WS 2003/04

Diskrete Strukturen I

Ernst W. Mayr

mayr@in.tum.de
Institut für Informatik
Technische Universität München

02-13-2004

Greedy Färbungsalgorithmus:

ı1 Gegeben: Graph $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ı1 Gesucht: Färbung $c : V \to \{1, \ldots, n\}$ ı1 ı1 $c[v_1] \leftarrow 1$; ı1 $i := 2$ n ı2
$c[v_i] \leftarrow \min\{k \in \mathbb{N};\ k \neq c(u)$ für alle ı6
$u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$; ı1

Für die Anzahl $C(G)$ von Farben, die der Algorithmus benötigt, gilt:

$$\chi(G) \leq C(G) \leq \Delta(G) + 1 \; .$$

Die Anzahl der verwendeten Farben hängt im Allgemeinen stark von der Reihenfolge ab, in der die Knoten betrachtet werden. So gibt es immer eine Reihenfolge, die nur $\chi(G)$ Farben benützt (ohne Beweis).

Für die Anzahl $C(G)$ von Farben, die der Algorithmus benötigt, gilt:

$$\chi(G) \leq C(G) \leq \Delta(G) + 1 \; .$$

Die Anzahl der verwendeten Farben hängt im Allgemeinen stark von der Reihenfolge ab, in der die Knoten betrachtet werden. So gibt es immer eine Reihenfolge, die nur $\chi(G)$ Farben benützt (ohne Beweis).

Für die Anzahl $C(G)$ von Farben, die der Algorithmus benötigt, gilt:

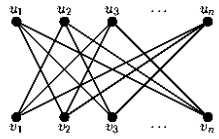
$$\chi(G) \leq C(G) \leq \Delta(G) + 1 \; .$$

Die Anzahl der verwendeten Farben hängt im Allgemeinen stark von der Reihenfolge ab, in der die Knoten betrachtet werden. So gibt es immer eine Reihenfolge, die nur $\chi(G)$ Farben benützt (ohne Beweis).

Aber es geht auch bedeutend schlechter.

Beispiel

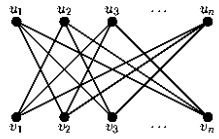
Entferne aus dem $K_{n,n}$ die Kanten zwischen gegenüberliegenden Knoten:



Aber es geht auch bedeutend schlechter.

Beispiel

Entferne aus dem $K_{n,n}$ die Kanten zwischen gegenüberliegenden Knoten:



Satz ((Brooks 1941))

Es gibt einen Algorithmus, der jeden Graph $G = (V, E)$ in Zeit $O(|V| + |E|)$ so färbt, dass für die Anzahl $A(G)$ der verwendeten Farben gilt:

$$\chi(G) \leq A(G) \leq \begin{cases} \Delta(G) + 1 & \text{falls } G = K_n \text{ oder } G = C_{2n+1}, \\ \Delta(G) & \text{sonst.} \end{cases}$$

(ohne Beweis)

Satz ((Brooks 1941))

Es gibt einen Algorithmus, der jeden Graph $G = (V, E)$ in Zeit $O(|V| + |E|)$ so färbt, dass für die Anzahl $A(G)$ der verwendeten Farben gilt:

$$\chi(G) \leq A(G) \leq \begin{cases} \Delta(G) + 1 & \text{falls } G = K_n \text{ oder } G = C_{2n+1}, \\ \Delta(G) & \text{sonst.} \end{cases}$$

(ohne Beweis)

Definition

Eine *Kantenfärbung (edge coloring) mit k Farben ist eine Abbildung $c : E \to \{1, \ldots, k\}$, so dass gilt*

$$c(e) \neq c(f) \qquad \text{für alle Kanten } e, f \in E,\ e \cap f \neq \emptyset\ .$$

Der chromatische Index (chromatic index) $\chi'(G)$ ist die minimale Anzahl Farben, die für eine Kantenfärbung von G benötigt wird.

