Kolloquium zur Bachelorarbeit

Referent: Benedikt Lüken-Winkels

Prüfer: Prof. Dr. Henning Fernau
Prof. Dr. Stefan Näher

07. März 2018

Universität Trier

Knotenüberdeckungsproblem

Knotenüberdeckung

EINGABE: *Graph* $G = (V, E)$, *positive Integer* $k \leq |V|$

AUSGABE: $S \subseteq V$ *mit* $|S| \leq k$, *sodass jede Kante aus* E *einen Endpunkt in* S *hat.*

Graphreduktion

Einfache Reduktionsregeln

Kronenregel

Kronenregel - Algorithmus

$G = (V, E)$
M_1 := Maximal Matching von G
  $M_1 := \emptyset$
  $\forall e \in$ E:
    $M_1 = M_1 \cup e$
    Entferne e und $N(e)$ aus der weiteren Betrachtung
O := nicht gepaarte Knoten in M_1
M_2 := maxmimum matching von $B = G[O \cup N(O)]$
I := nicht gepaarte Knoten aus O in M_2
$I' := \emptyset$
while $I' \neq I$
  $I' := I$
  $H := N(I)$
  $I := I \cup \{\forall u \in O | \exists v \in H(uv \in M_2)\}$
Entferne $N(I)$ aus G

Nemhauser-Trotter-Regel

Nemhauser-Trotter-Regel

Nemhauser-Trotter-Theorem

Für einen Graphen $G = (V, E)$ können zwei disjunkte Mengen C_0 und V_0 gefunden werden, sodass

1. *C_0 in einer minimalen Knotenüberdeckung von G enthalten ist,*
2. *der Teilgraph $G[V_0]$ eine Knotenüberdeckung der Größe $\leq |V_0|/2$ hat,*
3. *und $VC(G) = VC(G[V_0]) \cup C_0$ gilt.*

Nemhauser-Trotter-Regel

Nemhauser-Trotter-Theorem

Für einen Graphen $G = (V, E)$ *können zwei disjunkte Mengen* C_0 *und* V_0 *gefunden werden, sodass*

1. C_0 *in einer minimalen Knotenüberdeckung von* G *enthalten ist,*
2. *der Teilgraph* $G[V_0]$ *eine Knotenüberdeckung der Größe* $\leq |V_0|/2$ *hat,*
3. *und* $VC(G) = VC(G[V_0]) \cup C_0$ *gilt.*

Nemhauser-Trotter-Regel

Nemhauser-Trotter-Theorem

Für einen Graphen $G = (V, E)$ *können zwei disjunkte Mengen* C_0 *und* V_0 *gefunden werden, sodass*

1. C_0 *in einer minimalen Knotenüberdeckung von* G *enthalten ist,*
2. *der Teilgraph* $G[V_0]$ *eine Knotenüberdeckung der Größe* $\leq |V_0|/2$ *hat,*
3. *und* $VC(G) = VC(G[V_0]) \cup C_0$ *gilt.*

$G = (V, E)$

Bipartiden Graphen erstellen $B = (V, V', E')$

  mit $E' := \{\{x, y'\}, \{x', y\} | \{x, y\} \in E\}$

Maximum Matching M von B bestimmen

$C_B := VC(B)$

$C_0 := \{x \in V \mid x \in C_B \text{ und } x' \in C_B\}$

$V_0 := \{x \in V \mid \text{entweder } x \in C_B \text{ oder } x' \in C_B\}$

NT-Regel - Algorithmus

```
G = (V, E)
Bipartiden Graphen erstellen B = (V, V', E')
  mit E' := {{x, y'}, {x', y}|{x, y} ∈ E}
Maximum Matching M von B bestimmen
C_B := VC(B)
C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

NT-Regel - Algorithmus

```
G = (V, E)
Bipartiden Graphen erstellen B = (V, V', E')
  mit E' := {{x, y'}, {x', y} | {x, y} ∈ E}
Maximum Matching M von B bestimmen
C_B := VC(B)
C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

NT-Regel - Algorithmus

$G = (V, E)$
`Bipartiden Graphen erstellen` $B = (V, V', E')$
`mit` $E' := \{\{x, y'\}, \{x', y\} | \{x, y\} \in E\}$
`Maximum Matching` M `von` B `bestimmen`
$C_B := VC(B)$
$C_0 := \{x \in V \mid x \in C_B \ \ und \ \ x' \in C_B\}$
$V_0 := \{x \in V \mid entweder \ \ x \in C_B \ \ oder \ \ x' \in C_B\}$

NT-Regel - Algorithmus

```
G = (V, E)
Bipartiden Graphen erstellen B = (V, V', E')
   mit E' := {{x, y'}, {x', y}|{x, y} ∈ E}
Maximum Matching M von B bestimmen
C_B := VC(B)
C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

NT-Regel - Algorithmus

```
G = (V, E)
Bipartiden Graphen erstellen B = (V, V', E')
  mit E' := {{x, y'}, {x', y}|{x, y} ∈ E}
Maximum Matching M von B bestimmen
C_B := VC(B)
C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

