# Efficient Parallel Algorithms for Edge Coloring Problems

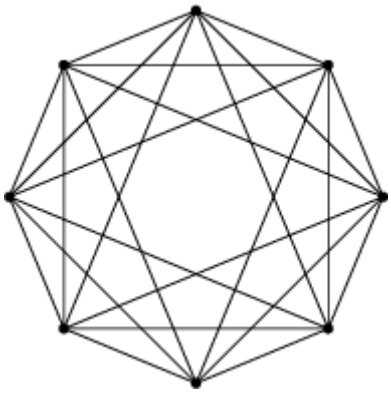
**H. Karloff, D. Shmoys**

Journal of Algorithms 8, 39-52 (1987)

## Ausarbeitung im Fach Parallele Algorithmen

## Dozent: Prof. Dr. Berrendorf

## Sommersemester 2002



**Fachhochschule Bonn-Rhein-Sieg**

**Dipl. Inform. Elias Delipetkos**
**Dipl. Inform. Peter Schöll**

# 1. Einführung

Die fortgeschrittene Technologie der Parallelrechner gilt als Grundvoraussetzung für eine neue Herangehensweise bei der Konstruktion von Algorithmen zur effizienten Lösung der Kanteneinfärbung bei Graphen, wie Howard J. Karloff und David B. Symoys in Ihrem Artikel „Efficient Parallel Algorithms for Edge Coloring Problems“, zeigen.

Da bisher teilweise sequentielle Techniken benutz wurden, welche bei der parallelen Verarbeitung an Ihre Grenzen stoßen oder gar nicht eingesetzt werden können, müssen neue Lösungen gefunden werden. Weiterhin zeigt sich, das es auch Probleme gibt, für die keine effizienten parallelen Algorithmen existieren.

In den abgehandelten Paper werden verschiedene Probleme des Einfärbens von Kanten bei Graphen besprochen. Allen gemeinsam ist, dass sie effizient von einfachen Algorithmen gelöst werden können. Da alle diese einfachen Algorithmen zur Lösung Verfahren benutzen, bei denen es sich anbietet auch sequentiell gelöst zu werden, sind auch sie erst mal nur rein sequentiell.

Karloff und Symoys zeigen neue Verfahren für die effiziente parallele Lösung auf.

Im Besonderen beschränken sie sich auf Algorithmen mit NC (siehe 1.1).

Weitere Einschränkung der Algorithmen sind folgende:

| | | |
|---|---|---|
| Benötigte Zeit: | $n^{\log(k)}$ | ; k >= 2 |
| Anzahl Prozessoren: | $n^{k}$ | ; k >= 2 |
| Architektur: | CRCW PRAM | |
| | (Gleichzeitiges Schreiben nur bei identischen Werten) | |

Diese angesprochenen Algorithmen sollen zum Einfärben von Kanten bei Graphen zum Einsatz kommen. Hierfür wird folgende Notation eingeführt:

Die kleinste Anzahl der benötigen verschiedenen Farben - der chromatische Index eines Graphen (G) wird als:

→ X`(G) bezeichnet.

Wenn $\triangle$ (G) der maximale Grad eines Graphen ist, ergibt sich daraus:

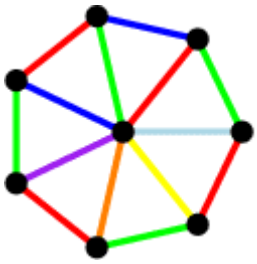
→ $X`(G) >= \triangle (G)$

## *1.1. Definition NC-Algorithmus*

Analog zur Theorie der NP-Vollständigen Probleme, wurde die Theorie der parallelen Komplexität von Berechenbarkeitsproblemen entwickelt. Eine Problem welches zur Klasse der *NC* Klassen (Nick´s Class) gehört, kann in polylogarithmischer Zeit mit höchstens einer polynominalen Anzahl von Prozessoren gelöst werden. Die Klasse NC in der parallelen Komplexitätstheorie spielt die gleiche Rolle wie die Rolle der P-Klasse in der sequentiellen Theorie.

*1.2. Beispiel eines Graphen mit eingefärbten Kanten:*

Graph G:



Maximaler Grad des Graphen: 7

➔ Knoten in der Mitte hat die höchste Anzahl Kanten

Benutzte Anzahl Farben: 7

➔ rot, grün, blau, violett, orange, gelb, hellblau

*1.3. Vizings Beweis:*

Vizing (1964) und Gupta (1966) haben bewiesen, dass die Kanten eines jeden Graphen mit △ (G) + 1 Farben eingefärbt werden können.

Die Autoren Karloff und Symoys stellen weiterhin folgende Behauptung auf:

➔ X`(G) <= △ (G) + 1 gilt für jeden einfachen Graphen !

Das heißt die Anzahl der benötigten Farben ist kleiner oder gleich des maximale Grad eines Graphen + 1 für jeden Einfachen Graphen.

*1.3.1. Definition eines Einfachen Graphen:*

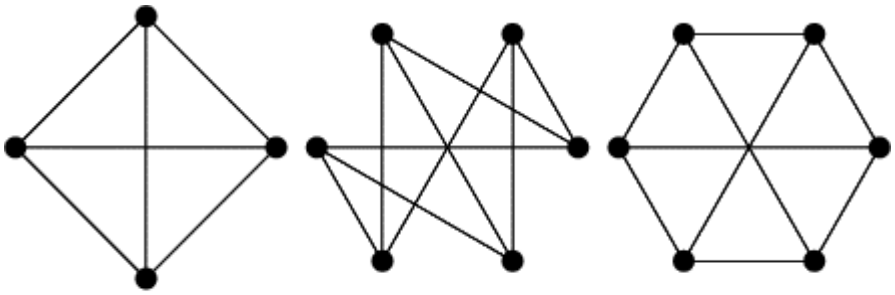
➔ Graphen bei denen höchstens *eine* Kante zwei Knoten miteinander verbindet. Graphen mit Schleifen gelten nicht zu den Einfachen Graphen.

Der Beweis von Vizing kann dazu verwendet werden um einen Algorithmus mit folgenden Eigenschaften zu entwerfen:

> ➔ Ein Graph G = (V, E) kann mit maximal $\triangle(G) + 1$ Farben in O ( |V| |E| ) Zeit eingefärbt werden.

Holyer (1981; Skiena 1990) hat gezeigt, dass das die Ermittlung des chromatischen Index eines Graphen als NP-vollständig gilt – selbst wenn man nur Kubische Graphen betrachtet.

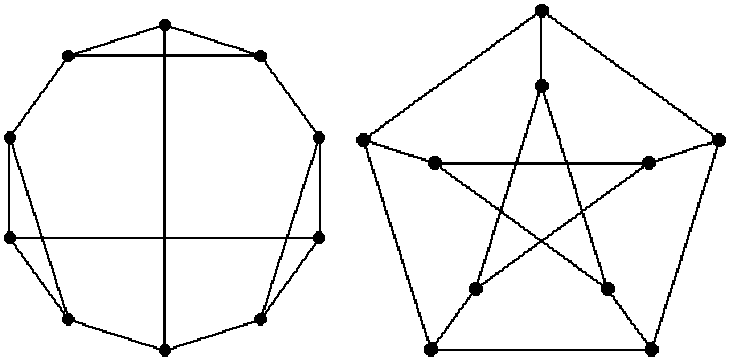
### *1.3.2. Definition Kubische Graphen*



$K_4$ $K_{3,3}$ $Ci_{1,3}(6)$

Kubische Graphen sind Graphen, bei denen alle Knoten einen Grad von drei haben. Kubische Graphen mit *n* Knoten gibt's es nur bei einer graden Anzahl von *n* (Harary 1994, p. 15).

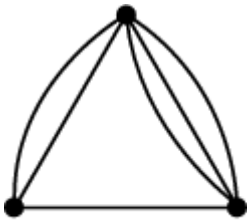
Beispiel zweier verbundenen kubischen Graphen mit $\triangle (G) = 3$ auch 3-regular genannt mit maximaler Knotenanzahl = 10. Von dieser Art Graphen gibt es 19 verschiedene. Bei einer Knotenanzahl von 26 hätten wir schon 2 094 480 864 verschiedene Graphen.



## *1.4. Multigraphen*

Für Multigraphen (siehe 1.4.1) werden mehr Farben benötigt als für einfache Graphen. Dies zeigt sich an folgenden Beispiel deutlich:

### *1.4.1. Definition eines Multigraphen:*



*multigraph*

Nicht-Einfache Graphen ohne Schleifen bei denen aber mehreren Kanten zwischen zwei Knoten vorkommen können.

