Themen - Approximationsalgorithmen WS19/20 (6 Seiten)

February 24, 2020

Es wird empfohlen den Stoff lesen+verstehen, nochmal lesen und wiedergeben können, am besten noch ein drittes Mal lesen. Alle gesehene Algorithmen können gefragt werden. Bitte nicht auswendig lernen etwas was jemand nicht versteht. Es wird ausdrücklich gebeten den Themen im Skript von Prof. Schnitger und/oder in den Lehrbüchern auch nachzulesen. Zum Stoff gehören noch die 2 extra Mathe-Blätter.

mA./mB. = mit Analyse/mit Beweis

oA./oB. = ohne Analyse/ohne Beweis

FÜR ALLE:

---

1 Einführung

- Grundbegriffe (reicht informal)
  - die Klasse $\mathcal{NP}$; Zeuge
  - die Klasse $\mathcal{NPO}$; Instanz, Lösung, Zielfunktion; α-approximativ (Lösung, Algorithmus), Laufzeit, Beispiele für Optimierungsprobleme/Entscheidungsprobleme (für Probleme s.a. Abschnitt 6)
  - Umgang mit schwierigen Problemen, Def: Approximationsfaktor einer Lösung/eines Algorithmus

- min-SCHEDULING
  - LIST Scheduling Algorithmus; Approximationsfaktor (mA.), Instanz für untere Schranke
  - Def: Polynomielles Approximationsschema; PTAS für min-SCHED-m (oA., Laufzeit grob)

- Klassen in $\mathcal{NPO}$
  - Def: Volles Polynomielles Approximationsschema (FPTAS); wann kann es nicht existieren; polynomiell beschränkte Probleme;
  - Def: $\mathcal{APX}$ Klassen;
  - Beispiele für Probleme in $\mathcal{FPTAS}$; $\mathcal{PTAS} \setminus \mathcal{FPTAS}$; $\mathcal{APX} \setminus \mathcal{PTAS}$, n-$\mathcal{APX} \setminus \mathcal{APX}$ (s.a. hier, Abschnitt 6)

2 Greedy Algorithmen und Heuristiken

- einfache greedy Algorithmen: für ungewichtetes VERTEX COVER (2-approximativ, mA.);
- grobe Definition von Greedy Algorithmen; priority Algorithmen
- Optimale greedy Algorithmen: INTERVALL SCHEDULING (mA.); Huffman Code (oA.); Kruskal für min-SPANNBAUM (oA.);
- Das (metrische) TSP
  - Def: Metrik, das metrische TSP;
  - die Spannbaum-Heuristik; die Nearest-Neighbor, Nearest-Insertion und Farthest-Insertion Heuristiken;
  - Euler Tour; Christofides' Algorithmus (mA.);
  - das allgemeine TSP: Nicht-Approximierbarkeit;
- BIN PACKING
  - Greedy Strategien: online (NF(mA.), FF), offline (FFD (mA.), BFD)
  - kein PTAS (streng genommen) (mit Reduktion)
  - 'effiziente' optimale Bepackung für konstant viele verschiedene Gewichte (nur die grobe Schätzung); Def: Asymptotisches PTAS; APTAS für BIN PACKING (oA.);

3 Dynamische Programmierung

- gewichtetes INTERVALL SCHEDULING und Alg. mit Dyn. Prog.
- wichtigste Merkmale; Wann wird dyn. Prog. verwendet?
- RUCKSACK
  - Dyn. Prog. Algorithmus für ganzzahlige Gewichte; Def: Pseudopolynomiell
  - FPTAS (Skalieren und Runden); der Skalierungsfaktor mit Begründung;
- Dyn. Prog. auf Bäumen: gewichtetes VERTEX COVER

4 Lokale Suche

- Cluster Probleme
  - k-CENTER; greedy Alg. (oA.); DOMINATING SET; Nichtapproximierbarkeit von k-CENTER;
  - k-MEDIAN; 5-approx. Alg. mit lokaler Suche

- Definition: lokale Suche; Nachbarschaften; k-Flip-Nachbarschaften; lokal optimale Lösung; Vergleich mit greedy Algorithmen; lok. Suche für VERTEX COVER (Bsp: Sterngraph)

- Wie gut approximiert ein lokales Optimum?

