

Parameterisierte Algorithmen

WS 2007/08 in Trier

Henning Fernau

fernau@uni-trier.de

Parameterisierte Algorithmen

Gesamtübersicht

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

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} , $L(\mathcal{P})$, ist Teilmenge von $\Sigma^* \times \mathbb{N}$.

Eine **Instanz** eines parameterisierten Problems \mathcal{P} ist daher ein Paar $(I, k) \in \Sigma^* \times \mathbb{N}$.

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

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 (I', k') von \mathcal{P} abbildet derart, dass

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

(I', k') heißt auch **Kern (kernel)** (von I), und $\text{size}(I')$ 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 $g(k) \leq k$.

Algorithm 1 A brute force *FPT* algorithm from kernelization

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

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

Compute kernel $(I', k') = K(I, k)$

Solve (I', k') by brute force, i.e., return $A(I', k')$.

Problemkerneigenschaft = FPT

Satz 1 *Ein parameterisiertes Problem liegt in FPT gdw. es einen Problemkern besitzt.*

Beweis: Problemkern \Rightarrow FPT \checkmark (s.o.)

Zu zeigen: Problemkern \Leftarrow FPT

Sei A ein Algorithmus, der ein parameterisiertes Problem P in Zeit $f(k)n^c$ löst.

Wähle triviale \checkmark -Instanz I^+ und triviale \times -Instanz I^- von P .

Bei Eingabe von (I, k) betrachte folgende Reduktion R :

Lasse A (höchstens) n^{c+1} viele Schritte laufen.

Liefert A hierbei Ergebnis (d.h., $f(k)n^c \leq n^{c+1}$), so gibt R entsprechend I^+ oder I^- zurück.

Sonst gilt: $n < f(k)$, sodass R als Ergebnis (I, k) zurückliefern kann. \square

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

Wo steckt das Problem bei anderen Größenmaßen ?

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.

Formaler: Zu Datenreduktionsregel R gibt es eine **Eintrittsbedingung** c_R und eine dadurch ausgelöste **Aktion** α_R , die eine eingegebene Instanz (I, k) modifiziert.

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

Eine Kernreduktion aus Datenreduktionsregeln

Algorithm 2 A generic kernelization algorithm from reduction rules

Input(s): a collection \mathcal{R} of reduction rules

Output(s): kernelization function $K[\mathcal{R}]$

Let $(I', k') = (I, k)$.

while $\exists R \in \mathcal{R} : c_R(I', k')$ is true **do**

 Choose some $R \in \mathcal{R}$ such that $c_R(I', k')$ is true

 Update $(I', k') := \alpha_R(I, k)$

end while

return (I', k')

In diesem Sinne vollständiges Regelsystem für VC

Betrachte Graph-Instanz (G, k) von VC, $G = (V, E)$:

0. $E \neq \emptyset \wedge k \leq 0 \Rightarrow \times$ -Instanz.
1. $\exists v \in V : \deg(v) > k \Rightarrow$ neue Instanz $(G - v, k - 1)$.
2. $(\forall v \in V : \deg(v) \leq k) \wedge |E| > k^2 \Rightarrow \times$ -Instanz.
3. $\exists v \in V : \deg(v) = 0 \Rightarrow$ neue Instanz $(G - v, k)$.

Eine weitere Transferaufgabe

Finde Reduktionsregeln für 3-HS !

Exkurs: **Eine Erbschaft von Stirling**

Lemma 3 For any $a > 1$,

$$\binom{ak}{k} \approx a^k \left(\frac{a}{a-1}\right)^{(a-1)k} = \left(\frac{a^a}{(a-1)^{(a-1)}}\right)^k \leq (ea)^k$$

