Kap. 7 Optimierung

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

Fakultät für Informatik, TU Dortmund

22. VO *2. TEIL* DAP2 SS 2009 9. Juli 2009

Überblick

- Einführung
  - Einige klassische Optimierungsprobleme,
  - z.B. TSP

- Heuristiken
  - Greedy-Heuristiken für TSP, Bin Packing, Rucksack-Problem

Motivation

„Warum soll ich heute hier bleiben?“

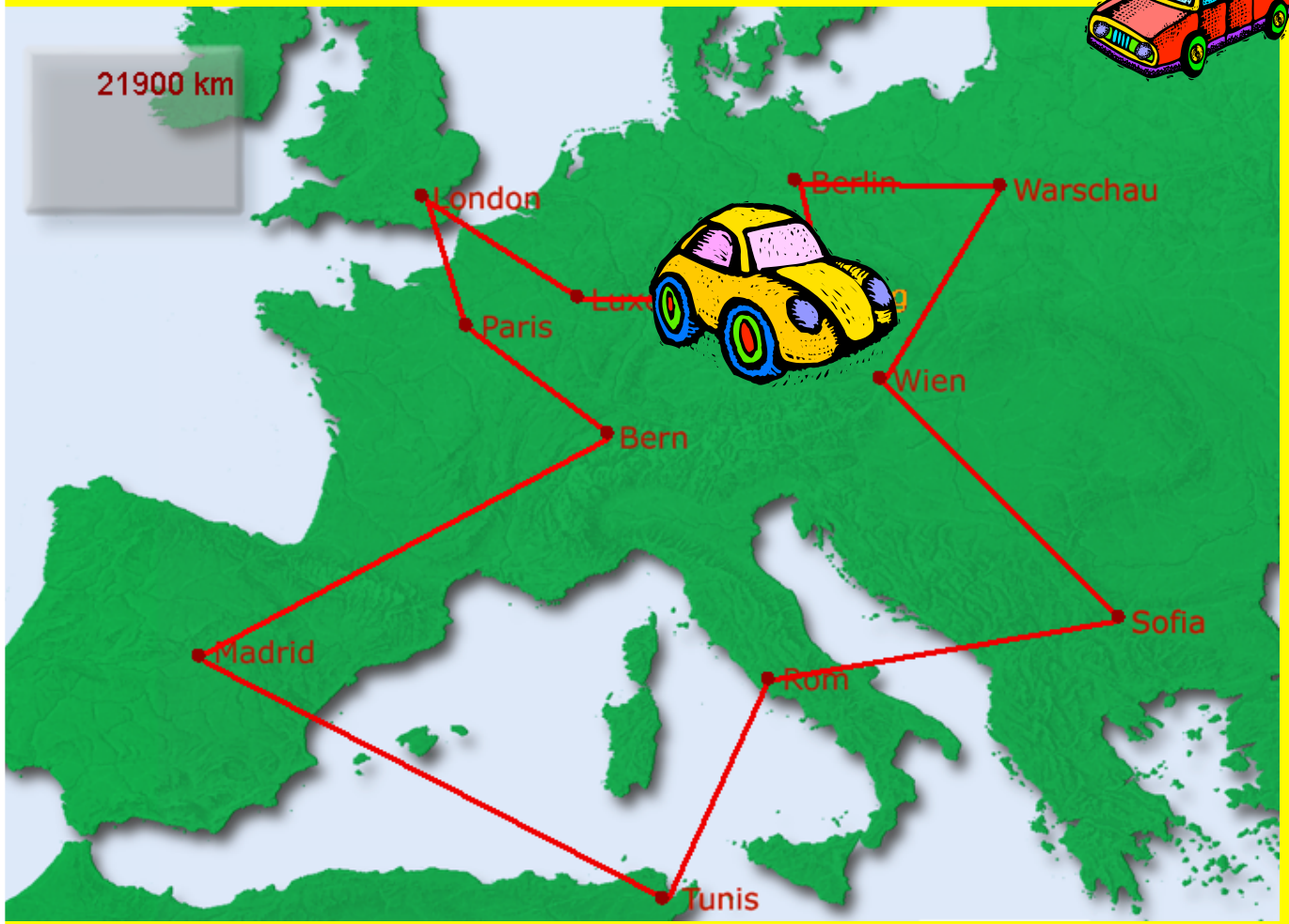
Optimierung ist Klasse!

keine zusätzliche Motivation notwendig ☺

Optimierung: Einführung

- Optimierungsprobleme sind Probleme, die i.A. viele zulässige Lösungen besitzen
- Jeder Lösung ist ein bestimmter Wert (Zielfunktionswert, Kosten) zugeordnet.
- Optimierungsalgorithmen suchen in der Menge aller zulässigen Lösungen diejenige mit dem optimalen Wert („die beste“ bzgl. Zielfunktion)
- Heuristiken sind Algorithmen, die irgendeine zulässige Lösung berechnen

Rundreiseprobleme (TSP)



Bei 12 Städten gibt es 19.958.400 viele verschiedene Touren.

Du möchtest in Deinem Urlaub verschiedene Städte Europas besuchen und nicht zu viel Zeit im Auto verbringen.

DEIN ZIEL:
Finde die kürzeste Rundtour durch alle Städte.

Beispiele für Optimierungsprobleme

- Minimal aufspannender Baum (MST)
- Kürzeste Wege in Graphen
- Längste Wege in Graphen
- Handlungsreisendenproblem (TSP)
- Rucksackproblem
- Scheduling (z.B. Maschinen, Crew)
- Zuschneideprobleme (z.B. Bilderrahmen)
- Packungsprobleme

Optimierungsprobleme (OP)

- Wir unterscheiden zwischen
  - OP, für die wir Algorithmen mit **polynomieller** Laufzeit kennen, und
  - OP, für die noch **kein polynomieller** Algorithmus bekannt ist.
- Die Klasse der **NP-schwierigen** Optimierungs-probleme: das Finden eines polynomiellen Algorithmus für eines dieser OPs zieht automatisch polynomielle Algorithmen für alle anderen OPs dieser Klasse nach sich.
- Allgemeine Vermutung: es existieren keine polynomielle Algorithmen für NP-schwierige OPs

Optimierungsprobleme (OP)

- Ein Algorithmus hat polynomielle Laufzeit, wenn die Laufzeitfunktion $T(n)$ durch ein Polynom in n beschränkt ist.
- Dabei steht n für die Eingabegröße der Instanz.

- **Beispiel:** Kürzestes Wegeproblem: Laufzeitfunktion in $|V|+|E|$:

  $T(|V|+|E|)=O((|V|+|E|) \log |V|)=O(|V|+|E|)^2$

- **Beispiel:** Längstes Wegeproblem: NP-schwierig

  (Denn: Transformation von TSP)

Beispiele für Optimierungsprobleme

- Minimal aufspannender Baum (MST) — polynomiell
- Kürzeste Wege in Graphen — polynomiell
- Längste Wege in Graphen — NP-schwierig
- Handelsreisendenproblem (TSP) — NP-schwierig
- Rucksackproblem — NP-schwierig
- Scheduling (z.B. Maschinen, Crew) — NP-schwierig
- Zuschneideprobleme (z.B. Bilderrahmen) — NP-schwierig
- Packungsprobleme — NP-schwierig

Strategien zur Lösung von OP

- Polynomielle OP:
  - oft: speziell entwickelte Algorithmen
  - machmal auch allgemeines Strategien, wie z.B. Greedy-Algorithmen oder Dynamische Programmierung
- NP-schwierige OP:
  - Exakte Verfahren, Enumeration, Branch-and-Bound, Dynamische Programmierung
  - Heuristiken: Konstruktionsheuristiken (z.B. Greedy, approximative Algorithmen), Verbesserungsheuristiken (z.B. Local Search)

Kap. 7.1: Heuristiken

- Vorstellung typischer OP:
  - Rundreiseproblem (TSP)
  - Rucksack-Problem
  - Bin-Packing Problem

- und einfache Greedy-Heuristiken

Greedy-Algorithmen

Greedy-Verfahren treffen ***lokale*** Entscheidungen
Sie wählen in jedem Schritt die aktuell günstigste Auswahl ohne Vorausschau

Greedy-Algorithmen sind in der Regel nicht optimal, können aber in einigen wichtigen Anwendungen dennoch gute oder gar optimale Lösungen erzeugen!

Greedy-Algorithmen

Greedy-Algorithmus („gefräßig“): iterative Konstruktion einer Lösung, die immer um die momentan besten Kandidaten erweitert wird.

