Kapitel 1

Randomisierte Algorithmen

Einleitung

Definition: 1
Ein Algorithmus, der im Laufe seiner Ausführung gewisse Entscheidungen zufällig trifft, heisst randomisierter Algorithmus.

Beispiel: Bei der randomisierten Variante von Quick Sort wird das Element, nach dem in die Teilfolgen aufgeteilt wird, zufällig gewählt. ■

Motivation für randomisierte Algorithmen

- Für viele Probleme sind randomisierte Algorithmen schneller als deterministische Algorithmen.
- Typischerweise sind randomisierte Algorithmen einfacher zu beschreiben und zu implementieren als deterministische Algorithmen.

Man unterscheidet zwei Typen von randomisierten Algorithmen:

1. Las Vegas:

   Randomisierte Algorithmen, die immer ein korrektes Ergebnis liefern, gehören zu diesem Typ. In Abhängigkeit von den Wahlen, die zufällig getroffen werden, variiert die Laufzeit dieser Algorithmen. Man analysiert dann die Verteilung der Anzahlen der durchgeführten Rechnungsschritte.

2. Monte Carlo:

   Randomisierte Algorithmen, die manchmal auch ein falsches Ergebnis liefern, fallen unter diese Kategorie von Algorithmen. Man untersucht hier die Wahrscheinlichkeit, mit der das Ergebnis falsch ist. Für Entscheidungsprobleme, d.h. deren mögliches Ergebnis JA/NEIN ist, gibt es zwei Arten von Monte Carlo-Algorithmen:

   (a) beidseitiger Fehler
   Ein Monte Carlo-Algorithmus hat einen beiseitigen Fehler, wenn für die beiden möglichen Antworten JA/NEIN die Wahrscheinlichkeit für eine falsche Antwort grösser ist.

   (b) einseitiger Fehler
   Ein Monte Carlo-Algorithmus hat einen einseitigen Fehler, wenn die Wahrscheinlichkeit, dass die Antwort falsch ist, in einem der beiden Fälle JA/NEIN gleich Null ist, d.h. zum Beispiel, wenn das Ergebnis „JA“ ausgegeben wird, ist dies immer richtig, während wenn „NEIN“ ausgegeben wird, ist dies nur mit einer bestimmten Wahrscheinlichkeit korrekt.

Definition: 2

1. *Die Klasse $\mathcal{RP}$ (randomisiert polynomial) ist die Klasse der Entscheidungsprobleme Π, für die es einen polynomialen, randomisierten Algorithmus A gibt, so dass für alle Instanzen I von Π gilt:*

$$\begin{cases} I \in Y_\Pi \longrightarrow \text{Pr}(\text{A(I) ist „JA“}) \geq \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \text{Pr}(\text{A(I) ist „NEIN“}) = 0 \end{cases}$$

   Y_Π ist die Menge der sogenannten „JA-Beispiele“ von Π. Dabei entspricht Pr(A(I) ist „JA“) der Wahrscheinlichkeit, dass die Antwort, die A bei der Eingabe von I gibt, „JA“ ist. Ein $\mathcal{RP}$-Algorithmus ist also ein einseitiger Monte Carlo-Algorithmus.

2. *Die Klasse $\mathcal{PP}$ (probalistic polynomial) ist die Klasse der Entscheidungsprobleme Π, für die es einen polynomialen, Algorithmus A gibt, so dass für alle Instanzen I gilt:*

$$\begin{cases} I \in Y_\Pi \longrightarrow \text{Pr}(\text{A(I) ist „JA“}) > \frac{1}{2} \\ I \notin Y_\Pi \longrightarrow \text{Pr}(\text{A(I) ist „JA“}) < \frac{1}{2} \end{cases}$$

   Ein $\mathcal{PP}$-Algorithmus ist ein beidseitiger Monte Carlo-Algorithmus.

3. *Die Klasse $\mathcal{BPP}$ (bounded error PP) ist die Klasse der Entscheidungsprobleme Π, für die es einen polynomialen, Algorithmus A gibt so dass für alle Instanzen I gilt:*

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr(\text{A(I) ist „JA“}) \geq \frac{3}{4} \\ I \notin Y_\Pi \longrightarrow \Pr(\text{A(I) ist „JA“}) \leq \frac{1}{4} \end{cases}$$

Die probalistische Schranke kann zu $\frac{1}{2} + \frac{1}{p(n)}$ beziehungsweise $\frac{1}{2} - \frac{1}{p(n)}$ verschärft werden, wobei $p(n)$ ein Polynom in der Eingabegrösse n ist.

