FernUniversität Hagen
Fakultät für Mathematik und Informatik

Diplomarbeit

Untersuchungen zur Komplexität der Berechnung eines Kernels in clique-azyklischen Orientierungen perfekter Graphen

Ute Mattheis
Matrikelnummer 7514948

16. Mai 2015

betreut durch Prof. Dr. Winfried Hochstättler

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Jena, 25. Februar 2015

(Ute Mattheis)

Danksagung

Herrn Professor Dr. Hochstättler danke ich für dieses bunte Thema, durch das ich sehr viel gelernt habe, sowie für seine gute Betreuung. Meinem Mann Konrad danke ich für seine Ermutigung sowie die praktische Entlastung, die er mir trotz seines 200% -Jobs gegeben hat. Meine Kinder, Luise, Anton und Hannah, haben mich mit Ihrer Selbständigkeit sowie ihrem Glauben daran, dass diese „Blumenarbeit“ zu einem guten Ende kommt, unterstützt. Meinen Eltern und Schwiegereltern, besonders Roland, danke ich für Asyl und Kinderbetreuung. Monika hat mir geholfen, indem sie in der letzten Woche tagsüber unsere kranken Kinder übernommen hat, so dass ich das hier fertig schreiben konnte.

Inhaltsverzeichnis

1 Einleitung

Ein perfekter Graph ist dadurch charakterisiert, dass er weder ungerade Löcher mit mehr als drei Knoten noch deren Komplement enthält. Bei einer Orientierung werden die Kanten eines Graphen mit einer Richtung versehen. Wenn dabei Paare entgegengesetzt gerichteter Kanten erlaubt sind, wird diese als Superorientierung bezeichnet. Ein Digraph ist clique-azyklisch, wenn jeder gerichtete Kreis, welcher in einer Clique enthalten ist, mindestens eine Kante besitzt, zu der es eine entgegengesetzt gerichtete Kante gibt. Ein Kernel ist eine Knotenmenge eines Digraphen, in der keine zwei Knoten durch eine Kante miteinander verbunden sind, und die zu jedem Knoten des Digraphen entweder diesen Knoten selbst oder einen seiner Vorgänger enthält. Diese Arbeit untersucht die Komplexität der Berechnung eines Kernels in clique-azyklischen Superorientierungen perfekter Graphen. Den Rahmen dafür bildet ein Artikel von Kintali, Poplawski, Rajamaran, Sundaram und Teng [1], in welchem die PPAD-Vollständigkeit des stark verwandten Problems STRONG KERNEL bewiesen wurde. Es wird überprüft, ob sich dieser Beweis auf Kernel übertragen lässt. Das Problem STRONG KERNEL basiert auf einem Lemma von Scarf [2]. Deshalb wird von den verschiedenen existierenden Beweisen [3] hier der Beweis von Aharoni und Holzman [4] genutzt, der ebenfalls unter Verwendung des Lemmas von Scarf zeigt, dass ein solcher Kernel immer existiert.
Diese Arbeit besteht aus vier Teilen. In den Grundlagen werden die notwendigen Sätze und Definitionen bereitgestellt, die in den späteren Beweisen verwendet werden. Hier werden auch die Komplexitätsklasse PPAD , das Suchproblem END OF THE LINE sowie solche Begriffe wie Reduktion, PPAD-schwere und PPAD-vollständige Probleme definiert.
Im zweiten Teil werden der Beweis von Aharoni und Holzman [4] sowie das von ihnen verwendete Lemma von Scarf [2] vorgestellt. Scarfs algorithmischer Beweis ist sehr technisch, deshalb wurde er nicht in den Beweis von Aharoni und Holzman eingebunden. Der Beweis von Aharoni und Holzman liefert einen Algorithmus zur Kernelberechnung. Es wurde darauf geachtet, dass beide Beweise ohne Sekundärliteratur vollständig nachvollzogen werden können. Für ein besseres Verständnis werden beide Algorithmen an je einem Beispiel veranschaulicht. Zum Abschluss dieses Kapitels werden einige Eigenschaften des Algorithmus zur Kernelberechnung vorgestellt.
Als Drittes wird die Reduktionskette von Kintali u.a. [1] dargestellt, an der sich auch der Aufbau dieses Kapitels orientiert. Die Ergebnisse dieser Reduktionskette werden im letzten Teil benötigt. Wenn die Beweise sehr umfangreich sind, wird sich auf einen Abriss beschränkt, um so einen Eindruck von der Struktur zu vermitteln. Im letzten Abschnitt wird die Komplexität der Kernelberechnung untersucht. Es wird gezeigt, dass das Problem KERNEL in der Komplexitätsklasse PPAD liegt. Anschließend wird untersucht, wie sich der Algorithmus zur Kernelberechnung verhält, wenn der eingegebene Digraph nicht clique-azyklisch ist. Danach wird getestet, ob der Beweis von Kintali et al. [1], welcher zeigt, dass das Problem STRONG KERNEL PPAD-schwer ist, auf das Problem KERNEL übertragbar ist. Betrachtungen zur Polynomialität der Laufzeit von KERNEL und SCARF beschließen diese Arbeit.

2 Grundlagen

In diesem Abschnitt werden die für diese Arbeit grundlegenden Definitionen und Sätze bereitgestellt. Dabei werden die Begriffe und Sätze, die bereits vor Beginn der Diplomarbeit bekannt waren, als bekannt vorausgesetzt.

Vorbemerkung 2.1. Die in dieser Arbeit betrachteten ungerichteten Graphen $G = (V, E)$ seien endlich, ohne Mehrfachkanten und Schlingen, also einfach. Die Digraphen $D = (V, A)$ seien ebenfalls endlich, ohne Schlingen und können Paare entgegengesetzt gerichteter Kanten enthalten.

2.1 Graphentheorie

Zunächst werden einige Begriffe der Graphentheorie wiederholt und neue eingeführt. Anschließend werden ein Lemma von Berge, der schwache Satz über perfekte Graphen sowie eine äquivalente Formulierung mit Ungleichungen bewiesen. Der starke Satz über perfekte Graphen wird ohne Beweis genannt.
Im Folgenden sei $G = (V, E)$ ein Graph mit $|V| = n$.

Bezeichnungen 2.2. Es bezeichnen

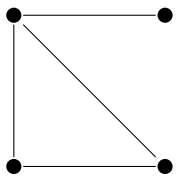
(i) $\alpha(G)$ die *Unabhängigkeitszahl* oder auch *Stabilitätszahl*, welche die Kardinalität einer größten unabhängigen Knotenmenge in G angibt,
(ii) $\omega(G)$ die *Cliquenzahl* von G, welche die Mächtigkeit einer größten Clique von G angibt,
(iii) $\chi(G)$ die *Färbungszahl* oder auch *chromatische Zahl* von G, also die minimale Anzahl unabhängiger Knotenmengen, die alle gemeinsamen Knoten von G bedecken,
(iv) $\kappa(G)$ die *Cliquenüberdeckungszahl* von G, also die minimale Anzahl von Cliquen, die gemeinsam alle Knoten von G bedecken,
(v) $\overline{G}$ das *Komplement* von G, wobei $\overline{G}$ dieselbe Knotenmenge wie G hat, und zwei Knoten genau dann in $\overline{G}$ adjazent sind, wenn sie in G nicht miteinander verbunden sind.

Bemerkung 2.3. Die Cliquen eines Graphs G entsprechen gerade den unabhängigen Mengen seines Komplements $\overline{G}$ und umgekehrt. Für alle Graphen G gilt deshalb

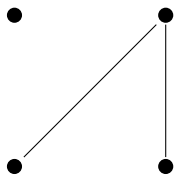
$$\alpha(\overline{G}) = \omega(G) \leq \chi(G) = \kappa(\overline{G})$$

.

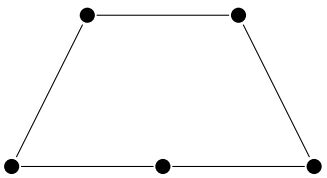
Definition 2.4. Ein Graph G heißt *perfekt*, wenn $\chi(H) = \omega(H)$ für alle seine induzierten Teilgraphen $H \subseteq G$ gilt.



G_1 ist perfekt.



$\overline{G_1}$ ist perfekt.



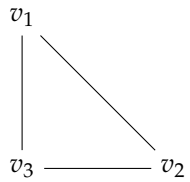
G_2 ist nicht perfekt, da $2 = \omega(G_2) \neq \chi(G_2) = 3$ ist.

Abb. 2.1: Beispiele für perfekte bzw. nicht perfekte Graphen G_1, $\overline{G_1}$ und G_2.

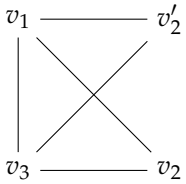
Bemerkung 2.5. Der komplementäre Graph $\overline{G}$ ist also perfekt, wenn für alle induzierten Teilgraphen $H \subseteq G$ von G deren Unabhängigkeitszahl und Cliquenüberdeckungszahl übereinstimmen, also $\alpha(H) = \kappa(H)$ ist.
Die Abbildung 2.1 zeigt einfache Beispiele für perfekte bzw. nicht perfekte Graphen.

Definition 2.6. Unter einer *Multiplikation* eines Knotens v mit $h \in \mathbb{N}_0$ wird das Ersetzen des Knotens v durch h unabhängige Knoten, die mit denselben Knoten verbunden sind wie v, verstanden. $G \circ v$ bezeichne den Graphen, bei dem v verdoppelt wird. $G \circ h$ mit $h \in \mathbb{N}_0^n$ bezeichne den Graphen, in dem jeder Knoten v_i durch eine stabile Menge der Kardinalität h_i ersetzt wird.

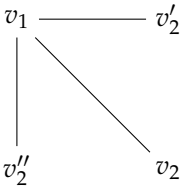
Bemerkung 2.7. Für $h \in \{0,1\}^n$ ist $G \circ h$ gerade der von den Knoten v_i mit $h_i = 1$ induzierte Teilgraph. In Abbildung 2.2 sind je ein Beispiel für die Knotenverdopplung eines Graphen G und die Multiplikation mit h dargestellt.



Graph G



$G \circ v_2$



$G \circ h$ mit $h^\top = (1,3,0)$

Abb. 2.2: Beispiele für eine Verdopplung $G \circ v_2$ und eine Multiplikation $G \circ h$

Lemma 2.8 (Berge)**.** *Sei $h \in \mathbb{N}_0^n$. Dann gilt:*
(1) G perfekt $\Rightarrow$ $G \circ h$ perfekt
(2) $\overline{G}$ perfekt $\Rightarrow$ $\overline{G \circ h}$ perfekt

Beweis. Dieses Lemma wurde von Berge [5] 1961 bewiesen.Beide Teile werden mit vollständiger Induktion gezeigt.
Die beiden Aussagen sind richtig, wenn G aus nur einem Knoten besteht. Nach Definition der Perfektheit sind sie auch für alle induzierten Teilgraphen richtig, also für alle $h \in \{0,1\}^n$. Mittels Induktion über $\sum_{i=1}^n h_i$ kann angenommen werden, dass h ein Vektor mit einer Komponente gleich zwei und allen anderen Komponenten gleich eins ist. Nun wird $G \circ v$ mit v' als Kopie von v betrachtet.
Sei G' ein echter induzierter Teilgraph von $G \circ v$. Dann ist entweder $G' \subsetneq G$ oder $G' = H \circ v$ für einen echten Teilgraphen $H \subsetneq G$. In beiden Fällen folgen die Behauptungen aus der Induktionsvoraussetzung. Es bleibt also nur noch $G \circ v$ zu betrachten.

(1) G perfekt $\Rightarrow$ $G \circ v$ perfekt:
Die Knoten v und v' sind nicht verbunden, liegen also nicht in einer Clique. Deshalb ist $\omega(G) = \omega(G \circ v)$. G ist perfekt, somit gibt es eine Färbung von G mit $\chi(G) = \omega(G)$ Farben. Der Knoten v' wird nun mit derselben Farbe gefärbt wie v. Dies ergibt eine zulässige Färbung von $G \circ v$ mit $\chi(G \circ v) = \omega(G)$ Farben. Also gilt $\chi(G \circ v) = \omega(G) = \omega(G \circ v)$, und $G \circ v$ ist perfekt.

(2) $\overline{G}$ perfekt $\Rightarrow$ $\overline{G \circ v}$ perfekt:
Sei Π eine minimale Cliquenüberdeckung von G, das heißt $|\Pi| = \kappa(G)$. Da $\overline{G}$ perfekt ist, ist $\kappa(G) = \alpha(G)$. Sei $C_v \in \Pi$ die Clique, die v enthält. Es werden zwei Fälle unterschieden:

(a) Der Knoten v ist in einer größten stabilen Menge S, das heißt $|S| = \alpha(G)$, enthalten:
Dann ist $S \cup \{v'\}$ stabil in $G \circ v$, also ist $\alpha(G \circ v) = \alpha(G) + 1$. Und $\Pi \cup \{v'\}$ ist eine Cliquenüberdeckung von $G \circ v$. Deshalb ist $\kappa(G \circ v) \leq \kappa(G) + 1$. Damit folgt $\kappa(G \circ v) \leq \kappa(G) + 1 = \alpha(G) + 1 = \alpha(G \circ v) \leq \kappa(G \circ v)$, somit ist $\alpha(G \circ v) = \kappa(G \circ v)$. Damit wurde $\omega(\overline{G \circ v}) = \chi(\overline{G \circ v})$ bewiesen.

(b) Der Knoten v ist in keiner größten stabilen Menge S enthalten:
Dann ist $\alpha(G) = \alpha(G \circ v)$. Da $|\Pi| = \kappa(G) = \alpha(G)$ ist, schneidet jede Clique in Π jede größte stabile Menge in G genau einmal. Dies gilt auch für C_v. Nach Voraussetzung ist v in keinem dieser Schnitte enthalten. Deshalb schneidet auch $C'_v = C_v \setminus \{v\}$ jede größte unabhängige Menge genau einmal. Sei H der von $V \setminus C'_v$ induzierte Teilgraph. Dann ist $\alpha(H) = \alpha(G) - 1$, denn aus jeder größten stabilen Menge von G wurde genau ein Knoten entfernt, und dies sind gerade die größten stabilen Mengen von H. Nach Induktionsvoraussetzung ist $\kappa(H) = \alpha(H) = \alpha(G) - 1 = \alpha(G \circ v) - 1$. Sei Π' eine Cliquenüberdeckung von H mit der Kardinalität $|\Pi'| = \alpha(G \circ v) - 1$. Dann liefert Π' zusammen mit $C'_v \dot{\cup} \{v'\}$ eine Cliquenüberdeckung von $G \circ v$ mit $\kappa(G \circ v) = \alpha(G \circ v)$. □

Der folgende Satz wurde von Berge [5] als „schwache Perfekte-Graphen-Vermutung“aufgestellt und von Lovász 1972 [6] bewiesen.

Satz 2.9. (schwacher Satz über perfekte Graphen)
Ein Graph G ist genau dann perfekt, wenn sein Komplement $\overline{G}$ perfekt ist.

Beweis. Da die Cliquen von G unabhängige Mengen in $\overline{G}$ sind, und umgekehrt, genügt es eine Richtung des Satzes zu beweisen. Mit vollständiger Induktion wird bewiesen, dass aus der Perfektheit von $\overline{G}$ die Perfektheit von G folgt.
Sei $\overline{G}$ perfekt. Per Induktion ist $\omega(H) = \chi(H)$ für alle echten induzierten Teilgraphen H von G bewiesen. Es werden wieder zwei Fälle unterschieden:

(a) G enthält eine stabile Menge S, die jede größte Clique schneidet. Dann ist $\omega(G_{V \setminus S}) = \omega(G) - 1$, wobei $G_{V \setminus S}$ der von $V \setminus S$ induzierte Teilgraph von G ist. Färbe $V \setminus S$ mit $\omega(G) - 1$ Farben. Dies ist möglich, da $\omega(G_{V \setminus S}) = \chi(G_{V \setminus S})$ ist. Die Menge S wird mit einer zusätzlichen Farbe gefärbt. Damit gilt $\omega(G) = \chi(G)$.

(b) Keine stabile Menge S schneidet alle größten Cliquen. Sei $\mathcal{S}$ die Menge aller stabilen Mengen von G. Nach Voraussetzung gibt es für jede stabile Menge $S \in \mathcal{S}$ eine größte Clique C_S, die S nicht schneidet,

das heißt $|C_S| = \omega(G)$ und $S \cap C_S = \emptyset$. Für alle Knoten $v_i \in V$ sei $h_i = |\{S \in \mathcal{S} \mid v_i \in C_S\}|$ die Anzahl der stabilen Mengen S, deren zugehörige größte Clique C_S den Knoten v_i enthält. Ersetze nun jeden Knoten v_i durch eine stabile Menge der Größe h_i. Sei $H = G \circ h$. Nach dem Lemma von Berge 2.8 ist $\overline{H} = \overline{G \circ v}$ ebenfalls perfekt. Weiter gilt:

$$\begin{aligned} |V_H| &= \sum_{v_i \in V} h_i = \sum_{v_i \in V} \sum_{S \in \mathcal{S}} |\{v_i\} \cap C_S| = \sum_{S \in \mathcal{S}} \sum_{v_i \in V} |\{v_i\} \cap C_S| = \sum_{S \in \mathcal{S}} |C_S| \\ &= \omega(G)|\mathcal{S}| \end{aligned}$$

nach Definition von C_S.

Nach Konstruktion von H enthält jede Clique von H höchstens eine Kopie eines Knotens aus V. Deshalb ist

$$\omega(H) \leq \omega(G),$$

und es ist

$$\begin{aligned} \alpha(H) &= \max\{\sum_{v_i \in T} h_i \mid T \in \mathcal{S}\} && \text{[die multipl. Knoten einer unabh. Menge sind unabh.]} \\ &= \max\{\sum_{v_i \in T} \sum_{S \in \mathcal{S}} |\{v_i\} \cap C_S| \mid T \in \mathcal{S}\} && \text{[nach Definition der } h_i\text{]} \\ &= \max\{\sum_{S \in \mathcal{S}} \sum_{v_i \in T} |\{v_i\} \cap C_S| \mid T \in \mathcal{S}\} \\ &= \max\{\sum_{S \in \mathcal{S}} |T \cap C_S| \mid T \in \mathcal{S}\} \\ &\leq |\mathcal{S}| - 1. && \text{[}|T \cap C_S| \leq 1 \text{ und } |S \cap C_S| = 0 \ \forall S,T \in \mathcal{S}\text{]} \end{aligned}$$

Alle Cliquen in H sind kleiner oder gleich $\omega(H)$. Das gilt auch für die Cliquen einer Cliquenüberdeckung. Daraus folgt:

$$\kappa(H) \geq \frac{|V_H|}{\omega(H)} \geq \frac{|V_H|}{\omega(G)} = \frac{\omega(G)|\mathcal{S}|}{\omega(G)} = |\mathcal{S}| > |\mathcal{S}| - 1 \geq \alpha(H).$$

Dies bedeutet, dass $\overline{H}$ nicht perfekt ist, was der Wahl von $\overline{H}$ widerspricht. Folglich gibt es eine stabile Menge, die alle größten Cliquen schneidet, und der Fall (a) gilt. □

Der folgende Satz liefert eine äquivalente Bedingung für $\omega(G) = \chi(G)$. Er wurde 1972 von Lovász [7] bewiesen, wobei darauf hingewiesen wurde, dass die Bedingung des Satzes eng mit der Max-Max-Ungleichung von Fulkerson in [8] verwandt ist, und die Multiplikation eines Knotens dasselbe ist, was jener „pluperfection“ nennt.

Satz 2.10 (eine äq. Bedingung für die Perfektheit)**.** *Ein Graph G ist genau dann perfekt, wenn für alle seine induzierten Teilgraphen H gilt:*

$$\alpha(H)\omega(H) \geq |V_H|.$$

Beweis. $\Rightarrow$: Dies wird mit Kontraposition bewiesen. Angenommen, G hat einen Teilgraphen H, der diese Ungleichung nicht erfüllt. Das heißt, es ist $\alpha(H)\omega(H) < |V_H|$. Daraus folgt $\alpha(H)\omega(H) < |V_H| = |V_{\overline{H}}| \leq \omega(\overline{H})\kappa(\overline{H}) = \alpha(H)\chi(H)$. Es gilt also $\omega(H) < \chi(H)$. Somit kann G nicht perfekt sein.
$\Leftarrow$: Dies wird mit vollständiger Induktion über $|V|$ bewiesen. Die Induktionsverankerung ist $|V| = 1$. Es kann angenommen werden, dass alle echten induzierten Teilgraphen von G und dem Komplement $\overline{G}$ ebenfalls perfekt sind.
Als *Vorbereitung für den Induktionsschritt* wird zunächst gezeigt, dass G_0 die Ungleichung erfüllt, wenn G_0 durch Multiplikation aus G hervorgeht:
Angenommen, es gibt ein $G_0 = G \circ h$, das diese Ungleichung verletzt und bezüglich dieser Eigenschaft minimale Anzahl an Knoten hat. Dann gibt es offensichtlich einen Knoten y in G, der mit $h_y \geq 2$ multipliziert wurde, denn für $h \leq 1$ in allen Komponenten gelten die Ungleichungen nach Induktionsvoraussetzung. Seien $y_1, \dots y_{h_y}$ die korrespondierenden Knoten von y in G_0. Da G_0 minimal gewählt wurde ist

$$|V_{G_0}| - 1 \leq \omega(G_0 - y_1)\alpha(G_0 - y_1) \leq \omega(G_0)\alpha(G_0) < |V_{G_0}|.$$

Da alle Zahlen natürlich sind, folgt daraus

$$\begin{aligned} \omega(G_0) = \omega(G_0 - y_1) =: p \\ \alpha(G_0) = \alpha(G_0 - y_1) =: r \\ |V_{G_0}| = pr + 1 \end{aligned}$$

Sei nun $G_1 = G_0 - \{y_1, \dots, y_{h_y}\}$. Dann geht G_1 durch Multiplikation aus $G - y$ hervor und ist nach dem Lemma von Berge 2.8 ebenfalls perfekt. Damit ist auch $\overline{G_1}$ perfekt, und G_1 kann durch $\kappa(G_1) = \chi(\overline{G_1}) = \omega(\overline{G_1}) = \alpha(G_1) \leq \alpha(G_0) = r$ disjunkte Cliquen überdeckt werden. Seien $C_1, \dots, C_r$ mit $|C_1| \geq \dots \geq |C_r|$ diese Cliquen. Offensichtlich ist $h_y \leq r$. Andernfalls wären $\{y_1, \dots, y_{h_y}\}$ eine Clique in $\overline{G_0}$ und $\omega(\overline{G_0}) \geq h_y > r$. Es ist $|V_{G_1}| = |V_{G_0}| - h_y = pr + 1 - h_y$. Gleichzeitig ist $|V_{G_1}| = \sum_{i=1}^{r} |C_i| = |V_{G_1}|$. Daraus folgt, dass $|C_1| = \dots = |C_{r-h_y+1}| = p$ sein müssen:
Angenommen es wäre $|C_1| < p$, dann wäre aufgrund der Wahl der Cliquen C_i

$$\sum_{i=1}^{r} |C_i| \leq \sum_{i=1}^{r} (p-1) = rp - r < rp - (h_y - 1).$$

Also können nicht alle Cliquen echt kleiner als p sein. Es ist noch zu klären, wieviele $|C_i|$ mindestens gleich p sein müssen, damit die Gleichung erfüllt ist. Es wird also ein l gesucht, so dass

$$rp + 1 - h_y = \sum_{i=1}^{r} |C_i| = \sum_{i=1}^{r-l} |C_i| + \sum_{i=r-l+1}^{r} |C_i| = \sum_{i=1}^{r-l} p + \sum_{i=r-l+1}^{r} (p-1)$$

ist. Dies ist für $l = h_y - 1$ der Fall. Damit wurde gezeigt, dass mindestens

$$|C_1| = \dots = |C_{r-h_y+1}| = p$$

gelten muss.
Sei G_2 der von $C_1 \dot\cup \dots \dot\cup C_{r-h_y+1} \dot\cup \{y_1\}$ induzierte Teilgraph von G_0. Dann ist

$$|V_{G_2}| = \sum_{i=1}^{r-h_y+1} |C_i| + 1 = (r - h_y + 1)p + 1 = rp + 1 - p(h_y - 1) < rp + 1 = |V_{G_0}|.$$

G_0 wurde minimal gewählt. Deshalb gelten für den echten Teilgraphen G_2 gleichzeitig $\omega(G_2)\alpha(G_2) \geq |V_{G_2}|$ und $\omega(G_2) \leq \omega(G_0) = p$. Damit gilt

$$\alpha(G_2) \geq \frac{|V_{G_2}|}{\omega(G_2)} \geq \frac{|V_{G_2}|}{p} = \frac{rp + 1 - p(h_y - 1)}{p} = r - h_y + 1 + \frac{1}{p}.$$

Da $\alpha(G_2) \in \mathbb{N}$ und $\frac{1}{p} > 0$ sind, kann dies noch etwas genauer abgeschätzt werden:

$$\alpha(V_{G_2}) \geq r - h_y + 2.$$

Damit wurde gezeigt, dass es in G_2 eine unabhängige Menge F der Größe $r - h_y + 2$ gibt. Da $|F \cap C_i| \leq 1$ ist für alle $i \in [r - h_y + 1]$, muss y_1 in F enthalten sein, denn andernfalls wäre F zu klein. Daraus folgt wiederum, dass $F \dot\cup \{y_2, \ldots, y_{h_y}\}$ in G_0 unabhängig ist. Womit widersprüchlicherweise

$$\alpha(G_0) \geq |F \cup \{y_2, \ldots, y_{h_y}\}| = r - h_y + 2 + h_y - 1 = r + 1 > r = \alpha(G_0)$$

gezeigt wurde.
Induktionsschritt: Gegeben sei ein Graph G, dessen induzierte Teilgraphen H alle die Ungleichung $\alpha(H)\omega(H) \geq |V_H|$ erfüllen. Nach Induktionsvoraussetzung sind alle echten Teilgraphen perfekt. Nun soll gezeigt werden, dass G ebenfalls perfekt ist, also $\chi(G) = \omega(G)$ gilt. Dafür genügt es, eine unabhängige Menge F zu finden, so dass $\omega(G_{V\setminus F}) < \omega(G)$ ist. Denn aus der Induktionsvoraussetzung folgt, dass $G_{V\setminus F}$ perfekt ist und mit $\chi(G_{V\setminus F}) = \omega(G_{V\setminus F})$ Farben gefärbt werden kann. Gleichzeitig ist $\omega(G_{V\setminus F}) \geq \omega(G) - 1$, weil F jede Clique in höchstens einem Knoten schneiden kann. Also kann $G_{V\setminus F}$ mit $\omega(G_{V\setminus F}) - 1$ Farben gefärbt werden. Durch Hinzufügen der Farbe F entsteht eine $\omega(G)$-Färbung von G.
Der Beweis erfolgt indirekt. Es wird angenommen, dass $G_{V\setminus F}$ für jede unabhängige Menge F aus G eine $\omega(G)$-Clique C_F enthält. Es wird $\omega(G) =: p$ gesetzt. Sei $x \in V$ beliebig. Dann sei $h(x)$ die Anzahl der Cliquen C_F, die x enthalten, das heißt $h(x) = |\{C_F \mid F \subseteq V \text{ unabhängig und } x \in C_F\}|$. Sei G_0 der Graph, der aus G entsteht, wenn jeder Knoten x mit $h(x)$ multipliziert wird. Wie oben bereits gezeigt wurde, ist dann $\alpha(G_0)\omega(G_0) \geq |V_{G_0}|$. Andererseits sind offensichtlich

$$|V_{G_0}| = \sum_{x \in G} h(x) = \sum_{F \subseteq V \text{ unabh.}} |C_F| = pf,$$

wobei f die Anzahl der unabhängigen Mengen in G ist, und

$$\begin{aligned}
&\omega(G_0) \leq \omega(G) = p, \\
&\alpha(G_0) = \max\{\sum_{x \in F} h(x) \mid F \text{ unabhängig in } G\} && \text{[Definition von } G_0\text{]} \\
&\quad = \max\{\sum_{F' \text{ unabh. in } G} |F \cap C_{F'}| \mid F \text{ unabh. in } G\} && \text{[Definition von } h(x)\text{]} \\
&\quad \leq \max\{\sum_{F' \text{ unabh. in } G, F' \neq F} 1 \mid F \text{ unabh. in } G\} && [F \cap C_F = \emptyset, |F \cap C_{F'}| \leq 1] \\
&\quad = f - 1.
\end{aligned}$$

Das liefert

$$pf = |V_{G_0}| \leq \omega(G_0)\alpha(G_0) \leq p(f - 1),$$

was offensichtlich nicht geht. Damit wurde gezeigt, dass es im Graphen G eine unabhängige Menge F gibt, die alle maximalen Cliquen trifft, also ist $\omega(G[V \setminus F]) < \omega(G)$. □

Satz 2.11 (starker Satz über perfekte Graphen)**.** *Ein Graph ist genau dann perfekt, wenn er weder einen ungeraden Kreis der Länge mindestens 5 noch das Komplement eines solchen Kreises als induzierten Subgraphen enthält.*

Beweis. Dies wurde von Claude Berge 1961 vermutet. Der Beweis wurde 2002 von Chudnovsky, Robertson, Seymour und Thomas bekanntgegeben und 2006 veröffentlicht [9]. □

2.2 Diskrete Mathematik

In diesem Abschnitt wird ein Satz von Chvátal bewiesen, der die Ganzzahligkeit der Lösungen gewisser Gleichungssysteme garantiert. Dieses Ergebnis wird in dem Beweis, der zeigt, dass ein Kernel immer existiert, verwendet. Es werden zunächst die notwendigen Definitionen und Lemmata bereitgestellt, bevor der eigentliche Satz formuliert und bewiesen wird.

