Technische Universität Braunschweig

Decision Support

Institut für Wirtschaftsinformatik

Operations Research

Vorlesung 12

Heuristiken: Metaheuristiken

Wiederholung

- Knotenorientierte Rundreisen: Traveling Salesman Problem

- Heuristische Verfahren
  - Ermittlung guter zulässiger Lösungen mit vertretbarem Aufwand, allerdings unter Aufgabe der Optimalitätsgarantie
- Heuristische Lösungsverfahren für das TSP
  - Unvollständig ausgeführte exakte Verfahren
    - Unvollständiger B&B
  - Eröffnungsverfahren
    - Bester Nachfolger
    - Sukzessive Einbeziehung
  - Verbesserungsverfahren
    - 2-opt

Überblick

1. Motivation für Metaheuristiken
2. Ausgewählte Metaheuristiken
    1. Tabu Suche
    2. Genetische Algorithmen

Überblick

1. **Motivation für Metaheuristiken**
2. Ausgewählte Metaheuristiken
    1. Tabu Suche
    2. Genetische Algorithmen

Verbesserungsverfahren: Wiederholung

- Starten (in der Regel) mit einer zulässigen Ausgangslösung x
- In jedem Iterationsschritt wird von der gerade betrachteten Lösung x zu einer Lösung aus der Nachbarschaft $\mathrm{NB}(x)$ fortgeschritten
- $\mathrm{NB}(x)$ enthält sämtliche Lösungen, die sich aus x durch einmalige Anwendung einer Transformationsvorschrift ergeben („Zug“)
- Beispiele möglicher Transformationsvorschriften (Züge):
  - Veränderung einer Lösung an genau einer Stelle (z.B. „Kippen eines Bits“ beim Rucksack-Problem: Entfernen bzw. Einpacken eines Gutes)
  - r-opt-Verbesserungsschritt beim TSP

Verbesserungsverfahren: Verfahren des steilsten Anstiegs

Das Verfahren bricht ab, sobald es keine bessere Lösung in der Nachbarschaft gibt.

Initialisierung mit gegebener Startlösung x_c

Verarbeitung

1 WHILE $true$ DO
2 Bestimme $x_n = \max_{y \in NB(x_c) \setminus x_c} f(y)$
3 IF $f(x_c) \leq f(x_n)$ THEN
4 $x_c := x_n$
5 ELSE
6 BREAK
7 ENDIF
8 ENDWHILE

Wähle beste Nachbarlösung x_n

Wenn x_n besser ist als x_c:
Setze $x_c := x_n$ und fahre fort

Wenn nicht, breche die Suche ab

Output Lösung x_c

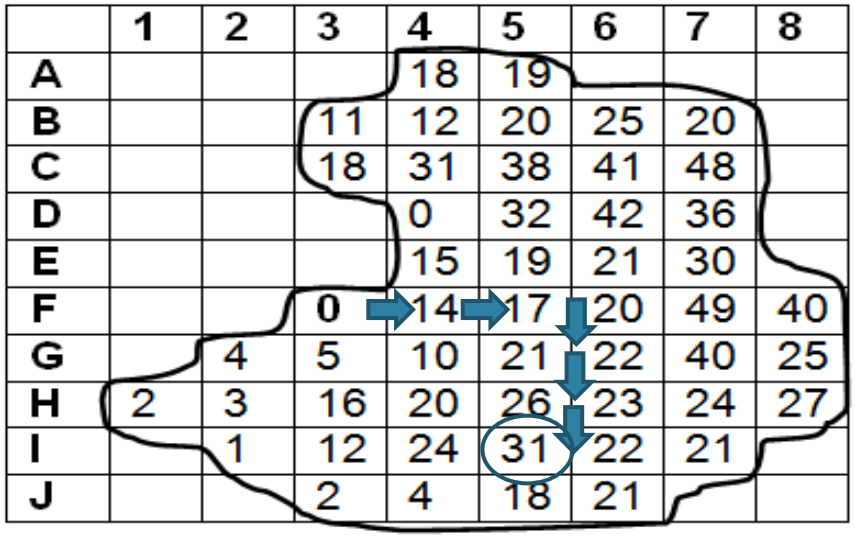
Der schiffbrüchige Kurt: Einführung

Der schiffbrüchige Kurt ist auf einer einsamen Insel gestrandet, deren in Planquadrate eingeteilte Karte unten dargestellt ist. Die Zahlen geben die Höhenmeter der Planquadrate an. Um von Suchmannschaften besser entdeckt zu werden, möchte sich Kurt auf den höchsten Punkt der Insel begeben. Leider ist der Dschungel so dicht, dass er seiner aktuellen Position in einem Planquadrat jeweils nur die Höhen der in den vier Himmelsrichtungen (Norden, Süden, Westen, Osten) benachbarten Planquadrate erkennen kann. Es sind also nur Züge nach Norden, Süden, Westen, Osten möglich.

	1	2	3	4	5	6	7	8
A				18	19			
B			11	12	20	25	20	
C			18	31	38	41	48	
D				0	32	42	36	
E				15	19	21	30	
F			0	14	17	20	49	40
G		4	5	10	21	22	40	25
H	2	3	16	20	26	23	24	27
I		1	12	24	31	22	21	
J			2	4	18	21		

Der schiffbrüchige Kurt: Einfaches Hill-Climbing

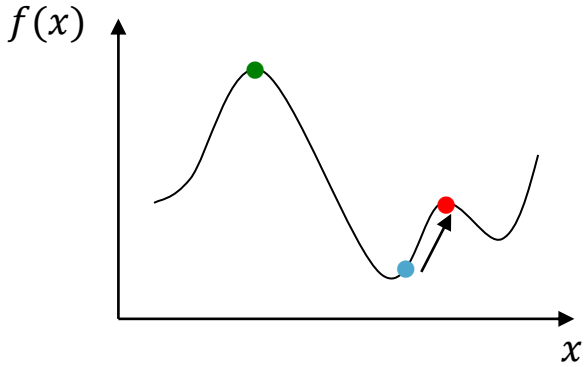
- Gehen Sie davon aus, dass Kurt in F3 gestrandet ist.
- Zu welchem Punkt auf der Karte gelangt er bei Anwendung des reinen Verbesserungsverfahrens mit steilstem Anstieg?
- Hier: Wahl des höchsten benachbarten Planquadrats (ohne sich zu verschlechtern)