Grundlagen der Wahrscheinlichkeitstheorie I

Definition: 3

1. *Ein* Wahrscheinlichkeitsraum *ist ein Paar (Ω, Pr), wobei Ω eine Menge und Pr eine Abbildung*

$$Pr : \Omega \longrightarrow \mathbb{R}_0^+ \quad \text{ist mit} \quad \sum_{\omega \in \Omega} Pr(\omega) = 1.$$

2. *Eine Teilmenge aus Ω heisst* Ereignis *und Pr wird erweitert auf Ereignisse durch*

$$Pr(A) := \sum_{\omega \in \Omega} Pr(\omega)$$

3. *Die Elemente aus Ω heissen* Elementarereignisse.

4. *Falls Ω endlich ist und*

$$Pr(\omega) = \frac{1}{|\Omega|} \qquad \text{für alle } \omega \in \Omega$$

ist, so heisst Pr Gleichverteilung über Ω. *Im folgenden bezeichne*

$$\Omega^+ := \{\omega \in \Omega : Pr(\omega) > 0\}.$$

Beispiel:

1. „fairer Würfel“

   Sei $\Omega := \{1, 2, 3, 4, 5, 6\}$ und $Pr(d) = \frac{1}{6}$ für alle $d \in \Omega$. Dann ist

$$\Omega_{even} := \{2, 4, 6\} \text{ und damit folgt für } Pr(\Omega_{even}) = 3 \cdot \frac{1}{6} = \frac{1}{2} .$$

2. „Zwei unabhängige Würfel“

   Sei $\Omega := \{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}$. Die Mächtigkeit von Ω ist dann $|\Omega| = 36$. Sei

   $\Omega_= := \{(1,1),(2,2),(3,3),(4,4),(5,5),(6,6)\}$, dann gilt für diese:

$$\begin{aligned} Pr(\Omega_=) &= 6 \cdot \tfrac{1}{36} = \tfrac{1}{6} \text{ und für} \\ Pr(\Omega_{\neq}) &= 1 - \tfrac{1}{6} = \tfrac{5}{6} \end{aligned}$$

■

Definition: 4

1. *Seien A_1 und $A_2 \subseteq \Omega$ Ereignisse. Die* bedingte Wahrscheinlichkeit *von A_1 unter der Bedingung A_2 ist definiert als*

$$Pr(A_1|A_2) := \frac{Pr(A_1 \cap A_2)}{Pr(A_2)},$$

   wobei $Pr(A_2) > 0$.

2. *Eine Menge von Ereignissen $\{A_i : i \in I\}$ heisst* unabhängig, *wenn für alle $S \subseteq I$ gilt:*

$$Pr[\cap_{i \in S} A_i] = \Pi_{i \in S} Pr[A_i]$$

Für Ereignisse A_1 und A_2 gilt:

$$\begin{aligned} Pr[A_1 \cap A_2] &= Pr[A_1|A_2] \cdot Pr[A_2] \\ &= Pr[A_2|A_1] \cdot Pr[A_1] \end{aligned}$$

Per Induktion kann man zeigen, dass für Ereignisse $A_1, \ldots, A_\ell$ gilt:

$$Pr\left[\bigcap_{i=1}^{\ell} A_i\right] = Pr[A_1] \cdot Pr[A_2|A_1] \cdot Pr[A_3|A_1 \cap A_2] \cdot \ldots \cdot Pr\left[A_\ell | \bigcap_{i=1}^{\ell-1} A_i\right]$$

Randomisierte Min Cut-Algorithmen

Ein einfacher Monte Carlo-Algorithmus für Min Cut

Man betrachtet hier folgendes Min Cut-Problem: Man fasst $G = (V, E)$ mit $c : E \longrightarrow \mathbb{N}$ auf als Multigraph, d.h. wenn $c(\{u, v\}) = \ell$, so gibt es im Multigraph ℓ Kanten, die u und v verbinden. Bezeichne nun also $G = (V, E)$ einen solchen Multigraphen. Gesucht ist nun eine Partition V_1 und V_2 von V, so dass

$$\text{cutsize}(V_1, V_2) := \left| \left\{ e \in E : \begin{array}{l} e \text{ verbindet Knoten } u \text{ und } v \text{ mit} \\ u \in V_1 \text{ und } v \in V_2 \text{ oder umgekehrt} \end{array} \right\} \right|$$

minimal ist.