  – Spannbaumprobleme; Nachbarschaften und Anzahl der Nachbarn; max-Leaf-SPANNING-TREE; 5-approximativer Alg mit lok. Suche; (mA; Analyse der LAUFZEIT und der 10-Approximation durch die Teilmengungen); jeder Baum hat mehr Blätter als Knoten mit Grad mindestens 3 (oB.);

- Komplexität der Berechnung lokaler Optima

  – polynomielle Suchprobleme ($\mathcal{PLS}$); PLS-Reduktion; PLS-Vollständigkeit; Beispiele: min-BALANCED-CUT mit 2-Flip und TSP mit 2k-Flip

- Approximative lokale Optima

  – FACILITY LOCATION; 3-approx. lok. Suche; nicht besser als 3-approximativ (mB.); effiziente $3+\epsilon$-approximative lokale Suche für FACILITY LOCATION (oA.)

- Verschlechterungen während der Suche

  – Kernighan-Lin Alg. für min-BALANCED-CUT (Einfrieren); Metropolis Alg. und simulated Annealing; Bsp: Vertex Cover auf Sterngraph

5 Lineare Programmierung

- LP in allgemeiner Form; Beispiele: Produktionsplanung, Vertex cover, Matching

- kanonische Form;

  – Äquivalenz mit allgemeiner Form
  – geometrische Grundlagen: Skalarprodukt; Hyperebene; Halbraum; Lösungspolyeder; Konvexität; (Un)lösbarkeit; optimale Lösung; (graphische) Optimierung in 2D und 3D;
  – 2 Definitionen für Ecken; Existenz mind. einer Ecke; (endliches) Minimum wird in mind. einer Ecke angenommen (oB.); benachbarte Ecken; entartete Ecken

- LP in Standardform

  – Äquivalenz mit kanonischer Form; Dimension und Darstellung des Lösungspolyeders in Standardform
  – der Simplex Algorithmus: Grobstruktur; der Begriff Pivot-Strategie; Laufzeit(grob); Geschichte; Namen der Polinomialzeit-Algorithmen;

- LP und Approximation
  - Definition und Formulierung als IP und LP Relaxierung von: min-VERTEX COVER, max-MATCHING, max-INDEPENDENT SET, min-SET COVER, max-SAT; Greedy Algorithmus für SET COVER, Approximationsfaktor (oB.)
  - max-MATCHING: Inzidenzmatrix vollständig unimodular, Def: vollständig unimodulare Matrizen, ganzzahligkeit der Ecken;
  - Deterministisches Runden: VERTEX COVER (mA.), SET COVER (oA.);
  - Randomisiertes Runden: SET COVER, das max erwartete Gewicht der Lösung, Erwarteter Approximationsfaktor (oA.); 4 Algorithmen für max-SAT (Analyse von Alg. 2 von S. LP55); Erw. Anzahl erfüllter Klauseln (LP55.)
  - Def: Integralitätslücke, Beispiel: VERTEX COVER, INDEPENDENT SET, max-SAT

6 (Nicht-)Approximierbarkeit

Def: $f(n)$-$\mathcal{APX}$ Beispiele in allen (Differenzen von) Klassen kennen, inkl. in log-$\mathcal{APX}$ \ $\mathcal{APX}$, und in $\mathcal{NPO} \setminus 2^n$-$\mathcal{APX}$, bzw. die einzelnen Probleme mit entsprechenden Algorithmen sowieso; die β Werte werden nicht gefragt:

Problemklassen gemäß Approximierbarkeit in Polynomialzeit Unter der Annahme $\mathcal{P} \neq \mathcal{NP}$:

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

n-$\mathcal{APX}$ max-CLIQUE und max-INDEPENDENT-SET sind nicht polynomiell approximierbar mit Faktor $\leq n^{1-\epsilon}$

(diese Probleme lassen sich leicht ineinander transformieren)

log-$\mathcal{APX}$ (sehr wahrscheinlich) min-SET-COVER ist nicht polynomiell approximierbar mit Faktor $\leq (1-\varepsilon)\ln n$ (hier ist n die Anzahl der Elemente)

$\Rightarrow$ der Greedy Algorithmus ist optimal!

$\mathcal{APX}$ Es ist jeweils eine $\beta > 1$ bekannt, so dass die folgenden nicht polynomiell β-approximierbar sind: min-VERTEX-COVER ($\beta \approx 1.36$; kein Alg. mit < 2 Approx ist bekannt); max-3-SAT ($\beta = 8/7 - \varepsilon$) und das metrische TSP ($\beta = 220/219$), k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, und SHORTEST COMMON SUPERSTRING

$\Rightarrow$ sie alle besitzen kein PTAS.