	1	2	3	4	5	6	7	8
A				18	19			
B			11	12	20	25	20	
C			18	31	38	41	48	
D				0	32	42	36	
E				15	19	21	30	
F			0	14	17	20	49	40
G		4	5	10	21	22	40	25
H	2	3	16	20	26	23	24	27
I		1	12	24	31	22	21	
J			2	4	18	21		

Verbesserungsverfahren: Lokale Optima

- Verbesserungsverfahren brechen ab, sobald keine bessere Nachbarlösung mehr gefunden wird
  $\Rightarrow$ Gefahr des „Steckenbleibens“ in lokalen Optima



Startlösung

Lokales Maximum

Globales Maximum

- Um lokale Optima verlassen zu können, müssen Züge erlaubt sein, die eine zwischenzeitliche Verschlechterung des Zielfunktionswertes zulassen $\Rightarrow$ **Metaheuristiken**

Ausgewählte Metaheuristiken

- Tabu Suche / Tabu Search (Glover, 1989)
- Genetische / Evolutionäre Algorithmen (Goldberg, 1989)
- Simulated Annealing (Kirckpatrick et al. 1983)
- Threshold Accepting (Dueck und Scheuer, 1990)
- Ant Colony Optimization (Dorigo, 1992)
- Particle Swarm Optimization (Kennedy und Eberhart, 1995)

Überblick

1. Motivation für Metaheuristiken
2. **Ausgewählte Metaheuristiken**
    1. **Tabu Suche**
    2. Genetische Algorithmen

Tabu Suche

Grundprinzip:

- Systematisches Absuchen der gesamten Nachbarschaft $\mathrm{NB}(x)$
- Aus allen (nicht verbotenen) Nachbarlösungen wird diejenige mit dem besten Zielfunktionswert für die nächste Iteration ausgewählt (auch wenn sie eine Verschlechterung im Vergleich zur augenblicklichen Lösung x darstellt)

$\Rightarrow$ Prinzip des steilsten Anstiegs / mildestem Abstieg (für Maximierungsprobleme)

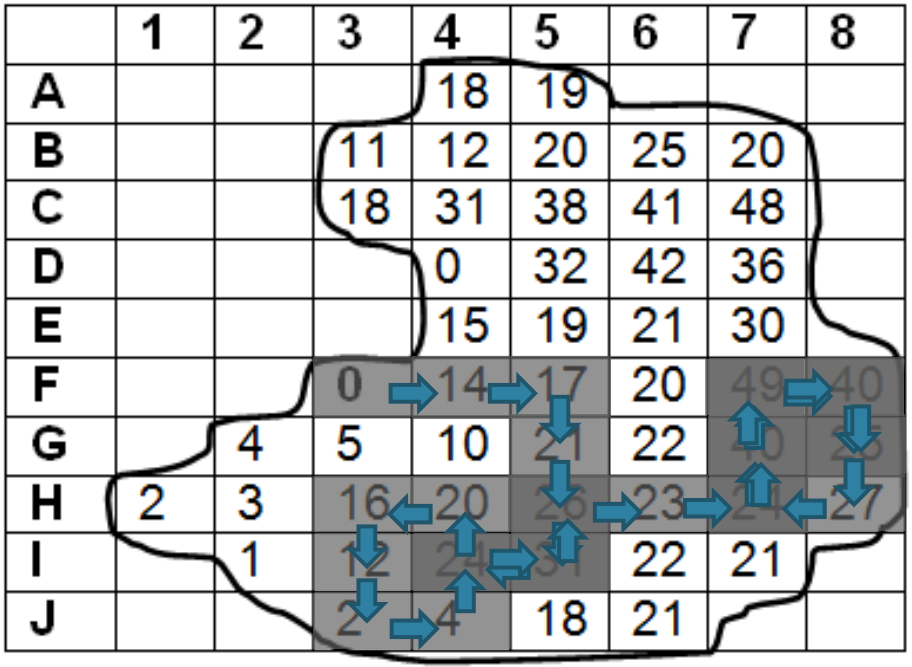
- Um nach Akzeptanz einer Verschlechterung in der nächsten Iteration nicht wieder zur Ausgangslösung zurückzukehren, müssen derartige Züge verboten („tabu“ gesetzt) werden
- Verbotene Züge werden in sog. Tabu-Listen gespeichert; die Dauer der Verbote (angegeben in Iterationen) beeinflusst die Lösungsqualität maßgeblich

Gefahr:

Verfahren kann ins Kreisen geraten (v.a. bei kurzen Tabu-Listen), bei langen Listen Gefahr der Verschlechterung von Lösungen

Der schiffbrüchige Kurt: Tabu Suche (I)

- Verschlechterungen werden in Kauf genommen
- Dafür werden die letzten vier Züge gesperrt



	1	2	3	4	5	6	7	8
A				18	19			
B			11	12	20	25	20	
C			18	31	38	41	48	
D				0	32	42	36	
E				15	19	21	30	
F			0	14	17	20	49	40
G		4	5	10	21	22	40	28
H	2	3	16	20	26	23	24	27
I		1	12	25	31	22	21	
J			2	4	18	21		

Der schiffbrüchige Kurt: Tabu Suche (II)

- Verschlechterungen werden in Kauf genommen
- Dafür werden die letzten vier Züge gesperrt

	1	2	3	4	5	6	7	8
A				18	19			
B			11	12	20	25	20	
C			18	31	38	41	48	
D				0	32	42	36	
E				15	19	21	30	
F			0	14	17	20	49	40
G		4	5	10	21	22	40	25
H	2	3	16	20	26	23	24	27
I		1	12	24	31	22	21	
J			2	4	18	21		

