Algorithmen für komplexe Probleme

Stefan Näher

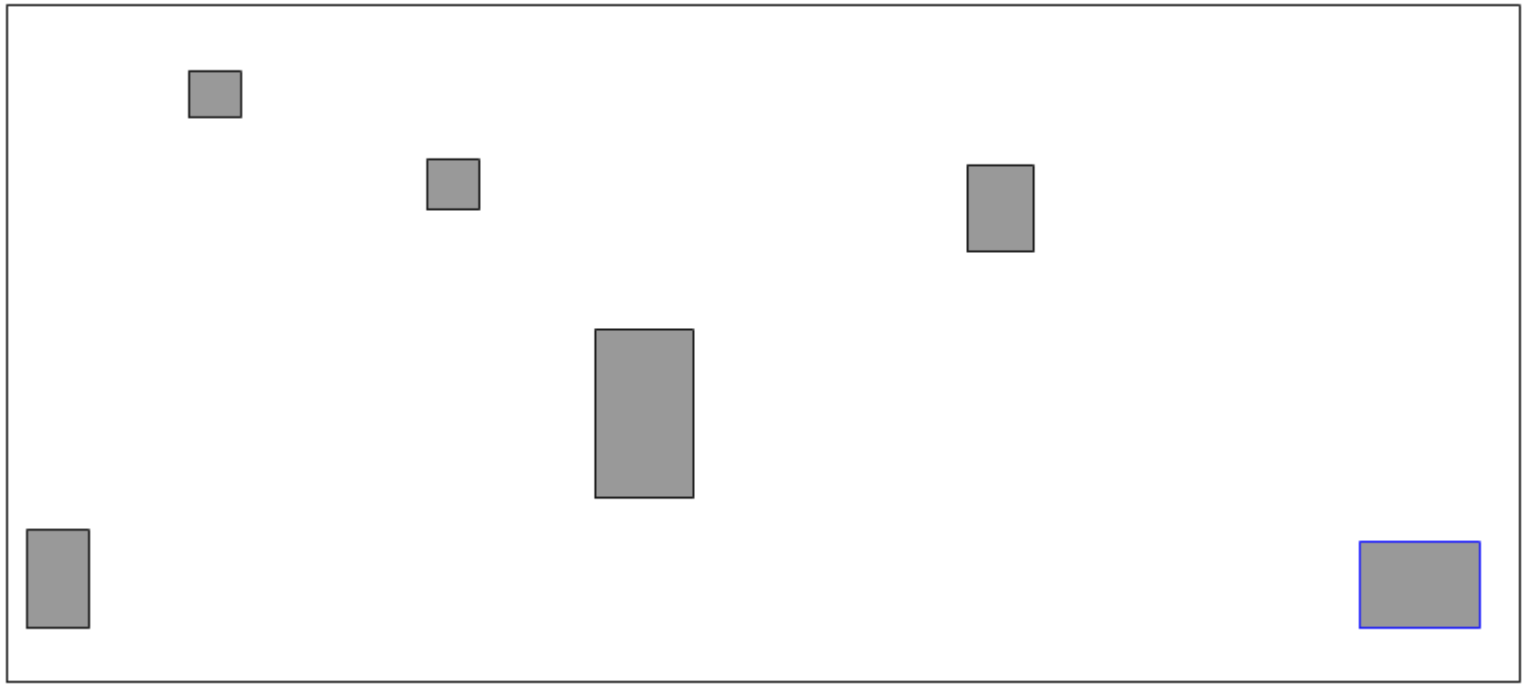
FB IV – Informatikwissenschaften
Universität Trier
naeher@uni-trier.de

