Local Ratio

Local Ratio

- Der Algorithmus A berechne für die Instanz x eines Minimierungsproblems die Lösung $A(x)$.
- $\mathrm{opt}(x)$ sei der optimale Wert, $u(x) \leqslant \mathrm{opt}(x)$ eine **untere Schranke**.
- Um zu zeigen, dass A r-approximativ ist, zeige

$$\frac{\mathbf{A(x)}}{\mathrm{opt}(\mathbf{x})} \leqslant \frac{\mathbf{A(x)}}{\mathbf{u(x)}} \overset{!}{\leqslant} \mathbf{r}.$$

- Mit der „Local Ratio“-Methode versucht man, die „Kosten“ $A(x)$ als Summen $A(x) = c_1 + \cdots + c_k$ lokaler Kosten aufzufassen.
- Wenn auch die untere Schranke als Summe $u(x) = u_1 + \cdots + u_k$ mit $c_i \leqslant r \cdot u_i$ aufgefasst werden kann, folgt

$$A(x) - r \cdot u(x) = \sum_i (c_i - r \cdot u_i) \leqslant 0.$$

  - Der globale Quotient ist durch r beschränkt, weil die lokalen Quotienten durch r beschränkt sind.

Local Ratio Algorithmen für Vertex Cover und Price Collecting Vertex Cover

VERTEX COVER

- Eine Instanz ist durch einen ungerichteten Graphen $G = (V, E)$ und eine Gewichtung $w_u \geqslant 0$ für die Knoten $u \in V$ gegeben.
- Bestimme eine möglichst leichte Knotenüberdeckung, also eine Teilmenge $V' \subseteq V$, so dass jede Kante einen Endpunkt in V' hat.
- Eine Anwendung
  - Wiederhole ein Experiment mehrmals, einige Testergebnisse sind allerdings verfälscht.
  - Setze eine Kante $\{i, j\}$ ein, wenn die Ergebnisse der Tests i und j zu stark voneinander abweichen.
  - Die verfälschten Tests bilden eine Knotenüberdeckung.
- V' ist genau dann eine Knotenüberdeckung, wenn $V \setminus V'$ eine unabhängige Menge ist.

  Wir zeigen, dass VERTEX COVER wesentlich schärfer als INDEPENDENT SET approximiert werden kann.

VERTEX COVER: Der Local Ratio Algorithmus

Der ungerichtete Graph $G = (V, E)$ sei gegeben.

(1) Solange es eine Kante $\{u, v\} \in E$ mit $\min\{w_u, w_v\} > 0$ gibt:
Für $\varepsilon = \min\{w_u, w_v\}$ setze
- ⋆ $w_u = w_u - \varepsilon$ und
- ⋆ $w_v = w_v - \varepsilon$.

(2) Gib $V' = \{u \in V \mid w_u = 0\}$ aus.

Kommentar: V' ist eine Knotenüberdeckung.

Geht es denn nicht viel einfacher?

Wähle stets den billigsten Knoten?!

- Betrachte den Sterngraphen mit einem Sternzentrum (Gewicht $1 < \Delta < 2$) und n Satelliten (Gewicht jeweils 1).
  - Wenn wir stets den billigsten Knoten wählen, dann häufen wir das Gewicht n an, während Gewicht Δ ausreicht.
- Der Local Ratio Algorithmus wählt einen Satelliten im ersten Schritt und das Sternzentrum im zweiten:

  Das Gewicht beträgt $\Delta + 1 \leqslant 2 \cdot \Delta$.

Analyse des Local Ratio Algorithmus

Der Algorithmus ist 2-approximativ und läuft in Zeit $O(|V| + |E|)$.

- Wir arbeiten alle Kanten in beliebiger Reihenfolge ab.
- Wenn wir Kante $\{u, v\}$ verarbeiten, dann bezahlen wir die Kosten $c_{i+1} = 2 \cdot \min\{w_u, w_v\}$:

  Die Knotengewichte wurden bereits gemäß den bisherigen Anzahlungen reduziert.
- Jede Knotenüberdeckung verursacht aber mindestens die Kosten $u_{i+1} = \min\{w_u, w_v\}$.