Algorithmus Random Min Cut:

Der Input des Algorithmus ist $G = (V, E)$.

Solange $|V| > 2$, wähle eine zufällige Kante $e \in E$ und bilde neuen Graph $G = (V, E)$, der entsteht, in dem die Endknoten von e verschmolzen werden und in dem man alle Kanten zwischen Endknoten von e entfernt.

In jedem Schritt nimmt $|V|$ um 1 ab, d.h. nach $n - 2$ Schritten endet das Verfahren mit 2 Knoten v_1 und v_2, die einen Schnitt(V_1, V_2) des Ausgangsgraphen G induzieren. Nun stellt sich die Frage, wie gross die Wahrscheinlichkeit ist, dass Random Min Cut einen minimalen Schnitt liefert. Die Antwort hierauf liefert der nächste Satz:

Satz 1.1
Die Wahrscheinlichkeit, dass Random Min Cut *einen bestimmten minimalen Schnitt (der möglicherweise auch der einzige ist) findet mit der Bedingung, dass alle Kanten die gleiche Wahrscheinlichkeit besitzen gewählt zu werden, ist grösser als* $\frac{2}{n^2}$*, wobei* $|V| = n$*.*

Beweis: Sei (V_1, V_2) ein beliebiger, vorgegebener Schnitt von G mit k Kanten. Dann hat G mindestens $k \cdot \frac{n}{2}$ Kanten, da alle Knoten in G mindestens Grad k haben. Man schätzt nun die Wahrscheinlichkeit, dass während der Durchführung von Random Min Cut niemals eine Kante zwischen V_1 und V_2 gewählt wird, ab. Sei A_i das Ereignis, dass im i-ten Schritt keine Kante

aus (V_1, V_2) gewählt wird und $1 \leq i \leq n-2$. Dann ist

$$Pr(A_1) \geq 1 - \frac{2}{n},$$

da die Wahrscheinlichkeit, dass im ersten Schritt gerade eine Kante aus (V_1, V_2) gewählt wird, höchstens $k \cdot \frac{2}{k \cdot n}$ ist. Nach dem ersten Schritt gibt es mindestens noch $k \cdot \frac{n-1}{2}$ Kanten. Entsprechend ist die Wahrscheinlichkeit, dass im zweiten Schritt eine Kante aus (V_1, V_2) gewählt wird, nachdem im ersten Schritt A_1 eingetreten ist höchstens

$$k \cdot \frac{2}{k \cdot (n-1)}, \quad \text{also} \quad Pr[A_2|A_1] \geq 1 - \frac{2}{n-1}.$$

Beim i-ten Schritt gibt es $n-i+1$ Knoten und damit also mindestens $k \cdot \frac{(n-i+1)}{2}$ Kanten. Nun folgt:

$$Pr\left[A_i | \bigcap_{j=1}^{i-1} A_j\right] \geq 1 - \frac{2}{n-i+1}$$

Die Wahrscheinlichkeit, dass in keinem der $n-2$ Schritte eine Kante aus (V_1, V_2) gewählt wird, ist dann

$$\begin{aligned} Pr\left[\bigcap_{i=1}^{n-2} A_i\right] &\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) \\ &= \frac{2}{n \cdot (n-1)} \\ &= \frac{1}{\binom{n}{2}}. \end{aligned}$$

□

Wenn man die Wahl einer zufälligen Kante in $\mathcal{O}(n)$ realisieren kann, so hat Random Min Cut eine Laufzeit von $\mathcal{O}(n^2)$. Diese Laufzeit ist deutlich besser als die des deterministischen Min Cut-Algorithmus. Wendet man Random Min Cut $\frac{n^2}{2}$ mal unabhängig voneinander an, so ist die Wahrscheinlichkeit, dass ein bestimmter Schnitt nicht gefunden wurde, höchstens

$$\left(1 - \frac{2}{n^2}\right)^{\frac{n^2}{2}} < \frac{1}{e} \qquad \text{wobei } e \text{ die EULER-Zahl ist.}$$

Dies ist schlechter als beim deterministischen Min Cut-Algorithmus.