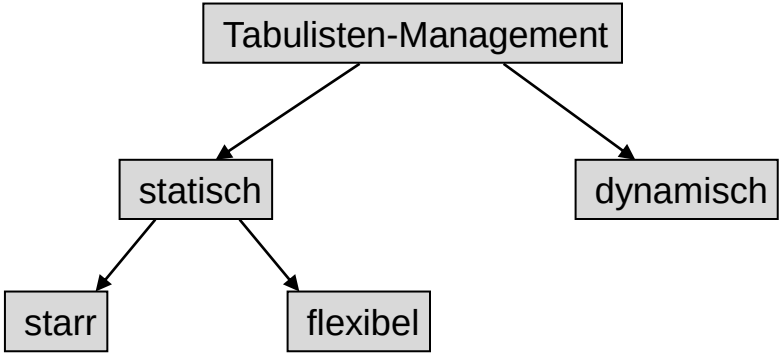
➔ Finden einer besseren, evtl. optimalen, Lösung möglich

➔ ABER: Lösung abhängig von Tabulistenlänge

Standort	Höhe	Gesperrt
F3	0	-
F4	14	F3
F5	17	F3,F4
G5	21	F3,F4,F5
H5	26	F3,F4,F5,G5
I5	31	F4,F5,G5,H5
I4	24	F5,G5,H5,I5
H4	20	G5,H5,I5,I4
H3	16	H5,I5,I4,H4
I3	12	I5,I4,H4,H3
J3	2	I4,H4,H3,I3
J4	4	H4,H3,I3,J3
I4	24	H3,I3,J3,J4
I5	31	I3,J3,J4,I4
H5	26	J3,J4,I4,I5
H6	23	J4,I4,I5,H5
H7	24	I4,I5,H5,H6
G7	40	I5,H5,H6,H7
F7	49	H5,H6,H7,G7
F8	40	H6,H7,G7,F7
G8	25	H7,G7,F7,F8
H8	27	G7,F7,F8,G8
H7	24	F7,F8,G8,H8
G7	40	F8,G8,H8,H7
F7	49	G8,H8,H7,G7
F8	40	H8,H7,G7,F7
G8	25	H7,G7,F7,F8

Schleife: Position & Tabuliste identisch

Möglichkeiten des Tabulisten-Managements



Starre Listenlänge: Während des gesamten Verfahrens werden die letzten ... Züge „tabu“ gesetzt

Flexible Listenlänge: Während des Verfahrens wird die Listenlänge zwischen einer unteren und einer oberen Schranke variiert

Dynamische Listenlänge: Liste wird bei jedem Zug angepasst, so dass nur solche Züge „tabu“ gesetzt werden, die zu bereits untersuchten Lösungen führen

Rucksackproblem: Einführung

Max $3x_1 + 4x_2 + 2x_3 + 3x_4$

u.d.N. $3x_1 + 2x_2 + 4x_3 + x_4 \leq 9$

$x_i \in \{0, 1\}$

Nachbarschaft NB der Lösung x:
alle Lösungen x^i, die sich von x an genau einer Position unterscheiden, z.B.

Lösung:	$x = (1, 0, 0, 1)$	(Rucksack mit Objekt 1 und 4 eingepackt)
NB:	$x^1 = (0, 0, 0, 1)$	(Objekt 1 wird ausgepackt)
	$x^2 = (1, 1, 0, 1)$	(Objekt 2 wird eingepackt)
	$x^3 = (1, 0, 1, 1)$	(Objekt 3 wird eingepackt)
	$x^4 = (1, 0, 0, 0)$	(Objekt 4 wird ausgepackt)

Mögliche Züge:
$x_i = 0 \rightarrow x_i = 1$: Objekt i wird eingepackt
$x_i = 1 \rightarrow x_i = 0$: Objekt i wird ausgepackt

Rucksackproblem: Tabu Suche (I)

- Züge, mit denen unzulässige Lösungen erreicht werden, sollen mit **Strafkosten von 5** belegt werden
- In der Tabuliste werden die **Umkehrungen** der beiden zuletzt ausgeführten Züge gespeichert (statisch starre Tabuliste) → Verhinderung der Rückkehr

Startlösung: $x = (0, 0, 0, 0)$, x zulässig

Iteration 1:		ZF-Wert	Gewicht	Tabuliste
Lösung:	$x = (0, 0, 0, 0)$		0	0
NB:	$x = (1, 0, 0, 0)$	3	3	
	$x = (0, 1, 0, 0)$	4	2	
	$x = (0, 0, 1, 0)$	2	4	
	$x = (0, 0, 0, 1)$	3	1	
Iteration 2:				
Lösung:	$x = (0, 1, 0, 0)$	4	2	2
NB:	$x = (1, 1, 0, 0)$	7	5	
	$x = (0, 1, 1, 0)$	6	6	
	$x = (0, 1, 0, 1)$	7	3	

$$\text{Max} \quad 3x_1 + 4x_2 + 2x_3 + 3x_4$$
$$\text{u.d.N.} \quad 3x_1 + 2x_2 + 4x_3 + x_4 \leq 9$$
$$x_i \in \{0, 1\}$$

Rucksackproblem: Tabu Suche (II)

		ZF-Wert	Gewicht	Tabuliste
Iteration 3:				
Lösung:	$x = (0, 1, 0, 1)$	7	3	2,4
NB:	$x = (1, 1, 0, 1)$	10	6	
	$x = (0, 1, 1, 1)$	9	7	
Iteration 4:				
Lösung:	$x = (1, 1, 0, 1)$	10	6	4,1
NB:	$x = (1, 0, 0, 1)$	6	4	
	$x = (1, 1, 1, 1)$	12-5=7	10	$(x = (1,1,1,1) \notin Z)$ → nicht zulässig, daher 5 Einheiten Strafkosten
Iteration 5:				
Lösung:	$x = (1, 1, 1, 1)$	7	10	1,3
NB:	$x = (1, 0, 1, 1)$	8	8	
	$x = (1, 1, 1, 0)$	9	9	

$$\begin{aligned} \text{Max} \quad & 3x_1 + 4x_2 + 2x_3 + 3x_4 \\ \text{u.d.N.} \quad & 3x_1 + 2x_2 + 4x_3 + x_4 \leq 9 \\ & x_i \in \{0,1\} \end{aligned}$$

Rucksackproblem: Tabu Suche (III)

Iteration 6:		ZF-Wert	Gewicht	Tabuliste
Lösung:	$x = (1, 1, 1, 0)$	9	9	3,4
NB:	$x = (0, 1, 1, 0)$	6	6	
	$x = (1, 0, 1, 0)$	5	7	

