h_da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Alexander del Pino
Fachbereich Informatik

Genetische Algorithmen

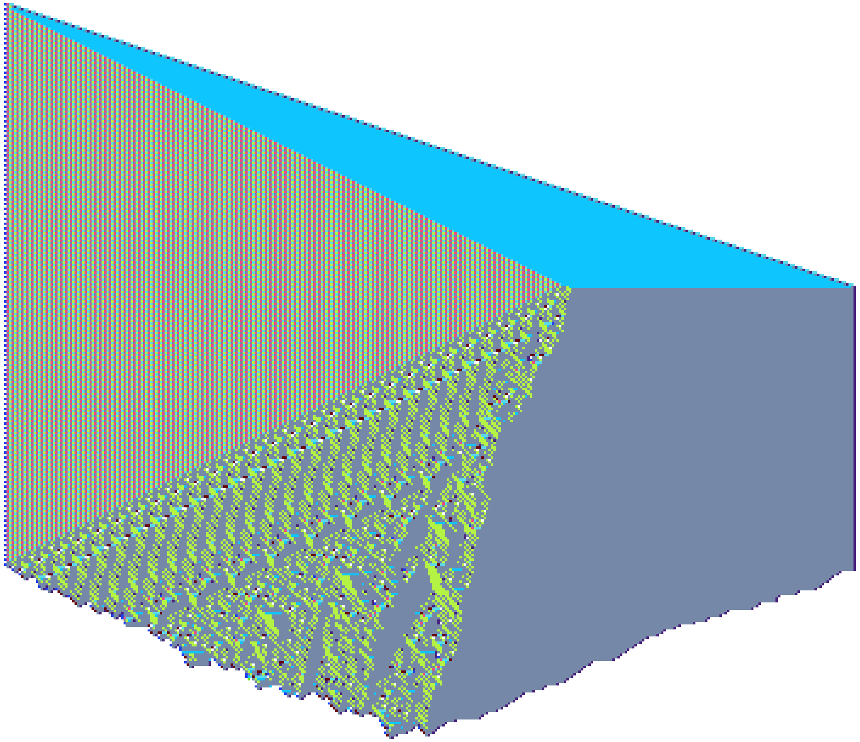


Image: aa8490b2a6cc0084769018ca264960c1. © Alexander del Pino

4. Teil

Kodierung von Lösungs-kandidaten

Kodierung von Lösungskandidaten

Kodierung

Der Erfolg eines genetischen Algorithmus hängt maßgeblich davon ab, wie die Lösungskandidaten für ein bestimmtes Problem kodiert werden.

Gerade bei genetischen Algorithmen werden oftmals Bitstrings fester Länge *(binary encoding)* zur Kodierung verwendet.

Kodierung von Lösungskandidaten

Kodierung

Der Erfolg eines genetischen Algorithmus hängt maßgeblich davon ab, wie die Lösungskandidaten für ein bestimmtes Problem kodiert werden.

Gerade bei genetischen Algorithmen werden oftmals Bitstrings fester Länge *(binary encoding)* zur Kodierung verwendet.

- Historische Gründe: Obwohl Biologen teilweise schon in den 1950er Jahren Computer zur Simulation von genetischen Systemen verwendet haben, wurden genetische Algorithmen entscheidend duch die Arbeiten von *John Holland* in den 1960er und 1970er Jahren geprägt. Ursprünglich arbeitete Holland und später auch z.B. *David Goldberg,* 1989, mit solchen Bitstrings fester Länge.

Kodierung von Lösungskandidaten

Kodierung

Der Erfolg eines genetischen Algorithmus hängt maßgeblich davon ab, wie die Lösungskandidaten für ein bestimmtes Problem kodiert werden.

Gerade bei genetischen Algorithmen werden oftmals Bitstrings fester Länge *(binary encoding)* zur Kodierung verwendet.

- Historische Gründe: Obwohl Biologen teilweise schon in den 1950er Jahren Computer zur Simulation von genetischen Systemen verwendet haben, wurden genetische Algorithmen entscheidend duch die Arbeiten von *John Holland* in den 1960er und 1970er Jahren geprägt. Ursprünglich arbeitete Holland und später auch z.B. *David Goldberg,* 1989, mit solchen Bitstrings fester Länge.
- Die Theorie zu den genetischen Algorithmen ist bei dieser Art von Kodierung am weitesten fortgeschritten.

Kodierung von Lösungskandidaten

Kodierung

Der Erfolg eines genetischen Algorithmus hängt maßgeblich davon ab, wie die Lösungskandidaten für ein bestimmtes Problem kodiert werden.

Gerade bei genetischen Algorithmen werden oftmals Bitstrings fester Länge *(binary encoding)* zur Kodierung verwendet.

- Historische Gründe: Obwohl Biologen teilweise schon in den 1950er Jahren Computer zur Simulation von genetischen Systemen verwendet haben, wurden genetische Algorithmen entscheidend duch die Arbeiten von *John Holland* in den 1960er und 1970er Jahren geprägt. Ursprünglich arbeitete Holland und später auch z.B. *David Goldberg,* 1989, mit solchen Bitstrings fester Länge.
- Die Theorie zu den genetischen Algorithmen ist bei dieser Art von Kodierung am weitesten fortgeschritten.
- Heuristiken für die Mutationsrate oder die Crossoverrate wurden oftmals unter der Annahme entwickelt, dass die Lösungskandidaten mit einer festen Länge binär kodiert sind.

Kodierung von Lösungskandidaten

Kodierung

Trotz all der Erfahrung die sich im Laufe der Jahrzehnte mit Bitstrings fester Länge angesammelt hat, wird diese Art der Kodierung heutzutage für viele Probleme als unnatürlich und unhandlich angesehen.

Kodierung von Lösungskandidaten

Kodierung

Trotz all der Erfahrung die sich im Laufe der Jahrzehnte mit Bitstrings fester Länge angesammelt hat, wird diese Art der Kodierung heutzutage für viele Probleme als unnatürlich und unhandlich angesehen.

Beispiel

Die Lösung für eine numerische Optimierungsaufgabe wird durch einen Vektor aus 100 Zahlen im Bereich [0 ... 1000] beschrieben, und soll auf sechs Nachkommastellen angenähert werden.

Für eine solche Zahl benötigt man log(1000 * 10^6) / log (2) = 29.897 Bit, also 30 Bit.

Ein Lösungskandidat benötigt also 100 * 30 = 3000 Bit.

Dies entspricht etwa einem Suchraum der Größe 10^903.

Für solche Probleme sind genetische Algorithmen ungeeignet.

Kodierung von Lösungskandidaten

Kodierung

Es ist sinnvoll, die Kodierung dem jeweiligen Problem anzupassen. Dadurch können auch auch die genetischen Operatoren problemspezifischer realisiert werden.

Kodierung von Lösungskandidaten

Kodierung

Es ist sinnvoll, die Kodierung dem jeweiligen Problem anzupassen. Dadurch können auch auch die genetischen Operatoren problemspezifischer realisiert werden.

- Derzeit gibt es allerdings keine allgemein gültigen Leitlinien, welche Art der Kodierung für ein bestimmtes Problem jeweils am besten ist.

How is one to decide on the correct encoding for one's problem? Lawrence Davis, a researcher with much experience applying GAs to real-world problems strongly advocates using whatever encoding is the most natural for your problem, and then devising a GA that can use that encoding.

Until the theory of GAs and encodings is better formulated, this might be the best philosophy; [...] most research is currently done by guessing at an appropriate encoding and then trying out a particular version of the GA on it.

Quelle: M. Mitchell: *An Introduction to Genetic Algorithms*, MIT Press, 1996

Kodierung von Lösungskandidaten

Kodierung

Wenn sich zwei Lösungskandidaten im Genotyp ähnlich sind, dann sollte dies auch zu ähnlichen Phänotypen führen.

Kodierung von Lösungskandidaten

Kodierung

Wenn sich zwei Lösungskandidaten im Genotyp ähnlich sind, dann sollte dies auch zu ähnlichen Phänotypen führen.

Ist dies bei der binären Kodierung der Fall ?

Kodierung von Lösungskandidaten

Kodierung

Wenn sich zwei Lösungskandidaten im Genotyp ähnlich sind, dann sollte dies auch zu ähnlichen Phänotypen führen.

Ist dies bei der binären Kodierung der Fall ?

Beispiel

Genotyp	***Phänotyp***
0111111111111111	*32767*
1000000000000000	*32768*

Alle 16 Bits ändern sich im Genotyp, im Phänotyp führt dies trotzdem nur zur kleinstmöglichen Veränderung, nämlich +1.

Kodierung von Lösungskandidaten

Kodierung

Wenn sich zwei Lösungskandidaten im Genotyp ähnlich sind, dann sollte dies auch zu ähnlichen Phänotypen führen.

Ist dies bei der binären Kodierung der Fall ?

Beispiel

Genotyp	***Phänotyp***
0111111111111111	*32767*
1000000000000000	*32768*

Alle 16 Bits ändern sich im Genotyp, im Phänotyp führt dies trotzdem nur zur kleinstmöglichen Veränderung, nämlich +1.

Genotyp	***Phänotyp***
0000000000000001	*1*
1000000000000001	*32769*

Im Genotyp ändert sich nur ein einziges Bit, im Phänotyp führt dies jedoch zu einer Veränderung von +32768.

Kodierung von Lösungskandidaten

Hamming-Distanz

Unter der *Hamming-Distanz* zweier Zahlen versteht man die Anzahl der Bits an denen sich beide Zahlen voneinander unterscheiden.

Kodierung von Lösungskandidaten

Hamming-Distanz

Unter der *Hamming-Distanz* zweier Zahlen versteht man die Anzahl der Bits an denen sich beide Zahlen voneinander unterscheiden.

Die Hamming-Distanz zweier aufeinander folgende Zahlen kann sehr unterschiedlich sein:

N	**N+1**	**H(N, N+1)**
0000	0001	1
0001	0010	2
0010	0011	1
0011	0100	3
0100	0101	1
0101	0110	2
0110	0111	1
0111	1000	4
1000	1001	1
1001	1010	2
1010	1011	1
1011	1100	3
1100	1101	1
1101	1110	2
1110	1111	1

Kodierung von Lösungskandidaten

Hamming-Distanz

Unter der *Hamming-Distanz* zweier Zahlen versteht man die Anzahl der Bits an denen sich beide Zahlen voneinander unterscheiden.

Die Hamming-Distanz zweier aufeinander folgende Zahlen kann sehr unterschiedlich sein:

N	N+1	H(N, N+1)
0000	0001	1
0001	0010	2
0010	0011	1
0011	0100	3
0100	0101	1
0101	0110	2
0110	0111	1
0111	1000	4
1000	1001	1
1001	1010	2
1010	1011	1
1011	1100	3
1100	1101	1
1101	1110	2
1110	1111	1

```java
// Berechnung der Hamming-Distanz
// zweier Zahlen in Java

public int hamming(int a, int b) {
   return Integer.bitCount(a ^ b);
}
```

Kodierung von Lösungskandidaten

Der Gray-Code

Der *Gray-Code* ist so konstruiert, dass die Hamming-Distanz zweier aufeinander folgenden Zahlen stets eins ist.

N	**Gray(N)**	**Gray(N+1)**	**H(Gray(N),Gray(N+1))**
0000	0000	0001	1
0001	0001	0011	1
0010	0011	0010	1
0011	0010	0110	1
0100	0110	0111	1
0101	0111	0101	1
0110	0101	0100	1
0111	0100	1100	1
1000	1100	1101	1
1001	1101	1111	1
1010	1111	1110	1
1011	1110	1010	1
1100	1010	1011	1
1101	1011	1001	1
1110	1001	1000	1
1111	1000	11000	1

Kodierung von Lösungskandidaten

Der Gray-Code

Der *Gray-Code* ist so konstruiert, dass die Hamming-Distanz zweier aufeinander folgenden Zahlen stets eins ist.

N	**Gray(N)**	**Gray(N+1)**	**H(Gray(N),Gray(N+1))**
0000	0000	0001	1
0001	0001	0011	1
0010	0011	0010	1
0011	0010	0110	1
0100	0110	0111	1
0101	0111	0101	1
0110	0101	0100	1
0111	0100	1100	1
1000	1100	1101	1
1001	1101	1111	1
1010	1111	1110	1
1011	1110	1010	1
1100	1010	1011	1
1101	1011	1001	1
1110	1001	1000	1
1111	1000	11000	1

Welchen Vorteil bringt die Verwendung eines Gray-Codes ?

Kodierung von Lösungskandidaten

Umwandlung Binärcode nach Gray-Code

Die Umwandlung einer Binärzahl $B = b_n,...,b_0$ in ihren Gray-code $G=g_n,...,g_0$ geschieht wie folgt:

$$
\begin{aligned}
&g_n = b_n \\
&\texttt{for (int i=n-1; i>=0; i--) \{} \\
&\quad g_i = b_{i+1} \otimes b_i \\
&\texttt{\}}
\end{aligned}
$$

Kodierung von Lösungskandidaten

Umwandlung Binärcode nach Gray-Code

Die Umwandlung einer Binärzahl B = $b_n,\ldots,b_0$ in ihren Gray-code G=$g_n,\ldots,g_0$ geschieht wie folgt:

$$g_n = b_n$$

```
for (int i=n-1; i>=0; i--) {
```

$$g_i = b_{i+1} \otimes b_i$$

```
}
```

In Java kann dies beispielsweise so implementiert werden:

```java
// Berechnung des Gray-Codes zu einem Binärcode
public int binaryToGray(int binary) {
    return binary ^ (binary  >> 1);
}
```

Kodierung von Lösungskandidaten

Umwandlung Gray-Code nach Binärcode

Die Umwandlung des Gray-codes $G=g_n,\dots,g_0$ in eine Binärzahl $B = b_n,\dots,b_0$ geschieht wie folgt:

```
bn = gn

for (int i=n-1; i>=0; i--) {

    bi = bi+1 ⊗ gi

}
```

Kodierung von Lösungskandidaten

Umwandlung Gray-Code nach Binärcode

Die Umwandlung des Gray-codes $G=g_n,...,g_0$ in eine Binärzahl $B = b_n,...,b_0$ geschieht wie folgt:

$$b_n = g_n$$

for (int i=n-1; i>=0; i--) {

$$b_i = b_{i+1} \otimes g_i$$

}

In Java kann dies beispielsweise so implementiert werden:

```java
// Umwandlung Gray-Code nach Binärcode in Java
public int grayToBinary(int gray) {
    int binary;
    for (binary = 0; gray != 0; gray >>= 1) {
        binary ^= gray;
    }
    return binary;
}
```

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bisher haben wir immer nur einen einzigen Parameter in dem Genotyp kodiert, gewissermaßen enthielt ein Chromosom bisher also lediglich ein einziges Gen.

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bisher haben wir immer nur einen einzigen Parameter in dem Genotyp kodiert, gewissermaßen enthielt ein Chromosom bisher also lediglich ein einziges Gen.

Oftmals sind jedoch die Lösungskandidaten Vektoren einer festen Größe. Bei dem *mapped multiparameter coding* nach *D. Goldberg* werden die einzelnen Elemente eines Vektors einzeln kodiert und anschließend konkateniert.

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bisher haben wir immer nur einen einzigen Parameter in dem Genotyp kodiert, gewissermaßen enthielt ein Chromosom bisher also lediglich ein einziges Gen.

Oftmals sind jedoch die Lösungskandidaten Vektoren einer festen Größe. Bei dem *mapped multiparameter coding* nach *D. Goldberg* werden die einzelnen Elemente eines Vektors einzeln kodiert und anschließend konkateniert.

Beispiel:

$V=<v_1,...,v_n>$ mit 8 Bit je Einzelelement

01001100	11010110	00010101	...	01110110
v_1	v_2	v_3		v_n

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bisher haben wir immer nur einen einzigen Parameter in dem Genotyp kodiert, gewissermaßen enthielt ein Chromosom bisher also lediglich ein einziges Gen.

Oftmals sind jedoch die Lösungskandidaten Vektoren einer festen Größe. Bei dem *mapped multiparameter coding* nach *D. Goldberg* werden die einzelnen Elemente eines Vektors einzeln kodiert und anschließend konkateniert.

Beispiel:

$V=<v_1,...,v_n>$ mit 8 Bit je Einzelelement

01001100	11010110	00010101	...	01110110
v_1	v_2	v_3		v_n

Mehrere Parameter werden einzeln und unabhängig voneinander kodiert und anschließend konkateniert.

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bei der Kodierung mehrerer Parameter auf die eben vorgestellte Art und Weise ergeben sich zwei natürliche Varianten für das Crossover:

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bei der Kodierung mehrerer Parameter auf die eben vorgestellte Art und Weise ergeben sich zwei natürliche Varianten für das Crossover:

Crossover ist nur an Elementgrenzen erlaubt:

Vorher	*0101 | 1101 | 0010*	*1110 | 0101 | 0111*
Links	*0101 | 1101 | 0010*	*1110 | 0101 | 0111*
Rechts	*0101 | 1101 | 0010*	*1110 | 0101 | 0111*
Nachher	*0101 | 0101 | 0111*	*1110 | 1101 | 0010*

Kodierung von Lösungskandidaten

Kodierung mehrerer Parameter

Bei der Kodierung mehrerer Parameter auf die eben vorgestellte Art und Weise ergeben sich zwei natürliche Varianten für das Crossover:

Crossover ist nur an Elementgrenzen erlaubt:

Vorher	*0101|1101|0010*	*1110|0101|0111*
Links	*0101|1101|0010*	*1110|0101|0111*
Rechts	*0101|1101|0010*	*1110|0101|0111*
Nachher	*0101|0101|0111*	*1110|1101|0010*

Crossover ist auch innerhalb von Einzelelementen erlaubt:

Vorher	*0101|1101|0010*	*1110|0101|0111*
Links	*0101|1101|0010*	*1110|0101|0111*
Rechts	*0101|1101|0010*	*1110|0101|0111*
Nachher	*0110|0101|0111*	*1101|1101|0010*

Kodierung von Lösungskandidaten

Kodierung mit Gleitkommazahlen

Bei einigen Problemen ist die natürliche Darstellung einer Lösung durch einen Vektor von Gleitkommazahlen gegeben.

