Einführung in Semidefinite Programmierung

Daniel Borchmann

Sommerakademie Görlitz 2007

12. September 2007

Optimierungsaufgaben

Gegeben sei eine *Zielfunktion* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. Dann betrachten wir

$$\begin{aligned} \text{maximiere} \quad & f(x) \\ \text{so dass} \quad & f_i(x) = b_i \text{ für } i \in \{1, \ldots, m\} \\ & h_i(x) \geq 0 \text{ für } i \in \{1, \ldots, l\} \end{aligned}$$

mit den *Nebenbedingungen* gegeben durch die Funktionen $f_i, h_i : \mathbb{R} \longrightarrow \mathbb{R}$. Wir definieren

$$Z := \{x \mid f_i(x) = b_i \text{ für alle } i \in \{1, \ldots, m\} \text{ und } h_i(x) \geq 0 \text{ für alle } i \in \{1, \ldots, l\}\}$$

die *Menge der zulässigen Lösungen*.

Problem: Effiziente Lösung solcher Aufgaben ist im allgemeinen schwierig bis hin zu praktisch unmöglich.

Idee: Relaxation (Vereinfachung) des Problems,

- um obere Schranken für den optimalen Wert zu erhalten oder
- Näherungslösungen zu berechnen

meist durch Vergrößerung von Z.

Problem: Effiziente Lösung solcher Aufgaben ist im allgemeinen schwierig bis hin zu praktisch unmöglich.

Idee: Relaxation (Vereinfachung) des Problems,

- um obere Schranken für den optimalen Wert zu erhalten oder
- Näherungslösungen zu berechnen

meist durch Vergrößerung von Z.

Lineare Optimierung

1. Versuch:

Wir betrachten die Standardform[1] einer linearen Optimierungsaufgabe

$$\begin{aligned} \text{maximiere} \quad & c \cdot x \\ \text{so dass} \quad & x \in \mathbb{R}^n, Ax = b, x \geq 0 \end{aligned} \tag{LP}$$

Probleme dieser Form treten häufig auf, bekannte Lösungsmethoden sind

- Simplexverfahren
- Innere-Punkt Methoden
- ...

[1] hier mit Gleichungsnebenbedingungen

Lineare Optimierung

1. Versuch:

Wir betrachten die Standardform[1] einer linearen Optimierungsaufgabe

$$\begin{aligned} \text{maximiere} \quad & c \cdot x \\ \text{so dass} \quad & x \in \mathbb{R}^n, Ax = b, x \geq 0 \end{aligned} \tag{LP}$$

Probleme dieser Form treten häufig auf, bekannte Lösungsmethoden sind

- Simplexverfahren
- Innere-Punkt Methoden
- ...

[1] hier mit Gleichungsnebenbedingungen

Lineare Optimierung

1. Versuch:

Wir betrachten die Standardform[1] einer linearen Optimierungsaufgabe

$$\begin{aligned} \text{maximiere} \quad & c \cdot x \\ \text{so dass} \quad & x \in \mathbb{R}^n, Ax = b, x \geq 0 \end{aligned} \tag{LP}$$

Probleme dieser Form treten häufig auf, bekannte Lösungsmethoden sind

- Simplexverfahren
- Innere-Punkt Methoden
- ...

[1] hier mit Gleichungsnebenbedingungen

Relaxation bestimmter Probleme führen unter anderem auf lineare Optimierungsaufgaben, so zum Beispiel

- MAX-CUT, die Berechnung eines maximalen Schnittes in einem gewichteten Graphen,
- MAX-BISECTION, eine Variante von MAX-CUT mit gleichmächtigen Knotenmengen,
- MAX-2SAT, die Berechnung einer optimalen Belegung einer Klausel in CNF mit jeweils genau zwei Variablen.

Diese Relaxationen liefern aber zu wenig Informationen!

Verallgemeinerung der Linearen Optimierung auf Semidefinite Optimierung (Semidefinite Programming, SDP)

Relaxation bestimmter Probleme führen unter anderem auf lineare Optimierungsaufgaben, so zum Beispiel

- MAX-CUT, die Berechnung eines maximalen Schnittes in einem gewichteten Graphen,
- MAX-BISECTION, eine Variante von MAX-CUT mit gleichmächtigen Knotenmengen,
- MAX-2SAT, die Berechnung einer optimalen Belegung einer Klausel in CNF mit jeweils genau zwei Variablen.

Diese Relaxationen liefern aber zu wenig Informationen!

Verallgemeinerung der Linearen Optimierung auf Semidefinite Optimierung (Semidefinite Programming, SDP)

Relaxation bestimmter Probleme führen unter anderem auf lineare Optimierungsaufgaben, so zum Beispiel

- MAX-CUT, die Berechnung eines maximalen Schnittes in einem gewichteten Graphen,
- MAX-BISECTION, eine Variante von MAX-CUT mit gleichmächtigen Knotenmengen,
- MAX-2SAT, die Berechnung einer optimalen Belegung einer Klausel in CNF mit jeweils genau zwei Variablen.

Diese Relaxationen liefern aber zu wenig Informationen!

