Parameterisierte Algorithmen

WS 2011/12 in Trier

Henning Fernau
fernau@uni-trier.de

Parameterisierte Algorithmen | Gesamtübersicht

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

Organisatorisches

Vorlesung: Dienstag 12-14 Uhr, H7
NEU ab 2. VL-Woche: Dienstag 16:00 - 17:30 h H 406
NEU ab 4. VL-Woche: Dienstag 14:15 - 15:45 h H 406

Übungen (Daniel Meister): Dienstag 10-12 Uhr, HZ 204
NEU ab 1. VL-Woche: Donnerstag 14:00 - 15:30 h H 405

Meine Sprechstunde: DO, 13-14 Uhr

Kontakt: fernau,daniel.meister@uni-trier.de

Hausaufgaben / Schein ?! n.V. (Master ?! $\rightarrow$ mündliche Prüfung)

Grundbegriffe 1: Graphentheorie

Graphen

Graph: $G = (V, E)$
V: (endliche) Knotenmenge (Punkte, vertices)
E: Kantenmenge (edges)
E ist binäre Relation auf V, d.h., $E \subseteq V \times V$.
Ist E symmetrisch, so ist G ein ungerichteter Graph.
Dann E auffassbar als Teilmenge von 2^V (siehe Hypergraphen).
Übliche Kantennotationen: (x, y), xy, $x\vec{y}$, $\{x, y\}$.
Nicht erfasst in dieser Formalisierung: Mehrfachkanten.
Schlingen: eine Kante xx.
(Ein Graph ohne Schlingen und Mehrfachkanten heißt auch schlicht.)
Ein Graph mit Mehrfachkanten heißt auch Multigraph.

Pfade in Graphen

$P(G)$: Menge der Pfade von G; ein Pfad ist eine Kantenfolge $p = e_1 e_2 \dots e_k$ mit $e_i \cap e_{i+1} \neq \emptyset$ für $i = 1, 2, \dots, k-1$;
k heißt auch die Länge des Pfades.
Ist $e_1 = u_1 u_2$ und $e_k = v_1 v_2$, so ist p ein Pfad zwischen u_1 und v_2.
$p = e_1 e_2 \dots e_k$ heißt einfach (simple) (oder knotendisjunkt) falls $e_i \cap e_{i+1} \cap e_j \neq \emptyset$ gdw. $j = i$ oder $j = i + 1$ für $i = 1, 2, \dots, k-1$ und $j = 1, 2, \dots, k$.

Ein einfacher Pfad $p = e_1 e_2 \dots e_k$ ist ein Kreis gdw. $e_1 \cap e_k \neq \emptyset$; k heißt auch Länge des Kreises.

C_k: Kreis der Länge k.
P_k: einfacher Pfad der Länge k, der kein Kreis ist.

Zusammenhang I ($\rightsquigarrow$ analysis situs)

Zwei Knoten x, y sind (einfach) zusammenhängend gdw. es gibt einen Pfad dazwischen.
Eine Knotenmenge K heißt zusammenhängend gdw. jedes Knotenpaar in K zusammenhängend ist.
Eine (inklusions-)maximale zusammenhängende Knotenmenge heißt Zusammenhangskomponente.
Ist V eine Zusammenhangskomponente von $G = (V, E)$, so heißt G zusammenhängend.
starker / schwacher Zusammenhang bei gerichteten Graphen.

Zusammenhang II

Zwei Knoten x, y sind zweifach zusammenhängend gdw. sie liegen auf einem Kreis.
Eine Knotenmenge K heißt zweifach zusammenhängend gdw. jedes Knotenpaar in K zweifach zusammenhängend ist.
Eine (inklusions-)maximale zusammenhängende Knotenmenge heißt zweifache Zusammenhangskomponente.
Ist V eine zweifache Zusammenhangskomponente von $G = (V, E)$, so heißt G zweifach zusammenhängend.

Teilgraphen

Ist G ein (Hyper-)graph, so heißt (V', E') ein Teilgraph (Untergraph, subgraph) von G gdw. $V' \subseteq V(G)$ und $E' \subseteq E(G)$.

Ein Graph heißt kreisfrei oder azyklisch wenn er keinen Kreis als Teilgraph enthält.

Ein Teilgraph (V', E') von $G = (V, E)$ ist knoteninduziert (durch V') gdw.

$$\forall e \in E : e \subseteq V' \iff e \in E'.$$

Schreibweise $G[V']$ für G'.

Ein Teilgraph (V', E') von $G = (V, E)$ ist kanteninduziert (durch E') gdw. $V' = \bigcup_{e \in E'} e$.
Schreibweise: $G[E']$ für G'.

