Parameterisierte Algorithmen

SoSe 2013 in Trier

Henning Fernau
fernau@uni-trier.de

Parameterisierte Algorithmen | Gesamtübersicht

- Einführung
- Grundbegriffe
- Problemkerne
- Suchbäume
- Graphparameter
- Weitere Methoden
- Komplexitätstheorie—parameterisiert

Weitere Methoden

- Dynamisches Programmieren (auf Teilmengen)
- Iterative Kompression
- Farbkodieren

Graphparameter als strukturelle Parameter:

- Knotenanzahl
- Kantenanzahl
- Baumweite, Pfadweite, Cliquenweite etc.

Parameter Knotenzahl $n \leadsto$ exakte Algorithmen

Beispiel: Knotenüberdeckungsproblem

Wieder: Verzweigen bis Maximalgrad drei; dann Polynomzeitlösung.
Dies führt sofort zur Laufzeitabschätzung:

$$T(n) \leq T(n-1) + T(n-4)$$

$\leadsto T(n) \leq 1.39^n$.

Mit deutlichem Mehraufwand kann man (andere) Algorithmen zum Auffinden kleinstmöglicher Knotenüberdeckungen mit Laufzeit $< 1.2^n$ erhalten.
(TR von M. Robson)

Dieser TR eine der Vorläufer von Measure & Conquer

Parameter Knotenzahl n

Meistens trivial: Laufzeit $\mathcal{O}^*(2^n)$.
Frage: Wie geht das beim Auffinden kleinstmöglicher dominierender Mengen (MDS) ?
Hinweis: Führe wieder Färbungen ein.

Bis vor wenigen Jahren galt diese Grenze bei MDS als scharf.

Mittlerweile kann man “fast” auf $\mathcal{O}^*(1.5^n)$ abschätzen.
Erste Ansätze hatten wir in der letzten Vorlesung gesehen.

Parameter Kantenanzahl m

HITTING SET (KANTENPARAMETERISIERT) (HSE)
Eingabe: Ein Hypergraph $G = (V, E)$
Parameter: $|E|$
Frage: Finde eine kleinstmögliche Knotenüberdeckung $C \subseteq V$!

Noch allgemeiner: WHSE für die Variante mit Knotengewichten $\omega : V \to \mathbb{R}_{\geq 1}$.

Grundidee und Aussage

Gegeben: Hypergraph $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$
$V_j := \{v_1, \ldots, v_j\}$ für $j = 0, \ldots, n$; speziell $V_0 = \emptyset$ und $V_n = V$.
$C \subseteq V_j$ heißt Überdeckungsmenge relativ zu V_j und $E' \subseteq E$, falls für jede Hyperkante $e \in E'$ gilt: $e \cap C \neq \emptyset$.
F: zweidimensionales Feld welches für $E' \subseteq E$ und für $j = 1, \ldots, n$ enthält:
$F[E', j] \leftarrow$ kleinste Mächtigkeit einer Überdeckungsmenge relativ zu V_j und E'.
Gibt es keine Überdeckungsmenge relativ zu V_j und E', setze $F[E', j] \leftarrow \infty$.

Einzelheiten folgen, sogar allgemeiner für den knotengewichteten Fall...

Satz 1 *Das gewichtete Knotenüberdeckungsproblem kann für einen Hypergraphen* $G = (V, E)$ *in Zeit* $\mathcal{O}^*(2^{|E|})$ *gelöst werden.*

Algorithm 1 A dynamic programming algorithm for HSE, called HSEdp

Input(s): a hypergraph $G = (V, E)$ with vertex weights ω, $V = \{v_1, \ldots, v_n\}$

Output(s): (implicitly) a hitting set $C \subset V$ of minimum weight

for all $E' \subseteq E$ **do**
  $F[E', 0] \leftarrow \infty$
$F[\emptyset, 0] \leftarrow 0$
for $j = 1, \ldots, n$ **do**
  for all $E' \subseteq E$ **do**
    Let $E'' \leftarrow \{e \in E' \mid v_j \in e\}$.
    $F[E', j] \leftarrow \min\{F[E', j-1], F[E' \setminus E'', j-1] + \omega(v_j)\}$
{$F[E, n]$ contains the minimum weight of a hitting set.}

Zur Korrektheit des Algorithmus

Der formale Beweis erfolgt durch Induktion über j für die Aussage:
$F[E', j]$ enthält das kleinste Gewicht einer Überdeckung relativ zu V_j und E'
(oder $F[E', j] = \infty$, falls es keine Überdeckung relativ zu V_j und E' gibt).

Die Aussage gilt für $j = 0$: Die leere Menge überdeckt die leere Menge.

Es gibt im Induktionsschritt zwei Fälle zu unterscheiden:
1. Eine kleinstgewichtige Überdeckung von E' relativ zu V_j enthält v_j.
Dann sollte $F[E', j]$ sich aus der Summe von $\omega(v_j)$ und dem Gewicht der kleinsten Kantenmenge ergeben, welche alle Kanten bis auf die v_j beinhaltenden abdeckt.
Nach Induktionsannahme steht diese Zahl in $F[\{e \in E' \mid v_j \notin e\}, j-1]$.
2. Keine kleinstgewichtige Überdeckung von E' relativ zu V_j enthält v_j.
Dann steht der richtige Wert bereits in $F[E', j-1]$ nach Induktion.
Welche Rolle spielt beim Nachweis der Korrektheit des Verfahrens ∞ ?