Verallgemeinerung der Linearen Optimierung auf Semidefinite Optimierung (Semidefinite Programming, SDP)

Relaxation bestimmter Probleme führen unter anderem auf lineare Optimierungsaufgaben, so zum Beispiel

- MAX-CUT, die Berechnung eines maximalen Schnittes in einem gewichteten Graphen,
- MAX-BISECTION, eine Variante von MAX-CUT mit gleichmächtigen Knotenmengen,
- MAX-2SAT, die Berechnung einer optimalen Belegung einer Klausel in CNF mit jeweils genau zwei Variablen.

Diese Relaxationen liefern aber zu wenig Informationen!

Verallgemeinerung der Linearen Optimierung auf Semidefinite Optimierung (Semidefinite Programming, SDP)

Relaxation bestimmter Probleme führen unter anderem auf lineare Optimierungsaufgaben, so zum Beispiel

- MAX-CUT, die Berechnung eines maximalen Schnittes in einem gewichteten Graphen,
- MAX-BISECTION, eine Variante von MAX-CUT mit gleichmächtigen Knotenmengen,
- MAX-2SAT, die Berechnung einer optimalen Belegung einer Klausel in CNF mit jeweils genau zwei Variablen.

Diese Relaxationen liefern aber zu wenig Informationen!

Verallgemeinerung der Linearen Optimierung auf Semidefinite Optimierung (Semidefinite Programming, SDP)

Definition

Eine Matrix $X \in \mathbb{R}^{n \times n}$ heißt *positiv semidefinit*, wenn für alle $x \in \mathbb{R}^n$ stets $x^T X x \geq 0$ gilt.

Theorem

Sei $X \in \mathbb{R}^{n \times n}$ symmetrisch. Dann sind folgende Aussagen äquivalent:

a) *X ist positiv semidefinit*

b) *X hat keine negativen Eigenwerte*

c) *Für jedes $y \in \mathbb{R}^n$ gilt $y^T X y \geq 0$*

d) *Es existiert eine Matrix $C \in \mathbb{R}^{n \times n}$ mit $X = CC^T$.*

Definition

Eine Matrix $X \in \mathbb{R}^{n \times n}$ heißt *positiv semidefinit*, wenn für alle $x \in \mathbb{R}^n$ stets $x^T X x \geq 0$ gilt.

Theorem

Sei $X \in \mathbb{R}^{n \times n}$ symmetrisch. Dann sind folgende Aussagen äquivalent:

a) *X ist positiv semidefinit*

b) *X hat keine negativen Eigenwerte*

c) *Für jedes $y \in \mathbb{R}^n$ gilt $y^T X y \geq 0$*

d) *Es existiert eine Matrix $C \in \mathbb{R}^{n \times n}$ mit $X = CC^T$.*

Semidefinite Programmierung

2. Versuch:

Statt der Optimierungsaufgabe (LP) betrachten wir nun

$$\begin{aligned} \text{maximiere} \quad & \operatorname{tr}(WX) \\ \text{so dass} \quad & \operatorname{tr}(A_j X) = b_j \text{ für } j \in \{1, \ldots, m\}, \\ & X \succeq 0 \end{aligned} \tag{SDP}$$

mit symmetrischen Matrizen W, A_j, X und

$$\operatorname{tr}(X) = \sum_{i=1}^{n} X_{ii}$$

der *Spur* von X.

Semidefinite Programmierung

2. Versuch:

Statt der Optimierungsaufgabe (LP) betrachten wir nun

$$\begin{aligned} \text{maximiere} \quad & \operatorname{tr}(WX) \\ \text{so dass} \quad & \operatorname{tr}(A_j X) = b_j \text{ für } j \in \{1, \ldots, m\}, \\ & X \succeq 0 \end{aligned} \tag{SDP}$$

mit symmetrischen Matrizen W, A_j, X und

$$\operatorname{tr}(X) = \sum_{i=1}^{n} X_{ii}$$

der *Spur* von X.

Vorteile

- Lösung von (SDP) ist immernoch „praktisch schnell“ (z.B. mit Inneren-Punkt Methoden):
  Semidefinite Optimierungsaufgaben sind innerhalb einer vorgegeben Fehlertoleranz $\varepsilon > 0$ polynomiell in n und $\log \frac{1}{\varepsilon}$ lösbar
- Relaxationen zu Problemen der Form (SDP) liefern im allgemeinen bessere Schranken als lineare Relaxationen
- Approximationsalgorithmen auf Basis von (SDP) liefern meist bessere Ergebnisse

Vorteile

- Lösung von (SDP) ist immernoch „praktisch schnell“ (z.B. mit Inneren-Punkt Methoden):
  Semidefinite Optimierungsaufgaben sind innerhalb einer vorgegeben Fehlertoleranz $\varepsilon > 0$ polynomiell in n und $\log \frac{1}{\varepsilon}$ lösbar
- Relaxationen zu Problemen der Form (SDP) liefern im allgemeinen bessere Schranken als lineare Relaxationen
- Approximationsalgorithmen auf Basis von (SDP) liefern meist bessere Ergebnisse

Vorteile

- Lösung von (SDP) ist immernoch „praktisch schnell“ (z.B. mit Inneren-Punkt Methoden):
  Semidefinite Optimierungsaufgaben sind innerhalb einer vorgegeben Fehlertoleranz $\varepsilon > 0$ polynomiell in n und $\log \frac{1}{\varepsilon}$ lösbar
- Relaxationen zu Problemen der Form (SDP) liefern im allgemeinen bessere Schranken als lineare Relaxationen
- Approximationsalgorithmen auf Basis von (SDP) liefern meist bessere Ergebnisse