Man kann nun nach dem eben vorgestellten Muster auch Gleitkommazahlen anstatt binäre Integerzahlen in einem Chromosom kodieren.

Die erreichbare Genauigkeit hängt dabei von der zugrunde liegenden Zahlendarstellung ab.

Kodierung von Lösungskandidaten

Kodierung mit Gleitkommazahlen

Bei einigen Problemen ist die natürliche Darstellung einer Lösung durch einen Vektor von Gleitkommazahlen gegeben.

Man kann nun nach dem eben vorgestellten Muster auch Gleitkommazahlen anstatt binäre Integerzahlen in einem Chromosom kodieren.

Die erreichbare Genauigkeit hängt dabei von der zugrunde liegenden Zahlendarstellung ab.

Beispiel:

In Java werden Gleitkommazahlen nach dem IEEE 754 Standard kodiert

	Vorzeichen	***Exponent***	***Mantisse***
float	*1 bit*	*8 bit*	*23 bit*
double	*1 bit*	*11 bit*	*52 bit*

Kodierung von Lösungskandidaten

Kodierung mit Gleitkommazahlen

Bei einigen Problemen ist die natürliche Darstellung einer Lösung durch einen Vektor von Gleitkommazahlen gegeben.

Man kann nun nach dem eben vorgestellten Muster auch Gleitkommazahlen anstatt binäre Integerzahlen in einem Chromosom kodieren.

Die erreichbare Genauigkeit hängt dabei von der zugrunde liegenden Zahlendarstellung ab.

Beispiel:

In Java werden Gleitkommazahlen nach dem IEEE 754 Standard kodiert

	Vorzeichen	***Exponent***	***Mantisse***
float	*1 bit*	*8 bit*	*23 bit*
double	*1 bit*	*11 bit*	*52 bit*

Welche Vor- bzw. Nachteile sehen Sie bei dieser Kodierung im Vergleich mit einer binären Kodierung fester Bitlänge ?

Kodierung von Lösungskandidaten

Benachbarte Loci vs. Crossover

Je mehr funktional zusammengehörige Bits auf dem Chromosom auch räumlich beieinander liegen, umso geringer ist die Wahrscheinlichkeit, durch Crossover wieder aufgetrennt zu werden.

Kodierung von Lösungskandidaten

Benachbarte Loci vs. Crossover

Je mehr funktional zusammengehörige Bits auf dem Chromosom auch räumlich beieinander liegen, umso geringer ist die Wahrscheinlichkeit, durch Crossover wieder aufgetrennt zu werden.

Beispiel

Bits mit gleicher Farbe gehören hier funktional zusammen.

1010010101000101101101010010101100101111 0111

Die blaue hat im Vergleich zu der roten oder grünen Bitgruppe eine viel höhere Wahrscheinlichkeit, durch Crossover wieder aufgetrennt zu werden.

Kodierung von Lösungskandidaten

Benachbarte Loci vs. Crossover

Je mehr funktional zusammengehörige Bits auf dem Chromosom auch räumlich beieinander liegen, umso geringer ist die Wahrscheinlichkeit, durch Crossover wieder aufgetrennt zu werden.

Beispiel

Bits mit gleicher Farbe gehören hier funktional zusammen.

1010010101000101101101010100101011001011110111

Die blaue hat im Vergleich zu der roten oder grünen Bitgruppe eine viel höhere Wahrscheinlichkeit, durch Crossover wieder aufgetrennt zu werden.

Wie könnte man die Wahrscheinlichkeit dass eine solche Bitgruppe durch Crossover aufgetrennt wird, berechnen ?

Kodierung von Lösungskandidaten

Das Linkage Problem

Welche Kodierung die Beste für ein spezielles Problem ist, ist oftmals unbekannt.

Man wählt also einfach eine Kodierung von der man glaubt dass sie geeignet sei.

Kodierung von Lösungskandidaten

Das Linkage Problem

Welche Kodierung die Beste für ein spezielles Problem ist, ist oftmals unbekannt. Man wählt also einfach eine Kodierung von der man glaubt dass sie geeignet sei.

Dabei kann es aber passieren, dass gute Lösungskandidaten nur dann im Suchraum erreicht werden, wenn sich eine Gruppe von Bits die über das ganze Chromosom verteilt ist, gleichzeitig auf eine bestimmte Art und Weise ändert.

Kodierung von Lösungskandidaten

Das Linkage Problem

Welche Kodierung die Beste für ein spezielles Problem ist, ist oftmals unbekannt. Man wählt also einfach eine Kodierung von der man glaubt dass sie geeignet sei.

Dabei kann es aber passieren, dass gute Lösungskandidaten nur dann im Suchraum erreicht werden, wenn sich eine Gruppe von Bits die über das ganze Chromosom verteilt ist, gleichzeitig auf eine bestimmte Art und Weise ändert.

Die Gründe dafür sind Epistasie-ähnliche Effekte zwischen solchen Bits.

Kodierung von Lösungskandidaten

Das Linkage Problem

Welche Kodierung die Beste für ein spezielles Problem ist, ist oftmals unbekannt. Man wählt also einfach eine Kodierung von der man glaubt dass sie geeignet sei.

Dabei kann es aber passieren, dass gute Lösungskandidaten nur dann im Suchraum erreicht werden, wenn sich eine Gruppe von Bits die über das ganze Chromosom verteilt ist, gleichzeitig auf eine bestimmte Art und Weise ändert.

Die Gründe dafür sind Epistasie-ähnliche Effekte zwischen solchen Bits.

Wenn eine solche Bitgruppe auf dem Chromosom eng beieinander kodiert wäre, so dass die Wahrscheinlichkeit durch Crossover getrennt zu werden sich verringert, dann könnte der genetische Algorithmus möglicherweise bessere Kandidaten finden.

Kodierung von Lösungskandidaten

Das Linkage Problem

Welche Kodierung die Beste für ein spezielles Problem ist, ist oftmals unbekannt. Man wählt also einfach eine Kodierung von der man glaubt dass sie geeignet sei.

Dabei kann es aber passieren, dass gute Lösungskandidaten nur dann im Suchraum erreicht werden, wenn sich eine Gruppe von Bits die über das ganze Chromosom verteilt ist, gleichzeitig auf eine bestimmte Art und Weise ändert.

Die Gründe dafür sind Epistasie-ähnliche Effekte zwischen solchen Bits.

Wenn eine solche Bitgruppe auf dem Chromosom eng beieinander kodiert wäre, so dass die Wahrscheinlichkeit durch Crossover getrennt zu werden sich verringert, dann könnte der genetische Algorithmus möglicherweise bessere Kandidaten finden.

Dies bezeichnet man als das *linkage problem*. Man möchte also, dass funktional zusammengehörige Bits eng beeinander liegen um nicht durch Crossover getrennt zu werden. Auf der anderen Seite weiss man am Anfang überhaupt nicht, welche Bits funktional miteinander zusammenhängen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Da wir uns ja sowieso mit genetischen Algorithmen befassen ist der Gedanke naheliegend, die Kodierung eines genetischen Algorithmus durch den genetischen Algorithmus selbst verbessern zu lassen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Da wir uns ja sowieso mit genetischen Algorithmen befassen ist der Gedanke naheliegend, die Kodierung eines genetischen Algorithmus durch den genetischen Algorithmus selbst verbessern zu lassen.

J. Holland schlug hierzu einen *Inversionsoperator (inversion operator)* vor:

Analog dazu, wie in der Biologie die Funktion eines Gens oftmals unabhängig davon ist, wo genau das Gen auf dem Chromosom kodiert wird, könnte man auch bei einem genetischen Algorithmus bestimmte Teile eines Chromosoms umdrehen, so dass im Laufe der Evolution einzelne Bits auch ihre Position auf dem Chromosom ändern können.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Da wir uns ja sowieso mit genetischen Algorithmen befassen ist der Gedanke naheliegend, die Kodierung eines genetischen Algorithmus durch den genetischen Algorithmus selbst verbessern zu lassen.

J. Holland schlug hierzu einen *Inversionsoperator (inversion operator)* vor:

Analog dazu, wie in der Biologie die Funktion eines Gens oftmals unabhängig davon ist, wo genau das Gen auf dem Chromosom kodiert wird, könnte man auch bei einem genetischen Algorithmus bestimmte Teile eines Chromosoms umdrehen, so dass im Laufe der Evolution einzelne Bits auch ihre Position auf dem Chromosom ändern können.

Dabei ergibt sich ein kleines Problem:

Vorher	*10010*
Nachher	*10010*

Woran sieht man, dass das grüne und das rote Bit getauscht wurden ?

Kodierung von Lösungskandidaten

Adaptive Kodierung

Jedes Bit wird mit einem Index versehen, der angibt welches die Originalposition des Bits ist. Dadurch wird es unabhängig von seiner aktuellen Position im Chromosom.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Jedes Bit wird mit einem Index versehen, der angibt welches die Originalposition des Bits ist. Dadurch wird es unabhängig von seiner aktuellen Position im Chromosom.

Beispiel

Original	1 0 1 1 0 1 0 0
Index	1 2 3 4 5 6 7 8

Dieses Chromosom könnte dann so kodiert werden:

{(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0)}

Kodierung von Lösungskandidaten

Adaptive Kodierung

Genauso gut wäre aber auch folgende Permutation, welche dadurch entstanden ist, dass die Bits 4 bis 7 durch den Inversionssoperator umgeordnet wurden:

Vorher	*{(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0)}*
Nachher	*{(1,1) (2,0) (3,1) (7,0) (6,1) (5,0) (4,1) (8,0)}*

Kodierung von Lösungskandidaten

Adaptive Kodierung

Genauso gut wäre aber auch folgende Permutation, welche dadurch entstanden ist, dass die Bits 4 bis 7 durch den Inversionssoperator umgeordnet wurden:

Vorher {(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0)}

Nachher {(1,1) (2,0) (3,1) (7,0) (6,1) (5,0) (4,1) (8,0)}

Was haben wir damit erreicht ? Angenommen, es gäbe in unserem Beispiel drei Bits, die in einem wichtigen funktionalen Zusammenhang stehen:

Vorher 1 0 1 1 0 1 0 0

Kodierung von Lösungskandidaten

Adaptive Kodierung

Genauso gut wäre aber auch folgende Permutation, welche dadurch entstanden ist, dass die Bits 4 bis 7 durch den Inversionssoperator umgeordnet wurden:

Vorher {(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0)}

Nachher {(1,1) (2,0) (3,1) (7,0) (6,1) (5,0) (4,1) (8,0)}

Was haben wir damit erreicht ? Angenommen, es gäbe in unserem Beispiel drei Bits, die in einem wichtigen funktionalen Zusammenhang stehen:

Vorher 1 0 1 1 0 1 0 0

Durch den Inversionsoperator wurden die Bits nun so umgeordnet, dass sie enger beieinander sind:

Nachher 1 0 1 0 1 0 1 0

Kodierung von Lösungskandidaten

Adaptive Kodierung

Genauso gut wäre aber auch folgende Permutation, welche dadurch entstanden ist, dass die Bits 4 bis 7 durch den Inversionssoperator umgeordnet wurden:

Vorher {(1,1) (2,0) (3,1) (4,1) (5,0) (6,1) (7,0) (8,0)}

Nachher {(1,1) (2,0) (3,1) (7,0) (6,1) (5,0) (4,1) (8,0)}

Was haben wir damit erreicht ? Angenommen, es gäbe in unserem Beispiel drei Bits, die in einem wichtigen funktionalen Zusammenhang stehen:

Vorher 1 0 1 1 0 1 0 0

Durch den Inversionsoperator wurden die Bits nun so umgeordnet, dass sie enger beieinander sind:

Nachher 1 0 1 0 1 0 1 0

Damit ist die Wahrscheinlichkeit, dass diese Bitgruppe eine Crossover-Operation überlebt stark angestiegen, und somit insgesamt die Wahrscheinlichkeit, dass eine gute Lösung gefunden wird.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Bei dieser Kodierung gibt es allerdings bei dem Crossover auch ein Problem:

Vorher	*1 0 1 1*	*0 1 0 1*
Vorher	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*

Kodierung von Lösungskandidaten

Adaptive Kodierung

Bei dieser Kodierung gibt es allerdings bei dem Crossover auch ein Problem:

Vorher	*1 0 1 1*	*0 1 0 1*
Vorher	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Links	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Rechts	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Nachher	*{(1,1) (1,0) (2,1) (4,1)}*	*{(3,0) (2,0) (4,1) (3,1)}*

Das linke Ergebnis hat nun zwei Bits für Position 1, aber keines für Position 3. Das Rechte hat kein Bit für Position 1, aber zwei Bits für Position 3.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Bei dieser Kodierung gibt es allerdings bei dem Crossover auch ein Problem:

Vorher	*1 0 1 1*	*0 1 0 1*
Vorher	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Links	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Rechts	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Nachher	*{(1,1) (1,0) (2,1) (4,1)}*	*{(3,0) (2,0) (4,1) (3,1)}*

Das linke Ergebnis hat nun zwei Bits für Position 1, aber keines für Position 3. Das Rechte hat kein Bit für Position 1, aber zwei Bits für Position 3.

Holland schlug für solche Situationen zwei Lösungsalternativen vor:

- Crossover nur bei gleicher Permutation durchzuführen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Bei dieser Kodierung gibt es allerdings bei dem Crossover auch ein Problem:

Vorher	*1 0 1 1*	*0 1 0 1*
Vorher	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Links	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Rechts	*{(1,1) (2,0) (4,1) (3,1)}*	*{(3,0) (1,0) (2,1) (4,1)}*
Nachher	*{(1,1) (1,0) (2,1) (4,1)}*	*{(3,0) (2,0) (4,1) (3,1)}*

Das linke Ergebnis hat nun zwei Bits für Position 1, aber keines für Position 3. Das Rechte hat kein Bit für Position 1, aber zwei Bits für Position 3.

Holland schlug für solche Situationen zwei Lösungsalternativen vor:

- Crossover nur bei gleicher Permutation durchzuführen.
- Master/Slave-Prinzip: Ein Chromosom ist das Master-Chromosom. Das Slave-Chromosoms übernimmt vor dem Crossover die Permutation des Masters-Chromosoms und wird danach wieder zurück permutiert.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Schaffer und *Morishima* schlugen 1987 einen anderen Ansatz vor, der sich mehr an die Natur anlehnt.

In der Tat ist es so, dass Crossovers nicht an allen Stellen gleich wahrscheinlich auftreten, sondern es haben sich im Laufe der biologischen Evolution *hot spots* gebildet, an denen ein Crossover wahrscheinlicher ist, als an anderen Stellen auf dem Chromosom.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Schaffer und *Morishima* schlugen 1987 einen anderen Ansatz vor, der sich mehr an die Natur anlehnt.

In der Tat ist es so, dass Crossovers nicht an allen Stellen gleich wahrscheinlich auftreten, sondern es haben sich im Laufe der biologischen Evolution *hot spots* gebildet, an denen ein Crossover wahrscheinlicher ist, als an anderen Stellen auf dem Chromosom.

Die Positionen an denen ein Crossover erlaubt und nützlich ist, können sich ja ebenfalls durch den genetischen Algorithmus entwickeln.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Schaffer und *Morishima* schlugen 1987 einen anderen Ansatz vor, der sich mehr an die Natur anlehnt.

In der Tat ist es so, dass Crossovers nicht an allen Stellen gleich wahrscheinlich auftreten, sondern es haben sich im Laufe der biologischen Evolution *hot spots* gebildet, an denen ein Crossover wahrscheinlicher ist, als an anderen Stellen auf dem Chromosom.

Die Positionen an denen ein Crossover erlaubt und nützlich ist, können sich ja ebenfalls durch den genetischen Algorithmus entwickeln.

In der Ausrufezeichen-Schreibweise *(exclamation mark notation)* wird hinter denjenigen Bits des Lösungskandidaten, an denen ein Crossover möglich ist, ein Ausrufezeichen angehängt.

Beispiel

```
1101!110!1
```

Kodierung von Lösungskandidaten

Adaptive Kodierung

Bei einer solchen Kodierung ist dann ein Crossover auch an mehreren Stellen möglich. Die Markierungen werden mitvererbt.

Beispiel

Vorher	1101!110!1	010000!10
Fragmente	1101! 11 0! 1	0100 00! 1 0
Crossover	1101! 00! 0! 0	0100 11 1 1
Nachher	1101!00!0!0	01001111

- In diesem Beispiel gibt es insgesamt drei Markierungen, also werden die Chromosomen an allen drei Positionen aufgespalten und ergeben somit jeweils vier Fragmente.
- In diesem Beispiel bekommt das linke Chromosom auch die Markierung des rechten Chromosoms, das somit nach dem Crossover keine einzige Markierung mehr hat.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Ein weiterer adaptiver Kodierungsansatz wurde 1989 von *Goldberg* et al. vorgeschlagen, welcher unter dem Namen *messy GA* bekannt wurde.

> *messy = durcheinander, unordentlich, chaotisch*

Kodierung von Lösungskandidaten

Adaptive Kodierung

Ein weiterer adaptiver Kodierungsansatz wurde 1989 von *Goldberg* et al. vorgeschlagen, welcher unter dem Namen *messy GA* bekannt wurde.

> *messy = durcheinander, unordentlich, chaotisch*

Er basiert auf der Erkenntnis, dass die Natur ja auch nicht mit Chomosomen mit über 3 Milliarden Basenpaaren angefangen hat, Menschen zu produzieren, sondern dass das genetische Material für neue, komplexere Lebewesen auf der Basis des bereits vorhandenen genetischen Materials von einfacheren Lebewesen entwickelt wurde.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Ein weiterer adaptiver Kodierungsansatz wurde 1989 von *Goldberg* et al. vorgeschlagen, welcher unter dem Namen *messy GA* bekannt wurde.

> *messy = durcheinander, unordentlich, chaotisch*

Er basiert auf der Erkenntnis, dass die Natur ja auch nicht mit Chomosomen mit über 3 Milliarden Basenpaaren angefangen hat, Menschen zu produzieren, sondern dass das genetische Material für neue, komplexere Lebewesen auf der Basis des bereits vorhandenen genetischen Materials von einfacheren Lebewesen entwickelt wurde.