Folgerung 1.1
Wendet man Random Min Cut *nur $n-\ell$ Schritte lang an, d.h. man stoppt, wenn ℓ Knoten übrig sind, so ist die Wahrscheinlichkeit, dass bis dahin keine Kante eines bestimmten minimalen Schnitts (V_1, V_2) gewählt wurde, mindestens*

$$\frac{\binom{\ell}{2}}{\binom{n}{2}}, \qquad \text{d.h. in } \Omega\left(\left(\frac{\ell}{n}\right)^2\right) \text{ und somit polynomial.}$$

Ein effizienterer randomisierter Min Cut**-Algorithmus**

Um mit Random Min Cut eine gute, d.h. geringe Fehlerwahrscheinlichkeit zu garantieren, muss man ihn „oft“ wiederholen. Um die Schranke $\frac{1}{e}$ (e ist die EULER-Zahl) zu garantieren benötigt man eine Laufzeit von $\mathcal{O}(n^4)$. Diese Laufzeit ist nicht gut und kann mit folgender Idee verbessert werden:

Wende Random Min Cut so viele Schritte an, bis der Graph mit $\frac{n}{\sqrt{2}}$ Knoten übrig ist und berechne darin rekursiv (random) einen minimalen Schnitt. Dies wird zweimal ausgeführt und der kleinere der beiden Schnitte ausgewählt.

Algorithmus Fast Random Min Cut:

Der Input ist ein Graph $G = (V, E)$ mit $|V| = n$.

1. Falls $n \leq 6$ berechne direkt deterministisch einen Min Cut, ansonsten führe aus:
2. Setze $\ell := \left\lceil \frac{n}{\sqrt{2}} \right\rceil$
3. Führe zweimal unabhängig Random Min Cut aus, bis nur noch ℓ Knoten übrig sind. Die zugehörigen Graphen seien G_1 und G_2.
4. Berechne rekursiv Fast Random Min Cut von G_1 und G_2.
5. Gib den kleineren der beiden Schnitte aus.

Satz 1.2
FAST RANDOM MIN CUT *hat eine Laufzeit von* $\mathcal{O}(n^2 \cdot \log n)$.

Beweis: Die Laufzeit $T(n)$ ergibt aus der folgenden Rekursionsabschätzung:

$$T(n) = 2 \cdot T\left(\left\lceil \frac{n}{\sqrt{2}} \right\rceil\right) + c \cdot n^2$$

Dabei ist c eine Konstante. Es folgt unmittelbar, dass $T(n) \in \mathcal{O}(n^2 \cdot \log n)$ ist. □

Satz 1.3
Die Wahrscheinlichkeit, dass FAST RANDOM MIN CUT *einen minimalen Schnitt findet, ist in* $\Omega\left(\frac{1}{\log n}\right)$.

Beweisidee: Der Beweis des letzten Satzes ist lang und schwierig. Deshalb ist hier nur eine Beweisskizze angegeben. Angenommen die Grösse eines minimalen Schnittes habe k Kanten und angenommen es existiert ein Graph G' mit ℓ Knoten, der aus G entstanden ist und ebenfalls einen Schnitt mit k Kanten habe. Jetzt betrachtet man einen Durchlauf für G' von FAST RANDOM MIN CUT. Das Ergebnis wird genau dann ein minimaler Schnitt von G' sein (und damit für G), wenn die Rekursion für G_1 oder für G_2 einen Schnitt der Grösse k ausgibt. Die Wahrscheinlichkeit, dass bei der Berechnung von G' keine Kante eines bestimmten Schnittes ausgewählt wurde, ist mindestens

$$\left\lceil \frac{1}{\sqrt{2}} \right\rceil \cdot \frac{\left\lceil \frac{1}{\sqrt{2}} \right\rceil - 1}{\ell \cdot (\ell - 1)} \geq \frac{1}{2} \, .$$

Bezeichne $P(\ell)$ die Wahrscheinlichkeit, dass FAST RANDOM MIN CUT in einem Graph mit ℓ Knoten einen minimalen Schnitt findet, so folgt:

$$\begin{aligned} P(\ell) &\geq 1 - \left(1 - \frac{1}{2} \cdot P\left(\left\lceil \frac{\ell}{\sqrt{2}} \right\rceil\right)\right)^2 \\ &= P\left(\left\lceil \frac{\ell}{\sqrt{2}} \right\rceil\right) - \frac{1}{4} \cdot P\left(\frac{\ell}{\sqrt{2}}\right)^2 \end{aligned}$$

Setze nun $\ell = \sqrt{2^{k+1}}$, dann folgt

$$P\left(\sqrt{2^{k+1}}\right) \geq P\left(\left(\sqrt{2}\right)^k\right) - \frac{1}{4} \cdot P\left(\left(\sqrt{2}\right)^k\right)^2 ,$$

also wenn

$$s(k) \quad := \quad P\left(\left(\sqrt{2}\right)^k\right), \quad \text{so ist} \quad s(k+1) \geq s(k) - \frac{1}{4} \cdot s(k)^2$$