Facility Location (Filialeröffnungsproblem)

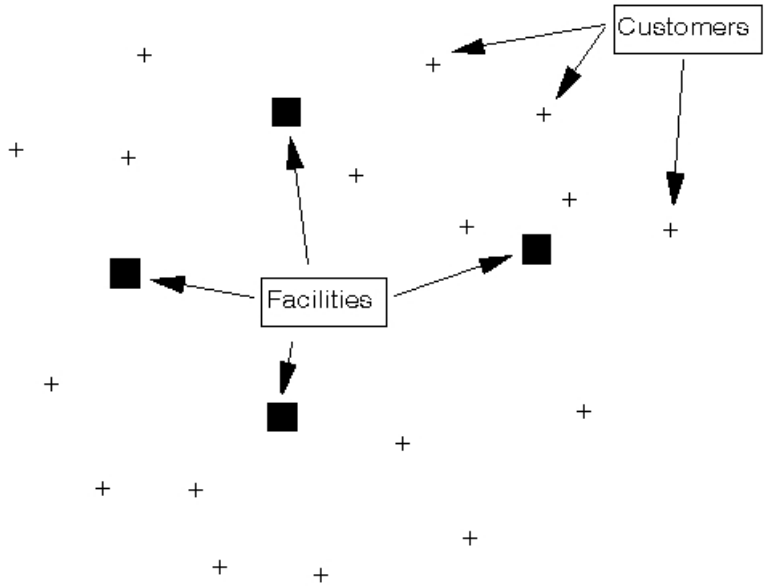
Allgemeines Problem:

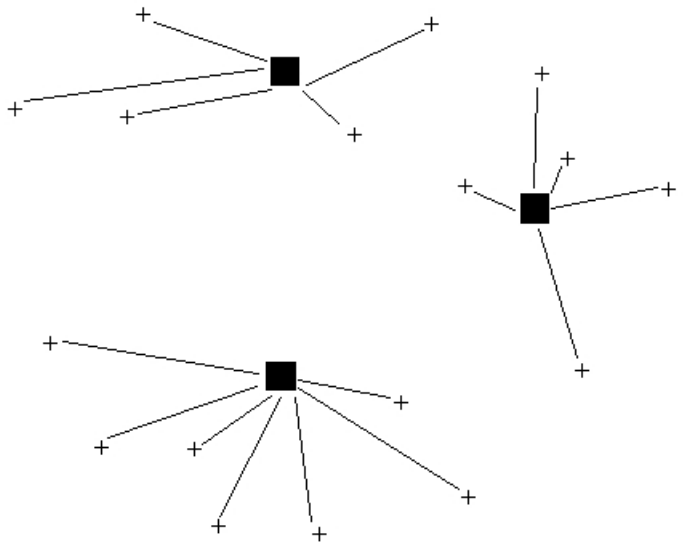
Gegeben ist eine Menge von möglichen Filialstandorten und eine Menge von Kunden(orten).

An welchen Orten sollen Filialen eröffnen, um möglichst kostengünstig die Kunden bedienen zu können ?

Teilproblem: Welche Kunden sollten von welchen Filialen aus bedient werden ?

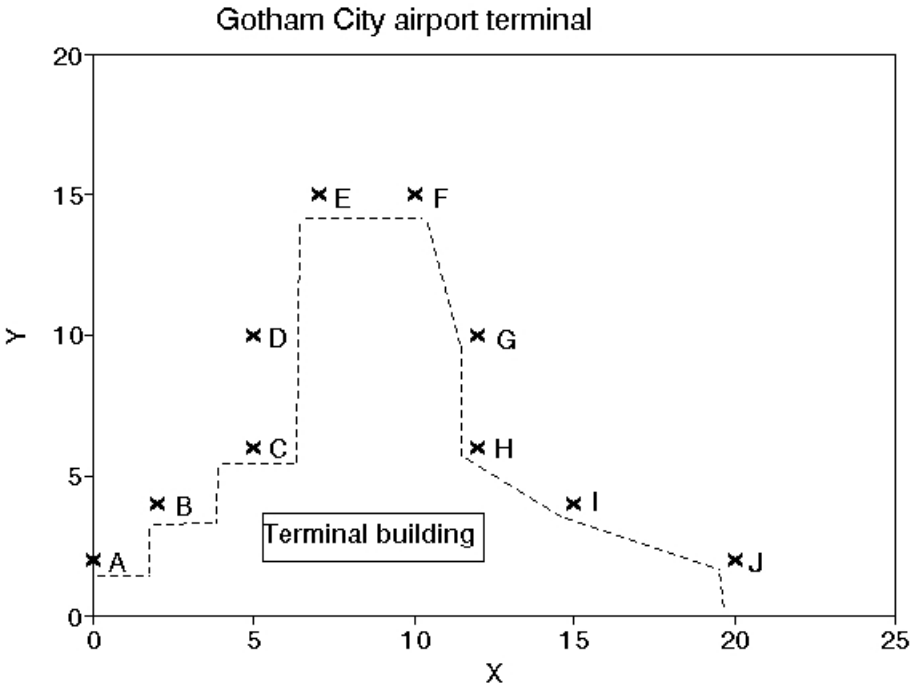
Facility Location (Filialeröffnungsproblem am Beispiel)





Ein konkreteres Beispiel: Wo soll die Gepäckausgabe beim (fiktiven) Gotham City Airport Terminal platziert werden, um die Gepäckbewegungen zu minimieren ? Dort gibt es 10 Ankunfts-Gates (A bis J).