> *Ein Lösungskandidat muss am Anfang gar nicht notwendigerweise alle Gene bereits kennen, die letztlich zu einer guten Lösung führen (unterspezifiziert), oder er besitzt von einem Gen manchmal mehrere verschiedene Allele (überspezifiziert).*

Kodierung von Lösungskandidaten

Adaptive Kodierung

Ein weiterer adaptiver Kodierungsansatz wurde 1989 von *Goldberg* et al. vorgeschlagen, welcher unter dem Namen *messy GA* bekannt wurde.

> *messy = durcheinander, unordentlich, chaotisch*

Er basiert auf der Erkenntnis, dass die Natur ja auch nicht mit Chomosomen mit über 3 Milliarden Basenpaaren angefangen hat, Menschen zu produzieren, sondern dass das genetische Material für neue, komplexere Lebewesen auf der Basis des bereits vorhandenen genetischen Materials von einfacheren Lebewesen entwickelt wurde.

> *Ein Lösungskandidat muss am Anfang gar nicht notwendigerweise alle Gene bereits kennen, die letztlich zu einer guten Lösung führen (unterspezifiziert), oder er besitzt von einem Gen manchmal mehrere verschiedene Allele (überspezifiziert).*

Damit wird also mit unserer bisherigen Annahme, dass auf einem Chromosom jedes Gen genau einmal vorhanden sein muss, gebrochen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Beispiel

Betrachten Sie folgende beiden 4-Bit Chromosome eines solchen genetischen Algorithmus in der Index-Notation:

Chromosom A $\{(1,1)\ (3,1)\ (3,0)\ (4,1)\}$

Chromosom B $\{(2,1)\ (2,0)\ (3,0)\ (3,1)\ (2,0)\ (2,1)\}$

- Bei Chromosom A ist das Gen Nr. 2 gar nicht vorhanden, also *unterspezifiziert*, während es für das Gen Nr. 3 gleich zwei verschiedene Allele gibt, es also überspezifiziert ist.
- Ebenso fehlen bei Chromosom B die Gene Nr. 1 und Nr. 3, dafür ist das Gen Nr. 3 doppelt und Nr. 2 sogar vierfach *überspezifiziert*.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Beispiel

Betrachten Sie folgende beiden 4-Bit Chromosome eines solchen genetischen Algorithmus in der Index-Notation:

Chromosom A	*{(1,1) (3,1) (3,0) (4,1)}*
Chromosom B	*{(2,1) (2,0) (3,0) (3,1) (2,0) (2,1)}*

- Bei Chromosom A ist das Gen Nr. 2 gar nicht vorhanden, also *unterspezifiziert*, während es für das Gen Nr. 3 gleich zwei verschiedene Allele gibt, es also überspezifiziert ist.
- Ebenso fehlen bei Chromosom B die Gene Nr. 1 und Nr. 3, dafür ist das Gen Nr. 3 doppelt und Nr. 2 sogar vierfach *überspezifiziert*.

Jetzt wissen Sie, warum genetische Algorithmen mit einer solchen Kodierung als unordentlich (messy) bezeichnet werden. Welches Problem ergibt sich aber dabei ?

Kodierung von Lösungskandidaten

Adaptive Kodierung

Problem: Wie wird die Fitness eines solchen unordentlichen Chromosoms berechnet ?

Kodierung von Lösungskandidaten

Adaptive Kodierung

Problem: Wie wird die Fitness eines solchen unordentlichen Chromosoms berechnet ?

Bei einer Überspezifizierung gilt *first-come-first serve*.

Beispiel

Chromosom A *{(1,1) (3,1) (3,0) (4,1)}*

Hierbei gilt also Gen Nr. 3 auf eins gesetzt. Das nachfolgende Allel (3,0) wird ignoriert.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Problem: Wie wird die Fitness eines solchen unordentlichen Chromosoms berechnet ?

Bei einer Überspezifizierung gilt *first-come-first serve*.

Beispiel

Chromosom A *{(1,1) (3,1) (3,0) (4,1)}*

Hierbei gilt also Gen Nr. 3 auf eins gesetzt. Das nachfolgende Allel (3,0) wird ignoriert.

Eine Unterspezifizierung besagt das ein oder mehrere Gene auf dem Chromosom fehlen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Problem: Wie wird die Fitness eines solchen unordentlichen Chromosoms berechnet ?

Bei einer Überspezifizierung gilt *first-come-first serve*.

Beispiel

Chromosom A *{(1,1) (3,1) (3,0) (4,1)}*

Hierbei gilt also Gen Nr. 3 auf eins gesetzt. Das nachfolgende Allel (3,0) wird ignoriert.

Eine Unterspezifizierung besagt das ein oder mehrere Gene auf dem Chromosom fehlen.

Ein erster Ansatz war es, für die fehlenden Gene Zufallswerte anzunehmen und danach die Fitness zu berechnen. Wenn man das mehrfach wiederholt, kann man auf diese Art und Weise die durchschnittliche Fitness dieses Chromosoms ermitteln.

Dieser Ansatz wurde allerdings aufgrund zu großer Varianzen wieder verworfen.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Goldberg schlug noch einen anderen Ansatz zur Berechnung der Fitness von unterspezifizierten Chromosomen vor:

Kodierung von Lösungskandidaten

Adaptive Kodierung

Goldberg schlug noch einen anderen Ansatz zur Berechnung der Fitness von unterspezifizierten Chromosomen vor:

- Hierzu muss zuerst irgendwie ein lokales Maximum berechnet werden. Ein lokales Maximum zeichnet sich dadurch aus, dass jede *Einzelbitänderung* zu einer Verschlechterung der Fitness führt.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Goldberg schlug noch einen anderen Ansatz zur Berechnung der Fitness von unterspezifizierten Chromosomen vor:

- Hierzu muss zuerst irgendwie ein lokales Maximum berechnet werden. Ein lokales Maximum zeichnet sich dadurch aus, dass jede *Einzelbitänderung* zu einer Verschlechterung der Fitness führt.
- Bei der Berechnung der Fitness eines unterspezifizierten Chromosoms werden bei den fehlenden Genen die Allele des lokalen Maximums verwendet.

Kodierung von Lösungskandidaten

Adaptive Kodierung

Goldberg schlug noch einen anderen Ansatz zur Berechnung der Fitness von unterspezifizierten Chromosomen vor:

- Hierzu muss zuerst irgendwie ein lokales Maximum berechnet werden. Ein lokales Maximum zeichnet sich dadurch aus, dass jede *Einzelbitänderung* zu einer Verschlechterung der Fitness führt.
- Bei der Berechnung der Fitness eines unterspezifizierten Chromosoms werden bei den fehlenden Genen die Allele des lokalen Maximums verwendet.
- Führt dies zu einer Fitness die höher als die des lokalen Maximums ist, dann liegt offensichtlich ein guter Lösungskandidat vor. Gegenüber dem lokalen Maximum muss also *mehr* als ein Bit verändert worden sein.

Kodierung von Lösungskandidaten

Epistasie

Epistasie lässt sich gut an dem Problem des Handlungsreisenden zeigen.

- *Bei dem Problem des Handlungsreisenden soll ein Handlungsreisender eine Rundreise durch verschiedene Städte machen und danach wieder zu seinem Ausgangspunkt zurückkehren. Jede Stadt darf nur ein einziges Mal besucht werden.*
- *Biologie: Wenn ein Gen am Anfang einer biochemischen Reaktionskette defekt ist, dann werden die nachfolgenden Gene irrelevant, weil ja ihr Ausgangsstoff fehlt. Ihre Wirkung wird also durch das defekte Gen am Anfang der Kette maskiert. Diesen Effekt nennt man Epistasie.*

Kodierung von Lösungskandidaten

Epistasie

Epistasie lässt sich gut an dem Problem des Handlungsreisenden zeigen.

- *Bei dem Problem des Handlungsreisenden soll ein Handlungsreisender eine Rundreise durch verschiedene Städte machen und danach wieder zu seinem Ausgangspunkt zurückkehren. Jede Stadt darf nur ein einziges Mal besucht werden.*
- *Biologie: Wenn ein Gen am Anfang einer biochemischen Reaktionskette defekt ist, dann werden die nachfolgenden Gene irrelevant, weil ja ihr Ausgangsstoff fehlt. Ihre Wirkung wird also durch das defekte Gen am Anfang der Kette maskiert. Diesen Effekt nennt man Epistasie.*

Bei genetischen Algorithmen bedeutet Epistasie, dass die Wirkung eines Gens die Wirkung anderer Gene stark überlagert oder überdeckt.

Ein solchermassen betroffenes Gen ist also nicht mehr unabhängig, sondern sein Einfluss auf die Fitness des Lösungskandidaten ist stark von der konkreten Ausprägung (Allele) des überdeckenden („epistatischen“) Gens abhängig.

Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

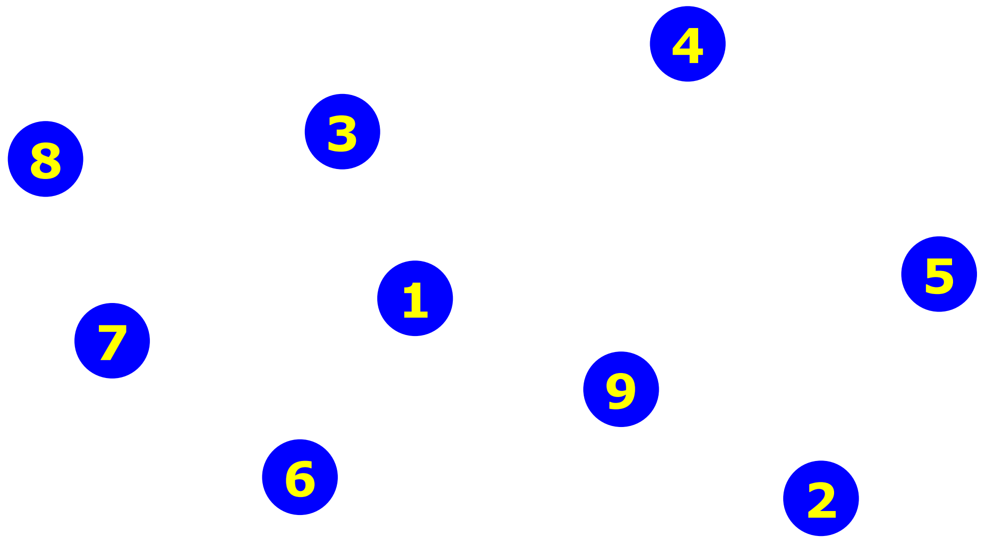
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung* *(path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

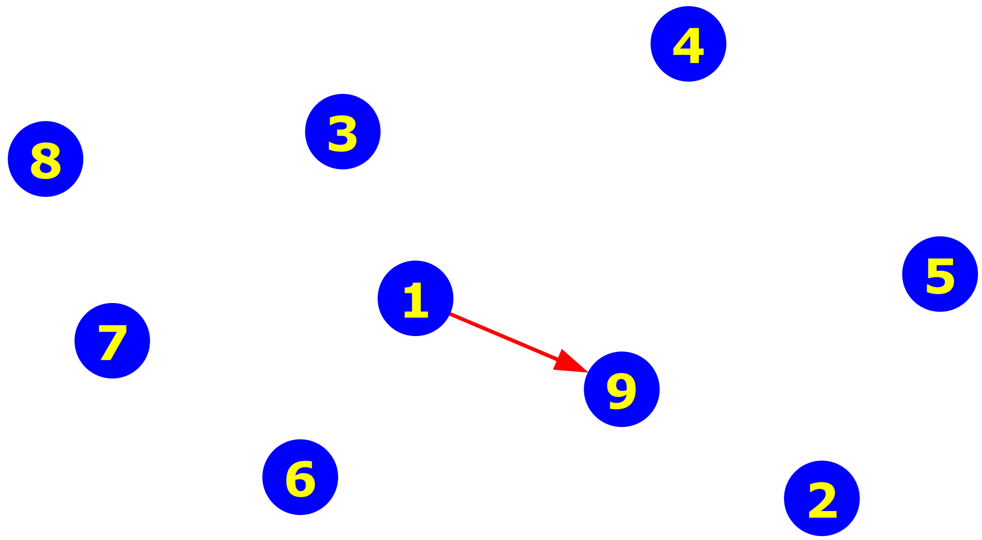
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

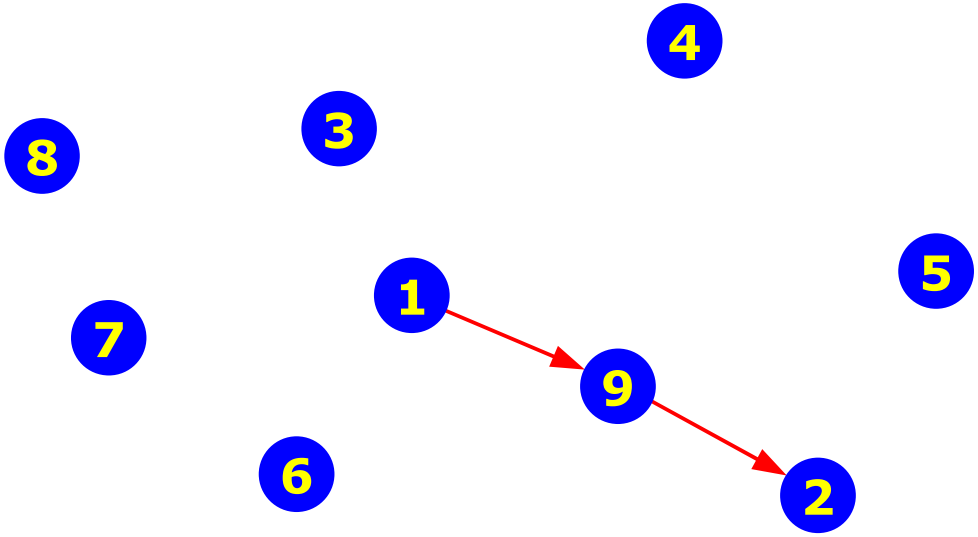
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

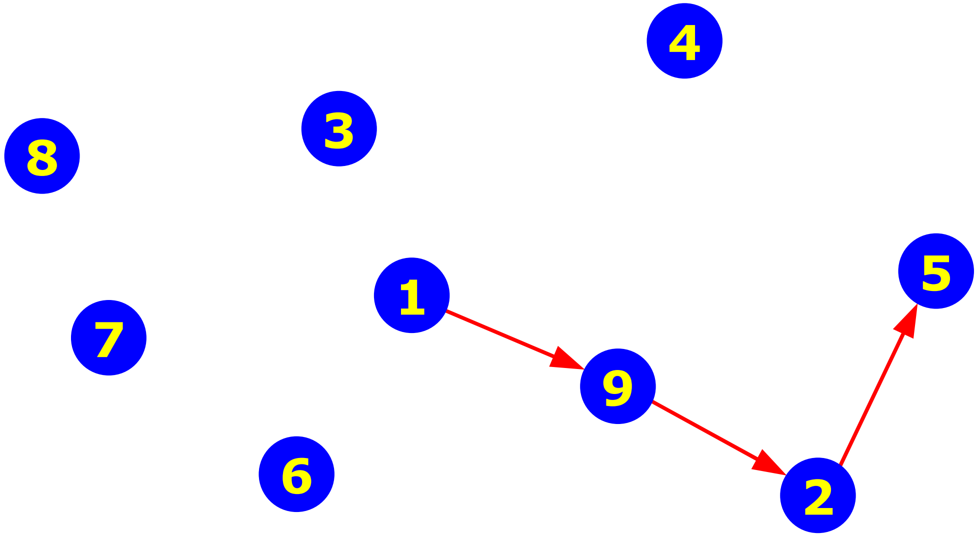
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

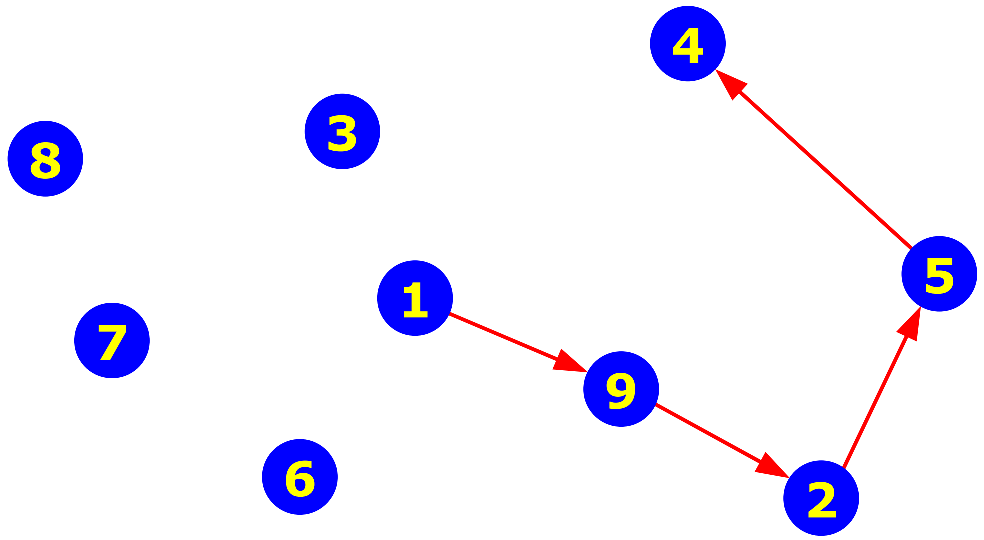
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