Beispiel:

Graph G mit 3 Knoten und k Kanten zwischen jedem Knotenpaar:

→ $\triangle (G) = 2 k$
→ $X`(G) = 3 k$ im schlechtesten Fall

Shannon hat 1949 gezeigt:

Für einen beliebigen Multigraphen G gilt:

→ $X`(G) <= 3 \triangle (G) / 2$

Für obiges Beispiel wären das:
$3 k <= ( 3 * 2 k ) / 2$
$3 k <= 3 k$

Übersichtstabelle der verschiedenen Verfahren zur parallelen Einfärbung von Kanten bezogen auf die Effizienz und benutzen Farben.

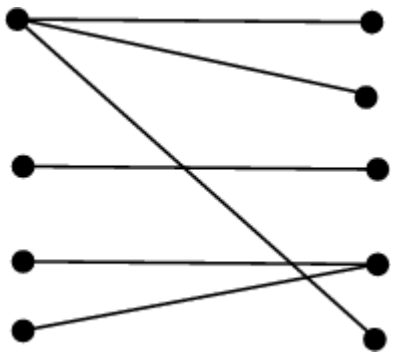
Notation : $\triangle = \triangle(G)$

| Graphen Klasse | Anzahl Farben | Zeit | Prozessoren |
|---|---|---|---|
| Multigraphen | $3\,[\triangle / 2\,]$ | $O(\log^3(\triangle \lvert V\rvert))$ | $O(\triangle \lvert V\rvert)$ |
| Multigraphen mit $\triangle(G) = 3$ | 4 | $O(\log \lvert V\rvert)$ | $O(\lvert V\rvert + \lvert E\rvert)$ |
| Einfach Graph | $\triangle + 1$ | $O(\triangle^{O(1)} \log^{O(1)} \lvert V\rvert)$ | $O(\lvert V\rvert^{O(1)})$ |
| Einfach Graph | $\triangle + 20 * \triangle^{1/2+\varepsilon}$ | $O(\log^{O(1)} \lvert V\rvert)$ | $O(\lvert V\rvert^{O(1)})$ |

## *1.5. Zweiteilige Graphen*

### *1.5.1. Definition von Zweiteiligen Graphen*

Ein Menge von Knoten welche in zwei disjunkte Mengen unterteilt ist, sodass keine zwei Knoten innerhalb der gleiche Untermenge benachbart sind – also über eine Kante miteinander verbunden (s. Beispiel unten). Ein zweiteiliger Graph ist ein Sonderfall eines k-geteilten Graphen mit $k = 2$.



Für zweiteilige Graphen gilt:

→ $X'(G) = \triangle(G)$

Dadurch ist es relativ einfach einen optimalen Algorithmus für das Einfärben von Kanten bei zweiteiligen Graphen zu entwickeln. Die Aufmerksamkeit wird hierfür auf das Auffinden eines Eulereschen Bereich gerichtet.

### *1.5.2. Definition Euleresher Weg (Eulerian Trail)*

Ein Weg über die Kanten des Graphen, wobei jede Kanten genau einmal benutzt werden darf. Ein verbundener Graph (siehe 2.1.1) hat ein Eulereschen Weg genau dann, und nur dann, wenn der Grad jedes Knotens gerade ist. Diese Graphen werden auch Eulersche Graphen genannt.

## *1.6. Algorithmus zum Teilen eines Graphen*

- ➔ Graph *G* mit △ *(G)* geradzahlig
- ➔ *G* zu einem Eulereschen Graphen $G^+$ erweitern
  *Die geschieht durch das Zufügen eines Knotens, welcher mit allen Knoten die einen ungeraden Grad haben, verbunden wird.*
- ➔ Eine Euleresche Tour (siehe 2.4.1) in $G^+$ finden
- ➔ Einen beliebigen Knoten *υ* aussuchen
- ➔ $e_i$ soll die i-te Kante des durchquerten Pfades, vom Startknoten *υ* aus, sein

Die Kanten des Graphen *G* können nun sehr leicht in zwei Hälften geteilt werden, abhängig davon ob *i* ungerade oder gerade ist.

Dadurch werden zwei Graphen mit dem halben maximalen Grad (△ (G) ) des Ursprungs-graphen erzeugt.

Durch die rekursive Anwendung dieses Algorithmussees können die Kanten zweiteiliger Graphen mit $\triangle (G) = 2^k$ durch △ *(G)* Farben eingefärbt werden.

Es gibt verschiedene Ansätze für den Fall das der maximale Grad eines Graphen ungerade ist.

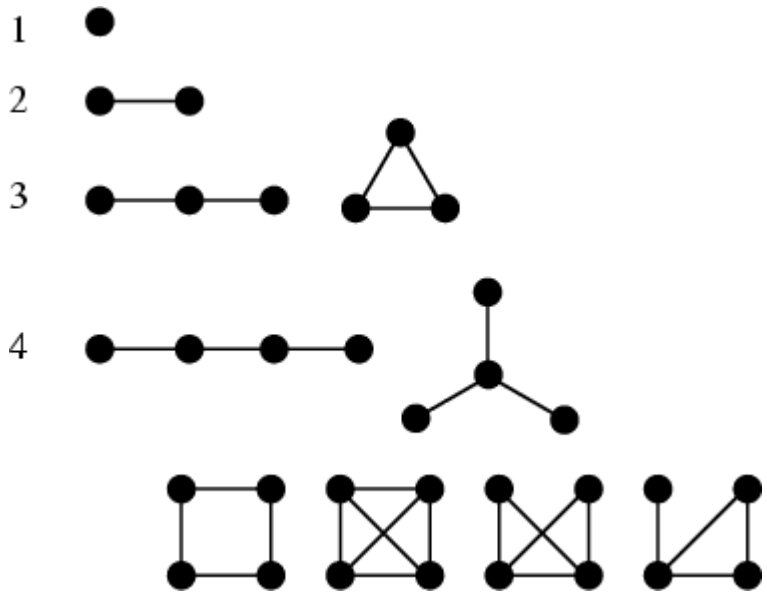
In (Lev, Pippenger, Valiant, 1981) wurde gezeigt, dass i.A. zweiteilige Multigraphen in $O ( log^3 (\triangle |V| ))$ Zeit und mit $O (\triangle |V| )$ Prozessoren optimal eingefärbt werden können.

# 2. Multigraphen einfärben

## *2.1. Vorverarbeitung mit einem "Connected component algorithm"*

Zuerst werden die verbundenen Komponenten des Multigraphen berechnet. Y. Shiloach und U. Vishkin haben 1980 einen entsprechenden Algorithmus vorgestellt. Dieser Algorithmus benötigt O ( log n ) Zeit.

### *2.1.1. Definition Verbundener Graph*




Ein Graph ist verbunden, wenn von jedem Knoten eine Weg über verschiedene Kanten zu einem beliebigen Knoten innerhalb des Graphs existiert.

*2.2. Konstruktion eines parallelen Algorithmus unter Verwendung von 2.1*

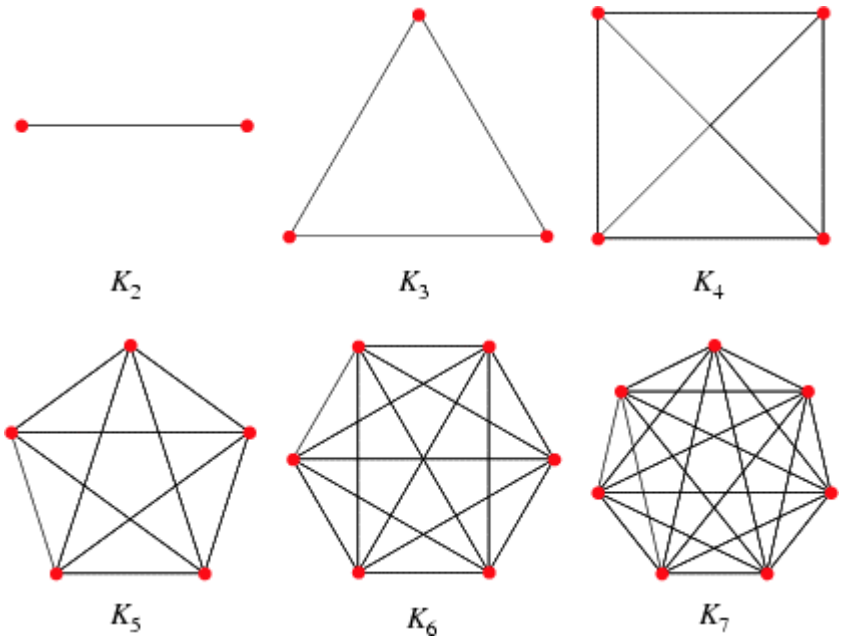
Ein beliebiger Multigraph G wird durch folgenden Algorithmus mit 3 * [△ (G) / 2 ] Farben eingefärbt.