Definition

Eine *Kantenfärbung (edge coloring) mit k Farben ist eine Abbildung $c : E \to \{1, \ldots, k\}$, so dass gilt*

$$c(e) \neq c(f) \qquad \text{für alle Kanten } e, f \in E,\ e \cap f \neq \emptyset\ .$$

Der chromatische Index (chromatic index) $\chi'(G)$ ist die minimale Anzahl Farben, die für eine Kantenfärbung von G benötigt wird.

Beobachtung:

Für jeden Graphen G gilt:

$$\chi'(G) \geq \Delta(G) \; .$$

Jedoch: Die Frage

„Gegeben ein Graph $G = (V, E)$, gilt $\chi'(G) = \Delta(G)$?“

ist wiederum $\mathcal{NP}$-vollständig.

Beobachtung:

Für jeden Graphen G gilt:

$$\chi'(G) \geq \Delta(G) \; .$$

Jedoch: Die Frage

„Gegeben ein Graph $G = (V, E)$, gilt $\chi'(G) = \Delta(G)$?“

ist wiederum $\mathcal{NP}$-vollständig.

Dies ist umso bemerkenswerter, da:

Satz ((Vizing 1964))

[2ex] Es gibt einen Algorithmus, der die Kanten eines Graphen $G = (V, E)$ in Zeit $O(|V| \cdot |E|)$ so färbt, dass für die Anzahl $A(G)$ der verwendeten Farben gilt:

$$\Delta(G) \leq \chi'(G) \leq A(G) \leq \Delta(G) + 1.$$

(ohne Beweis)

Dies ist umso bemerkenswerter, da:

Satz ((Vizing 1964))

[2ex] Es gibt einen Algorithmus, der die Kanten eines Graphen $G = (V, E)$ in Zeit $O(|V| \cdot |E|)$ so färbt, dass für die Anzahl $A(G)$ der verwendeten Farben gilt:

$$\Delta(G) \leq \chi'(G) \leq A(G) \leq \Delta(G) + 1.$$

(ohne Beweis)

$\mathcal{NP}$-Vollständigkeit

Grundlagen

Definition

(i) Eine aussagenlogische Formel ist aus Variablen, booleschen Operatoren $(\vee, \wedge, \neg)$, Konstanten 0, 1 und Klammern aufgebaut.

(ii) Eine boolesche Formel heißt *erfüllbar, wenn es Belegungen der Variablen mit Werten aus* $\{0,1\}$ *gibt, so dass die Formel* $\equiv 1$ *wird. Z.B.* $(x \wedge y) \vee (\bar{x} \wedge \bar{y})$.

(iii) *SAT ist die Sprache, die (genau) die erfüllbaren aussagenlogischen Ausdrücke enthält (in geeigneter Kodierung).*

(iv) *Eine aussagenlogische Formel* F *ist in konjunktiver Normalform, falls* F *die Form* $F = C_1 \wedge \cdots \wedge C_m$ *hat, wobei* $C_i = (z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_{r_i}})$ *für* $i \in [1:m]$ *(die* C_i *nennt man auch Klauseln) und* $z_{i_j} \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_m, \bar{x}_m\}$ *für* $i \in [1:m]$, $j \in [1:r_i]$ *(die* z_{i_j} *nennt man Literale).*

$\mathcal{NP}$-Vollständigkeit

Grundlagen

Definition

(i) Eine aussagenlogische Formel ist aus Variablen, booleschen Operatoren $(\vee, \wedge, \neg)$, Konstanten 0, 1 und Klammern aufgebaut.

(ii) Eine boolesche Formel heißt *erfüllbar, wenn es Belegungen der Variablen mit Werten aus* $\{0,1\}$ *gibt, so dass die Formel* $\equiv 1$ *wird. Z.B.* $(x \wedge y) \vee (\bar{x} \wedge \bar{y})$.