Greedy-Algorithmus führen

- bei manchen OP immer zu optimalen Lösungen (Bsp.: MST, SSSP)
- bei manchen OP können sie auch zu Lösungen führen, die „beliebig“ weit von der optimalen Lösung entfernt sind.

Das Travelling Salesman Problem

Auch: Handlungsreisendenproblem, Rundreiseproblem, TSP

- **Gegeben:** Vollständiger ungerichteter Graph $G=(V,E)$ mit Kantenkosten c_e
- **Gesucht:** Tour T (Kreis, der jeden Knoten genau einmal enthält) mit minimalen Kosten $c(T)=\sum c_e$

Buch von F. Voigt, Ilmenau 1831

Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein. Von einem alten Commis=Voyageur.

„ . . . Durch geeignete Auswahl und Planung der Tour kann man oft so viel Zeit sparen, daß wir einige Vorschläge zu machen haben. . . . Der Wichtigste Aspekt ist, so viele Orte wie möglich zu erreichen, ohne einen Ort zweimal zu besuchen. . . .“

Wie schwierig ist das TSP?

Anzahl der Städte	Anzahl der möglichen Touren
– 3 Städte	1 Tour
– 4 Städte	3 Touren
– 5 Städte	12 Touren
– 6 Städte	60 Touren
– 7 Städte	360 Touren
– 8 Städte	2.520 Touren
– 9 Städte	20.160 Touren
– 10 Städte	181.440 Touren
– 11 Städte	1.814.400 Touren
– 12 Städte	19.958.400 Touren

Das Travelling Salesman Problem

Ann.: Rechner schafft 40 Mio. Touren pro Sekunde

Anzahl der verschiedenen Touren: $(|V|-1)! / 2$

n	# Touren	Zeit
10	181.440	0.0045 Sek.
17	ca. 10^{13}	3 Tage
19	ca. 10^{15}	2,5 Jahre
20	ca. 10^{17}	48 Jahre
25	ca. 10^{23}	10^{8} Jahre
60		

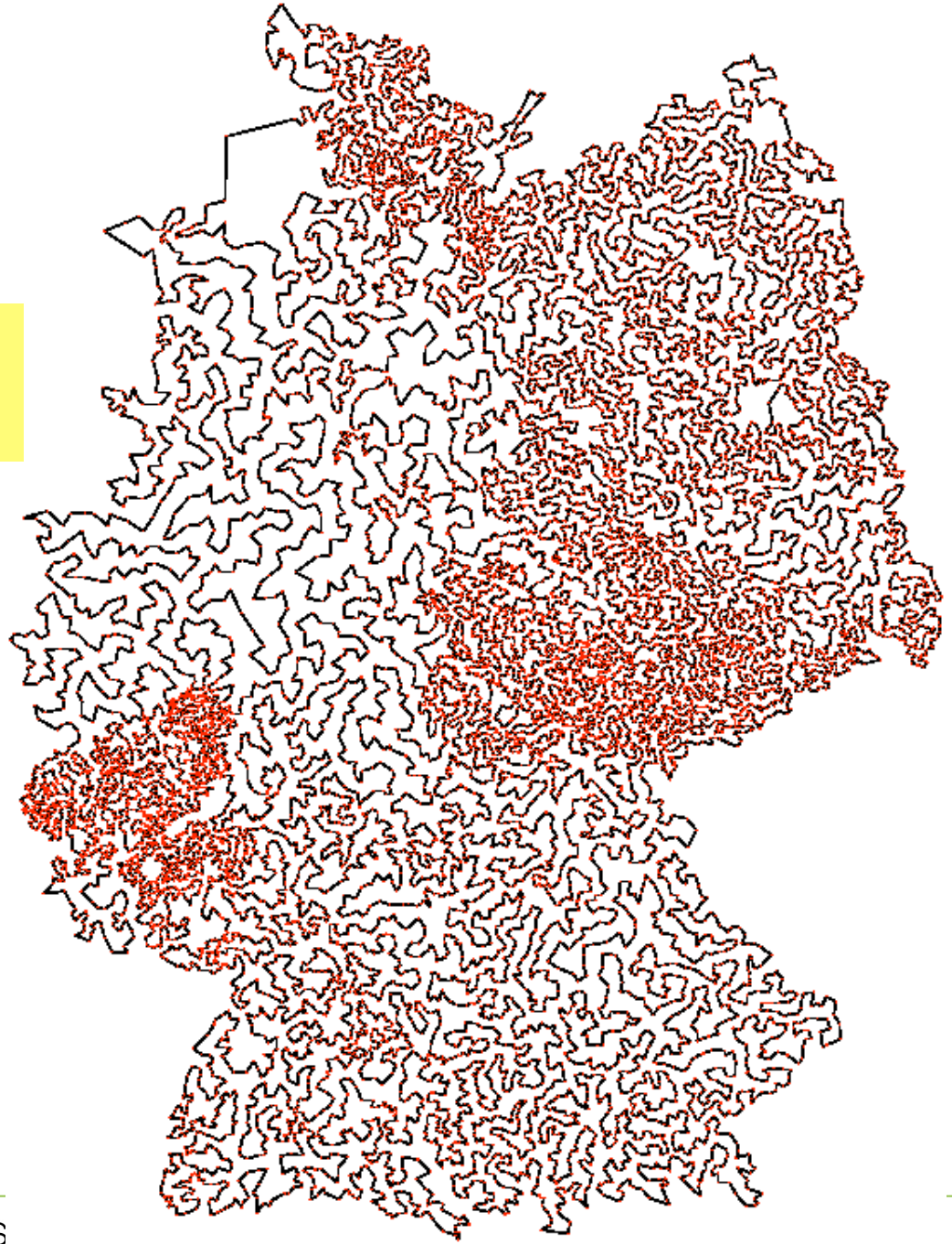
Anzahl der Atome im Weltall: ca. 10^{80}

Meilensteine für TSP Lösungen

1954	Dantzig,Fulkerson,Johnson	49
1971	Held, Karp	64
1977	Grötschel	120
1980	Crowder, Padberg	318
1987	Padberg, Rinaldi	532
1987	Grötschel, Holland	666
1987	Padberg, Rinaldi	2.392
1994	Applegate, Bixby, Chvátal, Cook	7.397
1998	Applegate, Bixby, Chvátal, Cook	13.509
2001	Applegate, Bixby, Chvátal, Cook	15.112
2004	Applegate, Bixby, Chvátal, Cook, Helsgaun	24.978
2006	Applegate, Bixby, Chvátal, Cook, Espinoza, Goycoolea, Helsgaun	85.900