  Wir haben einen 2-approximativen Algorithmus erhalten.

PRICE COLLECTING VERTEX COVER

- Ein ungerichteter Graph $G = (V, E)$ sowie Gewichtungen $w_u \geqslant 0$ der Knoten $u \in V$ und w_e der Kanten $e \in E$ sind gegeben.
- Eine Knotenmenge $V' \subseteq V$ besitzt die Kosten

$$\sum_{u \in V'} w_u + \sum_{e=\{u,v\} \in E;\ u \notin V', v \notin V'} w_e.$$

- Bestimme eine leichteste Knotenmenge.

- Eine nicht-überdeckte Kante e verursacht die Kosten w_e.
- VERTEX COVER ist ein Spezialfall, wenn die Kantengewichte auf beliebig große Werte gesetzt werden.

Der Local Ratio Algorithmus

(1) Solange es eine Kante $e = \{u, v\} \in E$ mit $\min\{w_u, w_v, w_e\} > 0$ gibt:

Für $\varepsilon = \min\{w_u, w_v, w_e\}$ setze

- $w_u = w_u - \varepsilon$,
- $w_v = w_v - \varepsilon$ und
- $w_e = w_e - \varepsilon$.

(2) Gib $V' = \{u \mid w_u = 0\}$ aus.

Kommentar: Alle mit Knoten aus V' inzidenten Kanten sind überdeckt.

Analyse

Der Algorithmus ist 2-approximativ.

- Was passiert, wenn die Kante $e = \{u, v\}$ zu überdecken ist?
  - Wir bezahlen 3ε für die Reduktion der Gewichte w_u, w_v und w_e.
  - Eine optimale Entscheidung verursacht aber mindestens die Kosten ϵ,

    denn entweder wird mindestens einer der Knoten u oder v aufgenommen oder e wird nicht überdeckt.
- Der Algorithmus ist also 3-approximativ. Warum ist er sogar 2-approximativ?

Das Local Ratio Theorem

Das Local Ratio Theorem

Wir betrachten Optimierungsprobleme der Form (w, L):

> Für $w \in \mathbb{R}^n$ und eine Lösungsmenge $L \subseteq \mathbb{R}^n$ bestimme $x \in L$, so dass das innere Produkt $w^T \cdot x = \sum_{i=1}^{n} w_i \cdot x_i$ möglichst klein ist.

Es gelte $w = w_1 + w_2$.

Wenn x eine r-approximative Lösung für (w_1, L) und (w_2, L) ist, dann ist x auch r-approximativ für (w, L).

- Seien x^*, x_1^*, x_2^* optimale Lösungen von $(w, L), (w_1, L)$ und (w_2, L).
- Dann gilt $w_1^T \cdot x_1^* \leqslant w_1^T \cdot x^*$ wie auch $w_2^T \cdot x_2^* \leqslant w_2^T \cdot x^*$.
- Aber $w_i^T \cdot x \leqslant r \cdot (w_i^T \cdot x_i^*)$ für $i = 1, 2$ und deshalb

$$\begin{aligned} w^T \cdot x &= w_1^T \cdot x + w_2^T \cdot x \leqslant r \cdot (w_1^T \cdot x_1^* + w_2^T \cdot x_2^*) \\ &\leqslant r \cdot (w_1^T \cdot x^* + w_2^T \cdot x^*) = r \cdot (w^T \cdot x^*). \end{aligned}$$

Ein rekursiver Algorithmus

(0) Es gelte $w \geqslant 0$.

(1) Wenn es eine Lösung $x \in L$ mit $x_i = 0$ für alle Komponenten i mit $w_i > 0$ gibt, dann gib x aus.

(2) Sonst bestimme eine Darstellung $w = w' + (w - w')$ mit $w' \leqslant w$, so dass jede Lösung r-approximativ für w' ist.

Kommentar: Später werden wir nur noch fordern, dass jede „nicht-verkleinerbare“ Lösung r-approximativ für w' ist.

(3) Bestimme rekursiv eine r-approximative Lösung $x \in L$ für $(w - w', L)$.

Als Konsequenz des Local Ratio Theorems:

Die Lösung x ist r-approximativ.