Gate	x coord.	y coord.	#Gepäck
A	0	2	3600
B	2	4	2500
C	5	6	1800
D	5	10	2200
E	7	15	1000
F	10	15	4500
G	12	10	5600
H	12	6	1400
I	15	4	1800
J	20	2	3000



Ein konkreteres Beispiel:

Bestmögliche Lösungen sind abhängig von der Metrik

$\rightsquigarrow$ Obacht bei der Modellbildung

Euklidischer Abstand

11-09-2000 14:34:36	Facility Name	X Axis	Y Axis	Flow To All Facilities	Cost To All Facilities
1	A	0	2	3600	44,266.76
2	B	2	4	2500	23,819.75
3	C	5	6	1800	10,667.54
4	D	5	10	2200	11,483.13
5	E	7	15	1000	6,776.21
6	F	10	15	4500	27,073.98
7	G	12	10	5600	11,965.45
8	H	12	6	1400	4,938.54
9	I	15	4	1800	12,556.59
10	J	20	2	3000	36,299.11
11	NEW	10.12	8.98	0	0
	Total			27400	189,847.06
	Distance	Measure:	Euclidean		

NEW gibt die Gepäckausgabe an.

Manhattan-Abstand

11-09-2000 14:33:37	Facility Name	X Axis	Y Axis	Flow To All Facilities	Cost To All Facilities
1	A	0	2	3600	50400
2	B	2	4	2500	25000
3	C	5	6	1800	9000
4	D	5	10	2200	19800
5	E	7	15	1000	12000
6	F	10	15	4500	40500
7	G	12	10	5600	33600
8	H	12	6	1400	2800
9	I	15	4	1800	12600
10	J	20	2	3000	42000
11	NEW	10	6	0	0
	Total			27400	247700
	Distance	Measure:	Rectilinear		

NEW gibt die Gepäckausgabe an.

Facility Location: ein Zoo von Problemen

Wir suchen uns daher eine konkrete Aufgabenstellung heraus:

FILIALERÖFFNUNGSPROBLEM (FL)

Eingabe: Ein bipartiter Graph $B = (F \uplus C, E)$, bestehend aus einer Menge F of potentieller Filialstandorte und einer Menge C von Kunden(orten), sowie eine Kantenrelation E,
wobei $\{f, c\} \in E$ bedeutet, dass c von der Filiale am Standort f bedient werden könnte;
schließlich Gewichtsfunktionen $\omega_F : F \to \mathbb{R}_{\geq 1}$ und $\omega_E : E \to \mathbb{R}_{\geq 1}$
Parameter: $k \in \mathbb{N}$
Frage: Gibt es eine Filialstandortsmenge $F' \subseteq F$ sowie Möglichkeiten $E' \subseteq E$, um alle Kunden zu bedienen so dass
(1) $\forall f \in F(f \in F' \iff \exists e \in E'(f \in e))$,
(2) $\forall c \in C \exists e \in E'(c \in e)$ und
(3) $\sum_{f \in F'} \omega_F(f) + \sum_{e \in E'} \omega_E(e) \leq k$?

ω_F modelliert die Kosten einer Filialeröffnung und ω_E die Kosten des laufenden Betriebs.
Für Mitgliedschaft in *FPT* ist $\omega_E(e) \geq 1$ wesentlich, jedoch $\omega_F(f) \geq 0$ wäre möglich.

Alternative Sicht: Minimierungsproblem mit strukturellem Parameter $|C|$.

Beobachtungen

(1) Der zugrunde liegende bipartite Graph könnte o.E. vollständig sein; fehlende Kanten müssen nur hinreichend großes Gewicht bekommen ($> k$).
(2) Gibt es mehr als k Kunden, so handelt es sich um eine NEIN-Instanz.
Denn: jede Kundenbedienung verursacht Kosten von wenigstens Eins nach Voraussetzung.
(3) Für jede der höchstens $2^{|C|} \leq 2^k$ Mengen von Kunden können wir in einer Tabelle os (one-serve) abspeichern, wieviel es kosten würde, wenn jene Kunden von nur einer Filiale aus bedient würden.
(4) Wir können dann beobachten, dass eine Kundenmenge entweder günstigstenfalls von einer einzigen Filiale bedient wird (das steht in os) oder aber in zwei je optimal bediente Kundenteilmengen zerfällt.

Satz 2 *Das Filialeröffnungsproblem kann für einen bipartiten Graphen* $B = (F \uplus C, E)$ *mit Gewichtsfunktionen* $\omega_F : F \to \mathbb{R}_{\geq 0}$ *und* $\omega_E : E \to \mathbb{R}_{\geq 0}$ *in Zeit* $\mathcal{O}^*(3^{|C|})$ *gelöst werden.*

Algorithm 2 A dynamic programming algorithm for facility location, called FLdp

Input(s): a bipartite graph $B = (F \uplus C, E)$ with edge weights ω_E and facility weights ω_F

Output(s): an implicit facility location strategy incurring minimum weight

if $|C| > k$ **then**
  return NO;
$s(\emptyset) \leftarrow 0$;
for all $X \subseteq C$ **do**
  compute $os(X)$;{ in time $\mathcal{O}(|X| \cdot |F|)$}
for $i \leftarrow 1, \ldots, |C|$ **do**
  for all $X \subseteq C$, $|X| = i$ **do**
    $s(X) \leftarrow \min_{\emptyset \subsetneq Y \subseteq X}(os(Y) + s(X \setminus Y))$
    {Every customer belongs either to Y, $X \setminus Y$ or to $C \setminus X$;}
    {Convention: $\min_\emptyset \cdots = \infty$.}

Anwendung des MDL Prinzips auf die DTD Inferenz

Garofalakis et al. 2003 XTRACT; Witten et al. 1994.

