

Technische Universität Chemnitz



Diplomarbeit

**Effiziente Approximation
unabhängiger Mengen in Graphen**

Matthias Baumgart

Chemnitz, 15. September 2004

Gliederung

1. Aufgabenstellung
2. Der Greedy Algorithmus
3. Der Ramsey Algorithmus
4. Der Algorithmus von Feige
5. Praktische Untersuchungen
6. Zusammenfassung

Aufgabenstellung

Ziel der Diplomarbeit:

- Untersuchung der Approximierbarkeit der Unabhängigkeitszahl in Graphen in Polynomialzeit
- Insbesondere: Algorithmen von Halldórsson und Boppana sowie der neuere Algorithmus von Feige
- Analyse der Approximationsgüte
- Implementierung und praktische Untersuchungen ausgewählter Algorithmen

Greedy Algorithmen

Einfache Vorgehensweise des Greedy Algorithmus:

0. Setze $I := \emptyset$
1. Falls der Graph $G = (V, E)$ keine Knoten enthält, HALT
2. Wähle einen beliebigen Knoten $v \in V$
3. Füge den Knoten v zur unabhängigen Menge I hinzu
4. Lösche den Knoten v sowie seine Nachbarn $N(v)$ und gehe zu 1.

- Nachteil: Approximationsgüte von $O(n)$, wobei $n = |V|$.
- Der Greedy Algorithmus liefert kleine unabhängige Mengen, wenn die Nachbarschaft eines gewählten Knoten eine große unabhängige Menge enthält.

- Nachteil: Approximationsgüte von $O(n)$, wobei $n = |V|$.
- Der Greedy Algorithmus liefert kleine unabhängige Mengen, wenn die Nachbarschaft eines gewählten Knoten eine große unabhängige Menge enthält.

Verbesserung von Boppana und Halldórsson: Suche ebenfalls in der Nachbarschaft weiter nach einer unabhängigen Menge.

Diese Vorgehensweise ist im Ramsey Algorithmus dargestellt.

Der Ramsey Algorithmus

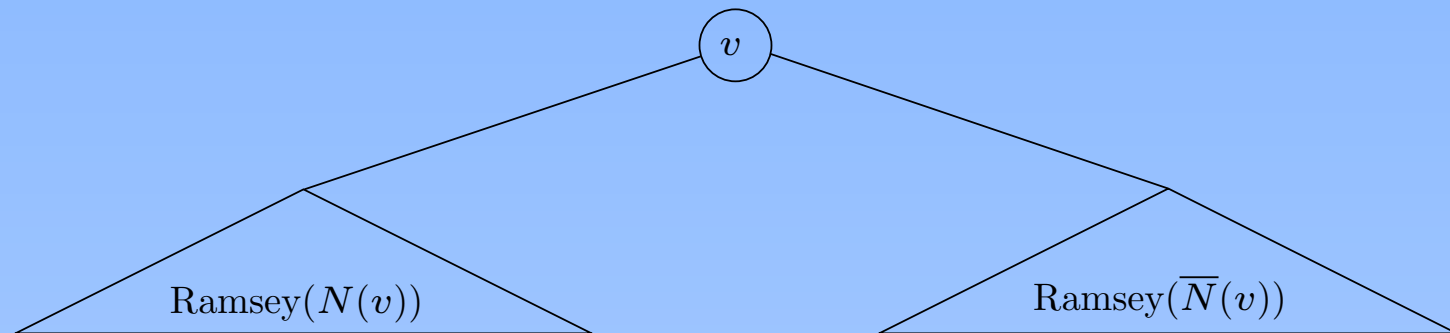
Eingabe: Graph $G = (V, E)$

Ausgabe: Clique C und unabhängige Menge I

Algorithmus Ramsey($G = (V, E)$)

- 1: **if** $V = \emptyset$ **then**
- 2: **return** (\emptyset, \emptyset)
- 3: **choose** $v \in V$
- 4: $(C_1, I_1) \leftarrow$ Ramsey($N(v)$)
- 5: $(C_2, I_2) \leftarrow$ Ramsey($\overline{N}(v)$)
- 6: $C \leftarrow \max\{C_1 \cup \{v\}, C_2\}$
- 7: $I \leftarrow \max\{I_1, I_2 \cup \{v\}\}$
- 8: **return** (C, I)

Die Funktionsweise des Ramsey Algorithmus lässt sich durch einen binären Baum veranschaulichen.