*2.3. Die einzelnen Schritte:*

- ➔ Zerlegung der Kanten des Graphen G in [△ / 2 ] verschiedene Graphen, welche allen einen maximalen Grad von 2 haben.
- ➔ Da jeder dieser Graphen mit 3 verschiedenen Farben eingefärbt werden kann, kann der komplette (s. 2.3.1) Multigraph mit 3 [△/ 2 ] Farben eingefärbt werden.

*2.3.1. Definition Kompletter Graph*

Ein Graph ist komplett, wenn jeder Knoten über eine Kante mit jedem anderen Knoten verbunden ist.



*2.4. Der Algorithmus im Detail:*

Gegeben sein ein *verbundener* Multigraph Graph *G*.

**Schritt (1):**

Einen zweiteiligen Multigraphen erzeugen $G_B = (X, Y, E_B)$:

➔ Wenn *G* kein Eulerscher Graph ist, einen weiteren Knoten $v$ zufügen, sodass dieser Knoten mit allen Knoten verbunden ist die einen ungeraden Knotengrad haben. (*Dadurch wird der Graph zu einem Eulereschen Graph.)*

*(Ein ungerichteter Graph ist ein Eulerscher Graph genau dann, wenn jeder Knoten einen geraden Grad hat)*

➔ Eine Euleresche Tour (siehe 2.4.1) in diesem Graphen suchen.

➔ Für jeden Knoten $v$ im originalen Graphen gibt es einen zugehörigen Knoten $u' \in X$ und $\acute{v} \in Y$

➔ Für jede Kante $\{ u, v \}$ wird eine ungerichtete Kante $\{ u', \acute{v} \}$ zu $E_B$ hinzugefügt.

**Schritt (2):**

➔ Eine optimale Kanteneinfärbung für $G_B$ finden.

**Schritt (3):**

➔ $M_i$ seien die Kanten, eingefärbt mit der Farbe $c_i$ . Für jede Farbe wird nun wiederum ein Multigraphen $G_i$ wie folgt konstruiert:

Für jede Kante $\{ u', \acute{v} \} \in M_i$ , kommt eine Kante zu $E_i$ .

Da $M_i$ immer eine Paarung ist,

hat der zugehörige Multigraph $G_i$ einen Grad $\triangle (G_i) <= 2$

dass heißt:

$G_i$ besteht nur aus Pfaden und Zyklen.

**Schritt (4):**

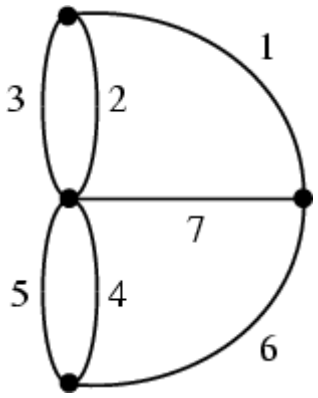
➔ Jeder Graph $G_i$ wird nun mit 3 Farben eingefärbt.

### *2.4.1. Definition Eulerscher Tour (Eulerian tour, circuit)*

Ein Eulerscher Weg (siehe 1.5.2, Eulerscher. Weg, Eulerian Trail) , welcher als Start- und Endknoten den gleichen Knoten benutzt. D.h. ein Zyklus in einem Graphen welcher jede Kante nur genau einmal durchläuft und Start- und Endknoten gleich sind.

Als Verallgemeinerung des Königsberger Brückenproblems (siehe 2.4.2), zeigte Euler, dass ein verbundener Graph ein Eulerscher Tour hat, genau dann, wenn er keine Knoten mit ungeraden Grad hat.

### *2.4.2. Königsberger Brücken Problem*




Das Königsberger Brücken Problem beschreibt das Problem, dass die sieben Brücken der Stadt Königsberg alle in einer einzigen Tour abgefahren werden sollen, ohne eine Brücke zweimal zu benutzen.

Dieses Problem ist gleichbedeutend mit der Suche nach einer Eulereschen Tour in einem Multigraphen (s. links) mit vier Knoten und sieben Kanten (Brücken). Euler zeigte 1736 das dies nicht möglich ist, und leitete damit die Graphen Theorie ein.

## *2.5. Zeitbedarf des Algorithmussees*

Jeder Schritt benötigt $O( \log^3 ( |V| ))$ Zeit bei $O (\triangle |V| )$ Prozessoren.

## *2.6. Theorem:*

Die Kanten eines jeden beliebigen Multigraphen G können mit

- → $3 [\triangle / 2 ]$ Farben unter Verwendung von
- → $O (\triangle |V| )$ Prozessoren mit dem Zeitbedarf von
- → $O ( \log^3 ( |V| ))$ eingefärbt werden.

## 3. Multigraphen maximal 3. Grades einfärben

Ein verbundener Multigraph G = *(V , E )* mit maximalen Grad 3, wird durch folgenden Algorithmus mit vier Farben eingefärbt.

*3.1. Die einzelnen Schritte:*

- 1.) Ein weiterer Knoten $v_0$ wird zu $G$ zugefügt, sodass er als benachbarte Knoten nur Knoten mit ungeraden Grad hat.

- 2.) Eine Euleresche Tour in diesem neuen Graphen suchen, wobei der Start- und Endpunkt in $v_0$ liegt. Die durchgelaufenen Kanten auf dieser Tour werden vom Startpunkt aus folgendermaßen durchnummeriert: $e_1, e_2, e_3, ...$

- 3.) Zwei neue Multigraphen $G_1 = (V , E_1 )$ und $G_2 = (V , E_2 )$ erzeugen, wobei $E_1$ die ungerade nummerierten Kanten aus $E$ beinhaltet, und $E_2$ die gerade nummerierten Kanten aus $E$.

- 4.) Alle Zyklen mit gerader Länge und alle Touren in $G_1$ werden mit den Farben $a$ und $b$ eingefärbt. Für $G_2$ gilt das gleiche, wobei die Farbe $c$ und $d$ verwendet werden.

- 5.) Für jeden Zyklus mit ungerader Länge in $G_1$ , werden zwei Knoten x und y gesucht, welche aufeinander folgend im Zyklus liegen, sodass, bei beiden die gesuchte Farben $c$ oder die Farbe $d$ ist. Die Kante $\{ x, y \}$ wird mit der gemeinsam gesuchten Farbe eingefärbt. Die übrigen uneingefärbten Kanten bilden eine Tour und werden mit den Farben $a$ und $b$ eingefärbt. Das gleiche wird mit dem Graphen $G_2$ gemacht.

*3.2. Erläuterung zum Algorithmus*

Die Grundlage des Algorithmussees liegt im letzten 5. Schritt. Es wird immer davon ausgegangen, dass immer ein Knotenpaar $\{ x, y \}$ gefunden wird, und das weder $x$ noch $y$ mit der gleichen Farben eingefärbt werden.

Für jeden Knoten $v$ in einem Zyklus in $G_1$ wissen wir, dass der Grad von $v$ in $G_2$ höchsten 1 ist.

Daraus resultierend, wird jede Kante in $G_2$ welche zugehörend zu einem Knoten eines ungeraden Zyklus in $G_1$ ist, noch vor Schritt 5 eingefärbt.

Weiterhin wird bis dahin mindestens $c$ oder $d$ noch gesucht.

Da der Zyklus eine ungerade Anzahl von Knoten hat, wird mindestens eine Farbe an zwei aufeinander folgende Knoten des Zyklussees gesucht.

*3.3. Theorem:*

Jeder Multigraph G mit $\triangle$ (G) = 3 kann mit

→ 4 Farben unter Verwendung von

→ O ( |E| + |V| ) Prozessoren mit dem Zeitbedarf von

→ O ( log |V| ) eingefärbt werden.

# 4. Einfache Graphen mit $\Delta+1$ Farben einfärben

## *4.1. Einleitung*

Im den nachfolgenden Kapitel werden Algorithmen und Verfahren vorgestellt, die ausschließlich einfache Graphen behandeln. Basierend auf Vizing´s Beweis kann grundsätzlich zwischen zwei unterschiedlichen Verfahren zur Kanteneinfärbung von Graphen unterschieden werden.