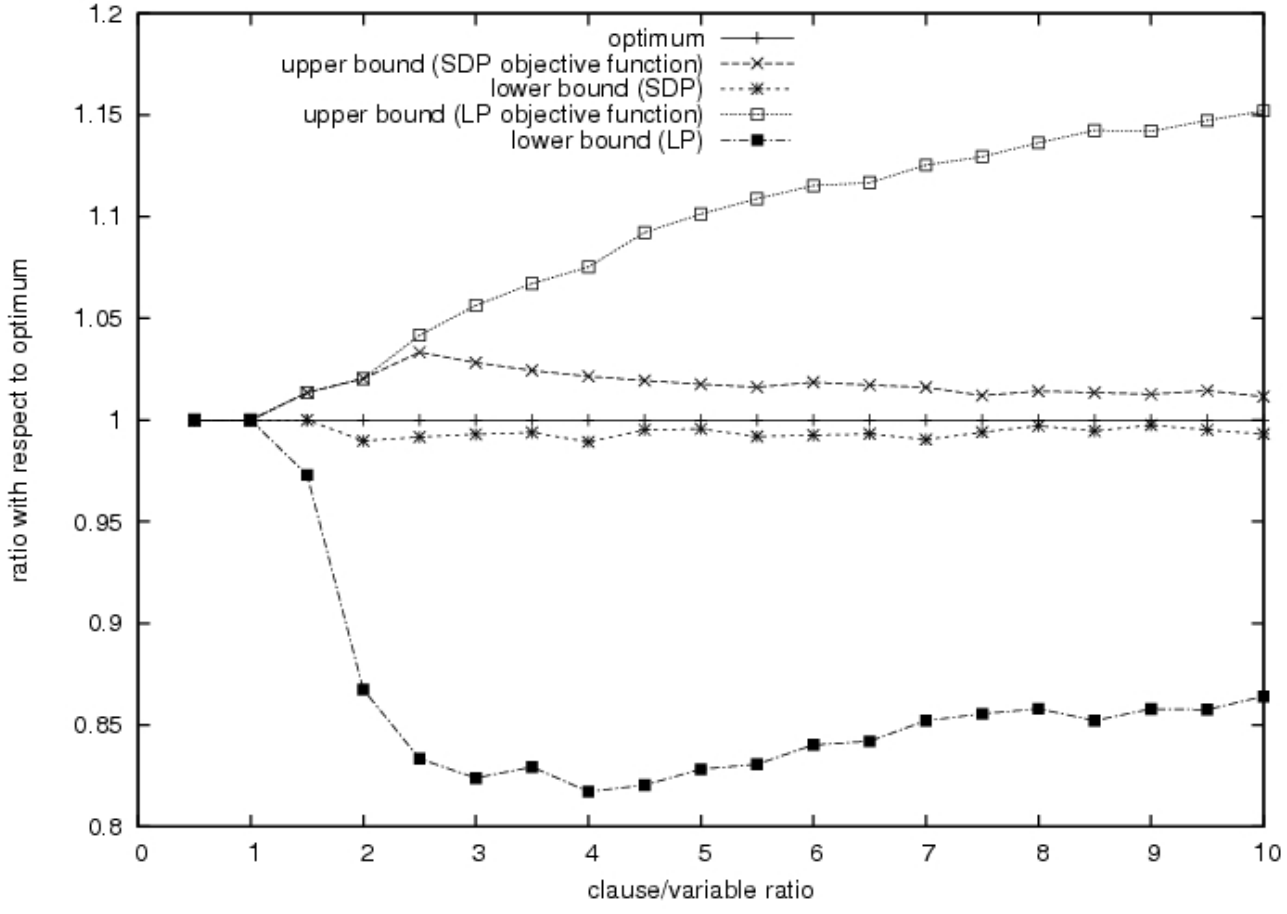

Abbildung: Vergleich von LP und SDP bei MAX-SAT, aus [GHL06]

Beispiel: MAX-CUT

Gegeben sei ein nichtnegativ gewichteter Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E, $|E| \geq 1$. Ohne Einschränkung sei G vollständig[2]. Es bezeichne weiterhin w_{ij} das Gewicht der Kante $(i, j) \in E$.

Es sei das *Gewicht w das Schnittes* $(S, V \setminus S)$ gegeben durch

$$w(S, V \setminus S) := \sum_{i \in S, j \notin S} w_{ij}.$$

[2] nicht vorhanden Kanten bekommen das Gewicht 0

Beispiel: MAX-CUT

Gegeben sei ein nichtnegativ gewichteter Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E, $|E| \geq 1$. Ohne Einschränkung sei G vollständig[2]. Es bezeichne weiterhin w_{ij} das Gewicht der Kante $(i, j) \in E$.

Es sei das *Gewicht w das Schnittes* $(S, V \setminus S)$ gegeben durch

$$w(S, V \setminus S) := \sum_{i \in S, j \notin S} w_{ij}.$$

[2] nicht vorhanden Kanten bekommen das Gewicht 0

Dann ist das Problem MAX-CUT gegeben durch

$$\begin{aligned} \text{maximiere} \quad & w(S, V \setminus S) \\ \text{so dass} \quad & S \subseteq V \end{aligned}$$

Theorem

MAX-CUT $\in \mathcal{NPC}$.

