

Bachelorarbeit
Untersuchung von Datenreduktionsregeln beim
Kontenüberdeckungsproblem

Benedikt Lücken-Winkels

12. Januar 2018

Inhaltsverzeichnis

I	TODO	1
1	Implementierung	2
2	Vorarbeit	2
II	Idee und Vorgehensweise	2
3	Thema	2
4	Fixed-Parameter-Algorithm	2
5	Definitionen/Erklärungen	3
5.1	Regulärer Graph	3
5.2	Dominating Set	3
5.3	Matching	3
5.3.1	Maximal Matching	3
5.3.2	Maximum (Cardinality) Matching	3
5.3.3	Perfect/Complete Matching	3
5.4	Independent Set	3
5.5	Bipartider Graph s19	3
6	Knotenüberdeckung/Vertexcover	4
6.1	Algorithmus s99	4
7	Graph-Reduktion	4
7.1	Reduktionsregeln	5
7.1.1	Einfache Regeln	5
7.1.2	Nemhauser/Trotter - Regel s64	5
7.1.3	Kronenregel	5

Teil I

TODO

1 Implementierung

- Maximal- und Maximummatching implementieren `mcb_matching.html`
- bipartiden Graphen erkennen

2 Vorarbeit

- Algorithmen in Tex schreiben
- KÜ-Algorithmus finden
- Andere Algorithmen ausarbeiten

Teil II

Idee und Vorgehensweise

3 Thema

Reihenfolge von Reduktionsregeln und Einfluss auf Laufzeit von Knotenüberdeckung

Zu prüfen:

- Wie oft feuert welche Regel?
- Welche Regeln haben den größten Effekt auf die Graphstruktur?
Ist der Graph bipartid
- Voraussetzungen für Anwendungen der Regeln
Wie sehen Graphen aus, auf die keine Regel anwendbar ist
Wie sehen Graphen aus, auf die nur eine Regel anwendbar ist
(und keine andere)

4 Fixed-Parameter-Algorithm

- NP-schwere Probleme
- exponentielle Laufzeit
- meist wird auf der Größe der Lösungsmenge parametrisiert
- exponentieller Faktor hängt nur von einem Parameter ab
- Fixed-Parameter-Algorithms lösen Probleme mit einer Eingabeinstanz der Größe n und parameter k in $f(k) * n^{O(1)}$

- Vorteile von FPAs
 - die Lösung ist garantiert optimal
 - die obere Schranke der Komplexität ist beweisbar

5 Definitionen/Erklärungen

5.1 Regulärer Graph

Ein Graph G ist *regulär*, wenn $\forall x, y \in G : \text{Grad}(x) = \text{Grad}(y)$ (für $x \neq y$)

5.2 Dominating Set

$G = (V, E)$, *nonnegative int* k , $S \subseteq V$, ($|S| \leq k$, $v \in S \vee v$ has a neighbor in S)

5.3 Matching

In einem Graphen $G = (V, E)$ ist $M \subseteq E$ ein Matching wenn keine 2 Kanten den selben Knoten haben

5.3.1 Maximal Matching

- Wenn irgendeine Kante zum Matching M hinzugefügt wird, ist das M kein Matching mehr
- M ist keine Teilmenge eines anderen Matchings

5.3.2 Maximum (Cardinality) Matching

- Größte Menge an Kanten
- *matching number* $v(G)$ ist die gröÙe eines Maximum Matchings von G

5.3.3 Perfect/Complete Matching

- Jeder Knoten ist indiziert in M

5.4 Independent Set

$$G = (V, E), U \subset V, \forall v, w \in U : (v, w) \notin E$$

5.5 Bipartider Graph s19

Jeder Knoten ist in genau einem von zwei Teilmengen. Innerhalb einer Teilmenge ist kein Knoten benachbart

Knotenfärbung:

```

0 Startknoten n wird Farbe U zugeordnet , dann allen Nachbarn Farbe V
1 Wiederhole Vorgang für alle Nachbarn
2 Wenn einem Knoten eine Farbe zugeordnet werden soll ,
3 die eine andere ist , als die , die er hat :
4     return Nicht-Bipartid
```

6 Knotenüberdeckung/Vertexcover

$G = (V, E)$, *nonnegative int* k , $C \subseteq V$, $(C : |C| \leq k, \text{ each edge in } E \text{ has one endpoint in } C)$

