1. DEFINITION

Sei Π ein Optimierungsproblem und A ein Approximationsalgorithmus für Π.

a) A hat bei Eingabe I die *absolute Güte* von
$$K_A(I) = |A(I) - OPT(I)| .$$
b) Die *absolute worst-case-Güte* von A ist die Funktion
$$K_A^{wc}(n) = max\{K_A(I) \mid I \epsilon D, |I| \leq n\} .$$
c) Sei $K_A: \mathbb{N} \to \mathbb{N}$ eine Funktion. A garantiert eine absolute Güte von $K_A(n)$, falls für alle n gilt: $K_A^{wc}(n) \leq K_A(n)$.
d) Sei $K_A': \mathbb{N} \to \mathbb{N}$ eine Funktion. A hat eine *absolute Abweichung* von $K_A'(n)$, falls für unendlich viele n gilt: $K_A'(n) \leq K_A^{wc}(n)$.

   Eine unendliche große Menge $D', D' \subseteq D$ heißt $K_A'(n)$-*Zeugenmenge* gegen A, wenn für alle $I \in D'$ gilt: $K_A(I) \geq K_A'(|I|)$. Eine einzelne solche Eingabe nennen wir dann (etwas ungenau) einen $K_A'(n)$-*Zeugen*.

2. GRAPHENFÄRBBARKEIT

Sei $G = (V, E)$ ein ungerichteter Graph. Für $u \in V$ ist $\Gamma_G(u) = \{v|\{u|v\} \in E\}$ die Menge der *Nachbarn* von u und $\deg_G(u) = |\Gamma_G(u)|$ der *Grad* von u. Der *Grad* von G ist $\Delta(G) = max\{\deg_G|u \in V\}$. G heißt *r-regulär*, wenn $\deg_G(u) = r$ für alle Knoten $u \in V$.

DEFINITION:
Gegeben sei ein Graph $G = (V, E)$.

a) Eine Abbildung $c_v: V \to \mathbb{N}$ heißt Knotenfärbung von G, falls für alle $\{u, v\} \in E$ gilt: $c_v(u) \neq c_v(v)$.
b) Eine Abbildung $c_v: E \to \mathbb{N}$ heißt Kantenfärbung von G, falls für alle an einem Knoten u aufeinandertreffenden Kanten $\{u, v\}, \{u, w\} \in E$ gilt: $c_E(\{u, v\}) \neq c_E(\{u, w\})$

$c_v(u)$ und $c_E(\{u, v\})$ werden in diesem Zusammenhang *Farben* genannt. $|c_v(V)|$ bzw. $|c_E(E)|$ ist die Anzahl der benutzten Farben.

Das *Knoten-* bzw. *Kantenfärbungsproblem* wird charakterisiert durch

- $D = \{\langle G\rangle|G = (V, E)\ ein\ Graph\ mit\ mindestens\ einer\ Kante\}$,
- $S(\langle G\rangle) = \{c_v|c_v\ ist\ Knotenf\ddot{a}rbung\ von\ G\}$ bzw.
  $S(\langle G\rangle) = \{c_E|c_E\ ist\ Kantenf\ddot{a}rbung\ von\ G\}$
- $f(c_v) = |c_v(V)|$ bzw. $|c_E(E)|$.
- min.

Ziel der Knoten- bzw. Kantenfärbung ist es also eine Färbung zu berechnen, die für einen Eingabegraphen möglichst wenige Farben benutzt. Für die Knotenfärbung heißt diese kleinste Anzahl *chromatische Zahl* von G und wird mit $\chi(G)$ bezeichnet, für die Kantenfärbung heißt sie *chromatischer Index* von G und wird mit $\chi'(G)$ bezeichnet.

3. KNOTENFÄRBUNG

Bei der Knotenfärbung unterscheidet man zwischen Algorithmen, die jeden beliebigen Graphen aus D färben können und solchen, die nur Graphen mit bestimmten Eigenschaften färben können. Betrachten wir zunächst einen sehr einfachen Algorithmus für beliebige Graphen.

ALGORITHMUS GREEDYCOL:

for i := 1 **to** n **do**
    $c_v(u_i) = \infty$;
for i := 1 **to** n **do**
    $c_v(u_i) := min(\mathbb{N} \backslash \{c_v(\Gamma(u_i))\})$;
gib c_v aus;

Der Algorithmus kennzeichnet zuerst alle Knoten mit dem Wert ∞ was angibt, dass der Knoten ungefärbt ist. Anschließend werden die Knoten der Reihe nach mit der kleinsten Farbe eingefärbt, mit der noch keiner seiner Nachbarn bereits gefärbt ist. Wie aus dem Namen bereits erkennbar ist handelt es sich um einen Greedy-Algorithmus.