Dann ist das Problem MAX-CUT gegeben durch

$$\begin{aligned} \text{maximiere} \quad & w(S, V \setminus S) \\ \text{so dass} \quad & S \subseteq V \end{aligned}$$

Theorem

MAX-CUT $\in \mathcal{NPC}$.

MAX-CUT als SDP

Idee: Assoziere mit einem Knoten $i \in V$ eine Variable $y_i \in \{-1, 1\}$, so dass $i \in S \iff y_i = 1$ gilt.

Dann ist mit $S = \{y_i \mid y_i = 1\}$ das Gewicht eines Schnittes gegeben durch

$$w(S, V \setminus S) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j).$$

MAX-CUT als SDP

Idee: Assoziere mit einem Knoten $i \in V$ eine Variable $y_i \in \{-1, 1\}$, so dass $i \in S \iff y_i = 1$ gilt.

Dann ist mit $S = \{y_i \mid y_i = 1\}$ das Gewicht eines Schnittes gegeben durch

$$w(S, V \setminus S) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j).$$

Dann ergibt sich MAX-CUT zu

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - y_i y_j) \\ \text{so dass} \quad & y_i \in \{-1, 1\} \text{ für alle } i \in V \end{aligned} \qquad \text{(MAX-CUT)}$$

Beobachtung: Die Variablen $y_i y_j$ treten als Produkt auf, d.h die Matrix

$$Y = (y_i y_j)_{i,j=1}^{|V|}$$

ist positiv semidefinit mit $Y_{ii} = 1$.

Dies ergibt:

SDP (MAX-CUT)

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & Y_{ii} = 1 \text{ für alle } i \in V \\ & Y \succeq 0 \end{aligned} \tag{MAX-CUT-SDP}$$

Beobachtung: Die Variablen $y_i y_j$ treten als Produkt auf, d.h die Matrix

$$Y = (y_i y_j)_{i,j=1}^{|V|}$$

ist positiv semidefinit mit $Y_{ii} = 1$.

Dies ergibt:

SDP (MAX-CUT)

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & Y_{ii} = 1 \text{ für alle } i \in V \\ & Y \succeq 0 \end{aligned} \qquad \text{(MAX-CUT-SDP)}$$

Approximation von MAX-CUT

Bemerkung

Mit Hilfe von (MAX-CUT-SDP) wurden signifikante Verbesserungen in der Berechnung von Näherungslösungen von MAX-CUT erzielt:
Der erwartete Wert bei der Lösung von MAX-CUT ist mindestens $0.87856 \cdot$ Optimalwert. (1.1382; siehe z.B. [GW95])

Idee dazu: Fasse Variablen y_i in (MAX-CUT) als eindimensionale Vektoren mit euklidischer Norm 1 auf.

Mögliche Relaxation ist dann: Statt eindimensional auch mehrdimensional zulassen.

Approximation von MAX-CUT

Bemerkung

Mit Hilfe von (MAX-CUT-SDP) wurden signifikante Verbesserungen in der Berechnung von Näherungslösungen von MAX-CUT erzielt:
Der erwartete Wert bei der Lösung von MAX-CUT ist mindestens $0.87856 \cdot$ Optimalwert. (1.1382; siehe z.B. [GW95])

Idee dazu: Fasse Variablen y_i in (MAX-CUT) als eindimensionale Vektoren mit euklidischer Norm 1 auf.

Mögliche Relaxation ist dann: Statt eindimensional auch mehrdimensional zulassen.

Approximation von MAX-CUT

Bemerkung

Mit Hilfe von (MAX-CUT-SDP) wurden signifikante Verbesserungen in der Berechnung von Näherungslösungen von MAX-CUT erzielt:
Der erwartete Wert bei der Lösung von MAX-CUT ist mindestens $0.87856 \cdot$ Optimalwert. (1.1382; siehe z.B. [GW95])

Idee dazu: Fasse Variablen y_i in (MAX-CUT) als eindimensionale Vektoren mit euklidischer Norm 1 auf.

Mögliche Relaxation ist dann: Statt eindimensional auch mehrdimensional zulassen.

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i^T v_j) \\ \text{so dass} \quad & v_i \in S_{n-1} \text{ für alle } i \in V \end{aligned} \qquad \text{(APPROX)}$$

Setze dann

$$Y = \text{Gram}(\{v_1, \ldots, v_{|V|}\}) = (v_i^T v_j)_{i,j=1}^{|V|}$$

womit (APPROX) wieder auf (MAX-CUT-SDP) führt.

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i^T v_j) \\ \text{so dass} \quad & v_i \in S_{n-1} \text{ für alle } i \in V \end{aligned} \tag{APPROX}$$

Setze dann

$$Y = \text{Gram}(\{v_1, \ldots, v_{|V|}\}) = (v_i^T v_j)_{i,j=1}^{|V|}$$

womit (APPROX) wieder auf (MAX-CUT-SDP) führt.

Dies führt dann zu

Algorithmus (Approximationsalgorithmus für MAX-CUT, [GW95])

1) Löse (APPROX) als SDP mit Lösung $\{v_1, \ldots, v_{|V|}\}$,
2) wähle zufällig einen Vektor $r \in S_{n-1}$,
3) und setze $S = \{i \mid v_i^T r \geq 0\}$.