Es gibt eine Knotenüberdeckung der Größe k , wenn es ein Independent Set der Größe $n - k$ gibt s32

- Parametrisierung: s41 (Auswahl der richtigen Parametrisierung)
Größe der Menge (k) der zu findenden Knotenüberdeckung
Gibt es eine Knotenüberdeckung der Größe $n - k$? ($n = |V|$) \rightarrow Nicht FPT

6.1 Algorithmus s99

- Suchbaum, dessen Tiefe durch k begrenzt ist

```

0  Branching:
1  Knoten  $x \in G$ , Vertex Cover( $G$ )= $C=\emptyset$ , Graph  $G$ 
2  switch( $\exists x$  mit  $\text{Grad}(x)$ ):
3      case = 1:  $C \cup N(x)$  (kein anderer Branch)
4      case > 4: Branch mit  $x$  und  $N(x)$ 
5      case 2-4 und  $G$  regulär: Branch mit  $x$  und  $N(x)$ 
6      case 2: verwende degree-two-vertices
7      case 3: verwende degree-three-vertices
8
9  Degree-two-vertices:
10  $N(x)=\{a, b\}$ 
11 switch:
12     case  $N(a)=b$ :  $C \cup \{a, b\}$ 
13     case  $N(a)=N(b)=c (\neq x)$ :  $C \cup \{x, c\}$ 
14     case default: Branch mit  $N(x)$  und  $N(a) \cup N(b)$ 
15
16 Degree-three-vertices:
17  $N(x)=\{a, b, c\}$ ,  $d \in G$ 
18 switch:
19     case Dreieck mit  $\{x, a, b\}$  (mehr Dreiecke möglich):
20         Branch mit  $N(x)$  und  $N(c)$ 
21     case Kreis/Cycle mit  $\{x, a, b, d\}$ :
22         Branch mit  $N(x)$  und  $\{x, d\}$ 
23     case  $a, b, c$ =keine Nachbarn  $\text{Grad}(a)=4$ :
24         Branch mit  $N(x)$  und  $N(a)$  und  $\{a\} \cup N(b) \cup N(c)$ 

```

Beispiel aus Algorithms on Trees and Graphs von Gabriel Valiente, Springer-Verlag Berlin Heidelberg 2002, Seite 333

7 Graph-Reduktion

- Einfache Teile des Graphen entfernen, sodass nur der Kern des Problems, bzw seine Schwierigkeit übrig bleibt. s51

- Knotenüberdeckung kann für Graphen mit fester Baumbreite effizient gelöst werden s51
- Reduktion der Eingabe auf den Problemkern immer sinnvoll s53

7.1 Reduktionsregeln

7.1.1 Einfache Regeln

- Ein isolierter Knoten ist automatisch in der KÜ s54
- Bei einem Knoten des Grades 1 wird der Nachbar automatisch hinzugefügt, da er evtl noch weitere Kantenabdeckt s54
- Ein Knoten des Grades $k+1$ ($|Lösungsmenge| \leq k$) wird automatisch hinzugefügt, da sonst $k+1$ Elemente in der Menge wären s54

7.1.2 Nemhauser/Trotter - Regel s64

- Maximum Matching:

Königs Minimax Theorie: Bei bipartiden Graphen ist die Größe des Maximum Matching gleich der Größe der Minimalen Knotenüberdeckung s65

Die Maximum Matching ist die Größte unter den gültigen Matchings des Graphen (Wikipedia, Matching)

```

0  G = (V, E)
1  Bipartiden Graphen erstellen B = (V, V', E')
2  V' ist Kopie von V
3  E' := { {x, y'}, {x', y} | {x, y} ∈ E }
4  Maximum Matching M mit mcb_machting bestimmen
5  VC(B) = CB mit Satz von König bestimmen (mcb_machting)
6  VC(G[V0]) ∪ C0 = VC(G)

```

7.1.3 Kronenregel

- Verallgemeinerung der Grad 1 Emiminierung → einfachste Kronenregel 69
- Die Krone eines besteht aus dem *Independent Set* I und $H = N(I)$ mit $H \cap I = \emptyset$ und die Kanten zwischen I und H indizieren (matchen) alle Knoten aus H 69

Cygan Parameterized Algorithm 69 (255)