Beweis:

$$\begin{aligned} \binom{ak}{k} &= \frac{(ak)!}{k!((a-1)k)!} \\ &\approx \frac{\sqrt{2\pi ak}}{\sqrt{2\pi k}\sqrt{2\pi(a-1)k}} \cdot \left(\frac{ak}{e}\right)^{ak} \cdot \left(\frac{e}{k}\right)^k \cdot \left(\frac{e}{(a-1)k}\right)^{(a-1)k} \\ &= \frac{\sqrt{a}}{\sqrt{2\pi(a-1)k}} a^k \left(\frac{a}{a-1}\right)^{(a-1)k} \\ &\leq a^k \left(\frac{a}{a-1}\right)^{(a-1)k} \\ &\leq (ea)^k \end{aligned}$$

□

Zur Existenz linearer Kerne

Häufig anwendbar: tiefe mathematische Resultate

Wenn nicht: Arbeit “zu Fuß” oft SEHR aufwändig

Beispiel. Vierfarbensatz impliziert: $4k$ -Kern für das Auffinden größtmöglicher unabhängiger Mengen in planaren Graphen.

Algorithm 3 Color-based FUNCTIONAL kernelization for PIS

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

Output(s): either independent set I with $|I| = k$ or $|V| \leq 4(k - 1)$

Find a 4-coloring of G , formalized as a mapping $c : V \rightarrow \{1, 2, 3, 4\}$.

Let $V_i = \{v \in V \mid c(v) = i\}$.

Let V' be the largest among the V_i .

{Each coloring is an independent set.}

5: **if** $|V'| \geq k$ **then**

 return $I \subseteq V'$ with $|I| = k$

else

$\{\forall \text{ colors } i: |V_i| < k; \text{ hence, } |V| \leq 4(k - 1)\}$

end if

Ein Satz von Nemhauser und Trotter

Satz 4 Gegeben sei ein Graph $G = (V, E)$ mit $n = |V|$ und $m = |E|$. Dann lassen sich in Zeit $\mathcal{O}(\sqrt{n} \cdot m)$ zwei disjunkte Knotenmengen C_c und V_c berechnen mit folgenden Eigenschaften:

- Ist $C' \subseteq V_c$ eine Knotenüberdeckung von $G[V_c]$, so ist $C = C' \cup C_c$ eine Knotenüberdeckung von G .
- Es gibt eine kleinstmögliche Knotenüberdeckung C^* von G , die C_c enthält.
- $G[V_c]$ hat kleinstmögliche Knotenüberdeckung der Mindestgröße $|V_c|/2$.

Folgerung 5 Sei (G, k) eine Eingabeinstanz von VC.

In Zeit $\mathcal{O}(|G| + k^3)$ lässt sich ein echter Problemkern (G', k') berechnen mit $|V(G')| \leq 2k$.

Beweis: Berechne erst Buss-Kern, dann wende den Satz von Nemhauser und Trotter an. $G[V_c]$ bildet den Problemkern. \square

Hinweis: Zusammenhang mit Approximation.

Exkurs: Matchings und Knotenüberdeckungen

Lemma 6 *Ist M ein Matching und C eine Knotenüberdeckung von G , so gilt:
 $|M| \leq |C|$.*

Beweis: Jede Matching-Kante muss durch C abgedeckt werden. \square

Bemerkung 7 *Die Größe eines größtmöglichen Matchings ist daher eine untere Schranke für die Größe einer kleinstmöglichen Knotenüberdeckung.*

Satz 8 (König und Egerváry) *Für bipartite Graphen gilt Gleichheit.*

Einige Definitionen für den Beweis von Satz 4

Sei $G = (V, E)$ ein Graph;

seine **bipartite Variante** $G_{BP} = (V \times \{1, 2\}, E_{BP})$ ist definiert durch:

$$E_{BP} = \{ \{(u, 1), (v, 2)\} \mid \{u, v\} \in E \}.$$

Für eine kleinstmögliche Knotenüberdeckung C_{BP} von G_{BP} sei ferner

$$C_{BP}^{\text{AND}}(G) = \{v \in V \mid (v, 1) \in C_{BP} \wedge (v, 2) \in C_{BP}\},$$

$$C_{BP}^{\text{XOR}}(G) = \left\{ v \in V \setminus C_{BP}^{\text{AND}}(G) \mid (v, 1) \in C_{BP} \vee (v, 2) \in C_{BP} \right\}.$$

Algorithm 4 A kernelization algorithm for VC, called VCNT

Input(s): an instance (G, k) of VC

Output(s): an equivalent instance (G', k') of VC with $V(G') \subseteq V(G)$, $|V(G')| \leq 2k'$ and $k' \leq k$, as well as a set $C \subseteq V(G)$ that must appear in any vertex cover for G (and that is not accounted into the budget k').