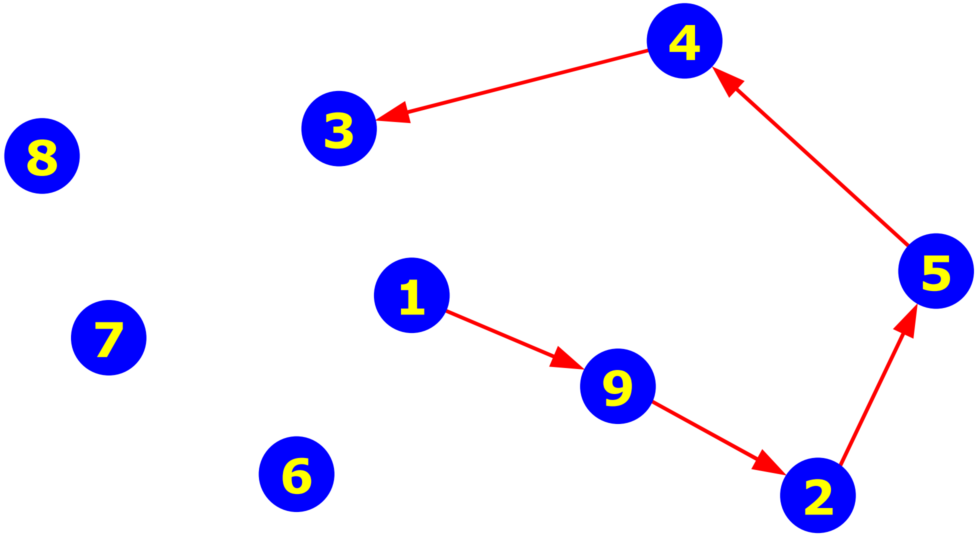
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

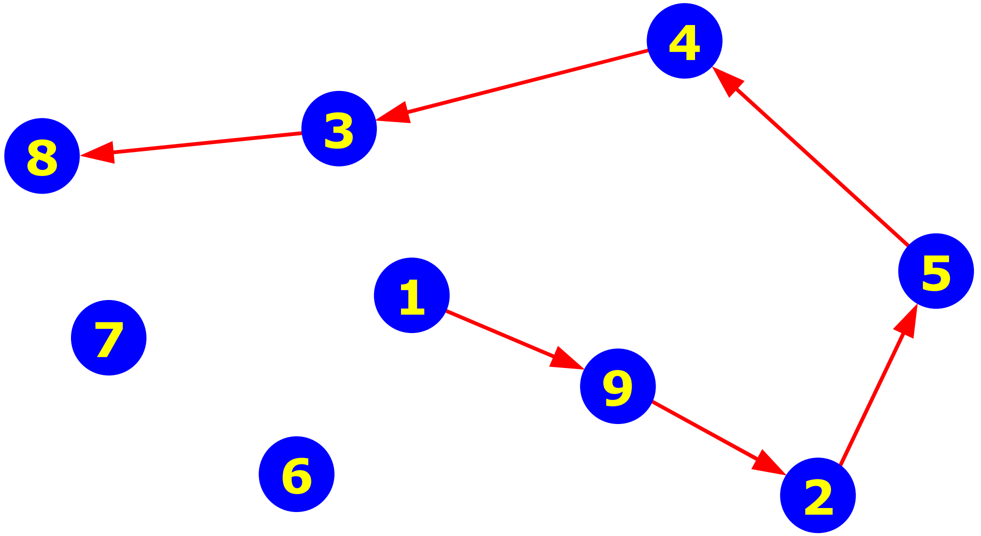
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

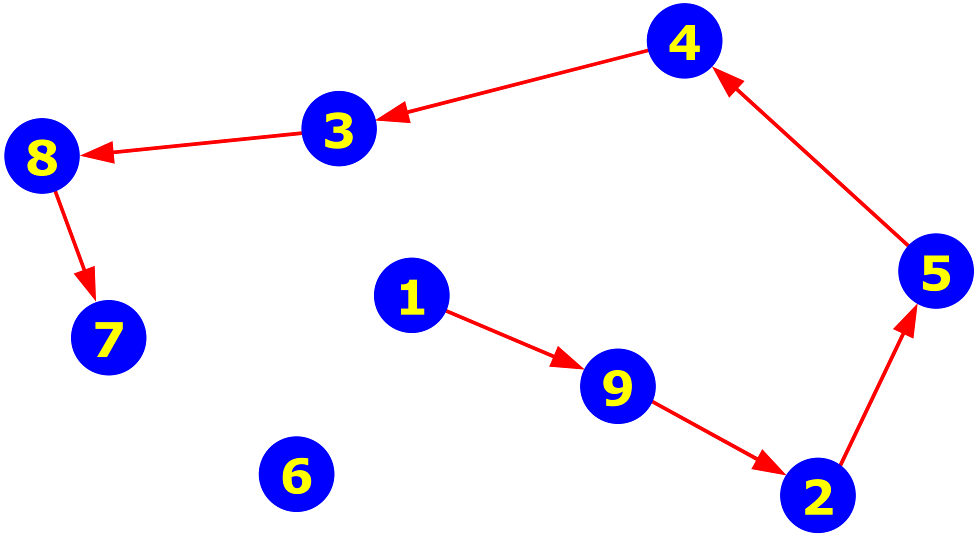
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung (path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

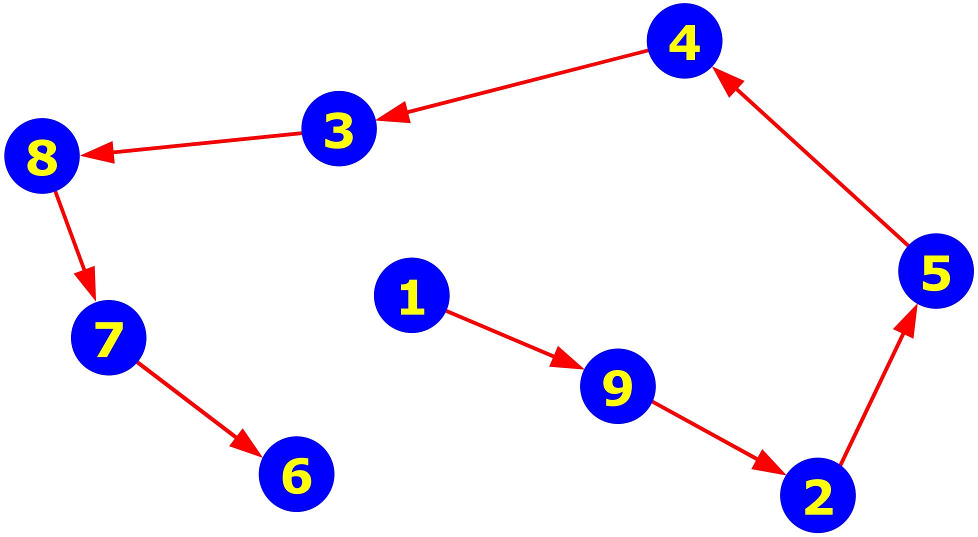
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung* *(path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Problem des Handlungsreisenden, Pfadkodierung

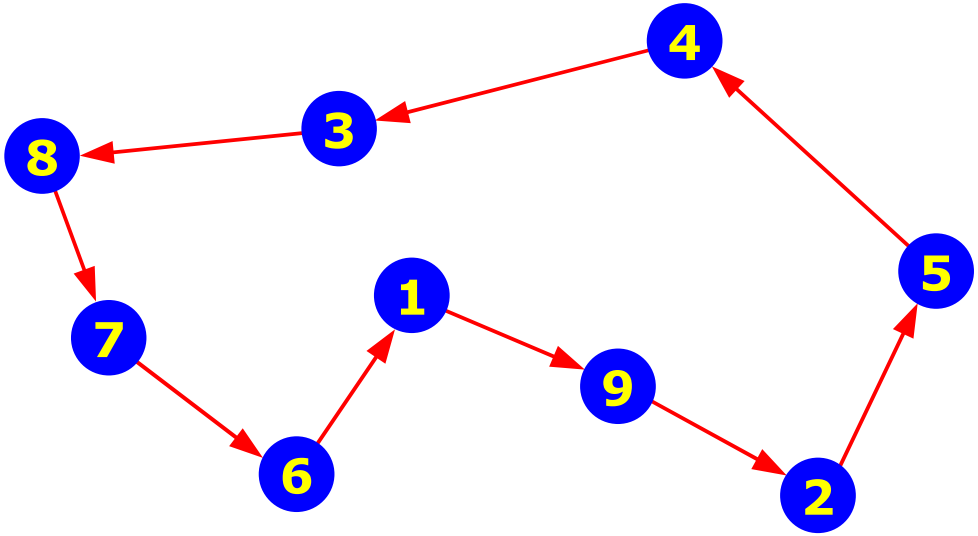
Ein genetischer Algorithmus soll nun dazu verwendet werden, bei bekannten Entfernungen zwischen den Städten die kürzeste Rundreise zu finden. Ein Lösungskandidat stellt im Phänotyp also eine ringförmige Liste von Städten dar.

Im Folgenden untersuchen wir drei Möglichkeiten wie der Genotyp zu einer Rundreise kodiert werden kann, und inwieweit dabei Epistasie auftritt.

Die naheliegendste Variante ist die *Pfadkodierung* *(path representation)*, wobei die Städte in genau der Reihenfolge der Rundreise kodiert werden.

Beispiel

Kodierung (1 9 2 5 4 3 8 7 6)



Kodierung von Lösungskandidaten

Kantenkodierung

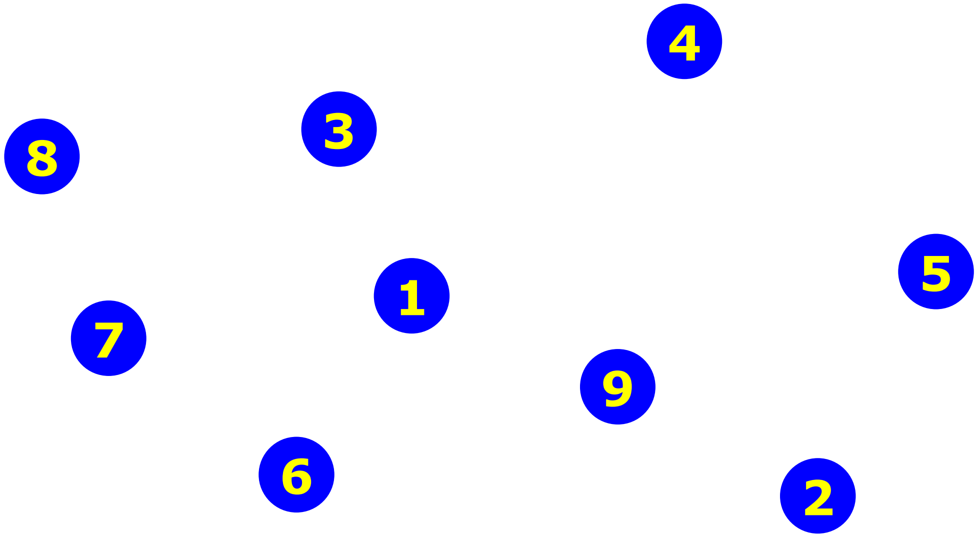
Die zweite Variante ist die *Kantenkodierung* *(adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 4 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

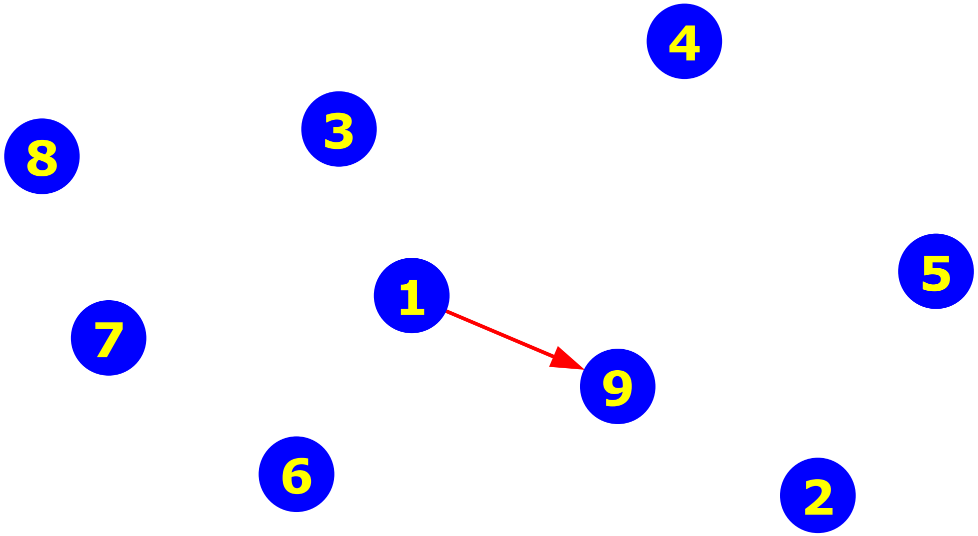
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (**9** 5 8 3 4 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

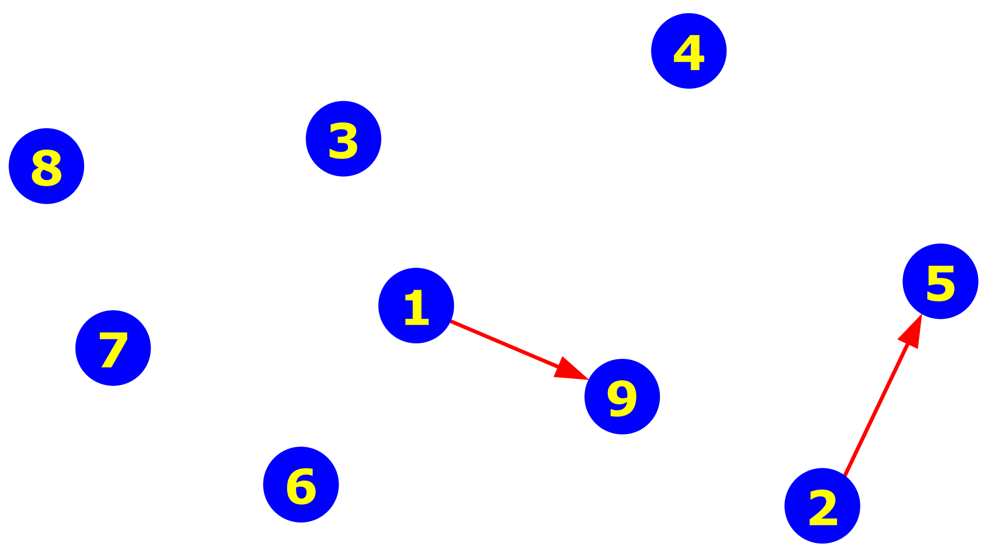
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 **5** 8 3 4 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

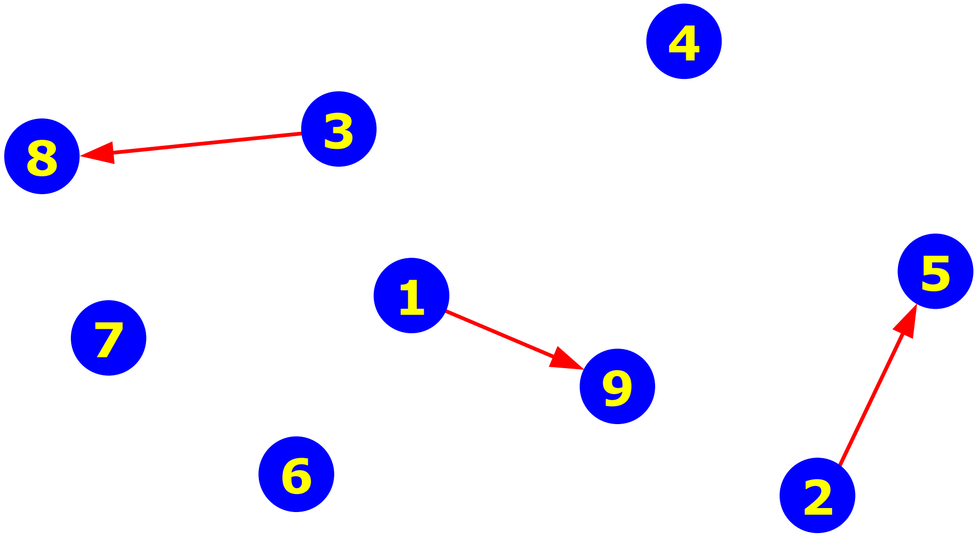
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 **8** 3 4 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

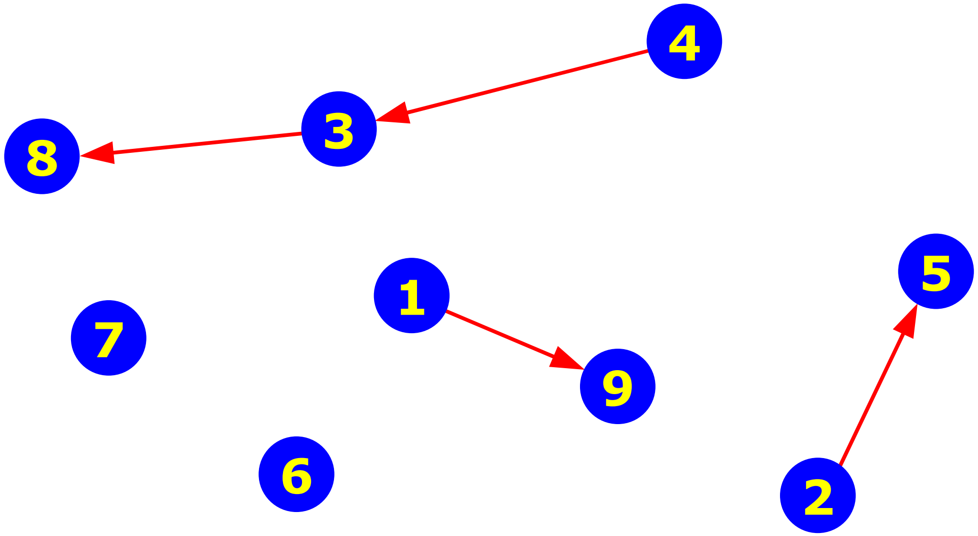
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 **3** 4 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