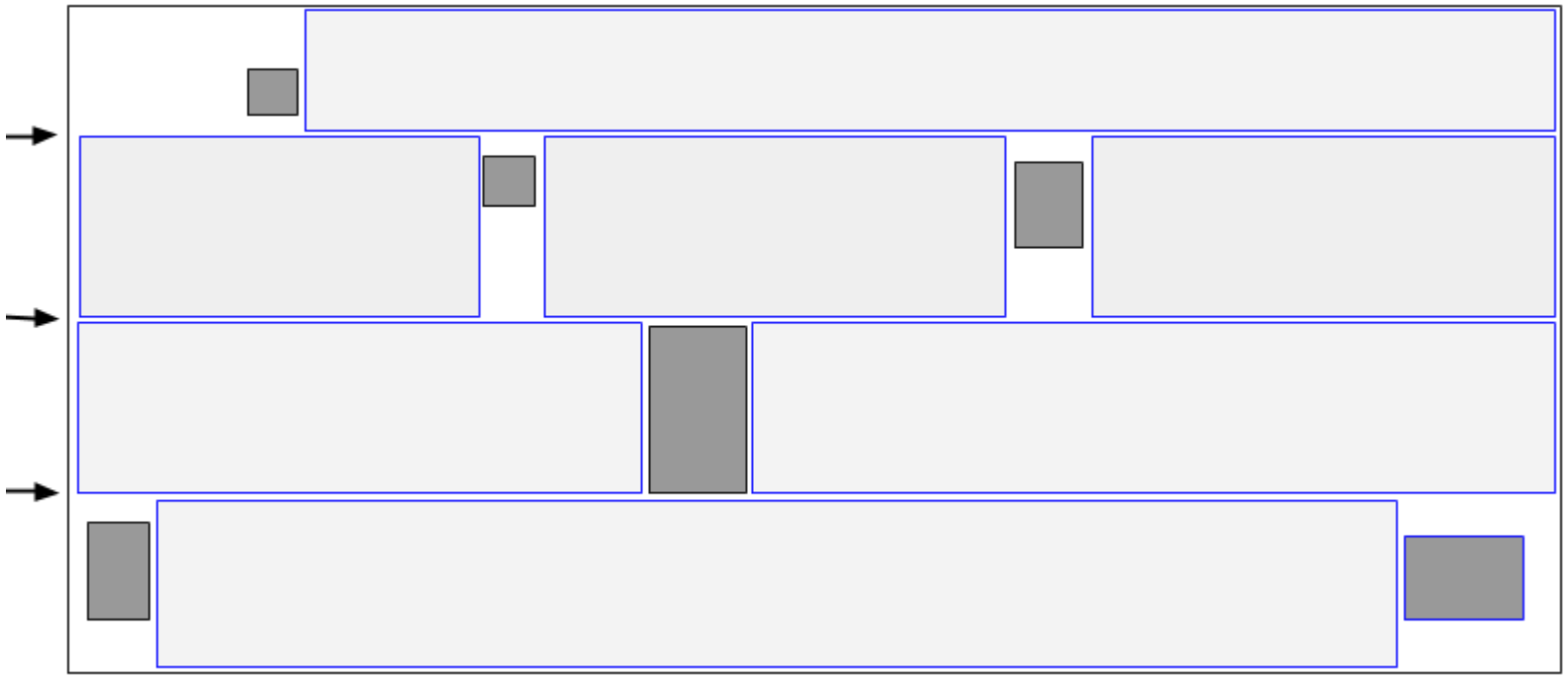
1. Korrektheit
2. Effizienz
3. Parallelität (Multi-Core)
4. Implementierung (Algorithm Engineering)