Create G_{BP} from G .

Compute a minimum vertex cover C_{BP} of G_{BP} .

Compute $C := C_{BP}^{AND}(G)$ and $C_{BP}^{XOR}(G)$.

Let $k' := k - |C|$.

5: **if** $k' < 0$ **then**

 Pick an edge $e = \{u, v\}$ from G .

 Define $G' = (\{u, v\}, \{e\})$ and $k' = 0$.

 Let $C := \emptyset$.

else

10: Let $G' = G[C_{BP}^{XOR}(G)]$.

end if

 return (G', k') as well as C .

Beweis: (von Satz 4) Setze $C_c = C_{BP}^{AND}(G)$, $V_c = C_{BP}^{XOR}(G)$.

(i) Ist C' VC von $G[V_c]$, so ist $C := C' \cup C_c$ VC von G .

Betrachte $xy \in E$; zu zeigen: $x \in C$ oder $y \in C$.

1. Liegt $x \notin C_c \cup V_c$, so $\{(x, 1), (x, 2)\} \cap C_{BP} = \emptyset \Rightarrow \{(y, 1), (y, 2)\} \subseteq C_{BP} \Rightarrow y \in C_c$.
2. $y \notin C_c \cup V_c \Rightarrow x \in C_c$ entsprechend.
3. $x \in C_c$ oder $y \in C_c$ ist trivial.
4. $\{x, y\} \subseteq V_c \Rightarrow (x \in C' \text{ oder } y \in C')$

(ii) Es gibt kleinstmögliches VC C^* von G mit $C_c \subseteq C^*$.

Sei zunächst C^* eine beliebige kleinstmögliche Knotenüberdeckung.

Definiere hierzu: $C_c^* = C_c \cap C^*$ und $V_c^* = V_c \cap C^*$.

Bem.: $C_B = \{(x, 1) \mid x \in V_c \cup C_c \cup C^*\} \cup \{(y, 2) \mid y \in C_c^*\}$ is VC von G_{BP}

Betrachte $e = \{(x, 1), (y, 2)\} \in E_{BP}$ (also $xy \in E$). Wird e überdeckt?

1. $x \in V_c \cup C_c \cup C^*$ ✓

2. $x \notin V_c \cup C_c \cup C^* \Rightarrow y \in C_c$ (siehe Beweis zu (i), Fall 1.); ferner: $x \notin C^* \Rightarrow y \in C^*$, da C^* VC. Also: $y \in C_c^*$, d.h., $(y, 2) \in C_B$.

(ii) Es gibt kleinstmögliches VC von G , die C_c enthält. (Forts.)

Genauer: $C_c \cup V_c^*$ ist kleinstmögliches VC.

z.z.: $|C_c \cup V_c^*| \leq |C^*|$ (über $|C_c| \leq |C^* \setminus V_c| \rightsquigarrow |C_c| + |V_c^*| \leq |C^* \setminus V_c^*| + |V_c^*| = |C^*|$):

$$\begin{aligned} |V_c| + 2|C_c| &= |V_c \cup C_c \times \{1, 2\}| \\ &= |C_{BP}| \text{ (Def. von } V_c \text{ und von } C_c) \\ &\leq |C_B| \text{ (Zwischenbehauptung / Bem.)} \\ &= |V_c \cup C_c \cup C^*| + |C_c^*| \\ &\leq |V_c| + |C_c| + |C^* \setminus (V_c \cup C_c)| + |C_c^*| \\ \Rightarrow |C_c| &\leq |C^* \setminus (V_c \cup C_c)| + \underbrace{|C_c^*|}_{|C_c \cap C^*|} \\ &= |C^* \setminus V_c| \end{aligned}$$

(iii) $G[V_c]$ hat ein kleinstmögliches VC der Größe $\geq |V_c|/2$.