Iteration 7:				
Lösung:	$x = (0, 1, 1, 0)$	6	6	4,1
NB:	$x = (0, 0, 1, 0)$	2	4	
	$x = (0, 1, 0, 0)$	4	2	

Iteration 8:				
Lösung:	$x = (0, 1, 0, 0)$	4	2	1,3
NB:	$x = (0, 0, 0, 0)$	0	0	
	$x = (0, 1, 0, 1)$	7	3	

$$\begin{aligned} \text{Max} \quad & 3x_1 + 4x_2 + 2x_3 + 3x_4 \\ \text{u.d.N.} \quad & 3x_1 + 2x_2 + 4x_3 + x_4 \leq 9 \\ & x_i \in \{0, 1\} \end{aligned}$$

Rucksackproblem: Tabu Suche (IV)

Iteration 9:		ZF-Wert	Gewicht	Tabuliste
Lösung:	$x = (0, 1, 0, 1)$	7	3	3,4
NB:	$x = (1, 1, 0, 1)$	10	6	
	$x = (0, 0, 0, 1)$	3	1	

$$\begin{aligned} \text{Max} \quad & 3x_1 + 4x_2 + 2x_3 + 3x_4 \\ \text{u.d.N.} \quad & 3x_1 + 2x_2 + 4x_3 + x_4 \leq 9 \\ & x_i \in \{0, 1\} \end{aligned}$$

Iteration 10:				
Lösung:	$x = (1, 1, 0, 1)$	10	6	4,1

Iteration 10 identisch mit Iteration 4 $\Rightarrow$ Verfahren gerät ins Kreisen

Beste gefundene (zulässige) Lösung: $x = (1, 1, 0, 1)$ mit ZF = 10

Überblick

1. Motivation für Metaheuristiken
2. **Ausgewählte Metaheuristiken**
    1. Tabu Suche
    2. **Genetische Algorithmen**

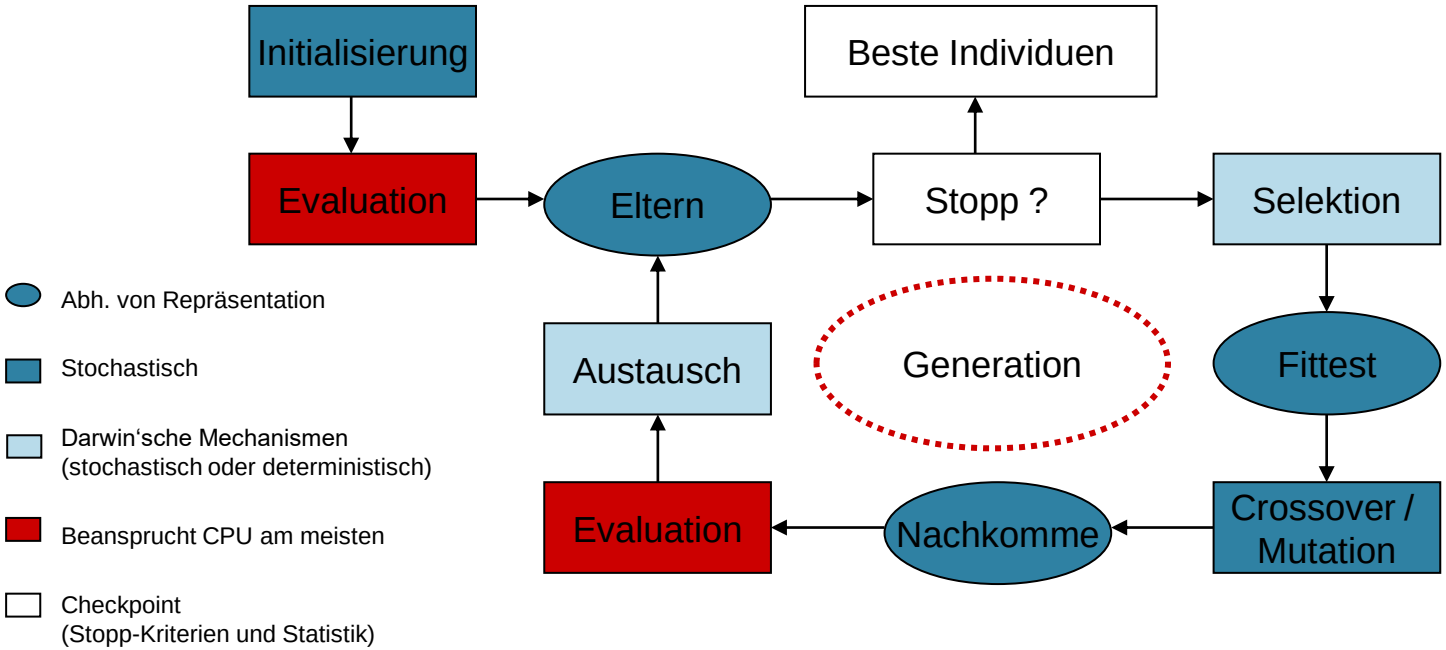
Grundprinzip Genetischer Algorithmen

Simulation von Darwins Satz „survival of the fittest“:

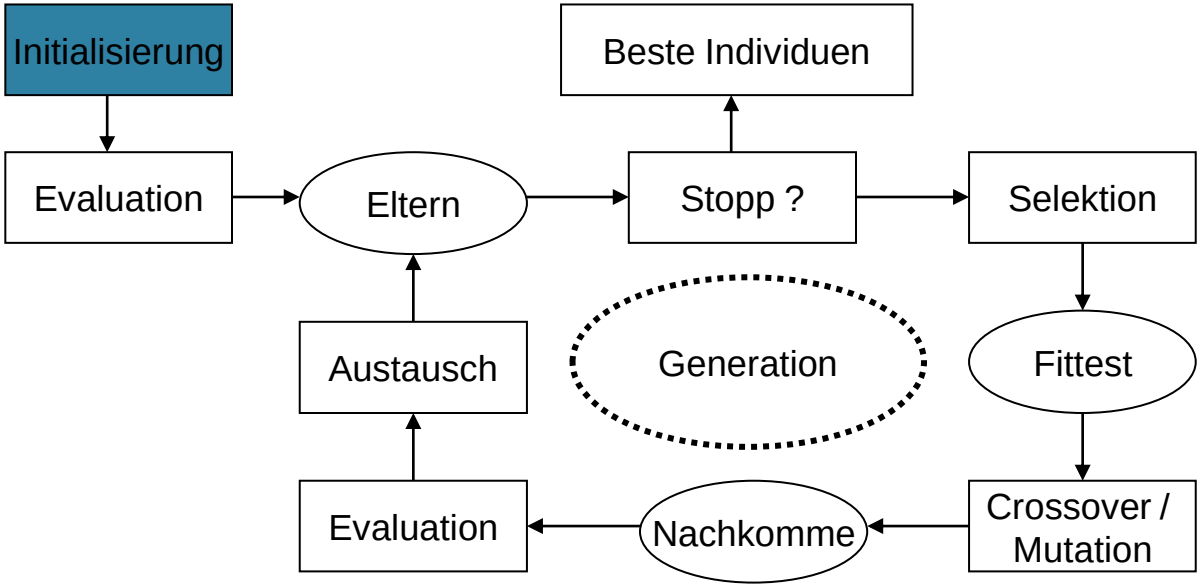
- Erzeugung ganzer Populationen (Mengen) von Lösungen
- „Selektion“ besonders guter Individuen (Lösungen) aus der Elterngeneration
- Nachkommen werden durch „Kreuzung“ guter Lösungen erzeugt
- Lösung wird an einer oder mehreren Positionen durch „Mutation“ verändert

Genetische Algorithmen sind besonders geeignet, wenn Lösungen in Form von Binärvektoren vorliegen
$\Rightarrow$ z.B. Rucksackproblem

Komponenten eines Genetischen Algorithmus



Repräsentation und Initialisierung



Repräsentation und Initialisierung

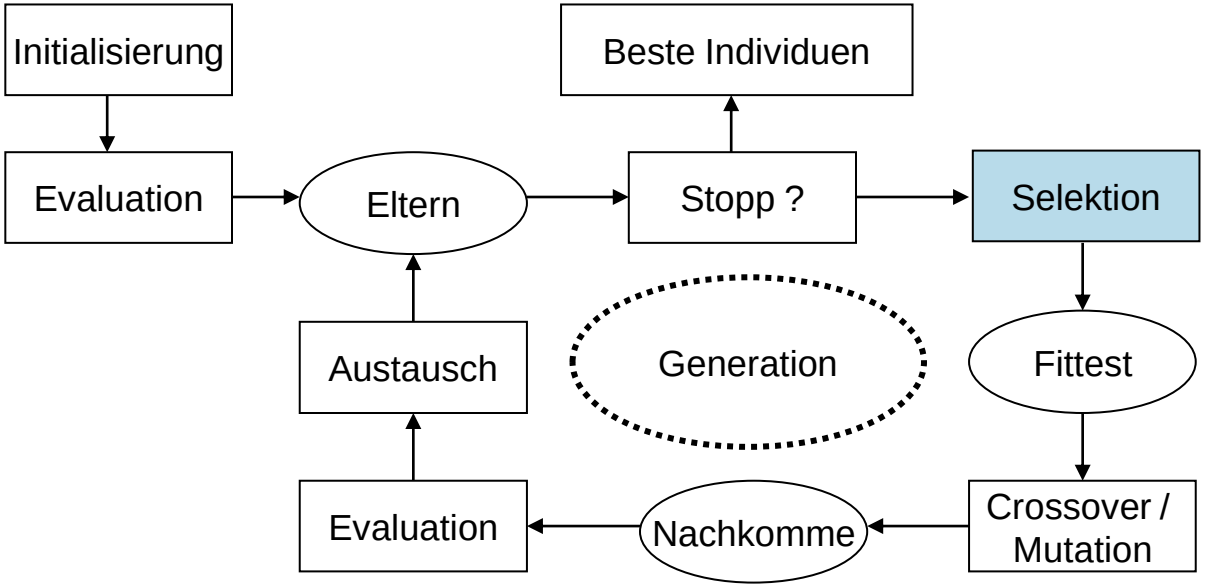
- **Binäre Kodierung für ein Rucksackproblem**
  - Rucksackproblem mit n Objekten ➔ 1 Bit je Objekt
    - Bit gleich 1, wenn Objekt in dem Rucksack ist
    - Ansonsten Bit gleich 0

- **Beispiel**

Codierung	➔	**Interpretation**
1 — 0 — 0 — 1		Objekte 1 und 4 sind in dem Rucksack

- **Initialisierung**
  - Zufällig
  - Heuristisch (Relaxation, siehe Kapitel 6, Folie 9)

Selektion



Selektion

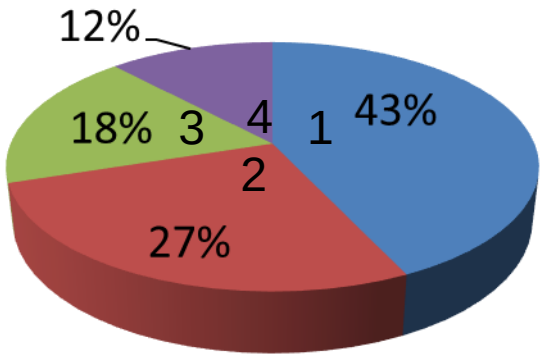
- Selektionskomponente
  - Ziel: bessere Lösungen (z.B. Touren) häufiger zur Rekombination heranziehen
    - erfolgreiche Individuen werden mit größerer Wahrscheinlichkeit „Eltern“
  - Vorgehen: Selektionsverfahren
  - Idee:
    - *Identifikation der besten Lösungen (Intensivierung)*
    - *Zusätzlich auch schlechtere Lösungen zulassen (Diversifikation)*
    - *=> Wahrscheinlichkeit der Auswahl*
  - Konvergenz des Verfahrens wird beeinflusst
  - Mutation kann Einfluss nehmen
- Kennzahl „selective pressure“ als Maß für Zeit, nach der ein Individuum die Population dominiert
  - „selective pressure“ groß => „takeover time“ klein
  - => Weitere Diversifikation wird verhindert