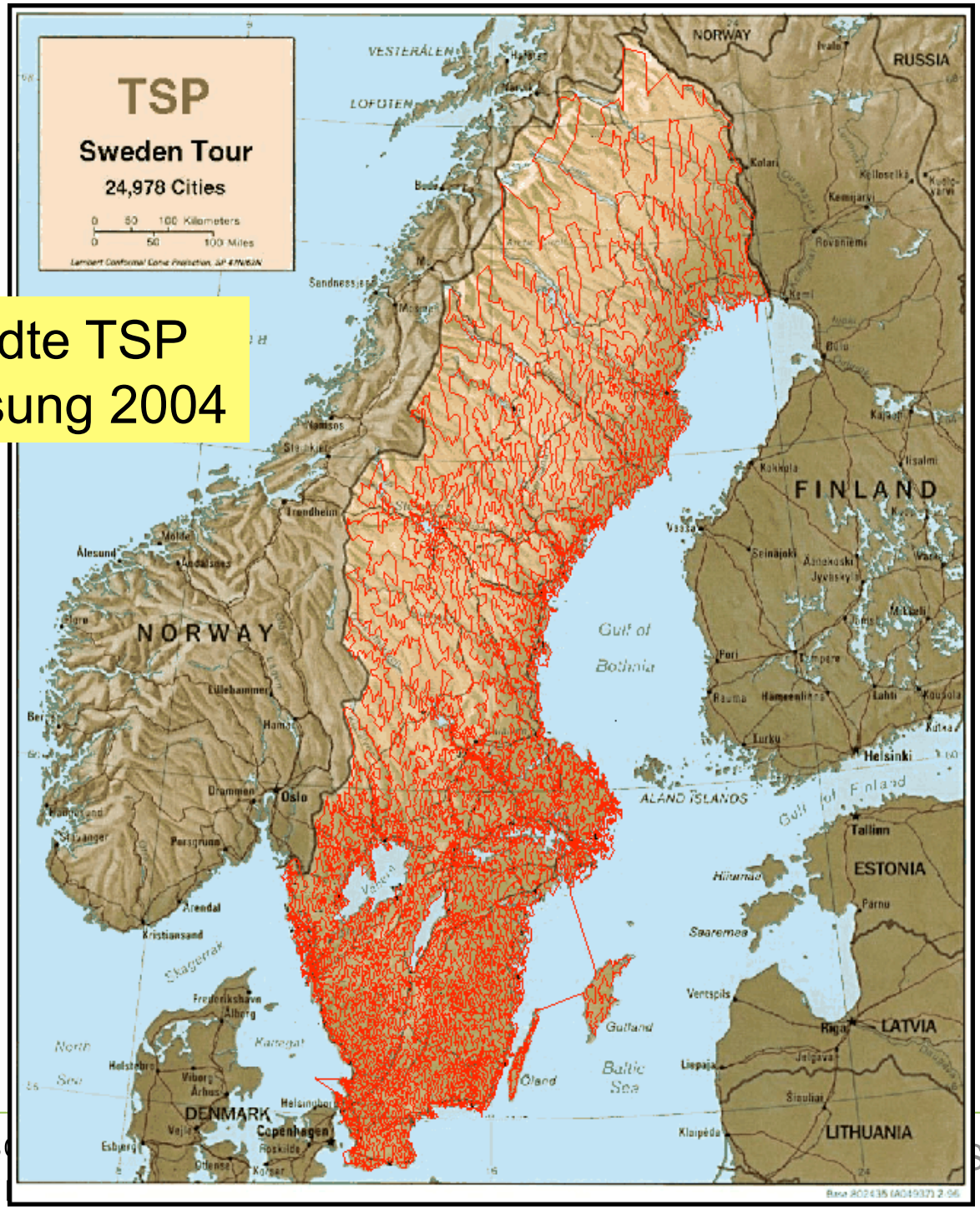
15112 Städte TSP

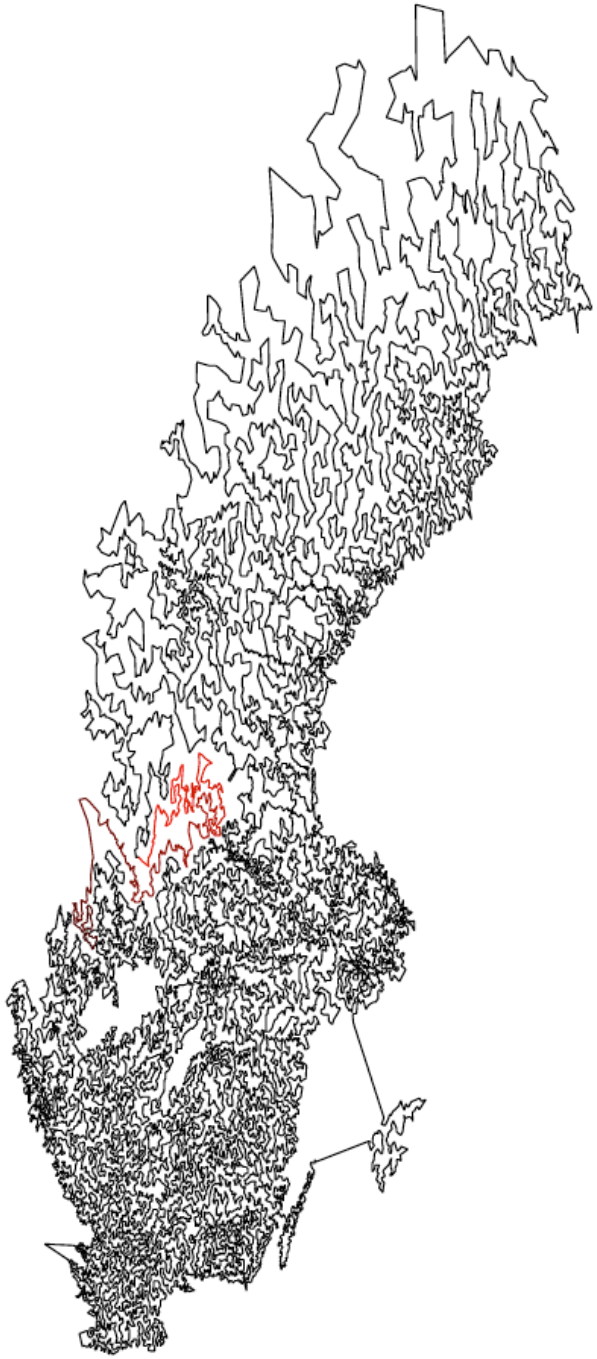
22,6 CPU Jahre
auf 110 Prozessoren



24978 Städte TSP
Optimallösung 2004

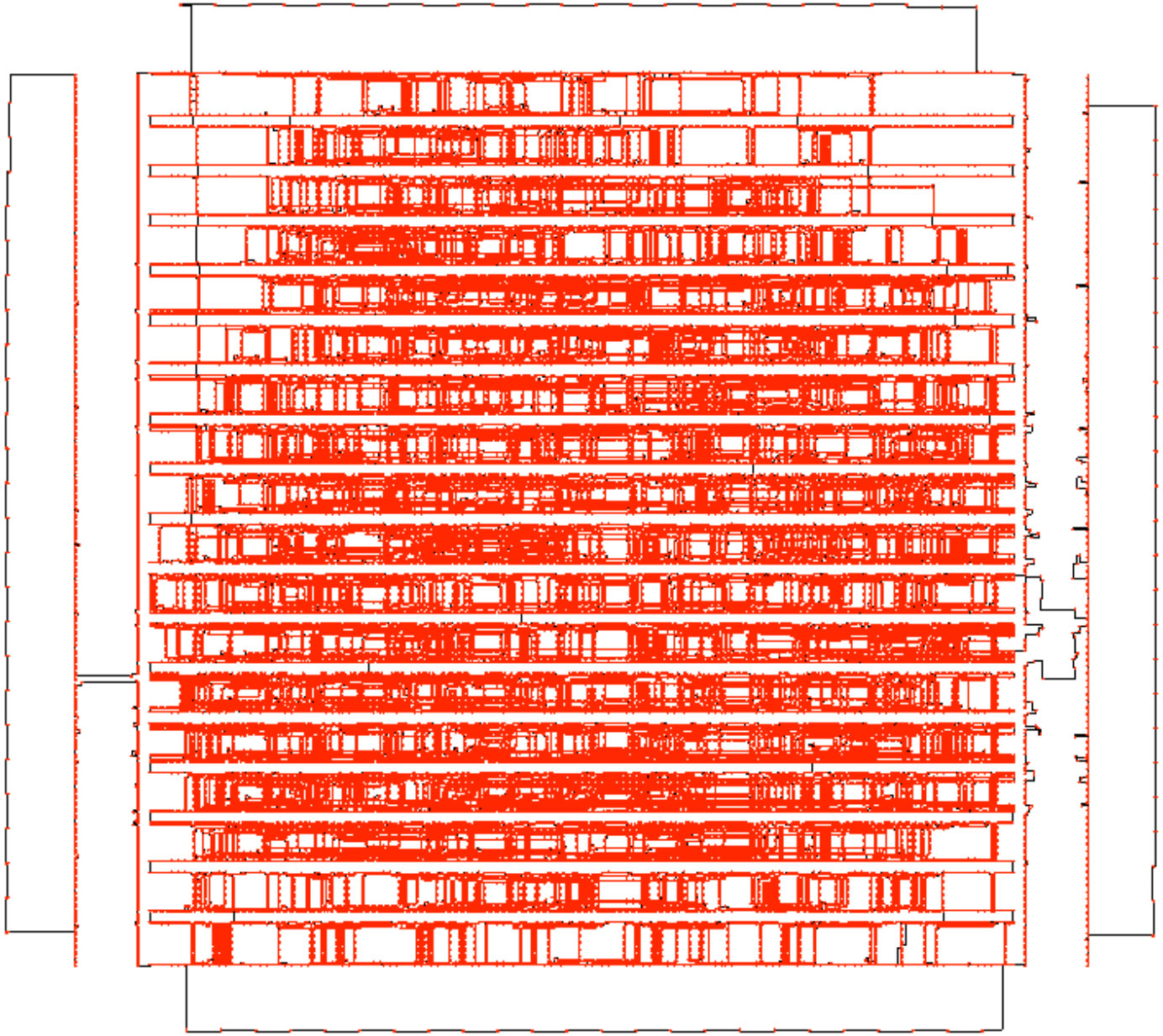
8 Jahre





- Größte gelöste TSP Instanz:

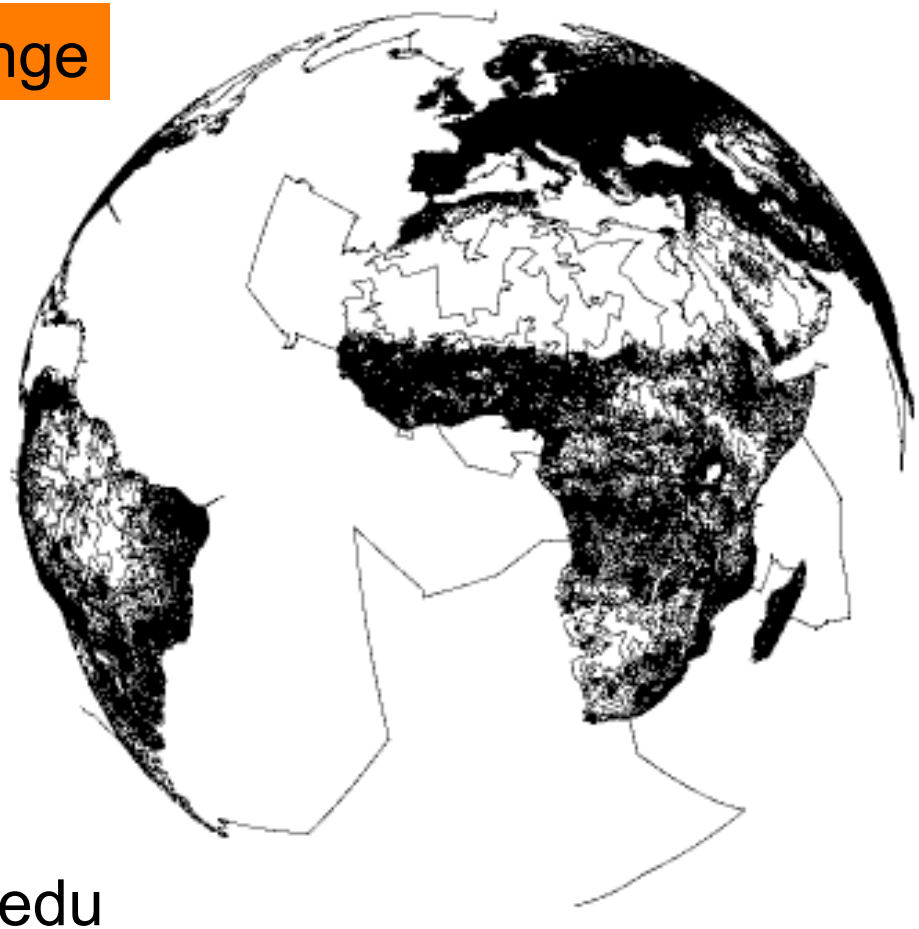
85.900 Städte TSP
Optimallösung 2006



World Tour: 1.904.711 Städte

Tour der Länge **7,515,877,991** (Keld Helsgaun, 2009)
Garantie: Tourlänge **0.0487%** nahe an OPT(Concorde TSP code)

aktuelle Challenge

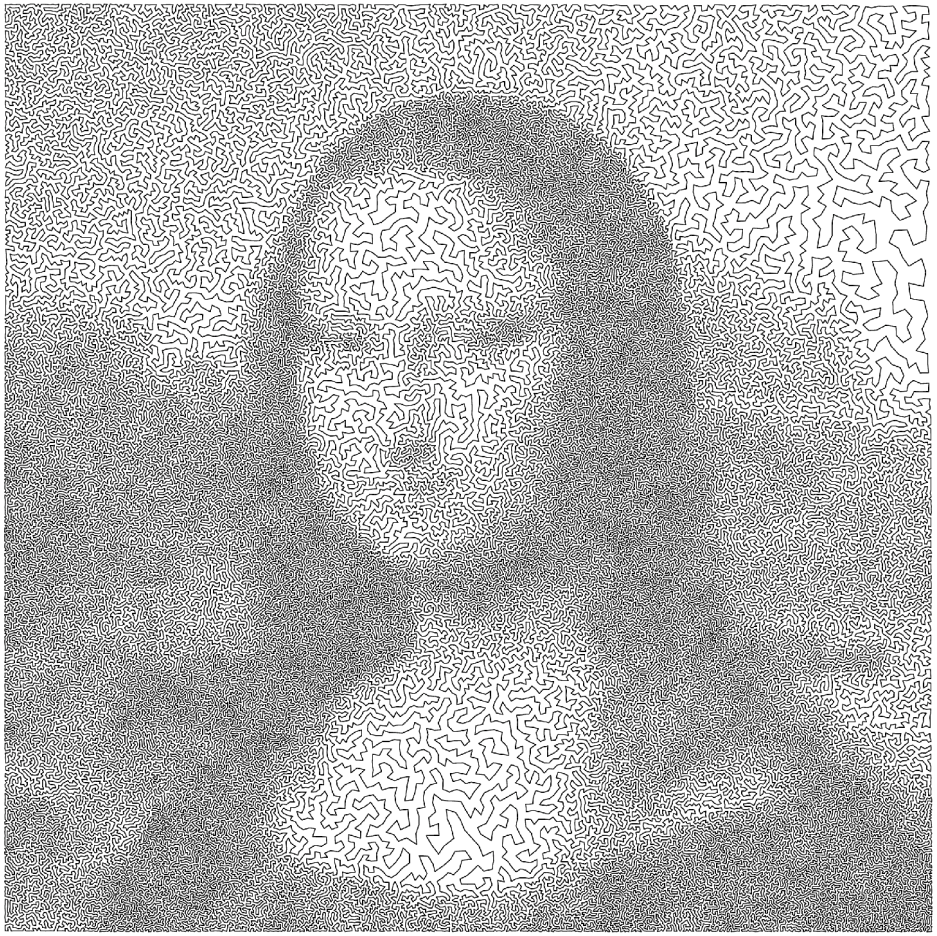


www.tsp.gatech.edu

Mona Lisa TSP Challenge: 100.000

Tour der Länge 5,757,191 (Yuichi Nagata, März 2009)
Garantie: Tourlänge 0.0032% nahe an OPT (DIFF=186)