1. Chain – recoloring (Ketten-Wiedereinfärbungsverfahren)
2. Fan – recoloring (Fächerartiges - Wiedereinfärbungsverfahren)

Bevor in die Thematik der zwei *„Recoloring-Verfahren“* eingegangen wird, ist es notewenig einige Definitionen und Erläuterungen anzuführen.

Eine „*unabhängige Menge“* ist eine Knotenmenge, in der es keine Kante gibt, die zwischen zwei Knoten dieser Knotenmenge liegt.

Eine „*maximal unabhängige Menge“* ist eine unabhängige Menge in die kein weitere Knoten hinzugefügt werden kann, ohne das eine neue Kanten hinzugefügt wird. Für den Fall, dass es einen Knoten in einer Kantenmenge eines Graphen $G$ gibt, der höchsten ein mal vorkommt, so wird dies auch als „*matching“* bezeichnet.

Ein Knoten wird als „*frei“* bezeichnet, wenn dieser Knoten an keiner Kante aus der Kantenmenge existiert, welche sich aus dem „*matching“* ergibt. Ein „*maximales matching“* $M$ ist ein „*matching“*, wobei es keine Kante zwischen zwei „*freien“* Knoten gibt. Ein maximaler Pfad, beginnend am Knoten $r$ ist ein einfacher Pfad $P$, wobei $P$ nicht beliebig erweitert werden kann, ohne dass auf einen Knoten getroffen wird, der bereits auf dem Pfad liegt.

## *4.2. Chain-Recoloring*

Zur Wiederholung: Vizing´s Theorem besagt, dass es möglich ist, einen einfachen Graphen $G(V,E)$ mit $\Delta+1$ Farbe einzufärben.

Es sei $G$ ein *„eged-colored“* Graph. Eine Farbe $a$ wird als *incident* zu einem Knoten $v$ bezeichnet, falls eine Kante mit der Farbe $a$ existiert, die *incident* zum Knoten $v$ ist. Andernfalls ist kann die Farbe $a$ am Knoten $v$ als „*frei“* bezeichnet werden. Solange $\Delta+1$ Farben vorhanden sind und der Grad eines jeden Knotens höchsten $\Delta$ entspricht, so existiert für jeden Knoten eine freie Farbe.

Für den Fall, dass in einem Graphen $G$, Kanten existieren, die eingefärbt sind und gleichzeitig Kanten existieren, die ungefärbt sind, so wird dieser Graph $G$ partiell oder teilweise eingefärbt genannt. Wird ein Graph mit zwei unterschiedlichen Farben $a$ und $b$ eingefärbt, so entsteht hieraus ein neuer Graph $G$. Dieser durch die Einfärbung neu entstandener Graph, wird auch als ein Subgraph bezeichnet. Dieser Subgraph muss folglich aus unabhängigen Pfaden und einfach Zyklen bestehen. Durch gegenseitiges Vertauschen der Farben $a$ und $b$, im Laufe eines Pfades oder Zyklus, entsteht ein neuer Graph, welcher wiederum partiell eingefärbt ist.

Angenommen es existiert ein Pfad von $u$ nach $v$, wobei die Farbe $a$ an der Kante $u$ und die Farbe $b$ an der Kante $v$ frei sind. Für den Fall, dass eine ungefärbte Kante $(w,u)$ im Graphen $G$ existiert, so dass die Farbe $b$ an der Kante $w$ frei und die Farbe $a$ bereits vergeben (nicht mehr frei) ist. Nachdem Vertauschen der Pfadfarben werden die Kanten $w$ und $u$ gemeinsam die freie Farbe $b$ verfügen. Somit ist nun möglich, die Kante $(w,u)$ mit der Farbe $b$ einzufärben. Diese Verfahrensweise wird als ***„Chain-Recoloring“*** bezeichnet.

*4.3. Fan-Recoloring und Fan-Sequence*

In der Vergangenheit wurden einige Definitionen für *„Fan-Sequences“* erstellt. Im Folgenden soll eine Definition erläutert werden, welche auf eine Definition von Misra und Gries[15] basiert. Als Beispiel sei ein Knoten $u$ gegeben, welcher als das Zentrum einer sog. *„Fan-Sequence“* definiert ist. Ein *„Fan“* $\{v..w\}$ von $u$ ist eine *geordnete* Knotenreihe, welche die nachfolgend beschriebenen Bedingungen erfüllt.

1. Der *„Fan“* $\{v..w\}$ ist eine nichtleere Reihe von angrenzenden Knoten von $u$, welche sich zum Knoten $u$ eindeutig unterscheiden.

2. Die Kante $(u,v)$ ist ungefärbt, und die Kante $(u,s)$ ist gefärbt, für den Fall, dass sich die Knoten $s$ und $v$ eindeutig voneinander unterscheiden.

3. Für den Fall, dass in der *„Fan-Sequence“* dem Knoten $s$ ein Knoten $t$ mit $s \le t$ folgt, dann ist die Farbe an der Kante $(u,t)$ für den Knoten $s$ *„frei“*.

Der *„Fan-recoloring“* - Algorithmus kann nun wie folgt zusammengefasst werden:

Schritt 1: Die Farben entlang des $ab$-Pfades von $u$, sind so zu Vertauschen, dass die Farbe $b$ für den Knoten $u$ *„frei“* wird.

Schritt 2: Es existiert ein Knoten $s$, welcher folgende Bedingungen erfüllt:

Der Konten $s \in \{v..w\}, \{v..s\}$ ist ein *„Fan“* des Knoten $u$ und die Farbe $b$ ist am Knoten $s$ *„frei“*. Durch erneutes Zuweisen einer neuen Kantenfarbe (Farbe $b$) für die Kante $(u,s)$, ändert sich folgedessen auch der *„Fan“* $\{v..s\}$ des Knoten $u$.

Um den Beweis für *„Vizing´s Theorem“* zu vervollständigen, ist es ausreichend festzuhalten, dass einer von zwei Knoten $v_i$ und $v_j$ einen farblich alternierenden Pfad mit dem Farben $c_0,...,c_i$ besitzt, der jedoch nicht im Knoten $v_0$ endet. Durch ständiges, zyklisches Wechseln (gegenseitiges Austauschen) der Kantenfarben besteht nun die Möglichkeit eine sog. *„fan-operation“* durchzuführen, wodurch das Einfärben optimiert (erweitert) wird. Nach dem nun gezeigt wurde, dass eine nicht eingefärbte Kante nach einigen *„recoloring“* - Durchläufen eingefärbt werden kann, ist Vizing´s Theorem nun bestätigt und bewiesen.

Es sei darauf hingewiesen, dass die Durchführen einer *„fan-operation“* recht begrenzt in ihrer Komplexität ist und somit sehr lokal. Dieser Vorteil, kommt erst in einer parallelen Anwendung zur Geltung.

Der Begriff der Lokalität soll nun im einzelnen etwas genauer Betrachtet werden.

Aussage 1: Wird ein teilweise eingefärbter Teil eines Graphen $G$ geändert, so dass die Menge der gesuchten Farben der Knoten $v_i \ mit \ i = 0,...,k$, sowie die Einfärbung des Pfades $\{v_0, v_i\} \ mit \ i = 1,...,k$, unverändert bleibt, so ergibt sich die bereits bekannte *„fan-sequence“* mit $(v_0, c_0, v_1, c_1, v_2, c_2, ..., v_t, c_t)$.

Aussage 2: Wird eine *„fan-operation”* mit der *„fan-sequence”* $(v_0, c_0, v_1, c_1, v_2, c_2, ..., v_t, c_t)$ ausgeführt, so bleibt die Menge der gesuchten Farbe eines Knotens $v$ ungleich $v_i \ mit \ i = 0,...,k$ unverändert. Ebenfalls unverändert bleibt die an den Konten $v$ angrenzenden Kanten.

Um eine parallele Implementierung verschieden ungefärbter Kanten vorzunehmen, ist es notwendig eine Menge von *„fans“* zu finden, die im gesamten Graphen $G$ *„dispersed“* verteilt unabhängig voneinander sind. Nach Ausssage 1 und Aussage 2, kann die *„fan-operation“* auf jeden einzelnen dieser *„fans“* angewandt werden. Hierbei werden die jeweils anderen *„fans“* der Menge nicht beeinträchtigt.