Definition 2.12. Es sei $B = B(G)$ die $(m \times n)$ Inzidenzmatrix aller maximalen Cliquen versus Knoten eines ungerichteten Graphen $G = (V, E)$ mit $|V| = n$, das heißt ihre Zeilen sind gerade die charakteristischen Vektoren der maximalen Cliquen. Die Matrix B wird als *Cliquenmatrix* bezeichnet.

Bemerkung 2.13. Diese Cliquenmatrix ist bis auf das Vertauschen der Zeilen bzw. Spalten eindeutig. Ohne Einschränkung kann angenommen werden, dass B keine Nullspalte hat.

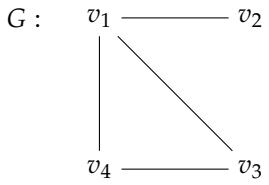


Abb. 2.3: Graph zu Beispiel 2.14

Beispiel 2.14. Der Graph G in Abbildung 2.14 hat zwei maximale Cliquen $C_1 = \{v_1, v_3, v_4\}$ und $C_2 = \{v_1, v_2\}$. Seine Cliquenmatrix ist

$$B = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

Definition 2.15. Es werden zwei Polyeder definiert:

$$P(B) = \{x \in \mathbb{R}^n \mid Bx \leq \mathbf{1}, x \geq 0\}$$

und

$$\begin{aligned} P_I(B) &= conv(\{x \mid x \in P(B), x \text{ ganzzahlig}\}) \\ &= conv\big(\{x \in \mathbb{R}^n \mid Bx \leq \mathbf{1}, x \in \{0,1\}^n\}\big). \end{aligned}$$

Lemma 2.16. *Seien G ein ungerichteter Graph und B dessen Cliquenmatrix. Dann gilt:*
x ist genau dann eine Ecke von $P_I(B)$, wenn x der charakteristische Vektor einer unabhängigen Menge von G ist.

Beweis. $\Rightarrow$: Aus der Linearen Optimierung ist bekannt, dass x eine Ecke von $P_I(B)$ ist, wenn x keine echte Konvexkombination von Punkten in $P_I(B)$ ist. Also ist jeder 0,1-Vektor, der die Ungleichungen erfüllt, eine Ecke. Nach der Konstruktion von B ist solch ein Vektor x der charakteristische Vektor einer Knotenmenge S. Angenommen, er träfe eine maximale Clique C_i in mehr als einem Knoten, dann gälte für die korrespondierende Zeile $B_{i.}x \geq 1 + 1 = 2 \not\leq 1$. Also wäre x nicht in dem Polyeder. Deshalb muss die Knotenmenge S unabhängig sein.
$\Leftarrow$: Sei x der charakteristische Vektor einer unabhängigen Menge von G. Diese Menge trifft jede maximale Clique in höchstens einem Knoten, also erfüllt x die Ungleichungen und ist als 0,1-Vektor eine Ecke. □

Für den Beweis des Satzes von Chvatál [10] wird noch ein Ergebnis der linearen Programmierung benötigt, das unter anderem von Edmonds [11] genutzt wurde.

Lemma 2.17. *[Trennungslemma von Edmonds]*
Seien S und T zwei beschränkte Polyeder in $\mathbb{R}^n$. Dann gilt:

$$S = T \Leftrightarrow \forall c \in \mathbb{Z}^n : \max_{x \in S} c^\top x = \max_{x \in T} c^\top x.$$

Beweis. Es ist nur $\Leftarrow$ zu beweisen. Angenommen, es ist $S \neq T$. Ohne Einschränkung kann $T \setminus S \neq \emptyset$ angenommen werden. Es wird zunächst gezeigt, dass es dann für jedes $y \in \mathbb{R}^n$ (also auch für alle $y \in T \setminus S$) ein $c \in \mathbb{R}^n$ und ein $x_0 \in S$ gibt, so dass $c^\top y > c^\top x_0 \geq c^\top x$ ist für alle $x \in S$. Dafür wird genutzt, dass das Minimierungsproblem

$$\min_{x \in S} \|y - x\|$$

eine optimale Lösung $x_0 \in S$ besitzt. (Denn die Funktion $x \mapsto \|x - y\|$ ist auf $\mathbb{R}^n$ stetig und nimmt demzufolge auf einem Kompaktum ihre Extremwerte an. Polyeder sind immer konvex und abgeschlossen. Hier sind sie zusätzlich beschränkt, also kompakt. Deshalb existiert eine Minimallösung $x_0 \in S$.) Sei y aus $\mathbb{R}^n \setminus S$ beliebig gewählt. Es wird $c := y - x_0$ gesetzt und nachgerechnet, dass c die gewünschten Eigenschaften hat. Als Erstes wird $c^\top x_0 < c^\top y$ gezeigt:

$$c^\top y - c^\top x_0 = c^\top (y - x_0) = (y - x_0)^\top (y - x_0) = \|y - x_0\|^2 > 0$$

Es wird nun ein beliebiger Punkt $x \in S$ betrachtet und

$$z(\lambda) = x_0 + \lambda(y - x_0),\ \lambda \in [0, 1]$$

gesetzt. Dann ist $z(\lambda)$ in S enthalten, und aufgrund der Wahl von x_0 gilt deshalb

$$\|c\|^2 = \|y - x_0\|^2 \leq \|y - z(\lambda)\|^2 = \|y - x_0 - \lambda(x - x_0)\|^2 = \|c - \lambda(x - x_0)\|^2.$$

Ausmultiplizieren ergibt

$$\|c\|^2 \leq \|c\|^2 - 2\lambda c^\top (x - x_0) + \lambda^2 \|x - x_0\|^2.$$

Wenn $\lambda \neq 0$ angenommen wird, dann gilt

$$0 \leq -2c^\top (x - x_0) + \lambda \|x - x_0\|^2.$$

Das liefert

$$c^\top x - c^\top x_0 = c^\top (x - x_0) \leq \frac{1}{2} \lim_{\lambda \to 0} \lambda \|x - x_0\|^2 = 0.$$

Damit wurde auch $c^\top x \leq c^\top x_0$ für alle $x \in S$ bewiesen.
Jetzt bleibt noch der Übergang von den reellen zu den ganzen Zahlen zu zeigen. Dafür genügt es, den Fall zu betrachten, dass der obige Vektor c mindestens eine irrationale Komponente hat. Da $\mathbb{Q}$ dicht in $\mathbb{R}$ ist, gibt es ein $c' \in \mathbb{Q}^n$ mit $c'^\top y > c'^\top x$ für alle $x \in S$. Die Multiplikation mit dem Hauptnenner $b > 0$ aller Komponenten von c' liefert den gewünschten Vektor $bc' \in \mathbb{Z}^n$. □

Satz 2.18. (Chvátal)
Sei G ein Graph mit Cliquenmatrix B. Dann gilt:
G ist genau dann perfekt, wenn $P(B) = P_I(B)$ ist, das heißt alle Ecken des Polyeders $P(B)$ sind ganzzahlig.

Beweis. $\Leftarrow$: Sei $P(B) = P_I(B)$. Seien $U \subseteq V$ beliebig, G_U der von U induzierte Teilgraph von G und u der charakteristische Vektor von U. Dann gilt

$$\alpha(G_U) = \max_{x \in P_I(B)} u^\top x = \max_{x \in P(B)} u^\top x = \min_{y^\top B \geq u^\top,\, y \geq 0} y^\top \mathbf{1}.$$

Die erste Gleichheit gilt, weil das Maximum immer in einer Ecke angenommen wird und die Ecken von $P_I(B)$ mit den unabhängigen Mengen korrespondieren (Lemma 2.16). Die zweite Gleichheit folgt mit $c = u$ aus dem Lemma von Edmonds (2.17) und die dritte Gleichheit aus der Dualität der Linearen Programmierung.
Es sei ein $y \geq 0$ gewählt, welches $y^\top B \geq u^\top$ und $\alpha(G_U) = y^\top \mathbf{1}$ erfüllt. (Ein solches y existiert aufgrund der obigen Gleichungen.) Damit gilt

$$|U| = u^\top u \leq y^\top B u \leq y^\top (\omega(G_U)\mathbf{1}) = \omega(G_U) y^\top \mathbf{1} = \omega(G_U)\alpha(G_U).$$

Also ist G nach dem Satz 2.10, der eine äquivalente Bedingung für die Perfektheit bereitstellt, perfekt.
$\Rightarrow$: Seien nun G perfekt und $c \in \mathbb{Z}^n$. Sei H der Graph, der entsteht, wenn jeder Knoten v mit $\max\{0, c_v\}$, multipliziert wird. Nach dem Lemma von Berge 2.8, ist der Graph H perfekt, weil er durch Multiplikation aus G hervorgeht. Es gelten folgende (Un)Gleichungen:

$$\begin{aligned}
\alpha(H) &= \max\{\sum_{v \in F} |\max\{0, c_v\}| \mid F \text{ unabhängige Menge in } G\} && =: \boldsymbol{\alpha_c(G)} \\
&= \max\left\{c^\top x \mid x \text{ char.Vektor einer unabh. Menge in } G\right\} \\
&= \max\{c^\top x \mid x \in P_I(B)\} && \text{[Lemma 2.16]} \\
&\leq \max\{c^\top x \mid x \in P(B)\} && [P_I(A) \subseteq P(A)] \\
&= \min\{y^\top \mathbf{1} \mid y^\top B \geq c^\top, y \geq 0\} && \text{[duales Problem]} \\
&\leq \min\{y^\top \mathbf{1} \mid y^\top B \geq c^\top, y \in \mathbb{N}^m\} && [M \subseteq N \Rightarrow \min M \geq \min N] \\
&= \min\{|\Pi_c| \mid \Pi_c \text{ ist eine Cliquenüberdeckung von } G, && =: \boldsymbol{\kappa_c(G)} \\
&\qquad \text{die jeden Knoten } v \text{ } \max\{0, c_v\}\text{-mal überdeckt}\} \\
&= \kappa(H)
\end{aligned}$$

Die letzte Gleichheit folgt daraus, dass einerseits jede Clique von H mit einer Clique von G korrespondiert, also $\kappa(H) \geq \kappa_c(G)$ ist, und andererseits $\kappa_c(G) \geq \kappa(H)$ ist, denn wenn ein Knoten v von G durch c_v Cliquen bedeckt wird, dann gibt es c_v Cliquen in H, von denen jede eine andere Kopie von v bedeckt. Der Graph H ist perfekt. Das heißt, es gilt $\alpha(H) = \kappa(H)$ und folglich ist $\max\{c^\top x \mid x \in P_I(B)\} = \max\{c^\top x \mid x \in P(B)\}$. Nach dem Trennungslemma von Edmonds 2.17 folgt $P_I(B) = P(B)$. □

2.3 Komplexitätstheorie: Die Klasse PPAD

Die folgenden Ausführungen orientieren sich an [12], [13] und [14].

Definition 2.19. Ein *Suchproblem* $\mathcal{S}$ ist eine Sprache, die aus Paaren (x, y) besteht. Das erste Element x ist eine Instanz des Problems, und das zweite Element y ist eine mögliche Lösung des Problems. Formal ist ein Suchproblem durch eine Relation $R_\mathcal{S}(x, y)$ definiert, so dass genau dann

$$R_\mathcal{S}(x, y) = 1$$

gilt, wenn y eine Lösung der Instanz x ist.
Alle Suchprobleme $\mathcal{S}$, zu denen es einen deterministischen Algorithmus gibt, der zu einer gegebenen Eingabe x in polynomialer Zeit ein y finden kann, so dass $R_\mathcal{S}(x, y) = 1$ ist, werden zur Klasse FP zusammengefasst.
Ein Suchproblem $\mathcal{S}$ ist genau dann in der Klasse FNP, wenn es einen effizienten Algorithmus $A_\mathcal{S}(x, y)$ und eine polynomiale Funktion $p_\mathcal{S}(\cdot)$ gibt, so dass die folgenden Aussagen gelten:

1. Wenn $A_\mathcal{S}(x, z) = 1$ ist, dann ist $R_\mathcal{S}(x, z) = 1$.
2. Wenn es ein y mit $R_\mathcal{S}(x, y) = 1$ gibt, dann gibt es ein z mit $|z| \leq p_\mathcal{S}(|x|)$, so dass $A_\mathcal{S}(x, z) = 1$ ist.

Salopp gesprochen ist ein Suchproblem in FNP, wenn es zu jeder Instanz x des Problems, die eine Lösung hat, eine polynomial kleine (d.h. polynomial in der Eingabelänge) und effizient verifizierbare Lösung gibt. Es gilt also FP $\subseteq$ FNP. Der Unterschied zwischen diesen beiden Klassen ist, dass ein Algorithmus für ein FNP-Problem nur eine Lösung y bestätigt, während ein Algorithmus für ein FP-Problem deren Wert finden muss.

Definition 2.20. Ein Suchproblem heißt *total*, wenn es zu jedem x ein y mit $R_\mathcal{S}(x, y) = 1$ gibt, das heißt, zu jeder Eingabe existiert garantiert eine Lösung. TFNP ist die Menge aller totalen Suchprobleme aus FNP.

Offensichtlich gilt FNP $\supseteq$ TFNP $\supseteq$ FP. Angenommen, es wäre leicht, eine Lösung zu finden, sobald man weiß, dass eine Lösung immer existiert. Dann würde FP $=$ TFNP gelten. Das ist eine der offenen Fragen der Komplexitätstheorie. Die herrschende Meinung geht von einer echten Inklusion aus. Eine positive Antwort würde P $=$ NP $\cap$ coNP implizieren, da Megiddo und Papadimitriou in [15] TFNP $=$ F(NP $\cap$ coNP) gezeigt haben. Mit anderen Worten, es gäbe für alle Probleme aus NP $\cap$ coNP einen „Polynomialzeitalgorithmus“. In diesem Fall wäre beispielsweise Faktorisierung ein Problem in P.

Definition 2.21. Ein Suchproblem $\mathcal{S}$ in FNP, mit dem $A_{\mathcal{S}}(x,y)$ und $p_{\mathcal{S}}$ assoziiert sind, ist auf ein Suchproblem $\mathcal{T}$ in FNP, mit dem $A_{\mathcal{T}}(x,y)$ und $p_{\mathcal{T}}$ assoziiert sind, *in polynomialer Zeit reduzierbar*, wenn es zwei effizient berechenbare Funktionen f und g gibt, so dass folgende Bedingungen erfüllt sind:

1. Wenn x eine Eingabe von $\mathcal{S}$ ist, dann ist $f(x)$ eine Eingabe von $\mathcal{T}$.
2. Wenn $A_{\mathcal{T}}(f(x),y) = 1$ ist, dann ist $A_{\mathcal{S}}(x,g(y)) = 1$.
3. Wenn $R_{\mathcal{T}}(f(x),y) = 0$ ist für alle y, dann ist $R_{\mathcal{S}}(x,y) = 0$ für alle y.

In diesem Fall wird $\mathcal{S} \leq_p \mathcal{T}$ geschrieben. Wenn $\mathcal{S}$ und $\mathcal{T}$ in TFNP liegen, dann ist das dritte Kriterium natürlich obsolet.
Ein Suchproblem $\mathcal{S}$ ist FNP-*schwer*, falls sich alle Probleme aus FNP in polynomialer Zeit auf dieses reduzieren lassen. Wenn $\mathcal{S}$ ebenfalls in FNP liegt, dann wird $\mathcal{S}$ als FNP-*vollständig* bezeichnet. Die beiden letzten Definitionen gelten analog für alle Teilklassen von FNP.

Papadimitriou hat in [12] eine nützliche und sehr elegante Klassifikation der Probleme in TFNP vorgeschlagen: TFNP ist eine Klasse, die in der Komplexität manchmal als „semantisch“ bezeichnet wird, weil sie kein generisch vollständiges Problem enthält. Deshalb wird die Komplexität der totalen Funktionen typischerweise mit „syntaktischen“ Teilklassen von TFNP untersucht. Die Idee ist die folgende: Wenn ein Problem total ist, dann sollte es einen Beweis geben, der zeigt, dass es immer eine Lösung hat. Die Probleme in TFNP werden zu Klassen zusammengefasst, die mit dem Typ des Totalitätsbeweises korrespondieren. Es hat sich herausgestellt, dass sich viele Probleme nach den folgenden Existenzbeweisen gruppieren lassen:

- „Wenn ein gerichteter Graph einen unbalancierten Knoten hat - ein Knoten, dessen Eingangsgrad sich von dessen Ausgangsgrad unterscheidet - dann muss er einen anderen unbalancierten Knoten haben.“ Dieses Gleichheitsargument für Digraphen führt zur Klasse PPAD.
- „Wenn ein ungerichteter Graph einen Knoten mit ungeradem Knotengrad hat, dann muss er einen anderen Knoten mit ungeradem Grad enthalten.“ Das ist das Gleichheitsargument für ungerichtete Graphen und führt zu einer weiteren Klasse, PPA genannt.
- „Jeder gerichtete azyklische Graph hat eine Senke.“ Die korrespondierende Klasse ist PLS und enthält Probleme, die durch lokale Suche lösbar sind.
- „Wenn eine Funktion n Elemente auf $n-1$ Elemente abbildet, dann gibt es eine Kollision.“ Das ist das Schubfachprinzip, und die zugehörige Klasse heißt PPP.

Ist es nicht trivial, in einem Graphen einen unbalancierten Knoten zu finden, eine Quelle in einem gerichteten azyklischen Graphen oder eine Kollision in einer Funktion? Das hängt sehr davon ab, wie dieser Graph oder diese Funktion gegeben sind. Wenn die Eingabe (Graph oder Funktion) implizit durch eine Schaltung erfolgt, dann ist es möglich, dass das Problem nicht so einfach ist, wie es klingt. Es wird das folgende Problem betrachtet, das mit dem Gleichheitsargument für gerichtete Graphen korrespondiert:

END OF THE LINE: Gegeben seien zwei Schaltungen S und P, jede mit n Eingabe- und n Ausgabebits, so dass $S(P(0^n)) \neq 0^n = P(S(0^n))$ gilt. Finde eine Eingabe $x \in \{0,1\}^n$, so dass $P(S(x)) \neq x$ oder $S(P(x)) \neq x \neq 0^n$ ist.

Hierbei spezifizieren P und S einen gerichteten Graphen mit der Knotenmenge $\{0,1\}^n$ wie folgt: Es gibt genau dann eine Kante vom Knoten u zum Knoten v, wenn $S(u) = v$ und $P(v) = u$ ist[1]. Insbesondere sind die Eingangs- und Ausgangsgrade aller Knoten höchstens eins. Wegen $S(P(0^n)) \neq 0^n = P(S(0^n))$ hat der Knoten 0^n den Eingangsgrad null und den Ausgangsgrad eins, ist also unbalanciert. Nach dem Gleichheitsargument für gerichtete Graphen muss es einen anderen unbalancierten Knoten geben, und es wird versucht, einen unbalancierten Knoten ungleich 0^n zu finden.
Der naheliegende Ansatz, END OF THE LINE zu lösen, ist, dem Pfad, der bei 0^n beginnt, bis zur Senke am anderen Ende zu folgen. Diese Vorgehensweise kann unter Umständen exponentiell in der Zeit sein, denn es gibt in dem Graphen 2^n Knoten, die durch S und P spezifiziert werden. Um das Problem schneller zu lösen, wird versucht durch scharfes Ansehen der Details der Schaltungen S und P einen Weg finden, um den Digraphen „ineinanderzuschieben“.

Definition 2.22. Die Klasse PPAD wird als die Menge aller Probleme in TFNP definiert , die in Polynomialzeit auf END OF THE LINE reduziert werden können. Demzufolge ist END OF THE LINE ein PPAD-vollständiges Problem.

Daskalakis [13] glaubt, dass PPAD eine Klasse mit schweren Problemen ist. Aber da PPAD zwischen FP und FNP liegt, lässt sich das vermutlich nicht beweisen, ohne gleichzeitig FP $\neq$ FNP zu zeigen. Einen solchen Beweis gibt es nicht. Es wird aus denselben Gründen wie für FNP angenommen, dass PPAD eine schwere Klasse ist (auch wenn diese Überzeugung für die Klasse PPAD etwas schwächer ist, da sie in FNP enthalten ist): PPAD enthält viele Probleme, für die Forscher schon seit Jahrzehnten versuchen, effiziente Algorithmen zu entwickeln. Dazu gehören unter anderem BROUWER, welches die Berechnung für approximierte Brouwersche Fixpunkte betrachtet, END OF THE LINE (Wie kann gehofft werden, exponentiell große Pfade in jedem implizit gegebenen Graphen ineinanderschieben zu können?), SPERNER, das mit Sperners Lemma zusammenhängt, und NASH, das auf das Nash-Gleichgewicht für Mehrpersonenspiele zurückgeht.

[1]So erklären sich auch die Bezeichnungen: P berechnet den Vorgänger („predecessor“) von x und S den Nachfolger („successor“)

3 Berechnung eines Kernels einer clique-azyklischen Superorientierung eines perfekten Graphen mit dem Algorithmus von Scarf

Es wird ein Lemma von Herbert E. Scarf [2] eingeführt. Dessen Beweis ist algorithmisch. Anschließend wird der Satz, welcher besagt, dass jede clique-azyklische Superorientierung eines perfekten Graphen einen Kernel besitzt, mit der Methode von Aharoni und Holzman [4] bewiesen. Deren Beweis basiert auf dem Algorithmus von Scarf. Das Besondere an diesem Algorithmus ist, dass er endlich ist und ohne Fixpunktsätze auskommt. Er verwendet nur Pivotschritte, wie sie aus der linearen Optimierung bekannt sind, und ordinale Vergleiche. Sowohl der Algorithmus von Scarf als auch der Algorithmus zur Kernelberechnung werden an Beispielen veranschaulicht. Zum Schluss werden einige Eigenschaften des Algorithmus zur Kernelberechnung festgehalten.

3.1 Lemma von Scarf

Satz 3.1 (Lemma von Scarf)**.** *Seien $m < n$ sowie B und C reelle $m \times n$-Matrizen, so dass die ersten m Spalten von B die Einheitsmatrix E_m bilden und $c_{ii} \leq c_{ik} \leq c_{ij}$ ist für alle $i, j \in [m]$ mit $i \neq j$ und für alle $k \in [n] \setminus [m]$. Sei b ein nicht-negativer Vektor in $\mathbb{R}^m$, so dass die Menge $\{x \in \mathbb{R}^n_{\geq 0} \mid Bx = b\}$ beschränkt ist. Dann gibt es eine Teilmenge J der Größe m von $[n]$, so dass*

(a) $\exists x \in \mathbb{R}^n_{\geq 0}$ *mit* $Bx = b$ *und* $\forall j \notin J : x_j = 0$ *und*

(b) $\forall k \in [n]\ \exists i \in [m]$*, so dass* $c_{ik} \leq u_i$ *ist, wobei wir* $u_i = \min\{c_{ij} \mid j \in J\}$ *setzen und als den* Zeilenminimierer *der i-ten Zeile von J bezeichnen. (Hierfür werden die Spalten von J als $m \times m$ - Matrix aufgefasst.)*

Bezeichnung 3.2. (i) Eine Teilmenge $J \subset [n]$ mit der Mächtigkeit m, die (a) genügt, ist eine *zulässige Basis* für das Gleichungssystem $Bx = b$. Diese wird mit $\mathcal{B}_{\mathbf{z}}$ bezeichnet.

(ii) Eine Teilmenge $J \subseteq [n]$ mit der Eigenschaft (b) wird als *unterordnend* bezeichnet. Wenn zusätzlich $|J| = m$ gilt, wird sie eine *ordinale Basis* genannt und mit $\mathcal{B}_{\mathbf{o}}$ bezeichnet.

Beispiel 3.3. Folgende 3×7 - Matrizen B und C und Vektor b erfüllen die Bedingungen des Satzes:

$$B = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad C = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} 0 & 8 & 7 & 2 & 1 & 6 & 5 \\ 8 & 0 & 7 & 6 & 2 & 1 & 5 \\ 8 & 7 & 0 & 6 & 5 & 2 & 1 \end{pmatrix} \end{array}$$

$\mathcal{B}_z = \{1, 2, 3\}$ ist eine zulässige Basis für (B, b). Und $\{3, 7\}$ ist eine unterordnende Menge für C, aber keine ordinale Basis.

Bezeichnung 3.4. Im Folgenden sei $[m] := \{1, \ldots, n\}$, und mit $\mathcal{B}_o + i - j$ ist $(\mathcal{B}_o \cup \{i\}) \setminus \{j\}$ gemeint. Für andere Mengen gilt dies analog.

Beweis des Lemmas von Scarf 3.1. Die *Beweisidee* geht auf eine Prozedur zurück, die Lemke und Howson [16] zur Lösung eines Nichtnullsummenspiels mit zwei Personen entwickelt haben. Da diese nach endlich vielen Schritten terminiert, muss es mindestens eine Lösung geben.
Es wird mit einer zulässigen Basis $\mathcal{B}_z$ und einer ordinalen Basis $\mathcal{B}_o$ gestartet, die (a) resp. (b) erfüllen und sich in genau einer Spalte unterscheiden, das heißt $|\mathcal{B}_z \cap \mathcal{B}_o| = m - 1$. Nun wird entweder ein Pivotschritt für $\mathcal{B}_z$ oder ein anderer, noch zu definierender, Schritt für $\mathcal{B}_o$ so durchgeführt, dass sich die neuen Teilmengen wieder in genau einer Spalte unterscheiden und weiterhin eine zulässige resp. ordinale Basis sind. Hierbei ist immer eindeutig, welcher Pivotschritt als nächstes genommen wird. Dies wird solange wiederholt, bis der Algorithmus mit $\mathcal{B}_z = \mathcal{B}_o$ endet, was gerade der gesuchten Teilmenge J entspricht.
Entartung soll ausgeschlossen werden. Deshalb wird angenommen, dass B und b so beschaffen sind, dass alle Variablen, die mit den m Spalten einer zulässigen Basis assoziiert sind, streng positiv sind. Außerdem wird für die Matrix C gefordert, dass keine zwei Elemente derselben Zeile gleich sind. Eine solche Matrix wird als *ordinal generisch* bezeichnet. Diese beiden Annahmen können gegebenenfalls durch kleine Störungen (Perturbationen) der Matrizen B und C und des Vektors b arrangiert werden. Wenn die Perturbationen klein genug sind, dann sind die Basen, die durch die perturbierten Daten bestimmt werden, auch Basen für die unperturbierten Daten.