Bemerkung

Für den Erwartungswert E_w des Schnittes gilt

$$E_w \geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

unter den genannten Nebenbedingungen mit

$$\alpha = \min_{0<\theta\leq\pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta} > 0.87856.$$

Dies führt dann zu

Algorithmus (Approximationsalgorithmus für MAX-CUT, [GW95])

1) Löse (APPROX) als SDP mit Lösung $\{v_1, \ldots, v_{|V|}\}$,
2) wähle zufällig einen Vektor $r \in S_{n-1}$,
3) und setze $S = \{i \mid v_i^T r \geq 0\}$.

Bemerkung

Für den Erwartungswert E_w des Schnittes gilt

$$E_w \geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

unter den genannten Nebenbedingungen mit

$$\alpha = \min_{0<\theta\leq\pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta} > 0.87856.$$

Berechnung der garantierten erwarteten Güte

Lemma

Es gilt

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \arccos(v_i^T v_j)$$

Beweis

$$E_w = \sum_{i<j} w_{ij} \cdot \Pr[v_i \text{ und } v_j \text{ werden durch } r \text{ getrennt}]$$

$$= \sum_{i<j} w_{ij} \cdot \Pr[\operatorname{sgn}(v_i^T r) \neq \operatorname{sgn}(v_j^T r)]$$

$$\overset{\text{Bild}}{=} \sum_{i<j} w_{ij} \cdot \frac{1}{\pi} \arccos(v_i^T v_j)$$

Berechnung der garantierten erwarteten Güte

Lemma

Es gilt

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \arccos(v_i^T v_j)$$

Beweis

$$E_w = \sum_{i<j} w_{ij} \cdot \Pr[v_i \text{ und } v_j \text{ werden durch } r \text{ getrennt}]$$

$$= \sum_{i<j} w_{ij} \cdot \Pr[\operatorname{sgn}(v_i^T r) \neq \operatorname{sgn}(v_j^T r)]$$

$$\overset{\text{Bild}}{=} \sum_{i<j} w_{ij} \cdot \frac{1}{\pi} \arccos(v_i^T v_j)$$

Berechnung der garantierten erwarteten Güte

Lemma

Es gilt

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \arccos(v_i^T v_j)$$

Beweis

$$\begin{aligned} E_w &= \sum_{i<j} w_{ij} \cdot \Pr[v_i \text{ und } v_j \text{ werden durch } r \text{ getrennt}] \\ &= \sum_{i<j} w_{ij} \cdot \Pr[\operatorname{sgn}(v_i^T r) \neq \operatorname{sgn}(v_j^T r)] \\ &\overset{\text{Bild}}{=} \sum_{i<j} w_{ij} \cdot \frac{1}{\pi} \arccos(v_i^T v_j) \end{aligned}$$

Berechnung der garantierten erwarteten Güte

Lemma

Es gilt

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \arccos(v_i^T v_j)$$

Beweis

$$\begin{aligned} E_w &= \sum_{i<j} w_{ij} \cdot \Pr[v_i \text{ und } v_j \text{ werden durch } r \text{ getrennt}] \\ &= \sum_{i<j} w_{ij} \cdot \Pr[\operatorname{sgn}(v_i^T r) \neq \operatorname{sgn}(v_j^T r)] \\ &\overset{\text{Bild}}{=} \sum_{i<j} w_{ij} \cdot \frac{1}{\pi} \arccos(v_i^T v_j) \end{aligned}$$

Lemma

Es ist für $-1 \leq y \leq 1$

$$\frac{1}{\pi}\arccos(y) \geq \alpha \cdot \frac{1}{2}(1-y)$$

mit

$$\alpha = \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} > 0.87856$$

Beweis.

$$\frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \geq \min_{-1\leq y<1} \frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \overset{\cos\theta = y}{=} \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta}$$

□

Lemma

Es ist für $-1 \leq y \leq 1$

$$\frac{1}{\pi}\arccos(y) \geq \alpha \cdot \frac{1}{2}(1-y)$$

mit

$$\alpha = \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} > 0.87856$$

Beweis.

$$\frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \geq \min_{-1\leq y<1} \frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \overset{\cos\theta = y}{=} \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta}$$

□

Lemma

Es ist für $-1 \leq y \leq 1$

$$\frac{1}{\pi}\arccos(y) \geq \alpha \cdot \frac{1}{2}(1-y)$$

mit

$$\alpha = \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta} > 0.87856$$

Beweis.

$$\frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \geq \min_{-1\leq y<1} \frac{\frac{1}{\pi}\arccos(y)}{\frac{1}{2}(1-y)} \overset{\cos\theta = y}{=} \min_{0<\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-\cos\theta}$$

□

Zusammen gilt nun

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \cdot \arccos(v_i^T v_j)$$

$$\geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

Bemerkung

- Die Berechnungsungenauigkeit bei Lösung von (APPROX) kann in den Approximationsfaktor verschoben werden.
- Die Güte von 1.1382 ist die bislang beste, bekannte Approximationsgüte (nach [Wa06])
- Verbesserung auf $0.942 \cdot$ Optimalwert (1.0615) ist in $\mathcal{NPC}$ (siehe [Has97])