Nachbarschaft und Grad

$N(v)$: offene Nachbarschaft,
d.h., Menge aller Nachbarn von v,
d.h., $N(v) = \{u \mid \exists e \in E(G) : \{u, v\} \subseteq e\}$.
$N[v] := N(v) \cup \{v\}$ ist die abgeschlossene Nachbarschaft von v.

$\deg(v)$ bezeichnet den Grad (degree) (oder die Valenz) des Knotens v, d.h., $\deg(v) = |N(v)|$.
Für $X \subseteq V$ benutzen wir auch $N(X) = \bigcup_{x \in X} N(x)$ und $N[X] = N(X) \cup X$.

Ein (Hyper-)Graph $G = (V, E)$ mit $\forall v \in V : \deg(v) = k$ heißt k-regulär.

Spezielle Mengen

Es sei $G = (V, E)$ ein (Hyper-)Graph.

$C \subseteq V$ heißt (Knoten-)Überdeckung (vertex cover) gdw. $\forall e \in E \exists v \in C : v \in e$.
$C \subseteq E$ heißt Kantenüberdeckung (edge cover) gdw. $\forall v \in V \exists e \in C : v \in e$.
$I \subseteq V$ heißt unabhängig (independent, stable) gdw. $I \cap N(I) = \emptyset$.
$D \subseteq V$ ist eine dominierende (Knoten-)Menge gdw. $N[D] = V$.

Graphen und binäre Relationen 1: Adjazenzmatrix

Es sei $G = (V, E)$ ein (gerichteter) Graph.
Die Adjanzenzmatrix von G, geschrieben $A(G)$, ist eine binäre zweidimensionale Matrix, deren Zeilen und Spalten durch V indiziert sind.
$A(G)[x, y] = 1$ gdw. $(x, y) \in E$.
$A(G)$ ist also die Relationenmatrix zur Relation E.
$A(G)$ ist symmetrisch gdw. G ist ungerichtet.
$A(G)$ enthält Einsen auf der Hauptdiagonalen gdw. G enthält Schlingen.

Hinweis: Pfade der Länge k durch $A(G)^k$.

Graphen und binäre Relationen 2: Inzidenzmatrix

Es sei $G = (V, E)$ ein Hypergraph.
Die Inzidenzmatrix von G, geschrieben $I(G)$, ist eine binäre zweidimensionale Matrix, deren Zeilen durch V indiziert sind und deren Spalten durch E indiziert sind. $I(G)[x, e] = 1$ gdw. $x \in e$.
Jede binäre Matrix M ist Inzidenzmatrix irgendeines Hypergraphs $G(M)$.
Ungerichtete schlichte Graphen sind gekennzeichnet durch Inzidenzmatrizen mit der Eigenschaft, dass genau zwei Einträge in jeder Spalte Einsen sind.
Beachte: Symmetrie von Knoten und Kanten bei Hypergraphen.

Hinweis: Kantenlistenimplementierung (für “dünne” Graphen).

Färbungen

Ist $G = (V, E)$ ein (Hyper-)graph, dann definiert eine Abbildung $c : V \to C$ eine Färbung (coloring) gdw. $\forall a \in C : c^{-1}(a)$ ist unabhängig.

Ein Graph ist k-färbbar gdw. es gibt eine Färbung $c : V \to C$ mit $|C| = k$.

2-färbbare Graphen heißen auch bipartit oder paar.
Ein Graph ist bipartit gdw. er keinen Kreis ungerader Länge als Teilgraph enthält.
Ist $G = (V, E)$ ein ungerichteter bipartiter Graph mit Bipartisierung $V = V_1 \cup V_2$, so hat $A(G)$ eine besondere Form: Die Teilmatrizen, die durch $V_1 \times V_1$ bzw. durch $V_2 \times V_2$ indiziert sind, enthalten nur Nullen. Die $V_1 \times V_2$- bzw. $V_2 \times V_1$-Teilmatrizen sind Transponierte voneinander. Daher ist die die gesamte Information in der durch $V_1 \times V_2$ indizierten Teilmatrix $A_B(G)$ (bipartite Adjanzenzmatrix) enthalten.
So kann jede binäre Matrix (und damit jeder Hypergraph) als bipartiter Graph gedeutet werden.

Spezielle Graphen: Bäume & Co.

Erinnerung: C_k, P_k

Zusammenhängende Graphen, die keinen Kreis als Teilgraphen enthalten, heißen Bäume.