Anwendungen: VERTEX COVER

(1) Wenn alle Kanten von $V' = \{u \in V \mid w_u = 0\}$ überdeckt werden, dann gib V' aus.

(2) Sonst wähle eine nicht-überdeckte Kante $\{u, v\} \in E$ aus und setze

$$w'_z = \begin{cases} \min\{w_u, w_v\} & z \in \{u, v\}, \\ 0 & \text{sonst.} \end{cases}$$

Kommentar: Jede Lösung für die Gewichtung w' hat mindestens Gewicht $\min\{w_u, w_v\}$.

(3) Bestimme rekursiv eine 2-approximative Lösung für $w - w'$.

Feedback Vertex Set

Anwendungen: Feedback Vertex Set für Turniere (1/2)

- Ein Turniergraph $G = (V, E)$ ist gegeben: G besitzt für je zwei Knoten u, v genau eine der beiden Kanten (u, v), bzw. (v, u).
- Bestimme eine leichteste Knotenmenge $V' \subseteq V$ für Knotengewichte $w_u \geqslant 0$, so dass G nach Herausnahme aller Knoten in V' kreisfrei ist.

Eine zentrale Beobachtung

G besitzt einen Kreis $\Leftrightarrow$
G besitzt ein Dreieck, also einen Kreis der Länge drei.

$\Leftarrow$ Offensichtlich.

$\Rightarrow$ Sei $(v_1, \ldots, v_k, v_1)$ ein Kreis **minimaler** Länge.

- Wenn G die Kante (v_3, v_1) besitzt, dann hat G das Dreieck (v_1, v_2, v_3, v_1).
- Ansonsten besitzt G die Kante (v_1, v_3): Überspringe den Knoten 2, um den **kürzeren** Kreis $(v_3, \ldots, v_k, v_1, v_3)$ zu erhalten.

Anwendungen: Feedback Vertex Set für Turniere (2/2)

(1) Wenn es eine Knotenmenge $V' \subseteq V$ mit Gewicht 0 gibt, deren Herausnahme alle Kreise zerstört, dann gibt V' aus.

(2) Sonst wähle ein beliebiges Dreieck $\{a, b, c\}$ aus, bestimme $\varepsilon = \min\{w_a, w_b, w_c\} > 0$ und setze

$$w'_z = \begin{cases} \varepsilon & z \in \{a, b, c\}, \\ 0 & \text{sonst.} \end{cases}$$

Kommentar: Jede Lösung ist mindestens 3-approximativ, denn einer der Knoten a, b oder c muss gewählt werden.

(3) Bestimme rekursiv eine Lösung für die Gewichtung $w - w'$.

Der beste bekannte Algorithmus ist 2.5-approximativ.

FEEDBACK VERTEX SET

- Ein ungerichteter Graphen $G = (V, E)$ und eine Gewichtung $w_u \geqslant 0$ der Knoten $u \in V$ ist gegeben.
- Bestimme eine leichteste Knotenmenge $V' \subseteq V$, so dass die Herausnahme von V' alle Kreise von G zerstört.

- Wir haben FEEDBACK VERTEX SET bereits für Turniergraphen, also für eine Klasse gerichteter Graphen kennengelernt und einen 3-approximativen Algorithmus entworfen.
- Unser Ziel ist der Entwurf eines 2-approximativen Algorithmus.

Und das ist gut?

FEEDBACK VERTEX SET und VERTEX COVER (1/4)

Sei $G = (V, E)$ eine Instanz für VERTEX COVER mit den Gewichten w_u für $u \in V$.

Wir transformieren G in eine Instanz $H = (V \cup E, E^*)$ für FEEDBACK VERTEX SET.

- Die Knoten von H entsprechen den Knoten und Kanten von G.
- Zwei Knoten $u, v \in V$ sind in H genau dann miteinander verbunden, wenn u und v in G miteinander verbunden sind.
- Zusätzlich verbinden wir $u \in V$ in H mit jedem „Kantenknoten“ $e = \{u, v\} \in E$.
- w_u bleibt das Gewicht des Knotens $u \in V$, das Gewicht des Kantenknotens $e = \{u, v\} \in E$ ist das Maximum von w_u und w_v.