Daraus folgt, dass diese Operation parallel ausgeführt werden kann. Die Anforderung das *„chain-recoloring“* - Verfahren ebenfalls anzuwenden, wirkt der gewünschten Parallelität jedoch entgegen.

Es sei darauf hingewiesen, dass einige Kanten in beiden *„chains“* mit einer und der selben Farbe, beispielsweise mit der Farbe $b$ vorkommen können. Es stellt sich nun die Berechtigte Frage, was soll denn nun *„recoloring“* bewirken? Mit dem Ziel, diese Problematik zu umgehen, wird im Folgenden nur noch eine Untermenge der *„fans“* betrachtet, wobei für die Wiedereinfärbung der *„chains“* mit Hilfe von zwei Farben realisiert wird.

Die Auswahl der *„fans“* wird durch Generierung und Verwendung eines Hilfsgraphen erreicht. Dieser Hilfsgraph stellt die Abhängigkeiten der Operationen zwischen den jeweiligen *„fans“* dar.

Im Folgenden besteht die Herausforderung der Auffindung einer unabhängigen Mengen von *„fans“* in dem generierten Hilfsgraphen. Ist eine solche Mengen gefunden worden, wobei es sich nun um eine Untermenge der *„fans“* handelt, so besteht nun die Sicherheit, dass die Operationen unabhängig von den jeweiligen *„fans“* durchgeführt werden können. Die Möglichkeit, die ausgewählte *„fan“* - Untermenge parallel zu bearbeiten ist ebenfalls gegeben. Handelt es sich bei der ausgewählten Menge um eine maximal parallel unabhängige Menge, so ist auch hier garantiert, dass die ausgewählte Menge über eine maximale Unabhängigkeit verfügt. Unter Verwendung des *NC* Algorithmus [KW, Lu] können Probleme zur Auffindung von maximalen unabhängigen Mengen gelöst werden.

In Folgenden soll der $\Delta+1$ Kantenfärbungs-Algorithmus erläutert werden. Zuvor jedoch noch erst ein paar Definitionen und Bezeichnungen. Falls ein Graph $G$ teilweise eingefärbt ist, dann bezeichne einen Knoten v als aktiv falls es eine angrenzende Kante zu dem Knoten v gibt, die nicht eingefärbt ist. Falls $G=(V,E)$ ein ungerichteter Graph ist, dann ist das Quadrat des Graphen $G: G^2=(V,E^2)$, und für das Quadrat der Kanten gilt:

$$E^2 = E \cup (\{x,z\} \mid \exists y \{x,y\}.\{y,z\} \in E\,.$$

Der $\Delta+1$ Edge-Color Algorithmus verfügt typischerweise über $\mathrm{O}\left(\Delta^5 \log|V|\right)$ Phasen. Darüber hinaus sei $A$ als die Menge einer aktive Knotenmenge definiert. Solange jeder aktive Knoten über maximal $\Delta$ ungefärbte Kanten verfügt, so wird in jeder Phase eine Teilmenge der Kanten, beschrieben durch

$$\Omega\left(\frac{1}{\Delta^5}\right)$$

eingefärbt. Daraus folgt, dass zur vollständigen Einfärbung maximal

$$O\left(\Delta^5 \cdot \log|V|\right)$$

Phasen benötigt werden. Deshalb ist es ausreichend, mit einem Algorithmus zu beweisen, dass für die Durchführung einer solchen Phase in einer Zeit von

$$O\left(\Delta\, O(1) \cdot \log O(1) \cdot |A|\right)$$

mit

$$O(|V| \cdot O(1))$$

Prozessoren definiert werden kann.

### *4.4. Der $\Delta+1$ Phasen Algorithmus*

Schritt 1: Finde eine maximale unabhängige Menge $I$ in $(G^2)_A$ unter Verwendung des Algorithmus in [Lu]. Mit Hilfe der maximal unabhängigen Menge $I$, kann eine Menge von *„fans“* erstellt werden, die parallel wiedereingefärbt werden kann.

Schritt 2: Wähle (parallel) für jeden Knoten $v \in I$ eine ungefärbte Kante $e(v)$ aus, die an den Knoten $v$ angrenzt. Bezeichne die daraus resultierende Menge als Menge $M$.

Schritt 3: Wähle (parallel) für jede Kante $\{v_0, v_1\}$ in $M$, wobei $v_0 \in M$ ist, eine gesuchten Farbe $c_0$ am Knoten $v_0$. Generiere (sequentiell) eine maximale *„fan-sequence“* mit

$$(v_0, c_0, v_1, c_1, v_2, c_2, ..., v_t, c_t)$$

aus der, falls möglich, eine gesuchte Farbe $c_i$ ausgewählt werden kann, die zur Einfärbung am Knoten $v_0$ verwendet werden kann. Falls nach $v_0$ ein Knoten $v_i$ folgt, welcher sich eine gesuchte Farbe mit dem Knoten $v_0$ teilt so endet die *„fan-sequence“* an dieser Stelle.

Für den Fall, dass für den Knoten $v_0$ die gesuchte Farbe $c_t$ ist, so ist die Farbzuweisung $c_0 \leftarrow c_t$ durchzuführen. Nachfolgend kann nach bekannten Muster die *„fan-operation“* mit der ebenfalls bekannten *„fan-sequence“*

$$(v_0, c_0, v_1, c_1, v_2, c_2, ..., v_t, c_t)$$

ausgeführt werden.

Für den Fall, dass $c_r = c_t$ mit $r < t$, so dass $c_t$ die gesuchte Farbe an einem Knoten $v_i \neq v_t$ innerhalb der *„fan-sequence“* ist, so wird der *„fan“* beendet. In diesem Fall wird die Gewissheit benötigt, dass der einzige Knoten des *„fans“* die gesuchte Farbe $c_0$ ist. Der Knoten $v_0$ selbst sowie zwei weitere Knoten benötigen die Farbe $c_t$. Es sei $v_r \neq v_t$. Hierbei ist $v_r$ ein Knoten der sich zu einem Knoten $v_t$ eindeutig unterscheidet, jedoch mit der Eigenschaft $c_r = c_t$. Überprüfe nun alle Knoten der *„fan-sequence“* in der Knotenreihenfolge: $v_1, v_2, v_3, ..., v_r$ auf der Suche nach einem Knoten $v_p$, welcher die Farbe $c_t$ benötigt. Überprüfe anschließend die Knoten $v_{r+2}, v_{r+3}, v_{r+4}, ..., v_r$ mit dem Ziel einen Knoten anzutreffen, der die Farbe $c_t$ benötigt. Hieraus ergibt sich im Folgenden eine modifizierte *„fan-sequence“*:

$$v_0, c_0, v_1, c_1, \ldots, v_{p-1}, c_{p-1}, v_p, c_p, v_{r+1}, c_{r+1}, \ldots, v_{q-1}, c_{q-1}, v_q, c_q$$

Es ist offensichtlich, dass es sich im Fall des Knoten $v_0$ um den einzige Knoten handelt, der die Farbe $c_0$ benötigt. Analog benötigen die Knoten $v_p$ und $v_q$ als einzige Knoten die Farbe $c_t$.

Die Reihenfolge der gesuchten Farbe für den o.g. *„fan“* beginnen mit $c_0$ und enden mit der Farbe $c_t$, welche an genau zwei Knoten der *„fan-sequence“* gesucht bzw. verwendet wird. Für die verwenden Farben ergibt sich hieraus folgende mathematische Definition:

$$a(v_0) = c_0$$

sowie

$$b(v_0) = c_t .$$

Darüber hinaus seien die zwei Knoten in der *„fan-sequence“* $b(v_0)$ als

$$x(v_0) = v_p$$

und

$$y(v_0) = v_q$$

definiert. $I*$ sei die Menge aller Knoten aus der Knotenmenge $I$, auf die noch keine *„fan-operation“* angewandt wurde.

Schritt 4: Für jeden Knoten $v \in I*$ wurde ein *„fan“* $(a(v), b(v))$ definiert. Wähle nun, das am häufig auftretende Farbenpaar $(a, b)$ aus und definiere $I_{ab}$ für die entsprechenden Konten aus der Knotenmenge $I*$. Der Graph $G[a, b]$ sein ein Subgraph des Graphen $G$, welcher aus der Kantenfärbung mit den Farben $a, b$ entstanden ist. Unter Verwendung dieser *„fan“* - Menge, ist es sicher, dass das *„chain-recoloring“* parallel durchgeführt werden kann.