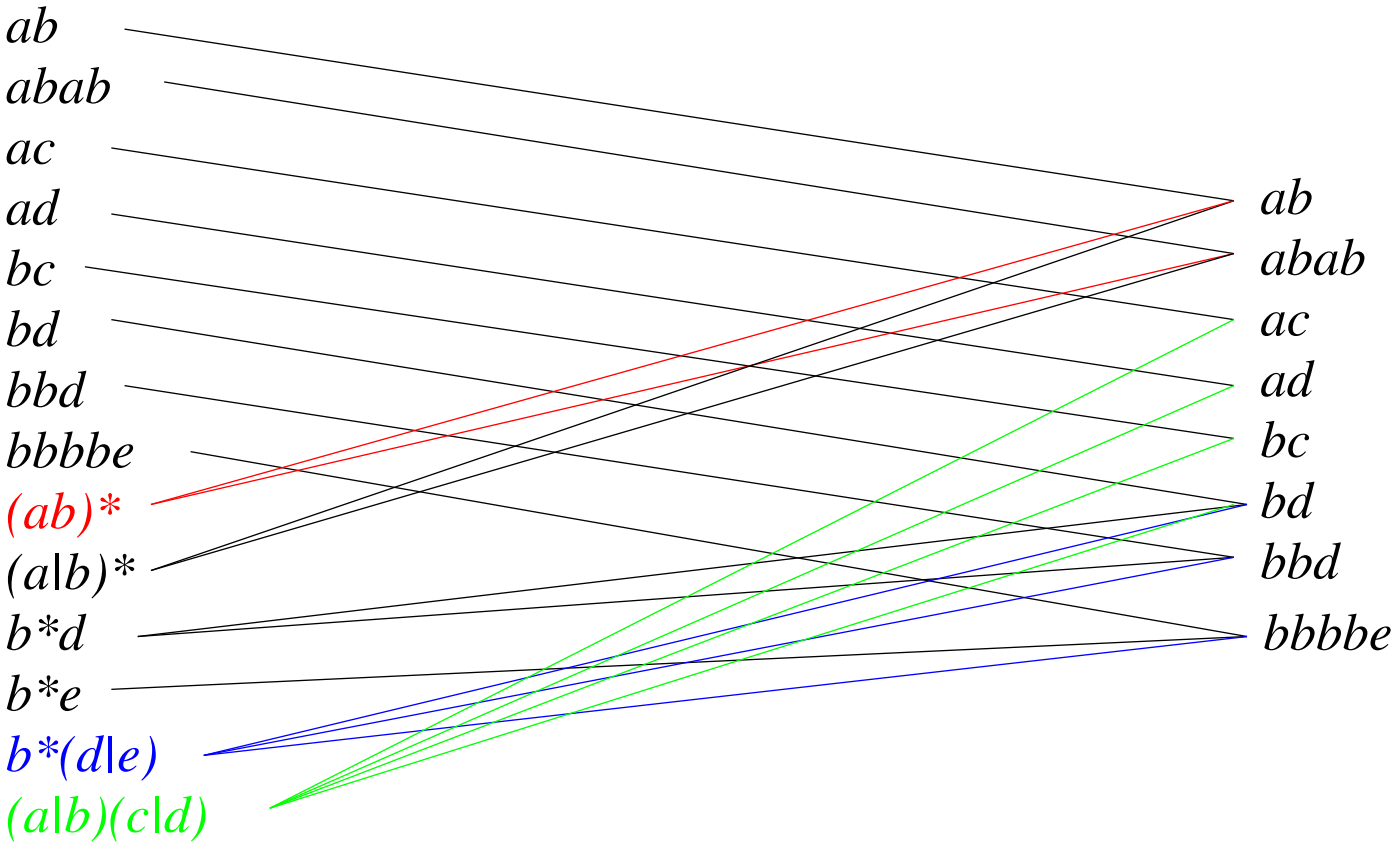
Die Beschreibung besteht aus zwei Teilen:

1. der Länge der Theorie (in Bits) und

2. der Länge der Daten (in Bits), kodiert mit Hilfe der gewählten Theorie.

Bsp.: $\{ab, abb, abbb, abbbb, abbbbb\}$ bei Theorie ab^+ codierbar durch $\{1, 10, 11, 100, 101\}$.

Ein abstraktes Beispiel



MDL als kombinatorisches Problem MDL-OPTIMAL-CODING
Als Parameter wählen wir eine Obergrenze auf die Länge der Codierung.

Satz: MDL-OPTIMAL-CODING ist NP-vollständig. (F. ICGI 2004)

Satz: MDL-OPTIMAL-CODING ist in *FPT*.

Ähnliche Probleme ergeben sich in vielen Bereichen der Datenkompression.

Das Steinerbaumproblem—eine wortreiche Erklärung aus Wikipedia

Das Steinerbaumproblem (STEINER TREE), ein nach dem Schweizer Mathematiker Jakob Steiner benanntes Problem der Graphentheorie, ist eine Verallgemeinerung des Problems des minimalen Spannbaums.
Beim Steinerbaumproblem sucht man in einem umgebenden Graphen einen kleinsten Teilgraphen (den Steinerbaum), welcher eine Menge vorgegebener Endpunkte (die Terminale) miteinander verbindet.

Das Steinerbaumproblem gehört zur Liste der 21 klassischen NP-vollständigen Probleme, von denen Richard Karp 1972 die Zugehörigkeit zu dieser Klasse zeigen konnte.

Steinerbäume—formaler

STEINERBAUM (ST)
Eingabe: Ein ungerichteter zusammenhängender Graph $G = (V, E)$ und eine Terminalmenge $T \subseteq V$ sowie eine Kantengewichtsfunktion $\omega : E \to \mathbb{R}_{\geq 1}$
Parameter: $|T|$
Frage: Finde Steinerbaum $B \subseteq E$ mit kleinstmöglichem Gewicht !

Dabei ist ein Steinerbaum ein Baum, der alle Terminale (aus T) verbindet. Dabei können, falls nötig, auch Knoten aus $V \setminus T$ verwendet werden; diese Knoten werden dann Steinerknoten genannt.