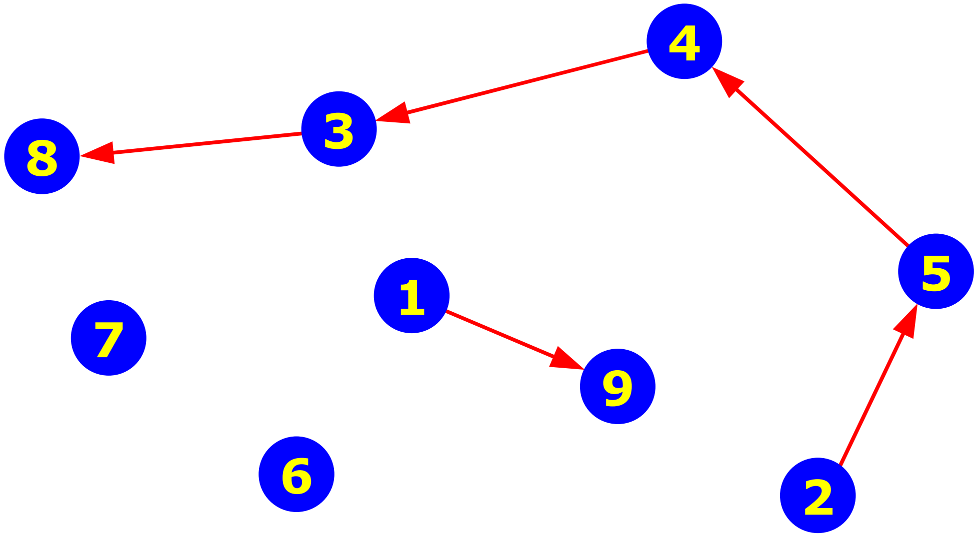
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 **4** 1 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

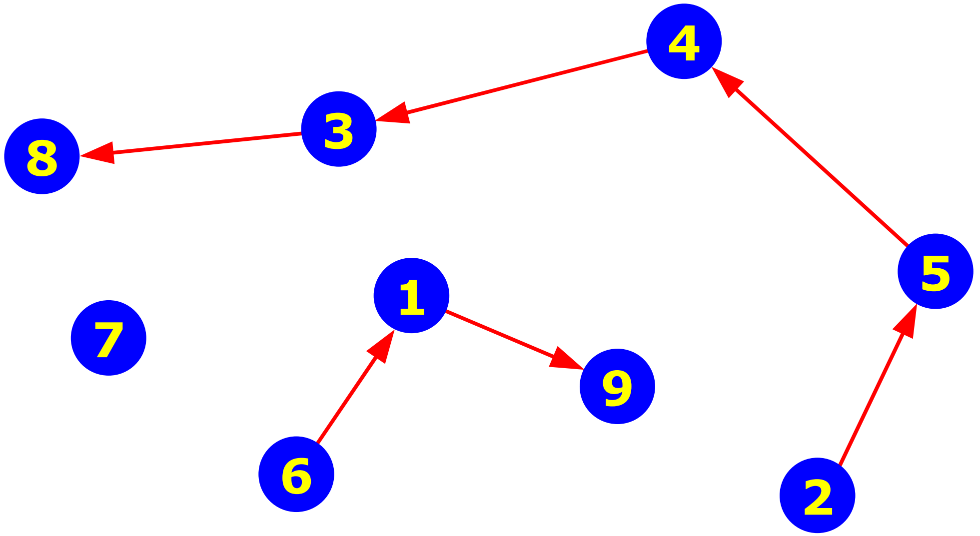
Die zweite Variante ist die *Kantenkodierung* *(adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 4 **1** 6 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

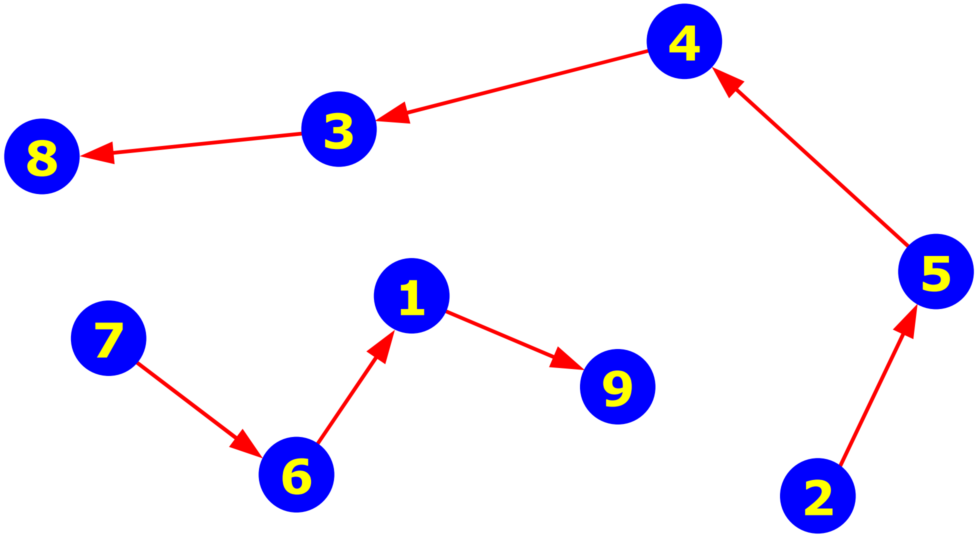
Die zweite Variante ist die *Kantenkodierung (adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 4 1 **6** 7 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

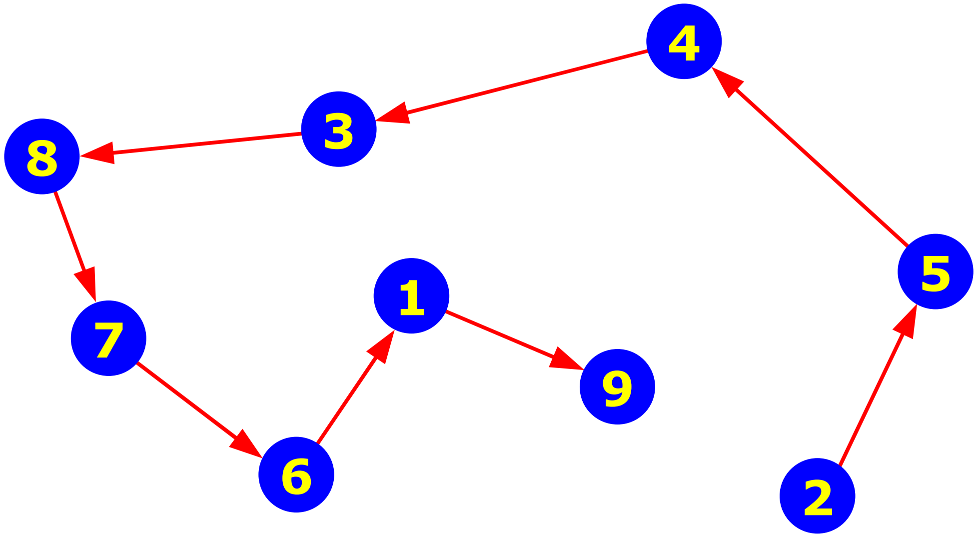
Die zweite Variante ist die *Kantenkodierung* *(adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 4 1 6 **7** 2)

Rundreise



Kodierung von Lösungskandidaten

Kantenkodierung

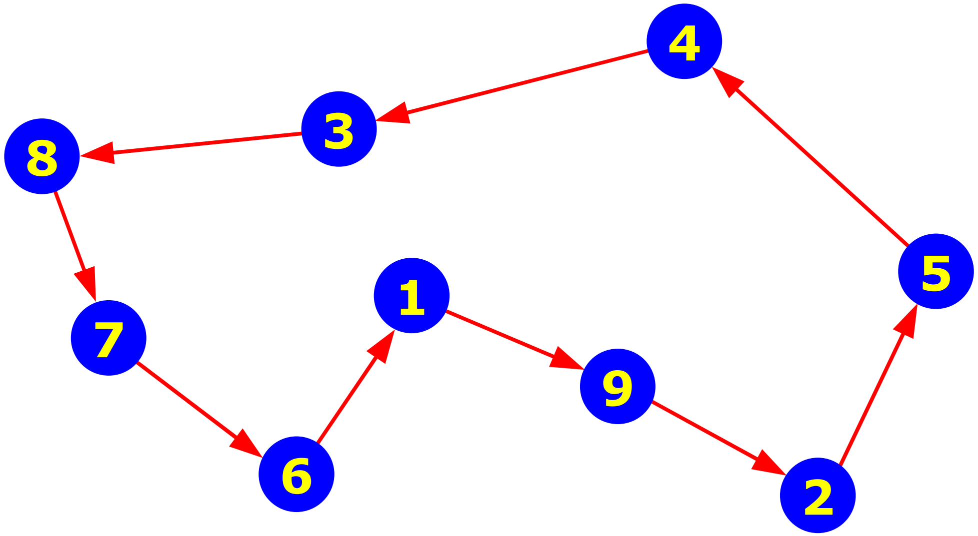
Die zweite Variante ist die *Kantenkodierung* *(adjacency representation)*.

Hierbei wird eine Rundreise durch eine Liste von *n* Städten dargestellt. Die *j*-te Stadt befindet sich in dieser Liste an der *i*-ten Position, wenn die Rundreise direkt von der Stadt *i* zur Stadt *j* führt.

Beispiel

Kodierung (9 5 8 3 4 1 6 7 **2**)

Rundreise



Kodierung von Lösungskandidaten

Ordinalkodierung

Die dritte Variante ist die *Ordinalkodierung* *(ordinal represenation)*.

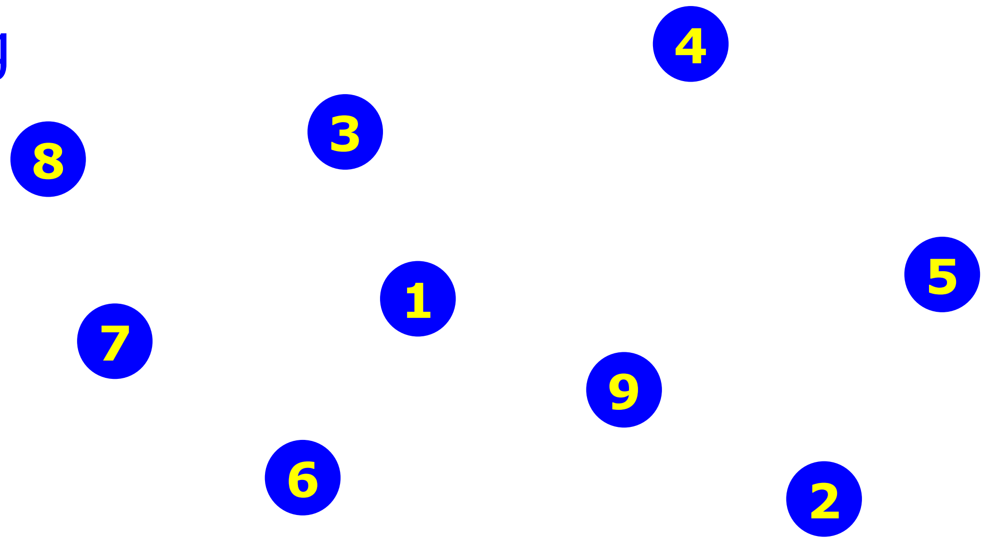
Auch hier wird eine Rundreise als eine Liste von n Städten dargestellt.

Bei dieser Kodierung geht man davon aus, dass eine *Referenz-Rundreise* vorhanden ist, z.B. $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$.

- Die Kodierung einer Rundreise wird in schrittweise aufgebaut.
- In jedem Schritt notiert man die Position der betreffenden Stadt in der aktuellen Referenz-Rundreise und entfernt diese Stadt anschließend aus der Referenz-Rundreise.

Kodierung von Lösungskandidaten

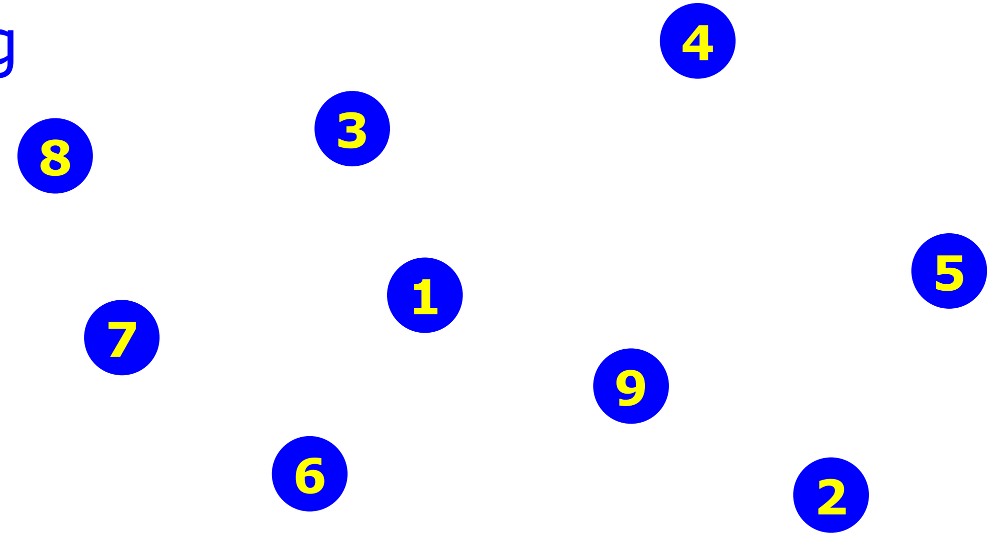
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	

Kodierung von Lösungskandidaten

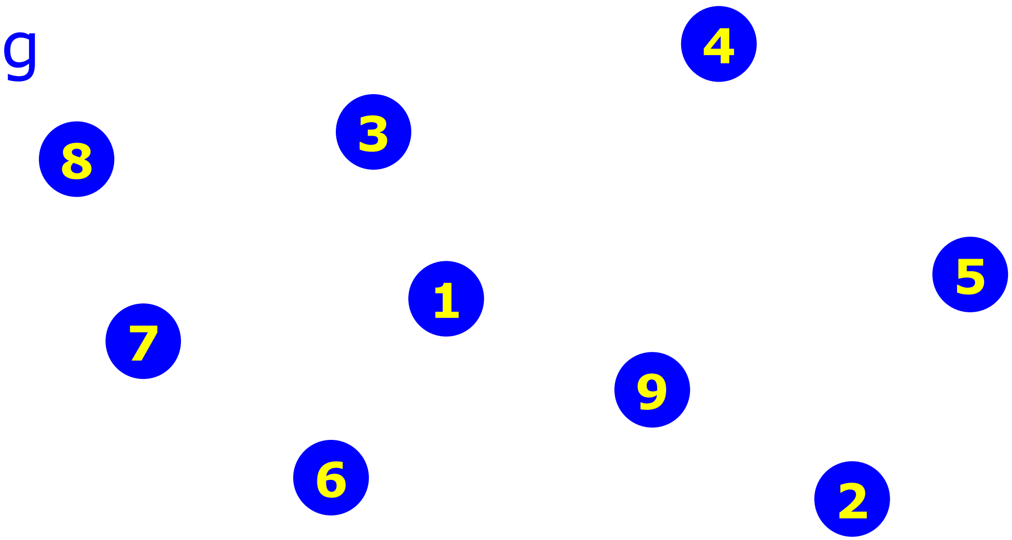
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)

Kodierung von Lösungskandidaten

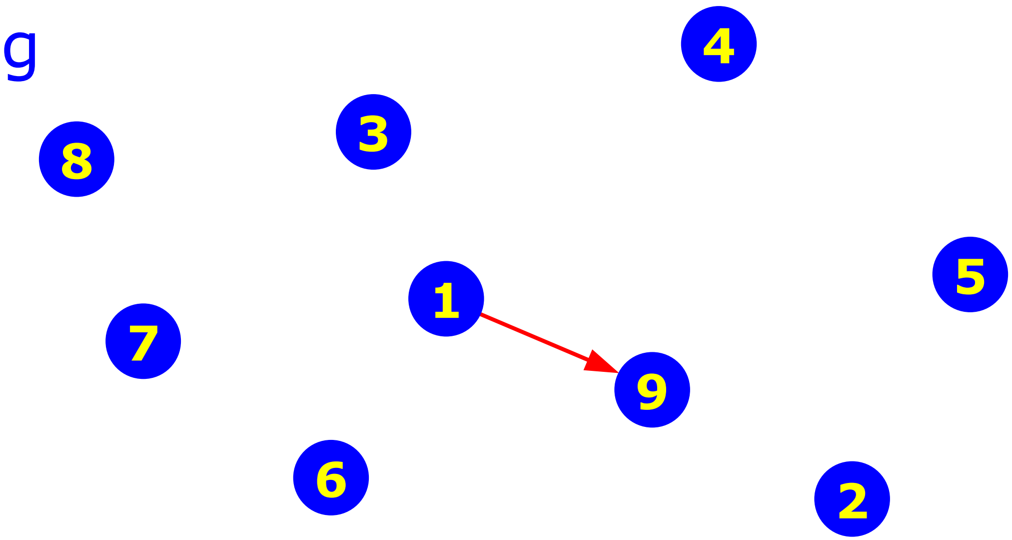
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	

Kodierung von Lösungskandidaten

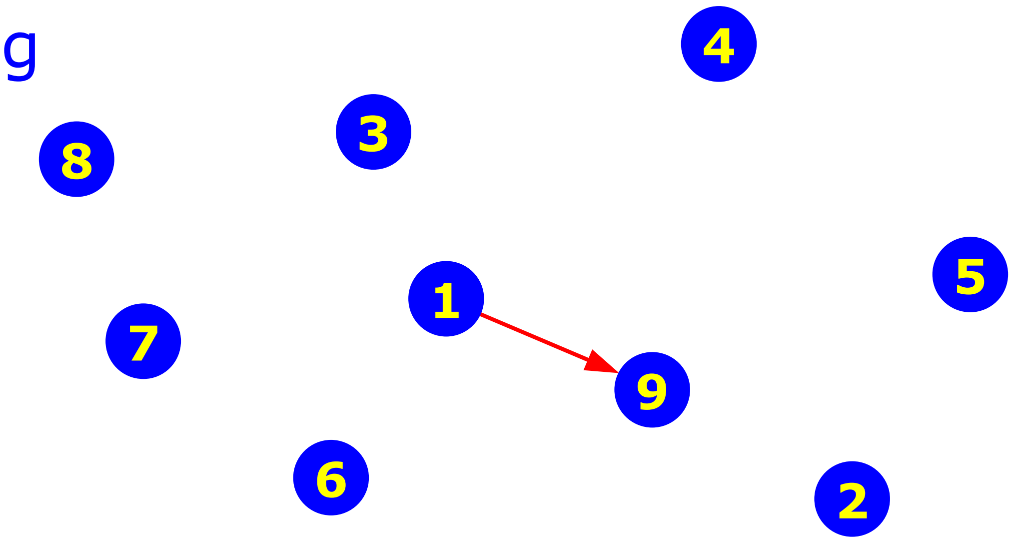
Beispiel zur Ordinalkodierung



Stadt	**Referenz-Rundreise**	**Kodierung**
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	