Probleme

1. Schnittprobleme
2. Rekonstruktion von Karten
2. Problem des Handlungsreisenden

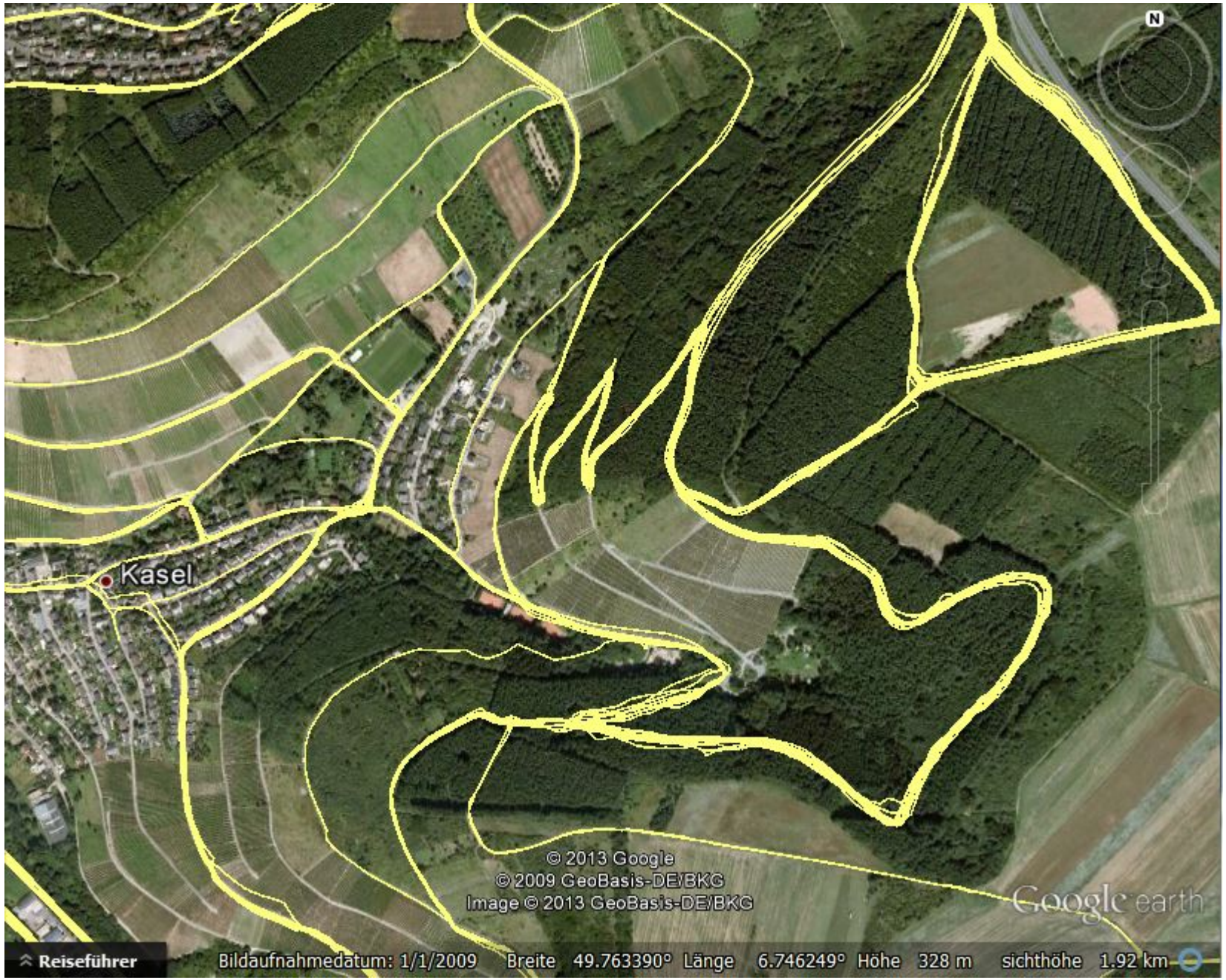
Schnittprobleme
(z.B. Holzverarbeitung)

