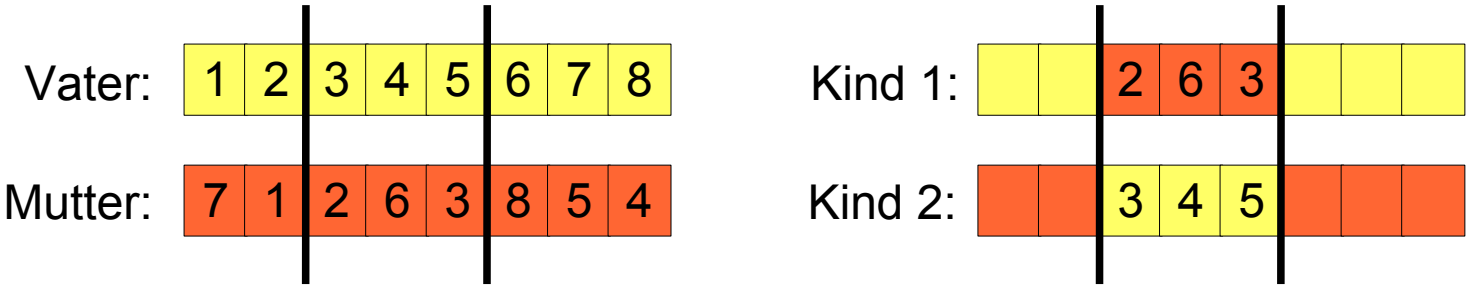
- Wähle zufällig den Crossover-Punkt
- Behalte für jeden Elter den Teil der Permutation bis zum Crossover-Punkt. Die Zahlen nach dem Crossover-Punkt werden so sortiert, wie sie im anderen Elternteil vorliegen.



Crossover

Partially matched crossover (PMX)

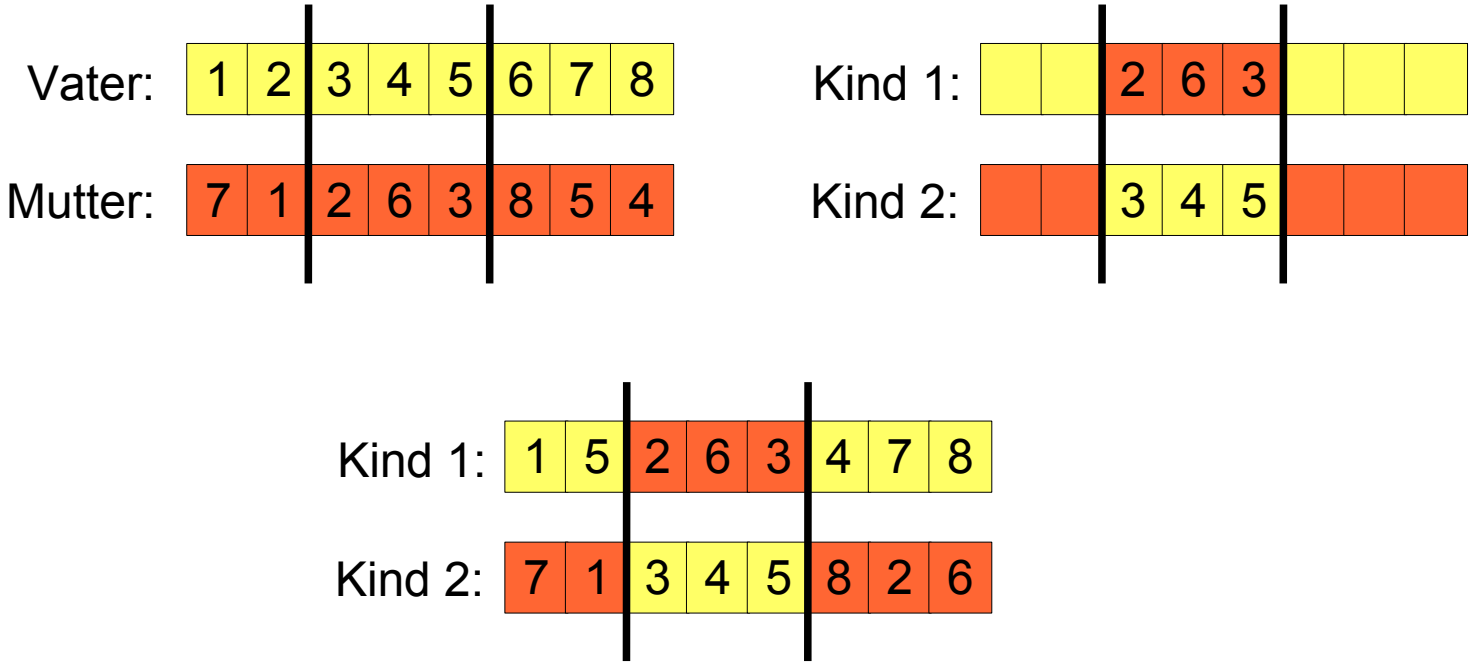
- Wähle zufällig zwei Crossover-Punkte, und tausche die dazwischenliegende Teilpermutation