(iii) *SAT ist die Sprache, die (genau) die erfüllbaren aussagenlogischen Ausdrücke enthält (in geeigneter Kodierung).*

(iv) *Eine aussagenlogische Formel* F *ist in konjunktiver Normalform, falls* F *die Form* $F = C_1 \wedge \cdots \wedge C_m$ *hat, wobei* $C_i = (z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_{r_i}})$ *für* $i \in [1:m]$ *(die* C_i *nennt man auch Klauseln) und* $z_{i_j} \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_m, \bar{x}_m\}$ *für* $i \in [1:m]$, $j \in [1:r_i]$ *(die* z_{i_j} *nennt man Literale).*

$\mathcal{NP}$-Vollständigkeit

Grundlagen

Definition

(i) Eine aussagenlogische Formel ist aus Variablen, booleschen Operatoren $(\vee, \wedge, \neg)$, Konstanten 0, 1 und Klammern aufgebaut.

(ii) Eine boolesche Formel heißt *erfüllbar, wenn es Belegungen der Variablen mit Werten aus* $\{0,1\}$ *gibt, so dass die Formel* $\equiv 1$ *wird. Z.B.* $(x \wedge y) \vee (\bar{x} \wedge \bar{y})$.

(iii) *SAT ist die Sprache, die (genau) die erfüllbaren aussagenlogischen Ausdrücke enthält (in geeigneter Kodierung).*

(iv) *Eine aussagenlogische Formel* F *ist in konjunktiver Normalform, falls* F *die Form* $F = C_1 \wedge \cdots \wedge C_m$ *hat, wobei* $C_i = (z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_{r_i}})$ *für* $i \in [1:m]$ *(die* C_i *nennt man auch Klauseln) und* $z_{i_j} \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_m, \bar{x}_m\}$ *für* $i \in [1:m]$, $j \in [1:r_i]$ *(die* z_{i_j} *nennt man Literale).*

Definition

(a) CNF-SAT ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält.

(b) 3-CNF-SAT (aka 3-SAT) ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält, bei denen jede Klausel aus maximal drei Literalen besteht.

Bemerkung: 2-SAT $\in \mathcal{P}$. (Idee: $(a \vee b) \Leftrightarrow (\bar{a} \Rightarrow b \wedge \bar{b} \Rightarrow a)$)

Satz

(i) *SAT ist $\mathcal{NP}$-vollständig.*

(ii) *CNF-SAT ist $\mathcal{NP}$-vollständig.*

(iii) *3-SAT ist $\mathcal{NP}$-vollständig.*

Definition

(a) CNF-SAT ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält.

(b) 3-CNF-SAT (aka 3-SAT) ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält, bei denen jede Klausel aus maximal drei Literalen besteht.

Bemerkung: 2-SAT $\in \mathcal{P}$. (Idee: $(a \vee b) \Leftrightarrow (\bar{a} \Rightarrow b \wedge \bar{b} \Rightarrow a)$)

Satz

(i) *SAT ist $\mathcal{NP}$-vollständig.*

(ii) *CNF-SAT ist $\mathcal{NP}$-vollständig.*

(iii) *3-SAT ist $\mathcal{NP}$-vollständig.*

Definition

(a) CNF-SAT ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält.

(b) 3-CNF-SAT (aka 3-SAT) ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält, bei denen jede Klausel aus maximal drei Literalen besteht.

Bemerkung: 2-SAT $\in \mathcal{P}$. (Idee: $(a \vee b) \Leftrightarrow (\bar{a} \Rightarrow b \wedge \bar{b} \Rightarrow a)$)

Satz

(i) *SAT ist $\mathcal{NP}$-vollständig.*

(ii) *CNF-SAT ist $\mathcal{NP}$-vollständig.*

(iii) *3-SAT ist $\mathcal{NP}$-vollständig.*

Definition

(a) CNF-SAT ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält.