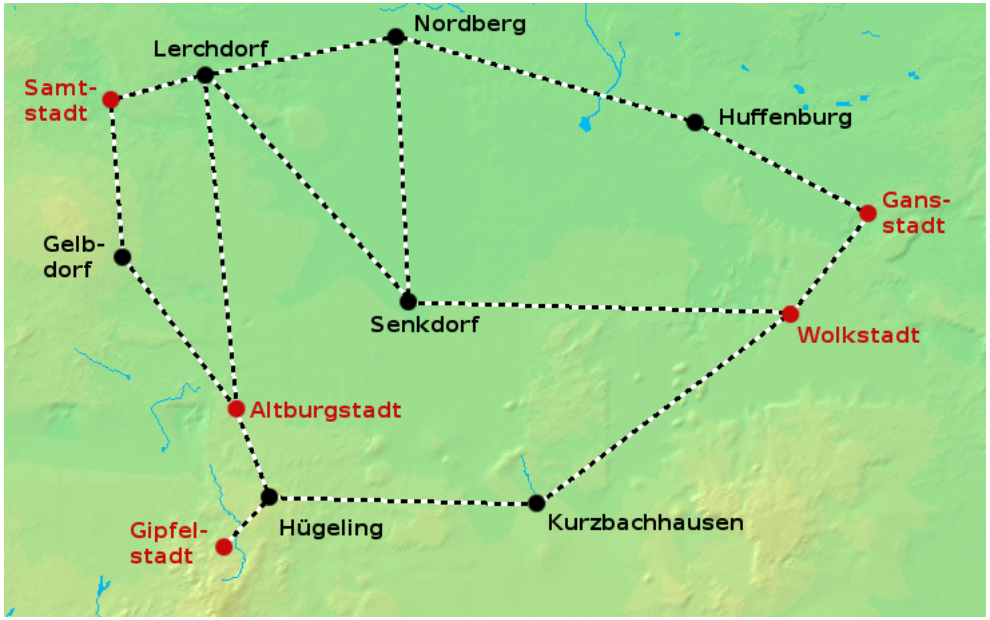
Ein Steinerbaum mit $T = V$ ist also ein minimaler Spannbaum von G.
Ein Steinerbaum mit $|T| = 2$ ist ein billigster / kürzester Weg.
Diese beiden Extremfälle können in Polynomzeit gelöst werden.

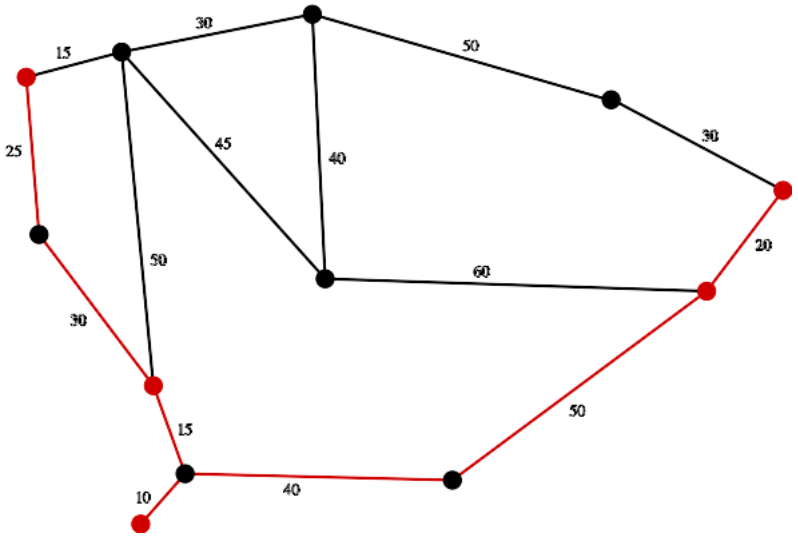
Ein Anwendungsbeispiel

Ein Staat möchte sein marodes Schienennetz modernisieren, damit es mit Elektrolokomotiven befahren werden kann. Dafür sollen die vorhandenen Gleise mit Oberleitungen versehen werden; neue Gleise sollen nicht verlegt werden. Allerdings reicht das Budget nicht für eine Elektrifizierung des gesamten Netzes, weshalb sich die Regierung entschließt, nur so viele Strecken zu elektrifizieren, dass es möglich ist, von jeder Großstadt mit einer E-Lok in jede andere Großstadt zu fahren (dabei wird nicht ausgeschlossen, dass die E-Lok auch durch Kleinstädte fährt). Gleichzeitig sollen die Gesamtkosten für die Modernisierung möglichst gering gehalten werden.

Das Streckennetz bildet einen Graphen:
Knoten sind die Städte, und Kanten vorhandene Bahnstrecken.
Kantengewicht: Kosten für Elektrifizierung des Streckenabschnitts.
Zu verbindene Großstädte: Terminale.

Das Beispiel in Zahlen





Dreyfus Wagner Algorithmus: Grundzüge

— Dynamische Programmierung
— Berechnet minimalen Steinerbaum für T aus minimalen Steinerbäumen für alle Teilmengen von T
— Also: Berechne minimale Steinerbäume für alle 2-elementigen Teilmengen (kürzeste Wege)
— Berechne daraus minimale Bäume für alle 3-elementigen Teilmengen
— usw.