(1) *Startmengen:* Offensichtlich ist $\mathcal{B}_z = [m]$ eine zulässige Basis für $Bx = b$, aber keine ordinale Basis für C. Für $i \in \mathcal{B}_z$ beliebig sei $c_{ij_i} = \max\{c_{ik} \mid k = m+1, \ldots, n\}$. Dann ist die Menge $\mathcal{B}_o = [m] - i + j_i$ eine ordinale Basis von C, denn für alle Spalten $k \in \{i, m+1, \ldots, n\}$ ist $c_{ik} \leq c_{ij_i} = u_i$ (wegen der Wahl von j_i) und für alle Spalten $k \in [m] - i$ ist $c_{kk} = \min\{c_{kj} \mid j \in \mathcal{B}_o\} = u_k$, also insbesondere $c_{kk} \leq u_k$.
Im Folgenden wird o.B.d.A. $i = 1$ und somit $j_i = j_1$ angenommen. Das heißt, es wird mit den Mengen $\mathcal{B}_z = [m]$ und $\mathcal{B}_o = \{j_1, 2, \ldots, m\}$ sowie $c_{1j_1} = \max\{c_{1k} \mid k = m+1, \ldots, n\}$ gestartet.

(2) Der zulässige Pivotschritt wird nur genannt, aber nicht bewiesen, da er als bekannt vorausgesetzt wird.

Lemma 3.5 (zulässiger Pivotschritt)**.** *Aus der linearen Programmierung ist bekannt, dass jede beliebige Spalte außerhalb der zulässigen Basis $\mathcal{B}_z$, die keine Nullspalte ist, in die zulässige Basis $\mathcal{B}_z$ aufgenommen werden kann und als Ergebnis des Pivotschritts genau eine Spalte eliminiert wird, vorausgesetzt, das Problem ist nicht entartet und die Menge der Bedingungen ist beschränkt.*

(3) Nun wird der zweite Schritt eingeführt, der in Analogie zum zulässigen Pivot als ordinaler Pivotschritt bezeichnet wird.

Lemma 3.6 (ordinaler Pivotschritt). *Seien $\mathcal{B}_o = \{j_1, \ldots, j_m\}$ eine ordinale Basis für C, j_1 eine beliebige aus $\mathcal{B}_o$ zu entfernende Spalte, $\mathcal{B}_o \setminus \{j_1\} \nsubseteq [m]$ (d.h. die in $\mathcal{B}_o$ verbleibenden Spalten sind nicht in den ersten m Spalten von C enthalten) und keine zwei Elemente einer Zeile von C gleich. Dann gibt es genau eine Spalte $j^* \neq j_1$, so dass $\mathcal{B}_o + j^* - j_1$ wieder (b) genügt, also eine ordinale Basis ist.*

Beweis. Existenz: Jede Spalte von $\mathcal{B}_o$ enthält genau einen Zeilenminimierer u_i. Andernfalls gäbe es eine Spalte $j \in \mathcal{B}_o$ ohne Zeilenminimierer und es wäre $u_i < c_{ij}$ für alle $i \in [m]$, da keine zwei Einträge einer Zeile von C gleich sind. Damit wäre (b) nicht erfüllt.
Die Spalte j_1 wird aus $\mathcal{B}_o$ entfernt. Nun werden die m Zeilenminimierer u'_i für die $m-1$ in $\mathcal{B}_o$ verbleibenden Spalten gebildet. Offensichtlich ist $u'_i \geq u_i$ für alle $i \in [m]$. Aufgrund der vorhergehenden Beobachtung enthält genau eine Spalte zwei Zeilenminimierer, wovon einer neu und der andere bereits für $\mathcal{B}_o$ ein Zeilenminimierer ist. Seien u'_{i_n} der neue und u'_{i_a} der alte Zeilenminimierer.
Ferner sei $\mathcal{M} = \{k \in [n] \mid c_{ik} > u'_i \quad \forall i \in [m] \setminus \{i_a\}\}$ die Menge der Spalten aus C, die (b) für alle Zeilen ungleich i_a nicht erfüllen. Es soll j^* geschickt aus $\mathcal{M}$ gewählt werden.
$\mathcal{M}$ ist nicht leer, weil die Spalte i_a in $\mathcal{M}$ enthalten ist. Das liegt an den Ungleichungen, die für die Einträge von C gelten und daran, dass die in $\mathcal{B}_o$ verbleibenden Spalten nicht in den ersten m Spalten von C enthalten sind.
Da $u'_i \geq u_i$ gilt, ist für alle Spalten k aus $\mathcal{M}$ auch $c_{ik} > u_i$ für alle $i \in [m] \setminus \{i_a\}$. Deshalb muss für diese $c_{i_a k} \leq u_{i_a}$ gelten, weil $\mathcal{B}_o$ nach Voraussetzung (b) erfüllt. Das heißt, wenn j^* aus $\mathcal{M}$ gewählt wird, dann liefert j^* automatisch den neuen Minimierer der Zeile i_a. Gleichzeitig muss für alle k aus $\mathcal{M}$ auch $c_{i_a j^*} = u_{i_a} \geq c_{i_a k}$ sein, damit (b) weiter gilt. Dies wird gerade von $c_{i_a j^*} = \max\{c_{i_a k} \mid k \in \mathcal{M}\}$ geleistet. Da C nicht entartet ist, ist das Maximum eindeutig bestimmt.
Eindeutigkeit: Es ist noch zu zeigen, dass j_1 durch keine andere Spalte als obiges j^* ersetzt werden kann. Es sei

$$\begin{pmatrix} c_{1j_1} & c_{1j_2} & \ldots & c_{1j_n} \\ c_{2j_1} & c_{2j_2} & \ldots & c_{2j_n} \\ \vdots & & & \\ c_{nj_1} & c_{nj_2} & \ldots & c_{nj_n} \end{pmatrix}.$$

die zu $\mathcal{B}_o$ gehörende quadratische Teilmatrix. Ohne Einschränkung wird angenommen, dass die Zeilenminimierer auf der Diagonalen liegen, d.h. $u_i = c_{ij_i}$, und dass c_{1j_2} das zweitkleinste Element in der ersten Zeile ist. Nach dem Entfernen der ersten Spalte hat die zweite Spalte deshalb die zwei Zeilenminimierer c_{1j_2} und c_{2j_2}. Wenn eine Spalte neu aufgenommen wird, müssen die Zeilenminimierer $c_{3j_3}, \ldots, c_{mj_m}$ ebensolche bleiben. Andernfalls gäbe es eine Spalte ohne Zeilenminimierer, was nicht sein darf, wie oben bereits gezeigt wurde. Weil C nicht entartet ist, ist $u_i < c_{ij^*}$ für $i = 2, \ldots, m$. Es gibt theoretisch zwei Möglichkeiten für die Wahl von j^*, damit wieder jede Spalte genau einen Zeilenminimierer enthält:

(i) $c_{1j^*} < c_{1j_2}$ und $c_{2j^*} > c_{2j_2}$: Der Zeilenminimierer u_2 bleibt also ebenfalls gleich und es wird $u_1 = c_{1j^*}$ der neue Zeilenminimierer. Es soll weiterhin (b) gelten, das heißt, es gebe für jedes k ein i mit $u_i \geq c_{ik}$. Dies ist für $k = j_1$ nur für $i = 1$ möglich, da ursprünglich j_1 in $\mathcal{B}_o$ und $u_1 = c_{1j_1}$ waren. Dann gilt $c_{1j^*} \geq c_{1j_1}$.
Es wurde bereits gezeigt, dass $c_{ij^*} > u_i$ sein muss für alle $i \neq 1$. Also muss gleichzeitig $c_{1j_1} \geq c_{1j^*}$ sein, weil j_1 ursprünglich in $\mathcal{B}_b$ war. Weil in C keine zwei Einträge einer Zeile gleich sind, folgt hieraus $j_1 = j^*$. Diese Variante entfällt nach Voraussetzung.

(ii) $c_{1j^*} > c_{1j_2}$ und $c_{2j^*} < c_{2j_2}$: Nun ist c_{1j_2} der Zeilenminimierer in der Spalte j_2 und c_{2j^*} der in j^*. Das heißt, es wird eine Spalte j^* gesucht, in der $c_{ij^*} > c_{ij_i}$ für jedes $i \in [m] \setminus \{2\}$ ist. Damit (b) von $\mathcal{B}_o + j^* - j_1$ erfüllt wird, ist j^* so aus diesen Spalten wählen, dass c_{2j^*} maximiert wird. Analog zum Existenzbeweis muss das Minimum der Menge $\{c_{2j} \mid j \in \mathcal{B}_o + j^* - j_1\}$ in der Spalte j^* liegen, das heißt, $c_{2j^*} = \min\{c_{2j} \mid j \in \mathcal{B}_o + j^* - j_1\}$. Dieses j^* ist gerade die im Existenzbeweis beschriebene Spalte. □

Da die neue Spalte eindeutig ist, liefert der Existenzbeweis bereits die Rechenregeln für den ordinalen Pivotschritt. Selbiger ist offensichtlich umkehrbar und einfach durchzuführen, weil er nur ordinale Vergleiche enthält.

(4) *Algorithmus:* Gegeben seien die Startmengen $\mathcal{B}_z = [m]$ und $\mathcal{B}_o = \{j_1, 2, \ldots, m\}$. Da verlangt wird, dass sich $\mathcal{B}_z$ und $\mathcal{B}_o$ immer in genau einer Spalte unterscheiden, gibt es zwei Möglichkeiten fortzufahren. Entweder wird j_1 durch einen Pivotschritt in $\mathcal{B}_z$ aufgenommen oder mit einem ordinalen Pivotschritt aus $\mathcal{B}_o$ entfernt. Letzteres ist nicht möglich, weil die in $\mathcal{B}_o$ verbleibenden Spalten in den ersten m Spalten von C enthalten wären. Also wird j_1 mit einem zulässigen Pivot in $\mathcal{B}_z$ aufgenommen. So entstehen die beiden neuen Teilmengen $\mathcal{B}_o = \{j_1, 2, \ldots, m\}$ und $\mathcal{B}_z = \{j_1, 1, \ldots, m\} \setminus \{j_*\}$ für ein $j_* \in [m]$. Nun kann entweder j_* aus $\mathcal{B}_o$ entfernt oder durch einen zulässigen Pivotschritt in $\mathcal{B}_z$ aufgenommen werden. Letzteres ist offensichtlich nicht sinnvoll, weil dann j_1 wieder aus $\mathcal{B}_z$ entfernt und zu den Ausgangsmengen zurückgekehrt würde. Also wird j_* mit einem ordinalen Pivotschritt aus $\mathcal{B}_o$ entfernt. Da j_1 in $\mathcal{B}_o$ verbleibt, kann dieser durchgeführt werden. Nun gibt es wieder zwei Möglichkeiten fortzufahren, wovon eine ausgeschlossen wird, weil diese zuletzt benutzt wurde, um zu den jetzigen Mengen $\mathcal{B}_z$ und $\mathcal{B}_o$ zu gelangen. Auf diese Art wird fortgefahren.

(5) *Der Algorithmus terminiert:* Es wird eine (ungerichtete) END-OF-THE-LINE Argumentation genutzt, um zu zeigen, dass der Algorithmus terminiert. Dafür werden zunächst eine neue Bezeichnung und eine andere Formulierung des ordinalen Pivotschritts eingeführt:

Eine Spaltenmenge $J \subseteq [n]$ für die Matrix C heißt *unterordnend*, wenn es für jede Spalte $k \in [n]$ einen Zeilenindex $i \in [m]$ gibt, so dass $c_{ik} \leq c_{ij}$ für alle $j \in J$ ist. Offensichtlich ist eine Teilmenge $J' \subseteq J$ einer unterordnenden Menge J ebenfalls unterordnend. Und eine ordinale Basis $\mathcal{B}_o$ ist eine unterordnende Menge mit der Mächtigkeit m.

Mit dieser Bezeichnung und den Überlegungen, die in (1) zu den Startmengen angestellt wurden, kann der obige ordinale Pivotschritt (Lemma 3.6) wie folgt umformuliert werden:

(3′) *ordinaler Pivotschritt:* Sei $\mathcal{B}$ eine unterordnende Menge der Größe $m - 1$ für die Matrix C. Falls $\mathcal{B}$ nicht in $[m]$ enthalten ist, dann gibt es genau zwei Elemente $j \in [n] \setminus \mathcal{B}$, so dass $\mathcal{B} + j$ für C eine ordinale Basis ist. Wenn $\mathcal{B} \subset [m]$ ist, dann gibt es genau ein solches j.

Es wird ein bipartiter Graph Γ mit den Seiten $\mathcal{Z}$ und $\mathcal{O}$ gebildet. Hierbei seien $\mathcal{Z}$ die Menge aller zulässigen Basen $\mathcal{B}_z$ mit $1 \in \mathcal{B}_z$ und $\mathcal{O}$ die Menge aller ordinalen Basen $\mathcal{B}_o$ mit $1 \notin \mathcal{B}_o$. Zwei Elemente $O \in \mathcal{O}$ und $Z \in \mathcal{Z}$ seien durch eine Kante miteinander verbunden, wenn $Z \setminus O = \{1\}$ ist.
Nun wird eine Menge $Z \in \mathcal{Z}$ betrachtet, die nicht unterordnend ist und positiven Knotengrad hat. Das heißt, es gibt eine ordinale Basis $O \in \mathcal{O}$, die mit Z verbunden ist, also $Z \setminus O = \{1\}$ erfüllt. Dann ist die Menge $\mathcal{B} = Z - 1$ als Teilmenge von O ebenfalls unterordnend. Angenommen $\mathcal{B}$ ist nicht in

$[m]$ enthalten. Dann werden mit dem ordinalen Pivotschritt (2′) zwei ordinale Basen $\mathcal{B} + j_1$ und $\mathcal{B} + j_2$ gefunden, die mit Z verbunden sind. Also hat Z den Knotengrad zwei. Analog lässt sich zeigen, dass $Z = [m]$ in Γ den Grad eins hat.
Als nächstes wird die Menge $O \in \mathcal{O}$ betrachtet, die keine zulässige Basis ist und positiven Knotengrad hat. Sei $Z \in \mathcal{Z}$ ein Nachbar von O und $\{o\} = O \setminus Z$. Nach der Definition des Pivotschritts gibt es ein eindeutiges Element z aus Z, so dass $Z' = Z + o - z$ eine zulässige Basis ist. Wenn $z = 1$ wäre, dann wäre $Z' = O$ eine zulässige Basis, was der Wahl von O widerspricht. Also ist $z \neq 1$, und Z' ist das einzige Element ungleich Z, welches mit O verbunden ist.
Bisher wurde gezeigt, dass bis auf $[m]$ jeder Knoten von Γ, der nicht gleichzeitig eine zulässige Basis und unterordnend ist, den Knotengrad zwei oder Null hat. Der Knoten $[m]$ ist vom Grad eins.
Mit ähnlichen Argumenten lässt sich zeigen, dass ein Knoten von Γ, der gleichzeitig eine zulässige und ordinale Basis ist, den Grad eins hat, falls solch ein Knoten überhaupt existiert:
Sei $Z \in \mathcal{Z}$ und unterordnend. Bekanntermaßen ist die Menge $[m]$ nicht unterordnend. Also kann $Z - 1$ nicht in $[m]$ enthalten sein. Zusätzlich ist $Z - 1$ als Teilmenge von Z ebenfalls unterordnend. Der ordinale Pivotschritt besagt, dass es genau zwei Elemente $j_1 \neq j_2 \in [n] \setminus Z - 1$ gibt, so dass $Z - 1 + j_1$ und $Z - 1 + j_2$ ordinale Basen sind. Offensichtlich ist Z eine dieser Mengen. Ohne Einschränkung sei $j_1 = 1$. Somit ist Z nur mit $Z - 1 + j_2 \in \mathcal{O}$ verbunden und hat ergo den Grad eins.
Sei $O \in \mathcal{O}$ und eine zulässige Basis. Wenn 1 mit einem Pivotschritt in O aufgenommen wird, dann wird dafür ein eindeutiges Element o aus O entfernt. So entsteht die zulässige Basis $O + 1 - o \in \mathcal{Z}$, die mit O verbunden ist. Also ist O ebenfalls vom Grad eins.
Da die Zusammenhangskomponente, die $[m]$ enthält, ein Pfad ist, muss sie bei einem anderen Knoten vom Grad eins enden. Das ist gerade ein Knoten, der sowohl eine zulässige als auch eine ordinale Basis ist. □

3.1.1 Algorithmus von Scarf

Der Beweis des Satzes 3.1 liefert den folgenden Algorithmus, der ab jetzt als *Algorithmus von Scarf* bezeichnen werden soll. Der zulässige Pivotschritt wird nicht ausformuliert, weil dieser als bekannt vorausgesetzt wird.

Algorithmus von Scarf:

Eingabe: Matrizen B und C sowie Vektor b wie im Satz von Scarf
$\mathcal{B}_z := \{1, \ldots, m\}$ zulässige Basis;
$c_{1j_1} = \max\{c_{1(m+1)}, \ldots, c_{1n}\}$;
$\mathcal{B}_o := \{j_1, 2, \ldots, m\}$ ordinale Basis;
for i=1 to m, i++
$\{u_i = \min\{c_{ij} \mid j \in \mathcal{B}_o\}$;
$\}$
While $\mathcal{B}_z \neq \mathcal{B}_o$
$\{k := \mathcal{B}_o \setminus \mathcal{B}_z$;
Nimm k mit einem zulässigen Pivot in $\mathcal{B}_z$ auf, entferne dabei p, d.h. $\mathcal{B}_z = \mathcal{B}_z + k - p$;
If $\mathcal{B}_z \neq \mathcal{B}_o$ Then
{ Ordinaler Pivotschritt für $\mathcal{B}_o$:

$\mathcal{B}_o = \mathcal{B}_o - p$;
for i=1 to m, i++
$\{u'_i = \min\{c_{ij} \mid j \in \mathcal{B}_o\}$;
$\}$
$\exists$ Spalte $q \in \mathcal{B}_o$ mit zwei Zeilenminimierern (ZM):
$u'_{i_n} \in q$ mit $u'_{i_n} \neq u_{i_n}$ ist der neue ZM;
$u_{i_a} = u'_{i_a} \in q$ ist der alte ZM;
Setze $\mathcal{M} = \{k \in [n] \mid c_{ik} > u'_i \ \forall i \in [m] - i_a\}$;
Bestimme $\max\{c_{i_a j} \mid j \in \mathcal{M} = c_{i_a r}\}$;
$\mathcal{B}_o = \mathcal{B}_o + r$;
$u_{i_n} = u'_{i_n}$;
$u_{i_a} = c_{i_a r}$;
$\}$
$\}$
Ausgabe: $J = \mathcal{B}_z$

3.1.2 Beispiel für Scarfs Algorithmus

Es wird das obige Beispiel (3.3) aufgegriffen und für diese Daten der Algorithmus von Scarf durchgeführt. Gegeben sind folgende Matrizen bzw. Vektoren:

$$B = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad C = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} 0 & 8 & 7 & 2 & 1 & 6 & 5 \\ 8 & 0 & 7 & 6 & 2 & 1 & 5 \\ 8 & 7 & 0 & 6 & 5 & 2 & 1 \end{pmatrix} \end{array}$$

In diesem Fall sind $m = 3$ und $n = 7$.

Es werden die Startmengen $\mathcal{B}_z$ und $\mathcal{B}_o$ gesucht und die zugehörigen Zeilenminimierer u_1, u_2 und u_3 bestimmt, die in dem Vektor u zusammengefasst werden:
Es ist $\max\{c_{1(m+1)}, \ldots, c_{1n}\} = \max\{c_{14}, \ldots, c_{17}\} = \max\{2,1,6,5\} = 6 = c_{16}$. Also wird $j_1 = 6$ in $\mathcal{B}_o$ aufgenommen, das heißt $\mathcal{B}_o = \{2,3,6\}$.
Der erste Zeilenminimierer ist $u_1 = \min\{c_{1j} \mid j \in \mathcal{B}_o\} = \min\{c_{12}, c_{13}, c_{16}\} = \min\{8,7,6\} = 6$. Analog werden $u_2 = 0$ und $u_3 = 0$ gefunden. Die Startmengen sind also:

$$\mathcal{B}_z = \{1,2,3\} \qquad \mathcal{B}_o = \{6,2,3\}$$
$$u = (6,0,0),$$

Anmerkung: Wenn eine Spalte $j \in [m] = [3]$ in $\mathcal{B}_o$ enthalten ist, ist aufgrund der Struktur der Matrix C automatisch $u_j = c_{jj}$.

(1) Es ist $\{6\} = \mathcal{B}_o \setminus \mathcal{B}_z$. Also wird $k = 6$ mit einem Pivotschritt in $\mathcal{B}_z$ aufgenommen. Da die sechste Spalte von B bereits einem Einheitsvektor entspricht, kann die Pivotzeile abgelesen werden. Das heißt, dass $p = 2$ aus $\mathcal{B}_z$ entfernt wird. Also ist nun

$$\mathcal{B}_z = \{1, 6, 3\}$$

Nun wird ein ordinaler Pivotschritt für $\mathcal{B}_o$ durchgeführt: Zunächst wird $\mathcal{B}_o = \mathcal{B}_o - p = \mathcal{B}_o - 2 = \{6, 3\}$ gebildet. Die zugehörigen neuen Zeilenminimierer sind $u'_i = \min\{c_{ik} \mid k \in \mathcal{B}_o\}$, also ist $u' = (c_{16}, c_{26}, c_{33}) = (6, 1, 0)$. Dann sind u'_1 der alte und u'_2 der neue Zeilenminimierer in der sechsten Spalte. Die Menge $\mathcal{M} = \{k \in [7] \mid c_{2k} > 1, c_{3k} > 0\} = \{1, 4, 5, 7\}$ wird gebildet und $\max\{c_{1k} \mid k \in \mathcal{M}\} = \max\{0, 2, 1, 5\} = 5 = c_{17}$ bestimmt. Anschließend wird $r = 7$ in die ordinale Basis $\mathcal{B}_o$ aufgenommen. Dann sind

$$\begin{aligned} \mathcal{B}_o &= \{6, 7, 3\} \\ u &= (5, 1, 0) \end{aligned}$$

(2) Es wird $k = 7$ in $\mathcal{B}_z$ pivotiert. Das Tableau $Bx = b$ wird dabei wieder nicht verändert, und es kann abgelesen werden, dass $p = 3$ aus $\mathcal{B}_z$ entfernt wird:

$$\mathcal{B}_z = \{1, 6, 7\}$$

Anschließend wird $p = 3$ aus $\mathcal{B}_o$ entfernt, d.h. $\mathcal{B}_o = \{6, 7\}$. Dann sind $u' = (c_{17}, c_{26}, c_{37}) = (5, 1, 1)$ die entsprechenden Zeilenminimierer. Die Spalte $q = 7$ enthält jetzt zwei Zeilenminimierer, den alten u'_1 und den neuen u'_3. Die Menge $\mathcal{M} = \{k \in [7] \mid c_{2k} > 1, c_{3k} > 1\} = \{1, 4, 5\}$ wird gebildet und danach $\max\{c_{1k} \mid k \in \mathcal{M}\} = \max\{0, 2, 1\} = 2 = c_{14}$ bestimmt. Anschließend wird $r = 4$ in $\mathcal{B}_o$ aufgenommen:

$$\begin{aligned} \mathcal{B}_o &= \{6, 7, 4\} \\ u &= (2, 1, 1) \end{aligned}$$

(3) Nun wird $k = 4$ in $\mathcal{B}_z$ pivotiert: Mit dem Minimalquotiententest $\frac{1}{1} < \frac{3}{1}$ wird das Pivotelement b_{14} bestimmt und anschließend $p = 1$ aus $\mathcal{B}_z$ entfernt. Das neue Tableau ist:

$$\begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & b \end{matrix} \\ \left(\begin{array}{cccc|ccc|c} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \\ -1 & 0 & 1 & 0 & -1 & 0 & 1 & 2 \end{array}\right) \end{array}$$

$$\mathcal{B}_z = \{4, 6, 7\}$$

Der Algorithmus endet, da jetzt $\mathcal{B}_z = \mathcal{B}_o$ ist. Die Menge $J = \{4, 6, 7\}$ wurde gefunden.
Der zugehörige Lösungsvektor, der $Bx = b$ erfüllt, lässt sich am letzten Tableau ablesen:

$$x^\top = (0, 0, 0, 1, 0, 2, 2).$$

3.2 Existenz eines Kernels

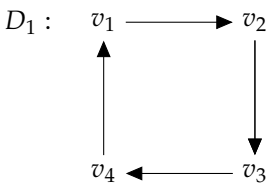
In Anlehnung an Aharoni und Holzman [4] wird gezeigt, dass jede clique-azyklische Superorientierung eines perfekten Graphen einen Kernel besitzt. Der ursprüngliche Beweis nutzt hierfür fraktionale Kernel. Der hier gezeigte geänderte Beweis kommt ohne diese aus und verwendet statt dessen einen Ganzzahligkeitssatz von Chvátal 2.18. Er ist dadurch kürzer und übersichtlicher. Es wird mit einigen notwendigen Definitionen begonnen, welche an zwei Beispieldigraphen veranschaulicht werden, bevor der eigentliche Satz behandelt wird.

Definition 3.7. Es sei $D = (V, A)$ ein Digraph.

(i) Die *In-Nachbarschaft* $\mathbf{I}(\mathbf{v})$ eines Knotens v ist v selbst zusammen mit der Menge aller Knoten, die einen Bogen nach v senden, also v und alle Vorgänger von v.
(ii) Eine Teilmenge W von V heißt *dominierend*, wenn sie für jedes $v \in V$ die Menge $I(v)$ trifft, das heißt $\forall\, v \in V : W \cap I(V) \neq \emptyset$.
(iii) Eine Knotenmenge $W \subseteq V$ heißt *unabhängig* bzw. *stabil*, wenn je zwei beliebige voneinander verschiedene Knoten aus W nicht adjazent sind.
(iv) Ein *Kernel* in D ist eine unabhängige und dominierende Knotenmenge.
(v) Eine Knotenmenge C ist eine *Clique* in D, wenn je zwei verschiedene Knoten durch mindestens einen Bogen verbunden sind[1], m.a.W. C ist im unterliegenden Graphen vollständig.
(vi) Ein Digraph ist *vollständig*, wenn seine Knotenmenge eine Clique ist.
(vii) Ein Bogen (u, v) ist *irreversibel*, wenn (v, u) kein Bogen des Digraphs ist.
(viii) Ein *Kreis* in D ist eine Knotenfolge $v_1 \dots v_k$ mit $(v_i, v_{i+1}) \in A$ für $i = 1, \dots k-1$ und $v_1 = v_k$ ist der einzige mehr als einmal auftretende Knoten.
(ix) Ein Kreis in D heißt *echt*, wenn alle seine Kanten irreversibel sind.
(x) Ein Digraph, in dem keine Clique einen echten Kreis enthält, heißt *clique-azyklisch*.
(xi) Wir sprechen von einer *Superorientierung* eines ungerichteten Graphen, wenn die Kanten nicht nur durch irreversible Bögen sondern auch durch Paare entgegengesetzt gerichteter Bögen ersetzt werden können.

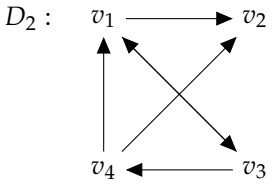
Bemerkung 3.8. Wenn im weiteren Verlauf von Orientierungen gesprochen wird, dann sind damit Superorientierungen gemeint, sofern nichts anderes gesagt wird.