Schritt 5: Partitioniere die Knotenmenge $I_{ab}$ in zwei neue Mengen $I_1$ und $I_2$. Verschieb den Knoten $v$ in die neune Kontenmenge $I_1$, falls kein Pfad im Graphen $G[a, b]$ existiert, der vom Konten $v$ bis zu $x(v)$ führt. Wiederhole dies $v$ sucht die Farbe $a$, jedoch nicht die Farbe $b$, wobei $x(v)$ die Farbe $b$, jedoch nicht die Farbe $a$ sucht.

Schritt 6: Konstruiere einen neuen Graphen

$$G_{fan} = (I_1, E_{fan})$$

wie folgt: Für jeden Pfad, der als eine Komponente aus $G[a,b]$ angesehen wird und mindestens mit einem Punkt in $x(v)$ endet, wobei $v \in I_1$ ist und der andere Endpunkt entweder gleich $u$ oder gleich $x(u)$ für einen beliebigen Knoten $u \in I_1$ mit der Kante $\{u,v\} \in I_{fan}$ ist. Es sei darauf hingewiesen, dass $\Delta(G_{fan}) \leq 2$ ist. Suche als nächstes aus der Menge $G_{fan}$ eine maximal unabhängige Menge $I_{fan}$. Vertausche im Pfad $G[a,b]$ und dem Pfadende, gekennzeichnet durch den Endpunkt $x(v)$, für jedes $v \in I_{fan}$ die Farbe. Solange $x(v)$ und $v$ gemeinsam die Farbe $a$ suchen, führe eine *„fan-operation“* auf diese *„fan-sequence“* durch, beginnend am Konten $v$. Die *„fan-sequence“* terminiert nun in $x(v)$. Dies ermöglicht, dass $e(v)$ eingefärbt wird und alle Kanten, welche zuvor eingefärbt wurden, sind ebenfalls noch eingefärbt.

Schritt 7: Es sei

$$I_3 = \{v \in I_2 \mid y(v)\ gesuchte\ \ Frabe\ \ b\}.$$

Obwohl anfänglich $y(v)$ als die gesuchte Farbe $b$ definiert wurde, gilt dies nicht mehr nach dem zuletzt durchgeführten Schritt 6. Wie dem auch sei, solange jedoch $v \in I_2$ ist, muss für $v$ gesuchte Farbe $a$ und für $x(v)$ die gesuchte Farbe $b$ gelten.

Schritt 8: Analog zu Schritt 6 wird auch Schritt 8 durchgeführt. Konstruiere einen neuen Graphen

$$G_{fan} = (I_3, E_{fan})$$

wie folgt: Für jeden Pfad, der als eine Komponente aus $G[a,b]$ angesehen wird und mindestens mit einem Punkt in $y(v)$ endet, wobei auch hier $v \in I_3$ ist und der andere Endpunkt entweder gleich $u$ oder gleich $x(u)$ für einen beliebigen Knoten $u \in I_3$ mit der Kante $\{u,v\} \in I_{fan}$ ist. Es sei darauf hingewiesen, dass $\Delta(G_{fan}) \leq 2$ ist. Suche als nächstes aus der Menge $G_{fan}$ eine maximal unabhängige Menge $I_{fan}$. Vertausche im Pfad $G[a,b]$ und dem Pfadende, gekennzeichnet durch den Endpunkt $y(v)$, für jedes $v \in I_{fan}$ die Farbe. Solange $x(v)$ und $v$ gemeinsam die Farbe $a$ suchen, führe eine *„fan-operation“* auf diese *„fan-sequence“* durch, beginnend am Konten $v$. Die *„fan-sequence“* terminiert nun in $y(v)$. Dies ermöglicht, dass $e(v)$ eingefärbt wird und alle Kanten, welche zuvor eingefärbt wurden, sind ebefalls noch eingefärbt.

Um nun zu zeigen, dass dieser Algorithmus korrekt ist, muss erst noch gezeigt werden, dass $v \in I$ mit $e(v)$, dass dieses Einfärben korrekt durchgeführt wurde. Als Beispiel seien zwei *„fan-sequences“* mit $u,v \in I \mid u \neq v$ angenommen. Solange es sich bei der Menge $I$ um eine

unabhängige Menge zu der Menge $G^2$ handelt, so gilt für jeden Knoten aus der *„fan-sequence“* $u$, dass er nun in dieser *„fan-sequence“* und auf keinen Fall in der *„fan-sequence“* $v$ zu finden ist.

Für den Fall, dass ein Knoten $v$, auf den in Schritt 3 eine *„fan-operation“* angewandt wurde. Durch die oben durchgeführte Argumentation, Alle diese Operationen können gleichzeitig (parallel) durchgeführt werden. Die parallel Durchführung ist jedoch an eine Einschränkung gebunden. Die parallel Berechnung kann nur durchgeführt werden, wenn es keine *„fan-sequence“* gibt, die von einer anderen *„fan-operation“* verändert (manipuliert) wird.

Ein weiteres Problem wird durch das *„chain-recoloring”* verursacht. Es stellt sich nun die Frage, ob es möglich ist, einen Pfade $x(v)$ unter Verwendung der *„fan-sequence“* beginnend bei $u$ wiedereinzufärben? Für die Endpunkte eines Pfades kann sich die Menge der gesuchten Farbe laufenden ändern, für alle übrigen Knoten des Graphs jedoch, bleibt die Menge der gesuchten Farben unverändert.

Es sei darauf hingewiesen, dass In Schritt 6 festgestellt wurde, dass jeder Knoten

$$v \in I_{fan}$$

ist. Es stellt sich nun die Frage, welche Änderungen während der Einfärbung durch vorangegangene oder gleichzeitige Maßnahmen, Auswirkungen auf das *„recoloring“* dieses Knotens hat. In Schritt 3, wurden vielleicht andere *„fan-operationen“* durchgeführt. Es ist jedoch zu Beachten, dass die *„fan-operationen“* keinerlei Auswirkungen auf die Umgebung des Knotens $v$ haben darf. Darüber hinaus, besteht noch die Möglichkeit, dass einige *„chain-recolorings“* durchgeführt wurden. Solange jedoch $I_{fan}$ eine unabhängige Menge $G_{fan}$ ist, gilt folgendes:

Keiner der wiedereingefärbten Pfade verfügt über einen Endpunkt, welcher in der *„fan-sequence“*

$$(v, a, ..., x(v), b)$$

ist.

Aus diesem Grund ist auch keine Kante aus der *„fan-sequence“* mit den Farben $a$ oder $b$ eingefärbt, so dass die Kantenfarbe sich in Verbindung mit der *„fan-sequence“* nicht verändert. Bekräftigt durch Aussage bleibt die *„fan-sequence“* korrekt und ist in ihrer Gültigkeit bestätigt.

Im Folgenden sollen die Knoten $v \ in \ I_{fan}$ betrachtet werden. Wie bereits gezeigt wurde, wissen wir, dass eine andere *„fan-operation“* während diesem oder einem vorangegangen Schritt durchgeführt wird, keinerlei Auswirkungen auf die Gültigkeit der *„fan-sequence“* verursacht. Daraus folgt, dass solange $I_{fan}$ eine Teilmenge der Menge $I_3$ ist, hat das durchgeführte *„chain-recoloring“* in Schritt 6 keine Auswirkung auf die Menge der Farben, für die in der *„fan-sequence“*, beginnend mit dem Knoten $v$ eine Farbe gesucht wird.

Solange $I_{fan}$ unabhängig ist, besteht die Gewissheit, dass jedes *„chain-recoloring“*, welches in Schritt 8 durchgeführt wird, keine Auswirkung auf die gesuchten Farben in dieser *„fan-sequence“*. Für den Fall, dass eine Kante aus der *„fan-sequence“* mit der Farbe *b* eingefärbt wurde, muss gezeigt werden, dass diese Farbe nicht alterniert.

Für diese Kante ist bekannt, dass sie sich im Pfad $G[a,b]$befindet und durch die Knoten $x(v)$ und $v$ charakterisiert ist. Es sei darauf hingewiesen, dass der Knoten $v \in I_2$ ist. Diese und alle übrigen Kantenfarbe aus der *„fan-sequence“*, beginnend mit dem Knoten $v$ behalten ihre ursprüngliche Farbe und werden nicht erneut eingefärbt.