Eine Analyse mit Hilfe der Ramsey-Theorie ergibt für die vom Ramsey Algorithmus gefundene unabhängige Menge I und Clique C

$$|I| \cdot |C| \geq \frac{1}{4} \cdot (\log n)^2 .$$

Weitere Verbesserung: Entferne gefundene Clique und suche erneut nach unabhängiger Menge, bis keine Knoten mehr vorhanden sind.

Algorithmus Clique-Removal($G = (V, E)$)

```
1:  $i \leftarrow 1$ 
2:  $(C_i, I_i) \leftarrow \text{Ramsey}(G)$ 
3:  $I \leftarrow I_i$ 
4: while  $V \neq \emptyset$  do
5:      $V \leftarrow V \setminus C_i$ 
6:      $i \leftarrow i + 1$ 
7:      $(C_i, I_i) \leftarrow \text{Ramsey}(G)$ 
8:     if  $|I_i| > |I|$  then
9:          $I \leftarrow I_i$ 
10: return  $(I, \{C_1, C_2, \dots, C_i\})$ 
```

Satz 1 *Der Algorithmus Clique-Removal approximiert für einen Graphen $G = (V, E)$ auf $n = |V|$ Knoten eine unabhängige Menge I mit Güte*

$$O\left(\frac{n}{(\log n)^2}\right).$$

Satz 1 *Der Algorithmus Clique-Removal approximiert für einen Graphen $G = (V, E)$ auf $n = |V|$ Knoten eine unabhängige Menge I mit Güte*

$$O\left(\frac{n}{(\log n)^2}\right).$$

Analog kann man einen Algorithmus Independent-Set-Removal konstruieren, welcher eine Clique mit Güte $O(n/(\log n)^2)$ approximiert.

Dazu wird statt einer Clique in jeder Iteration eine unabhängige Menge aus dem Graphen $G = (V, E)$ entfernt, bis keine Knoten mehr in G vorhanden sind.

Der Algorithmus von Feige

Idee von Feige: Entferne Untergraphen mit „wenigen“ Kanten.

Der Algorithmus von Feige

Idee von Feige: Entferne Untergraphen mit „wenigen“ Kanten.

Definition 2 Sei $G = (V, E)$ ein Graph mit einer Clique C der Kardinalität $|C| \geq |V|/k$. Eine Knotenmenge S in G heißt schwach, wenn der auf S induzierte Untergraph von G keine Clique C_S der Kardinalität $|C_S| \geq |S|/(2k)$ enthält.

Der Algorithmus von Feige

Idee von Feige: Entferne Untergraphen mit „wenigen“ Kanten.

Definition 2 Sei $G = (V, E)$ ein Graph mit einer Clique C der Kardinalität $|C| \geq |V|/k$. Eine Knotenmenge S in G heißt schwach, wenn der auf S induzierte Untergraph von G keine Clique C_S der Kardinalität $|C_S| \geq |S|/(2k)$ enthält.

Der Algorithmus von Feige liefert nach Eingabe eines Graphen $G = (V, E)$, welcher eine Clique der Größe $|V|/k$ enthält, eine Clique C der Kardinalität

$$|C| \geq t \cdot \log_{3k}(|V|/t - 3) .$$

Einteilung des Algorithmus in Phasen und Iterationen:

Jede Phase arbeitet auf einem Graphen $G' = (V', E')$, welcher eine Clique der Größe $|V'|/k$ enthält.

Nach Abarbeitung einzelner Iterationen endet eine Phase, so dass eine der folgenden zwei Bedingungen erfüllt ist:

1. Eine Clique C der Kardinalität

$$|C| \geq t \cdot \log_{3k} (|V'|/(6kt))$$

wurde gefunden.

2. Ein schwacher Untergraph $G'' = (V'', E'')$ wurde gefunden.

Jede Iteration arbeitet auf einem Graphen $G'' = (V'', E'')$ und führt folgende Schritte aus:

1. Falls $|V''| < 6kt$, dann beende Phase und gib C aus.
2. Partitioniere V'' in disjunkte Knotenmengen P_i der Größe $2kt$.
3. Betrachte alle möglichen t -elementigen Teilmengen S von P_i .
4. Sei $N(S)$ alle Knoten in $V'' \setminus S$, die mit jedem Knoten aus S in G'' verbunden sind. Bezeichne S als *gut*, falls S eine Clique ist und $N(S) \geq |V''|/(2k) - t$ erfüllt.
5. Falls eine *gute* Knotenmenge S gefunden wird, dann setze $C = C \cup S$ und starte neue Iteration mit dem auf $N(S)$ induzierten Untergraphen von G'' .
6. Sonst bezeichne V'' als *schwach* und beende Phase.