(b) 3-CNF-SAT (aka 3-SAT) ist die Sprache, die die erfüllbaren aussagenlogischen Formeln in konjunktiver Normalform enthält, bei denen jede Klausel aus maximal drei Literalen besteht.

Bemerkung: 2-SAT $\in \mathcal{P}$. (Idee: $(a \vee b) \Leftrightarrow (\bar{a} \Rightarrow b \wedge \bar{b} \Rightarrow a)$)

Satz

(i) *SAT ist $\mathcal{NP}$-vollständig.*

(ii) *CNF-SAT ist $\mathcal{NP}$-vollständig.*

(iii) *3-SAT ist $\mathcal{NP}$-vollständig.*

Beweis

(Skizze)

3-SAT ist die stärkste Behauptung.

(i) 3-SAT $\in \mathcal{NP}$:

Teste nichtdeterministisch alle Belegungen der Variablen. Ist mindestens eine erfüllend, so akzeptiert die $\mathcal{NP}$-Maschine.

(ii) 3-SAT $\mathcal{NP}$-hart:

Sei N irgendeine $\mathcal{NP}$-Maschine.

Zu zeigen: $L(N) \leq_m^p$ 3-SAT

Da N eine $\mathcal{NP}$-Maschine ist, gibt es ein Polynom p, so dass N auf Eingabe x maximal $p(|x|)$ Schritte macht.

Betrachte Berechnung von N auf x. (Ohne Beschränkung der Allgemeinheit akzeptiert N mit leerem Band und Kopf ganz links – also eindeutigem Endzustand).

Beweis

(Skizze)

3-SAT ist die stärkste Behauptung.

(i) 3-SAT $\in \mathcal{NP}$:
Teste nichtdeterministisch alle Belegungen der Variablen. Ist mindestens eine erfüllend, so akzeptiert die $\mathcal{NP}$-Maschine.

(ii) 3-SAT $\mathcal{NP}$-hart:
Sei N irgendeine $\mathcal{NP}$-Maschine.
Zu zeigen: $L(N) \leq_m^p$ 3-SAT
Da N eine $\mathcal{NP}$-Maschine ist, gibt es ein Polynom p, so dass N auf Eingabe x maximal $p(|x|)$ Schritte macht.
Betrachte Berechnung von N auf x. (Ohne Beschränkung der Allgemeinheit akzeptiert N mit leerem Band und Kopf ganz links – also eindeutigem Endzustand).

(ii) 3-SAT $\mathcal{NP}$-hart (Fortsetzung):
Definiere boolesche Variablen

$$x_{t,i,a} \quad \hat{=} \quad \text{zum Zeitpunkt } t \text{ enthält das } i\text{-te Feld des Bandes das Zeichen } a$$

also mit $t, i \in [1, p(n)], \quad a \in \Sigma \times Q$.

(ii) 3-SAT $\mathcal{NP}$-hart (Fortsetzung):

Formel:

	„erste Zeile richtig“	Startzustand p_0; Eingabe $x_0 \dots x_{n-1}$, sonst leer
$\wedge$	„$p(n)$-te Zeile richtig“	links q_a; Band leer
$\wedge$	„in jeder Zeile steht nur ein Zustand“	ein *Paar* (q, x) taucht in jeder Zeile genau einmal auf
$\wedge$	„stimmt die Übergangsfunktion“	alle „T-Felder“ müssen konsistent gemäß der δ-Relation sein.

Man kann zeigen, dass die Formel polynomiell groß (etwa $p^3(n)$) und in polynomieller Zeit berechenbar (etwa $p^3(n)$) ist.

q. e. d.

Definition

CLIQUE ist die Sprache
$\{(G, k)$; der (ungerichtete) Graph G enthält eine Clique mit $\geq k$ Knoten$\}$.

Satz

CLIQUE ist $\mathcal{NP}$-vollständig.