Im Folgenden muss noch gezeigt werden, eine ausreichende Anzahl ungefärbter Kanten in einer definierten Zeit (Phase) eingefärbt werden können. Es sei $i_j$ die Anzahl der Kanten, welche im Schritt $j \; mit \; j = 3,6,8$ eingefärbt wurden. In Schritt 4, konnten die Knoten $I^* - I_{a,b}$ aus der Kantenfärbung ausgeschlossen werden. Solange es nur $\Delta(G)+1$ Farben gibt, so verfügt ein Farbenpaar $(a,b)$ im Durchschnitt über $|I^*|/(2\Delta(G)^2)$ Knoten. Als Konsequenz muss eine Farbenpaar diese Knoten abdecken. Es sei $k$ die Anzahl der Konten der Menge $I_{a,b}$. Es ist nun möglich die Anzahl der Knoten, in der unabhängigen Menge $I_{a,b}$, zu ermitteln. Von Interesse sind jedoch nur die Knoten, dessen angrenzende Kanten sich in einem ungefärbten Zustand befinden. In Schritt 6 werden nun diese Knoten, resultierend aus $I_1 - I_{fan}$ ergeben, eingefärbt, welche keine angrenzenden eingefärbten Kanten aufweisen. Es ist jedoch zu beachten, dass solange $\Delta(G_{fan}) \le 2$ ist, gibt es maximal $2 \cdot i_6$ solcher potentiellen Kanten. In Schritt 7 wird eine Anzahl Knoten aus $I_2 - I_3$ ermittelt, welche nicht $i_6$ übersteigt, solange kann mit jedem *„chain-recoloring“* nur die Kanten $y(v) \in I_2$ einfärbt werden. In Schritt 8 kann die Anzahl der Knoten, resultierend aus $I_3 - I_{fan}$ , maximal $2 \cdot i_8$ betragen. Somit ist ein Teil der Knoten aus $I_{a,b}$ für welche eine Kanten eingefärbt wurde, mindestens

$$\frac{(i_6 + i_8)}{(4 \cdot i_6 + 3 \cdot i_8)} \ge \frac{1}{4}.$$

Daraus folgt, dass insgesamt mindestens

$$i_3 + \left(\frac{k}{4}\right)$$

Kanten eingefärbt sind. Die ursprünglich unabhängige Menge $I$ enthält maximal

$$i_3 + 2 \cdot k \cdot \Delta(G)^2$$

Knoten. Für den Fall, dass

$$\Delta\left(\left(G^2\right)_A\right) \le (\Delta(G))^2$$

ist,

so folgt für jede maximal unabhängige Menge aus $\left(G^2\right)_A$: Eine maximal unabhängige Menge $\left(G^2\right)_A$ verfügt über

$$\Omega \cdot \frac{|A|}{(\Delta(G))^2}$$

Knoten. Somit wurde eine Kante eingefärbt, welche in

$$\Omega \cdot \left( \left( \frac{1}{\Delta(G)^2} \right) \cdot \left( \frac{|A|}{\Delta(G)^2} \right) \right)$$

Knoten enthalten ist. Wie bereits erwähnt, ist somit sichergestellt, dass ein Teil der ungefärbten Kanten, welche durch

$$\Omega \left( \frac{1}{\Delta(G)^5} \right)$$

beschrieben werden kann, im Laufe einer Phase eingefärbt wird. Hieraus ergibt sich, dass die Phaseanzahl

$$O\left(\Delta(G)^5 \cdot \log|V|\right)$$

beträgt. Die Zeitkomplexität beträgt somit:

$$O\left(\Delta(G)^{O(1)} \cdot \log^{O(1)}|V|\right)$$

unter Verwendung von:

$$O\left(|V| \cdot O(1)\right)$$

Prozessoren.

Der Beweis für folgendes Theorem wurde somit vollständig erbracht.

Theorem: Die Kanten eines einfachen Graphen $G$ kann mit $\Delta(G)+1$ Farben in einer polynominellen Zeit zwischen $\Delta(G)$ und $\log|V|$ unter Verwendung einer polynominellen Anzahl Prozessoren $|V|$.

Folglich gilt folgender Satz: Die dargsestellte Graphenklasse verfügt über $\Delta(G) = O\left(\log^{O(1)} \cdot |V|\right)$ *„edge coloring“* mit $\Delta(G)+1$ Farben. Diese Klasse kann in *NC* durchgeführt werden.

# 5. Ein Zufälliger NC-Algorithmus für einfache Graphen

## *5.1. Einleitung und Grundlage des NC-Algorithmus*

Im diesem letzten Kapitel soll gezeigt werden, auf welche Weise einen Algorithmus zum Einfärben von Graphen mit einem polylogarithmischen Grad unter Verwendung von $\Delta+1$ Farben in *NC*. Zur Wiederholung, *NC* (Nick´s Class) [siehe Kapitel 1.1] beschreibt eine Klasse von Problemen, welche in polylogarithmischer Zeit $\log|V|$ mir höchstens einer polynominalen Anzahl von Prozessoren gelöst werden kann. Die Anzahl der verwendeten Farbe beträgt in diesem Fall maximal

$$\Delta + 20 \cdot \Delta^{5+\varepsilon}$$

wobei

$$\varepsilon \leq \frac{1}{4}$$

beträgt. Für die nachfolgende Betrachtung wird angenommen, dass

$$\varepsilon \leq \frac{1}{k} \text{ mit } k \geq 4$$

ist. Die Grundidee des Algorithmus ist, für den Fall, dass $\Delta$ sehr groß ist, muss der Graph in zwei Graphen partitioniert werden. Nach dieser Partitionierung entstehen zwei neune Graphen, $G_A$ und $G_B$. Im Falle von $G_B$ handelt es um einen sog. *„bipartiten“* Graphen [siehe Kapitel 1.5]. Ein *„bipartiter“* Graph ist ein Graph, welcher Ein Menge von Knoten, die in zwei disjunkte Mengen unterteilt ist, sodass keine zwei Knoten innerhalb der gleiche Untermenge benachbart sind – also über eine Kante miteinander verbunden. Für die partitionierten Graphen $G_A$ und $G_B$ gilt somit:

$$\Delta(G_A) \leq \left( \frac{\Delta(G)}{2} \right) + \Delta(G)^{5+\varepsilon}$$

sowie

$$\Delta(G_B) \leq \left( \frac{\Delta(G)}{2} \right) + \Delta(G)^{5+\varepsilon}$$

Eine effiziente und optimale Einfärbung eines „bipartiten“ Graphs kann unter Verwendung des [LPV] - Algorithmus von *Pippenger* und *Valiant* erzielt werden. Die Einfärbung des Graphs $G_A$ kann durch rekursives Anwenden der Farben erreicht werden. Es sein jedoch darauf hingewiesen, dass bereits verwendete Farbe nicht erneut eingesetzt werden dürfen.

Im Folgeden soll nun bewiesen werden, dass der beschriebene Algorithmus maximal

$$\Delta(G)+20\cdot\Delta(G)^{5+\varepsilon}$$

Farben benötigt und verwendet.

Es sei $q(d)$ das Maximum der Farben, welche zur Einfärbung von Graphen mit einem maximalen Grad $d$ benötigt werden. Für den Fall, dass

$$d^{5+\varepsilon}\geq\frac{d}{4}$$

ist, dann kommt der Algorithmus für Graphen kleineren Grades zum Einsatz. Folglich verwenden diese Graphen maximal $d+1$ Farben. Diese Bedingung ist äquivalent zu:

$$d<16^{\frac{1}{(1-2\varepsilon)}}$$

Aus diesem Grund kann nun folgende Forderung, für den anstehenden Beweis, definiert werden:

$$q(d)\leq d+c\cdot d^{5+\varepsilon}$$

Die Beweisführung wird durch induktiv durchgeführt, wobei für den Beweis *c = 20* vollkommen ausreichend ist.

Durch Betrachtung des Algorithmus für $d\geq16^{\frac{1}{(1-2\varepsilon)}}$ ist leicht erkennbar, dass

$$q(d)\leq\frac{d}{2}+d^{5+\varepsilon}+q\left(\left\lfloor\frac{d}{2}+d^{5+\varepsilon}\right\rfloor\right)\text{ ist,}$$

welches durch induktives Einsetzen maximal

$$2\cdot\left(\frac{d}{2}+d^{5+\varepsilon}\right)+c\cdot\left(\frac{d}{2}+d^{5+\varepsilon}\right)^{5+\varepsilon}\leq d+2\cdot d^{5+\varepsilon}+c\cdot\left(\frac{3d}{4}\right)^{5+\varepsilon}\leq d+c\cdot d^{5+\varepsilon}$$

ist.