Beispiel 3.9.



Der Digraph D_1 ist nicht vollständig, weil einerseits v_1 und v_3 und andererseits v_2 und v_4 nicht miteinander verbunden sind. D_1 enthält einen echten Kreis $\{v_1, v_2, v_3, v_4\}$. Dieser ist allerdings in keiner Clique enthalten. Also ist D_1 dennoch ein clique-azyklischer Digraph. D_1 enthält kein Paar entgegengesetzt gerichteter Kanten, ist also eine „gewöhnliche“ Orientierung eines Graphen G (und keine Superorientierung).

[1] Dies ist nicht die einzige Möglichkeit, Cliquen in einem Digraphen zu definieren. Eine andere Definition verlangt bspw., dass eine Clique C in D aus mindestens drei Knoten besteht, je zwei verschiedene Knoten durch entgegengesetzt gerichtete Kanten miteinander verbunden sind und C größtmöglich ist. Diese Definition ist für unsere Zwecke jedoch ungeeignet.



$I(v_3) = \{v_1, v_3\}$ ist die In-Nachbarschaft des Knotens v_3 in D_2. Die Knotenmenge $U_2 = \{v_1, v_4\}$ ist dominierend, weil v_1 in $I(v_1), I(v_2)$ und $I(v_3)$, sowie v_4 in $I(v_4)$ enthalten sind. Sie ist aber nicht unabhängig, da v_1 und v_4 adjazent sind. Die Knotenmenge $U_1 = \{v_4\}$ ist unabhängig, aber nicht dominierend, da sie $I(v_3)$ nicht trifft. Die Knotenmenge $W = \{v_2, v_3\}$ ist unabhängig und dominierend, also ein Kernel. Die Mengen $\{v_1, v_3, v_4\} \supset \{v_1, v_3\} \supset \{v_1\}$ sind Cliquen. Der einzige Kreis $K = \{v_1, v_3, v_4\}$ ist nicht echt, da der Bogen (v_1, v_3) nicht irreduzibel ist. Also ist D_2 ein clique-azyklischer Digraph. Der Digraph ist nicht vollständig, weil v_2 und v_3 nicht adjazent sind. D_1 enthält das Paar (v_1, v_3) und (v_3, v_1) entgegengesetzt gerichteter Kanten, ist also eine Superorientierung des unterliegenden Graphen.

Satz 3.10. *Jede clique-azyklische Orientierung eines perfekten Graphen hat einen Kernel.*

Beweis. Sei $D = (V, A)$ ein clique-azyklischer Digraph mit n Knoten $v_1, \ldots, v_n$, dessen unterliegender Graph $G = (V, E)$ perfekt ist.

Finde Matrizen B und C, sowie einen Vektor b, so dass die Voraussetzungen des Satzes von Scarf erfüllt sind:
Sei $C_1, \ldots, C_m$ eine Aufzählung aller maximalen Cliquen in D. Durch Hinzufügen der Knoten $z_1, \ldots, z_m$ und aller Bögen der Form (u, z_i) mit $u \in C_i$ wird der Digraph D' gebildet. Der unterliegende Graph von D' wird mit G' bezeichnet. Es gilt also

$D' = (V', A')$ mit
$V' = V \cup \{z_1, \ldots, z_m\}$
$A' = A \cup \{(u, z_i) \mid u \in C_i, i \in [m]\}$
$C_i' = C_i \cup \{z_i\}$ für alle $i \in [m]$

D ist clique-azyklisch und alle zusätzlichen Bögen führen zu z_i hin und keiner von z_i weg. Also ist z_i in keinem echten Kreis enthalten. Deshalb ist auch D' clique-azyklisch.
Nun kann auf jeder Clique C_i' eine strenge Totalordnung $>_i$ definiert werden, die sich mit den irreversiblen Bögen in C_i' verträgt. Das heißt, wenn $u, v \in C_i'$ sind und (u, v) ein irreversibler Bogen in D' ist, dann ist $u >_i v$. (Da keine Clique einen echten Kreis enthält, ist die Ordnung transitiv. Weil jede Clique vollständig ist, ist auch die Trichotomie erfüllt, das heißt für alle $u, v \in C_i'$ gilt entweder $u >_i v$ oder $v >_i u$ oder $v = u$.)
Sei $V' = \{w_1 = z_1, \ldots, w_m = z_m, w_{m+1}, \ldots, w_n\}$ eine Aufzählung aller Knoten von D'. Damit ist $V = \{w_{m+1}, \ldots, w_n\}$. Die $m \times n -$ Matrix C wird wie folgt definiert:
Es wird ein beliebiges $M > |V| = n - m$ gewählt und

$$c_{ij} = \begin{cases} M & \text{, für } w_j \notin C_i' \\ \text{Höhe von } w_j \text{ in der Ordnung } >_i & \text{, sonst} \end{cases}$$

gesetzt für alle $i \in [m]$ und $j \in [n]$. Beispielsweise ist $c_{ij} = 0$, wenn w_j in C_i' minimal ist, was nur für $w_j = z_i$ der Fall ist, oder $c_{ij} = 1$, falls w_j der zweite Knoten von unten ist, usw. So entsteht die Matrix

$$C = \begin{array}{c} \\ C'_1 \\ C'_2 \\ \vdots \\ C'_m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\ \left(\begin{array}{cccc|c} 0 & M & \dots & M & \\ M & & & \vdots & \text{Einträge aus} \\ \vdots & & \ddots & M & [n-m] \cup \{M\} \\ M & \dots & M & 0 & \end{array}\right) \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} m+1 \quad \dots \quad n \\ \\ \\ \\ \\ \end{array},$$

wobei die Spalte i mit dem Knoten w_i korrespondiert.
B sei die Cliquenmatrix von D'. Also ist

$$B = \begin{array}{c} \\ C'_1 \\ C'_2 \\ \vdots \\ C'_m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\ \left(\begin{array}{cccc|c} 1 & 0 & \dots & 0 & \\ 0 & & & \vdots & \text{Einträge aus } \{0,1\} \\ \vdots & & \ddots & 0 & \\ 0 & \dots & 0 & 1 & \end{array}\right) \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \end{array} \begin{array}{c} m+1 \quad \dots \quad n \\ \\ \\ \\ \\ \end{array}$$

Es sei $b = \mathbf{1} \in \mathbb{R}^m$. Da die Menge $\{x \in \mathbb{R}^n_{\geq 0} \mid Bx = \mathbf{1}\}$ beschränkt ist, sind alle Voraussetzungen des Satzes von Scarf (3.1) erfüllt, und der Algorithmus von Scarf (3.1.1) kann durchlaufen werden.

Scarfs Algorithmus durchlaufen:
Um Entartung auszuschließen, werden zuvor die Matrix C und der Vektor b entsprechend (vgl. Beweis von 3.1) perturbiert. Diese Änderungen werden am Ende des Algorithmus wieder zurück genommen.
Wie bereits gezeigt wurde, terminiert Scarfs Algorithmus in jedem Fall. Am Ende gibt er eine Teilmenge $J \subseteq [n]$ der Mächtigkeit m, die gleichzeitig eine ordinale Basis für C und eine zulässige Basis für $Bx = \mathbf{1}, x \geq 0$, ist. Der zugehörige Lösungsvektor kann entweder am aktuellen (letzten) Tableau abgelesen werden, sofern dieses vorliegt, oder kann durch Einsetzen des Vektors x, mit $x_j = 0$ für alle $j \notin J$, in das Gleichungssystem und anschließendes Lösen desselben gefunden werden.

Der Lösungsvektor x ist ganzzahlig:
Nach Voraussetzung ist G perfekt. Man kann sich leicht überlegen, dass für alle Teilgraphen $H \subseteq G'$ die Gleichheit $\omega(H) = \chi(H)$ erfüllt ist. Also ist auch G' perfekt. Des Weiteren stimmen die Cliquenmatrizen für den Digraphen D' und den Graphen G' überein. Nach dem Ganzzahligkeitssatz von Chvátal (2.18) hat dann das Polyeder $P(B)$ ganzzahlige Ecken. Da sich keine der Ecken des Polyeders $\{x \in \mathbb{R}^n \mid Bx = b\}$ als echte Konvexkombination von Elementen aus $P(B)$ darstellen lässt, sind sie alle ebenfalls Ecken von $P(B)$ und als solche ganzzahlig. Damit wurde $x \in \{0,1\}^n$ bewiesen.

Der Lösungsvektor x liefert einen Kernel:
Der Lösungsvektor $x \in \{0,1\}^n$ ist der charakteristische Vektor einer unabhängigen Knotenmenge F. Denn wenn x eine Clique C_i in mehr als einem Knoten träfe, dann wäre im Ausgangsgleichungssystem $B_{i.}x \geq 1 + 1 \neq 1$. Analog folgt, dass x (und damit auch J) jede Clique C'_i trifft. Denn andernfalls gäbe es ein $i \in [m]$ mit $B_{i.}x = 0 \neq 1$.

Es wird nun die Menge

$$\begin{aligned} K &= \{w_i \mid i \in \{m+1, \ldots, n\} \text{ und } x_i \neq 0\} \\ &= \{v_i \mid i \in [n-m] \text{ und } x_{m+i} \neq 0\} \subseteq V. \end{aligned}$$

betrachtet. Diese ist als Teilmenge von F ebenfalls unabhängig. Wenn noch bewiesen werden kann, dass sie dominierend ist, dann wurde ein Kernel in D gefunden. Dafür ist zu zeigen, dass $K \cap I(v) \neq \emptyset$ ist für alle $v \in V$.
Dafür wird ausgenutzt, dass J eine ordinale Basis für C ist, das heißt

$$\forall\, k \in [n]\ \exists\, i \in [m] \text{ mit } c_{ik} \leq c_{ij}\ \forall j \in J. \tag{3.1}$$

Es ist bekannt, dass $V = \{w_{m+1}, \ldots, w_n\}$ ist. Seien nun $k^* \in \{m+1, \ldots, n\}$ beliebig und i^* der Zeilenindex wie in (3.1). Setze $C'^J_{i^*} = \{w_j \in C'_{i^*} \mid j \in J\}$. Offensichtlich ist $C'^J_{i^*} \neq \emptyset$. Als Teilmenge von C'_{i^*} ist $C'^J_{i^*}$ ebenfalls eine Clique. Es soll gezeigt werden, dass $C'^J_{i^*}$ in der In-Nachbarschaft $I(w_{k^*})$ enthalten ist.
Zuvor wird bemerkt, dass $V \subseteq V'$ ist, also ist $I(w_{k^*})$ auch in V' enthalten. Die In-Nachbarschaft von w_{k^*} in D ist gleich der in D', da $w_{k^*} \notin \{w_1, \ldots, w_m\} = \{z_1, \ldots, z_m\}$ gewählt wurde, und es für kein $i \in [m]$ einen Bogen (z_i, w_{k^*}) von z_i nach w_{k^*} gibt.
Der folgende Beweis ist indirekt. Es wird angenommen, dass $C'^J_{i^*}$ nicht in $I(w_{k^*})$ enthalten ist. Das heißt, es gibt ein $w_{j^*} \in C'^J_{i^*} \setminus I(w_{k^*})$. Da $w_{j^*} \in C'^J_{i^*} \subseteq C'_{i^*}$ ist, gilt $c_{i^*j^*} < M$ nach Definition von C. Aufgrund der Wahl von k^* und i^* ist $c_{i^*k^*} \leq c_{i^*j}$ für alle $j \in J$ (also auch für j^*), woraus $c_{i^*k^*} < M$ folgt. Deshalb liegt w_{k^*} nach Definition von C in C'_{i^*}. Da $C'^J_{i^*}$ eine Clique ist und sowohl w_{j^*} als auch w_{k^*} enthält, sind diese zwei Knoten durch mindestens einen Bogen verbunden. Der Knoten w_{j^*} ist nicht in $I(w_{k^*})$ enthalten, weshalb (w_{k^*}, w_{j^*}) ein irreversibler Bogen in D' ist. Nach Definition von $>_i$ ist deshalb $w_{k^*} >_i w_{j^*}$, woraus $c_{i^*k^*} > c_{i^*j^*}$ folgt. Dies ist ein Widerspruch zur Wahl von k^* und j^*.
Damit wurde $C'^J_{i^*} \subseteq I(w_{k^*}) \subseteq V$ bewiesen. Das heißt, alle Elemente aus J, die $I(w_{k^*})$ treffen, sind größer als m. Da der Vektor x jede Clique C'_i trifft, gilt

$$\emptyset \neq F \cap C'_{i^*} \subseteq J \cap C'_{i^*} = C'^J_{i^*} = \{w_j \in C'_{i^*} \mid j \in J \text{ und } j > m\},$$

also ist $j > m$ für alle $w_i \in F \cap C'_{i^*}$. Damit wurde $F \cap C'_{i^*} = K \cap C'_{i^*}$ und $K \cap C'^J_{i^*} \neq \emptyset$ bewiesen, woraus $K \cap I(w_{k^*}) \neq \emptyset$ folgt. □

3.3 Algorithmus zur Kernelberechnung

Der Beweis des Satzes 3.10 von Aharoni und Holzman liefert den folgenden Algorithmus für die Berechnung eines Kernels in einem clique-azyklischen Digraphen, dessen unterliegender Graph perfekt ist.

Algorithmus zur Kernelberechnung:

Eingabe: Clique-azyklische Orientierung $D = (V, A)$ eines perfekten Graphs G mit $V = \{v_1, \ldots, v_n\}$
Berechne alle maximalen Cliquen $C_1, \ldots C_m$ von D
Erweitere D zu D':
$V' = V \cup \{z_1, \ldots, z_m\}$
$A' = A \cup \{(v, z_i) \mid v \in C_i, i \in [m]\}$
$\forall i \in [m] : C_i' = C_i \cup \{z_i\}$
B sei die Cliquenmatrix von D'
Erstellen der Matrix C:
Bestimme die Ordnungen $>_i$ für die Cliquen C_i'
Knoten umbenennen: $V' = \{w_1 = z_1, \ldots, w_m = z_m, w_{m+1} = v_1, \ldots, w_{n+m} = n\}$
Wähle $M > n$
for $i = 1, \ldots, m$ und $j = 1, \ldots, n$
{ if $w_j \in C_i'$ then c_{ij} = Höhe von w_j in der Ordnung $>_i$ else $c_{ij} = M$.
}
$b := \mathbf{1} \in \mathbb{R}^m$
Perturbation von C und b, um Entartung auszuschließen
Durchlaufen des Algorithmus von Scarf:
Eingabe Scarf: Matrizen B und C, Vektor b
Startmengen bestimmen:
zulässige Basis $\mathcal{B}_z := \{1, \ldots, m\}$
$\max\{c_{1(m+1)}, \ldots, c_{1n}\} = c_{1j_1}$
ordinale Basis $\mathcal{B}_o := \{j_1, 2, \ldots, m\}$
Zeilenminimierer: for i=1 to m, i++
$\{u_i = \min\{c_{ij} \mid j \in \mathcal{B}_o\}$
}
While $\mathcal{B}_o \neq \mathcal{B}_z$
$\{k := \mathcal{B}_o \setminus \mathcal{B}_z$
Pivotschritt für $\mathcal{B}_z$: k aufnehmen und p entfernen, d.h. $\mathcal{B}_z = \mathcal{B}_z + k - p$
While $\mathcal{B}_z \neq \mathcal{B}_o$, ordinalen Pivotschritt für $\mathcal{B}_o$ durchführen:
$\{\mathcal{B}_o = \mathcal{B}_o - p$
for i=1 to m, i++
$\{u_i' = \min\{c_{ij} \mid j \in \mathcal{B}_o\}$
}
$\exists$ Spalte $q \in \mathcal{B}_o$ mit zwei Zeilenminimierern (ZM):
$u_{i_n}' \in q$ mit $u_{i_n}' \neq u_{i_n}$ ist der neue ZM
$u_{i_a} = u_{i_a}' \in q$ ist der alte ZM
Setze $\mathcal{M} = \{k \in [n] \mid c_{ik} > u_i' \ \ \forall i \in [m] - i_a\}$

$\max\{c_{i_a j} \mid j \in \mathcal{M}\} = c_{i_a r}$
$\mathcal{B}_o = \mathcal{B}_o + r$
$u_{i_n} = u'_{i_n}$
$u_{i_a} = c_{i_a r}$
}
}
Ausgabe Scarf: $J = \mathcal{B}_z$
Perturbation von b und C zurücknehmen
Berechne Lösung x für $Bx = b$ mit $x_k = 0 \; \forall k \notin J$
Ausgabe: Kernel $K = \{w_i \in V \mid x_i = 1\}$

3.4 Beispiel für den Algorithmus zur Kernelberechnung

Der Algorithmus für die Kernelberechnung, welcher auf dem Algorithmus von Scarf basiert, soll an einem Beispiel veranschaulicht werden. Gegeben sei die clique-azyklische Orientierung $D = (V, A)$ des perfekten Graphen $G = (V, E)$ wie in Abbildung 3.1 dargestellt. Dieser Digraph hat die vier maximalen Cliquen $C_1 = \{v_1, v_2, v_5\}, C_2 = \{v_2, v_3, v_5\}, C_3 = \{v_3, v_4, v_5\}$ und $C_4 = \{v_1, v_4, v_5\}$. Also ist $m = 4$. Mit den Zusatzknoten $z_1, \ldots, z_4$ sowie den Kanten (v, z_i) für alle $v \in C_i$, und für alle $i = [4]$ wird der Digraph D zu dem Digraphen $D' = (V', A')$ erweitert. Letzterer hat neun Knoten.

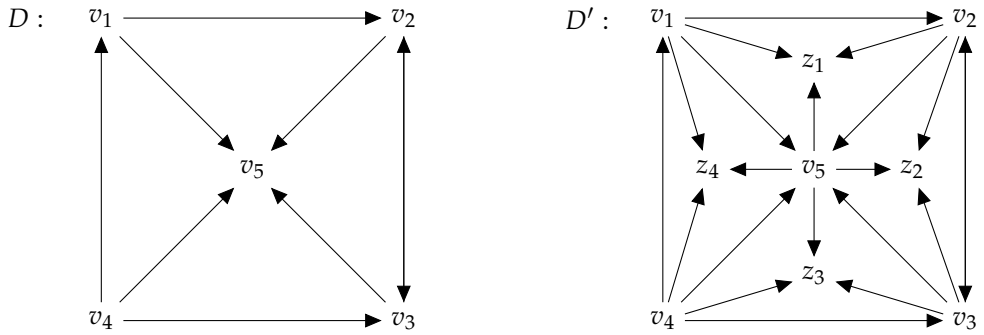


Abb. 3.1: Der Digraph D sowie der erweiterte Digraph D' des Beispiels für die Kernelberechnung mit dem Algorithmus von Scarf

Abweichend vom Beweis des Satzes 3.10 wird keine Knotenumbenennung in $w_1, \ldots, w_9$ vorgenommen, sondern die Knoten selbst werden als Spaltenindizes verwendet, denn so sind die Struktur der Basen und der Zeilenminimierer besser zu erkennen und die später folgenden Vereinfachungen besser nachvollziehbar. Aus demselben Grund werden die beiden mit den Zusatzknoten z_i und den ursprünglichen Knoten v_i korrespondierenden Teilmatrizen von C und B durch eine Trennlinie voneinander abgehoben.
Es werden die Cliquenmatrix B von D', den Vektor $b = \mathbf{1} \in \mathbb{R}^4$, sowie die Matrix C erstellt. Für Letztere werden die Ordnungen $>_i$ der Knoten in den Cliquen $C'_i = C_i + z_i$ benötigt:

$$C_1' : v_1 >_1 v_2 >_1 v_5 >_1 z_1$$
$$C_2' : v_2 >_2 v_3 >_2 v_5 >_2 z_2$$
$$C_3' : v_4 >_3 v_3 >_3 v_5 >_3 z_3$$
$$C_4' : v_4 >_4 v_1 >_4 v_5 >_4 z_4$$

(Hierbei ist die Ordnung $>_2$ nicht eindeutig. Das Vertauschen der Knoten v_2 und v_3 würde die andere mögliche Ordnung liefern. Alternativ könnten $v_2 =_2 v_3$ gesetzt und diese Gleichheit später beim Perturbieren aufgehoben werden.) Es sei $M > |V| = 5$ beliebig. Es sei

$$c_{ij} = \begin{cases} M & \text{, für } w_j \notin C_i' \\ \text{Höhe von } w_j \text{ in der Ordnung } >_i & \text{, sonst} \end{cases}$$

für alle $i \in [4]$ und $j \in [9]$. Entartung soll ausgeschlossen werden. Deshalb wird M in der Matrix C spaltenweise in den Spalten z_i durch N_i und in den Spalten v_i durch M_i ersetzt, wobei $N_1 > \ldots > N_4 > M_1 > \ldots > M_5 > 5$ gilt (die Unterscheidung in M_i und N_i soll wieder dem besseren Verständnis dienen). Die Komponenten des Vektors b werden durch $b_i = 1 + \epsilon_i$ mit sehr kleinen $\epsilon_4 > \epsilon_3 > \epsilon_2 > \epsilon_1 > 0$ ersetzt. Für das Gleichungssystem $Bx = b$ entsteht so das folgende Tableau:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \left(\begin{array}{ccccc|ccccc||c} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_3 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1+\epsilon_4 \end{array}\right) \end{array} \quad \begin{array}{c} b \\ \\ \\ \\ \\ \end{array}$$

Die perturbierte Matrix C hat die Gestalt:

$$C = \begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{ccccccccc} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \left(\begin{array}{cccc|ccccc} 0 & N_2 & N_3 & N_4 & 3 & 2 & M_3 & M_4 & 1 \\ N_1 & 0 & N_3 & N_4 & M_1 & 3 & 2 & M_4 & 1 \\ N_1 & N_2 & 0 & N_4 & M_1 & M_2 & 2 & 3 & 1 \\ N_1 & N_2 & N_3 & 0 & 2 & M_2 & M_3 & 3 & 1 \end{array}\right) \end{array}$$

(1) Die Startmengen werden bestimmt. Es ist $\max\{c_{1k} \mid k = v_1, \ldots, v_5\} = M_3 = c_{1v_3}$. Es wird also $j_1 = v_3$ in $\mathcal{B}_o$ aufgenommen, das heißt, $\mathcal{B}_o = \{v_3, z_2, z_3, z_4\}$. Die zugehörigen Zeilenminimierer sind $u_1 = \min\{c_i k \mid k \in \mathcal{B}_o\} = M_3$ und analog $u_i = c_{iz_i} = 0$ für $i = 2, 3, 4$. Diese werden zu dem Vektor $u = (u_1, \ldots, u_4)$ zusammen gefasst. Die Startmengen sind:

$$\mathcal{B}_z = \{z_1, z_2, z_3, z_4\} \qquad \begin{aligned} \mathcal{B}_o &= \{v_3, z_2, z_3, z_4\} \\ u &= (M_3, 0, 0, 0) \\ &= (c_{1v_3}, c_{2z_2}, c_{3z_3}, c_{4z_4}), \end{aligned}$$

(2) Zulässiger Pivotschritt: Es ist $\{v_3\} = \mathcal{B}_o \setminus \mathcal{B}_z$. Es wird also die Spalte $k = v_3$ in $\mathcal{B}_z$ aufgenommen. Der Minimalquotiententest liefert das Pivotelement c_{2v_3}. Also wird z_2 aus der zulässigen Basis entfernt. Das neue Tableau ist:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc||c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_2 \\ 0 & -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & \epsilon_3-\epsilon_2 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1+\epsilon_4 \end{array}\right) \end{array}$$

Ordinaler Pivotschritt: Nun wird die Spalte z_2 aus $\mathcal{B}_o$ entfernt. Die Zeilenminimierer der in $\mathcal{B}_o$ verbleibenden Spalten sind $u_1' = c_{1v_3} = M_3, u_2' = c_{2v_3} = 2, u_3' = c_{3z_3} = 0$ sowie $u_4' = c_{4z_4} = 0$. Die Spalte v_3 enthält nun zwei Zeilenminimierer, den alten u_1' und den neuen u_2'. Die Menge $\mathcal{M}$, welche alle Spalten enthält, deren Einträge größer als die Zeilenminimierer u_2', u_3' und u_4' sind, wird gebildet. Das heißt $\mathcal{M} = \{k \mid c_{ik} > u_i' \text{ für } i = 2,3,4\} = \{z_1, v_1, v_2, v_4\}$. Mit deren Hilfe wird $\max\{c_{1k} \mid k \in \mathcal{M}\} = \max\{0,3,2,M_4\} = M_4 = c_{1v_4}$ gefunden. Also wird v_4 in die ordinale Basis aufgenommen, und die Zeilenminimierer werden entsprechend angepasst.
Nach den beiden Pivotschritten sehen die Mengen so aus:

$$\begin{aligned} \mathcal{B}_z &= \{z_1, v_3, z_3, z_4\} \\ \mathcal{B}_o &= \{v_3, v_4, z_3, z_4\} \\ u &= (M_4, 2, 0, 0) \\ &= (c_{1v_4}, c_{2v_3}, c_{3z_3}, c_{4z_4}), \end{aligned}$$

(3) Ein zulässiger Pivotschritt nimmt v_4 in $\mathcal{B}_z$ auf und entfernt dafür z_3. Das neue Tableau ist:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc||c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_2 \\ 0 & -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & \epsilon_3-\epsilon_2 \\ 0 & 1 & -1 & 1 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_4-\epsilon_3+\epsilon_2 \end{array}\right) \end{array}$$

Ein ordinaler Pivotschritt entfernt z_3 aus $\mathcal{B}_o$ und nimmt dafür z_2 (wieder) auf. Der alte Zeilenminimierer war $u_2' = 2$, der neue $u_3' = 2$. Die neuen Mengen sind:

$$\begin{aligned} \mathcal{B}_z &= \{z_1, v_3, v_4, z_4\} \\ \mathcal{B}_o &= \{v_3, z_2, v_4, z_4\} \\ u &= (M_4, 0, 2, 0) \\ &= (c_{1v_4}, c_{2z_2}, c_{3v_3}, c_{4z_4}), \end{aligned}$$

(4) Es wird z_2 in $\mathcal{B}_z$ pivotiert und dabei v_3 entfernt. Das neue Tableau ist:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc||c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_3 \\ 0 & 0 & -1 & 1 & 1 & 0 & -1 & 0 & 0 & \epsilon_4-\epsilon_3 \end{array}\right) \end{array}$$

Mit einem ordinalen Pivotschritt wird v_3 aus $\mathcal{B}_o$ (der alte Zeilenminimierer ist $u_1' = M_4$, der neue ist u_3') entfernt und dafür v_1 aufgenommen. Die neuen Mengen sind:

$$\begin{aligned}\mathcal{B}_z &= \{z_1, z_2, v_4, z_4\} & \mathcal{B}_o &= \{v_1, z_2, v_4, z_4\}\\ & & u &= (3,0,3,0)\\ & & &= (c_{1v_1}, c_{2z_2}, c_{3v_4}, c_{4z_4}),\end{aligned}$$

(5) Mit einem zulässigen Pivotschritt wird v_1 in $\mathcal{B}_z$ aufgenommen und z_4 daraus entfernt. Das neue Tableau ist:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc||c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 1 & -1 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_1+\epsilon_3-\epsilon_4 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_3 \\ 0 & 0 & -1 & 1 & 1 & 0 & -1 & 0 & 0 & \epsilon_4-\epsilon_3 \end{array}\right) \end{array}$$

Mit einem ordinalen Pivotschritt wird z_4 aus $\mathcal{B}_o$ entfernt und dafür v_2 aufgenommen (alter Zeilenminimierer ist $u_1' = 3$, der neue ist $u_4' = 2$). Damit sind

$$\begin{aligned}\mathcal{B}_z &= \{z_1, z_2, v_4, v_1\} & \mathcal{B}_o &= \{v_1, z_2, v_4, v_2\}\\ & & u &= (2,0,3,2)\\ & & &= (c_{1v_2}, c_{2z_2}, c_{3v_3}, c_{4v_1}).\end{aligned}$$

(6) Mit einem zulässigen Pivotschritt wird z_1 in $\mathcal{B}_z$ durch v_2 ersetzt. Das neue Tableau ist:

$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc||c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 1 & -1 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_1+\epsilon_3-\epsilon_4 \\ -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & \epsilon_2-\epsilon_1+\epsilon_4-\epsilon_3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_3 \\ 0 & 0 & -1 & 1 & 1 & 0 & -1 & 0 & 0 & \epsilon_4-\epsilon_3 \end{array}\right) \end{array}$$

Hiermit endet der Algorithmus, weil $\mathcal{B}_z = \mathcal{B}_o$ ist:

$$\begin{aligned}\mathcal{B}_z &= \{v_2, z_2, v_4, v_1\} & \mathcal{B}_o &= \{v_1, z_2, v_4, v_2\}\\ & & u &= (2,0,3,2)\\ & & &= (c_{1v_2}, c_{2z_2}, c_{3v_3}, c_{4v_1}).\end{aligned}$$

Den Lösungsvektor $x^\top = (0,0,0,0,0,1,0,1,0)$ kann am letzten Tableau abgelesen werden (mit dem Wissen, dass alle Komponenten außerhalb der Basis $\mathcal{B}_z$ gleich Null sind). Dies liefert den Kernel $K = \{v_2, v_4\}$.