Diese enthalten weiter die Approximations-Klassen:

$\mathcal{PTAS}$ min-SCHEDULING, euklidsches TSP, BIN-PACKING (asymptotisch)

$\mathcal{FPTAS}$ min-SCHEDULING-m, RUCKSACK

$\mathcal{PO}$ LINEARE PROGRAMMIERUNG, max-MATCHING, min-SPANNBAUM und seine verallgemeinerung: max-unabh.-Mengen in MATROIDEN, KÜRZESTE WEGE

NUR FÜR 8 ODER 10 CP:

– ROUTING mit Pfad-Zerlegungen (oA.);

7 LP Dualität

– das Duale eines LP in Standardform; (LPD6. NEIN); das Duale eines allgemeinen LP; Bestimmung des dualen Problems nicht auswendig aber mit Hilfe der Ungleichungen; das Duale des dualen LP

– der schwache Dualitätssatz; Beweis für die allgemeine Form; zwei Korollare

– starke Dualität; Illustration mit minmax (grob); Zusammenfassung;

– komplementäre Slackness (nur mit Hilfe der Ungleichungen);

– Primal-Duale Algorithmen: Merkmale, Grobstruktur; P-D Algorithmus für SET COVER (oA.); P-D-Algorithmus für SHORTEST s-t-PATH (oA.), (jeweils mit IP/LP-Formulierung und Interpretation des Primalen und Dualen Problems)

8 SHORTEST COMMON SUPERSTRING (Greedy)

– Grundbegriffe; $S^* = S(\pi)$; Länge des $S(\pi)$

– Der Greedy-Superstring Algorithmus

– Overlap-Graph; maximale Kreis-Zerlegung; Greedy Alg. (oA.), Monge-Ungleichung (mA.); Zyklen, minimale Zyklus- Überdeckung; ihre Länge als Schranke für $|S^*|$ (oB.);

– Alg. für Shortest Common Superstring, Approx. Faktor (oA.); Overlap-Lemma (oB.);

9 Matroide (Greedy)

– Def: monotones Teilmengensystem; Maximalitätseigenschaft; Ergänzungseigenschaft; Illustration durch Beispiele: Graph-Matroid und Matrix-Matroid (oB.);

– der greedy Alg. für monotone Teilmengensysteme; Def: Matroide; Äquivalenz der 3 Eigenschaften ($a \Leftrightarrow b$ mB., $b \Leftrightarrow c$ oB.) (Bedeutung von Matroiden bzgl. Greedy Algorithmen verstehen)

– Austauschbarkeit; kreisfreie Kantenmengen sind austauschbar (mB.); MATROID SCHEDULING; ausführbare Aufgabenmengen sind austauschbar (mB.);

– Def: k-Matroide; k-Maximalitätseigenschaft; k-Ergänzungseigenschaft; Illustration: max-MATCHING erfüllt diese für k=2 (Beweis nur für Approx. Faktor von Greedy);

– k-Austauschbarkeit; von max-MATCHING; von max-TSP (mB.); Durchschnitt von k Matroiden ist k-Matroid, Beispiel: Matching (oB.)

NUR FÜR 10 CP:

10 Arora's PTAS für das euklidsche TSP (Dynamische Programmierung)

– Jeder Binärbaum hat ein Blatt mehr als innere Knoten; jeder quadrrer Baum hat etwa dreimal mehr Bltter als innere Knoten;

– Vorbereitungen (Verschieben, Skalieren, Runden); ein Baum für Dyn. Prog.; Trennlinien und Türen; legale Rundreisen; Besuchsmuster

– zufällige Gitterwahl; erwartete Approximation (oA.); Approximation mit Wahrscheinlichkeit 1/2 (oA.); Laufzeit (grobe Schranke) (oA.), Derandomisierung;

11 Branch & Bound

- Grobstruktur von B&B Algorithmen (Branching Schritt; Branching Operator; B&B Baum; Bounding Schritt);
- das fraktionale RUCKSACK Problem; Greedy Algorithmus; untere Schranke; B&B Alg. für das (ganzzahlige) RUCKSACK (Code nicht auswendig, nur wichtigste Komponenten);
- B&B für Δ-TSP: untere Schranke mit 1-Bäumen, Definition der Held-Karp Schranke, Branching-Operator (alternativ zur Held-Karp Schranke: die Ellipsoid-Methode skizzieren können)
- Cutting Planes: Grobstruktur; Beispiel: für max-INDEPENDENT SET
- aus dem Section 9.10 Nature of Computation (Ausdruck abholbar) wird eine leichte Frage gefragt