Das führt auf:

Satz 3 *Das Steinerbaumproblem kann für einen zusammenhängenden ungerichteten Graphen* $G = (V, E)$ *mit Terminalmenge* T *und Kantengewichtsfunktion* ω *in Zeit* $\mathcal{O}^*(3^{|T|})$ *gelöst werden.*

Algorithm 3 A dynamic programming algorithm for Steiner trees, called DW

Input(s): a graph $G = (V, E)$ with edge weights ω, $T \subseteq V$, $V = \{v_1, \ldots, v_n\}$

Output(s): a Steiner tree $B \subset E$ of minimum weight

for all $v, w \in V$ **do**

compute $p(v, w)$;{shortest / cheapest paths}

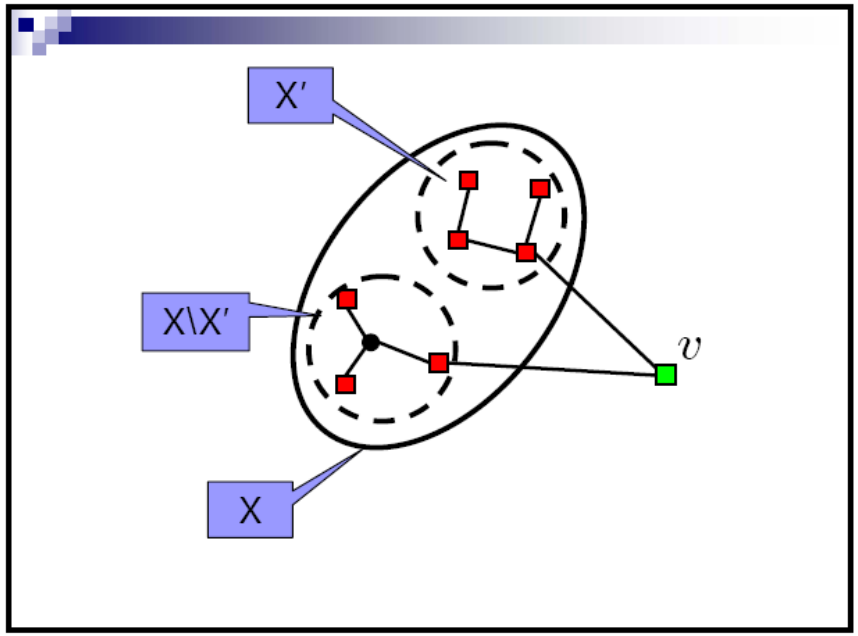
for all $\{x, y\} \subseteq V, \{x, y\} \cap T \neq \emptyset$ **do**

$s(\{x, y\}) \leftarrow p(x, y)$;

for $i \leftarrow 2, \ldots, |T| - 1$ **do**
  for all $X \subseteq T$, $|X| = i$ **do**
    for all $v \in V \setminus X$ **do**
      $\mathrm{Aux}_{\mathrm{no-leaf}}(v) \leftarrow \min_{\emptyset \subsetneq X' \subsetneq X}(s(X' \cup \{v\}) + s((X \setminus X') \cup \{v\})$
    for all $v \in V \setminus X$ **do**
      $s(X \cup \{v\}) \leftarrow \min(\min_{w \in X} p(v, w) + s(X), \min_{w \in V \setminus X}(p(v, w) + \mathrm{Aux}_{\mathrm{no-leaf}}(w)))$

output $s(T)$.

Eine Beweisskizze für die Korrektheit des Dreyfus-Wagner-Algorithmus



Notationen: Für festes $X \subseteq T$ und $v \in V \setminus X$ sei:
$s(X \cup \{v\})$: das Kantengewicht eines kleinstgewichtigen Steinerbaums für $X \cup \{v\}$ und
$\text{Aux}_{\text{no-leaf}}(v)$: das Kantengewicht eines kleinstgewichtigen Steinerbaums für $X \cup \{v\}$ mit der Einschränkung: v ist kein Blatt im Baum.
Die Abbildung illustriert die erste Rekursion (für $\text{Aux}_{\text{no-leaf}}(v)$):
v ist kein Blatt und unterteilt daher den zu konstruierenden kleinstgewichtigen Steinerbaum in zwei kleinere Steinerbäume; v ist dort möglicherweise Blatt.
Zur Bestimmung von $s(X \cup \{v\})$ unterscheide:

1. v ist kein Blatt in $B \rightsquigarrow$ wie $\text{Aux}_{\text{no-leaf}}$ (mit $v = w$ im zweiten Teil enthalten).
2. v ist Blatt, und der in v startende Weg endet in $w \in X$
$\rightsquigarrow$ erster Teil der Rekursion: B vereinigt Steinerbaum für X und billigsten Weg von v nach w.
3. v ist Blatt, aber der in v startende Weg endet in $w \notin X$
$\rightsquigarrow$ zweiter Rekursionsteil: B vereinigt Steinerbaum für $X \cup \{w\}$ und billigsten Weg von v nach w.

Die Laufzeit $\mathcal{O}^*(3^{|T|})$ sieht man wie bei dem (verwandten ?) Filialproblem.