Wenn man nun

$$\begin{aligned} q(k) &:= \frac{4}{s(k)-1} \qquad \text{setzt, d.h.} \\ s(k) &= \frac{4}{q(k)+1} \qquad \text{dann folgt:} \end{aligned}$$

$$\begin{aligned} s(k+1) = \frac{4}{q(k+1)+1} &\geq \frac{4}{q(k)+1} - \frac{4}{(q(k)+1)^2} \\ q(k)+1 &\geq q(k+1)+1 - \frac{q(k+1)+1}{q(k)+1} \\ &= (q(k+1)+1) \cdot \left(1 - \frac{1}{q(k)+1}\right) \\ \Longrightarrow \quad q(k+1)+1 &\leq (q(k)+1) \cdot \left(\frac{1}{1-\frac{1}{q(k)+1}}\right) \\ q(k+1) &\leq q(k)+1-1+\frac{\frac{1}{q(k)+1}}{1-\frac{1}{q(k)+1}} \\ &= \frac{q(k)+\frac{1}{q(k)+1}}{1-\frac{1}{q(k)+1}} \\ &= q(k)+1+\frac{1}{q(k)}\,. \end{aligned}$$

Induktiv lässt sich nun zeigen, dass

$$q(k) < k + \sum_{i=1}^{k-1} \frac{1}{i} + 3 \quad \in \quad \Theta(k + \log k)$$

ist. Daraus folgt $s(k) \in \Omega\left(\frac{1}{k}\right)$ und $P(\ell) \in \Omega\left(\frac{1}{\log \ell}\right)$. □

Grundlagen der Wahrscheinlichkeitstheorie II

Definition: 5
Zu einem Wahrscheinlichkeitsraum (Ω, Pr) heisst eine Funktion X, definiert als Abbildung

$$X : \Omega \longrightarrow \mathbb{R}$$

Zufallsvariable.

Die Definition der Zufallsvariablen ermöglicht die Darstellung komplexer Ereignisse in kompakter Form. Man schreibt:

$$\begin{aligned} X &= x \quad \text{für } \{\omega \in \Omega \mid X(\omega) = x\} \quad \text{und} \\ Pr[X = x] &= Pr[\{\omega \in \Omega \mid X(\omega) = x\}] \end{aligned}$$

Beispiel 1.1

X	:	*obenliegender Würfelseite wird entsprechende Punktzahl zugeordnet*
X'	:	*obenliegender Würfelseite wird entsprechende Punktzahl der Rückseite zugeordnet*

Definition: 6

1. *Zwei Zufallsvariablen X und Y heissen* unabhängige *Zufallsvariablen, falls*

$$Pr[X = x \ \wedge \ Y = y] = Pr[X = x] \cdot Pr[Y = y] \ .$$

2. *Der* Erwartungswert $E(X)$ *einer Zufallsvariablen X ist definiert durch*

$$E(X) := \sum_{x \in X(\Omega^*)} x \cdot Pr(X = x) \ ,$$

wobei Ω^* die Menge aller Elementarereignisse Ω mit positiver Wahrscheinlichkeit ist.

Beispiel: Für den Erwartungswert bei einem Würfel gilt:

$$E(X) = \sum_{i=1}^{6} i \cdot \frac{1}{6} = \frac{27}{6} = \frac{7}{2} = 3.5$$

■

Aus der Wahrscheinlichkeitstheorie sind ferner folgende Ergebnisse bekannt:

1. Für Zufallsvariablen X und Y und einen skalaren Faktor $c \in \mathbb{R}$ gilt:

$$E(c \cdot X) = c \cdot E(X) \quad \text{und}$$

$$E(X + Y) = E(X) + E(Y)$$

2. Falls X und Y unabhängige Zufallsvariablen sind, so gilt:

$$E(X \cdot Y) = E(X) \cdot E(Y)$$

Beispiele für randomisierte Algorithmen

MAXIMUM SATISFIABILITY PROBLEM

Gegeben:

Man hat eine Menge von m Klauseln über einer Variablenmenge V mit der Mächtigkeit $|V| = n$ gegeben.

Gesucht:

Man möchte eine Wahrheitsbelegung finden, die eine maximale Anzahl von Klauseln erfüllt.