Wir erhalten für jede Kante $e = \{u, v\}$ in G einen Kreis $u - e - v - u$ in H, der von einem FEEDBACK VERTEX SET zerstört werden muss.

FEEDBACK VERTEX SET und VERTEX COVER (2/4)

Eine Knotenüberdeckung $V' \subseteq V$ ist ein Feedback Vertex Set in H.

Warum?

- Die Herausnahme von V' in G zerstört alle Kanten in G.
- Nach Herausnahme von V' im Graphen H erhalten alle Kantenknoten den Grad $\leqslant 1$, gehören also nicht mehr einem Kreis an.
- Kreise in H können nur durch Knoten $u \in V$ gebildet werden.

  Aber alle Kanten zwischen Knoten aus V wurden durch die Herausnahme von V' zerstört.

FEEDBACK VERTEX SET und VERTEX COVER (3/4)

Zu jedem Feedback Vertex Set W in H gibt es eine Knotenüberdeckung W' von gleichem Gewicht in G.

- Alle Kreise in H werden durch die Herausnahme von W zerstört.
- Die Herausnahme eines Kantenknotens $e = \{u, v\} \in E$ zerstört nur den einen Kreis $u - e - v - u$, der auch durch die Herausnahme von u oder v zerstören werden kann.
- O.B.d.A. besteht W nur aus Knoten in V.

  Da W alle Dreier-Kreise $u - e - v - u$ für $e = \{u, v\}$ zerstören muss, ist W auch eine Knotenüberdeckung für den Graphen G.

FEEDBACK VERTEX SET und VERTEX COVER (4/4)

- Die Gewichte einer leichtesten Knotenüberdeckung für G und eines leichtesten Feedback Vertex Set für H stimmen überein.
- Wir haben sogar eine „approximationserhaltende“ Reduktion von VERTEX COVER auf FEEDBACK VERTEX SET erhalten.

Um VERTEX COVER zu approximieren,

1. transformieren wir zuerst eine Instanz G für VERTEX COVER auf die Instanz H für FEEDBACK VERTEX SET,
2. benutzen dann einen Approximationsalgorithmus für FEEDBACK VERTEX SET, um eine Knotenmenge $W \subseteq V \cup E$ zu berechnen
3. und erhalten eine gleich schwere Knotenüberdeckung für G.

- Wenn FEEDBACK VERTEX SET einen effizienten r-approximativen Algorithmus besitzt, dann auch VERTEX COVER.
- Es sind nur effiziente 2-approximative Algorithmen für VERTEX COVER bekannt.

Spezielle Knotengewichte

Wir betrachten die Knotengewichte

$$w_u = \alpha \cdot (\mathrm{grad}(u) - 1).$$

für eine beliebige Konstante $\alpha > 0$.

- Kreise bestehen nur aus Knoten vom Grad mindestens zwei.
  - Entferne iterativ alle Knoten vom Grad eins.
  - Der entstehende Graph ist **sauber**, enthält also keine Knoten vom Grad eins.
- Ein Feedback Vertex Set W ist „**nicht-verkleinerbar**“, wenn keine echte Teilmenge von W ein Feedback Vertex Set ist.
- Wir zeigen für saubere Graphen, dass **jeder** nicht-verkleinerbare Feedback Vertex Set für die Gewichte $w_u = \alpha \cdot (\mathrm{grad}(u) - 1)$ eine 3-approximative Lösung ist.

  Wir beschränken uns auf $\alpha = 1$.

Eine obere Schranke

Der Graph G sei zusammenhängend mit n Knoten und m Kanten. Ein nicht-verkleinerbarer Feedback Vertex Set hat höchstens $m - n + 1$ Knoten.

- Sei W ein kleinster Feedback Vertex Set für G.
- Wenn wir die Knoten in W aus V entfernen, dann erhalten wir einen Wald F.
- Da W minimal ist, gibt es zu jedem Knoten $u \in W$ einen Baum T des Walds F, so dass u mit zwei Knoten von T verbunden ist.
- G ist zusammenhängend: Wenn wir die Knoten in W wieder einfügen, werden die Bäume des Walds verschmolzen.
- Jeder zusammenhängende Graph besitzt aber mindestens $n - 1$ Kanten und das zweimalige Treffen von Bäumen des Walds für jeden Knoten aus W fügt weitere $|W|$ Kanten hinzu.
- Also folgt $m \geqslant |W| + n - 1$ und deshalb gilt $|W| \leqslant m - n + 1$.