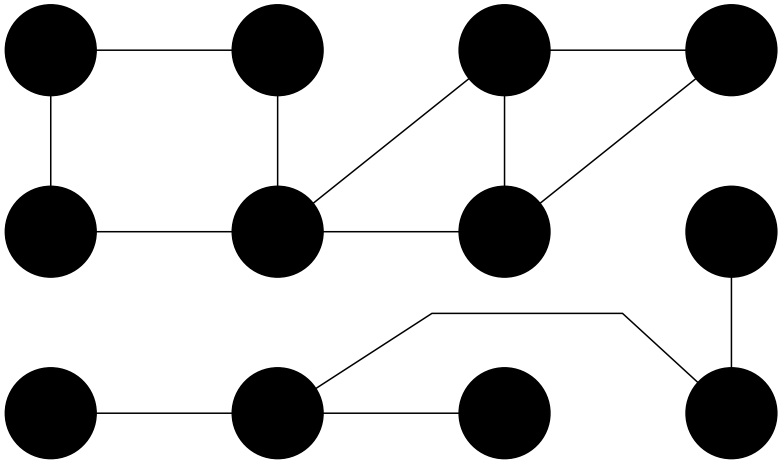
Graphen mit Bäumen als Zusammenhangskomponenten heißen Wälder.

Daher ist ein Graph kreisfrei oder azyklisch gdw. er ist ein Wald.

Knoten vom Grad Eins (insbesondere in Wäldern) heißen Blätter, die übrigen Knoten innere Knoten.

Knoten vom Grad Null heißen auch isolierte Knoten.

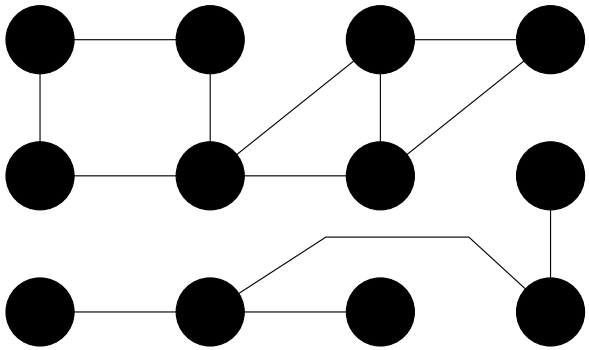
Ein Beispiel



Zugehörige Adjazenzmatrix

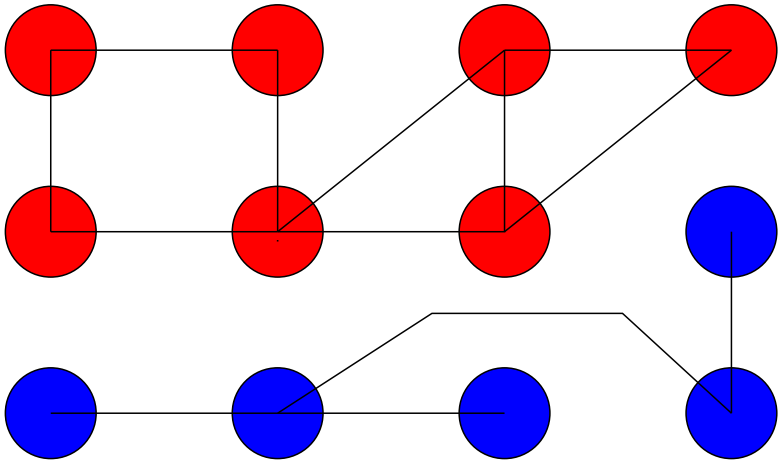
0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	0	1	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	1	0	0

Zusammen. . .

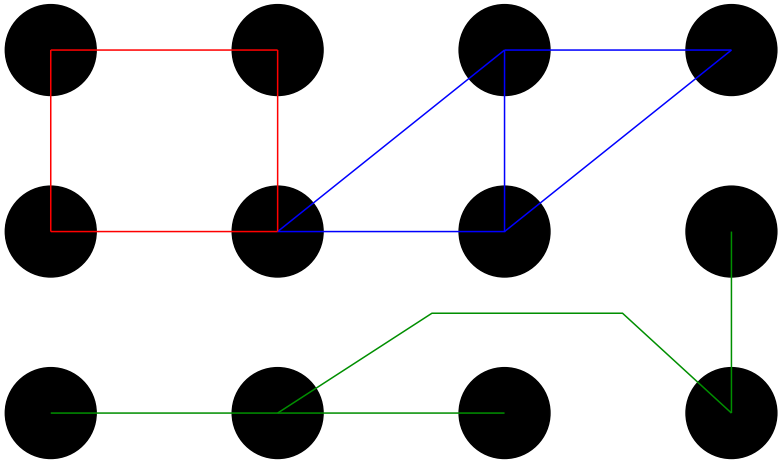


0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	0	1	0	1	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	1	0	0

Zwei Komponenten



Knoteninduziert oder nicht?!