Bemerkung: 1
Das MAXIMUM SATISFIABILITY PROBLEM, *auch bekannt als* MAX SAT, *ist ein $\mathcal{NP}$-schweres Problem. Es ist sogar schon $\mathcal{NP}$-schwer, wenn man die Anzahl der Literale pro Klausel auf maximal 2 beschränkt (man spricht dann vom* MAX-2-SAT-*Problem).*

Beispiel 1.2

1. Klausel: $X_1 \vee \overline{X_2}$ *2. Klausel:* $\overline{X_1} \vee \overline{X_2}$

3. Klausel: $X_1 \vee X_2$ *4. Klausel:* $\overline{X_1} \vee X_3$

5. Klausel: $X_2 \vee \overline{X_3}$

Diese Klauseln sind nicht alle gleichzeitig erfüllbar, denn falls zum Beispiel $X_1 = falsch$, so ist $\overline{X_2} = wahr$ und aus der 3. Klausel würde folgen, dass auch $X_2 = wahr$ sein müsste, also Widerspruch. Belegt man nun $X_1 = wahr$, so folgt $\overline{X_2} = wahr$ und $X_3 = wahr$, aber die 5. Klausel liefert dann falsch

zurück. Eine maximale Anzahl von Klauseln mit wahr zu belegen, liefert $X_1 = wahr$, $X_2 = falsch$ *und* $X_3 = wahr$. *Dann sind 4 von 5 Klauseln erfüllt.*

Algorithmus RANDOM SAT

Für jede Variable $x \in V$ setze $\omega(x) := wahr$ mit der Wahrscheinlichkeit $\frac{1}{2}$, wobei V eine Menge von Variablen ist mit der Mächtigkeit $|V| = n$.

Bezeichne $X_{RS}(I)$ die Zufallsvariable, die den Wert der Lösung von RANDOM SAT bei der Eingabe von I angibt.

Satz 1.4
Für eine Instanz I *von* MAX SAT, *in der jede Klausel höchstens* k *Literale enthält, erfüllt der erwartete Wert der Lösung von* RANDOM SAT*:*

$$E(X_{RS}(I)) \geq \left(1 - \frac{1}{2^k}\right) \cdot m$$

Beweis: Die Wahrscheinlichkeit, dass eine Klausel mit k Literalen nicht erfüllt wird, ist $\frac{1}{2^k}$. Entsprechend ist die Wahrscheinlichkeit, dass eine Klausel mit mindestens k Literalen erfüllt wird mindestens $1 - \frac{1}{2^k}$. Damit ist der erwartete Beitrag einer Klausel zu $E(X_{RS}(I))$ mindestens $\frac{1}{2^k}$, also folgt:

$$E(X_{RS}(I)) \geq 1 - \frac{1}{2^k} \cdot m$$

□

Das MAX CUT**-Problem**

Gegeben:

Man betrachte einen Graphen $G = (V, E)$ mit Gewichtsfunktion $c\colon E \longrightarrow \mathbb{N}$.
Gesucht:
Man möchte nun einen Schnitt $(S, V \setminus S)$ von G mit maximalen Gewicht, d.h.

$$c(S, V \setminus S) := \sum_{u,v \in E} c(\{u, v\}) \qquad \text{soll maximal sein, wobei}$$

$u \in S$ und $v \in V \setminus S$ ist. MAX CUT ist ein ähnliches Problem wie MIN CUT, aber im Gegensatz zu diesem ist es $\mathcal{NP}$-schwer.

1.0.1 Randomisierter Algorithmus für MAX CUT basierend auf semidefiniter Programmierung

Sei I eine Instanz für MAX CUT. Dann definiere dazu ein ganzzahliges quadratisches Programm *IQP(I)*. Zu i und $j \in V := \{1, \ldots, n\}$ definiere

$$c_{ij} := \begin{cases} c(\{i,j\}) & \text{falls } \{i,j\} \in E \\ 0 & \text{sonst} \end{cases}$$

c_{ij} heisst *Gewichtsmatrix* zum Graphen G. Man betrachtet nun

$$\max \frac{1}{2} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot (1 - x_i \cdot x_j) \quad \text{, wobei}$$

$x_i, x_j \in \{-1, 1\}$ und $1 \leq i, j \leq n$. Dies ist das $IQP(I)$.