3.5 Eigenschaften des Algorithmus zur Kernelberechnung

Es werden im Folgenden einige Eigenschaften des Algorithmus Kernel aufgeführt, die an dem vorhergehenden Beispiel **??** und/oder dem Beweis des Satzes 3.10 von Aharoni und Holzman [4] nachvollzogen werden können. Diese Eigenschaften sind in die Überlegungen zur Entwicklung des Beweises, dass das Problem der Kernelberechnung in PPAD liegt, eingeflossen.

1. Die zulässige Basis $\mathcal{B}_z$ und die ordinale Basis $\mathcal{B}_o$ müssen nicht unabhängig sein. Folglich ist die Unabhängigkeit der Basen beim zulässigen Pivotschritt nicht als Kriterium für die Wahl des zu entfernenden Knotens geeignet. In dem Beispiel **??** ist es sogar so, dass nur die zulässige Basis im Schritt (1) unabhängig ist, sonst sind beide Basen immer abhängige Knotenmengen. Die Unabhängigkeit des Kernels wird erst durch den Lösungsvektor des Gleichungssystems $Bx = \mathbf{1}$ sicher gestellt. Es wurde im Beweis unter Verwendung das Satzes von Chvátal 2.18 gezeigt, dass dieser aus $\{0,1\}^n$ sein muss. Deshalb trifft er jede maximale Clique in genau einem Knoten.

2. Der Algorithmus ist unabhängig von der Reihenfolge der Cliquen und der Knotennummerierung. Die erste Clique hat insofern eine besondere Rolle, als dass sie als einzige Clique den Zusatzknoten z_1 enthält, welcher für der Terminierung des Algorithmus von Bedeutung ist.

3. Der Algorithmus terminiert entweder mit der Aufnahme des Zusatzknotens z_1 in die ordinale Basis oder mit dessen Entfernung aus der zulässigen Basis. Denn für die beiden Basen gilt während des Algorithmus die Beziehung $\mathcal{B}_z \setminus \mathcal{B}_o = \{z_1\}$ und $\mathcal{B}_o \setminus \mathcal{B}_z = \{w\}$, wobei w sich mit jedem Schritt verändert. Es wurde gezeigt, dass immer genau ein Pivotschritt möglich ist. In dem einen Fall wird w mit einem zulässigen Pivotschritt in die zulässige Basis aufgenommen. Wenn dafür die Spalte z_1 entfernt wird, dann stimmen die beiden Basen überein. Im anderen Fall wird der Knoten w mit einem ordinalen Pivotschritt aus der ordinalen Basis entfernt. Wenn dabei z_1 aufgenommen wird, dann sind die beiden Basen ebenfalls gleich.

4. Es ist möglich, dass ein Knoten aus $V \cap C_1'$, mit einem zulässigen Pivotschritt in die zulässige Basis aufgenommen wird, ohne dass dafür der Zusatzknoten z_1 aus derselben entfernt werden muss. Vergleiche Schritt (5) in Beispiel **??**.

5. Es ist möglich, dass ein Knoten, der aus den Basen entfernt wurde, zu einem späteren Zeitpunkt wieder in diese aufgenommen wird. Im Beispiel wird der Zusatzknoten z_2 im Schritt (2) aus beiden Basen entfernt und mit den Schritten (3) und (4) wieder aufgenommen.

6. Bei jedem ordinalen Pivotschritt ändern sich genau zwei Zeilenminimierer, alle anderen bleiben gleich. Angenommen die Spalte j wird aus der ordinalen Basis entfernt. Diese hat bisher den Zeilenminimierer u_{i_n} enthalten. Dann wird der Zeilenminimierer u_{i_n} durch $u'_{i_n} = \min\{c_{i_n k} \mid k \in \mathcal{B}_o - j\} = c_{i_n q}$ ersetzt, und der Zeilenminimierer u'_{i_a} wird durch $c_{i_a r} = \max\{c_{i_a k} \mid k \in \mathcal{M}\}$ ersetzt.
   Die Matrix C sei ordinal generisch, das heißt keine zwei Elemente einer Zeile sind gleich. Dann ist immer $u'_{i_a} > c_{i_a r}$, weil die Spalte, die den alten Zeilenminimierer u'_{i_a} enthält, in der ordinalen Basis verbleibt. Es gilt immer $u'_{i_n} > u_{i_n}$, weil sich die Spalte, die u'_{i_n} enthält, schon vor dem Entfernen der Spalte j in der ordinalen Basis befunden und nicht den Zeilenminimierer u_{i_n} gestellt hat. Wenn C nicht entartet ist, dann sind Gleichheiten möglich.

7. Genau dann, wenn der Zusatzknoten z_i in der ordinalen Basis liegt, ist der Zeilenminimierer $u_i = c_{iz_i} = 0$. Wenn z_i nicht in der Basis liegt, dann ist der Zeilenminimierer $u_i > 0$. Die Zeilenminimierer, welche größer als Null sind, korrespondieren immer mit Knoten aus V. Das liegt an der Konstruktion der Matrix C im Beweis des Satzes 3.10, die die Ungleichungen im Lemma von Scarf 3.1 erfüllt.

8. Der Algorithmus ist unabhängig von den gewählten Perturbationen der Matrix C und des Vektors b, solange diese geeignet gewählt wurden. Eine Veränderung der Perturbationen kann den Verlauf des Algorithmus beeinflussen. Aber es wird am Ende immer ein Kernel ausgegeben. Dies kann einfach an dem Beispiel nachvollzogen werden, indem beispielsweise die Ungleichungen zwischen den ϵ_i und den M_i umgedreht werden. Das heißt es gelte $\epsilon_1 > \ldots > \epsilon_4 > 0$, analog für die M_i.

9. Die Zeilenminimierer ungleich u_1 sind immer kleiner als $|V| = n$. Begründung: Zu Beginn sind die Zeilenminimierer $u_2 = \ldots = u_m = 0$, weil sie mit den Zusatzknoten aus der ordinalen Basis zusammenhängen. Der Zeilenminimierer $u_i = 0$ ändert sich genau dann, wenn der Zusatzknoten z_i mit einem ordinalen Pivot aus der ordinalen Basis entfernt wird. Dies ist wiederum genau dann der Fall, wenn dieser Zusatzknoten zuvor bei einem zulässigen Pivot aus der Basis $\mathcal{B}_z$ entfernt wurde. Das ist nur dann möglich, wenn dafür ein Knoten aufgenommen wurde, der in der Clique C_i' enthalten ist. Da jeder Zusatzknoten in genau einer Clique enthalten ist, kann der für z_i aufgenommene Knoten kein Zusatzknoten sein. Sobald sich ein Knoten aus $V \cap C_i'$ in der ordinalen Basis befindet, muss aufgrund der Konstruktion der Matrix C der Zeilenminimierer kleiner-gleich n sein. Dies gilt ähnlich für die folgenden Schritte.

10. Der Zeilenminimierer u_1 muss kleiner-gleich n sein, wenn der Algorithmus terminiert. In dem Fall, dass der Zusatzknoten z_1, in die ordinale Basis aufgenommen wurde, ist $u_1 = 0$. In dem Fall, dass z_1 aus der zulässigen Basis entfernt wurde, muss dafür ein Knoten aus $C_i' \cap V$ aufgenommen werden. Da beide Basen nun gleich sind, ist dieser Knoten auch in der ordinalen Basis enthalten, weshalb der Zeilenminimierer $u_1 \leq n$ ist, wie oben gezeigt wurde.

4 Reduktionskette von Kintali

Kintali u.a. haben in [1] unter anderem die folgende Reduktionskette (Abb. 4.1) aufgestellt und gezeigt, dass alle diese Probleme PPAD-vollständig sind. Der Aufbau dieses Kapitels orientiert sich an der Reduktionskette. Das Problem END OF THE LINE kommt hier nicht vor, weil es bereits im Kapitel Grundlagen im Zusammenhang mit der Komplexitätsklasse PPAD abgehandelt wurde. Wofür wird diese Reduktionskette benötigt? Der Teil der Reduktionskette, der bei PREFERENCE GAME beginnt und bei STRONG KERNEL endet, dient als Vorlage für die Überprüfung, ob das Problem KERNEL PPAD-schwer ist. Dafür wird auch die PPAD-Vollständigkeit von PREFERENCE GAME benötigt. Die ersten zwei Reduktionen zeigen, dass PREFERENCE GAME PPAD-schwer ist, die letzten vier die Zugehörigkeit zu PPAD. Die Reduktion von SCARF auf END OF THE LINE wird beim Nachweis der PPAD-Zugehörigkeit angewandt. Zusätzlich zu dieser Reduktionskette wird unter Verwendung des Lemmas von Sperner gezeigt, dass BROUWER in PPAD ist, denn dies ist hilfreich für das Verständnis der Formulierung des Suchproblems 3D-BROUWER.

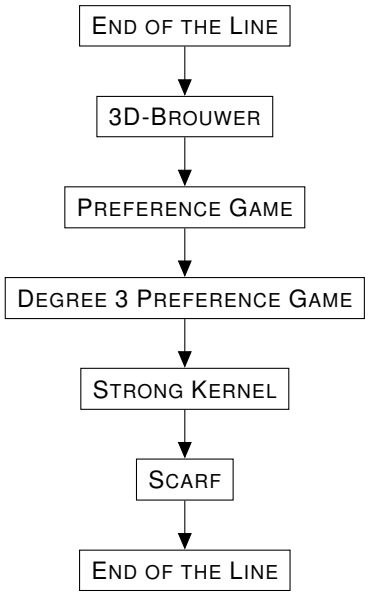


Abb. 4.1: Reduktionskette

4.1 BROUWER

Die Reduktion von BROUWER auf END OF THE LINE verwendet ein elegantes Resultat der Kombinatorik, das Lemma von Sperner. Mit diesem wird begonnen. Anschließend wird BROUWER formuliert und dessen PPAD-Vollständigkeit bewiesen. Die folgende Darstellung stammt im Wesentlichen von Dakalakis [13] und wurde an einigen Stellen durch [12] und [14] ergänzt.

4.1.1 Lemma von Sperner

Satz 4.1 (Lemma von Sperner 1928)**.** *Die Knoten einer kanonischen Triangulierung des Hyperwürfels* $[0,1]^m$ *seien so mit den Farben* $0, 1, \ldots, m$ *gefärbt, dass die Färbung auf dem Rand die folgende Eigenschaft besitzt:*

(Sp_m) *Für alle* $i \in [m]$ *wird keiner der Knoten auf der Seite* $x_i = 0$ *mit der Farbe i gefärbt; zusätzlich wird die Farbe 0 für keinen Knoten auf der Seite* $x_i = 1$ *für ein* $i \in [m]$ *genutzt.*[1]

Dann gibt es in dieser Zerlegung ein panchromatisches Simplex *(das ist ein Simplex, dessen Knoten alle* $m+1$ *Farben haben). Die Anzahl aller panchromatischen Simplexe ist ungerade.*

Was ist unter einer *kanonischen Triangulierung* zu verstehen? Damit ist das m-dimensionale Äquivalent einer Triangulierung gemeint. Papadimitriou [12] und Daskalakis [14] bezeichnen dies als „simplicization“. Allgemein wird ein m-dimensionaler Würfel in m-dimensionale Teilwürfel (deren Größe hängt nun auch von der Anzahl der Dimensionen m ab) untereilt und jeder Teilwürfel in m-Simplizes zerlegt. Dafür gibt es viele Möglichkeiten. Es wird die folgende gewählt: Angenommen der Teilwürfel ist $[0,\lambda]^m$. Für jede Permutation $\sigma = (i_1, \ldots, i_m)$ von $(1, \ldots, m)$ sei h_σ die Teilmenge des Teilwürfels, die die Punkte enthält, die den Ungleichungen $0 \leq x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_m} \leq \lambda$ genügen. Da σ alle Permutationen von $[m]$ durchläuft, entstehen so $m!$ Simplizes h_σ, die alle zusammen eine Unterteilung des Teilwürfels liefern. In Abbildung 4.2 ist eine kanonische Triangulierung eines Würfels für $m = 3$ dargestellt.

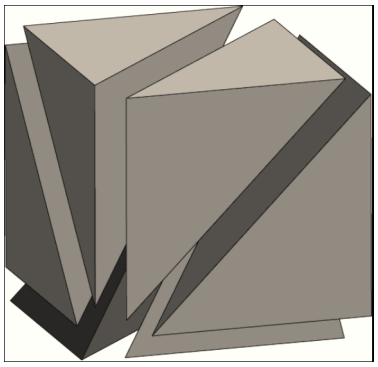


Abb. 4.2: Kanonische Triangulierung eines Würfels. Alle Tetraeder dieser Unterteilung nutzen die Ecken 000 und 111 des Würfels. Aus: [14]

Warum wurde das Spernersche Lemma vermittels kanonischer Triangulierung formuliert? Eigentlich ist keine Beschränkung auf eine kanonische Triangulierung eines Hyperwürfels notwendig. Denn das Lemma

[1] In (Sp_m) steht Sp für Sperner-Färbung und m für m-dimensional

von Sperner gilt für jede Zerlegung des Würfels in m-Simplizes, solange die Färbung die Spernereigenschaft (Sp_m) hat. Allerdings wird so erreicht, dass die zugehörige Version des Problems, welches eine Lösung von Sperner berechnet, „algorithmusfreundlich“ ist: Die Triangulierung und deren Simplexe können auf einfache Art definiert werden, und die Nachbarn eines Simplex können effizient berechnet werden. etc. Durch diesen Aufbau können alle Schritte (bis auf die Länge des Weges) im Beweis des Lemmas von Sperner konstruktiv ausgeführt werden.

Im Folgenden wird der Beweis im anschaulichen zweidimensionalen Fall skizziert. Es wird mit einem Einheitsquadrat gestartet. Dieses wird zunächst in kleinere Quadrate der Größe δ unterteilt und anschließend jedes dieser kleinen Quadrate in zwei rechtwinklige Dreiecke, wie in Abbildung 4.3 gezeigt (die Farben der Knoten, die Graufärbung einiger Dreiecke und die Pfeile sind vorerst zu ignorieren) zerlegt. Die Knoten dieser Unterteilung dürfen fast beliebig mit drei Farben, rot, blau und gelb, gefärbt werden, dabei ist nur die folgende Regel zu beachten:

(Sp_2) An der unteren Seite wird kein Knoten rot gefärbt, an der linken Seite keiner blau und kein Knoten der zwei anderen Seiten gelb.

Die daraus resultierende Färbung kann wie in Abbildung 4.3 aussehen, wobei die Pfeile und die Graufärbung einiger Dreiecke immer noch ignoriert werden sollen. Nach dem Lemma von Sperner gibt es in jeder Färbung mit der Eigenschaft (Sp_2) mindestens ein kleines panchromatisches Dreieck. Diese Dreiecke sind in Abbildung 4.3 grau gefärbt. Es wird davon ausgegangen, dass die Farben der Knoten des Gitters nicht explizit gegeben sind. Stattdessen wird $\delta = 2^{-n}$ gesetzt und eine Schaltung als gegeben angenommen, welche zu einer Eingabe (x, y) mit $x, y \in \{0, 1, \ldots, 2^n\}$, die Farbe des Punktes $2^{-n} \cdot (x, y)$ ausgibt.

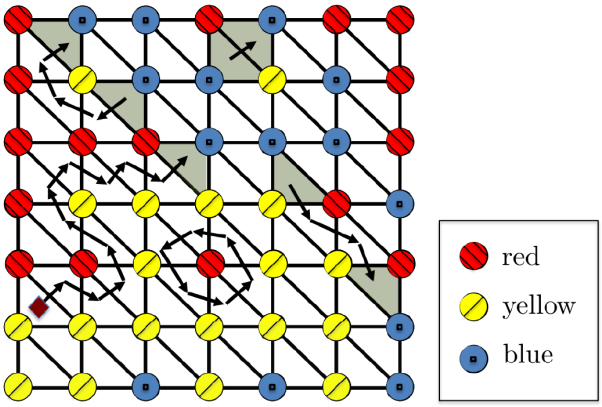


Abb. 4.3: Eine Veranschaulichung des Lemmas von Sperner und des zugehörigen Beweises. Die Dreiecke korrespondieren mit den Knoten des END OF THE LINE-Graphen und die Pfeile mit den Kanten; der Quellknoten T^* ist durch eine Raute markiert. Aus: [13]

Wie schwer ist es nun, ein dreifarbiges Dreieck zu finden? Es wird gezeigt, dass dieses Problem zu PPAD gehört. Als Nebenprodukt wird so ein Beweis des Spernerschen Lemmas geliefert. Dieser Beweis wird einen END OF THE LINE-Graphen konstruieren, dessen Lösungen mit den panchromatischen Dreiecken der Sperner Instanz korrespondieren.
Der Einfachheit halber wird angenommen, dass es auf der linken Seite des Einheitsquadrates nur einen Wechsel von gelb zu rot gibt, wie in Abbildung 4.3. Im allgemeinen Fall kann das Quadrat erweitert werden, indem links von der linken Seite des Quadrates ein vertikales Feld mit Knoten hinzufügt wird, die alle - bis auf den untersten - rot sind. Der unterste Knoten sei gelb gefärbt. Offensichtlich führt diese

Erweiterung kein neues dreifarbiges Dreieck ein, da die Kante des Quadrates vor dieser Ergänzung nur rote und gelbe Knoten enthalten hat. Nun wird die Konstruktion des END OF THE LINE-Graphen wie folgt vorgenommen: Die Knoten des Graphen werden mit den rechtwinkligen Dreiecken der Unterteilung so identifiziert, dass der Knoten 0^n mit dem Dreieck T^* korrespondiert, welches nach Voraussetzung das einzige Segment der linken Seite des Einheitsquadrates enthält, in dem der Wechsel von gelb zu rot auftritt (dieses Dreieck ist in Abbildung 4.3 durch eine Raute markiert). Eine Kante von u nach v wird eingeführt, wenn die korrespondierenden rechtwinkligen Dreiecke T_u und T_v eine gemeinsame rot-gelbe Kante haben, die in T_u im Uhrzeigersinn von rot nach gelb geht. Diese Kanten wurden in Abbildung 4.3 durch Pfeile dargestellt. Die Dreiecke ohne ein- oder ausgehende Pfeile korrespondieren mit isolierten Knoten.
Es ist nicht schwer zu überprüfen, dass in dem konstruierten Graphen die Ein- und Ausgangsgrade höchstens eins sind. Zusätzlich ist das Dreieck T^* eine Quelle eines Pfades, sofern es nicht dreifarbig ist, und dieser Pfad hat garantiert eine Senke, weil er weder sich selbst schneiden noch das Quadrat verlassen kann, da es keine rot-gelbe Außenkante gibt, die gekreuzt werden kann. Ein Dreieck kann nur dann eine Senke des Pfades sein, wenn es dreifarbig ist! Dies beweist, dass mindestens ein dreifarbiges Dreieck existiert. Selbstverständlich kann es weitere dreifarbige Dreiecke geben, die mit zusätzlichen Quellen und Senken in dem Graphen korrespondieren würden (vgl. Abbildung 4.3), so dass die Anzahl aller dreifarbigen Dreiecke ungerade ist. Die Schaltung, die die Farben der Dreiecke der Unterteilung berechnet, kann zur Konstruktion der Schaltungen S und P verwendet werden, welche für die Spezifizierung einer Instanz von END OF THE LINE notwendig sind. Dies beschließt die Konstruktion. Für die Dimensionen $m \geq 3$ folgt der Beweis demselben Prinzip.

Es gibt zwei Möglichkeiten das Problem, das eine Lösung des Spernerschen Lemmas berechnet, zu formulieren:

SPERNER (1): Gegeben sei eine Schaltung C, die die Knoten einer kanonischen Triangulierung des Einheitswürfels $[0,1]^m$ färbt. Finde entweder einen panchromatischen Simplex oder einen Punkt auf dem Rand, der die zulässige Färbungseigenschaft verletzt.

SPERNER (2): Gegeben sei eine Schaltung C, die die Knoten färbt. Finde einen panchromatischen Simplex in einer Färbung, die von einer anderen Schaltung C' vorgenommen wird, welche innerhalb des Hyperwürfels mit C übereinstimmt und auf dem Rand eine „umhüllende Färbung“(analog zur Erweiterung um eine linke Kante mit einem Wechsel von gelb zu rot im Zweidimensionalen) erzeugt.

Beide Probleme korrespondieren mit totalen Problemen, das heißt zu jeder Eingabe gibt es eine Lösung. Beide Probleme können mit dem obigen Algorithmus gelöst werden.

Dies wird ausführlich von Papadimitriou [12] und Daskalakis [14] beschrieben. Papadimitriou [12] hat gezeigt, dass SPERNER PPAD-vollständig ist. Er startet mit der klassischen Formulierung des Spernerschen Lemmas für triangulierte m-Simplexe und schlägt einen Bogen zu triangulierten Hyperwürfeln, denn dies erleichtert später den Beweis von BROUWER. Insbesondere führt er dabei die kanonische Triangulierung ein. Dann erläutert er den Beweis, dass Sperner in PPAD liegt, bevor er den ausführlichen Beweis, dass SPERNER für $m = 3$ PPAD-vollständig ist, vorlegt (die Fälle $m > 3$ folgen analog). Ein detaillierter und reich bebilderter Beweis der PPAD-Zugehörigkeit von SPERNER kann bei Daskalakis [14] nachgelesen werden.

4.1.2 Das Problem BROUWER

Der Brouwersche Fixpunktsatz lässt sich sehr leicht veranschaulichen. Gegeben seien zwei identische Blätter Papier, die mit einem Koordinatensystem versehen wurden. Eins wird flach auf den Tisch gelegt. Das andere wird zerknüllt, ohne es dabei einzureißen, und irgendwie auf das erste Papier gelegt. Wichtig ist nur, dass das obere Papier nicht über das untere hinausragt. Dann gibt es auf dem zerknüllten Blatt mindestens einen Punkt, der genau über dem korrespondierenden Punkt (das ist der Punkt mit denselben Koordinaten) auf dem glatten Papier liegt. Formal besagt der *Brouwersche Fixpunktsatz*, dass jede stetige Abbildung von einer kompakten und konvexen Teilmenge des euklidischen Raumes, das heißt diese Menge ist abgeschlossen, beschränkt und hat keine Löcher, auf sich selbst immer einen Fixpunkt hat.
Der Satz von Brouwer suggeriert das folgende interessante Problem: Gegeben sei eine stetige Funktion F von einer kompakten und konvexen Teilmenge des euklidischen Raumes auf sich selbst. Finde einen Fixpunkt. Die folgenden drei Punkte sind zu klären, wenn das Problem berechenbar gemacht werden soll:

- Wie wird eine kompakte und konvexe Teilmenge des euklidischen Raumes dargestellt?
- Wie wird die stetige Abbildung von dieser Menge auf sich selbst spezifiziert?
- Wie wird damit umgegangen, dass möglicherweise irrationale Punkte auftreten?

Der Einfachheit halber wird als kompakte und konvexe Menge der Einheitshyperwürfel $[0,1]^m$ fixiert. Der Fall, dass F einen allgemeineren Definitionsbereich hat, kann auf den Einheitshyperwürfel überführt werden, indem der Definitionsbereich so geschrumpft wird, dass er innerhalb des Hyperwürfels liegt und anschließend die Funktion so auf den ganzen Würfel erweitert wird, dass keine neuen Fixpunkte entstehen.
Es wird angenommen, dass die Funktion F durch einen effizienten Algorithmus Π_F gegeben ist, der für jeden binär geschriebenen Punkt x des Würfels $F(x)$ berechnet. Zusätzlich wird angenommen, dass F Lipschitz-stetig ist:

(L) $d(F(x), F(y)) \leq K \cdot d(x, y)$ für alle $x, y \in [0,1]^m$

Hierbei sind d der euklidische Abstand und K die Lipschitz-Konstante von F. Diese Bedingung stellt sicher, dass approximative Fixpunkte lokalisiert werden können, indem der Wert $F(x)$ untersucht wird, während x sich auf einem diskreten Netz über dem Definitionsbereich bewegt. Indem die Suche auf approximative Fixpunkte von F beschränk wird, können irrationale Lösungen behandelt werden. Es lässt sich sogar zeigen, dass es zu jedem $\epsilon > 0$ einen ϵ-approximativen Fixpunkt x gibt (das heißt $d(F(x), x) < \epsilon$), dessen Koordinaten ganzzahlige Vielfache von 2^{-d} sind, wobei 2^d linear in $K, 1/\epsilon$ und der Dimension m ist. Ohne die Lipschitzkonstante K gäbe es diese Garantie nicht, und das Problem der Fixpunktberechnung wäre schwieriger.
Dies alles wird zusammengesetzt und das Problem BROUWER definiert, welches approximative Fixpunkte berechnet.

> BROUWER: Gegeben seien ein effizienter Algorithmus Π_F für die Ermittlung einer Funktion $F : [0,1]^m \to [0,1]^m$, eine Konstante K, so dass F die Bedingung (L) erfüllt, sowie die gewünschte Genauigkeit ϵ. Finde einen Punkt x mit $d(F(x), x) < \epsilon$.

4.1.3 BROUWER ist in PPAD

Es soll gezeigt werden, dass BROUWER in PPAD ist. Dafür ist ein END OF THE LINE-Graph zu bilden, der mit einer BROUWER-Instanz assoziiert ist. Zunächst wird ein Netz aus winzigen Simplizes über dem Definitionsbereich konstruiert, und jedes dieser Simplizes mit einem Knoten des Graphen identifiziert. Dann werden Kanten zwischen Simplizes definiert, welche bezogen auf eine Knotenfärbung des Netzes eine gemeinsame Facette haben. Die Knoten werden abhängig von der Richtung, in die sie durch F verschoben werden, gefärbt. Wenn es einen Simplex gibt, dessen Knoten alle möglichen Farben tragen, dann versucht F diese Knoten in entgegengesetzte Richtungen zu verschieben. Also muss dieser nah bei einem Fixpunkt sein. Dies wird näher ausgeführt, wobei sich auf den zweidimensionalen Fall konzentriert wird:

Zunächst wird wie beim Spernerschen Lemma eine kanonische Triangulierung des Hyperwürfels vorgenommen. Dann wird jeder Knoten x dieser Triangulierung mit einer der $m+1$ Farben gefärbt. Die Wahl der Farbe hängt davon ab, in welche Richtung x durch $F(x)$ verschoben wird. Im Zweidimensionalen kann der Winkel zwischen dem Vektor $F(x) - x$ und der Horizontalen genommen werden. Ein Knoten wird rot gefärbt, wenn dessen Richtung zwischen $0°$ und $-135°$ liegt, blau, wenn sie sich zwischen $90°$ und $225°$ bewegt, und sonst gelb (vgl. Abb.4.4). Falls die Richtung $0°$ ist, dann sind entweder rot oder gelb erlaubt. Selbiges gilt für die beiden anderen Grenzfälle. Ausgehend von dieser Vereinbarung können die Knoten so gefärbt werden, dass eine Sperner-Färbung (das ist eine Färbung, die (Sp_2) erfüllt) vorliegt. Vergleiche hierzu auch Abbildung 4.3. Der mehrdimensionale Fall ist etwas komplizierter, folgt aber demselben Prinzip. Die Menge der Richtungen wird so in die Farben $0, 1, \ldots, m$ zerlegt, dass die Knotenfärbung die Eigenschaft (Sp_m) hat. Grob gesprochen wird die Farbe 0 mit den Richtungen identifiziert, die mit dem positiven Quadranten des m-dimensionalen Raumes korrespondieren. Die übrigen Richtungen werden gleichmäßig so in die Farben $1, \ldots, m$ zerlegt, dass die Eigenschaft (Sp_m) erfüllt ist.