Graphoperationen

- Ist $G = (V, E)$ ein Graph, so ist sein Graphkomplement der Graph $G^c = (V, E^c)$ mit $xy \in E^c$ gdw. $xy \notin E$. (Bem.: Darstellung in $A(G)$)

- Sind $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ Graphen, so ist ihre Graphvereinigung der Graph $(V_1 \cup V_2, E_1 \cup E_2)$.

- Ist $G = (V, E)$ ein Graph mit $x, y \in V$, so bezeichnet

$$G[x = y] = ((V \cup \{[x, y]\}) \setminus \{x, y\}, E')$$

die Verschmelzung oder Identifikation von x und y (Kontraktion falls $e = \{x, y\}$ in E), wobei

$$E' = \{\{u, v\} \mid u, v \in V \setminus \{x, y\}\} \cup \{\{u, [x, y]\} \mid \{\{u, x\}, \{u, y\}\} \cap E \neq \emptyset\}.$$

- $G - v$ (Knotenlöschen) bezeichnet den Graphen, der aus $G = (V, E)$ entsteht, indem v aus V sowie alle v enthaltenden Kanten aus E gelöscht werden. Ist $V_\ell \subseteq V$ mit $|V_\ell| = \ell$, d.h. $V_\ell = \{v_1, \ldots, v_\ell\}$, so definiere $G - V_\ell$ induktiv: $G - V_1 = G - v_1$, $G - V_i = (G - V_{i-1}) - v_i$ für $i > 1$.

- $G - e$ entsprechend (Kantenlöschen)

Weitere Zusammenhänge

Satz 1 *Verschmilzt man alle Knoten in jeder Einfach-Zusammenhangskomponente, so entsteht eine Menge isolierter Knoten. Die Anzahl der Knoten entspricht der Anzahl der Zusammenhangskomponenten des ursprünglichen Graphen.*

Satz 2 *Verschmilzt man alle Knoten in jeder Zweifach-Zusammenhangskomponente, so entsteht ein Wald. Ist der ursprüngliche Graph einfach zusammenhängend, so entsteht bei Verschmelzung der Zweifach-Zusammenhangskomponenten ein Baum.*

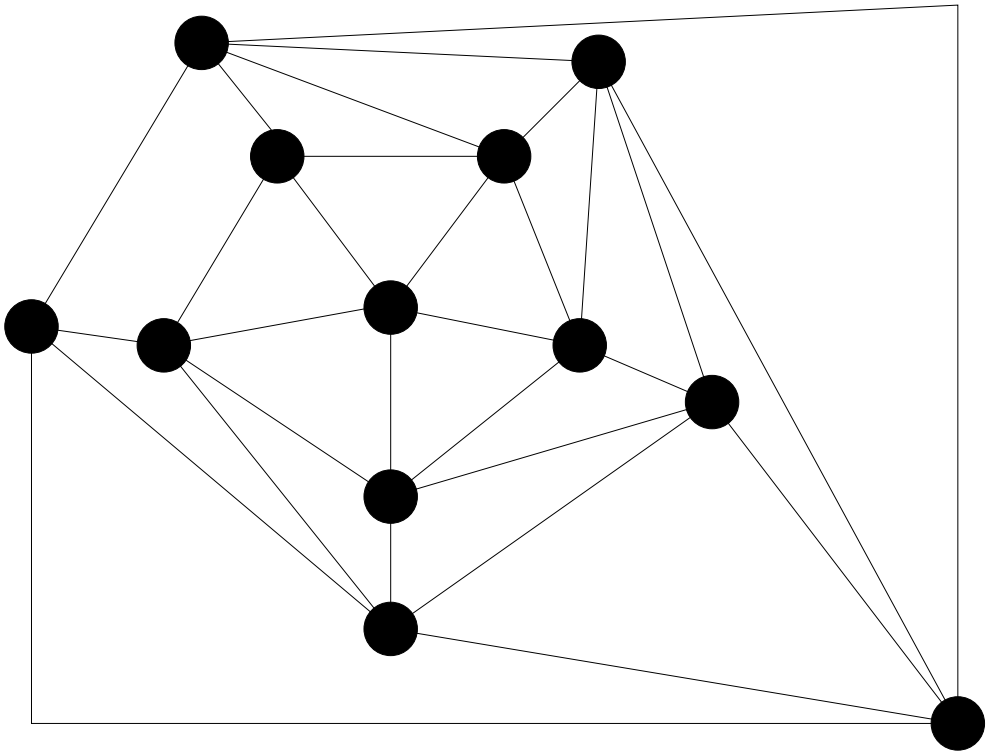
Die Kanten im ursprünglichen Graphen, die denjenigen im entstehenden Wald entsprechen, heißen Brücken.