Nun sind folgende Fälle möglich:

$i = j$	:	Knoten x_i und x_j sind auf derselben Seite, d.h. die Kante wird nicht gezählt	:	$(1-1) \cdot c_{ij}$	$= 0$
$i \neq j$	:	Knoten x_i und x_j sind auf verschiedenen Seiten, d.h. die Kante wird gezählt	:	$\frac{1}{2} \cdot c_{ij} \cdot 2$	$= c_{ij}$

Dann induziert die Belegung der x_i und x_j eine Partition $(S, V \setminus S)$ von V mit

$$c(S, V \setminus S) = \frac{1}{2} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot (1 - x_i \cdot x_j) \; .$$

Denn falls i und $j \in S$ oder i und $j \in V \setminus S$, so gilt $x_i = x_j$ und damit folgt $(1 - x_i \cdot x_j) = 0$, andernfalls erhält man $\frac{1}{2} \cdot (1 - x_i \cdot x_j) = 1$. Eine optimale Lösung von $IQP(I)$ induziert also einen MAX CUT zu I. Mit dieser Fassung des Problems ist es immer noch $\mathcal{NP}$-schwer. Jede Variable x_i kann als eindimensionaler Vektor der Norm 1 aufgefasst werden.

Relaxierung des IQP:

Sei X^i ein Vektor mit der Norm 1 im zweidimensionalen Raum und sei QP-CUT(I) definiert als

$$\max \frac{1}{2} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot (1 - x^i \cdot x^j), \quad \text{wobei}$$

$x^i, x^j \in \mathbb{R}^2$ mit Norm 1 und $1 \leq i, j \leq m$. Damit gilt für das Produkt von x^i und x^j:

$$x^i \cdot x^j = x_1^i \cdot x_1^j + x_2^i \cdot x_2^j$$

QP-Cut(I) ist tatsächlich eine Relaxierung des $IQP(I)$, denn jede Lösung $(x_1, \ldots, x_n)$ von $IQP(I)$ induziert eine Lösung $(x^1, \ldots, x^n)$ von QP-Cut(I) mittels $x^i = (x_i, 0)$.

Idee eines randomisierten Algorithmus zur Lösung des IQP:

Berechne eine optimale Lösung $(\widetilde{X^1}, \ldots, \widetilde{X^n})$ von QP-Cut(I) und konstruiere daraus die Lösung zu $IQP(I)$ mittels eines zufällig gewählten zweidimensionalen Norm-1-Vektors r: S enthalte genau die Knoten $i \in V$, für die der Vektor $\widetilde{x^i}$ oberhalb der zu r senkrechten Linie ℓ liegt.

Algorithmus Random Max Cut

Der Input ist ein Graph $G = (V, E)$ mit einer Gewichtsfunktion $c : E \longrightarrow \mathbb{N}$.

1. Stelle QP-Cut auf.
2. Berechne die optimale Lösung $(\widetilde{X^1}, \ldots, \widetilde{X^n})$ des QP-Cut.
3. Wähle zufällig einen zweidimensionalen Norm-1-Vektor r.
4. Setze $S := \{i \in V : \widetilde{x^i} \cdot r \geq 0\}$

Der Algorithmus gibt einen Schnitt $(S, V \setminus S)$ zurück.

Random Max Cut ist polynomial, falls Schritt 2 polynomial ist. Es ist derzeit nicht bekannt, ob ein QP-Cut mit polynomialer Laufzeit gelöst werden kann. Man kann allerdings QP-Cut so modifizieren, dass Random Max Cut polynomial wird.

Satz 1.5
Sei I eine Instanz für Max Cut *und $C_{RMC}(I)$ der Wert der Lösung, die* Random Max Cut *für I berechnet. Wenn die Vektoren in Schritt 3 gleichverteilt angenommen werden, so gilt:*

$$E(C_{RMC}(I)) = \frac{1}{\pi} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot \arccos(\widetilde{x^i} \cdot \widetilde{x^j})$$

Beweis: Definiere sgn : $\mathbb{R} \longrightarrow \{1, -1\}$ als

$$\text{sgn}(x) := \begin{cases} 1 & \text{falls} \quad x \geq 0 \\ -1 & \text{sonst} \end{cases}$$

Offensichtlich gilt:

$$E(C_{RMC}(I)) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot Pr[\text{sgn}(\widetilde{x^i} \cdot r)] \quad \neq \quad \text{sgn}(\widetilde{x^j} \cdot r) \text{ , wobei}$$

r zufällig und gleichverteilt gewählt ist. Nun genügt es zu zeigen, dass

$$Pr[\text{sgn}(\widetilde{x^i} \cdot r) \ \neq \ \text{sgn}(\widetilde{x^j} \cdot r)] \ \neq \ \frac{\arccos(\widetilde{x^i} \cdot \widetilde{x^j})}{\pi} \text{ ist.}$$

Für die Funktion sgn gilt, dass $\text{sgn}(\widetilde{x^i} \cdot r) \ \neq \ \text{sgn}(\widetilde{x^j} \cdot r)$ ist, genau dann wenn die zugehörige Zufallslinie ℓ senkrecht zu r gerade $\widetilde{x^i}$ und $\widetilde{x^j}$ trennt. Seien s und t die Schnittpunkte des Einheitskreises um den Ursprung mit ℓ.