NT-Regel - Laufzeit

0 $G = (V, E)$

1 `Bipartiden Graphen erstellen` $B = (V, V', E')$

2 `mit` $E' := \{\{x, y'\}, \{x', y\} | \{x, y\} \in E\}$

3 `Maximum Matching` M `von` B `bestimmen`

4 $C_B := VC(B)$

5 $C_0 := \{x \in V \mid x \in C_B \;\; und \;\; x' \in C_B\}$

6 $V_0 := \{x \in V \mid entweder \;\; x \in C_B \;\; oder \;\; x' \in C_B\}$

Zeilen 1-2: $n \cdot 2m$

NT-Regel - Laufzeit

```
0  G = (V, E)
1  Bipartiden Graphen erstellen B = (V, V', E')
2     mit E' := {{x, y'}, {x', y} | {x, y} ∈ E}
3  Maximum Matching M von B bestimmen
4  C_B := VC(B)
5  C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
6  V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

Zeilen 3-4: $\sqrt{n} \cdot m$ (LEDA:mcb_matching, Hopcroft and Karp)

NT-Regel - Laufzeit

```
0  G = (V, E)
1  Bipartiden Graphen erstellen B = (V, V', E')
2     mit E' := {{x, y'}, {x', y} | {x, y} ∈ E}
3  Maximum Matching M von B bestimmen
4  C_B := VC(B)
5  C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
6  V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

Zeilen 5-6: $2n + k \cdot d$

NT-Regel - Laufzeit

```
G = (V, E)
Bipartiden Graphen erstellen B = (V, V', E')
  mit E' := {{x, y'}, {x', y}|{x, y} ∈ E}
Maximum Matching M von B bestimmen
C_B := VC(B)
C_0 := {x ∈ V | x ∈ C_B und x' ∈ C_B}
V_0 := {x ∈ V | entweder x ∈ C_B oder x' ∈ C_B}
```

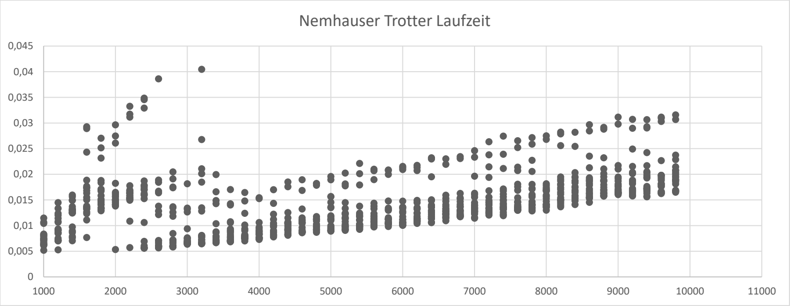
$n \cdot 2m + \sqrt{n} \cdot m + 2n + k \cdot d$

NT-Regel - Laufzeit

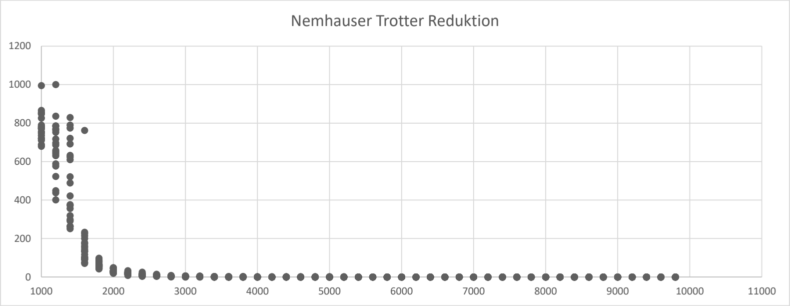
$G = (V, E)$
`Bipartiden Graphen erstellen` $B = (V, V', E')$
`mit` $E' := \{\{x, y'\}, \{x', y\} | \{x, y\} \in E\}$
`Maximum Matching` M `von` B `bestimmen`
$C_B := VC(B)$
$C_0 := \{x \in V \mid x \in C_B \;\; und \;\; x' \in C_B\}$
$V_0 := \{x \in V \mid entweder \;\; x \in C_B \;\; oder \;\; x' \in C_B\}$

$n \cdot 2m + \sqrt{n} \cdot m + 2n + k \cdot d \Rightarrow O(2mn)$

NT-Regel - Laufzeit



NT-Regel - Ergebnisse



Vergleich

Anwendung

Tabelle 1: Anwendung kombinierter Reduktionsregeln

Kombination	**Anwendungen$_1$**	**Anwendungen$_2$**	**Anwendungen$_3$**	**Reduktion**
K - G_1	3.63	4.3	-	331.8
G_1 - K	4.37	3.22	-	331.17
K - NT	0.8	0.38	-	68.28
NT - K	0.45	0.56	-	68.6
G_1 - NT	1.33	0.017	-	99.87
NT - G_1	0.28	1.13	-	99.87
K - G_1 - NT	3.61	4.29	0.11	334.67
K - NT - G_1	3.6	0.87	3.39	334.83
G_1 - NT - K	4.36	0.12	3.2	334.17
G_1 - K - NT	3.61	3.2	0.65	334.16
NT - K - G_1	0.39	3.44	4.03	335.2
NT - G_1 - K	0.91	3.42	3.2	334.16

Implementierung

Fazit