Darstellung von Graphen

Vielleicht "natürlichste" Variante: Zeichnen in der Ebene.
Ein Graph $G = (V, E)$ heißt planar, falls er ohne Kreuzungen zwischen Kanten in der Ebene gezeichnet werden kann.
Genauer heißt dies: es gibt eine Einbettung (embedding) von G in die *Euklidische Ebene* $\mathbb{R}^2$, das ist eine Abbildung, welche jedem Knoten v (injektiv) einen Punkt $\phi(v) \in \mathbb{R}^2$ zuordnet und jeder Kante $\{u, v\}$ eine Jordan-Kurve $\phi((u, v))$, welche $\phi(u)$ und $\phi(v)$ verbindet, derart dass zwei solche Jordan-Kurven gemeinsame Punkte nur in den Endpunkten haben.

Ist die Einbettung explizit mitgegeben, spricht man auch von eingebetteten Graphen. Eingebettete Graphen unterteilen die Ebene in Teilflächen, so genannte Facetten (faces).

Ein Beispiel: ein 5-regulärer planarer Graph



Exkurs planare Graphen

Eulersche Flächenformel / Polyederformel

für zusammenhängende planare Graphen (schlingenfrei, ohne Mehrfachkanten) mit wenigstens zwei Knoten:

$$f - 2 = m - n$$

n: Knotenzahl

m: Kantenzahl

f: Facettenzahl

Folgerung 3 *In einem planaren Graphen gibt es stets einen Knoten vom Grad höchstens fünf.*

Exkurs planare Graphen

Widerspruchs-Beweis des Korollars:
Angenommen, das Korollar wäre falsch.
Dann gäbe es ein Gegenbeispiel, d.h., einen planaren Graphen $G = (V, E)$ mit Mindestgrad sechs.
Betrachte Gegenbeispiel mit möglichst wenigen Knoten $\rightsquigarrow G$ ist einfach zusammenhängend.
Sodann betrachte Gegenbeispiel mit möglichst vielen Kanten $\rightsquigarrow G$ ist zweifach zusammenhängend.
Doppeltes Abzählen $\rightsquigarrow 2m = \sum_{v \in V} \deg(v) \geq 6n \rightsquigarrow n \leq m/3$.
$\rightsquigarrow m - n \geq 2/3 \cdot m$
Zweifachzusammenhang $\rightsquigarrow$ Jede Kante ist in zwei verschiedenen Facetten.
Jede Facette hat mindestens drei anliegende Kanten.
Doppeltes Abzählen $\rightsquigarrow 2m \geq 3f \rightsquigarrow f - 2 \leq 2/3 \cdot m - 2$

Graphentheoretische Probleme

Finde kleinstmögliche Knotenüberdeckung VC (Hitting Set HS) ! (*NP*-vollständig)

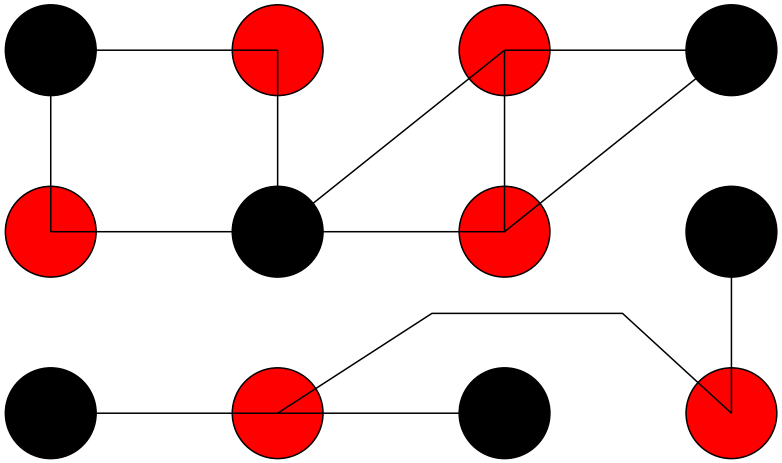
Finde kleinstmögliche Kantenüberdeckung EC ! (in *P*; Wie geht's ?)