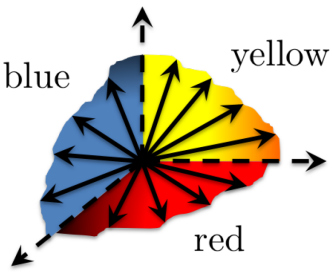


Abb. 4.4: Bestimmung der Färbung eines Knotens x abhängig von der Richtung von $F(x) - x$. Aus: [DaskalakisNash]

Nun sind alle Voraussetzungen für die Anwendung des Lemmas von Sperner erfüllt. Folglich gibt es einen Simplex, dessen Knoten alle $m+1$ Farben haben. Da die Größe der Simplizes hinreichend klein gewählt wurde, und alle Farben den Raum der Richtungen gleichmäßig aufspannen, kann gezeigt werden, dass jeder Knoten eines panchromatischen Dreiecks ein approximativer Fixpunkt ist. Da die Funktion F Lipschitz-stetig ist (F genügt (L)), kann sie nicht zu sehr schwanken. Deshalb können $m+1$ Punkte, die dicht beieinander liegen, nur dann in $m+1$ verschiedene Richtungen abgebildet werden, wenn sie alle näherungsweise fix sind. Also reduziert sich BROUWER auf das Finden eines panchromatischen Simplex in einer Instanz von SPERNER. Es wurde bereits gezeigt, dass SPERNER in PPAD ist. Dies beschließt den Beweis von BROUWER.

4.1.4 BROUWER ist PPAD-vollständig

Es soll gezeigt werden, dass BROUWER PPAD-vollständig ist. Dafür ist nachzuweisen, dass sich ein END OF THE LINE-Graph G in eine stetige, leicht zu berechnende Brouwerfunktion F übersetzen lässt. Dies ist unglücklicherweise ziemlich kompliziert.
Es wird wieder mit dem dreidimensionalen Einheitswürfel als Definitionsbereich von F gestartet. Als nächstes soll das Verhalten von F anhand ihres Betragens auf einem sehr feinen rechtwinkligen und geradlinigen Netz von Gitterpunkten in dem Würfel definiert werden. Jeder Gitterpunkt liegt im Zentrum eines winzigen Teilwürfels. Das Verhalten von F außerhalb der Mittelpunkte der Teilwürfel soll durch Interpolation der nächsten Gitterpunkte gewonnen werden. Jeder Gitterpunkt x soll eine der vier „Farben“ $\{0, 1, 2, 3\}$ erhalten, die den Wert des dreidimensionalen Verschiebungsvektors $F(x) - x$ repräsentieren. Die vier Farben können so gewählt werden, dass sie voneinander wegweisen, so dass $F(x) - x$ nur in der Nachbarschaft aller vier Farben approximativ Null sein kann. Es werden die folgenden vier Vektoren für die Farben gewählt: $(1, 0, 0,)$, $(0, 1, 0)$, $(0, 0, 1)$ und $(-1, -1, -1)$.
Nun kann G selbst in diese Struktur eingepasst werden. Die geniale Konstruktion wird in Abbildung 4.5 dargestellt.

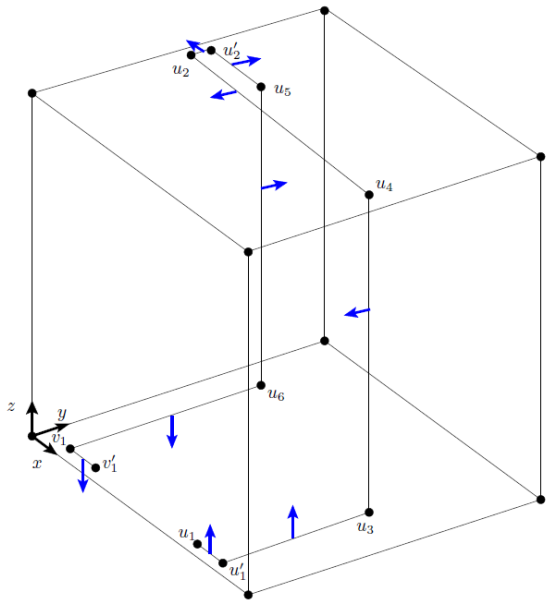


Abb. 4.5: Dies zeigt den orthogonalen Pfad, der bei der Reduktion END OF THE LINE $\leq_p$ BROUWER die Kante (u, v) repräsentiert. Die Pfeile zeigen die Orientierung der Farben an, die den Pfad umgeben. Aus: [13]

Jeder der 2^n Knoten von G korrespondiert mit einem kleinen Segment parallel zu einer Würfelkante (in Abb. 4.5 korrespondieren die Segmente $v_1 \to v_1'$ und $u_1 \to u_1'$ mit den Knoten v respektive u von G; es werden Positionen verwendet, die leicht aus der Identität eines Knotens von G berechnet werden können). Jede Kante von G korrespondiert mit einer Folge zusammenhängender Segmente im Inneren des Würfels (Abb. 4.5 zeigt den Pfad, der mit der gerichteten Kante $u \to v$ korrespondiert; er startet bei dem Knoten u_1' und endet bei dem Knoten v_1). Es ist wichtig, dass lokal bei dem Punkt nur unter Verwendung der Schaltungen S und P der END OF THE LINE-Instanz berechnet werden kann, ob ein Pfad durch einen Gitterpunkt verläuft, und in welche Richtung er geht. Denn diese Pfade sind die Grundlage für die Definition einer effizient berechenbaren Färbung der Gitterpunkte, so dass die Fixpunkte der resultierenden Funktion F mit den unbalancierten Knoten von G korrespondieren.

Hier ist die Kernaussage der Konstruktion: Gegeben sei eine Linie, die mit Pfaden und Kreisen von G korrespondiert. Innerhalb eines gewissen Radius R um diese Linie werden die Gitterpunkte von F auf komplizierte Art so bewegt, dass einerseits die Werte der Koordinaten x, y und z nicht abnehmen und andererseits die einzigen Punkte von F, die in eine Richtung bewegt werden, in der gleichzeitig alle drei Koordinaten erhöht werden, innerhalb eines noch kleineren Radius r um die Linie liegen. Die letztgenannten Punkte werden als *zunehmend* bezeichnet. Dann ist der Abstand der Punkte, die von F in Richtung $(-1,-1,-1)$ bewegt werden, von der Linie größer als R. Diese Punkte werden als *abnehmend* bezeichnet. Insgesamt garantiert die Konstruktion von F, dass ein approximativer Fixpunkt nur dann auftritt, wenn ein zunehmender Punkt dicht bei einem abnehmenden Punkt liegt. Allerdings trennt eine Schicht der Dicke $R - r$ die zunehmenden von den abnehmenden Punkten. Deshalb sind die einzigen Gebiete des Würfels, in denen die zunehmenden und die abnehmenden Punkte „ungeschützt" sind, nah bei den Endpunkten der Segmente, die mit unbalancierten Knoten des END OF THE LINE-Graphen korrespondieren. Dies vervollständigt die Skizze der PPAD-Vollständigkeit von BROUWER.

Dieser wunderbare Beweis kann ausführlich bei Daskalakis, Goldberg und Papadimitriou in [17] nachgelesen werden.

4.1.5 Das Problem 3D-BROUWER

Für die Reduktionskette wird nur der dreidimensionalen Fall des Problems BROUWER benötigt. Es wird das Problem 3D-BROUWER verwendet, welches eine diskrete und vereinfachte Version des Suchproblems ist, das mit Brouwers Fixpunktsatz assoziiert ist. Gegeben sei eine wie im Abschnitt 4.1.2 beschriebene stetige Funktion ϕ von dem dreidimensionalen Einheitswürfel auf sich selbst, die durch die Werte definiert wird, die sie im Zentrum der 2^{3n} Teilwürfel mit der Seitenlänge 2^{-2n} für ein $n \geq 0$ annimmt.[2] Die Teilwürfel K_{ijk} werden als

$$\begin{aligned} K_{ijk} = \{(x,y,z) : \quad & i \cdot 2^{-n} \leq x \leq (i+1) \cdot 2^{-n}, \\ & j \cdot 2^{-n} \leq y \leq (j+1) \cdot 2^{-n}, \\ & k \cdot 2^{-n} \leq z \leq (k+1) \cdot 2^{-n}\} \end{aligned}$$

definiert, wobei i, j, k aus $\{0, 1, \ldots, 2^n - 1\}$ sind, also binär mit n Bits dargestellt werden können. Im Zentrum c_{ijk} des Teilwürfels K_{ijk} ist $\phi(c_{ijk}) = c_{ijk} + \delta_{ijk}$ der Wert von ϕ. Dabei ist δ_{ijk} einer der vier folgenden Vektoren, die auch als Farben bezeichnen werden:

- $\delta_0 = \alpha \cdot (1,0,0)$,
- $\delta_1 = \alpha \cdot (0,1,0)$,
- $\delta_2 = \alpha \cdot (0,0,1)$,
- $\delta_3 = \alpha \cdot (-1,-1,-1)$.

Hierbei ist α viel kleiner als die Seiten der Teilwürfel, beispielsweise sei $\alpha = 2^{-2n}$.
Für die Berechnung von ϕ im Zentrum des Würfels K_{ijk} braucht also nur bekannt zu sein, welche der vier Verschiebungen zu addieren ist. Diese Verschiebung wird durch eine Schaltung C (welche die einzige Eingabe des Problems ist) mit $3n$ Eingabebits und 2 Ausgabebits berechnet. $C(ijk)$ ist der Index r, so dass $\phi(c) = c + \delta_r$ ist, wenn c das Zentrum des Teilwürfels K_{ijk} ist. Die Schaltung C soll am Rand den Bedingungen $C(0,j,k) = 00$ (dies entspricht beispielsweise δ_0), $C(i,0,k) = 01$, $C(i,j,0) = 10$ sowie $C(2^n - 1, j, k) = C(i, 2^n - 1, k) = C(i, j, 2^n - 1) = 11$ genügen (wobei Konflikte beliebig gelöst werden), so dass die Funktion ϕ den Rand des Einheitswürfels ins Innere abbildet. Eine Ecke eines Teilwürfels wird

[2] Der Wert der Funktion dicht bei den Grenzen der Teilwürfel kann durch Interpolation bestimmt werden. Es gibt dafür viele einfache Möglichkeiten. Die konkrete Methode ist unwichtig.

als *panchromatisch* bezeichnet, wenn vier der höchstens acht mit dieser Ecke adjazenten Teilwürfel alle vier Verschiebungen $\delta_0, \delta_1, \delta_2, \delta_3$ haben. Sperners Lemma garantiert, dass für jede Schaltung mit diesen Eigenschaften ein panchromatischer Knoten existiert, und die Fixpunkte von ϕ können nur in der Nähe eines panchromatischen Knoten auftreten. Dies liefert das folgende Suchproblem:

3D-BROUWER: Gegeben sei eine wie oben beschriebene Schaltung C mit $3n$ Eingabebits und 2 Ausgabebits. Finde eine panchromatische Ecke.

4.2 PREFERENCE GAME

Die Darstellung dieses Abschnitts orientiert sich an [1] und [17]. Es wird ein sehr einfaches Spiel definiert, das *Spiel mit Präferenzen*, welches kurz als Präferenzspiel bezeichnet wird. Jeder Spieler hat eine Präferenzliste über der Menge der Spieler und muss jedem Spieler ein Gewicht zuweisen. Kein Spieler darf einen anderen Spieler mit einem größeren Gewicht versehen, als dieser sich selbst zuteilt. Ein Spieler erhält dann eine beste Antwort, wenn es ihm nicht möglich ist, Gewicht von einem Spieler mit niedrigerer Präferenz zu einem Spieler mit höherer Präferenz zu verschieben. Wenn dies für alle Spieler gilt, befindet sich das Spiel im Gleichgewicht. Ein solches existiert immer. Es wird das Problem PREFERENCE GAME definiert und gezeigt, dass es PPAD-vollständig ist.

4.2.1 Das Problem PREFERENCE GAME

Definition 4.2. Ein *Spiel mit Präferenzen*, kurz Präferenzspiel, habe eine Spielermenge S sowie die Strategiemenge S für jeden Spieler. Jeder Spieler $i \in S$ hat eine Präferenzrelation (Rangfolge) $\succeq_i$ zwischen den Strategien. Für Strategien j und k zeigt $j \succeq_i k$ an, dass der Spieler i die Strategie j mindestens so gerne spielt wie k. Es wird $j \succ_i k$ geschrieben, wenn nur $j \succeq_i k$ gilt, das heißt $k \succeq_i j$ ist falsch. Wenn aus dem Zusammenhang hervorgeht, dass über die Präferenzen des Spielers i gesprochen wird, dann wird $\succeq$ statt $\succeq_i$ geschrieben.
Mit der Präferenzrelation eines Spielers i kann eine *Präferenzliste* für den Spieler i definiert werden, die die Strategien so ordnet, dass j nur dann vor k steht, wenn $j \succeq_i k$ gilt. Präferenzlisten sind nicht notwendig eindeutig, da Gleichheiten erlaubt sind.
Jeder Spieler wählt eine *Gewichtsfunktion*, welche eine Zuordnung $w_i : S \to [0, 1]$ ist, so dass die folgenden Bedingungen erfüllt sind:

(a) die Gewichte addieren sich zu 1: $\sum_{j \in S} w_i(j) = 1$
(b) kein Spieler i darf einem anderen Spieler j mehr Gewicht zuweisen, als dieser sich selbst zuteilt: $w_i(j) \leq w_j(j)$, $\forall i, j \in S$

Mit w_{-i} wird die Menge aller Gewichtsfunktionen ungleich i bezeichnet. Es seien Gewichtsfunktionen w_i, w_i' und w_{-i} gegeben, so dass sowohl (w_i, w_{-i}) als auch (w_i', w_{-i}) zulässig sind (das heißt sie erfüllen (a) und (b)). Es wird gesagt, dass w_i (bezüglich w_{-i}) *lexikographisch mindestens* w_i' ist, wenn für alle $j \in S$ gilt:

$$\sum_{k \succeq_i j} w_i(k) \geq \sum_{k \succeq_i j} w_i'(k).$$

Die Gewichtsfunktion w_i wird (bezüglich w_{-i}) als *lexikographisch maximal* bezeichnet, wenn (w_i, w_{-i}) zulässig ist und lexikographisch mindestens jede Funktion w_i' ist, für die (w_i', w_{-i}) zulässig ist.

Lemma 4.3. *Die Gewichtsfunktion w_i ist genau dann (bezüglich w_{-i}) lexikographisch maximal, wenn es ein j gibt, so dass gilt:*

(a) $w_i(k) = w_k(k)$ *für alle* k *mit* $k \succ_i j$,
(b) $w_i(k) = 0$ *für alle* k *mit* $j \succ_i k$.

Beweis. Das folgt aus den obigen Definitionen. Die Belegung $w_i(j)$ wird hierbei so gewählt, dass die Summation auf 1 weiterhin erfüllt ist. □

Salopp gesprochen befindet sich ein Spiel im Gleichgewicht, wenn kein Spieler einen Anreiz hat, von seinem Verhalten abzuweichen. Ein solcher Zustand kann als selbst bindend oder strategisch stabil beschrieben werden, weil es keines irgendwie beschaffenen Verhaltenskontrollmechanismus bedarf, um die Spieler zur Einhaltung des Gleichgewichts zu bewegen. Auf Präferenzspiele übertragen heißt das, dass es im Gleichgewicht keinem Spieler möglich ist, Gewicht von einem Spieler mit niedrigerer Präferenz zu einem Spieler mit höherer Präferenz zu verschieben. Darum gilt:

Lemma 4.4. *Ein Gleichgewicht in einem Präferenzspiel ist eine Zuordnung*

$$w := \{w_i \mid i \in S\},$$

so dass w_i bezüglich w_{-i} lexikographisch maximal ist für alle $i \in S$.

Lemma 4.5. *Jedes Präferenzspiel hat ein Gleichgewicht.*

Beweis. Kintali et al. führen in [1] ein sogenanntes personalisiertes Gleichgewicht für Matrixspiele ein und zeigen, dass ein solches immer existiert. Da Präferenzspiele auch Matrixspiele sind, haben sie ebenfalls immer ein Gleichgewicht. □

Es kann nun das folgende Suchproblem definiert werden:

PREFERENCE GAME: Gegeben seien eine Menge von Spielern $[n]$, jeder mit einer Strategiemenge $[n]$ und einer Präferenzrelation $\succeq_i$ zwischen seinen Strategien. Finde eine zulässige Gewichtszuordnung w, so dass w_i für alle i bezüglich w_{-i} lexikographisch maximal ist.

Satz 4.6. PREFERENCE GAME *liegt in* PPAD.

Beweis. Dies liefert eine Verkettung der Reduktionen in der Reduktionskette, die bei PREFERENCE GAME beginnen und bei END OF THE LINE enden. All diese Beweise werden im Folgenden vorgestellt. Ein alternativer Beweis kann in [1] nachgelesen werden. Er verwendet Matrixspiele mit personalisiertem Gleichgewicht. □

Analog zu BROUWER wird bei Präferenzspielen mit approximativen Gleichgewichten gearbeitet, um die irrationalen Fälle abzudecken. Das wird hier aber nicht weiter beleuchtet.

4.2.2 PREFERENCE GAME ist PPAD-schwer

Satz 4.7. PREFERENCE GAME *ist* PPAD*-schwer.*

Der Beweis von Kintali u.a. [1] überträgt den Beweis von Daskalakis u.a. [17], wo gezeigt wird, dass das Finden eines Nash-Gleichgewichts in graphischen Spielen mit dem Grad drei PPAD-schwer ist, auf Präferenzspiele. Im Folgenden werden die Begriffe Knoten und Spieler teilweise synonym verwendet.

Beweis. Dies wird bewiesen, indem das PPAD-vollständige Problem 3D-BROUWER auf PREFERENCE GAME reduziert wird. Gegeben sei eine Instanz von 3D-BROUWER, das heißt eine Schaltung C mit $3n$ Eingabebits und 2 Ausgabebits, die eine Brouwerfunktion wie in Abschnitt 4.1.5 beschreibt. Es wird ein Präferenzspiel $\mathcal{P}$ konstruiert, welches die Schaltung C simuliert. Dabei wird die Präferenzrelation für jeden Spieler P durch eine geordnete Liste der Spieler spezifiziert, deren letztes Element P ist. Dieses wird auch als *Selbststrategie* bezeichnet. Wenn gesagt wird, dass ein Spieler P sich selbst mit einem Gewicht a spielt, dann ist damit gemeint, dass P der Strategie P das Gewicht a zuordnet. Das Spiel $\mathcal{P}$ ist in dem Sinne binär, dass jeder Spieler darin in jedem Gleichgewicht die Selbststrategie mit einem Gewicht aus $\{0, 1\}$ belegt.[3] Es wird drei verschiedene Spieler X, Y, Z geben, die die Koordinaten eines Punktes in dem dreidimensionalen Würfel repräsentieren. Aufgrund der Konstruktion ist das Spiel nur dann im Gleichgewicht, wenn die Gewichte, die die Spieler X, Y, Z sich selbst zugewiesen haben, erfolgreich die Eingaben und Ausgaben der acht Kopien der Schaltung wiedergeben, die den Lösungsknoten der 3D-BROUWER-Instanz umgeben. Die Bausteine werden die Spielgadgets $\mathcal{P}_{\frac{1}{2}}, \mathcal{P}_{\times\frac{1}{2}}, \mathcal{P}_{=}, \mathcal{P}_{+}, \mathcal{P}_{-}, \mathcal{P}_{2}$ sowie die logischen Gadgets $\mathcal{P}_{\vee}, \mathcal{P}_{\wedge}, \mathcal{P}_{\neg}$ sein. Der Beweis besteht im Wesentlichen aus vier Teilen. Als Erstes werden die Gadgets bereitgestellt, die benötigt werden, um den Zusammenhang zwischen den Spielern X, Y, Z und der Eingabe der Schaltung C herzustellen. Danach wird der entsprechende Algorithmus vorgestellt, den mit Hilfe dieser Gadgets simuliert wird. Als Drittes werden die logischen Gadgets bereitgestellt, mit denen die Schaltung C simuliert werden kann. Zum Schluss wird gezeigt, dass das Gleichgewicht des Spiels und der panchromatische Knoten von Brouwer miteinander korrespondieren.

[3] Die Strategie- und die Spielermenge stimmen überein. Wenn der Spieler i die Strategie i wählt, dann spielt er sich selbst. Dies wird auch als „Selbststrategie“ bezeichnet. Das zugehörige Gewicht ist natürlich $w_i(i)$.

Der Beweis beginnt damit, dass einige notwendige Gadgets bereitgestellt werden:

Lemma 4.8. *Es gibt Präferenzspiele $\mathcal{P}_{\times\frac{1}{2}}, \mathcal{P}_{-}, \mathcal{P}_{\frac{1}{2}}, \mathcal{P}_{=}, \mathcal{P}_{+}$ und $\mathcal{P}_{\times 2}$, jedes mit höchstens fünf der Spieler P, Q, H_1, H_2, H_3 und R, so dass in allen Spielen die Gewichte a resp. b, die P und Q sich selbst zuweisen, nicht von den Gewichten abhängen, die die anderen Spieler H_1, H_2, H_3, R sich selbst zugeteilt haben. Es gelte:*

1. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{\times\frac{1}{2}}$ spielt R sich selbst mit dem Gewicht $a/2$,*
2. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{-}$ spielt R sich selbst mit dem Gewicht $\max\{0, a-b\}$,*
3. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{\frac{1}{2}}$ spielt R sich selbst mit dem Gewicht $1/2$,*
4. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{=}$ spielt R sich selbst mit dem Gewicht a,*
5. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{+}$ spielt R sich selbst mit dem Gewicht $\min\{1, a+b\}$,*
6. *in jedem Gleichgewicht des Spiels $\mathcal{P}_{\times 2}$ spielt R sich selbst mit dem Gewicht $\min\{1, 2a\}$.*

Beweis. 1. $\mathcal{P}_{\times\frac{1}{2}}$: Gegeben sei der Spieler P, der sich selbst mit dem Gewicht a spielt. Es soll ein Spieler R hinzugefügt werden, der sich selbst in jedem Gleichgewicht mit dem Gewicht $a/2$ spielt. Dafür wird der (Hilfs)Spieler H_1 mit der Präferenzliste (P, H_1) geschaffen. H_1 spielt sich selbst mit dem Gewicht $1-a$. Dann werden zwei weitere Spieler H_2 und H_3 geschaffen. Der Spieler H_2 hat die Präferenzliste (H_1, H_3, H_2), und H_3 hat die Präferenzliste (H_1, R, H_3). Die Präferenzliste für R wird (H_1, H_2, R) gesetzt. Jeder der Spieler R, H_2, H_3 wird seine erste Wahl mit dem Gewicht $1-a$ spielen. Somit steht jedem von ihnen das Gewicht a für die zwei anderen Strategien zur Verfügung. Das heißt es gilt

$$\begin{aligned} a &= w_R(H_2) + w_R(R) \\ a &= w_{H_3}(R) + w_{H_3}(H_3) \\ a &= w_{H_2}(H_3) + w_{H_2}(H_2) \end{aligned}$$

Wegen Lemma 4.3 müssen deshalb in jedem Gleichgewicht die folgenden Gleichheiten gelten

$$\begin{aligned} w_R(H_2) &= w_{H_2}(H_2) \\ w_{H_3}(R) &= w_R(R) \\ w_{H_2}(H_3) &= w_{H_3}(H_3) \end{aligned}$$

Auflösen dieses Gleichungssystems liefert $w_R(R) = W_R(H_2) = w_{H_2}(H_2) = w_{H_2}(H_3) = w_{H_3}(H_3) = w_{H_3}(R) = a/2$.

2. $\mathcal{P}_{-}$: Gegeben seien die Spieler P und Q, die sich selbst mit dem Gewicht a resp. b spielen. Es soll ein Spieler R hinzugefügt werden, der sich selbst mit dem Gewicht $\max\{0, a-b\}$ spielt. Dafür wird der Hilfsspieler H_1 mit der Präferenzliste (P, H_1) erzeugt. Dieser spielt sich selbst mit dem Gewicht $1-a$. Nun wird die Präferenzliste für R auf (H_1, Q, R) gesetzt. Dann spielt R sich selbst mit dem Gewicht $\max\{0, 1-(1-a)-b\} = \max\{0, a-b\}$.
3. $\mathcal{P}_{\frac{1}{2}}$: Es soll ein Spieler R hinzugefügt werden, der sich selbst in jedem Gleichgewicht mit dem Gewicht $1/2$ spielt. Es wird der Hilfsspieler H_1 erzeugt, dessen erste Präferenz er selbst ist (das heißt er spielt sich selbst mit dem Gewicht 1), und es wird das Präferenzspiel $\mathcal{P}_{\times\frac{1}{2}}$ auf diesen angewendet.
4. $\mathcal{P}_{=}$: Gegeben sei ein Spieler P, der sich selbst mit dem Gewicht a spielt. Ein neuer Spieler R, der sich selbst ebenfalls mit dem Gewicht a spielt, soll hinzugefügt werden. Der Spieler H_1 mit der Präferenzliste (P, H_1) wird erzeugt. Dieser spielt sich selbst mit dem Gewicht $1-a$. Die Präferenzliste von R wird auf (H_1, R) gesetzt. Dann versieht R den Spieler H_1 mit dem Gewicht $1-a$, was a für R übrig lässt.
5. $\mathcal{P}_{+}$: Gegeben seien die Spieler P und Q, die sich selbst mit dem Gewicht a resp. b spielen. Es kann ein Knoten R hinzugefügt werden, der sich selbst in jedem Gleichgewicht mit dem Gewicht $\min\{1, a+b\}$

spielt. Es wird der Spieler H_1 mit der Präferenzliste (P, Q, H_1) geschaffen. Es sei (H_1, R) die Präferenzliste von R. Nun wird offensichtlich H_1 sich selbst mit dem Gewicht $\max\{0, 1-a-b\}$ spielen, und R spielt H_1 mit demselben Gewicht. Deshalb wird R sich selbst mit dem Gewicht $1 - \max\{0, 1-a-b\}$ spielen. Anders gesagt, wenn $a + b > 1$ ist, dann spielt R sich selbst mit 1. Andernfalls spielt R sich selbst mit dem Gewicht $1 - (1 - a - b) = a + b$.