Jedoch unter der Vorraussetzung, dass

$$c\geq2+c\cdot\left(\frac{3}{4}\right)^{5+\varepsilon}$$

ist.

Folglich besteht die Notwendigkeit, zu beweisen, auf welche Weise die Partitionierung der Kanten realisiert werden kann. Nachfolgend wird gezeigt, dass eine solche Partitionierung durch zufälliges partitionieren der Knoten erreicht werden kann. Für den Fall, dass beide Partitionierungen ein und den selben Endpunkt beinhalten, so enthält diese Partitionierung

schließlich eine Kante im *„bipartiten“* Graphen. Das Partitionierungsverfahren kann wie folgt durchgeführt werden:

Schritt 1: Wähle $|V|$ unabhängige und zufällige Variablen, welche entweder den Wert *0* oder *1* haben.

Schritt 2: Füge den *i-ten* Knoten in die Menge $X$ ein, falls die *i-te* Variable *0* ist. Für den Fall, dass die *i-te* Variable *0* ist, füge diesen *i-ten* Knoten in die Menge $Y$ ein. Die Knotenmengen $X$ und $Y$ repräsentieren die zwei neuen Partitionen der ursprünglichen Knotenmenge $V$.

Ein qualitatives Merkmal bei Graphpartitionierungen, ist der sog. *„split“* oder *„good split“*.

Ein *„good split”* einer Knotenmenge $V$ in zwei neuen Partitionen und $Y$ ist gegeben, wenn die Graphen:

$$G_B = (X, Y, E_B) \ mit \ \ E_B = \{\{u,v\} \in E | u \in X, v \in Y\}$$

und

$$G_A = (V, E - E_B)$$

beide über den maximalen Grad:

$$\frac{\Delta(G)}{2} + \Delta(G)^{5+\varepsilon}$$

verfügen.

Daraus ergibt sich folgende Aussage (*„Chernoff bound“* [EG]):

Die Wahrscheinlichkeit, dass bei einer $d$ - mal geworfenen Münze, mehr als

$$\left(\frac{d}{2}\right) + t \cdot \sqrt{d}$$

mal Kopf fällt, ist im Maximum durch

$$e^{-\frac{t^2}{2}}$$

begrenzt.

Daraus folgt folgender Satz: Es sei $G$ ein Graph mit

$$\Delta(G) \geq 2 \cdot \left(2\sqrt{\ln \cdot n}\right)^{\frac{1}{\varepsilon}} \ mit \ \ n = |V|$$

Ein *„good split“* einer zufälligen Partitionierung ist genau dann gegeben, wenn die Wahrscheinlichkeit mindestens

$$1-\left(\frac{2}{n}\right)$$

beträgt.

Beweis: Im ersten Schritt wird die Wahrscheinlichkeit für einen *„bad split“* berechnet. Unter Annahme, dass der Grad der Konten $v=d$ ist. Für den Fall, dass

$$d \le \frac{\Delta(G)}{2}$$

ist, spielt es keine Rolle, welche benachbarte Knoten zum Knoten $v$ partitioniert sind. Der Grad des Knotens $v$ im Graphen $G_A$ oder im Graphen $G_B$ beträgt jedoch mindestens:

$$\frac{\Delta(G)}{2}$$

Aus diesem Grund ist es möglich die Knoten mit

$$d > \frac{\Delta(G)}{2} \ge \left(2 \cdot \sqrt{\ln \cdot n}\right)^{\frac{1}{\varepsilon}}$$

zu vernachlässigen.

Es sei $N(v)$ die Menge der benachbarten (adjazenten) Knoten zum Konten $v$. Es wurde bereits festgestellt, dass diese Knoten mittels Partitionierung in die Mengen $X$ und $Y$ eingefügt wurden. Für diese Partitionierung gilt:

$$\Pr\left[\left|X \cap N(v)\right| \ge \left(\frac{d}{2}\right) + t \cdot \sqrt{d}\right] \le e^{-\frac{t^2}{2}}$$

Für den Fall, dass

$$t = 2 \cdot \sqrt{\ln \cdot n}$$

ist, dann ist

$$e^{-\frac{t^2}{2}} = \frac{1}{n^2}$$

Analog zu der *„X-Partition“* folgt für die Partition mit allen $Y$ - Knoten:

$$\Pr\left[\left|Y \cap N(v)\right| \ge \left(\frac{d}{2}\right) + t \cdot \sqrt{d}\right] \le \frac{1}{n^2}$$

Solange $t = 2 \cdot \sqrt{\ln \cdot n}$ und $d \geq \left(2 \cdot \sqrt{\ln \cdot n}\right)^{\frac{1}{\varepsilon}}$ ist, bedeutet dies, dass $t \leq d^{\varepsilon}$ ist.

Daraus lässt sich nun ableiten, dass eine Kante, welche inzident zum Konten $v$ ist, einen *„bad split“* mit einer Wahrscheinlichkeit von maximal

$$\frac{2}{n^2}$$

hervorruft. Daraus folgt eine Wahrscheinlichkeit für einen *„good split“* mit

$$1 - n \cdot \left(\frac{2}{n^2}\right) = 1 - \left(\frac{2}{n}\right)$$

Somit gilt der Satz als bewiesen.

Aus den nun gewonnen Erkenntnissen lässt ein Algorithmus ($\Delta + \Delta^{5+\varepsilon}$) ableiten, der im Folgenden beschrieben wird.

*5.2. Der $\Delta + \Delta^{5+\varepsilon}$ - Algorithmus*

Schritt 1: Falls

$$\Delta(G) \leq \max\left\{2 \cdot \left(2 \cdot \sqrt{\lfloor \lg \cdot n \rfloor}\right)^{\frac{1}{\varepsilon}}, 16^{\frac{1}{(1-2\cdot\varepsilon)}}\right\}$$

ist, dann verwende den Algorithmus $\Delta + 1$ und beende anschließend die Berechnung.

Schritt 2: Suche nach einem *„good split“*.

Schritt 3: Färbe den Graphen $G_B$ mit den Farben $\Delta(G_B)$, unter Verwendung des Algorithmus [LPV] ein.

Schritt 4: Färbe den Graphen $G_A$, unter Verwendung neuer Farben und durch rekursiven Aufruf dieses $\Delta + \Delta^{5+\varepsilon}$ - Algorithmus, ein.

Aus den Algorithmus ist sehr deutlich die Anzahl der rekursiven Aufrufe von $O(\log \Delta)$ erkennbar. Die Zeitkomplexität in Schritt 2 beträgt $O(1)$ und ist somit konstant in ihrer Ausführungszeit.

Abschließend resultiert aus diesen Erkenntnissen das folgende Theorem:

Für jedes positive

$$\varepsilon = \frac{1}{k} \leq \frac{1}{4}$$

färbt der $\Delta + \Delta^{5+\varepsilon}$ - Algorithmus einen arbitrierten Graphen $G = (V, E)$ mit maximal

$$\Delta(G) + 20 \cdot \Delta(G)^{5+\varepsilon}$$

Farben ein. Die Ausführungszeit ist polynominal in $\log|V|$. Die Anzahl der verwendeten (benötigten) Prozessoren ist mit $|V|$ festgelegt.

## 6. Referenzen

**Brinkmann**, G. "Fast Generation of Cubic Graphs." *J. Graph Th.* 23, 139-149, 1996.

**Gupta**, R. P. "The Chromatic Index and the Degree of a Graph." *Not. Amer. Math. Soc.* 13, 719, 1966.

**Harary**, F. *Graph Theory.* Reading, MA: Addison-Wesley, 1994.

**Holyer**, I. "The NP-Completeness of Edge Colorings." *SIAM J. Comput.* 10, 718-720, 1981.

**Vizing**, V. G. "On an Estimate of the Chromatic Class of a *p*-Graph" [Russian]. *Diskret. Analiz* 3, 23-30, 1964.

**G. F. Lev, N. Pippenger, and L.G. Valiant**. A fast parallel algorithm for routing in permutaion networks. IEEE Trans. Comput. 30 (1981), 93-100

**C.E. Shannon**, A theorem on coloring lines of a network, J. Math. Phys. 28 (1949), 148-151.

**W. Liang, X. Shen, and Q. Hu.** Parallel Algorithms for the Edge-Coloring and Edge-Coloring Update Problems. Jornal of Papers and distri. Computing 32, 66-73 (1996) Article No. 0005.

**J. Misra and D. Gries.** A constructive proof of Vizing´s theorem. Inform. Process. Lett. 41 (1992), 131-133.