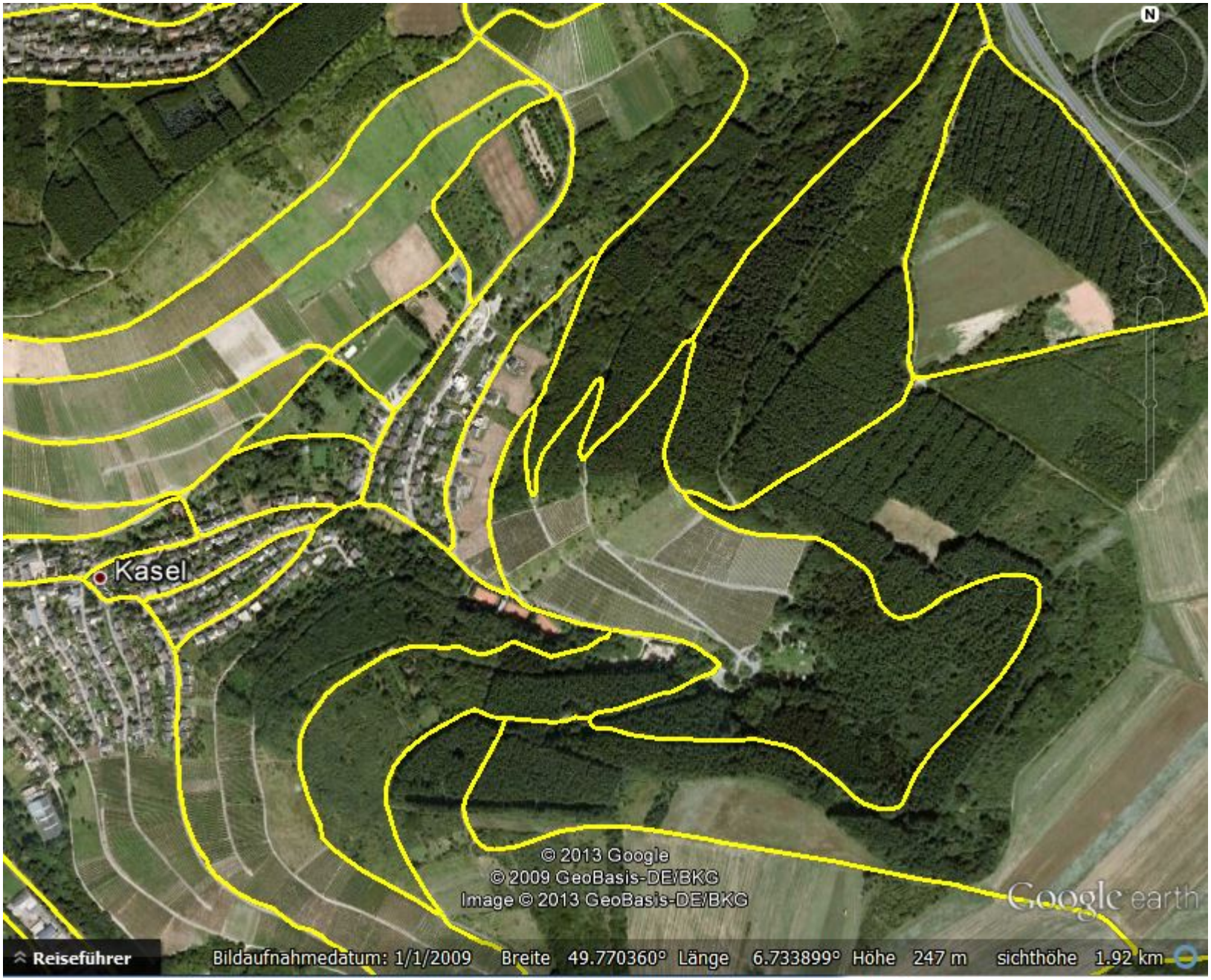




Map Reconstruction


Ruwer
© 2009 GeoBasis-DE/BKG
© 2013 Google
Image © 2013 GeoBasis-DE/BKG


N
Kasel
© 2013 Google
© 2009 GeoBasis-DE/BKG
Image © 2013 GeoBasis-DE/BKG
Google earth
Reiseführer
Bildaufnahmedatum: 1/1/2009 Breite 49.763390° Länge 6.746249° Höhe 328 m sichthöhe 1.92 km