6. $\mathcal{P}_{\times 2}$: Gegeben sei ein Spieler P, der sich selbst mit dem Gewicht a spielt. Ein neuer Spieler R wird hinzufügt, der sich selbst in jedem Gleichgewicht mit dem Gewicht $\min\{1, 2 \cdot a\}$ spielt. Ein Spieler $H_1 = \mathcal{P}_{=}(P)$ wird erzeugt und $R = \mathcal{P}_{+}(P, H_1)$ gesetzt. □

Bezeichnung 4.9. $\mathcal{P}_{+}(P, Q)$ bzeichnet die Anwendung des Gadgets $\mathcal{P}_{+}$ auf die Spieler P und Q. Das gilt analog für alle anderen Spiele. Bei manchen Spielen ist die Reihenfolge der Argumente zu beachten. Beispielsweise unterscheidet sich $\mathcal{P}_{-}(P, Q)$ von $\mathcal{P}_{-}(Q, P)$.
Mit $\mathcal{P}^{i}_{\times \frac{1}{2}}$ wird die i-malige Anwendung von $\mathcal{P}_{\times \frac{1}{2}}$ zusammengefasst.

Lemma 4.10. *Es gibt ein Präferenzspiel $\mathcal{P}_{<}$ mit zwei gegebenen Spielern P und Q, die sich selbst mit dem Gewicht a resp. b spielen. In diesem spielt der Spieler R sich selbst in jedem Gleichgewicht mit dem Gewicht*

$$\begin{cases} 1 & \text{, für } a \geq b + \epsilon \\ 0 & \text{, für } a \leq b \end{cases}$$

für eine gewisse Konstante $\epsilon = \frac{1}{2^k}$, für ein $k \in \mathbb{N}$. Das Präferenzspiel $\mathcal{P}_{<}$ wird als Vergleichsspiel bezeichnet.

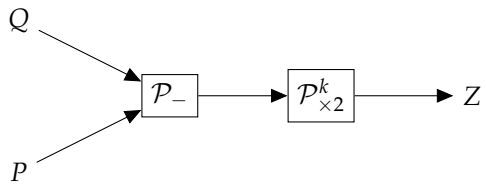


Abb. 4.6: Konstruktion des Vergleichsspiels $\mathcal{P}_{<}$

Beweis. Gegeben seien $\epsilon = \frac{1}{2^k}$ für ein $k \in \mathbb{N}$ sowie zwei Spieler P und Q, die sich selbst in jedem Gleichgewicht mit a bzw. b spielen. In Abbildung 4.6 ist die Konstruktion des Vergleichsspiels $\mathcal{P}_{<}(X, Y)$ schematisch dargestellt. Es wird ausgenutzt, dass $2^k = \frac{1}{\epsilon}$ ist. Zunächst wird mit dem Differenzspiel $\mathcal{P}_{-}(P, Q)$ ein Spieler erzeugt, der sich selbst in jedem Gleichgewicht mit dem Gewicht $\max\{0, a-b\}$ spielt. Auf diesen wird k-mal das Verdopplungsspiel $\mathcal{P}_{\times 2}$ angewandt und so der Spieler R erzeugt. Dieser spielt sich selbst in jedem Gleichgewicht mit dem Gewicht $\min\{1, \max\{0, 2^k(a-b)\}\} = \min\{1, \max\{0, \frac{a-b}{\epsilon}\}\}$. Das heißt R spielt sich selbst im Gleichgewicht mit dem Gewicht

$$\begin{cases} 1 & \text{, für } a \geq b + \epsilon \\ \frac{a-b}{\epsilon} & \text{, für } b < a < b + \epsilon \\ 0 & \text{, für } a \leq b \end{cases}$$

(Wer unbedingt möchte, kann auch mit einer beliebigen Konstante $0 < \epsilon \leq \frac{1}{2}$ und $k = -\lfloor \log_2 \epsilon \rfloor$ arbeiten. Das ist allerdings nicht notwendig, da mit den Zweierpotenzen jede gewünschte Genauigkeit erreicht werden kann und das Rechnen einfacher ist.) □

Bemerkung 4.11. Das Vergleichsspiel $\mathcal{P}_{<}$ ist inakkurat, wenn a und b sehr dicht beieinander liegen. Denn der Spieler R spielt sich selbst mit einem zuvor nicht definierten Gewicht, falls $b < a < b + \epsilon$ ist. In allen anderen Fällen spielt er sich selbst entweder mit dem Gewicht 1 oder dem Gewicht 0. Das ist auch so

gewollt. Daskalakis u.a. haben in [17] anhand eines Beispiels für graphische Spiele verdeutlicht, dass ein Vergleichsspiel $\mathcal{G}_<$ an der Stelle $a = b$ nicht robust sein darf. Denn mit einem solchen Spiel, kann ein graphisches Spiel konstruiert werden, das kein Gleichgewicht besitzt. Der Vollständigkeit halber wird ein analoges von uns entwickeltes Beispiel für Präferenzspiele in Beispiel 4.12 angegeben.

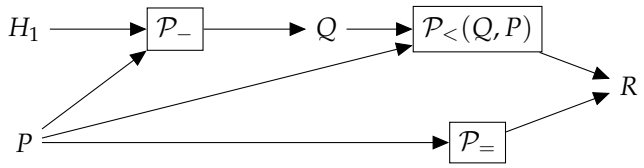


Abb. 4.7: Beispiel, dass ein robustes Vergleichsspiel zu einem Präferenzspiel ohne Gleichgewicht führen kann

Beispiel 4.12. Es wird angenommen, dass das Vergleichsgadget $\mathcal{P}_<(P, Q)$ so beschaffen ist, dass R sich selbst in jedem Gleichgewicht mit dem Gewicht

$$\begin{cases} 1 & \text{, für } a > b \\ 0 & \text{, für } a \leq b \end{cases}$$

spielt. Nun wird ein Präferenzspiel konstruiert, welches kein Gleichgewicht hat. Dieses Spiel wird in Abbildung 4.7 dargestellt. Gegeben seien ein Spieler P, der sich selbst mit dem Gewicht a in jedem Gleichgewicht spielt und ein Spieler H_1, der sich selbst immer mit Gewicht 1 spielt. Mit dem Differenzspiel $\mathcal{P}_-(H_1, P)$ wird ein Spieler Q erzeugt, der sich selbst in jedem Gleichgewicht mit dem Gewicht $1 - a$ spielt. Der Spieler R spielt einerseits sich selbst ebenfalls mit dem Gewicht a in jedem Gleichgewicht, da er wegen $\mathcal{P}_=(P)$ eine Kopie von P ist. Andererseits spielt R sich selbst wegen des Vergleichsspiels $\mathcal{P}_<(Q, P)$ mit dem Gewicht 1, falls $1 - a > a$ ist bzw. mit 0, wenn $1 - a \leq a$ ist. Im Folgenden werden drei Fälle unterschieden: Wenn $a \in (0, 1)$ ist, spielt R sich selbst mit a und gleichzeitig mit einem Wert aus $\{0, 1\}$, was nicht möglich ist. Wenn $a = 1$ ist, spielt R sich selbst mit $a = 1$ und 0, da $1 - a = 0 < 1 = a$ ist. Dies ist ebenfalls ein Widerspruch. Für $a = 0$ spielt R sich selbst mit 0 und 1, weil $1 - a = 1 > 0 = a$ ist, was wieder nicht möglich ist. Also hat dieses konstruierte Spiel kein Gleichgewicht. Dies widerspricht dem Lemma 4.5, welches besagt, dass jedes Präferenzspiel ein Gleichgewicht hat. Folglich darf das Vergleichsspiel nicht robust sein.

Nun kann mit der Reduktion von 3D-BROUWER auf PREFERENCE GAME begonnen werden. Dafür werden die folgenden Spieler in das Präferenzspiel aufgenommen:

- die drei Koordinatenspieler X, Y, Z, einen für jede der drei Dimensionen. Wenn das Präferenzspiel im Gleichgewicht ist, spielt sich jeder Koordinatenspieler mit dem Gewicht a_x, a_y bzw. a_z, das mit seiner Koordinate im Lösungsknoten von 3D-BROUWER übereinstimmt.
- für $i \in [n]$ die Bitspieler $B_i(x), B_i(y), B_i(z)$, einen für jeden Bit der drei Koordinaten. Deren Gewichte, mit denen sie sich selbst spielen, korrespondieren mit dem Wert des i-ten höchstwertigen Bits von a_x, a_y resp. a_z. Das heißt sie spielen sich selbst immer mit einem Gewicht aus $\{0, 1\}$.
- für $i \in [n]$ die Hilfsspieler X_i, Y_i, Z_i. Diese werden für die korrekte Berechnung der Bitspieler benötigt. Das Gewicht a_{x_i}, mit dem X_i die Selbststrategie spielt, ist gleich dem Gewicht a_x, mit dem der Spieler X sich selbst spielt, abzüglich aller Bruchzahlen, die mit den $i - 1$-ten höchstwertigen Bits von X korrespondieren (analog für Y und Z).

Diese Werte können gewonnen werden, indem die Binärdarstellung von $\lfloor a_x 2^n \rfloor$ berechnet wird (analog für Y und Z). Das sind die Binärdarstellungen der Zahlen i, j, k so dass $(x, y, z) = (a_x, a_y, a_z)$ in dem Teilwürfel K_{ijk} liegt. Dies wird durch ein Präferenzspiel erreicht, welches unter Verwendung der arithmetischen Gadgets der Lemmata 4.8 und 4.10 den folgenden Algorithmus simuliert:

$x_1 = x$;
for $i = 1, \ldots, n$ do:
$\{b_i(x) :=< (x_i, 2^{-i}); x_{i+1} := x_i - b_i(x) \cdot 2^{-i}\}$;
analog für y und z;

Der Algorithmus wird so in $\mathcal{P}$ eingebunden, dass die darin berechneten Werte $x_i, b_i(x)$ etc. den Gewichten $a_{X_i}, a_{B_i(x)}$ etc. entsprechen. Solange a_x, a_y und a_z nicht zu dicht bei einem Vielfachen von 2^{-n} liegen (diese Einschränkung kommt durch die Verwendung des Vergleichsgadgets $\mathcal{P}_<(x_i, 2^{-i})$ im Algorithmus) berechnet der Teil von $\mathcal{P}$, welcher den obigen Algorithmus implementiert, i, j, k so, dass der Punkt (a_x, a_y, a_z) im Teilwürfel K_{ijk} liegt. Das heißt, es gibt $3n$ Spieler des Spiels $\mathcal{P}$, deren Gewichte mit den n Bits der Binärdarstellung von i, j, k übereinstimmen.
Wenn a_x, a_y oder a_z zu dicht bei einem Vielfachen von 2^{-n} sind, werden die Bits nicht korrekt extrahiert und die Simulation der Schaltung kann einen beliebigen Wert zurückgeben. Dieses Problem wird mit derselben Methode der Mittelwertbildung wie in [17] überwunden: Die Schaltung wird für eine große konstante Anzahl an Punkten berechnet, die den Knoten umgeben, und der Mittelwert der resultierenden Vektoren wird genommen. Dafür werden die positiven und negativen Komponenten der Ergebnisvektoren so separiert, wie es weiter unten noch dargestellt wird. Dies liefert einen Mittelwertvektor $\Delta = (\Delta x^+, \Delta x^-, \Delta y^+, \Delta y^-, \Delta z^+, \Delta z^-)$. Anschließend wird der Vektor unter mehrfacher Anwendung von $\mathcal{P}_{\times \frac{1}{2}}$ so herunter skaliert, dass die Größenordnung hinreichend kleiner als die Seitenlänge eines Teilwürfels ist. Für jede Dimension wird als nächstes diese Dimension des skalierten Vektors zu einer Kopie des geeigneten Koordinatenspielers hinzugerechnet, indem als erstes die positive Komponente addiert und als zweites die negative Komponente abgezogen wird. So wird der ursprüngliche Koordinatenspieler erhalten. Wenn einmal die Binärdarstellungen von i, j, k vorliegen, dann können diese in einen anderen Teil von $\mathcal{P}$, der die Schaltung C simuliert, eingegeben werden. Die Schaltung kann durch den Einsatz von Spielern, die Gatter repräsentieren, simuliert werden. Addition (mit Obergrenze 1) kann verwendet werden, um ODER zu simulieren, Multiplikation für UND und $1 - a$ für NICHT. Es gibt einen einfacheren Weg, um Boolesche Funktionen zu simulieren, wenn die Eingaben immer 0 oder 1 sind. Dieser vermeidet die Komplikationen, die mit der Genauigkeit zusammenhängen.

Lemma 4.13. *Es gibt Präferenzspiele $\mathcal{P}_\vee, \mathcal{P}_\wedge$ und $\mathcal{P}_\neg$ mit zwei Eingabespielern P, Q (einem Eingabespieler für $\mathcal{P}_\neg$), die sich selbst in jedem Gleichgewicht mit einem Gewicht a bzw. b aus $\{0, 1\}$ spielen, und einem Ausgabespieler R, der sich selbst ebenfalls mit einem Gewicht c aus $\{0, 1\}$ spielt. Hierbei ist c das Ergebnis der Anwendung der jeweiligen Booleschen Funktion auf die Eingaben.*
Es werden $\mathcal{P}_\vee$ als ODER-Gadget, $\mathcal{P}_\wedge$ als UND-Gadget und $\mathcal{P}_\neg$ als NICHT-Gadget bezeichnet.

Beweis. 1. $\mathcal{P}_\vee(P, Q)$: Es wird ein Hilfsknoten H_1 mit der Präferenzliste (P, Q, H_1) geschaffen. Die Präferenzliste von R sei (H_1, R). Falls a und/oder b gleich 1 sind, dann spielt H_1 sich selbst mit dem Gewicht 0, ergo R sich selbst mit dem Gewicht 1. Wenn sowohl a als auch b gleich 0 sind, dann spielt H_1 sich selbst mit dem Gewicht 1 und R sich selbst mit dem Gewicht 0. Übrigens implementiert $\mathcal{P}_+$ das Spiel $\mathcal{P}_\vee$, wenn $a, b \in \{0, 1\}$ sind.

2. $\mathcal{P}_\neg(P)$: Es sei (P, R) die Präferenzliste von R. Dann spielt R sich selbst in jedem Gleichgewicht mit dem Gewicht $1 - a$.

3. $\mathcal{P}_\wedge$: Hierfür werden die Spiele $\mathcal{P}_\vee$ und $\mathcal{P}_\neg$ wie folgt verkettet:
$\mathcal{P}_\neg(\mathcal{P}_\vee(\mathcal{P}_\neg(P), \mathcal{P}_\neg(Q)))$. □

Es soll sicher gestellt werden, dass sich das Präferenzspiel genau dann im Gleichgewicht befindet, wenn alle vier Vektoren in den Ergebnissen der acht Schaltungen vertreten sind. Wie in [17] ist es praktisch, anzunehmen, dass die Ausgabe von C etwas detaillierter als zwei Bits ist: Es sollen sechs Bits $\delta x^+, \delta x^-, \delta y^+, \delta y^-, \delta z^+, \delta z^-$ von C berechnet werden, so dass höchstens eins von δx^+ und δx^- gleich 1 ist, höchstens eins von δy^+ und δy^- gleich 1 ist und analog für z. Und der Zuwachs der Brouwerfunktion im Zentrum von K_{ijk} ist $\alpha \cdot (\delta x^+ - \delta x^-, \delta y^+ - \delta y^-, \delta z^+ - \delta z^-)$, also gleich einem der Vektoren $\delta_0, \delta_1, \delta_2, \delta_3$ wie sie in der Definition von 3D-BROUWER (vgl. Abschnitt 4.1.5) spezifiziert wurden. Dort wurde $\alpha = 2^{-2n}$ gesetzt. Zurück zur Ausgabe: Die vier möglichen Vektoren δ_i werden als 100000, 001000, 000010 und 010101 dargestellt (Das heißt eine Addition dieser vier Ausgaben liefert 111111. Das wird im Folgenden benutzt). Diese Ergebnisse können in die ursprünglichen Koordinatenspieler zurückübersetzt werden. Als Erstes werden die Bits δx^+, die mit den acht adjazenten Knoten korrespondieren, durch ODER-Gadgets verknüpft. Dieses liefert als Ergebnis genau dann eine eins, wenn mindestens eins der δx^+ gleich eins ist. Dies wird für die anderen fünf Bits $\delta x^-, \delta y^+, \delta y^-, \delta z^+, \delta z^-$ wiederholt. So werden genau dann sechs einsen erhalten, wenn dies ein Lösungsknoten ist, also alle vier Verschiebungsvektoren vertreten sind. Deswegen wird ein AND-Spiel für jede Koordinate genau dann drei einsen zurückgeben, wenn dies ein Lösungsknoten ist. Andernfalls ist mindestens eine der Koordinaten gleich null. Wenn das herumgedreht wird, indem für jede Koordinate das NICHT-Gadget verwendt wird, werden genau dann drei nullen erhalten, wenn es sich um einen Lösungsknoten handelt. Schließlich werden diese Ergebnisse wie folgt unter Verwendung der Spiele $\mathcal{P}_=$ und $\mathcal{P}_+$ zu einer Kopie der ursprünglichen Koordinaten addiert: Mit $\Delta = (\Delta x^+, \Delta x^-, \Delta y^+, \Delta y^-, \Delta z^+, \Delta z^-)$ wird die Zusammenrechnung der Schaltungsausgaben, die jeweils aus sechs Bits bestehen, bezeichnet. Dann wird $a_x = a'_x + (\Delta x^+ - \Delta x^-)$ berechnet, indem als erstes $\mathcal{P}_-(\Delta x^+, \Delta x^-)$ ermittelt wird, und das Ergebnis mit $\mathcal{P}_+$ zu einer mit $\mathcal{P}_=$ erzeugten Kopie a'_x von der x-Koordinate addiert wird. Selbiges wird für a_y und a_z wiederholt. Wenn die Koordinaten eine Lösung der 3D-BROUWER-Instanz repräsentieren, dann sind alle Werte, die zurück addiert wurden, gleich Null. Deshalb können sich die Koordinatenspieler nicht verbessern, indem sie ihre Strategien verändern. Wenn die Koordinaten keinen Lösungsknoten bilden, dann kann dargelegt werden, dass die Spieler sich nicht im Gleichgewicht befinden: Es befinde sich der Punkt (a_x, a_y, a_z) (gegeben durch die Strategien der Koordinatenspieler) vollständig im Inneren des Würfels, das heißt er ist in keinem der Teilwürfel enthalten, die am Rand liegen. Dann ändert jede Koordinate in Δ, die nicht Null ist, die Strategien der Spieler. Demzufolge liegt kein Gleichgewicht vor. Für Punkte am Rand werden die Randbedingungen aufgerufen, die für die Färbung spezifiziert wurden (vgl. Abschnitt 4.1.5). Dafür sind verschiedene Fälle zu unterscheiden. Zwei davon werden repräsentativ betrachtet.
Es sei (a_x, a_y, a_z) in einem Teilwürfel enthalten, der mit der Seite $x = 0$ adjazent ist, aber mit keiner der Seiten $y = 0$ oder $z = 0$. Dann ist mindestens einer der benachbarten Ergebnisvektoren gleich 100000 (wegen der gegebenen Randbedingung für die Seite $x = 0$ in 3D-BROUWER). Da davon ausgegangen wurde, dass kein Brouwerfixpunkt vorliegt, sind nicht alle vier gewünschten Vektoren vertreten. Also muss einer der anderen drei Vektoren fehlen. Falls 010101 fehlt, dann befindet sich der x-Koordinatenspieler nicht im Gleichgewicht. Andernfalls fehlt einer der Vektoren 001000 oder 000010, und der korrespondierende Koordinatenspieler ist nicht im Gleichgewicht.
Nun wird ein anderer Fall betrachtet. Es sei (a_x, a_y, a_z) in einem Teilwürfel enthalten, der mit den Seiten

$x = 1$ und $y = 0$ adjazent ist, aber nicht mit der Seite $z = 0$. Dann sind mindestens zwei der benachbarten Vektoren 010101 und 001000. Da wieder nicht alle vier gewünschten Vektoren vertreten sind, fehlt einer der anderen beiden Vektoren. Falls 100000 fehlt, dann befindet sich der x-Koordinatenspieler nicht im Gleichgewicht, andernfalls gilt selbiges für den z-Spieler.
Die anderen Fälle am Rand können ebenso abgehandelt werden.
Im Beweis kann die Präferenzrelation, die für jedes Gadget definiert wird, durch eine lineare Relation ersetzt werden, so dass jeder Knoten eine strikte Präferenzordnung zwischen den Knoten des Spiels hat. Das ist so, weil alle Präferenzlisten, die in den Gadgets definiert wurden, eine lineare Ordnung bilden. Die Ordnung der Knoten, die nicht in der Präferenzliste eines beliebigen Knotens enthalten sind, ist unwichtig und kann folglich beliebig besetzt werden. □

4.3 CONSTANT DEGREE PREFERENCE GAME

Hierfür werden zunächst der Eingangs- und Ausgangsgrad sowie der daraus resultierende Grad sowohl für die einzelnen Spieler eines Präferenzspiels als auch für das Spiel selbst definiert. Diese Definitionen vertragen sich mit den entsprechenden Definitionen für Digraphen.

Definition 4.14. Es sei ein Präferenzspiel gegeben. Für jeden Spieler v werden $in(v) = \{u \mid v \succ_u u\}$ als die Menge aller Spieler, die v der Selbststrategie (streng) vorziehen, $out(v) = \{u \mid u \succ_v v\}$ als die Menge aller Spieler, die v sich selbst gegenüber (strikt) bevorzugt, der *Eingangsgrad* als $|in(v)|$, der *Ausgangsgrad* als $|out(v)|$ sowie der *Grad* als Summe des Eingangs- und des Ausgangsgrades definiert.
Der *Eingangsgrad (Ausgangsgrad, Grad) eines Präferenzspiels* ist das Maximum der Menge der Eingangsgrade (Ausgangsgrade, Grade) aller Spieler.

DEGREE d PREFERENCE GAME: Finde ein Gleichgewicht in einem Präferenzspiel mit konstantem Grad d.

Bemerkung 4.15. Der Digraph des Präferenzspiels kann so definiert werden, dass die Spieler die Knoten, sind und eine Kante von u nach v bedeutet, dass v von u lieber gespielt wird als die Selbststrategie. Das heißt u steht in der Präferenzliste von v vor v, beeinflusst also das Gewicht, mit dem v sich selbst spielt. Dann vertragen sich die obigen Grad-Definitionen für Präferenzspiele mit den entsprechenden Definitionen für Digraphen.

Satz 4.16. PREFERENCE GAME$\leq_p$ DEGREE d PREFERENCE GAME. *Tatsächlich ist dies eine Reduktion in polynomialer Zeit von allgemeinen Präferenzspielen auf Präferenzspiele mit dem Grad 3, Ausgangsgrad 2 und Eingangsgrad 1.*

Beweisstruktur. Die Ausgangsgrade aller Spieler, die bei der Reduktion von BROUWER auf PREFERENCE GAME definiert wurden (vgl. Abschnitt 4.2.2), sind höchstens zwei. Das heißt, es hängt von höchstens zwei anderen Spielern ab, mit welchem Wert ein Spieler sich selbst spielt. Es gibt keine implizite Konstante, die die Eingangsgrade beschränkt. Das heißt, ein Spieler kann in beliebig vielen Präferenzlisten anderer Spieler auftauchen und so das Gewicht beeinflussen, mit dem diese sich selbst spielen. Durch Hinzufügen von Copy-Gadgets $\mathcal{P}_=$, kann garantiert werden, dass der Eingangsgrad ebenfalls höchstens zwei ist. Darüber hinaus kann sichergestellt werden, dass der Grad des Spieles höchstens drei ist, weil alle Gadgets $\mathcal{P}_=$ den Ausgangsgrad eins haben. Dann bleibt noch zu zeigen, dass sich das Gleichgewicht des neuen

Präferenzspiels in Polynomialzeit auf ein Gleichgewicht des ursprünglichen Präferenzspiels abbilden lässt. Der Beweis kann ausführlich in [1] nachgelesen werden. □

4.4 STRONG KERNEL

Definition 4.17. Es seien $D = (V, A)$ ein Digraph und f eine nichtnegative Funktion auf V. Diese wird als *fraktional dominierend* bezeichnet, wenn $\sum_{u \in I(v)} f(u) \geq 1$ ist für alle Knoten v. Wenn es zu jedem Knoten v eine Clique K gibt, die in der In-Nachbarschaft $I(v)$ enthalten ist, und für die $\sum_{u \in K} f(u) \geq 1$ ist, dann heißt f *stark dominierend*. Die Funktion f heißt *fraktional unabhängig*, wenn $\sum_{u \in K} f(u) \leq 1$ für jede Clique K erfüllt ist. Ein *fraktionaler Kernel* ist eine Funktion f, die sowohl fraktional unabhängig als auch fraktional dominierend ist. Wenn sie zusätzlich stark dominierend ist, wird von einem *stark fraktionalen Kernel* gesprochen.

Bemerkung 4.18. Ein gerichtetes Dreieck (also ein echter Kreis mit drei Knoten) zeigt, dass nicht jeder Digraph einen fraktionalen Kernel besitzt.

Später wird noch das folgende Lemma benötigt:

Lemma 4.19. *Die charakteristische Funktion eines Kernels ist offensichtlich ein stark fraktionaler Kernel.*

Beweis. Es sei $K \subseteq V$ ein Kernel. Dann ist charakteristische Funktion f von K offensichtlich eine nicht negative Funktion auf V. Diese Funktion ist fraktional unabhängig, weil der Kernel eine unabhängige Knotenmenge ist, und stark dominierend, weil K dominierend ist. □

Satz 4.20. *Jeder clique-azyklische Digraph hat einen stark fraktionalen Kernel.*

Beweis. Der Beweis erfolgt analog zur Kernelberechnung, also Erweiterung des Digraphen zu D', Erstellen der Matrizen und Durchlaufen des Algorithmus von Scarf sowie Berechnen der Lösung des Gleichungssystems $Bx = b$. Dann wird die nicht negative Abbildung f auf V durch $f(v_j) = x_{v_j}$ definiert und gezeigt, dass diese sowohl fraktional unabhängig als auch stark dominierend ist. Hierbei kann der Satz von Chvátal 2.18 nicht angewandt werden, weil der Digraph nicht perfekt ist. Das heißt, der Lösungsvektor und damit der stark fraktionale Kernel müssen nicht ganzzahlig sein. □

> STRONG KERNEL: Gegeben sei ein clique-azyklischer Digraph, dessen größte Clique konstante Größe habe. Finde eine Gewichtsfunktion auf den Knoten, die stark dominierend und fraktional unabhängig ist.

Bemerkung 4.21. Kintali et al. nehmen die Einschränkung auf Digraphen, deren größte Clique konstante Größe hat, vor, damit nur polynomial viele maximale Cliquen auftreten können. Das hat zur Folge, dass alle maximalen Cliquen und somit auch die Matrizen in Polynomialzeit berechnet werden können. Also läuft ein Schritt des Algorithmus von Scarf auch polynomial in den Eingabedaten durch. Kintali et al. haben zu dieser Behauptung keine Quelle angegeben. Das beste Ergebnis, welches dazu gefunden wurde, stammt von Nielsen [18]. Er hat gezeigt, dass die Anzahl der maximalen unabhängigen Mengen, die genau die Größe k haben, in jedem Graphen der Größe n höchstens $\lfloor n/k \rfloor^{k-(n \mod k)} (\lfloor n/k \rfloor + 1)^{n \mod k}$ ist. Für

alle maximalen unabhängigen Mengen mit der Größe höchstens k gilt dieselbe Grenze, solange $k \leq n/3$ ist. Für $k > n/3$ ist eine Grenze von näherungsweise $3^{n/3}$ gegeben. Da die unabhängigen Mengen im komplementären Graphen den Cliquen entsprechen, gelten diese Grenzen auch für maximale Cliquen. Für die Sicherstellung der Polynomialität kann nicht gefordert werden, dass alle maximalen Cliquen dieselbe Größe haben, weil sonst später die Reduktion von DEGREE 3 PREFERENCE GAME auf STRONG KERNEL dieses Kriterium nicht erfüllen würde. Denn bei dieser Reduktion wird ein Digraph mit maximalen Cliquen der Größe drei und zwei gebildet.