†

Zusammen gilt nun

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \cdot \arccos(v_i^T v_j)$$

$$\geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

Bemerkung

- Die Berechnungsungenauigkeit bei Lösung von (APPROX) kann in den Approximationsfaktor verschoben werden.
- Die Güte von 1.1382 ist die bislang beste, bekannte Approximationsgüte (nach [Wa06])
- Verbesserung auf $0.942 \cdot$ Optimalwert (1.0615) ist in $\mathcal{NPC}$ (siehe [Has97])

†

Zusammen gilt nun

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \cdot \arccos(v_i^T v_j)$$
$$\geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

Bemerkung

- Die Berechnungsungenauigkeit bei Lösung von (APPROX) kann in den Approximationsfaktor verschoben werden.
- Die Güte von 1.1382 ist die bislang beste, bekannte Approximationsgüte (nach [Wa06])
- Verbesserung auf $0.942 \cdot$ Optimalwert (1.0615) ist in $\mathcal{NPC}$ (siehe [Has97])

†

Zusammen gilt nun

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \cdot \arccos(v_i^T v_j)$$
$$\geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

Bemerkung

- Die Berechnungsungenauigkeit bei Lösung von (APPROX) kann in den Approximationsfaktor verschoben werden.
- Die Güte von 1.1382 ist die bislang beste, bekannte Approximationsgüte (nach [Wa06])
- Verbesserung auf $0.942 \cdot$ Optimalwert (1.0615) ist in $\mathcal{NPC}$ (siehe [Has97])

†

Zusammen gilt nun

$$E_w = \frac{1}{\pi} \sum_{i<j} w_{ij} \cdot \arccos(v_i^T v_j)$$
$$\geq \alpha \cdot \frac{1}{2} \sum_{i<j} w_{ij}(1 - v_i^T v_j) = \alpha \cdot \mathrm{OPT}$$

Bemerkung

- Die Berechnungsungenauigkeit bei Lösung von (APPROX) kann in den Approximationsfaktor verschoben werden.
- Die Güte von 1.1382 ist die bislang beste, bekannte Approximationsgüte (nach [Wa06])
- Verbesserung auf $0.942 \cdot$ Optimalwert (1.0615) ist in $\mathcal{NPC}$ (siehe [Has97])

†

Beispiel: MAX-BISECTION

Wir betrachten MAX-CUT mit der Zusatzbedingung, dass $|E| \in 2\mathbb{N}$ und dass beide Knotenmengen gleichmächtig sein sollen:

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \\ \text{so dass} \quad & y_i \in \{-1, 1\} \text{ für alle } i \in V \\ & \sum_{i=1}^{|V|} y_i = 0 \end{aligned}$$

MAX-BISECTION als SDP

Beobachtung: Schreibe $\sum_{i=1}^{|V|} y_i = 0$ als $e^T y = 0$ mit $e = (1)_{i=1}^{|V|} \in \mathbb{R}^{|V|}$ und $y = (y_i)_{i=1}^{|V|} \in \mathbb{R}^{|V|}$. Weiterhin setzen wir wieder $Y = (y_i y_j)_{i,j=1}^{|V|}$.

Statt der Bedingung $e^T y = 0$ setzen wir dann

$$\operatorname{tr}(ee^T Y) = 0$$

Dies ergibt:

SDP (MAX-BISECTION)

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & \operatorname{tr}(ee^T Y) = 0 \\ & Y_{ii} = 1 \text{ für alle } i \in V \\ & Y \succeq 0 \end{aligned} \qquad \text{(MAX-BISEC)}$$

MAX-BISECTION als SDP

Beobachtung: Schreibe $\sum_{i=1}^{|V|} y_i = 0$ als $e^T y = 0$ mit $e = (1)_{i=1}^{|V|} \in \mathbb{R}^{|V|}$ und $y = (y_i)_{i=1}^{|V|} \in \mathbb{R}^{|V|}$. Weiterhin setzen wir wieder $Y = (y_i y_j)_{i,j=1}^{|V|}$.

Statt der Bedingung $e^T y = 0$ setzen wir dann

$$\operatorname{tr}(ee^T Y) = 0$$

Dies ergibt:

SDP (MAX-BISECTION)

$$\begin{aligned} \text{maximiere} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & \operatorname{tr}(ee^T Y) = 0 \\ & Y_{ii} = 1 \text{ für alle } i \in V \\ & Y \succeq 0 \end{aligned} \tag{MAX-BISEC}$$

Algorithmus (MAX-BISECTION nach Frieze/Jerrum, [Ye99])

1) Löse die Relaxation (MAX-BISEC) und bestimme den Schnitt S wie bei MAX-CUT
2) falls die erhaltenen Mengen nicht gleichmächtig sind, dann tausche solange „leichte“ Knoten aus der größeren Menge in die kleiner Menge, bis dies erfüllt ist

Bemerkung

- Der oben angegebene Algorithmus hat einen erwarteten Wert von $0.651 \cdot$ Optimalwert (1.536)
- Die Relaxation (MAX-BISEC) kann genutzt werden, um Näherungsalgorithmen zu konstruieren, deren erwarteter Wert größer oder gleich $0.699 \cdot$ Optimalwert ist. (1.4306, siehe [Ye99])

†

Algorithmus (MAX-BISECTION nach Frieze/Jerrum, [Ye99])

1) Löse die Relaxation (MAX-BISEC) und bestimme den Schnitt S wie bei MAX-CUT
2) falls die erhaltenen Mengen nicht gleichmächtig sind, dann tausche solange „leichte“ Knoten aus der größeren Menge in die kleiner Menge, bis dies erfüllt ist