Kodierung von Lösungskandidaten

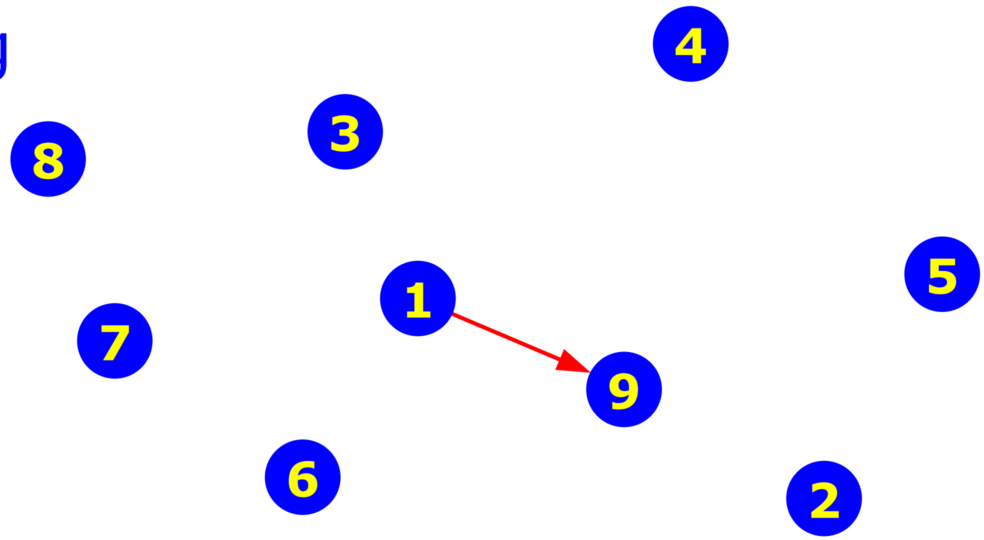
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)

Kodierung von Lösungskandidaten

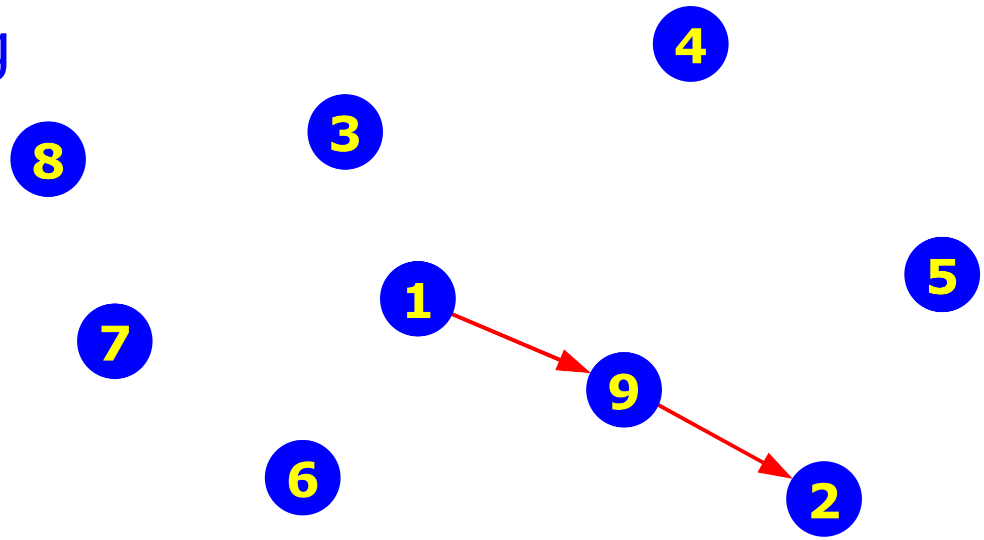
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	

Kodierung von Lösungskandidaten

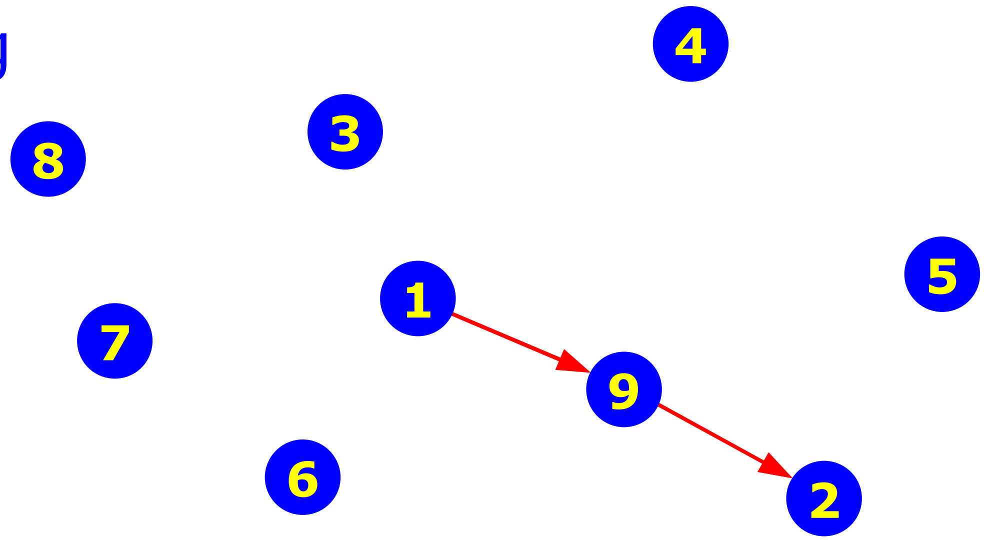
Beispiel zur Ordinalkodierung



Stadt	**Referenz-Rundreise**	**Kodierung**
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	

Kodierung von Lösungskandidaten

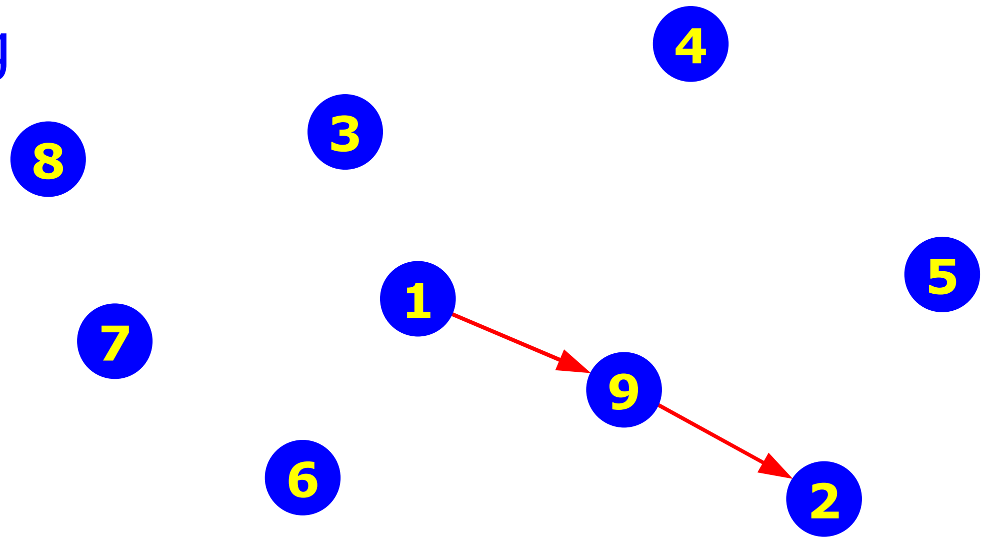
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)

Kodierung von Lösungskandidaten

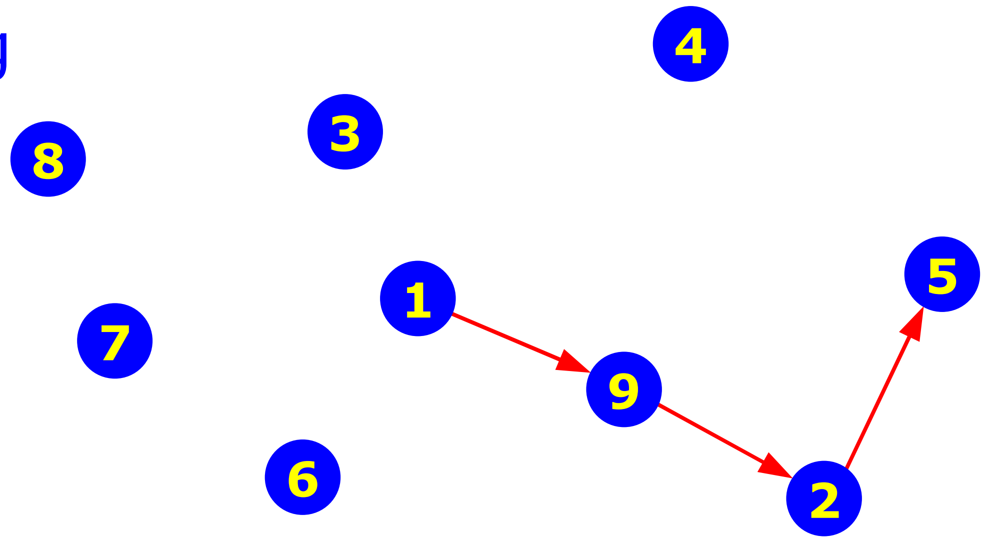
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

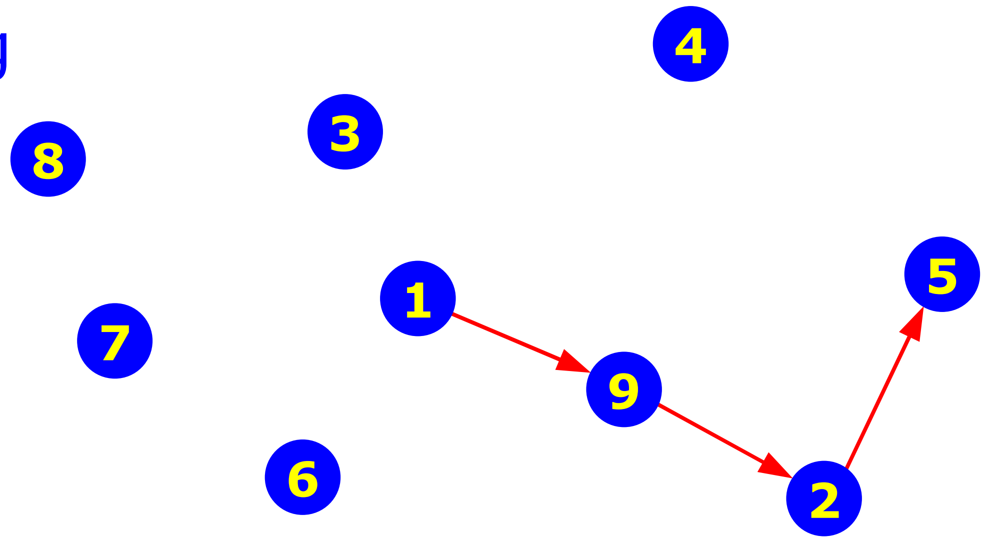
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

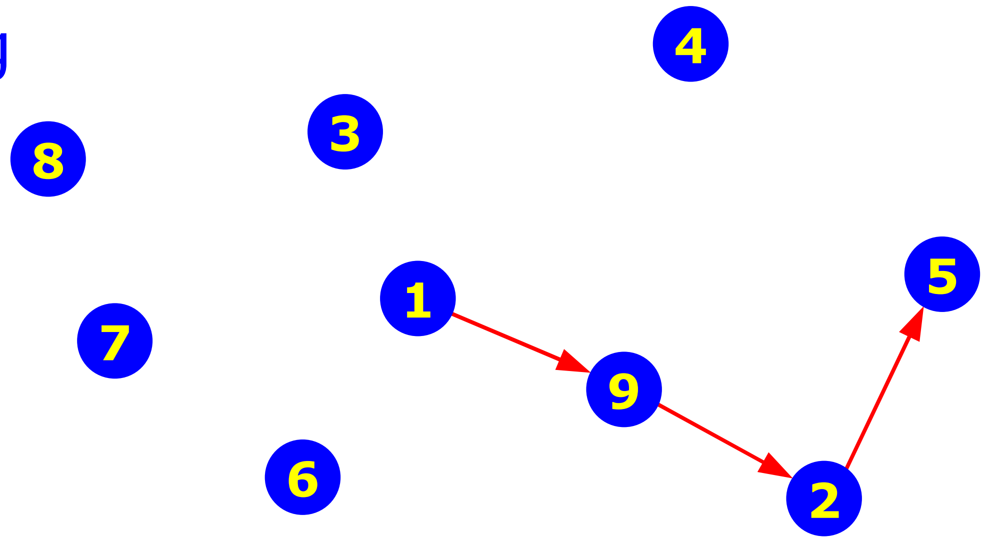
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)

Kodierung von Lösungskandidaten

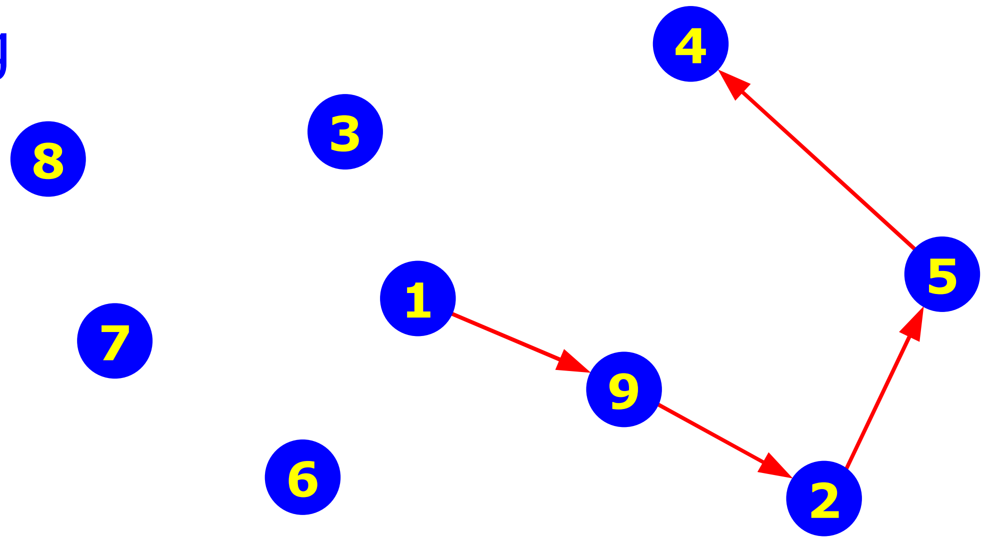
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
	$3 \to 4 \to 6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

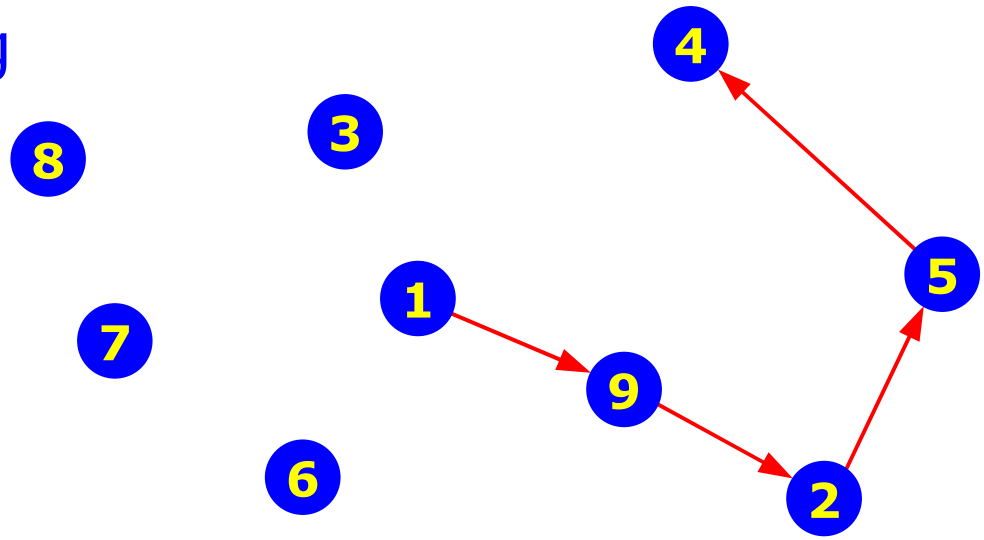
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1)
5	$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3)
4	$3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$	

Kodierung von Lösungskandidaten

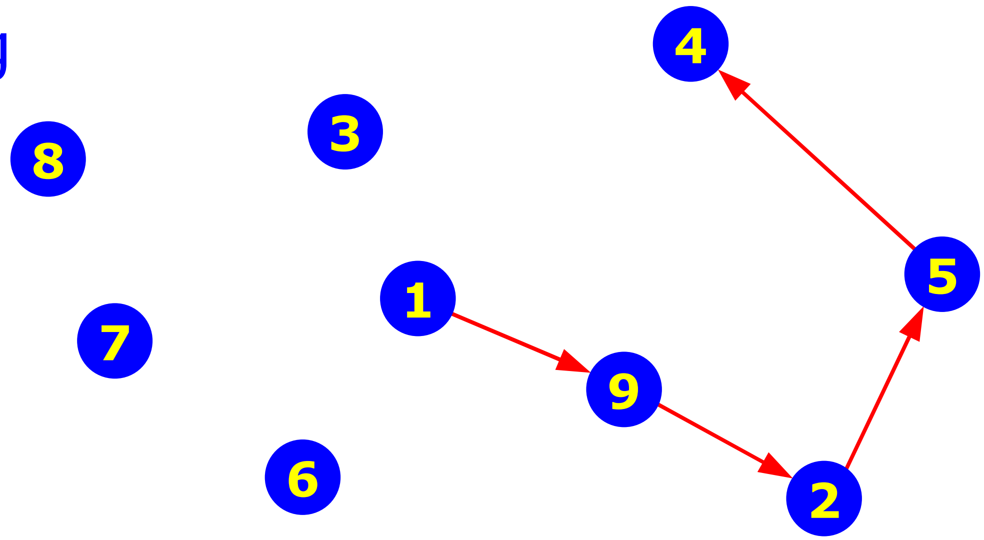
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)