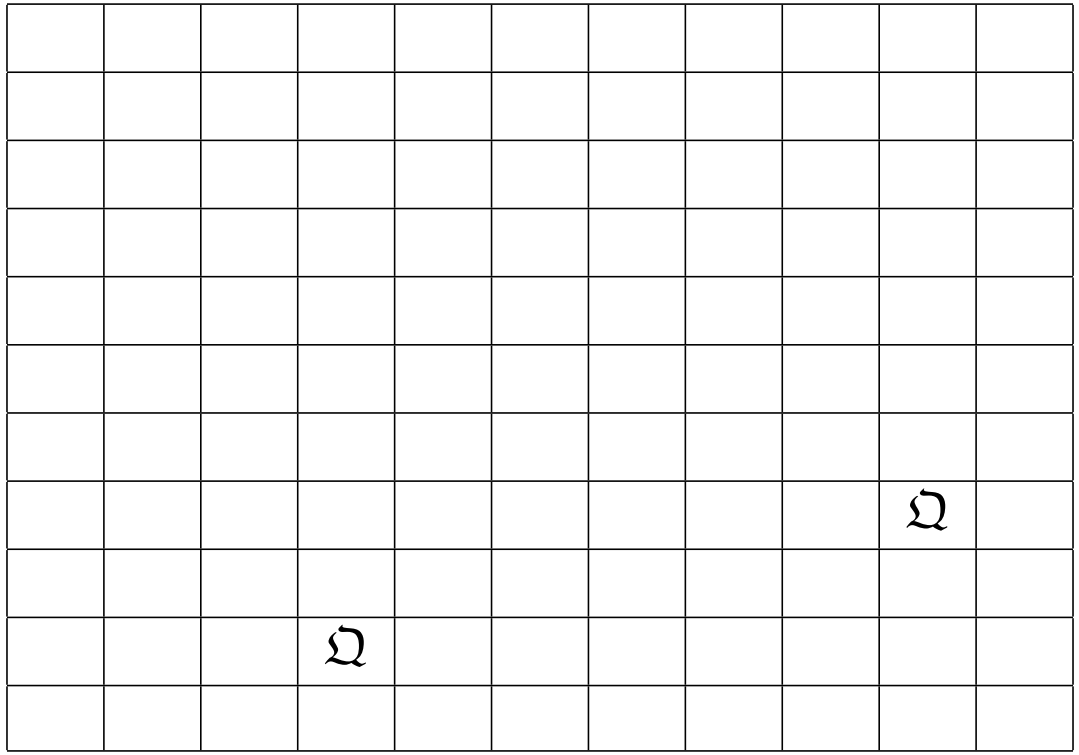
Anwendungsbeispiele

Total Vertex Cover: Gibt es Knotenüberdeckung K, sodass jeder $x \in K$ einen Nachbarn $y \in K$ besitzt?
Nach der Aufzählungsphase der minimalen VC C der Maximalgröße k berechne man für diejenigen Knoten $C' \subseteq C$, die keinen Nachbarn in C haben, ein kleinstmögliches Hitting Set mit der Kantenmenge $\{N_G(v) \mid v \in C'\}$.

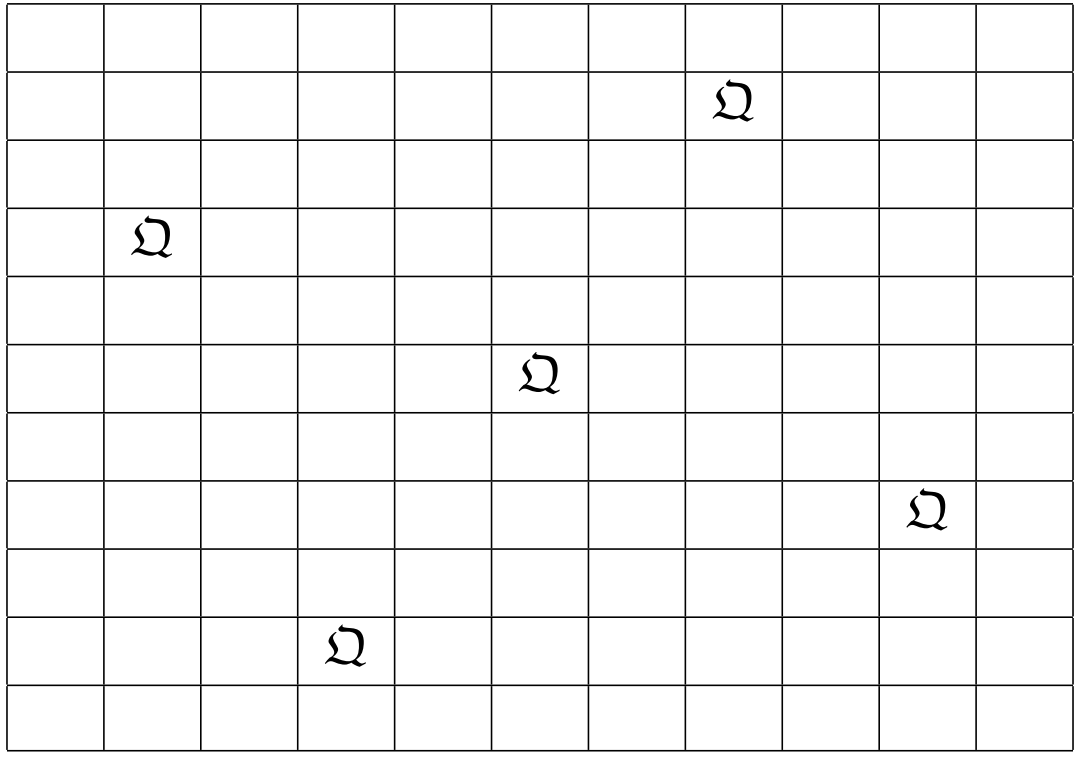
Connected Vertex Cover: Nach der Aufzählungsphase der minimalen VC C der Maximalgröße k berechne man (mit Einheitskantengewichten) einen minimalen Steinerbaum mit Terminalmenge C.