Finde größtmögliche unabhängige Menge IS ! (*NP*-vollständig)

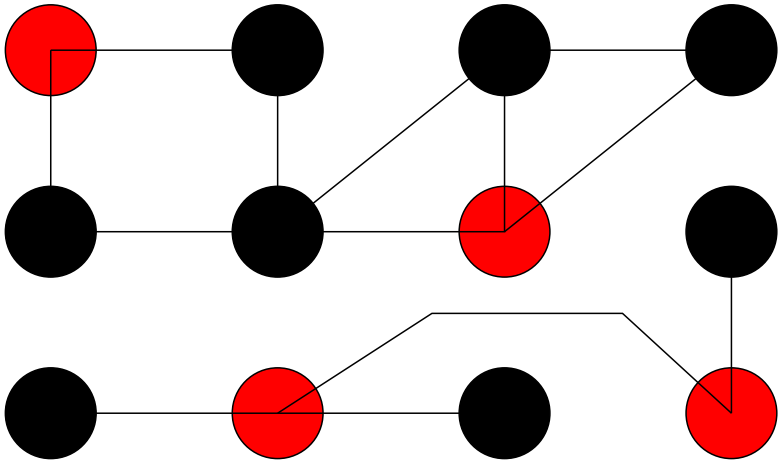
Finde kleinstmögliche dominierende Menge DS ! (*NP*-vollständig)

Beobachte: HS als DS auf bipartiten Graphen.
Frage: Was entspräche dabei der Begrenzung der Kantengröße?!

Unser Beispiel: VC



Unser Beispiel: DS



Einfache Aussagen & Aufgaben

1. Finde einen Zusammenhang zwischen unabhängigen Mengen und Knotenüberdeckungen!

2. Finde eine Kennzeichnung für unabhängige Mengen in Graphkomplementen!

   (Hierzu nützlich zu kennen ist der vollständige (n-Knoten) Graph K_n; ob hierbei wie üblich nur schlichte Graphen betrachtet werden oder Schlingen zugelassen werden ist (algorithmisch) unwichtig.)

3. Die oben aufgelisteten *NP*-harten Probleme sind in *P* für Bäume.

Optimal und Optimum

Folgende Konvention hat sich eingebürgert:
Eine minimale Knotenüberdeckung ist eine Knotenüberdeckung, die keine kleinere Überdeckung enthält (Minimalität bzgl. der Inklusionsordnung).
Eine minimum Knotenüberdeckung ist eine Knotenüberdeckung von kleinster Mächtigkeit.
Da diese sprachliche Unterscheidung im Deutschen sehr unschön ist, verwenden wir kleinstmöglich für *minimum*, im Ggs. zum Wörterbuch ist daher kleinstmöglich für uns nicht das selbe wie minimal.

Entsprechendes gilt für maximal / maximum = größtmöglich.

Ein Beispiel zum Verständnis (oder zur Verwirrung)

IDS:
Finde kleinstmögliche Menge, die sowohl unabhängig als auch dominierend ist!

MMIS:
Finde kleinstmögliche Menge, die eine maximale unabhängige Menge ist!

Behauptung: IDS = MMIS.

Entscheidungs- und Optimierungsprobleme

Besteht die Aufgabe darin, z.B. eine Menge mit Eigenschaft P von kleinstmöglicher Mächtigkeit zu finden, so besteht das natürliche "entsprechende" Entscheidungsproblem darin, bei "kombinierter" Eingabe aus "Minimierungsproblemeingabe" und Zahl k eine Menge mit Eigenschaft P der Mächtigkeit höchstens k zu finden.

"Entsprechendes" gilt bei Maximierungsproblemen.

Optimierungsprobleme liefern "natürlich parameterisierte" Aufgabenstellungen.

Grundbegriffe 2: Parameterisierte Probleme

Grunddefinition

Ein parameterisiertes Problem $\mathcal{P}$ ist ein Entscheidungsproblem zusammen mit einem ausgezeichneten so genannten Parameter.
Formal bedeutet dies: Die Sprache der ✓-instanzen von $\mathcal{P}$, $\mathrm{L}(\mathcal{P})$, ist Teilmenge von $\Sigma^* \times \mathbb{N}$.
Eine Instanz eines parameterisierten Problems $\mathcal{P}$ ist daher ein Paar $(\mathrm{I}, \mathrm{k}) \in \Sigma^* \times \mathbb{N}$.

Ist $\mathcal{P}$ ein parameterisiertes Problem mit $\mathrm{L}(\mathcal{P}) \subseteq \Sigma^* \times \mathbb{N}$ und $\{\mathrm{a}, \mathrm{b}\} \subseteq \Sigma$, dann ist $\mathrm{L_c}(\mathcal{P}) = \{\mathrm{I a b^k} \mid (\mathrm{I}, \mathrm{k}) \in \mathrm{L}(\mathcal{P})\}$ die zugeordnete klassische Sprache.