Der Graph G sei sauber. Wenn $w_u = \text{grad}(u) - 1$ und wenn $\texttt{fvs}$ das **Gewicht** eines nicht-verkleinerbaren Feedback Vertex Sets ist, dann folgt

$$m - n + 1 \leqslant \texttt{fvs} \leqslant 3 \cdot (m - n) + 1.$$

- Sei W ein nicht-verkleinerbares Feedback Vertex Set. Es ist

$$\texttt{gewicht}(W) = \sum_{u \in W} w_u = \sum_{u \in W} (\text{grad}(u) - 1) = \sum_{u \in W} \text{grad}(u) - |W|.$$

- Wie groß ist $\sum_{u \in W} \text{grad}(u)$ mindestens?
  - Der nach Herausnahme von W entstandene Wald hat höchstens $n - 1 - |W|$ Kanten.
  - Die Anzahl der mit Knoten aus W inzidenten Kanten ist mindestens $m - (n - 1 - |W|)$.
  - $\sum_{u \in W} \text{grad}(u) \geqslant m - (n - 1 - |W|)$ und als Konsequenz

$$\texttt{gewicht}(W) = \sum_{u \in W} \text{grad}(u) - |W| \geqslant m - (n - 1 - |W|) - |W| = m - n + 1.$$

$m - n + 1 \leqslant \texttt{fvs} \leqslant 3 \cdot (m - n) + 1$

- Die Summe der Knotengrade ist die doppelte Knotenzahl:

$$\texttt{gewicht}(W) = \sum_{u \in W} \operatorname{grad}(u) - |W| = 2m - \sum_{u \notin W} \operatorname{grad}(u) - |W|.$$

- Angenommen, der nach Herausnahme von W entstandene Wald besteht aus k Bäumen.
  - In $\sum_{u \notin W} \operatorname{grad}(u)$ werden zuerst alle Kanten zwischen Knoten aus $V \setminus W$ zweifach gezählt und dies sind $2 \cdot (n - |W| - k)$ Kanten.
  - Und dann alle Kanten zwischen Knoten aus $V \setminus W$ und W.
    - ★ Der Graph ist sauber: Jeder der k Bäume ist mit mindestens zwei Knoten aus W verbunden.
    - ★ Es gibt also mindestens $2k$ Kanten zwischen $V \setminus W$ und W.
- Also $\sum_{u \notin W} \operatorname{grad}(u) \geqslant 2 \cdot (n - |W| - k) + 2k = 2 \cdot (n - |W|)$ und

$$\begin{aligned} \texttt{gewicht}(W) &= 2m - \sum_{u \notin W} \operatorname{grad}(u) - |W| \leqslant 2(m - n) + |W| \\ &\leqslant 3(m - n) + 1. \end{aligned}$$

Der erste Algorithmus

(1) Wenn $V = \emptyset$, dann gib die leere Menge als Lösung aus.
Enferne alle Knoten vom Grad 1 aus V.

(2) Bestimme $\alpha = \min_{u \in V} \frac{w_u}{d_u - 1}$ und die neuen Gewichte

$$w'_u = \alpha \cdot (d_u - 1).$$

Setze $W' = \{u \in V \mid w'_u = w_u\}$.

(3) Entferne W' und wende das Verfahren rekursiv auf den kleineren Graphen mit den Knotengewichten $w_u - w'_u$ an:

Wir erhalten einen Feedback Vertex Set W'' für den kleineren Graphen.

(4) Mit Induktion über die Anzahl der Knoten:

$W' \cup W''$ ist ein Feedback Vertex Set.

Bestimme einen Feedback Vertex Set $W \subseteq W' \cup W''$, der nicht verkleinerbar ist.