Kodierung von Lösungskandidaten

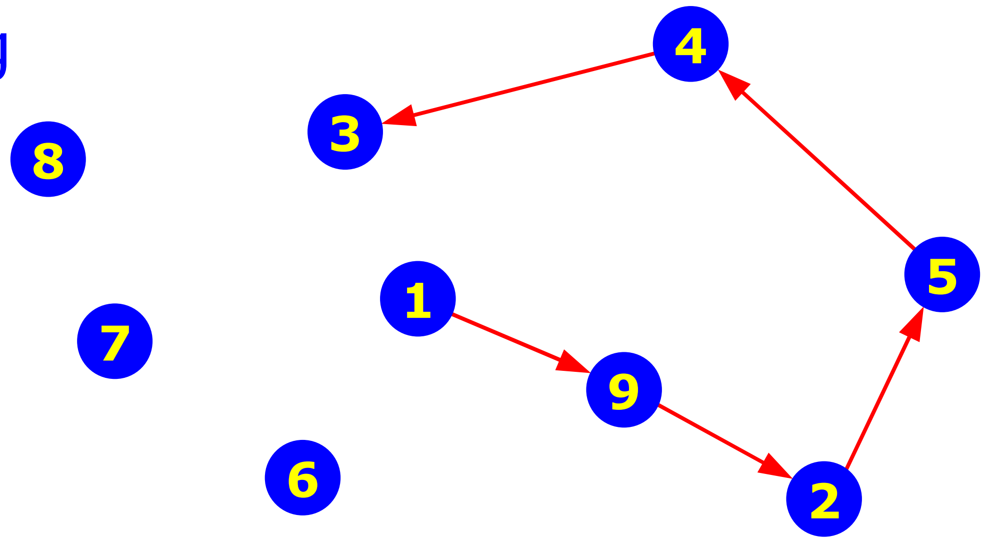
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1)
5	$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3)
4	$3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2)
	$3 \rightarrow 6 \rightarrow 7 \rightarrow 8$	

Kodierung von Lösungskandidaten

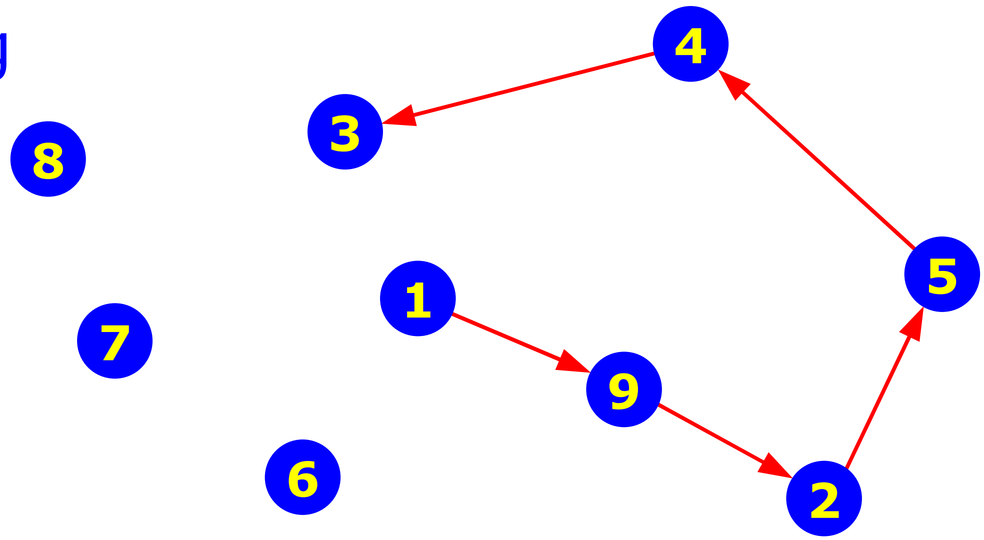
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

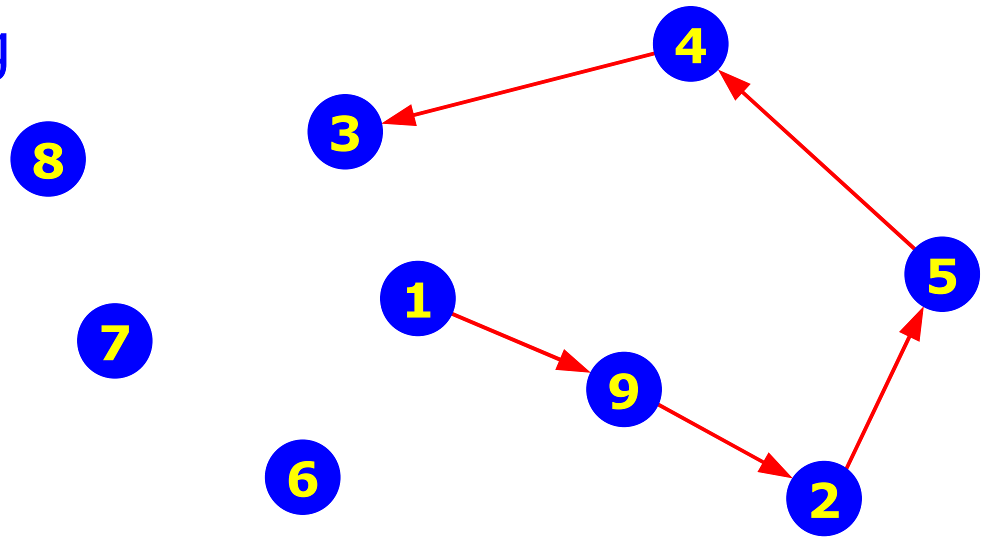
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)

Kodierung von Lösungskandidaten

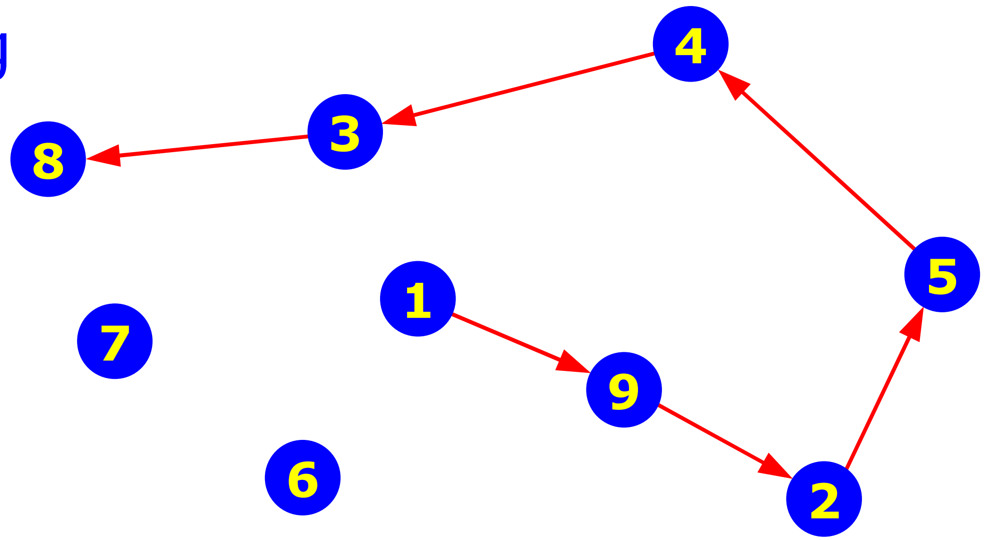
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
	$6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

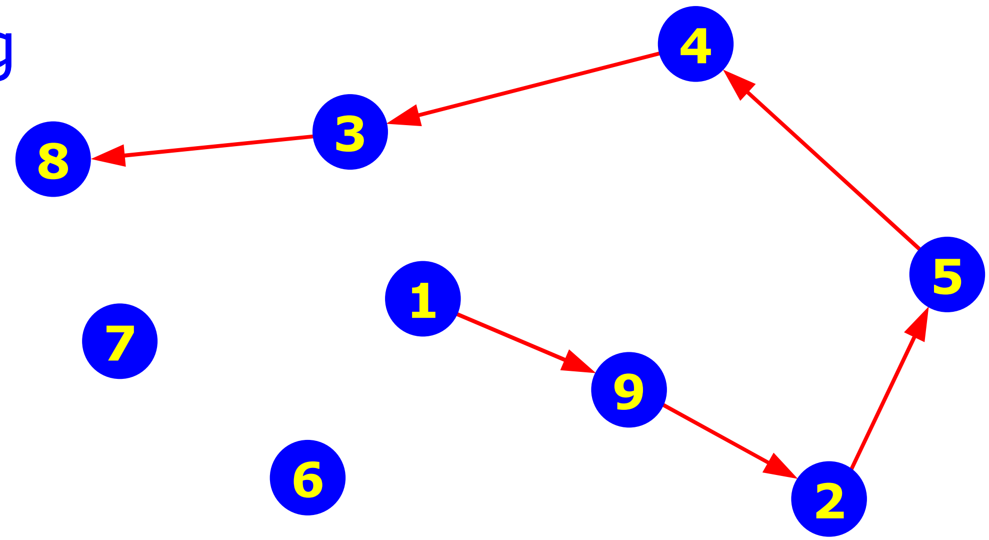
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
8	$6 \to 7 \to 8$	

Kodierung von Lösungskandidaten

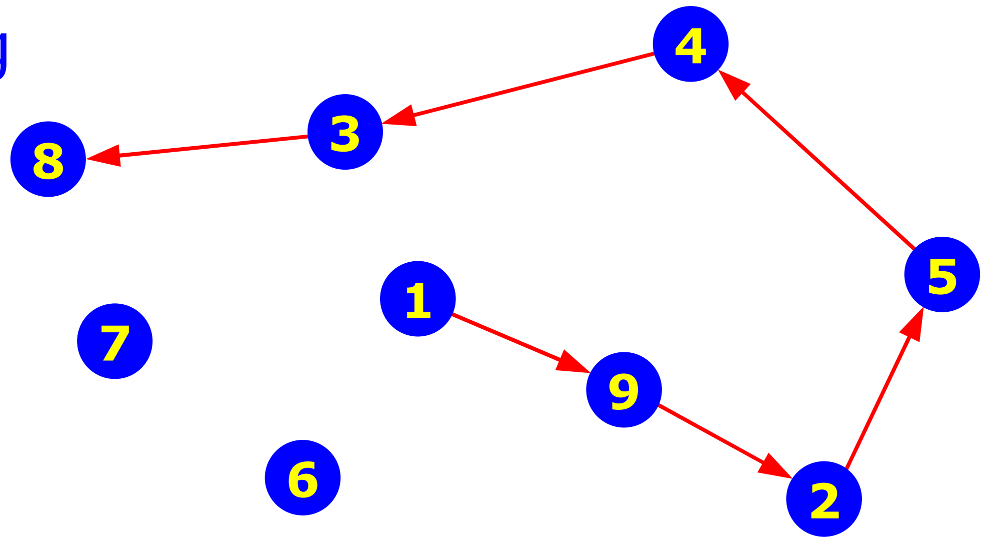
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1)
5	$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3)
4	$3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2)
3	$3 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1)
8	$6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1 3)

Kodierung von Lösungskandidaten

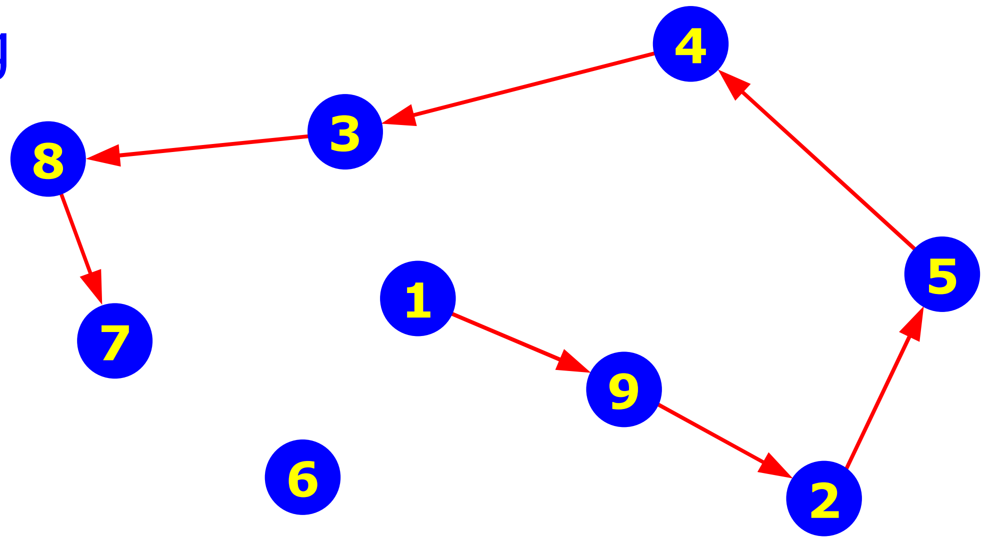
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
8	$6 \to 7 \to 8$	(1 8 1 3 2 1 3)
	$6 \to 7$	

Kodierung von Lösungskandidaten

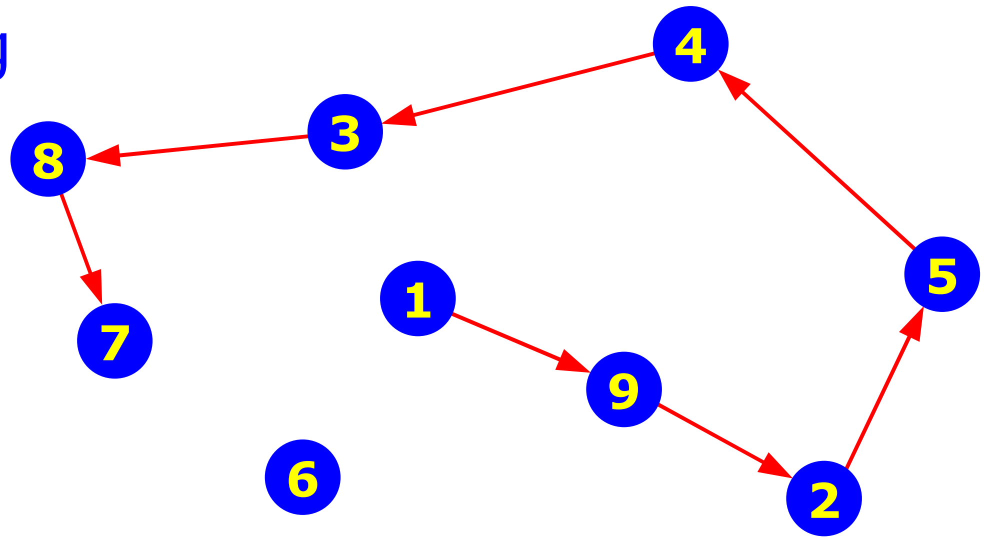
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1)
5	$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3)
4	$3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2)
3	$3 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1)
8	$6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1 3)
7	$6 \rightarrow 7$	

Kodierung von Lösungskandidaten

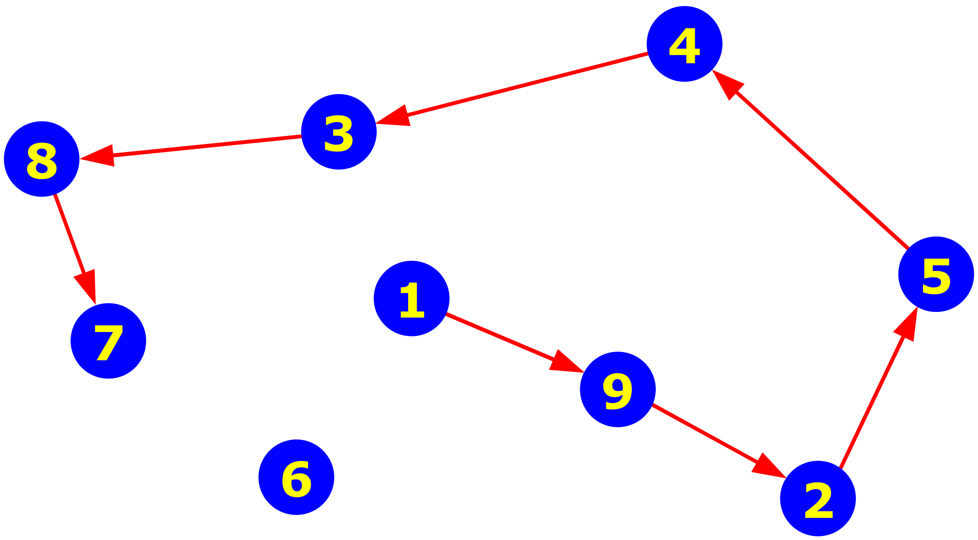
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
8	$6 \to 7 \to 8$	(1 8 1 3 2 1 3)
7	$6 \to 7$	(1 8 1 3 2 1 3 2)