N
Kasel
© 2013 Google
© 2009 GeoBasis-DE/BKG
Image © 2013 GeoBasis-DE/BKG
Google earth
Reiseführer
Bildaufnahmedatum: 1/1/2009
Breite 49.770360° Länge 6.733899° Höhe 247 m
sichthöhe 1.92 km

Das Problem des Handlungsreisenden

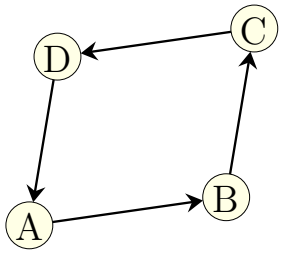
Die Brute-Force Methode

betrachtet **alle** möglichen Rundreisen und sucht die kürzeste.

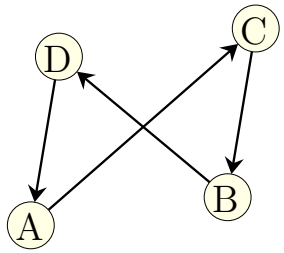
Zahl der Rundreisen für n Städte $= (n-1)\,!$

$$= 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot (n-1)$$

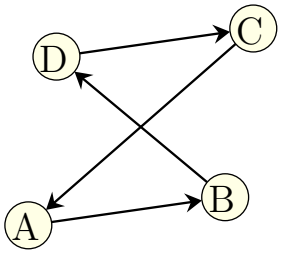
Zahl der Rundreisen für 4 Städte: $\quad 1 \cdot 2 \cdot 3 = 6$



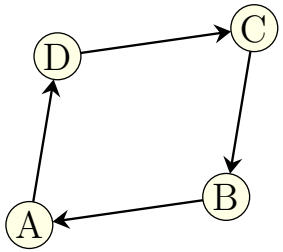
1: A–B–C–D–A



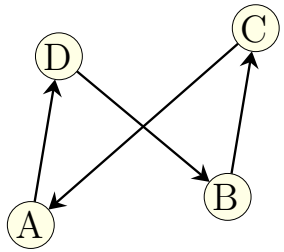
2: A–C–B–D–A



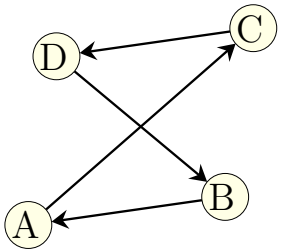
3: A–B–D–C–A



4: A–D–C–B–A



5: A–D–B–C–A



6: A–C–D–B–A

Anzahl der möglichen Rundreisen und Laufzeit

n	$\frac{1}{2} \cdot (n-1)!$ Rundreisen	Laufzeit
3	1	1 msec
4	3	3 msec
5	12	12 msec
6	60	60 msec
7	360	360 msec
8	2.520	2,5 sec
9	20.160	20 sec
10	181.440	3 min
11	1.814.400	0,5 Stunden
12	19.958.400	5,5 Stunden
13	239.500.800	2,8 Tage
14	3.113.510.400	36 Tage
15	43.589.145.600	1,4 Jahre
16	653.837.184.000	**21 Jahre**

Dynamisches Programmieren

$L(i, A)$ = die Länge eines kürzesten Weges, der

1. in der Stadt i beginnt,
2. jede Stadt in A einmal besucht,
3. in Stadt 1 endet.
4. $L(1, \{\ 2, \ldots, n\ \})$ ist die Länge einer optimalen Reise.

Berechne die L-Werte in einer Tabelle für immer größere Mengen A.

- maximal $n \cdot 2^n$ Einträge,
- pro Eintrag n Rechenschritte,
- also insgesamt $n^2 \cdot 2^n$ Schritte.