Bemerkung

- Der oben angegebene Algorithmus hat einen erwarteten Wert von $0.651 \cdot$ Optimalwert (1.536)
- Die Relaxation (MAX-BISEC) kann genutzt werden, um Näherungsalgorithmen zu konstruieren, deren erwarteter Wert größer oder gleich $0.699 \cdot$ Optimalwert ist. (1.4306, siehe [Ye99])

†

Beispiel: MAX-2SAT

Gegeben sei $n \in \mathbb{N}$ und eine Menge

$$C = \{C_j \mid C_j = x_{j_1}^{\alpha_{j_1}} \vee x_{j_2}^{\alpha_{j_2}}, j \in \{1, \ldots, n\}, \alpha_{j_1}, \alpha_{j_2} \in \{-1, 1\}\}$$

von Klauseln.

Dann ist MAX-2SAT die Maximierungsaufgabe

> *Finde eine Belegung der Variablen x_i derart, dass die Anzahl der gültigen Klauseln C_j maximal wird.*

Beispiel: MAX-2SAT

Gegeben sei $n \in \mathbb{N}$ und eine Menge

$$C = \{C_j \mid C_j = x_{j_1}^{\alpha_{j_1}} \vee x_{j_2}^{\alpha_{j_2}}, j \in \{1, \ldots, n\}, \alpha_{j_1}, \alpha_{j_2} \in \{-1, 1\}\}$$

von Klauseln.

Dann ist MAX-2SAT die Maximierungsaufgabe

> *Finde eine Belegung der Variablen x_i derart, dass die Anzahl der gültigen Klauseln C_j maximal wird.*

MAX-2SAT als SDP

Vorbetrachtungen

- assoziere mit jeder Variablen $x_i \in \{\texttt{true}, \texttt{false}\}$ eine Variable $y_i \in \{-1, 1\}$
- Einführung einer weiteren Variablen $y_0 \in \{-1, 1\}$

Wir setzen x_i auf `true` dann und nur dann, wenn $y_0 = y_i$ gilt.

MAX-2SAT als SDP

Vorbetrachtungen

- assoziere mit jeder Variablen $x_i \in \{\texttt{true}, \texttt{false}\}$ eine Variable $y_i \in \{-1, 1\}$
- Einführung einer weiteren Variablen $y_0 \in \{-1, 1\}$

Wir setzen x_i auf `true` dann und nur dann, wenn $y_0 = y_i$ gilt.

MAX-2SAT als SDP

Vorbetrachtungen

- assoziere mit jeder Variablen $x_i \in \{\texttt{true}, \texttt{false}\}$ eine Variable $y_i \in \{-1, 1\}$
- Einführung einer weiteren Variablen $y_0 \in \{-1, 1\}$

Wir setzen x_i auf `true` dann und nur dann, wenn $y_0 = y_i$ gilt.

Ziel: Definiere Funktion $v\colon C \longrightarrow \mathbb{N}$, welche die wahren Klauseln „zählt“.

Es bezeichne N die Anzahl der in C vorkommenden Variablen.

Für die Variablen x_i mit $i \in \{0, \ldots, N\}$ definiere

$$v(x_i) = \begin{cases} 0 & \text{falls } x_i = \texttt{false} \\ 1 & \text{falls } x_i = \texttt{true} \end{cases}$$

oder mit Hilfe der Variablen y_i:

$$v(x_i) = \frac{1 + y_0 y_i}{2}.$$

Entsprechend setze

$$v(\overline{x_i}) = \frac{1 - y_0 y_i}{2}.$$

Ziel: Definiere Funktion $v\colon C \longrightarrow \mathbb{N}$, welche die wahren Klauseln „zählt“.

Es bezeichne N die Anzahl der in C vorkommenden Variablen.

Für die Variablen x_i mit $i \in \{0, \ldots, N\}$ definiere

$$v(x_i) = \begin{cases} 0 & \text{falls } x_i = \texttt{false} \\ 1 & \text{falls } x_i = \texttt{true} \end{cases}$$

oder mit Hilfe der Variablen y_i:

$$v(x_i) = \frac{1 + y_0 y_i}{2}.$$

Entsprechend setze

$$v(\overline{x_i}) = \frac{1 - y_0 y_i}{2}.$$

Ziel: Definiere Funktion $v\colon C \longrightarrow \mathbb{N}$, welche die wahren Klauseln „zählt“.

Es bezeichne N die Anzahl der in C vorkommenden Variablen.

Für die Variablen x_i mit $i \in \{0, \ldots, N\}$ definiere

$$v(x_i) = \begin{cases} 0 & \text{falls } x_i = \texttt{false} \\ 1 & \text{falls } x_i = \texttt{true} \end{cases}$$

oder mit Hilfe der Variablen y_i:

$$v(x_i) = \frac{1 + y_0 y_i}{2}.$$

Entsprechend setze

$$v(\overline{x_i}) = \frac{1 - y_0 y_i}{2}.$$

Definiere nun v auf den Klauseln $x_i \vee x_k$:

$$\begin{aligned} v(x_i \vee x_k) &= 1 - (\overline{x_i} \wedge \overline{x_k}) \\ &= 1 - \frac{1 - y_0 y_i}{2} \frac{1 - y_0 y_k}{2} \\ &= \frac{1 + y_0 y_i}{4} + \frac{1 + y_0 y_k}{4} + \frac{1 - y_i y_k}{4} \end{aligned}$$

Damit nun MAX-2SAT als Optimierungsaufgabe

$$\begin{aligned} \text{maximiere} \quad & \sum_{c \in C} v(c) \\ \text{so dass} \quad & y_i \in \{-1, 1\} \text{ für alle } i \in \{0, \ldots, N\} \end{aligned} \tag{MAX-2SAT}$$

Es ist $\sum_{c \in C} v(c) = \sum_{i<j} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j)$.