Kodierung von Lösungskandidaten

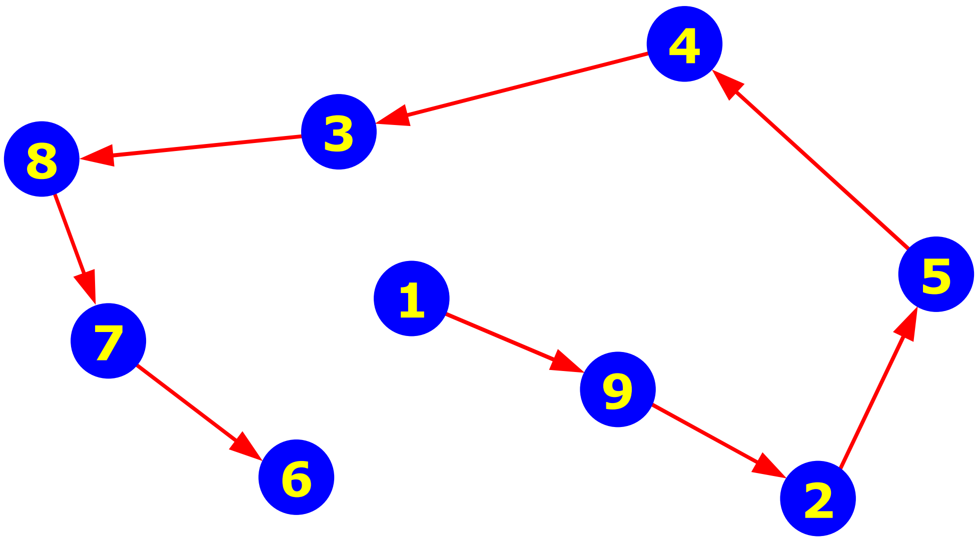
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1)
9	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$	(1 8)
2	$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1)
5	$3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3)
4	$3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2)
3	$3 \rightarrow 6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1)
8	$6 \rightarrow 7 \rightarrow 8$	(1 8 1 3 2 1 3)
7	$6 \rightarrow 7$	(1 8 1 3 2 1 3 2)
	6	

Kodierung von Lösungskandidaten

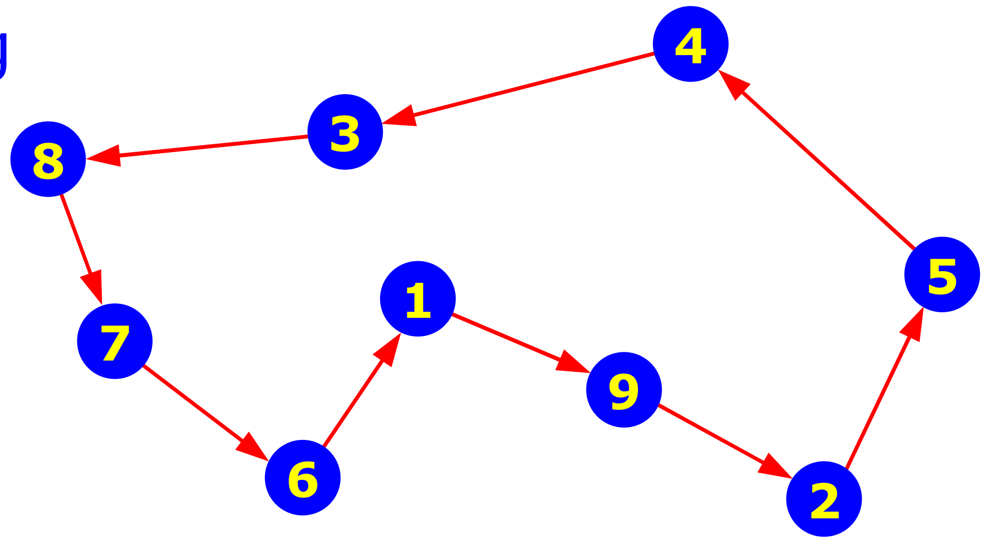
Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
8	$6 \to 7 \to 8$	(1 8 1 3 2 1 3)
7	$6 \to 7$	(1 8 1 3 2 1 3 2)
6	6	

Kodierung von Lösungskandidaten

Beispiel zur Ordinalkodierung



Stadt	Referenz-Rundreise	Kodierung
1	$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1)
9	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9$	(1 8)
2	$2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1)
5	$3 \to 4 \to 5 \to 6 \to 7 \to 8$	(1 8 1 3)
4	$3 \to 4 \to 6 \to 7 \to 8$	(1 8 1 3 2)
3	$3 \to 6 \to 7 \to 8$	(1 8 1 3 2 1)
8	$6 \to 7 \to 8$	(1 8 1 3 2 1 3)
7	$6 \to 7$	(1 8 1 3 2 1 3 2)
6	6	(1 8 1 3 2 1 3 2 1)

Kodierung von Lösungskandidaten

Epistasie

Bei der Ordinalkodierung können hohe Epistasie-Effekte auftreten.

Angenommen, das erste Gen in unserem Beispiel wird mutiert:

Vorher (1 8 1 3 2 1 3 2 1)

Nachher (3 8 1 3 2 1 3 2 1)

Welche Rundreise ergibt sich dabei ?

Kodierung von Lösungskandidaten

Epistasie

Bei der Ordinalkodierung können hohe Epistasie-Effekte auftreten.

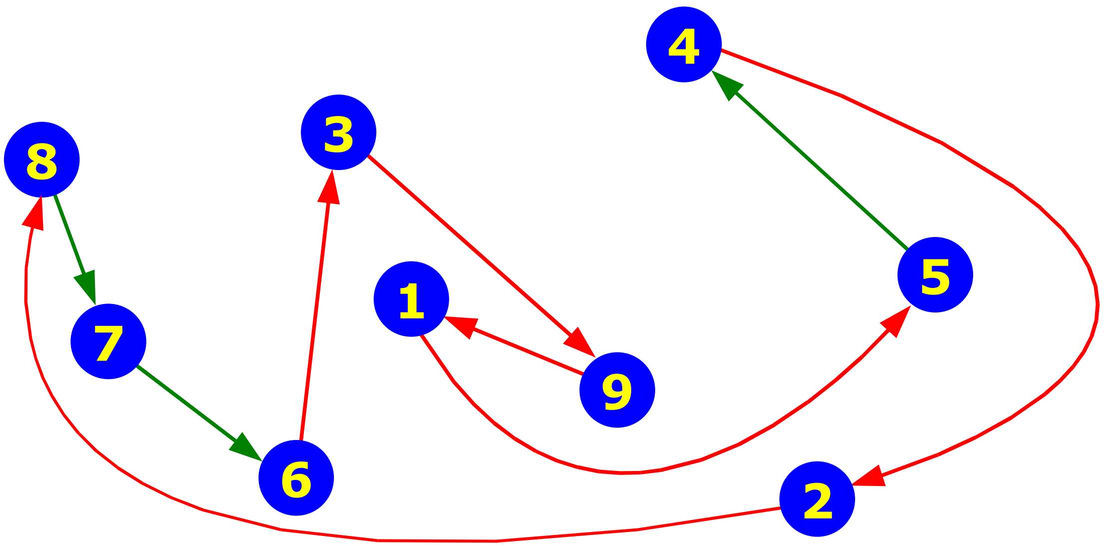
Angenommen, das erste Gen in unserem Beispiel wird mutiert:

Vorher (1 8 1 3 2 1 3 2 1)

Nachher (3 8 1 3 2 1 3 2 1)

Welche Rundreise ergibt sich dabei ?

Es ergibt sich die Rundreise 3 → 9 → 1 → 5 → 4 → 2 → 8 → 7 → 6, die nur noch drei Kanten mit der nicht mutierten Rundreise (grün markiert) gemeinsam hat.



Kodierung von Lösungskandidaten

Epistasie

Bei der Kantenkodierung treten eher geringe Epistasie-Effekte auf.

Angenommen, drei Gene unseres Beispiel werden mutiert:

Vorher (9 5 8 3 4 1 6 7 2)

Nachher (6 5 8 3 4 9 1 7 2)

Welche Rundreise ergibt sich dabei ?

Kodierung von Lösungskandidaten

Epistasie

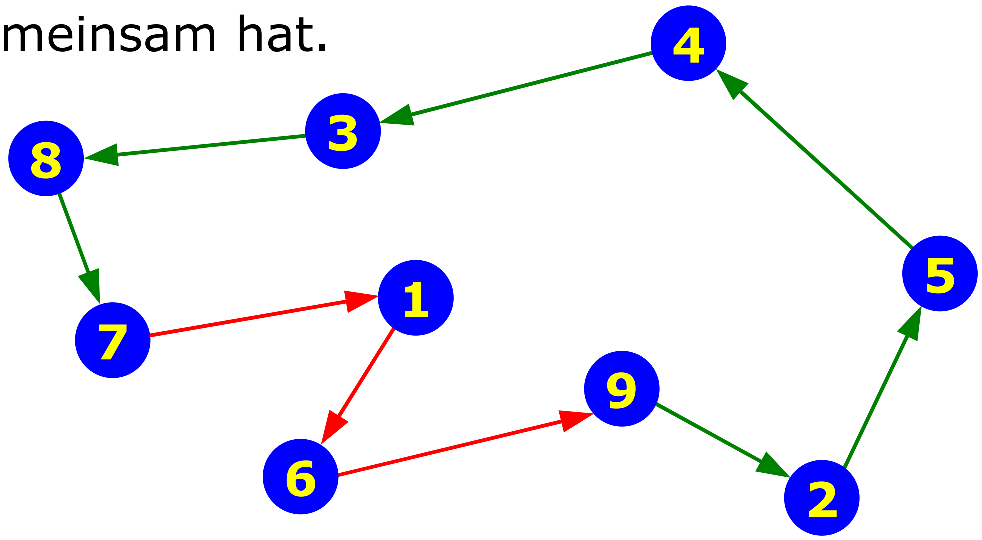
Bei der Kantenkodierung treten eher geringe Epistasie-Effekte auf.

Angenommen, drei Gene unseres Beispiel werden mutiert:

Vorher	(9 5 8 3 4 1 6 7 2)
Nachher	(6 5 8 3 4 9 1 7 2)

Welche Rundreise ergibt sich dabei ?

Obwohl drei Gene mutiert wurden, ergibt sich eine Rundreise 1 → 6 → 9 → 2 → 5 → 4 → 3 → 8 → 7, die immer noch sechs Kanten mit der nicht mutierten Rundreise (grün markiert) gemeinsam hat.



Kodierung von Lösungskandidaten

Epistasie

Schlussfolgerung:

Kodierungen die zu einer hohen Epistasie führen, sollten vermieden werden.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Die Kodierung zu einem gegebenen Problem sollte so gewählt sein, dass alle Lösungskandidaten abgebildet werden können.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Die Kodierung zu einem gegebenen Problem sollte so gewählt sein, dass alle Lösungskandidaten abgebildet werden können.

- Da genetische Algorithmen immer auf der gewählten Kodierung operieren, kann es nun passieren, dass damit auch Elemente erzeugt werden können, die gar nicht im Suchraum liegen.
- Bei einer ungeschickten Kodierung wird der Suchraum also vergrößert. Der so aufgespannte effektive Suchraum umfasst auch die ungültigen Lösungskandidaten.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Die Kodierung zu einem gegebenen Problem sollte so gewählt sein, dass alle Lösungskandidaten abgebildet werden können.

- Da genetische Algorithmen immer auf der gewählten Kodierung operieren, kann es nun passieren, dass damit auch Elemente erzeugt werden können, die gar nicht im Suchraum liegen.
- Bei einer ungeschickten Kodierung wird der Suchraum also vergrößert. Der so aufgespannte effektive Suchraum umfasst auch die ungültigen Lösungskandidaten.

Beispiel

Ein GA soll einen strategisch günstigen Kaufpreis ermitteln. Die Lösungskandidaten sind Preise, die in binär kodierter Dezimaldarstellung (BCD) kodiert wurden. Ein Byte umfasst zwei BCD-Ziffern. Die 256 möglichen Werte eines Bytes spalten sich hierbei in 100 gültige und 156 ungültige Kodierungen auf.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Wie geht man mit ungültigen Lösungskandidaten um? Es bieten sich folgende Strategien an:

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Wie geht man mit ungültigen Lösungskandidaten um? Es bieten sich folgende Strategien an:

- Am besten ist es, wenn man eine andere, bessere Kodierung verwenden kann. Dies wird allerdings nicht immer möglich sein.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Wie geht man mit ungültigen Lösungskandidaten um? Es bieten sich folgende Strategien an:

- Am besten ist es, wenn man eine andere, bessere Kodierung verwenden kann. Dies wird allerdings nicht immer möglich sein.
- Beim Dekodieren ungültige Lösungskandidaten reparieren. Ein Byte könnte beispielsweise solange hochgezählt werden, bis sich wieder eine gültige BCD-Zahl ergibt.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Wie geht man mit ungültigen Lösungskandidaten um? Es bieten sich folgende Strategien an:

- Am besten ist es, wenn man eine andere, bessere Kodierung verwenden kann. Dies wird allerdings nicht immer möglich sein.
- Beim Dekodieren ungültige Lösungskandidaten reparieren. Ein Byte könnte beispielsweise solange hochgezählt werden, bis sich wieder eine gültige BCD-Zahl ergibt.
- Bei den ungültigen Lösungskandidaten die Fitness sehr niedrig bewerten, so dass diese dazu tendieren, auszusterben. Wenn die Kodierung allerdings so schlecht ist, dass der effektive Suchraum viele ungültige Lösungskandidaten enthält, wird der GA dadurch sehr ineffizient.

Kodierung von Lösungskandidaten

Ungültige Lösungskandidaten

Wie geht man mit ungültigen Lösungskandidaten um? Es bieten sich folgende Strategien an:

- Am besten ist es, wenn man eine andere, bessere Kodierung verwenden kann. Dies wird allerdings nicht immer möglich sein.
- Beim Dekodieren ungültige Lösungskandidaten reparieren. Ein Byte könnte beispielsweise solange hochgezählt werden, bis sich wieder eine gültige BCD-Zahl ergibt.
- Bei den ungültigen Lösungskandidaten die Fitness sehr niedrig bewerten, so dass diese dazu tendieren, auszusterben. Wenn die Kodierung allerdings so schlecht ist, dass der effektive Suchraum viele ungültige Lösungskandidaten enthält, wird der GA dadurch sehr ineffizient.
- Die genetischen Operatoren auf die gewählte Kodierung anpassen. Die Mutation einer BCD-Zahl könnte beispielsweise so realisiert sein, dass immer eine gültige BCD-Zahl zurückgeliefert wird.

Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

Beispiel

(a) Mit absoluten Koordinaten (x_0, y_0) (x_1, y_1) ...

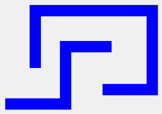
Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

Beispiel

(a) Mit absoluten Koordinaten (x_0, y_0) (x_1, y_1) ...

Probleme:



Die Kette belegt nicht notwendigerweise zusammenhängende Zellen auf dem Gitter, d. h. es gibt mindestens eine Stelle *i* in der Konformation für die folgendes gilt: $|x_i - x_{i+1}| + |y_i - y_{i+1}| > 1$

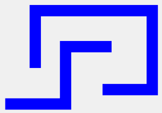
Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

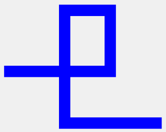
Beispiel

(a) Mit absoluten Koordinaten (x_0, y_0) (x_1, y_1) ...

Probleme:



Die Kette belegt nicht notwendigerweise zusammenhängende Zellen auf dem Gitter, d. h. es gibt mindestens eine Stelle *i* in der Konformation für die folgendes gilt: $|x_i - x_{i+1}| + |y_i - y_{i+1}| > 1$



Mehrere Elemente können die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein, d. h. es gibt mindestens zwei verschiedene Stellen *i* und *j*, für die gilt: $x_i=x_j$ und $y_i=y_j$.

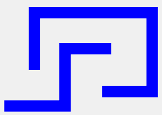
Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

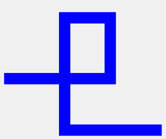
Beispiel

(a) Mit absoluten Koordinaten (x_0, y_0) (x_1, y_1) ...

Probleme:



Die Kette belegt nicht notwendigerweise zusammenhängende Zellen auf dem Gitter, d. h. es gibt mindestens eine Stelle *i* in der Konformation für die folgendes gilt: $|x_i - x_{i+1}| + |y_i - y_{i+1}| > 1$



Mehrere Elemente können die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein, d. h. es gibt mindestens zwei verschiedene Stellen *i* und *j*, für die gilt: $x_i=x_j$ und $y_i=y_j$.

Wie würden Sie mit dieser Situation umgehen ?

Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

Beispiel

(b) Mit relativen Gitterkoordinaten

```
public enum Richtung { NORD, SÜD, OST, WEST }
...
List<Richtung> konformation;
```

Kodierung von Lösungskandidaten

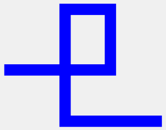
Kodierungen im 2d HP-Modell

Beispiel

(b) Mit relativen Gitterkoordinaten

```
public enum Richtung { NORD, SÜD, OST, WEST }
...
List<Richtung> konformation;
```

Problem:



Auch in dieser Kodierung können mehrere Elemente die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein.

Kodierung von Lösungskandidaten

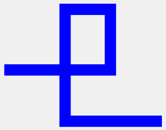
Kodierungen im 2d HP-Modell

Beispiel

(b) Mit relativen Gitterkoordinaten

```
public enum Richtung { NORD, SÜD, OST, WEST }
...
List<Richtung> konformation;
```

Problem:



Auch in dieser Kodierung können mehrere Elemente die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein.

Wie würden Sie mit dieser Situation umgehen ?

Kodierung von Lösungskandidaten

Kodierungen im 2d HP-Modell

Beispiel

(c) Mit relativen Konformationskoordinaten

```
public enum Richtung { LINKS, RECHTS, GERADEAUS }
...
List<Richtung> konformation;
```

Kodierung von Lösungskandidaten

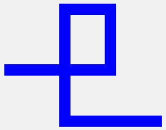
Kodierungen im 2d HP-Modell

Beispiel

(c) Mit relativen Konformationskoordinaten

```
public enum Richtung { LINKS, RECHTS, GERADEAUS }
...
List<Richtung> konformation;
```

Problem:



Auch in dieser Kodierung können mehrere Elemente die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein.

Kodierung von Lösungskandidaten

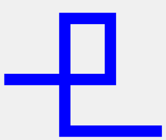
Kodierungen im 2d HP-Modell

Beispiel

(c) Mit relativen Konformationskoordinaten

```
public enum Richtung { LINKS, RECHTS, GERADEAUS }
...
List<Richtung> konformation;
```

Problem:



Auch in dieser Kodierung können mehrere Elemente die gleichen Koordinaten aufweisen und somit auf die gleiche Zelle im Gitter platziert worden sein.

Welche der beiden Kodierungen mit relativen Koordinaten ist besser geeignet ? Warum ?