Satz:
Sei $G = (V, E)$ ein Graph. Algorithmus GREEDYCOL berechnet in Zeit $O(|V| + |E|)$ eine Knotenfärbung aus höchstens $\Delta(G) + 1$ Farben, d.h. $\text{GREEDYCOL}(G) \leq \Delta(G) + 1$.

Beweis:
Die obere Schranke können wie folgt angeben: Für einen Knoten u_i können höchstens $deg(u_i)$ Farben verbraucht sein, da es ansonsten mehr Farben wären, als der Knoten Nachbarn hat. Aus der Menge $\{1, \ldots \deg(u_i) + 1\}$ kann also noch eine kleinste Farbe gewählt werden.

Für die untere Schranke können wir angeben: Da jeder gültige Eingabegraph mindesten eine Kante haben muss, benötigt der Algorithmus auch mindestens zwei Farben. Daher ist die chromatische Zahl bzw. die optimale Anzahl an Farben $OPT(G) \geq 2$. Wir haben somit eine untere Schranke, die allerdings sehr weit vom optimalen Wert entfernt sein kann.

Satz:
Algorithmus GREEDYCOL garantiert eine absolute Güte von

$$K_{\text{GREEDYCOL}}(G) = \text{GREEDYCOL}(G) - \text{OPT}(G) \leq \Delta(G) + 1 - 2 = \Delta(G) - 1$$

Aus dieser Aussage können wir auch sehen, dass die Güte nicht als Funktion der Eingabelänge (Knoten und Kanten) gesehen werden muss, sondern in diesem Fall vom Grad von G abhängt, den man aus der Eingabelänge nicht erkennt.

Wir haben hierdurch aber lediglich gezeigt, dass der Algorithmus maximal $\Delta(G) + 1$ Farben benötigt. Wir wissen aber weder, ob er immer genau so viele Farben benötigt, noch wissen wir, ob dies die optimale Menge an Farben ist, die zur Färbung benötigt werden. Ein einfaches Beispiel eines Graphen, für den wir bei ungünstiger Nummerierung der Knoten mehr Farben als die optimale Lösung benötigen ist ein bipartiter Graph mit 4 Knoten, bei dem bis auf die direkt gegenüberliegenden Knoten alle gegenüberliegenden Knoten miteinander verbunden sind.

ALGORITHMUS GREEDYCOL_VAR:

for alle Knoten u **do** $c_{v(U)} := \infty$;
while es noch Knoten u mit $c_v(u) = \infty$ gibt **do**
    $c_v(u) := min(\mathbb{N} \backslash \{c_v(\Gamma(u))\})$;
gib c_v aus;

Der Algorithmus stellt eine Variante zu GREEDYCOL mit unveränderter absoluter Güte dar. Hier werden die Knoten nicht nach einer vorher festgelegten Reihenfolge eingefärbt, sondern zufällig.

Um einen schlechten Zeugen zu finden, der uns zeigt, dass die absolute Güte dieser Variante unverändert schlecht ist, muss neben dem Zeugen selbst auch die Reihenfolge vorgegeben werden. Es lässt sich für jeden Graphen eine Reihenfolge finden, bei mit der GREEDYCOL die optimale Lösung findet. Diese Reihenfolge zu finden ist allerdings NP-schwer.

4. KNOTENFÄRBUNG PLANERER GRAPHEN

Wie bereits zu Beginn erwähnt gibt es Algorithmen, die sich auf die Färbung bestimmter Graphen beschränken. Wir schränken unsere Menge D nun dahingehen ein, dass die Graphen planar sein sollen. Zu planaren Graphen gibt es einige Fakten, die für uns bei der Betrachtung der absoluten Güte wichtig sind:

Fakt:

a) Jeder Graph kann in Polynomzeit mit 6 Farben knotengefärbt werden.
b) [Der berühmte Vier-Farben-Satz; 1977 beweisen von Appel & Haken]
Jeder planare Graph kann mit 4 Farben knotengefärbt werden.
c) Das Entscheidungsproblem „ist der planare Graph G knoten-3-färbbar?“ ist NP-vollständig.
d) Es kann in Polynomzeit entschieden werden, ob ein Graph knoten-2-färbbar ist, und falls ja, kann eine solche Färbung berechnet werden.

Der Beweis des Vier-Farben-Satz kann in einen Algorithmus mit Laufzeit $O(|V|^2)$ verwandelt werden. Er basiert auf der Fallunterscheidung, die 633 Fälle betrachtet (bei Appel & Haken waren es 1818 Fälle). Daher begnügen wir uns bei der Betrachtung des folgenden Algorithmus auf eine Knoten-6-Färbung.