Ergänzendes Material

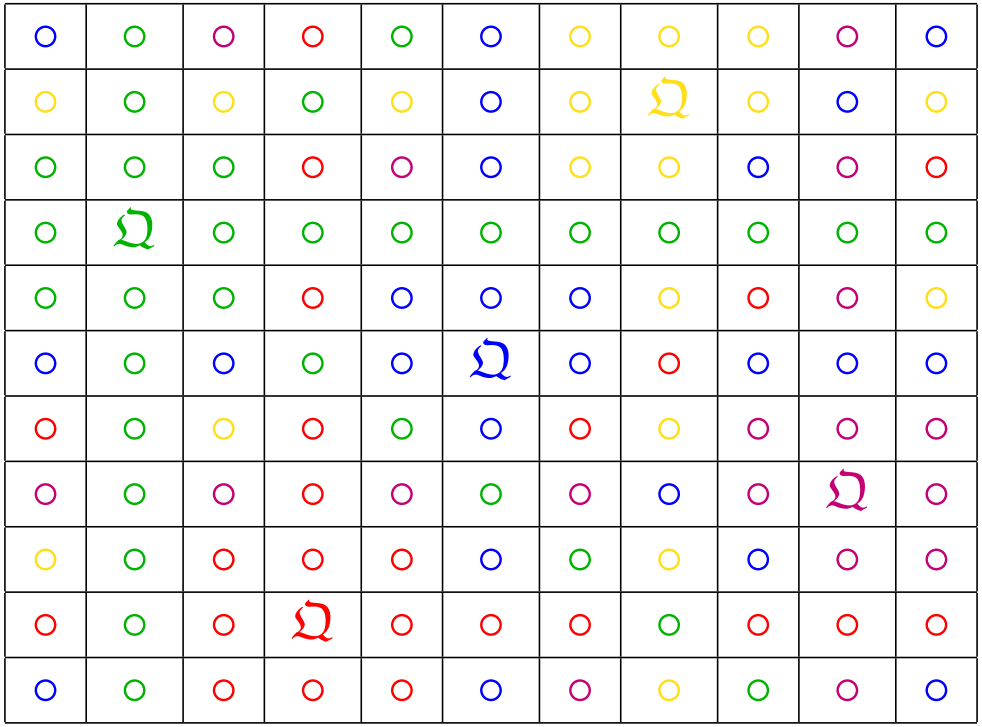
Damendominierung mit $n = 11$: Wie kann man die Vorgabe erweitern ?



Eine zulässige Lösung ?



Klar durch farbliche Dominierungsbereiche



Algorithm 4 A dynamic programming algorithm for MINQDS

Input(s): positive integer n

Output(s): min. # queens necessary to dominate all squares on an $n \times n$ board

Set Q to $\{(\emptyset, 0)\}$.

{ $dp(X)$: Dominierungsmuster (Menge von Schachfeldern), durch Linienmenge X induziert.
$(dp(X), i)$: Bislang kleinstmögliche Dominierungsmenge für $dp(X)$ durch i Damen. }

for all squares s of the chessboard **do**
  Let T be the set of (at most four) lines that contain s.
  for all line sets S **do**
    Look for some $(dp(S), i)$ in Q
    if $dp(T) \not\subseteq dp(S)$ **then**
      {putting a queen on s would dominate new squares}
      Update $(dp(S \cup T), ?)$

Let M be the set of all possible lines. Find $(dp(M), j)$ in Q. Return j.

Zu Komplexität des Verfahrens

Es gibt $6n$ Linien (Zeilen, Spalten und $4n$ Diagonalen).
Die Linienmengen bestimmen im Wesentlichen die Komplexität.

Satz 4 *MINQDS kann in Zeit $\mathcal{O}^*(64^n)$ mit Alg. 4 berechnet werden.*

Die Abschätzung lässt sich mit der aus der Kombinatorik bekannten Tatsache, dass nur wenig mehr als $n/2$ Damen zur Dominierung eines $n \times n$-Felds benötigt werden, unter Verwendung unserer Erbschaft von Stirling verschärfen zu knapp $\mathcal{O}^*(40^n)$.

Man beachte, dass diese Abschätzungen eine Wurzel im Exponenten zu stehen haben, so man als Schachbrettgröße n^2 ansetzt.

Zusammenfassung: Pro und Contra

Dynamisches Programmieren (auf Teilmengen) liefert oft “einfach” *FPT*-Algorithmen.

Die dahinterliegende Denkweise ist zunächst gewöhnungsbedürftig.

Die Algorithmen sind an und für sich von einfacher Gestalt, aber schwieriger zu verstehen und oft deutlich schlechter als Suchbaumalgorithmen.

Ein weiterer Nachteil: Diese Algorithmen benötigen inhärent exponentiellen Platz.

Literatur / Nachweise

Der gewichtete Hitting Set Algorithmus findet sich in: H. Fernau. Edge dominating set: efficient enumeration-based exact algorithms. In H. L. Bodlaender and M. Langston (editors) International Workshop on Parameterized and Exact Computation IWPEC 2006, vol. 4169 of LNCS, pp. 142–153. Berlin: © Springer, 2006.

Gute Notizen über Facility Location finden sich bei J.E. Beasley: OR Notes Daher stammen auch einige Beispiele.

Bei Steinerbäumen ist die deutsche Wikipedia mal ausnahmsweise (in 2009) aussagekräftiger als das englische Gegenstück. Dort habe ich auch u.a. das Eisenbahnbeispiel entnommen.

Wer noch mehr darüber wissen will: Hans Jürgen Prömel, Angelika Steger. The Steiner Tree Problem. A Tour through Graphs, Algorithms, and Complexity, Vieweg 2002.

Daran orientiert sich auch die Darstellung einer hübschen Vorlesung von der TU Wien, der ich auch Anregungen entnahm.

Dreyfus / Wagner (und auch etliche Baumzerlegungsbasierte Algorithmen, insbesondere für dominierende Mengen, siehe nächste VL) können mit einer Art “Fouriertransformation” für Mengen

beschleunigt werden:
Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto: Fourier meets Möbius: fast subset convolution. In Proceedings of the 39th Annual ACM Symposium on Theory of Computing STOC 2007, San Diego, CA, June 11–13, 2007, Association for Computing Machinery, New York, 2007, S. 67—74.

Connected Vertex Cover wird zuerst in der angegebenen Originalarbeit behandelt und später in einer Reihe von Arbeiten verbessert. Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke: Parameterized complexity of vertex cover variants. Theory of Computing Systems, 41(3):501—520, 2007.

Über das Schachproblem findet sich etwas in
H. Fernau: Minimum Queens Dominating Set: a trivial programming exercise? *Discrete Applied Mathematics*, Band 158 (2010), 308–318.