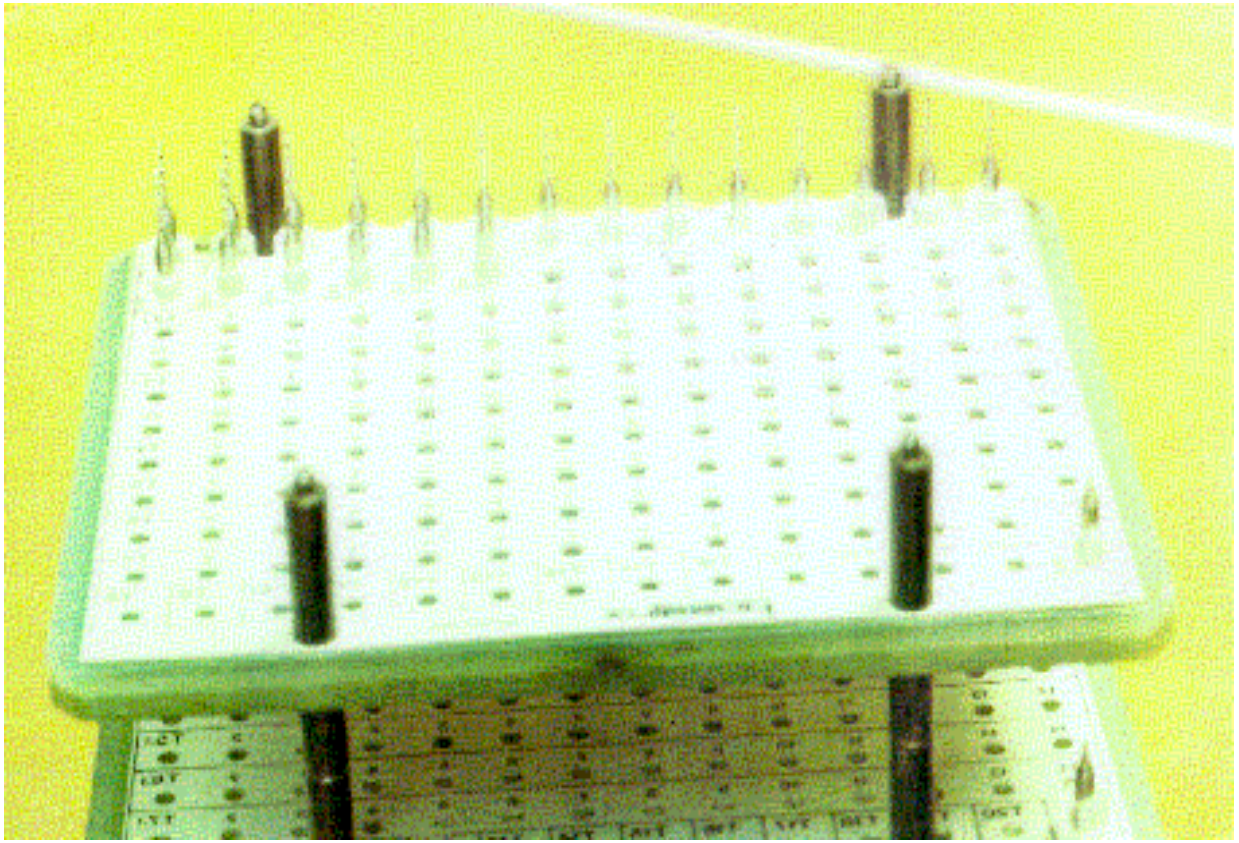
aktuelle Challenge



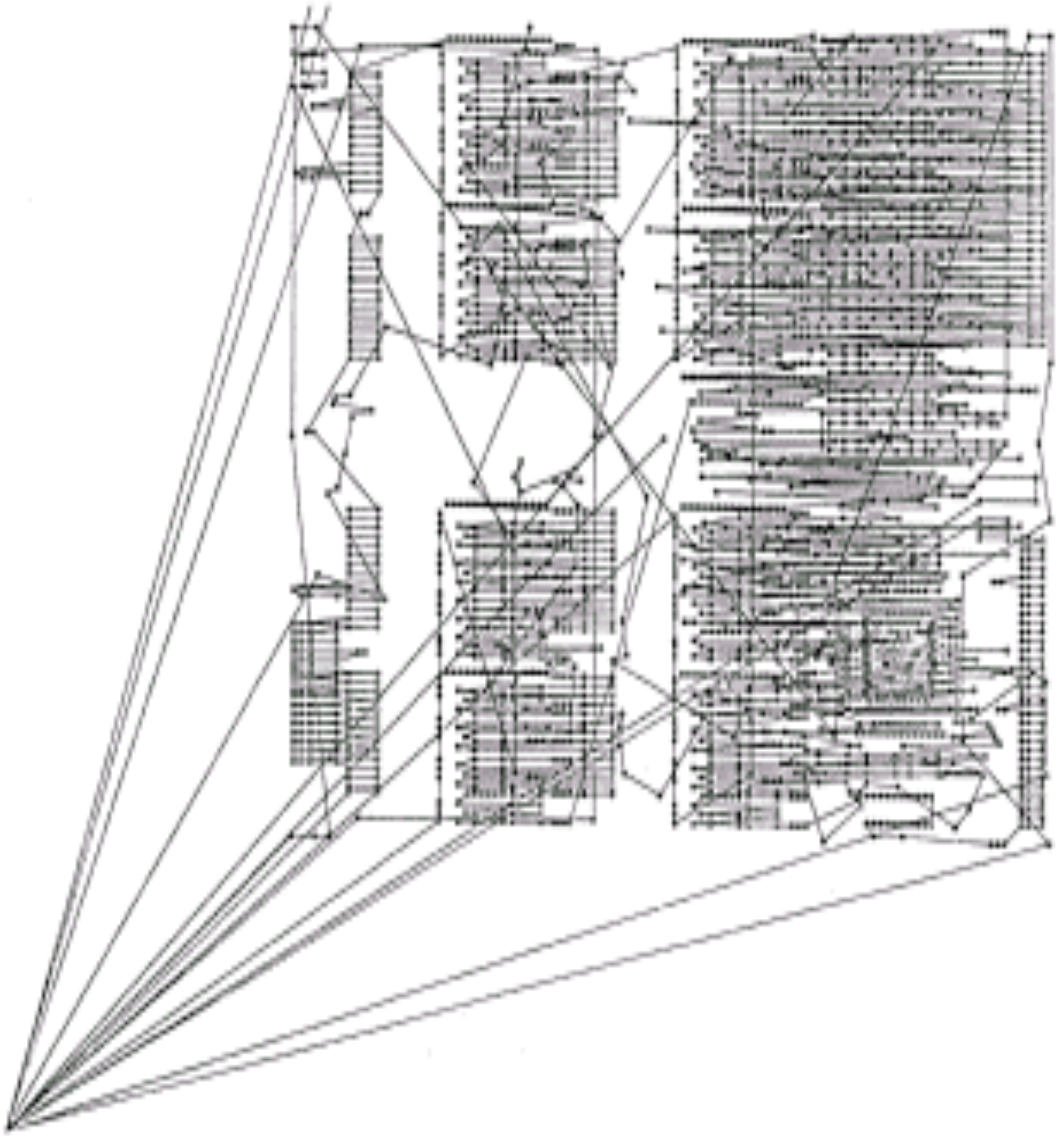
www.tsp.gatech.edu

TSP in der Praxis: Bohrmaschine

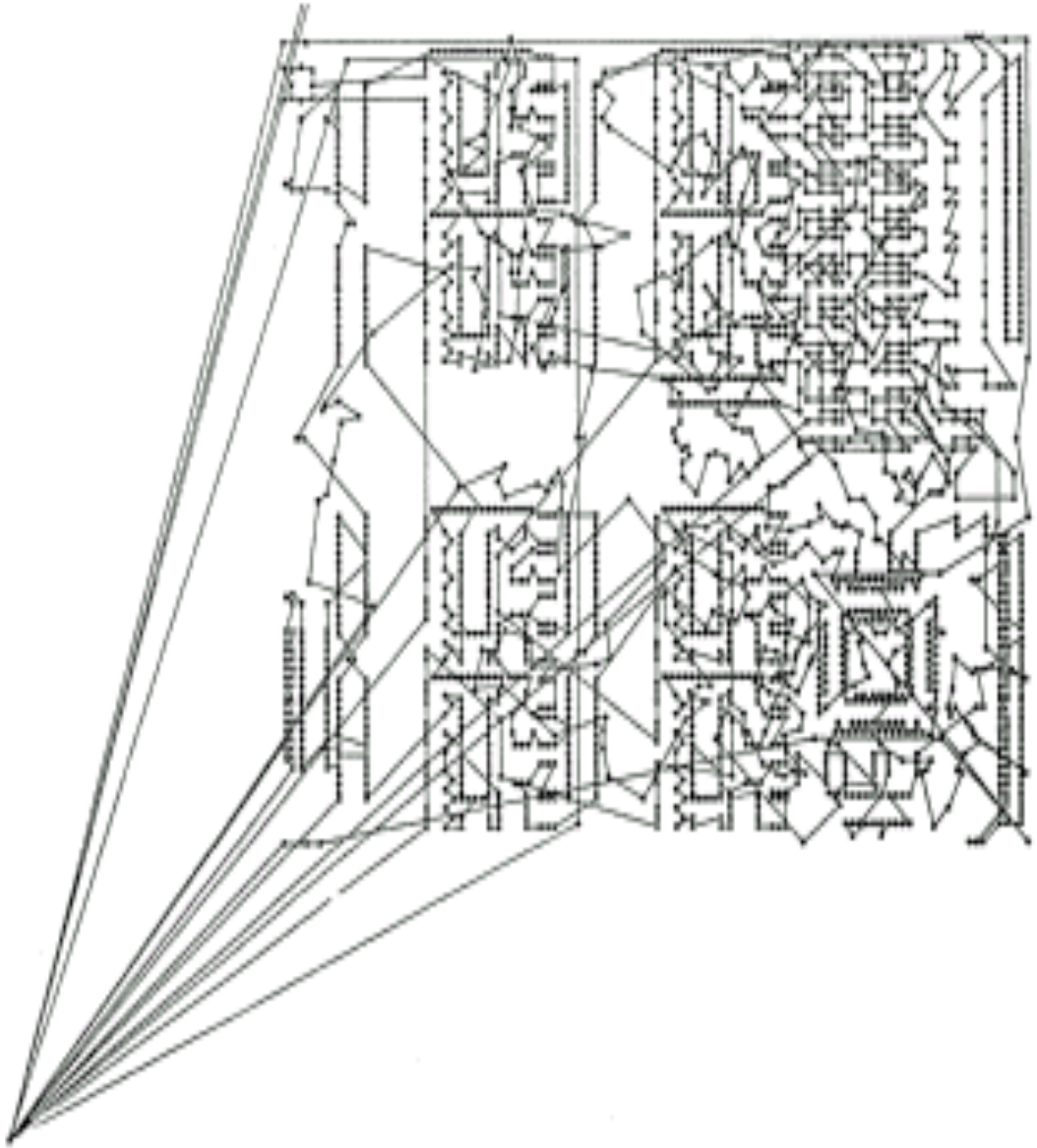
Bohrköpfe



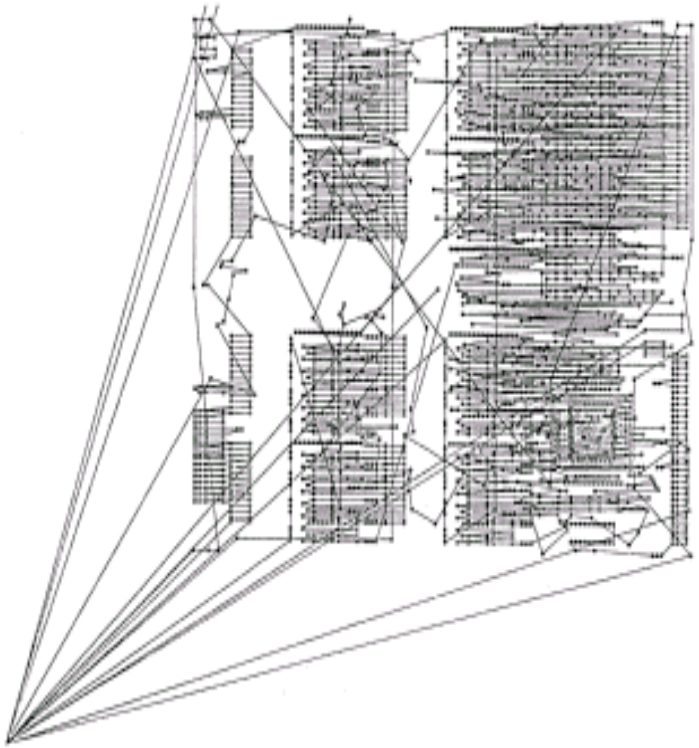
Verfahrwege vorher

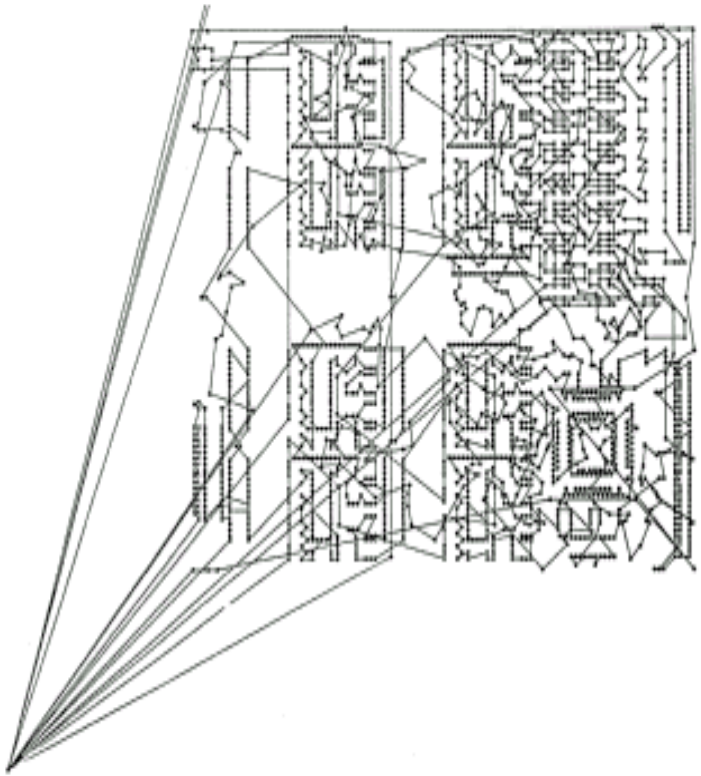


Verfahrwege nachher



Vergleich





Greedy-Heuristiken für TSP