- Laufzeit ist polynomiell in n , wenn $\binom{2kt}{t}$ polynomiell in n ist.
- Die Wahl des Parameters t beeinflusst neben der Laufzeit des Algorithmus auch die Größe der gefundenen Clique.

- Laufzeit ist polynomiell in n , wenn $\binom{2kt}{t}$ polynomiell in n ist.
- Die Wahl des Parameters t beeinflusst neben der Laufzeit des Algorithmus auch die Größe der gefundenen Clique.
- Um eine möglichst große Clique und eine polynomiell in n beschränkte Laufzeit zu erhalten, setzt man

$$t = \Theta \left(\frac{\log n}{\log \log n} \right) .$$

- Die gefundene Clique C hat dann die Kardinalität

$$|C| = \Omega \left(\left(\frac{\log n}{\log \log n} \right)^2 \right) .$$

Gegeben: ein Graph $G = (V, E)$ mit einer Clique C der Kardinalität

$$|C| \geq \frac{n}{k} \quad (k \text{ minimal})$$

Gegeben: ein Graph $G = (V, E)$ mit einer Clique C der Kardinalität

$$|C| \geq \frac{n}{k} \quad (k \text{ minimal})$$

Für k kommen nur folgende n Werte in Frage:

$$k \in \left\{ \frac{n}{1}, \frac{n}{2}, \dots, \frac{n}{n-1}, 1 \right\}$$

Ohne Einschränkung sei k daher bekannt.

Fall 1: $k \geq (\log n)^3$

- durch Ausgabe eines beliebigen Knotens $v \in V$ erreicht man eine Approximationsgüte von $O(n/(\log n)^3)$

Fall 1: $k \geq (\log n)^3$

- durch Ausgabe eines beliebigen Knotens $v \in V$ erreicht man eine Approximationsgüte von $O(n/(\log n)^3)$

Fall 2: $k \leq \log n / (2 \log \log n)$

- benutze Algorithmus Independent-Set-Removal von Boppana und Halldórsson
- bei einem Graphen G , welcher eine Clique der Kardinalität $2n \log \log n / \log n$ enthält, liefert dieser eine Clique der Größe $(\log n)^3 / (6 \log \log n)$
- Approximationsgüte von $O(n \log \log n / (\log n)^3)$

Fall 3: $\log n / (2 \log \log n) < k < (\log n)^3$

- der Algorithmus von Feige erreicht hier nur eine Approximationsgüte von $O(n(\log \log n)^3 / (\log n)^3)$
- für $k > \log n$ ist die Güte schon $O(n(\log \log n)^2 / (\log n)^3)$
- müssen also einen Faktor von $\Omega(\log \log n)$ einsparen

Fall 3: $\log n / (2 \log \log n) < k < (\log n)^3$

- der Algorithmus von Feige erreicht hier nur eine Approximationsgüte von $O(n(\log \log n)^3 / (\log n)^3)$
- für $k > \log n$ ist die Güte schon $O(n(\log \log n)^2 / (\log n)^3)$
- müssen also einen Faktor von $\Omega(\log \log n)$ einsparen

⇒ modifiziere Algorithmus von Feige

⇒ ändere Definition einer *guten* Knotenmenge

Modifikation:

- bezeichne eine Knotenmenge S als *gut*, falls S eine Clique ist und $N(S) > n_{test} - t$ gilt, wobei n_{test} der größte Wert ist für den gilt

$$n_{test} \leq \frac{\log n_{test}}{2 \log \log n_{test}} \cdot \frac{|V''|}{2k}$$

Modifikation:

- bezeichne eine Knotenmenge S als *gut*, falls S eine Clique ist und $N(S) > n_{test} - t$ gilt, wobei n_{test} der größte Wert ist für den gilt

$$n_{test} \leq \frac{\log n_{test}}{2 \log \log n_{test}} \cdot \frac{|V''|}{2k}$$