So können wir auch von der *NP*-Härte eines parameterisierten Problems sprechen.

Ein parameterisiertes Problem $\mathcal{P}$, formal also eine Sprache $L \subseteq \Sigma^* \times \mathbb{N}$, gehört zur

Klasse *FPT* (fixed parameter tractable)

falls es einen Algorithmus gibt, der die Frage "$(x, k) \in L$?" in Zeit

$$f(k) \cdot p(|x|)$$

entscheidet (f bel., p Polynom).

Zum Unterschied zu Brute Force

Brute Force für VC testet alle k-elementigen Mengen einer n-elementigen Knotenmenge, also $\binom{n}{k}$ Möglichkeiten.

Betrachte den Bruch $n^k/(2^k n)$:

	$n = 50$	$n = 150$
$k = 2$	625	5625
$k = 5$	390625	31640625
$k = 20$	$1,8 * 10^{26}$	$2,1 * 10^{35}$

Problemgrößen

Üblich: Länge der Eingabeinstanz in Bits, also $|\mathrm{I}| = \mathfrak{n}$
Achtung: Abhängigkeit von der gewählten Codierung
Daher manchmal “codierungsunabhängige Größenmaße”, bei Graphen:

- Kantenanzahl
- Knotenanzahl

Daher allgemeinere Schreibweise: $\mathrm{size}(\mathrm{I})$

Kerne

Es sei ein parameterisiertes Problem. Eine Kernreduktion (kernelization) ist eine in Polynomzeit berechenbare Abbildung K, welche eine Instanz (I, k) von $\mathcal{P}$ auf eine Instanz $(\mathrm{I}', \mathrm{k}')$ von $\mathcal{P}$ abbildet derart, dass

- (I, k) ist eine ✓-Instanz von $\mathcal{P}$ gdw. $(\mathrm{I}', \mathrm{k}')$ ist eine ✓-Instanz von $\mathcal{P}$,
- $\mathrm{size}(\mathrm{I}') \leq \mathrm{f}(\mathrm{k})$, und
- $\mathrm{k}' \leq \mathrm{g}(\mathrm{k})$ für beliebige Funktionen f und g.

$(\mathrm{I}', \mathrm{k}')$ heißt auch Kern (kernel) (von I), und $\mathrm{size}(\mathrm{I}')$ die Kerngröße.

Von besonderem Interesse: Kerne polynomieller oder gar linearer Größe; hat man sie, so kann man eine Lösung durch naives Durchprobieren “halbwegs effizient” (s.u.) finden.

Eine Kernreduktion heißt echt, falls if $\mathrm{g}(\mathrm{k}) \leq \mathrm{k}$, d.h., $\mathrm{k}' \leq \mathrm{k}$.

Algorithm 1 A brute force *FPT* algorithm from kernelization

Input(s): kernelization function K, a brute-force solving algorithm $\mathcal{A}$ for $\mathcal{P}$, instance (I, k) of $\mathcal{P}$

Output(s): solve (I, k) in *FPT*-time

Compute kernel $(\mathrm{I}', \mathrm{k}') = \mathrm{K}(\mathrm{I}, \mathrm{k})$

Solve $(\mathrm{I}', \mathrm{k}')$ by brute force, i.e., return $\mathcal{A}(\mathrm{I}', \mathrm{k}')$.

Beispiel: Knotenüberdeckungsproblem gemäß S. Buss

Reduktionsregel 1 (Buss Regel) *Ist v ein Knoten vom Grad größer k in der Graph-Instanz (G, k), so lösche v und erniedrige den Parameter um Eins; d.h., die resultierende Instanz ist $(G - v, k - 1)$.*

Lemma 4 *Wird die Größe von Graphen durch # Kanten gemessen, so gilt: Die Regel von Buss liefert eine Kernreduktion.*

Wo steckt das Problem bei anderen Größenmaßen ?

Eine weitere Reduktionsregel (wird leicht vergessen)

Reduktionsregel 2 *Lösche isolierte Knoten (ohne Parameteränderung).*

Lemma 5 *Ist* (G, k) *eine Knotenüberdeckungs-Instanz mit isolierten Knoten* I. (G, k) *ist eine* ✓*-Instanz gdw.* $(G - I, k)$ *ist eine* ✓*-Instanz.*

Jetzt andere Größenmaße ?

Wie wird aus Datenreduktionsregel(n) eine Kernreduktion ?