Wie schwer ist W?

$$\begin{aligned}\texttt{gewicht}(W) &= \sum_{u\in W} w_u = \sum_{u\in W} \alpha\cdot(d_u-1) + \sum_{u\in W}[w_u - \alpha\cdot(d_u-1)]\\ &= \sum_{u\in W} w'_u + \sum_{u\in W}(w_u - w'_u).\end{aligned}$$

- W ist nicht verkleinerbar: $\sum_{u\in W} w'_u \leqslant 3\cdot \texttt{opt}'$

  für das Gewicht $\texttt{opt}'$ eines optimalen Feedback Vertex Sets für die Knotengewichte w'.
- Wenn $\texttt{opt}''$ das Gewicht eines optimalen Feedback Vertex Sets für den kleineren Graphen ist:

$$\begin{aligned}\sum_{u\in W}(w_u - w'_u) &\leqslant \sum_{u\in W'}(w_u - w'_u) + \sum_{u\in W''}(w_u - w'_u)\\ &= \sum_{u\in W''}(w_u - w'_u) \overset{\text{Induktion}}{\leqslant} 3\cdot\texttt{opt}''.\end{aligned}$$

Ein erstes Fazit

- Wir wissen $\texttt{gewicht}(W) \leqslant 3 \cdot (\texttt{opt}' + \texttt{opt}'')$.
- Übungsaufgabe: $\texttt{opt}' + \texttt{opt}'' \leqslant \texttt{opt}$,
  wenn $\texttt{opt}$ das Gewicht eines optimalen Feedback Vertex Sets für den ursprünglichen Graphen ist.

Der Algorithmus ist 3-approximativ.

- Ein Kreis heißt fast-disjunkt, wenn der Kreis höchstens einen Knoten vom Grad mindestens drei durchläuft.
- Wir modifizieren unseren Algorithmus, wenn es fast-disjunkte Kreise gibt.

Der verbesserte Algorithmus

(1) Wenn G keine fast-disjunkte Kreise besitzt, dann verfahre im nicht-rekursiven Schritt wie vorher.

(2) Besitzt G hingegen einen fast-disjunkten Kreis K, dann:

(a) Bestimme den Knoten $u_{\min,K}$ des Kreises K mit kleinstem Gewicht.

(b) Bestimme den Knoten u_K in K mit Grad mindestens drei.

(c) Entferne alle Knoten des Kreises bis auf bis auf u_K:

Wir zerstören K, wenn wir $u_{\min,K}$ entfernen. Nur u_K kann in weiteren Kreisen auftreten.

(d) Nenne den neuen Graphen G''. Reduziere das Gewicht von u_K um das Gewicht von $u_{\min,K}$.

Wir machen eine Anzahlung auf das Gewicht von u_K.

(e) Bestimme einen Feedback Vertex Set W'' für G'' rekursiv.

Kommentar: Wenn wir induktiv annehmen, dass W'' eine 2-approximative Lösung für G'' ist, dann ist $W'' \cup \{u_{\min}\}$ eine 2-approximative Lösung für G, denn

$\{u_{\min}\}$ ist ein leichtester Feedback Vertex Set für K.

Die Analyse

- Unser Algorithmus ist 2-approximativ auf fast-disjunkten Kreisen.
- Der Restgraph G'' besitzt keine fast-disjunkten Kreise.
  - Zeige: Wenn $w_u = \alpha \cdot (\mathrm{grad}(u) - 1)$ für alle Knoten u und wenn $\mathtt{fvs}$ das Gewicht eines nicht-verkleinerbaren Feedback Vertex Sets ist, dann folgt für G''

$$m - n + 1 \leqslant \mathtt{fvs}/\alpha \leqslant 2 \cdot (m - n) + 1.$$

  - Wende jetzt das Argument aus der Analyse des ersten Algorithmus auf G'' an.

Der neue Algorithmus ist 2-approximativ

Zusammenfassung

Der Local Ratio Ansatz kann mit Erfolg auf „Hitting Probleme“ angewandt werden:

- In VERTEX COVER müssen alle Kanten
- und in FEEDBACK VERTEX SET müssen alle Kreise getroffen werden.

Wir werden später sehen, dass Local Ratio Algorithmen als primal-duale Algorithmen aufgefasst werden können.