Beispiel: Roulette-Selektion

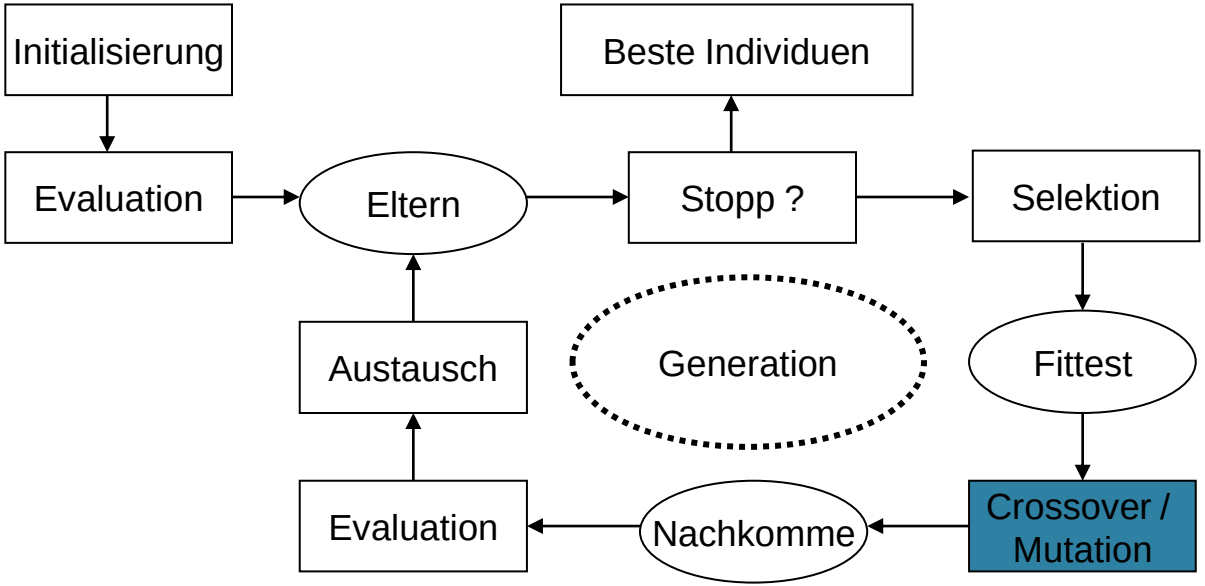
- Roulette-Selektion
  - Anteil am Rad ist proportional zur Fitness (Zielfunktionswert)
    - Berechnung:
      Anteil (Individuum) = Zielfunktionswert (Individuum) / Summe (Alle Zielfunktionswerte)
  - Probleme
    - Minimierung nicht möglich => Transformation in Max.-Problem
    - „Supersolutions“ nehmen größten Wert auf Rad ein (keine Diversifikation)
    - Zufällige Auswahl bei gleicher Fitness von Lösungen
  - Beispiel :

Individuum	Zielfunktionswert
1	43
2	27
3	18
4	12

Summe: 100



Crossover und Mutation



Crossover und Mutation

Crossover (Kreuzung, Rekombination):

Zufällige Rekombination zweier Elternlösungen zu einem oder zwei Nachkommen:

Eltern:

Nachkommen:

Mutation

Nach dem Crossover werden die einzelnen Nachkommen an einer oder mehreren Stellen zufallsabhängig verändert, Änderungshäufigkeit abhängig von Mutationsrate

Fragestellung:

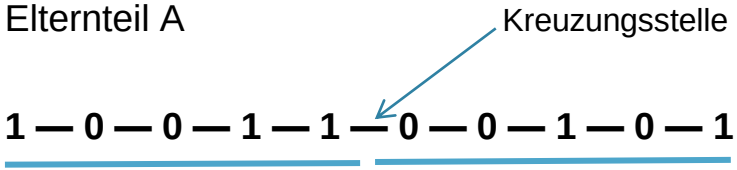
Ist die durch Crossover / Mutation Lösung zulässig?

→ Unzulässige Lösungen entweder verwerfen oder reparieren oder:

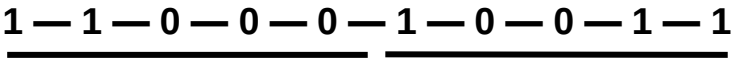
→ Crossover / Mutation direkt so ausgestalten, dass nur zulässige Lösungen entstehen

Beispiel: Rekombination bei binärer Codierung

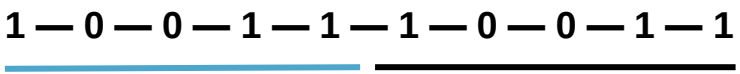
Elternteil A



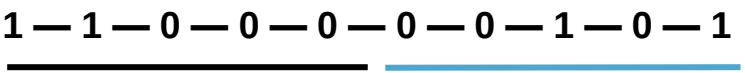
Elternteil B



Nachkomme A



Nachkomme B



Beispiel: Mutation bei binärer Codierung

Nach Crossover:

Nachkomme A

1 — 0 — 0 — 1 — 1 — 1 — 0 — 0 — 1 — 1

Nachkomme B

1 — 1 — 0 — 0 — 0 — 0 — 0 — 1 — 0 — 1

Mutation:

Zufallsabhängiges Kippen von einem oder mehreren Bits, z.B.:

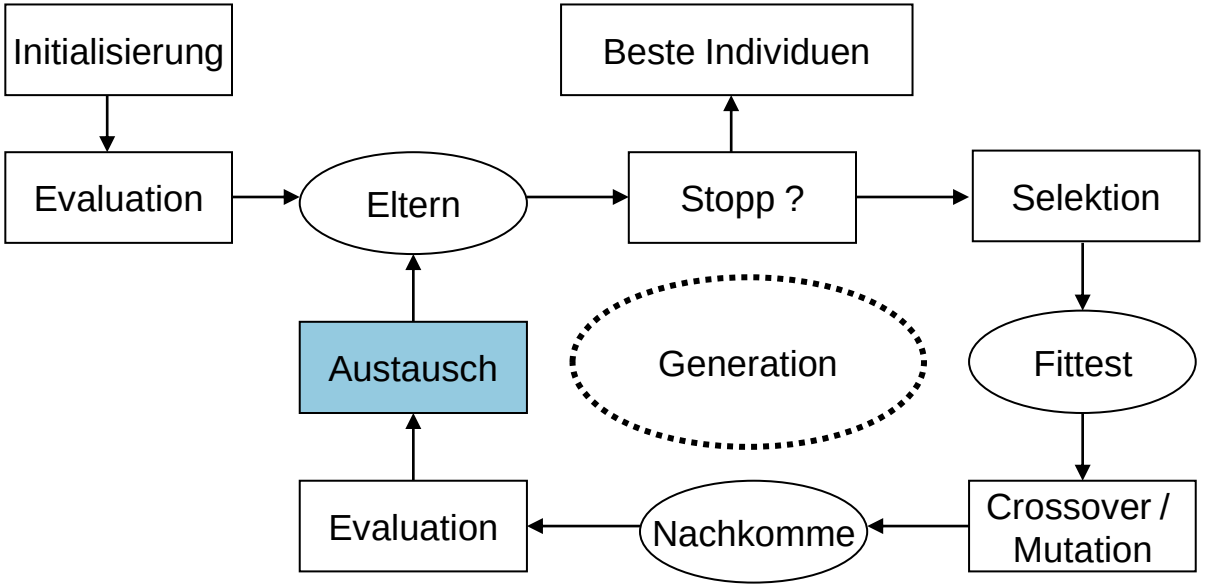
Nachkomme A nach Mutation

1 — 0 — 0 — 1 — 1 — 1 — 0 — 1 — 1 — 1

Nachkomme B nach Mutation

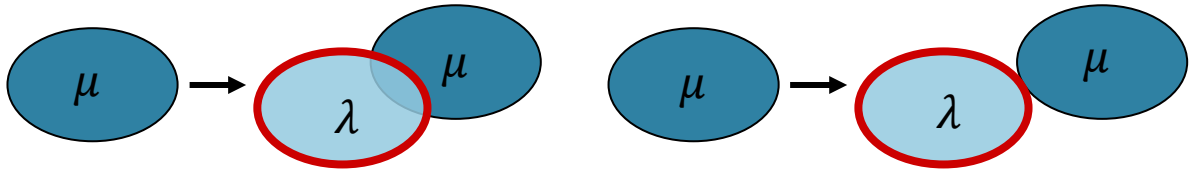
1 — 1 — 0 — 1 — 0 — 0 — 0 — 0 — 0 — 1

Austausch



Populationsmanagement

- Gesucht: Mögliche Strategien um Population zu bestimmen für
  - Vermehrung und
  - Löschung von Individuen
- Populationsmanagement („generation gap methods“)
  - Überlappende $(\mu + \lambda)$
  - Nicht überlappende Generationen (μ, λ)



- 2 Ausprägungen:
  - Generational systems: Ganze Population in jeder Generation auswechseln
  - Elitist strategies: Lebenszeit der Individuen an Fitness gekoppelt

Rucksackproblem: Genetischer Algorithmus (I)

Gegeben: ein Rucksackproblem mit $\boldsymbol{n}$ Objekten

Festzulegen: Anzahl $\boldsymbol{m}$ (gerade) der Individuen einer Generation; Mutationsrate P_M

Initialisierung: Zufällige Generierung der Generation $i = 0$ ($\boldsymbol{m}$ Lösungen)

Iteriere bis Abbruchkriterium erfüllt ist (z.B. max. Anzahl Iterationen)

a) Selektion

Berechne Reproduktionswahrscheinlichkeiten der Lösungen auf Basis der ZF-Werte
Wähle auf dieser Basis dieser Wahrscheinlichkeiten mit Hilfe von [0,1]-Zufallszahlen **m** Eltern aus Generation i zur Kreuzung aus (eine Lösung kann mehrfach Elternteil sein)

Rucksackproblem: Genetischer Algorithmus (II)

b) Kreuzung (Rekombination)

Kreuze jeweils 2 der Elternlösungen, bestimme dabei jeweils mit Hilfe von [0,1]- Zufallszahlen die Kreuzungsstelle ($n - 1$ mögliche Kreuzungsstellen)

→ Es entsteht Generation $i + 1$

c) Mutation

Für jedes der $(m * n)$ Bits in der neuen Generation $i + 1$:

Ziehe eine [0,1]- Zufallszahl z; wenn $z > 1 - PM$: Invertiere das Bit

Rucksackproblem: Genetischer Algorithmus (III)

Max $3x_1 + 4x_2 + 2x_3 + 3x_4$

u.d.N. $3x_1 + 2x_2 + 4x_3 + x_4 \leq 9$

$x_i \in \{0, 1\}$

Verfahrensschritte:

Initialisierung / Iteration (Generation) 0:
Zufällige Auswahl von 4 zulässigen Lösungen

x	ZF
(1, 0, 1, 0)	5
(0, 0, 0, 1)	3
(0, 1, 0, 0)	4
(0, 0, 1, 1)	5

Iteration (Generation) 1 – Selektion

Berechnung der Reproduktionswahrscheinlichkeiten und der Intervalle zur Zufallsauswahl

x	ZF	P_R	Intervall
(1, 0, 1, 0)	5	0,294	[0,000; 0,294]
(0, 0, 0, 1)	3	0,176	]0,294; 0,471]
(0, 1, 0, 0)	4	0,235	]0,471; 0,706]
(0, 0, 1, 1)	5	0,294	]0,706; 1,000]
Summe	**17**	**1,000**	

P_R: Reproduktionswahrscheinlichkeit

$P_R = f(x) / \sum f(x)$
Maß für die „Fitness“

Zerlegung des Einheitsintervalls in disjunkte Teilintervalle **unter Berücksichtigung der Reproduktionswahrscheinlichkeiten**

Iteration (Generation) 1 – Selektion

x	ZF	P_R	Intervall	
(1, 0, 1, 0)	5	0,294	[0,000; 0,294]	0,21; 0,15; 0,27
(0, 0, 0, 1)	3	0,176	]0,294; 0,471]	
(0, 1, 0, 0)	4	0,235	]0,471; 0,706]	
(0, 0, 1, 1)	5	0,294	]0,706; 1,000]	0,82
Summe	**17**	**1,000**		