Dann kann die Optimierungsaufgabe (MAX-2SAT) geschrieben werden als

$$\begin{aligned} \text{maximiere} \quad & \sum_{i<j} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j) \\ \text{so dass} \quad & y_i \in S_0 \text{ für alle } i \in \{0, \ldots, N\} \end{aligned}$$

Es ist $\sum_{c \in C} v(c) = \sum_{i<j} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j)$.

Dann kann die Optimierungsaufgabe (MAX-2SAT) geschrieben werden als

$$\begin{aligned} \text{maximiere} \quad & \sum_{i<j} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j) \\ \text{so dass} \quad & y_i \in S_0 \text{ für alle } i \in \{0, \ldots, N\} \end{aligned}$$

Schließlich wieder die Relaxation: Lasse statt S_0 auch S_{n-1} zu:

SDP (MAX-2SAT)

$$\begin{aligned} \text{maximiere} \quad & \sum_{i<j} a_{ij}(1 + Y_{ij}) + b_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & Y_{ii} = 1 \text{ für alle } i \in \{0, \ldots, N\}, \\ & Y \succeq 0 \end{aligned}$$

Bemerkung

Wir erhalten damit als Lösung $\{v_0, \ldots, v_{|V|}\}$, wählen wieder $r \in S_{n-1}$ zufällig und setzen alle Variablen x_i auf `true`, die die Gleichung $\operatorname{sgn} v_i^T r = \operatorname{sgn} v_0^T r$ erfüllen, den Rest auf `false`.

Bemerkung

Eine Verbesserung auf $0.931 \cdot$ Optimalwert (also einer Güte von 1.0741) ist möglich!

Schließlich wieder die Relaxation: Lasse statt S_0 auch S_{n-1} zu:

SDP (MAX-2SAT)

$$\begin{aligned} \text{maximiere} \quad & \sum_{i<j} a_{ij}(1 + Y_{ij}) + b_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & Y_{ii} = 1 \text{ für alle } i \in \{0, \ldots, N\}, \\ & Y \succeq 0 \end{aligned}$$

Bemerkung

Wir erhalten damit als Lösung $\{v_0, \ldots, v_{|V|}\}$, wählen wieder $r \in S_{n-1}$ zufällig und setzen alle Variablen x_i auf `true`, die die Gleichung $\operatorname{sgn} v_i^T r = \operatorname{sgn} v_0^T r$ erfüllen, den Rest auf `false`.

Bemerkung

Eine Verbesserung auf $0.931 \cdot$ Optimalwert (also einer Güte von 1.0741) ist möglich!

Schließlich wieder die Relaxation: Lasse statt S_0 auch S_{n-1} zu:

SDP (MAX-2SAT)

$$\begin{aligned} \text{maximiere} \quad & \sum_{i<j} a_{ij}(1 + Y_{ij}) + b_{ij}(1 - Y_{ij}) \\ \text{so dass} \quad & Y_{ii} = 1 \text{ für alle } i \in \{0, \ldots, N\}, \\ & Y \succeq 0 \end{aligned}$$

Bemerkung

Wir erhalten damit als Lösung $\{v_0, \ldots, v_{|V|}\}$, wählen wieder $r \in S_{n-1}$ zufällig und setzen alle Variablen x_i auf `true`, die die Gleichung $\operatorname{sgn} v_i^T r = \operatorname{sgn} v_0^T r$ erfüllen, den Rest auf `false`.

Bemerkung

Eine Verbesserung auf $0.931 \cdot$ Optimalwert (also einer Güte von 1.0741) ist möglich!

Danke

Literaturhinweise I

[Ali93] Farid Alizadeh
Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization
(online)

[BV04] Stephen Boyd und Lieven Vandenberghe
Convex Optimization
Cambridge University Press, 2004

[FJ95] Alan Frieze und Mark Jerrum
Improved Approximation Algorithms for MAX k-CUT and MAX-BISECTION
(online)

Literaturhinweise II

[GHL06] Carla P. Gomes et. al
The Power of Semidefinite Programming Relaxations for MAXSAT
In Proc. Conf. Intregration of AI/OR (CPAIOR06), 2006 (online)

[GW95] Michel X. Goemans und David P. Williamson
Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming
Proceedings of the 26th Annual ACM Symposium on Theory of Computing, (online)

[Has97] Johan Håstad
Some optimal inapproximability results
(online)

Literaturhinweise III

[helm] Christop Helmberg
Semidefinite Programming
zu finden unter http://www-user.tu-chemnitz.de/~helmberg/semidef.html

[Wa06] Rolf Wanka
Approximationsalgorithmen
Teubner Verlag, Wiesbaden, 2006

[Ye99] Yinyu Ye
A .699-Approimation Algorithm for Max-Bisection
(online)