(ohne Beweis)

Definition

CLIQUE ist die Sprache
$\{(G, k)$; der (ungerichtete) Graph G enthält eine Clique mit $\geq k$ Knoten$\}$.

Satz

CLIQUE ist $\mathcal{NP}$-vollständig.

(ohne Beweis)

Eine kleine Sammlung $\mathcal{NP}$-vollständiger Probleme

Definition

(i) *Partition ist die Sprache*

$$\{(b_1, \ldots, b_n);\ b_i \in \mathbb{N}, (\exists I \subseteq \{1, \ldots, n\})[\sum_{i \in I} b_i = \sum_{i \notin I} b_i]\}$$

(ii) *Binpacking ist die Sprache*

$\{(a_1, \ldots, a_n, b, h);$ $a_i, b, h \in \mathbb{N}$, *die Menge der* a_i *kann in* $\leq h$ *Klassen partitioniert werden, wobei die Summe in jeder Klasse maximal* b *ist*$\}$

(iii) *Hamiltonscher Kreis ist die Sprache*

$\{G = (V, E);$ G *ungerichteter Graph, der einen einfachen Kreis der Länge* $|V|$ *enthält*$\}$

Eine kleine Sammlung $\mathcal{NP}$-vollständiger Probleme

Definition

(i) *Partition ist die Sprache*

$$\{(b_1, \ldots, b_n);\ b_i \in \mathbb{N}, (\exists I \subseteq \{1, \ldots, n\})[\sum_{i \in I} b_i = \sum_{i \notin I} b_i]\}$$

(ii) *Binpacking ist die Sprache*

$\{(a_1, \ldots, a_n, b, h);$ $a_i,\ b,\ h \in \mathbb{N}$, *die Menge der* a_i *kann in* $\leq h$ *Klassen partitioniert werden, wobei die Summe in jeder Klasse maximal* b *ist*$\}$

(iii) *Hamiltonscher Kreis ist die Sprache*

$\{G = (V, E);$ G *ungerichteter Graph, der einen einfachen Kreis der Länge* $|V|$ *enthält*$\}$

Eine kleine Sammlung $\mathcal{NP}$-vollständiger Probleme

Definition

(i) *Partition ist die Sprache*

$$\{(b_1, \ldots, b_n);\ b_i \in \mathbb{N}, (\exists I \subseteq \{1, \ldots, n\})[\sum_{i \in I} b_i = \sum_{i \notin I} b_i]\}$$

(ii) *Binpacking ist die Sprache*

$\{(a_1, \ldots, a_n, b, h);$ $a_i,\ b,\ h \in \mathbb{N}$, *die Menge der* a_i *kann in* $\leq h$ *Klassen partitioniert werden, wobei die Summe in jeder Klasse maximal* b *ist*$\}$

(iii) *Hamiltonscher Kreis ist die Sprache*

$\{G = (V, E);$ G *ungerichteter Graph, der einen einfachen Kreis der Länge* $|V|$ *enthält*$\}$

Definition (Fortsetzung)

(iv) *TSP ist die Sprache*

$$\{(G, w, C); \quad G = (V, E) \text{ gerichtet, } w : E \to \mathbb{R}^+ \text{ Gewichtung der Kanten von } G, \; G \text{ enthält einen hamiltonschen Kreis der Länge (bzgl. Gewicht)} \leq C\}$$

Satz

Partition, Hamiltonscher Kreis und TSP sind $\mathcal{NP}$-vollständig.

Korollar 26: Entweder *alle $\mathcal{NP}$-vollständigen Probleme haben einen polynomiellen Algorithmus, oder keines dieser Probleme ist $\in \mathcal{P}$.*

Satz

Partition, Hamiltonscher Kreis und TSP sind $\mathcal{NP}$-vollständig.

Korollar 26: Entweder *alle $\mathcal{NP}$-vollständigen Probleme haben einen polynomiellen Algorithmus, oder keines dieser Probleme ist $\in \mathcal{P}$.*