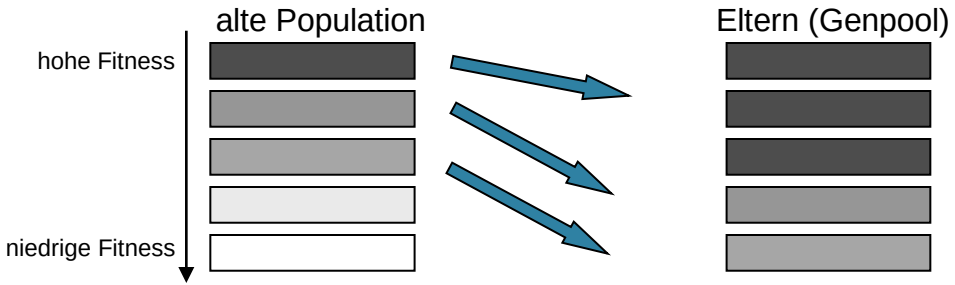
Bildung neuer Elternteile:

- Ziehung von 4 Zufallszahlen im Intervall [0; 1]:
  0,21; 0,82; 0,15; 0,27
- Zuordnung der Zufallszahlen zu den Teilintervallen
  $0{,}21 \in [0; 0{,}294]$, $0{,}82 \in]0{,}706; 1]$, $0{,}15 \in [0; 0{,}294]$, $0{,}27 \in [0; 0{,}294]$

$\Rightarrow$ **Elternteile x = (1, 0, 1, 0), x = (0, 0, 1, 1), x = (1, 0, 1, 0), x = (1, 0, 1, 0)**

Bemerkung zur Selektion

Aufgrund der Berücksichtigung der Reproduktionswahrscheinlichkeiten ist es möglich, dass der Genpool von Individuen mit hoher Fitness u.U. mehrere „Kopien“ enthält, während Individuen mit geringer Fitness nicht enthalten sein können.



Iteration (Generation) 1 – Kreuzung

Ermittlung von 2 Kreuzungsstellen $\ell_1, \ell_2 \in \{1, 2, 3\}$

- Zerlegung des Einheitsintervalls in die gleichverteilten disjunkten Teilintervalle

  [0; 0,333] ,]0,333; 0,666] ,]0,666; 1]
- Ziehung von 2 Zufallszahlen im Intervall [0; 1]: 0,14; 0,73
- Zuordnung der Zufallszahlen zu den Teilintervallen: 0,14 $\in$ [0; 0,333], 0,73 $\in$]0,666; 1]

$\Rightarrow$ Kreuzungsstellen $\ell_1 = 1$, $\ell_2 = 3$

Kreuzung der ersten beiden Eltern - Lösungen nach der 1. Stelle

1 | 0 1 0	$\Rightarrow$	1 | 0 1 1
0 | 0 1 1		0 | 0 1 0

Kreuzung der beiden letzten Eltern - Lösungen nach der 3. Stelle

1 0 1 | 0	$\Rightarrow$	1 0 1 | 0
1 0 1 | 0		1 0 1 | 0

Nachkommen: x = (1, 0, 1, 1), x = (0, 0, 1, 0), x = (1, 0, 1, 0), x = (1, 0, 1, 0)

Iteration (Generation) 1 – Mutation

Simulation von Fehlern bei der Codierung der Abkömmlinge

Annahme: Fehlerrate P_M = 0,02

Erzeugung von 16 Zufallszahlen im Intervall [0; 1]
Korrektur des zugehörigen Bits, falls Zufallszahl > 1-0,02 = 0,98

Beispiel: Zufallszahlen:

0,343 ; 0,432 ; 0,675 ; 0,159 ; 0,973 ; 0,295 ; 0,583 ; 0,871

0,048 ; 0,774 ; 0,124 ; 0,479 ; 0,635 ; 0,993 ; 0,726 ; 0,032

→ nur Zufallszahl 14 = 0,993 > 0,98

→ Mutation an 14. Stelle, d.h. 2. Bit in Nachkomme 4

x = (1, 0, 1, 1), x = (0, 0, 1, 0), x = (1, 0, 1, 0), x = (1, 0, 1, 0) ⇒

x = (1, 0, 1, 1), x = (0, 0, 1, 0), x = (1, 0, 1, 0), x = (1, 1, 1, 0)

Iteration (Generation) 1 – Ergebnis (neue Population)

x = (1, 0, 1, 1), x = (0, 0, 1, 0), x = (1, 0, 1, 0), x = (1, 1, 1, 0)

x	ZF
(1,0,1,1)	8
(0,0,1,0)	2
(1,0,1,0)	5
(1,1,1,0)	9
Summe	**24**

Erzeugung der Generationen i = 2, 3, ... Analog

Abbruch:

- z.B. falls über 3 Generationen keine besseren Nachkommen erzeugt wurden
- Maximalzahl untersuchter Generationen

Zusammenfassung

- Um lokale Optima verlassen zu können, müssen Züge erlaubt sein, die eine zwischenzeitliche Verschlechterung des Zielfunktionswertes zulassen $\Rightarrow$ Metaheuristiken

- Tabu Suche:
  - Absuchen der Nachbarschaft
  - Verbotene Züge werden in sog. Tabu-Listen gespeichert

- Genetischer Algorithmus:
  - Simulation von Darwins Satz „survival of the fittest“
  - Selektion, Crossover/Mutation, Evaluation

Zusammenfassung

Vorlesung Operations Reserach

Zusammenfassung Operations Reserach Vorlesung

- Problemstellungen in lineare mathematische Modelle überführen
- Das resultierende Gleichungssystem wird durch iteratives Umformen gelöst
- Die notwendigen Umformungen durch den Simplex-Algorithmus formuliert
- Die entstehenden Gleichungssysteme können ökonomisch interpretiert werden
- Diskrete Entscheidungen müssen über ganzzahlige Variablen abgebildet werden
- Für ganzzahlige Entscheidungsprobleme wird das Branch&Bound-Verfahren verwendet

- Die graphentheoretische Modellierung stellt eine Alternative dar
- Standardproblemstellungen können durch effiziente Algorithmen gelöst werden
- Komplexe Probleme werden auch mit Hilfe der Graphentheorie nicht effizient gelöst
- Problemspezifische Heuristiken generieren gute Lösungen in absehbarer Zeit
- Metaheuristiken verbessern die Lösungsqualität von Heuristiken noch einmal