Sei V^* kleinstmögliches VC von $G[V_c]$.

Mit (i) ist $C_c \cup V^*$ ein VC von G .

Per Def. von G_{BP} ist $(C_c \cup V^*) \times \{1, 2\}$ VC von G_{BP} .

$$\begin{aligned} |V_c| + 2|C_c| &= |C_{BP}| \quad (\text{Def. von } V_c \text{ und von } C_c) \\ &\leq |(C_c \cup V^*) \times \{1, 2\}| \quad (\text{Optimalität von } C_{BP}) \\ &= 2|V_c| + 2|V^*| \end{aligned}$$

$$\Rightarrow |V_c| \leq 2|V^*|$$

□

Greedy-Algorithmen für Kernreduktionen

Klappt manchmal bei **Maximierungsproblemen**

Erinnerung planare Graphen

Eulersche Flächenformel / Polyederformel

für zusammenhängende planare Graphen mit wenigstens zwei Knoten

$$f - 2 = m - n$$

n: Knotenzahl

m: Kantenzahl

f: Facettenzahl

Folgerung 9 *In einem planaren Graphen gibt es stets einen Knoten vom Grad höchstens fünf.*

Folgerung 10 *Greedy gestattet einen $6k$ Kern für PIS.*

Frage: MMVC auf planaren Graphen?

Frage: MMIS auf planaren Graphen?

Algorithm 5 Greedy kernelization for PIS

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

Output(s): either independent set I with $|I| = k$ or $|V| \leq 6(k - 1)$

$G' := G; i := 0; I := \emptyset$

while $V(G') \neq \emptyset$ and $i < k$ **do**

 pick vertex $v \in V(G')$ of smallest degree

$G' := (G' - N[v])$ $\{|N[v]| \leq 6 (*)\}$

5: $i := i + 1$

$I := I \cup \{v\}$ $\{I \text{ is an independent set of size } i \text{ in } G. [+]\}$

end while

if $i = k$ **then**

I is an independent set of size k in G .

10: **else** $\{V(G') = \emptyset \text{ and } i < k\}$

$\{ (*) \Rightarrow \leq 6(k - 1) \text{ vertices have been taken out before exhausting } V.\}$

end if

Parameterisierte Dualität

Reparameterisierungsmethode “bekannter” Probleme

Bsp.: VC vs. IS durch Komplementierung.

Bsp.: Nonblocker Komplement von DS.

Greedy bei Nonblocker

Satz 11 (Ore 1962) *Ist $G = (V, E)$ ein Graph ohne isolierte Knoten, so gibt es eine dominierende Menge $D \subseteq V$ mit $|D| \leq |V|/2$.*

Beweis: Sei zusätzlich G zusammenhängend. Dann gibt es in G einen aufspannenden Baum $T = (V, E')$, $E' \subseteq E$. Wähle willkürlich Wurzel $r \in V$ und definiere:

$$V_i = \{v \in V \mid d_T(r, v) = i\}, \quad i = 0, 1, 2, \dots$$

$$D_j = \bigcup_{i \equiv j \pmod{2}} V_i, \quad j = 0, 1$$

sind dominierende Mengen, die kleinere erfüllt die Behauptung. \square

Wie wird daraus ein Kern für Nonblocker?

Wenn Graph G keine isolierten Knoten aber mehr als $2k$ Knoten hat, so ist (G, k) eine ✓-Instanz.

Umgang mit isolierten Knoten?

Reduktionsregel 2 *Nimm isolierte Knoten in Nonblocker-Menge!*

Satz 12 *Nonblocker besitzt einen $2k$ Kern.*

Geht es besser?!

Versuche, “schlimmste Fälle” zu konstruieren. . .