Nearest-Neighbor Heuristik:

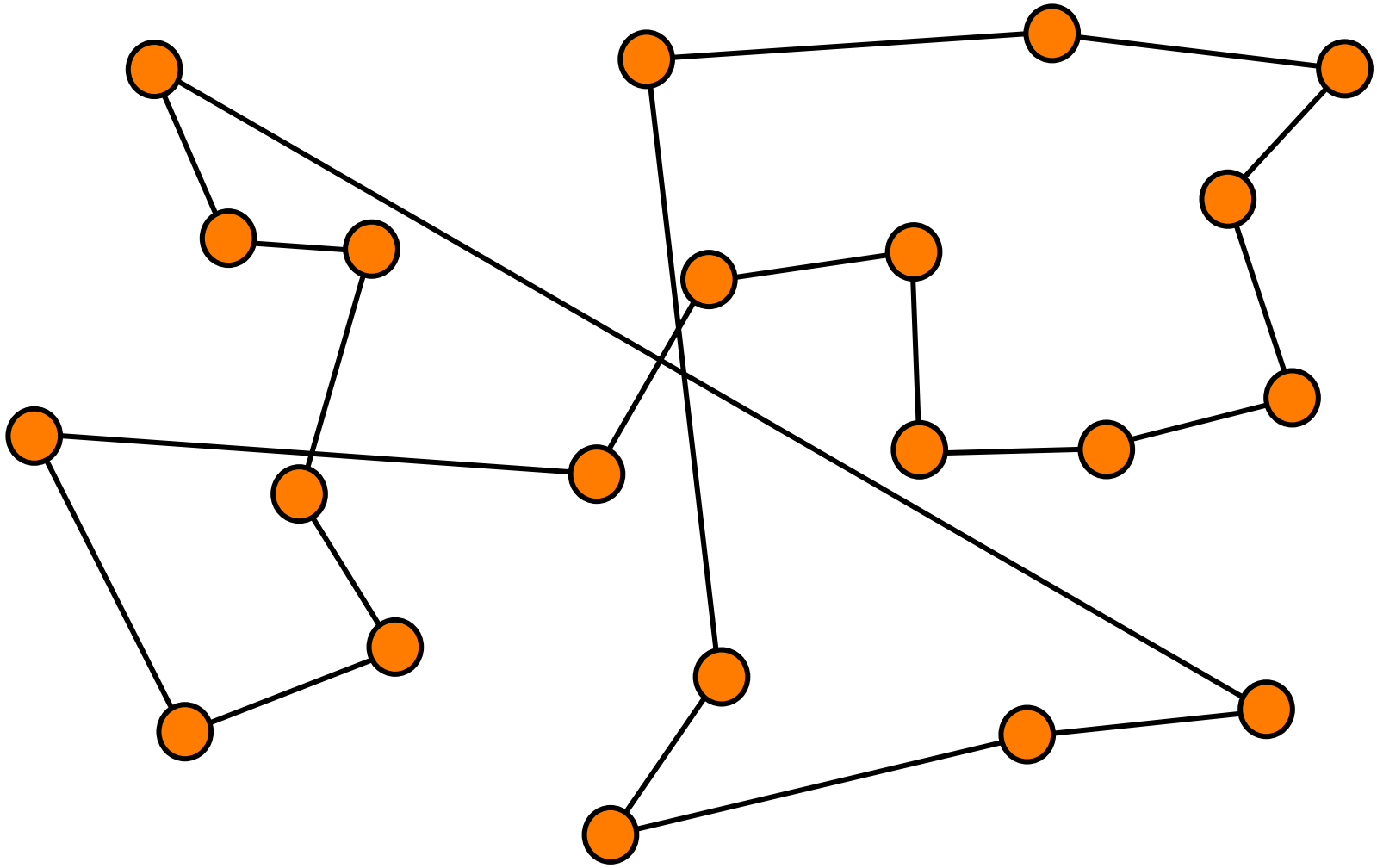
- Beginne mit leerer Tour $T:=\emptyset$
- Beginne an einem Knoten $v=v_0$: Ende der Tour
- Solange noch „freie“ Knoten existieren:
  - Suche den nächsten freien (noch nicht besuchten) Knoten zu v (d.h. die billigste Kante (v,w)) und addiere Kante (v,w) zu T.
- Addiere die Kante (v,v_0)