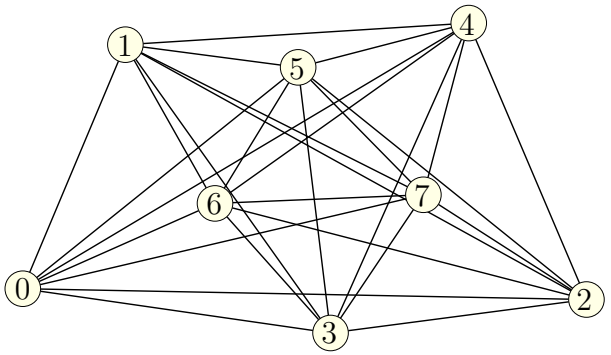
Dynamisches Programmieren

n	$n^2 \cdot 2^n$ Schritte	Laufzeit	
3	72	72	msec
4	256	0,4	sec
5	800	0,8	sec
6	2304	2,3	sec
7	6272	6,3	sec
8	16.384	16	sec
9	41.472	41	sec
10	102.400	102	sec
11	247.808	4,1	Minuten
12	589.824	9,8	Minuten
13	1.384.448	23	Minuten
14	3.211.264	54	Minuten
15	7.372.800	2	Stunden
16	16.777.216	**4,7**	**Stunden**

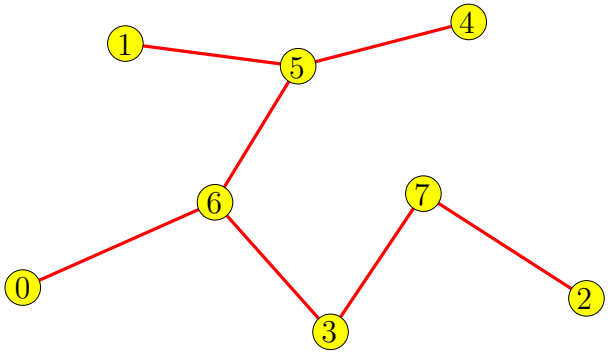
Näherungslösungen oder Heuristiken

finden eine **gute** aber nicht immer optimale Rundreise.

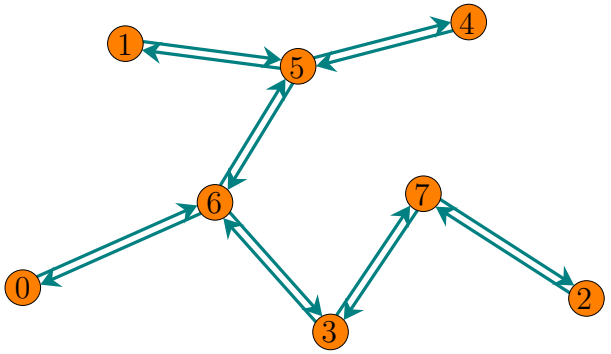
1. Der vollständige Graph aller direkten Reisewege



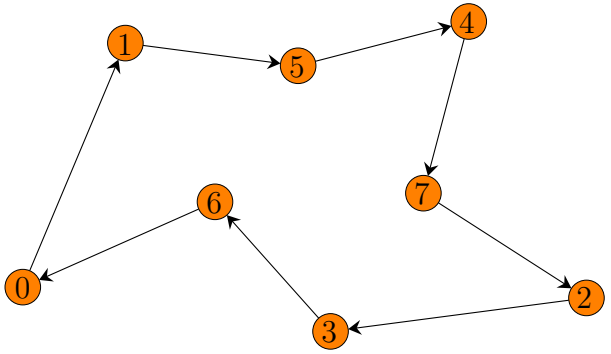
2. Der minimal aufspannende Baum



3. Einmal um den Minimum-Spanning-Tree



4. Die Minimum-Spanning-Tour



Laufzeit der MST-Heuristik

Städte	Laufzeit
100	0,01 sec
200	0,08 sec
300	0,36 sec
400	1,30 sec
500	3,62 sec
600	8,27 sec
700	16,07 sec
800	29,35 sec
900	50,22 sec
1000	85,38 sec

Weitere Informationen und Links

- Stefan Näher
  `www.informatik.uni-trier.de/~naeher`

- LEDA Buch
  `www.people.mpi-inf.mpg.de/~mehlhorn/LEDAbook`

- Algorithmic Solutions
  `www.algorithmic-solutions.com`