Algorithm 2 A kernelization algorithm for VC, called Buss-kernel

Input(s): A VC instance (G, k)

Output(s): an instance (G', k') with $k' \leq k$, $|E(G')| \leq (k')^2$, $|V(G')| \leq 2(k')^2$, such that (G, k) is a ✓-instance of VC iff (G', k') is a ✓-instance

if possible **then**
  Apply Rule 2 or Rule 1; producing instance (G', k').
  return Buss-kernel(G', k').
else if $|E(G)| > k^2 \vee k < 0$ **then**
  return $((\{x, y\}, \{\{x, y\}\}), 0)$ {encoding ×}
else
  return (G, k)
end if

Wie wird aus einer Kernreduktion ein FPT-Algorithmus ?

Aufgabe: Wie ist die Laufzeit des Verfahrens ?
Was hat das im Allgemeinen mit der Kerngröße zu tun ?

Algorithm 3 A brute-force algorithm for VC, called VC-kernelization-based

Input(s): A VC instance (G, k)

Output(s): ✓ iff (G, k) is a ✓-instance of VC.

Let (G', k'):=Buss-kernel(G, k). Let $G' = (V, E)$.
if $k' \leq 0$ **then**
  return ($k' = 0$ AND $E = \emptyset$)
else
  for all $C \subseteq V$, $|C| = k'$ **do**
    if C is a vertex cover of G' **then**
      return ✓
    end if
  end for
  return ×
end if

Laufzeit

Datenstrukturabhängig: z.B.: Kann Gradbedingung in $\mathcal{O}(1)$ getestet werden ?

Konkret: Kantenlisten als Datenstruktur.
Dann: Gradbedingung in $\mathcal{O}(\mathrm{k})$ zu testen.
Daher kostet die einmalige Anwendung der Buss Regel $\mathcal{O}(\mathrm{n} \cdot \mathrm{k})$, wobei (wie "üblich") n die Knotenzahl bezeichnet.
Die Regel feuert höchstens k mal (sonst $\times$-Instanz).
Amortisierte Laufzeit: $\mathcal{O}(\mathrm{n} \cdot \mathrm{k})$

Allgemeine Bemerkungen

Einzelne Datenreduktionsregeln liefern häufig “insgesamt” eine Kernreduktion (erschöpfende Anwendung).

Bei Regeln formulieren wir gerne: Wenn . . . , so $\times$-Instanz; dies bedeutet formal, dass eine (triviale, konkrete) $\times$-Instanz zurückgeliefert wird.

Datenreduktionsregeln sind häufig “lokal” und daher “schnell” zu berechnen.

Techniken für *FPT*; Übersicht

1. Wohl(quasi)ordnungen;
2. Graphminoren-Theoreme;
3. Farbcodierung;
4. Baumweite, Verzweigungsweite, . . . ;
5. Reduktionsregeln;
6. Suchbäume.

Zusammenfassung für VC

Satz 6 *(Buss und Mehlhorn)* k*-VC kann in Zeit* $\mathcal{O}(nk + 2^k k^2)$ *gelöst werden.*

Frage: Beweis?!

Frage: MMVC (Maximum minimal vertex cover)

Algorithm 4 A simple search tree algorithm, called VCMH

Input(s): a graph $G = (V, E)$, a positive integer k

Output(s): ✓ if there is a vertex cover $C \subseteq V$, $|C| \leq k$, (and it will implicitly produce such a small cover then) or

× if no vertex cover of size at most k exists.

if $k \leq 0$ and $E \neq \emptyset$ **then**
    return ×
else if $k \geq 0$ and $E = \emptyset$ **then**
    return ✓
else
    Choose edge $e = \{x, y\} \in E$
    if VCMH$(G - x, k - 1)$ **then**
        return ✓
    else
        return VCMH$(G - y, k - 1)$
    end if
end if

Algorithm 5 A simple search tree algorithm, called VCMH-C

Input(s): a graph $G = (V, E)$, a positive integer k

Output(s): a vertex cover $C \subseteq V$, $|C| \leq k$, (if possible) or
$\times$ if no vertex cover of size at most k exists.

if $k \leq 0$ and $E \neq \emptyset$ **then**
  return $\times$
else if $k \geq 0$ and $E = \emptyset$ **then**
  return $\emptyset$
else
  Choose edge $e = \{x, y\} \in E$
  if $C :=$ VCMH-C$(G - x, k - 1) \neq \times$ **then**
    return $C \cup \{x\}$
  else if $C :=$ VCMH-C$(G - y, k - 1) \neq \times$ **then**
    return $C \cup \{y\}$
  else
    return $\times$
  end if{branching}
end if