- falls $|V''|/(2k) - t \leq N(S) \leq n_{test} - t$ gilt, dann führe den Algorithmus Independent-Set-Removal auf $G[S \cup N(S)]$ aus
 - \implies erhält man eine Clique der Größe $(\log n_{test})^3 / (6 \log \log n_{test})$, dann fügt man diese Clique zu C hinzu und gibt C aus
 - \implies sonst ist S eine *schwache* Knotenmenge

Modifikation:

- bezeichne eine Knotenmenge S als *gut*, falls S eine Clique ist und $N(S) > n_{test} - t$ gilt, wobei n_{test} der größte Wert ist für den gilt

$$n_{test} \leq \frac{\log n_{test}}{2 \log \log n_{test}} \cdot \frac{|V''|}{2k}$$

- falls $|V''|/(2k) - t \leq N(S) \leq n_{test} - t$ gilt, dann führe den Algorithmus Independent-Set-Removal auf $G[S \cup N(S)]$ aus

⇒ erhält man eine Clique der Größe $(\log n_{test})^3 / (6 \log \log n_{test})$, dann fügt man diese Clique zu C hinzu und gibt C aus

⇒ sonst ist S eine *schwache* Knotenmenge

⇒ Approximationsgüte von $O(n(\log \log n)^2 / (\log n)^3)$

Praktische Untersuchungen

- Implementiert wurden folgende Algorithmen
 - Greedy
 - Min-Greedy
 - Ramsey
 - Clique-Removal
- Als Implementierungsumgebung wurde das Computeralgebrasystems Matlab Version 6.5 verwendet.
- Testrechner war ein AthlonXP mit 1.54 GHz und 512MB Ram.

Graph	Greedy		Min-Greedy		Ramsey		Clique-Removal	
	$ I $	Zeit in s	$ I $	Zeit in s	$ I $	Zeit in s	$ I $	Zeit in s
G_1000_10	48	0.0300	48	0.7610	48	0.7110	49	78.4820
G_1000_25	22	0.0200	23	0.3500	22	0.7210	22	57.1520
G_1000_50	8	0.0000	10	0.1800	10	0.7610	12	36.3420
G_1000_75	5	0.0000	5	0.0610	6	0.7710	7	20.1590
G_5000_10	60	0.1410	67	33.9190	60	7.3500	64	2994.2360
G_5000_25	25	0.0400	30	12.9680	26	7.8810	29	2197.5400
G_5000_50	12	0.0300	14	6.3290	14	8.5030	14	1454.9320
G_5000_75	6	0.0200	7	3.8560	8	9.2040	9	773.5720
G_10000_10	63	0.1710	67	160.9210	63	23.4130	72	16244.2980
G_10000_25	24	0.0800	32	61.2380	27	25.4570	31	12453.0160
G_10000_50	13	0.0400	12	30.1330	14	28.7520	15	8340.2130
G_10000_75	7	0.0300	7	18.2760	9	32.0060	9	4576.6510

Tabelle 1: Ergebnisse der implementierten Algorithmen

Graph	Greedy		Greedy (r+m)	Ramsey		Ramsey (r+m)
	$ I $	Zeit in s	$ I $	$ I $	Zeit in s	$ I $
G_1000_10	48	0.0300	50	48	0.7110	52
G_1000_25	22	0.0200	24	22	0.7210	23
G_1000_50	8	0.0000	11	10	0.7610	12
G_1000_75	5	0.0000	7	6	0.7710	8
G_5000_10	60	0.1410	65	60	7.3500	65
G_5000_25	25	0.0400	30	26	7.8810	30
G_5000_50	12	0.0300	14	14	8.5030	14
G_5000_75	6	0.0200	8	8	9.2040	9
G_10000_10	63	0.1710	71	63	23.4130	71
G_10000_25	24	0.0800	32	27	25.4570	30
G_10000_50	13	0.0400	17	14	28.7520	15
G_10000_75	7	0.0300	9	9	32.0060	9

Tabelle 2: Vergleich mit mehrfachgestarteten randomisierten Algorithmen

Zusammenfassung

- Eine unabhängige Menge maximaler Kardinalität in einem Graphen auf n Knoten kann mit Güte

$$O(n(\log \log n)^2 / (\log n)^3)$$

approximiert werden.

- Für praktische Anwendungen, bei denen die Laufzeit eine wesentliche Rolle spielt, ist der Einsatz des Greedy Algorithmus jedoch eine bessere Wahl.