Greedy-Heuristiken für TSP

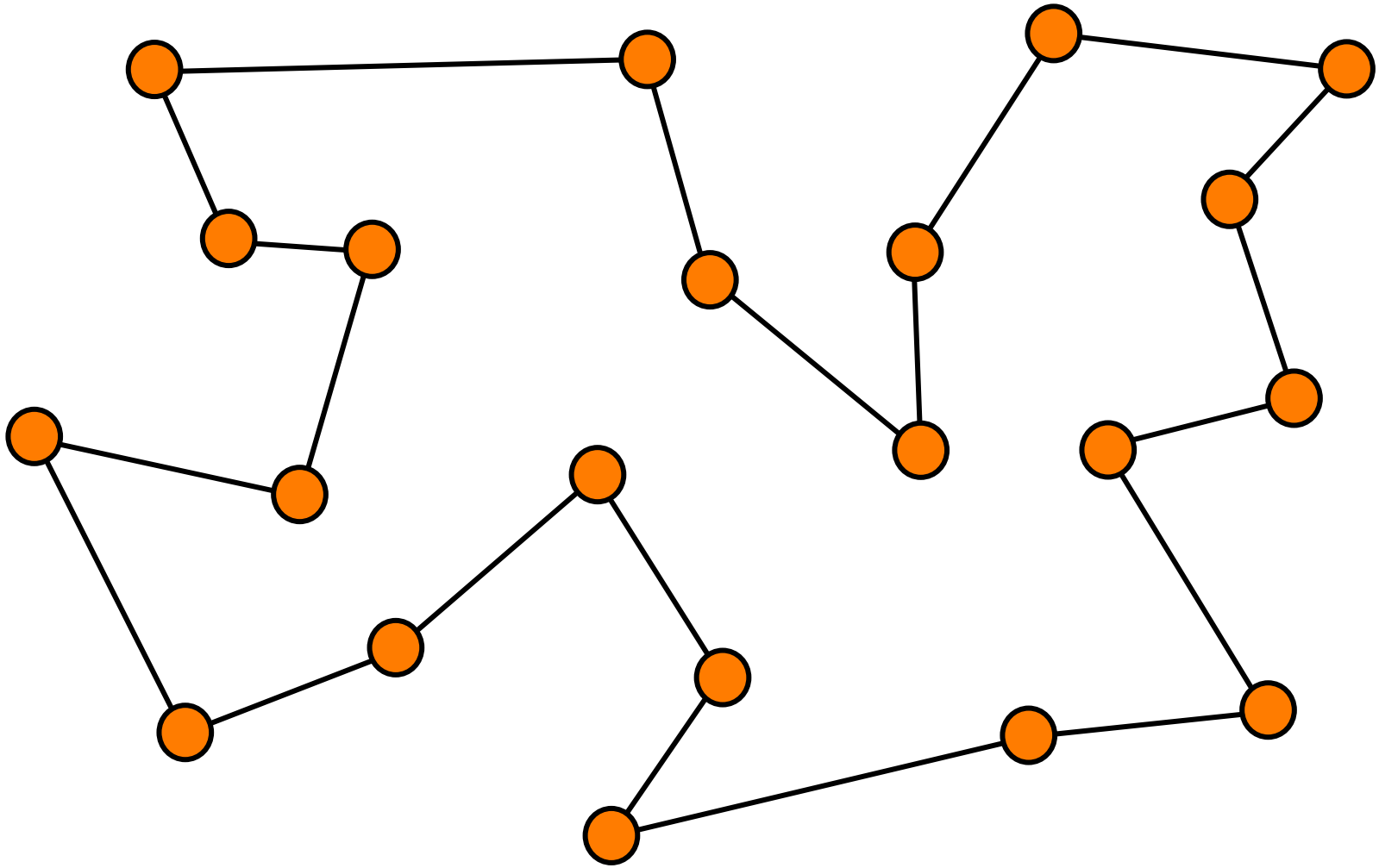
Nearest-Neighbor Heuristik:

- Beginne mit leerer Tour $T:=\emptyset$
- Beginne an einem Knoten $v=v_0$: Ende der Tour
- Solange noch „freie“ Knoten existieren:
  - Suche den nächsten freien (noch nicht besuchten) Knoten zu v (d.h. die billigste Kante (v,w)) und addiere Kante (v,w) zu T.
- Addiere die Kante (v,v_0)

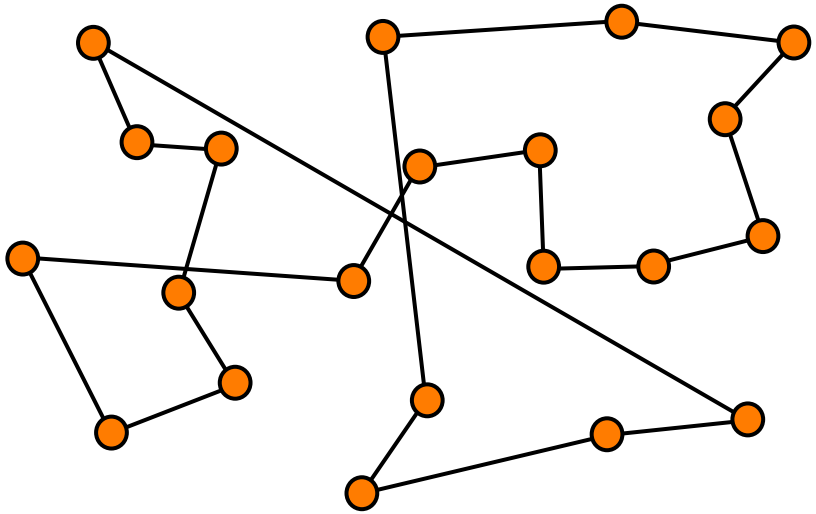
Nearest-Neighbor Heuristik für TSP



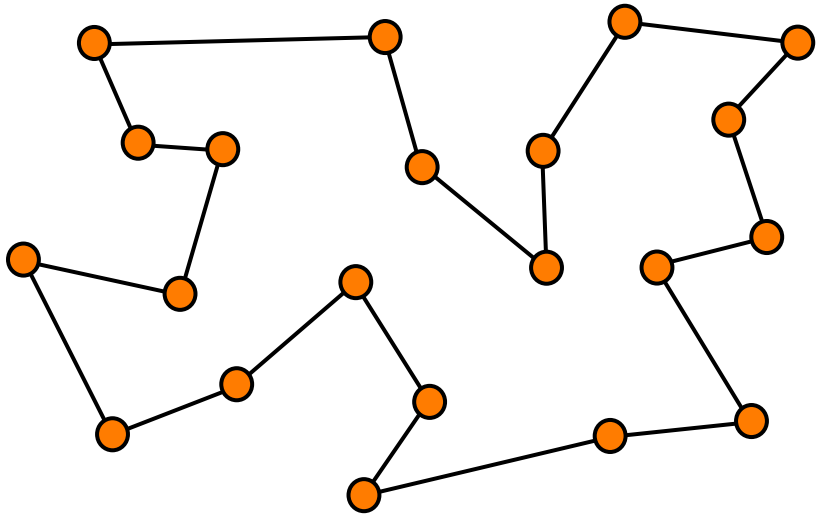
Optimale Lösung der TSP-Instanz



TSP - Vergleich der Lösungen



Greedy



optimal

Wie gut ist die Nearest-Neighbor Heuristik (NN)?