$\widetilde{x^i}$ und $\widetilde{x^j}$ werden genau dann von ℓ getrennt, wenn entweder s oder t auf dem kürzeren Kreisbogen zwischen $\widetilde{x^i}$ und $\widetilde{x^j}$ liegt. Die Wahrscheinlichkeit, dass s oder t auf diesem Kreisbogen liegen, ist

$$\frac{\arccos(\widetilde{x^i} \cdot \widetilde{x^j})}{2 \cdot \pi} + \frac{\arccos(\widetilde{x^i} \cdot \widetilde{x^j})}{2 \cdot \pi} = \frac{\arccos(\widetilde{x^i} \cdot \widetilde{x^j})}{\pi}$$

Daraus folgt die Gütegarantie. □

Satz 1.6
Für eine Instanz I von Max Cut *berechnet* Random Max Cut *eine Lösung mit dem Wert $C_{RMC}(I)$, für die gilt:*

$$\frac{E(C_{RMC}(I))}{OPT(I)} \geq 0.8785$$

Modifiziere QP-Cut in ein effizient lösbares Problem:

Ersetze QP-Cut durch folgendes n-dimensionales QP-Cut-Problem:

$$\max \frac{1}{2} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot (x^i \cdot x^j) \text{ , wobei}$$

x^i und x^j n-dimensionale Norm-1-Vektoren über $\mathbb{R}$ sind. x^i und x^j erfüllen gewisse Bedingungen, die polynomiale Lösbarkeit garantieren. Doch zunächst benötigt man noch einige Begriffe.

Definition: 7
Eine $n \times m$-Matrix M heisst positiv semidefinit*, falls für jeden Vektor $x \in \mathbb{R}^n$ gilt:*

$$X^{transponiert} \cdot M \cdot X \geq 0$$

Es ist bekannt, dass eine symmetrische Matrix M genau dann positiv semidefinit ist, wenn es eine $m \times n$-Matrix P $(m \leq n)$ gibt, so dass

$$M = P^{transponiert} \cdot P \qquad \text{ist.}$$

P kann in polynomialer Laufzeit berechnet werden, falls M positiv semidefinit ist.

Betrachte nun Vektoren $x^1, \ldots, x^n \in \mathbb{R}^n$ mit Norm 1 und definiere $M := (m_{ij})$ mit $m_{ij} := x^i \cdot x^j$. Dann ist M positiv definit. Andererseits gilt für jede positiv semidefinite $n \times n$-Matrix M mit $m_{ij} = 1$ für $1 \leq i \leq n$, dass eine Menge von n Vektoren $x^1, \ldots x^n \in \mathbb{R}^n$ mit Norm 1 und $m_{ij} = x^i \cdot x^j$ in polynomialer Zeit berechnet werden können. Nun ist QP-Cut äquivalent zu

$$\max \frac{1}{2} \cdot \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \cdot (1 - m_{ij}) \quad \text{, wobei}$$

$M = (m_{ij})$ eine positiv semidefinite Matrix ist und $m_{ii} = 1$ ist für $1 \leq i \leq n$. Dieses Problem heisst Semi-Definit-Cut(I) oder SD-Cut(I). Man kann nun beweisen, dass es für OPT_{SD-Cut} einen optimalen Lösungswert von SD-Cut und für jedes $\varepsilon > 0$ es einen polynomialen Algorithmus $\mathcal{A}_\varepsilon$ gibt mit

$$\mathcal{A}_\varepsilon(I) \geq \mathrm{OPT}_{SD-Cut}(I) - \varepsilon$$

$\mathcal{A}_\varepsilon$ ist polynomial in der Eingabegrösse von I und in $\log(\frac{1}{\varepsilon})$. Daraus folgt, dass $\mathcal{A}_\varepsilon$ ein polynomialer, exakter Algorithmus für SD-Cut ist. Man kann nun zeigen, dass mit $\varepsilon = 10^{-5}$ die Approximationsgarantie von 0.8785 für Random Max Cut erreicht werden kann.