ALGORITHMUS COLPLAN

(1) Teste gemäß Fakt (d), ob G knoten-2-färbbar ist;
(2) Falls nicht: Färbe den Knoten gemäß Fakt (a) mit 6 Farben.

Für diesen Algorithmus gilt offensichtlich:

Satz:
COLPLAN garantiert eine absolute Güte von $K_{\mathrm{ColPlan}}(\mathrm{g}) \leq 3$, d.h.

$$|\mathrm{ColPlan}(\mathrm{G}) - \mathrm{OPT}(\mathrm{G})| \leq 3 .$$

Würden wir in (2) den Fakt (b) verwenden, bekämen wir als absolute Güte sogar den Wert 1.

5. EIN UNMÖGLICHKEITSERGEBNIS FÜR DAS RUCKSACKPROBLEM

Es gibt viele Probleme, für die es wohl keinen polynomiellen Approximationsalgorithmus gibt, der konstante absolute Güte erreichen kann. Wir werden dazu eine untere Schranke für RUCKSACK angeben.

Satz:
Falls $P \neq NP$ ist, gibt es keine Konstante $k \in \mathbb{N}$, so dass es einen polynomiellen Approximationsalgorithmus A für das Rucksackproblem gibt mit:

$$|A(I) - OPT(I)| \leq k .$$

Beweis:
In einem Widerspruchsbeweis versuchen wir einen exakten Algorithmus A‘ zu finden, der das Rucksackproblem in polynomieller Zeit löst und aus dem somit $P = NP$ folgt.

Zu einer Probleminstanz $I = \langle W, vol, p, B\rangle$ konstruieren wir eine Probleminstanz $I' = \langle W', vol', p', B'\rangle$ mit $W' = W, vol' = vol, B' = B$ und $p'(w) = (k+1) * p(w)$. Dabei ist offensichtlich eine zulässige Lösung σ für I auch eine zulässige Lösung für I' und umgekehrt, da vol unverändert geblieben ist, d.h. $S(I) = S(I')$. Die Werte von $\sigma \in S(I) = S(I')$ sind:

$$f_I(\sigma) = \sum_{w \in \sigma} p(w)$$

$$f_{I'}(\sigma) = \sum_{w \in \sigma} (k+1) * p(w) = (k+1) * f_I(\sigma)$$

Wegen der Monotonie der Multiplikation mit $k+1$ ist eine optimale Lösung σ_{opt} für I auch eine optimale Lösung für I'.

Nun stellen wir uns die zulässigen Lösungen ansteigend nach ihren Werten angeordnet vor, d.h. $S(I) = \{\sigma_1, \dots, \sigma_{|S(I)|}\}$ mit $f_I(\sigma_i) \leq f_I(\sigma_{i+1})$.

Sei j der Index mit $f_I(\sigma_{j-1}) < OPT(I)$ und $f_I(\sigma_j) = OPT(I)$. Damit ist $f_I(\sigma_j) - f_I(\sigma_{j-1}) \geq 1$ und $f_{I'}(\sigma_j) - f_{I'}(\sigma_{j-1}) = (k+1) * \left(f_I(\sigma_j) - f_I(\sigma_{j-1})\right) > k$.

Algorithmus *A*‘:

1. berechne die Probleminstanz I';
2. bestimme mittels A eine Lösung σ für I';
3. gib $\sigma'_A = \sigma$ aus.

A' läuft offensichtlich in polynomieller Zeit. Nun gilt $OPT(I') - f_{I'}(\sigma_{A'}) \leq k$ wegen Schritt 2. Da es für I' keine zulässigen Lösungen mit dieser Eigenschaft außer den optimalen Lösungen gibt, ist σ_A, eine optimale Lösung für I' und damit auch für I. ■

Wir haben in dem Beweis das Problem auf sich selbst reduziert. Damit kann man sagen, dass die Berechnung der exakten Lösung nicht schwieriger ist, als das Problem mit absoluter Güte zu approximieren. Man sagt, dass RUCKSACK die (absolute) *self-improvement property* habe.

Das Rucksackproblem kann also (falls, wie alle vermuten $P \neq NP$ ist) nicht mit konstanter absoluter Güte approximiert werden. Wenn wir aber nur voraussetzen, dass die absolute Abweichung von der optimalen Lösung in einem kleinen Verhältnis sein soll (z.B. eine Abweichung von 10 bei einem optimalen Wert von 1000), dann ist dieses Ziel tatsächlich erreichbar.

LITERATUR

"Approximationsalgorithmen eine Einführung" von Rolf Wanka, Veröffentlicht vom Vieweg+Teubner Verlag, 2006