- Fülle nun die restlichen Positionen mit Zahlen aus dem entsprechenden Elternteil auf. Würde dadurch eine Zahl doppelt vorkommen, betrachte die Teilpermutation zwischen den Crossover-Punkten als Zuordnung und verwende zugeordnete Zahl.
  In unserem Beispiel: 2 → 3, 6 → 4, und 3 → 5

Crossover

Partially matched crossover (PMX)



Crossover

Edge recombination crossover (ERX)

- Ziel: Möglichst viele Kanten der Nachkommen sollen auch in einem Elternteil vorgekommen sein.
- Dazu wird die Nachbarschaftsinformation jedes Knotens extrahiert:

Vater:	1	2	3	4	5	6
Mutter:	4	1	2	6	3	5

Nachbarschaften:

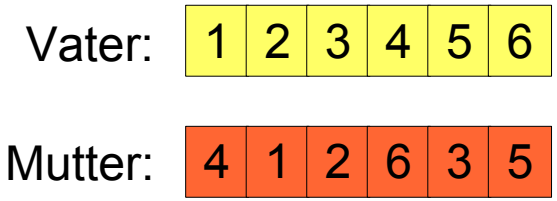
1: 6, 2, 4
2: 1, 3, 6
3: 2, 4, 5, 6
4: 3, 5, 1
5: 4, 6, 3
6: 5, 1, 2, 3

Crossover

Edge recombination crossover (ERX)

Nachbarschaften:

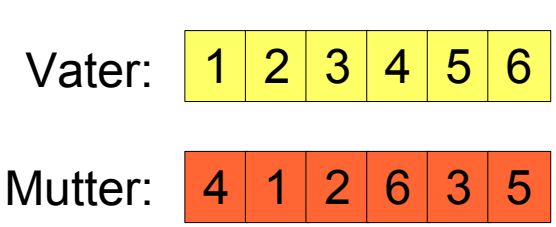
1: 6, 2, 4
2: 1, 3, 6
3: 2, 4, 5, 6
4: 3, 5, 1
5: 4, 6, 3
6: 5, 1, 2, 3

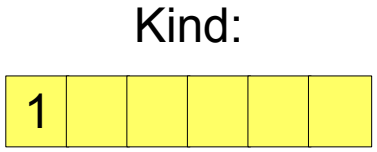


- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählte Zahl aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)





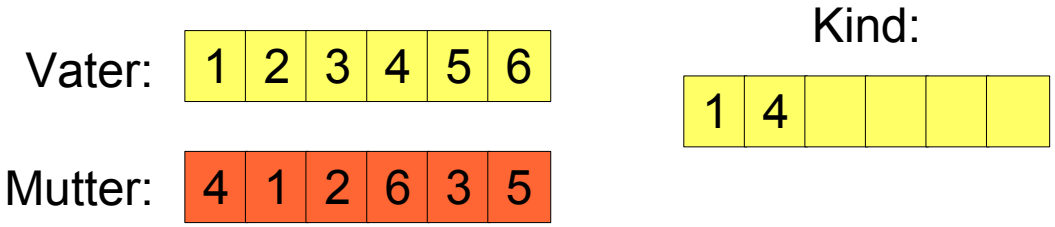
Nachbarschaften:

1: 6, 2, 4
2: 1, 3, 6
3: 2, 4, 5, 6
4: 3, 5, 1
5: 4, 6, 3
6: 5, 1, 2, 3

- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)



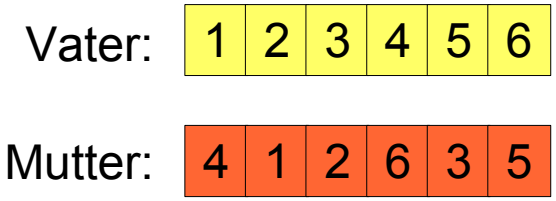
Nachbarschaften:

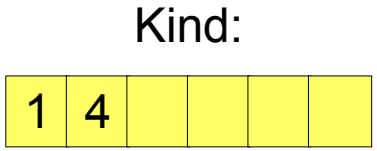
~~1: 6, 2, 4~~
2: ~~1~~, 3, 6
3: 2, 4, 5, 6
4: 3, 5, ~~1~~
5: 4, 6, 3
6: 5, ~~1~~, 2, 3

- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)





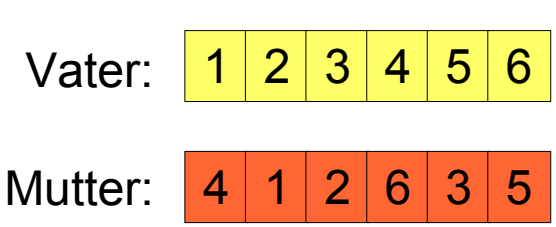
Nachbarschaften:

2: 3, 6
3: 2, 4, 5, 6
4: 3, 5
5: 4, 6, 3
6: 5, 2, 3

- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)



Kind:

1	4	5			

Nachbarschaften:

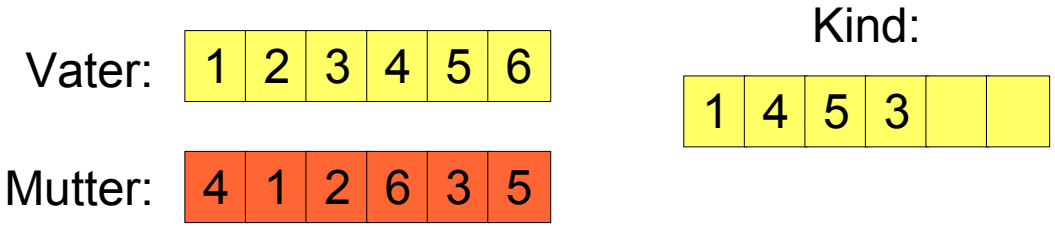
2: 3, 6
3: 2, 5, 6

5: 6, 3
6: 5, 2, 3

- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)



Nachbarschaften:

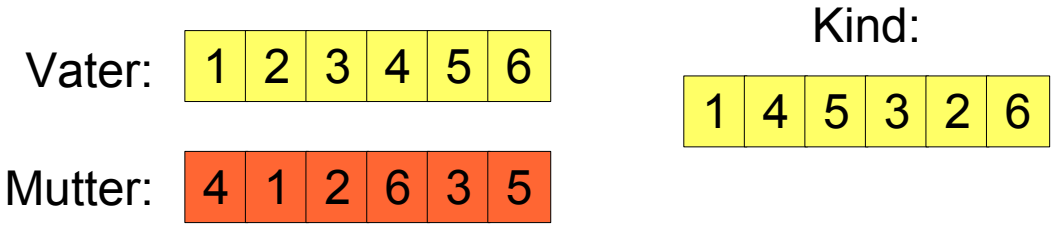
2: 3, 6
3: 2, 6

6: 2, 3

- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Crossover

Edge recombination crossover (ERX)



- Starte mit dem Anfangswert eines zufälligen Elternteils
- Solange das Kind nicht fertig ist:
  - Wähle als Nachfolger den Nachbar mit der kürzesten Nachbarliste (zufällige Wahl falls mehrere solche existieren)
  - Gibt es keinen Nachbarn: Wähle zufällig freie Zahl
  - Lösche gewählten Knoten aus allen Nachbarschaften

Der evolutionäre Zyklus

Wie könnte eine konkrete Implementierung für das TSP aussehen?

1. Erstelle 50 zufällige Permutationen (Initialpopulation) und bewerte diese.

2. Wiederhole 500-mal:
   - **Selektion 1**: Wähle 50 Elternpaare mittels Turnierselektion
   - **Crossover**: Erstelle 50 Kinder mittels ERX
   - **Mutation**: Führe auf jedem Kind mit einer Wahrscheinlichkeit von 1% Inversion mutation aus.
   - Bewerte die Nachkommen
   - **Selektion 2:** Wähle aus den nun 100 vorhandenen Individuen 50 verschiedene Individuen mittels rangbasierter Selektion für die nächste Generation aus. Übernehme auf jeden Fall das beste Individuum (Elitismus).

Literatur

- David E. Goldberg: ***Genetic Algorithms in Search, Optimization & Machine Learning.*** Addison-Wesley Publishing Company, 1989
- Karsten Weicker: ***Evolutionäre Algorithmen.*** Teubner GmbH, 2002