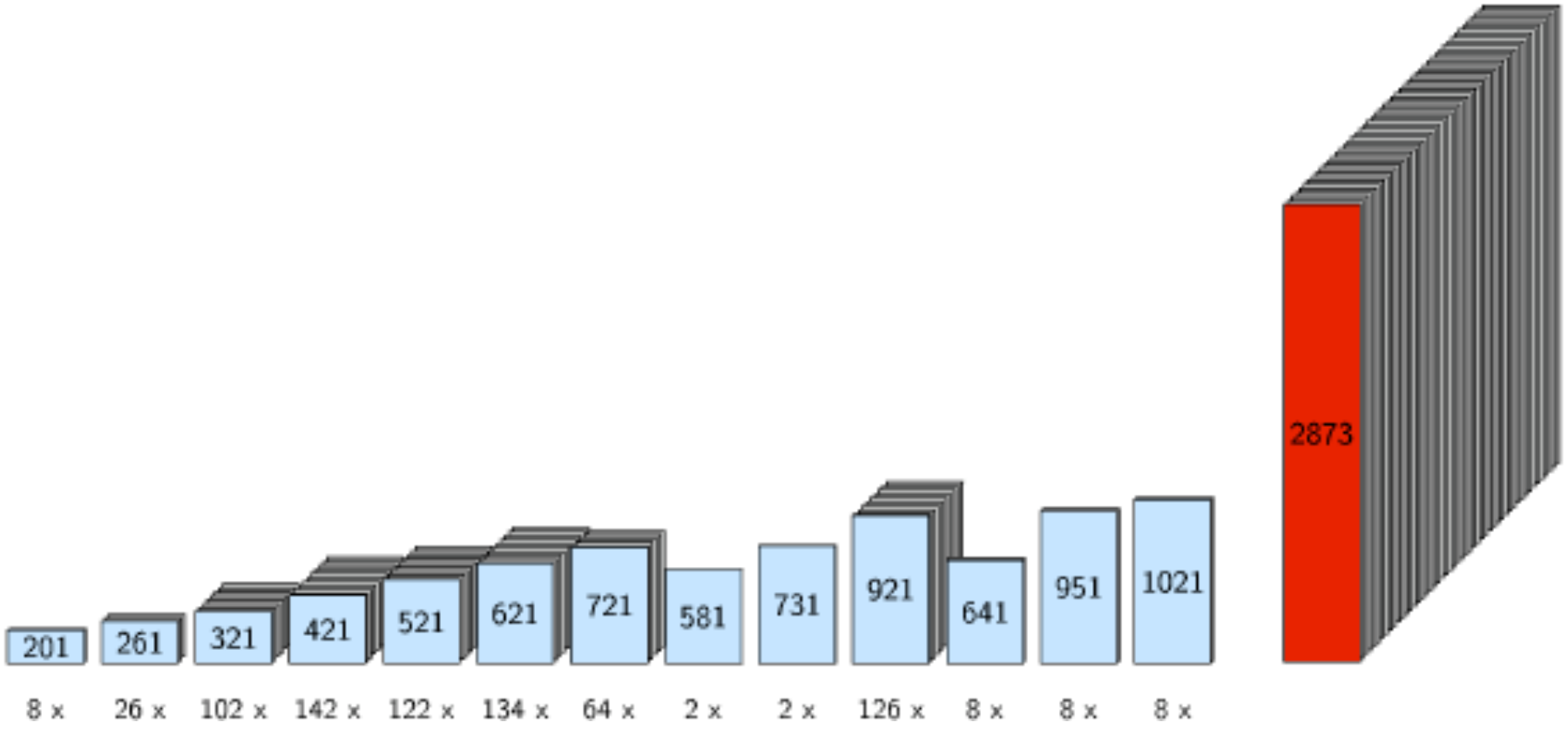
IHRE ERGEBNISSE ☺

Eindimensionale Verschnittoptimierung

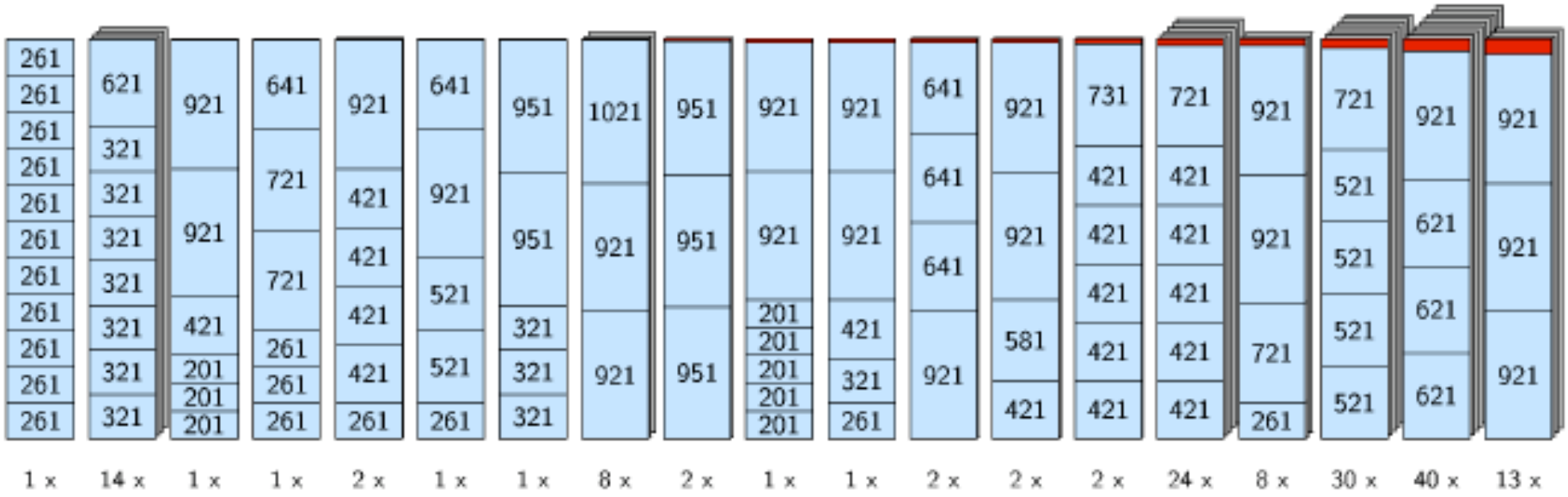
- **Geg.:** Gegenstände 1,…,N der Größe w_i; und beliebig viele Rohlinge der Größe K.

- **Gesucht:** Finde die kleinste Anzahl von Rohlingen, aus denen alle Gegenstände geschnitten werden können.

Bilderrahmenproblem: Beispiel



Minimaler Rohmaterialeinsatz



Bin-Packing / Packen von Kisten

- **Geg.:** Gegenstände $1,\ldots,N$ der Größe w_i und beliebig viele Kisten der Größe K.

- **Gesucht:** Finde die kleinste Anzahl von Kisten, die alle Gegenstände aufnehmen.

First-Fit Heuristik: Jeder Gegenstand wird in die erstmögliche Kiste gelegt, in die er paßt.

First-Fit Heuristik für Bin-Packing

- Gegeben sind beliebig viele Kisten der Größe 101 und 37 Gegenstände der Größen:
- 7x Größe 6
- 7x Größe 10
- 3x Größe 16
- 10x Größe 34
- 10x Größe 51

Insgesamt: 17 Kisten

Kiste 1:	7x Größe 6	5x Größe 10	Summe=92
Kiste 2:	2x Größe 10	3x Größe 16	Summe=68
Kisten 3-7:	2x Größe 34		Summe=68
Kiste 8-17:	1x Größe 51		Summe=51

Wie gut ist die First-Fit Heuristik?

IHRE ERGEBNISSE ☺

Das 0/1-Rucksackproblem

- **Geg.:** N Gegenstände mit Gewicht (Größe) w_i und Wert c_i, und ein Rucksack der Größe K.

- **Gesucht:** Menge der in den Rucksack gepackten Gegenstände mit maximalem Gesamtwert; dabei darf das Gesamtgewicht den Wert K nicht überschreiten.

Greedy-Heuristik an Beispiel für das 0/1-Rucksackproblem

K=17

Gegenstand	a	b	c	d	e	f	g	h
Gewicht	3	4	4	6	6	8	8	9
Wert	3	5	5	10	10	11	11	13

- Sortierung nach Nutzen:= Wert $c_{i,}$ / Gewicht w_i ergibt: d,e,h,f,g,b,c,a
- Wir packen also in den Rucksack:

d (6) | e (6) | b(4) | a(3)

Wert=25

Greedy-Heuristik für das 0/1-Rucksackproblem

- Sortiere die N Gegenstände nach ihrem Nutzen:= Wert c_i, / Gewicht w_i
- Für alle Gegenstände i in der sortierten Reihenfolge:
  - Falls i noch in den Rucksack paßt: füge es hinzu.

Wie gut ist die Greedy-Heuristik für das 0/1-Rucksackproblem?

IHRE VORSCHLÄGE ☺

Hausaufgabe bis Dienstag:

- Finden Sie Beispiele bei denen die heute besprochenen Greedy-Algorithmen möglichst schlecht abschneiden.
- Bringen Sie am Donnerstag je eine Folie mit Ihrem Beispiel (für Tageslichtprojektor) mit.