Satz 4.22. DEGREE 3 PREFERENCE GAME $\leq_p$ STRONG KERNEL, *das heißt* STRONG KERNEL *ist* PPAD*-schwer.*

Beweis. Gegeben sei ein Präferenzspiel mit der Spielermenge $[n]$. Der Digraph $D = (V, A)$ wird konstruiert. Für jeden Spieler i wird ein Knoten $\langle i, i\rangle$ eingeführt und für jeden Spieler j in $out(i)$ ein Knoten $\langle i, j\rangle$. Es gibt eine Kante von $\langle i, j\rangle$ nach $\langle i, k\rangle$, wenn j von i lieber gespielt wird als k, d.h. $j \succeq_i k$. Für jeden Knoten $\langle i, j\rangle$ mit $i \neq j$ gibt es einen zusätzlichen Knoten $J(i, j)$, der mit zwei weiteren Knoten verbunden ist, einmal durch eine von $\langle j, j\rangle$ kommende Kante, sowie durch eine nach $\langle i, j\rangle$ führende Kante. (Diese Konstruktion kann am Beispiel 5.20 nachvollzogen werden.) Die Anzahl der Knoten ist höchstens quadratisch in n und die Anzahl der Kanten höchstens kubisch. Also ist der Digraph polynomial in n.
Nun ist noch zu zeigen, dass ein stark fraktionaler Kernel f in Polynomialzeit auf ein Gleichgewicht w des Präferenzspiels abgebildet werden kann. Dafür wird

$$w_i(j) = \begin{cases} f(\langle i, j\rangle) & \text{, für } \langle i, j\rangle \in V \\ 0 & \text{, sonst} \end{cases}$$

gesetzt. Die Funktion f ist fraktional unabhängig. Das heißt, für jede Clique K gilt $\sum_{u \in K} f(u) \leq 1$. Außerdem ist sie stark dominierend. Es gibt also in jeder In-Nachbarschaft $I(v)$ eine Clique K, so dass $\sum_{u \in k} f(u) \geq 1$ ist. Dies wird auf die Cliquen der Gestalt $\{J(i, j), \langle i, j\rangle)\}$ angewendet. Es sei $C' = \{\langle i, j\rangle, J(i, j)\}$ und $C''\{J(i, j), \langle j, j\rangle\}$. Aufgrund der Konstruktion des Digraphen treten entweder beide Cliquen gleichzeitig oder keine von beiden auf. Außerdem ist die In-Nachbarschaft von $J(i, j)$ gleich der Clique C''. Da f ein stark fraktionaler Kernel ist, gelten die folgenden Ungleichungen: $\sum_{u \in C'} f(u) \leq 1$ und $1 \leq \sum_{u \in C''} f(u) \leq 1$. Daraus folgt $f(\langle i, j\rangle) \leq f(\langle j, j\rangle)$. Damit wurde gezeigt, dass immer $w_i(j) \leq w_j(j)$ erfüllt ist. Da ein stark fraktionaler Kernel per se immer nicht negativ ist, sind auch die Gewichtsfunktionen w_i nicht negativ. Für die Zulässigkeit bleibt noch zu zeigen, dass $\sum_{j \in [n]} w_i(j) = 1$ ist. Hierfür wird zunächst

$$I = \{i\} \cup out(i) \subseteq [n]$$

gesetzt. Dann ist die Menge $C = \{\langle i, j\rangle \mid j \in I\}$ nach Definition von D eine maximale Clique und gleichzeitig die In-Nachbarschaft des Knotens $\langle i, i\rangle$. Für alle $j \in [n] \setminus I$ wird $w_i(j) = 0$ gesetzt. Damit wurde

$$\sum_{j \in [n]} w_i(j) = \sum_{j \in I} w_i(j) = \sum_{j \in I} f(\langle i, j\rangle) = \sum_{u \in C} f(u) = 1$$

gezeigt.
Als Vorletztes bleibt noch zu beweisen, dass alle Gewichtsfunktionen w_i bezüglich w_{-i} lexikographisch maximal sind. Es wird ein beliebiger Knoten der Gestalt $\langle i, j\rangle$ betrachtet. Die Menge aller Knoten, die eine Kante nach $\langle i, j\rangle$ schicken, ist die Vereinigung zweier Cliquen. Die erste Clique C_1 ist die Menge aller Knoten $\langle i, k\rangle$ mit $k \succeq_i j$, die zweite Clique ist die Menge $C_2 = \{J(i, j), \langle i, j\rangle\}$. Wenn $w_i(j) \neq w_j(j)$ ist, dann ist wegen

$$\sum_{u \in C_2} f(u) = f(\langle i, j\rangle) + f(J(i, j)) < f(\langle j, j\rangle) + f(J(i, j)) = \sum_{u \in C''} = 1.$$

die Summe der Gewichte in der zweiten Clique kleiner als eins. Also muss C_1 die Clique in der In-Nachbarschaft von $\langle i, j\rangle$ sein, für die die Summe der Gewichte gleich eins ist. Wenn die Summe der Gewichte über der Clique C_2 gleich 1 ist, dann muss $w_i(j) = w_j(j)$ gelten (und anders herum). Damit wurde gezeigt, dass entweder $w_i(j) = w_j(j)$ ist oder $\sum_{k \succeq_{ij}} w_i(j) = 1$ gilt. Daraus folgt, dass es ein j gibt, so dass das Lemma 4.3 gilt.
Als Letztes wird noch gezeigt, dass der Digraph clique-azyklisch ist. Es gibt zwei verschiedene Typen von maximalen Cliquen. Der erste Typ ist von der Form $\{\langle i, j\rangle \mid j \in out(i) \subset V\}$ für ein gegebenes i. Wenn diese Cliquen einen Kreis enthalten würden, dann müsste es einen Kreis in der Präferenzordnung $\succ_i$ geben, was ein Widerspruch ist. Der zweite Typ maximaler Cliquen sind einzelne Kanten der Form $(\langle j, j\rangle, J(i, j))$ und $(J(i, j), \langle i, j\rangle)$. Keine von beiden kann einen Kreis enthalten. □

4.5 SCARF

Im Abschnitt 3.1 wurde das Lemma von Scarf 3.1 bereits eingeführt und bewiesen. Das zugehörige Suchproblem wird definiert. Im folgenden ist mit (B, b) das Gleichungssystem $Bx = b$ gemeint.

> SCARF: Gegeben seien zwei Matrizen B und C sowie ein Vektor $b \in \mathbb{R}^m$, die die Bedingungen des Lemmas von Scarf 3.1 erfüllen. Finde eine Teilmenge von m Spalten, die sowohl eine zulässige Basis für (B, b) als auch unterordnend für C ist.

Satz 4.23. SCARF *liegt in* PPAD*, das heißt* SCARF $\leq_p$ END OF THE LINE.

Beweis. Es soll gezeigt werden, dass der ursprüngliche Beweis von Scarf [2] zusammen mit einer Orientierungstechnik von Todd [19] ein END OF THE LINE-Argument für die Existenz einer unterordnenden und zulässigen Basis gibt, was gerade SCARF $\in$ PPAD beweist.
Als Erstes werden die Standardtechniken für die Perturbation angewandt, um Entartungen in der Eingabe zu entfernen. Das Paar (B, b) wird als *entartet* bezeichnet, wenn b in einem Kegel liegt, der von weniger als m Spalten von B aufgespannt wird. Andernfalls wird es als *nicht entartet* bezeichnet. Zunächst wird eine kleine Pertubation b' auf b angewandt, so dass das Paar (B, b') nicht entartet und jede zulässige Basis von (B, b') auch für (B, b) eine zulässige Basis ist. Solch eine Perturbation kann unter Verwendung von Standardtechniken der Linearen Programmierung in Polynomialzeit gefunden werden (vgl. Kapitel [10] von [20]).
Ähnlich liefert eine leichte Perturbation von C eine ordinal-generische Matrix C' (d.h. alle Elemente einer jeden Zeile von C' sind verschieden), die den Bedingungen des Lemmas von Scarf 3.1 genügt. Wenn die Perturbationen klein genug gewählt wurden, dann ist jede unterordnende Menge von C' auch für C unterordnend. Der Vollständigkeit halber wird eine Perturbation, die in Polynomialzeit erfolgen kann, präsentiert. Sei δ gleich das Minimum von $|c_{ij} - c_{i'j'}|$ über alle i, j, i', j' mit $c_{ij} \neq c_{i'j'}$. Das heißt δ ist der kleinste positive Abstand zwischen zwei beliebigen Einträgen von C. Die Einträge von C' werden wie folgt definiert:

$$c'_{ij} = \begin{cases} c_{ii} & \text{, für } j = i \in [m] \\ c_{ij} + \delta(j - m)/(n + 1) & \text{für } i \in [m], j > m \\ c_{ij} + \delta(n - m + j)/(n + 1) & \text{für } i \neq j \in [m] \end{cases}$$

Dann ist C' ordinal generisch, das heißt, alle Elemente einer Reihe von C' sind verschieden. Denn falls $c'_{ij} = c'_{ik}$ für beliebige $j \neq k$ wäre, dann müsste $c_{ij} + \delta_j = c_{ik} + \delta_k$ mit $0 < |\delta_j - \delta_k| < \delta$ sein. Das ist aufgrund der Wahl von δ aber nicht möglich.
Nun wird gezeigt, dass C' den Bedingungen des Lemmas von Scarf genügt. Das heißt für alle $i \neq j \in [m]$ und für alle $m < k \leq n$ gilt $c'_{ii} \leq c'_{ik} \leq c'_{ij}$. Dies folgt daraus, dass die Matrix C diese Ungleichungen erfüllt, also $c_{ii} \leq c_{ik} \leq c_{ij}$ gilt, und aus der Wahl von δ und der Definition der c'_{ii}, c'_{ij} und c'_{ik}.
Jetzt ist nur noch zu zeigen, dass jede unterordnende Menge von C' auch für C unterordnend ist. Es sei $c'_{ik} \leq c'_{ij}$ für i, j, k beliebig. Analog zum Beweis, dass C' ordinal generisch ist, folgt dann, dass $c_{ik} \leq c_{ij} + \delta'$ für ein δ' mit $|\delta'| < \delta$ ist. Aufgrund der Wahl von δ muss dann $\delta' = 0$ sein. Damit wurde $c_{ik} \leq c_{ij}$ bewiesen. Dies beschließt die gewünschte Behauptung.
Für den Rest des Beweises wird angenommen, dass (B, b) nicht entartet und C ordinal generisch sind. Der Beweis des ordinalen Pivotschritts (vgl. Lemma 3.6) geht davon aus, dass C ordinal generisch ist.
Jetzt beginnt der zweite Teil des Beweises. Der Beweis des Lemmas von Scarf 3.1 nutzt ein ungerichtetes END OF THE LINE-Argument. Um zu zeigen, dass SCARF in PPAD liegt, wird aber ein gerichtetes END OF THE LINE-Argument benötigt. Shapley [21] hat eine Index-Theorie für Bimatrixspiele eingeführt, die die Pfade, die durch den Lemke-Howson Algorithmus [16] erzeugt werden, orientiert und so die Gleichgewichtspunkte in zwei Mengen unterteilt. Todd [19] hat darauf basierend eine ähnliche Orientierungstheorie für verallgemeinerte komplementäre Pivotalgorithmen entwickelt. Nun wird Todds Orientierungstechnik angewandt, um zu zeigen, dass SCARF in PPAD liegt. Dafür werden die folgenden Definitionen und Lemmata gebraucht.

Definition 4.24. Es sei $X = [n]$. Eine m-elementige Teilmenge von X heißt *m-Teilmenge*. Mit X_m wird die Sammlung aller geordneten m-Tupel von verschiedenen Elementen aus X bezeichnet. Zwei m-Tupel in X_m sind genau dann *äquivalent*, wenn das eine eine gerade Permutation des anderen ist. Sei P ein beliebiges Element einer Äquivalenzklasse. Die korrespondierende Äquivalenzklasse wird mit $\overline{P}$ bezeichnet. Wenn $P' \in X_m$ eine ungerade Permutation von $P \in X_m$ ist, dann wird die Äquivalenzklasse $\overline{P'}$ das *Negative* von $\overline{P}$ genannt und $\overline{P'} = -\overline{P}$ geschrieben. Sei $P = (e_1, \ldots, e_m) \in X_m$. Mit $P \setminus e_i$ wird das Tupel $(e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_m) \in X_{m-1}$ bezeichnet. Für $\mu = \pm 1$ wird gesagt, dass $\mu \cdot (\overline{P \setminus e_i})$ positiv (negativ) in $\overline{P}$ enthalten ist, wenn $\mu \cdot (-1)^i$ positiv (negativ) ist. Für $f \notin P$ bezeichne $P \cup f$ das Tupel $(f, e_1, , \ldots, e_m) \in X_{m+1}$.

Definition 4.25. Es sei $X = [n]$ die Menge der Spaltenindizes von B und C. Im Folgenden seien $e \in X$, $\mathcal{F}$ die Menge aller zulässigen Basen, die e enthalten, und $\mathcal{S}$ die Menge aller ordinalen Basen, die e nicht enthalten. Dann sind alle Elemente aus $\mathcal{F}$ bzw. $\mathcal{S}$ m-elementige Teilmengen von $[n]$.
$V(\mathcal{F}, \mathcal{S}, e)$ sei die Menge der Paare $(\overline{F}, \overline{S})$, die jeweils einer der beiden folgenden Bedingungen genügen:

(i) $(\overline{F}, \overline{S})$ ist ein *gematchtes Paar*, das heißt $\overline{F} = \pm\overline{S}$, wobei entweder $\overline{F} \in \mathcal{F}$ oder $\overline{S} \in \mathcal{S}$ ist (beides gleichzeitig geht nicht, da der Schnitt von $\mathcal{F}$ und $\mathcal{S}$ leer ist)
(ii) $(\overline{F}, \overline{S})$ sind ein *ungematchtes Paar*, das heißt $(\overline{F}, \overline{S}) \in \mathcal{F} \times \mathcal{S}$ mit $e \in F, e \notin S$ und $F \setminus S = \{e\}$

Bemerkung 4.26. Ein gematchtes Paar $(\overline{T}, \overline{T})$ ist positiv, während ein gematchtes Paar $(\overline{T}, -\overline{T})$ negativ ist. Ein ungematchtes Paar $(\overline{F}, \overline{S})$ ist positiv (negativ), wenn F positiv (negativ) in $\overline{S \cup e}$ enthalten ist.

Lemma 4.27. *[Todd [19], S. 56]*

(a) Jedes gematchte Paar ist entweder durch einen zulässigen Pivotschritt oder durch einen ordinalen Pivotschritt mit einem ungematchten Paar adjazent, nicht beides gleichzeitig.
(b) Jedes ungematchte Paar ist durch einen zulässigen Pivotschritt mit einem Paar adjazent und durch einen ordinalen Pivotschritt mit einem Paar adjazent.

Lemma 4.28. *[Todd [19], S. 56]*

(a) Wenn zwei ungematchte Paare durch einen zulässigen Pivotschritt verbunden sind, dann haben sie entgegengesetzte Vorzeichen.
(b) Wenn ein gematchtes Paar und ein ungematchtes Paar durch einen zulässigen Pivot adjazent sind, dann haben sie dasselbe Vorzeichen.
(c) Wenn zwei Paare durch einen ordinalen Pivotschritt verbunden sind, dann haben sie entgegengesetzte Vorzeichen.

Ähnlich zu [19] wird ein gerichteter Graph konstruiert, dessen Ecken die Paare in $V(\mathcal{F}, \mathcal{S}, e)$ repräsentieren. Wenn zwei ungematchte Paare durch einen zulässigen Pivot verbunden sind, dann wird eine gerichtete Kante vom negativen zum positiven Paar hinzugefügt. Wenn ein gematchtes Paar durch einen zulässigen Pivot mit einem ungematchten Paar adjazent ist, dann wird eine Kante vom gematchten zum ungematchten Paar hinzugefügt, falls beide positiv sind, und in entgegengesetzter Richtung, falls beide negativ sind. Wenn zwei Paare durch einen ordinalen Pivotschritt adjazent sind, dann wird eine Kante vom positiven Paar zum negativen Paar ergänzt. Aus den Lemmata 4.27 und 4.28 folgt, dass jedes ungematchte Paar Eingangsgrad 1 und Ausgangsgrad 1 hat. Jedes positive gematchte Paar hat Eingangsgrad 0 und Ausgangsgrad 1. Jedes negative gematchte Paar hat Eingangsgrad 1 und Ausgangsgrad 0.
Es ist leicht zu sehen, dass $[m]$ in $\mathcal{F}$ und nicht unterordnend ist. Nach Lemma 3.6 (das ist der ordinale Pivotschritt) gibt es ein $f \neq e$, so dass $[m] - e + f$ in $\mathcal{S}$ ist. Das Paar $([m], [m])$ wird als die Anfangsquelle für END OF THE LINE genutzt und mit einem ordinalen Pivotschritt gestartet, der e (bspw. $e = 1$) aus $[m]$ entfernt. Dies liefert die gewünschte PPAD-Eigenschaft. □

Beispiel 4.29. In Abbildung 4.8 ist der gerichtete Pfad zu Beispiel 3.1.2 dargestellt. Hierbei bezeichnet beispielsweise der Eintrag -236 das (geordnete) Tupel $-(2, 3, 6)$, wobei die Menge $\{2, 3, 6\}$ eine ordinale Basis von C ist. In dem Paar $(167, 367)$ korrespondiert der erste Eintrag mit der zulässigen Basis $\mathcal{B}_z = \{1, 6, 7\}$ und der zweite Eintrag mit der ordinalen Basis $\mathcal{B}_o = \{3, 6, 7\}$. Die Vorzeichen der Tupel hängen von den Pivotschritten ab, und ob die Paare gematcht oder ungematcht sind.

Satz 4.30. STRONG KERNEL$\leq_p$ SCARF, *das heißt* SCARF *ist* PPAD*-schwer*

Beweis. Da in STRONG KERNEL verlangt wird, dass die Mächtigkeit der größten Clique so durch eine Konstante begrenzt wird, dass die Anzahl der maximalen Cliquen polynomial in n ist, kann der Digraph in Polynomialzeit auf die Matrizen B und C abgebildet und der Vektor $\mathbf{1} \in \mathbb{R}^m$ gebildet werden. Der Algorithmus von Scarf gibt eine zulässige Basis J aus, die gleichzeitig eine ordinale Basis ist. Da das Gleichungssystem $Bx = \mathbf{1}$ polynomial in n ist, und durch J die Struktur des Lösungsvektors bekannt ist, kann in Polynomialzeit eine Lösung x gefunden werden. Die Komponenten, die mit Knoten in V korrespondieren und ungleich null sind, liefern den gesuchten Kernel. Das heißt, die Ausgabe von SCARF kann in Polynomialzeit auf eine Ausgabe von STRONG KERNEL abgebildet werden. Also ist die Reduktion von STRONG KERNEL auf SCARF in Polynomialzeit möglich. □

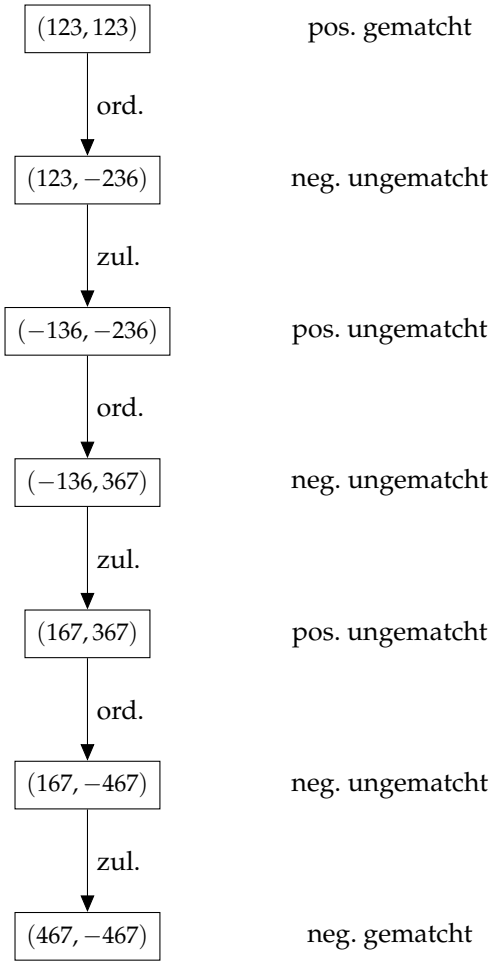


Abb. 4.8: Der gerichtete Graph zu Beispiel 3.1.2

5 Untersuchungen zur Komplexität der Kernelberechnung

In den vorhergehenden Kapiteln wurden die Voraussetzungen mit einem starken Bezug zur Literatur und einigen Ergänzungen zusammengetragen. Jetzt kann mit der eigentlichen Aufgabe, nämlich der Untersuchung der Komplexität von KERNEL begonnen werden. Mit dem Beweis der PPAD-Zugehörigkeit wird begonnen. Die vorgenommenen Vereinfachungen werden anschließend an einem Beispiel verdeutlicht. Daran schließt sich die Frage an, wie der Algorithmus sich verhält, wenn der eingegebene Digraph nicht clique-azyklisch ist. Danach wird getestet, ob der Beweis von Kintali et al. [1], der zeigt, dass STRONG KERNEL PPAD-schwer ist, auf KERNEL übertragen werden kann. Einige Bemerkungen zu einem möglichen polynomialen Verlauf von KERNEL beschließen dieses Kapitel.

5.1 Das Problem KERNEL

Begonnen wird mit der Wiederholung des Satzes 3.10: Jede clique-azyklische Superorientierung eines perfekten Graphen hat einen Kernel.
Dieser Satz wurde im Abschnitt 3.2 bewiesen. Der Beweis liefert einen Algorithmus für die Kernelberechnung, der auf dem Algorithmus von Scarf basiert. Es kann also das zugehörige totale Suchproblem definiert werden:

> KERNEL: Gegeben sei eine clique-azyklische Superorientierung $D = (V, A)$ eines perfekten Graphen $G = (V, E)$ mit $|V| = n$. Finde einen Kernel.

Bemerkung 5.1. Es wird noch einmal daran erinnert, dass eine Knotenmenge W genau dann eine Clique in D ist, wenn sie eine Clique im unterliegenden Graphen G ist, und dass nur maximale Cliquen betrachtet werden.

5.2 KERNEL ist in PPAD

Satz 5.2. KERNEL *ist in* PPAD.

Beweis. Es soll gezeigt werden, dass die Reduktion KERNEL$\leq_p$ END OF THE LINE in Polynomialzeit möglich ist. Im vorhergehenden Kapitel war bereits zu sehen, dass das Problem SCARF in PPAD liegt. Wenn

es gelingt zu zeigen, dass die notwendigen Daten polynomial in der Länge der Eingabe gespeichert werden können, und die entsprechenden Schritte ebenfalls polynomial in der Länge der Eingabe durchgeführt werden können sowie die Ordnung $>_i$ in Polynomialzeit bestimmt werden kann, dann liegt KERNEL ebenfalls in PPAD. Das folgende Beispiel zeigt, dass die Anzahl der maximalen Cliquen in einem perfekten Graphen durchaus exponentiell in der Anzahl der Knoten sein kann.

Beispiel 5.3. Sei G der perfekte Graph mit n Knoten, der aus $n/3$ disjunkten Dreiercliquen besteht (n ist hier natürlich ein Vielfaches von 3). Dieser hat $3^{n/3}$ maximale unabhängige Mengen. Nach dem schwachen Satz über perfekte Graphen 2.9 ist der komplementäre Graph $\overline{G}$ ebenfalls perfekt. Da die Cliquen in $\overline{G}$ gerade die unabhängigen Mengen in G sind (und umgekehrt), gibt es $3^{n/3}$ maximale Cliquen in $\overline{G}$. In Abbildung 5.1 wird ein solcher Graph für den Fall $n = 6$ dargestellt.

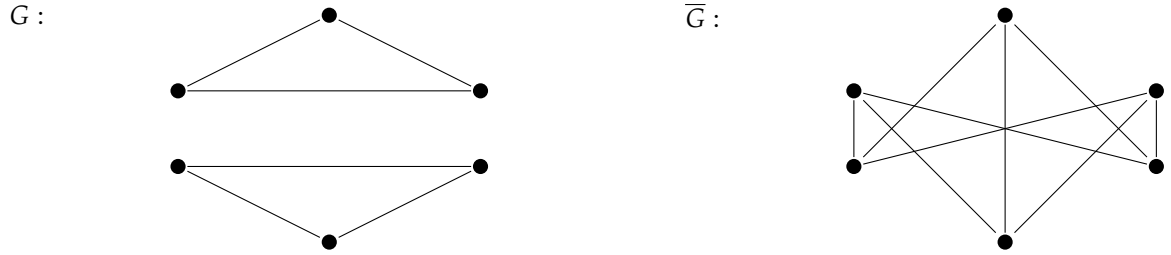


Abb. 5.1: Beispiel für einen perfekten Graphen G mit $n = 6$ Knoten, dessen Komplement $3^{n/3}$ maximale Cliquen enthält

Der Algorithmus zur Kernelberechnung ermittelt zunächst zu dem gegebenen Digraphen D' die folgenden Daten, mit denen der Algorithmus von Scarf durchlaufen wird:

m Einträge in der zulässigen Basis $\mathcal{B}_z$
m Einträge in der ordinalen Basis $\mathcal{B}_o$
$m \times (m + n)$ Cliquenmatrix B von D'
$m \times (m + n)$ Matrix C
m Zeilenminimierer u_i,

wobei m die Anzahl der maximalen Cliquen in G (und damit auch in D) ist, welche bekanntlich exponentiell in der Knotenanzahl n sein kann. Folglich können alle diese Daten exponentiell sein. Also sind die folgenden Punkte zu klären:

1.) Gibt es eine Möglichkeit, sich diese Daten polynomial in n zu merken und dabei alle notwendigen Informationen zu erhalten bzw. diese schnell rekonstruieren zu können?
2.) Ist die Suche nach dem Pivotelement für den zulässigen Pivotschritt polynomial in n möglich? Das heißt, ist es möglich mit nur polynomial vielen der m maximalen Cliquen das Pivotelement zu finden?
3.) Ist die Suche nach dem Element, das beim ordinalen Pivotschritt in $\mathcal{B}_o$ aufgenommen wird, mit polynomialem Aufwand möglich?
4.) Ist es möglich, den Kernel mit einem linearen Gleichungssystem zu berechnen, das polynomial in der Länge der Eingabe ist?
5.) Kann die Ordnung $>_i$ in Polynomialzeit berechnet werden?

All dies soll untersucht werden. Hierfür wird noch einmal daran erinnert, dass der gegebene Digraph n Knoten hat, also versucht wird zu zeigen, dass all dies polynomial in n möglich ist.

Vorbemerkung 5.4. Im Folgenden werden einerseits die Begriffe „Zeile“ und „Clique“ und andererseits „Spalte“ und „Knoten“ teilweise synonym gebraucht. Die Zeilen der Matrizen werden manchmal mit i und manchmal mit C_i' resp. C_i bezeichnet. Beispielsweise ist c_{iv_j} derselbe Eintrag wie $c_{C_i'v_j}$ in der Matrix C.

Vereinfachung der Basen und der Zeilenminimierer

Lemma 5.5. *Wenn für jede Spalte $v_j \in \mathcal{B}_o \cap V$ gespeichert wird, in welcher Zeile C_i' der zugehörige Zeilenminimierer liegt, dann können daraus die ordinale Basis und alle Zeilenminimierer schnell rekonstruiert werden. Das heißt, mit höchstens $2n$ Daten können $2m$ Daten bestimmt werden.*

Beweis. Die Knotenmenge des Digraphen D' soll wieder aus n Knoten $v_j \in V$ und m Zusatzknoten z_i bestehen. Für die ordinal generische Matrix C gelten die Ungleichungen

$$0 = c_{iz_i} < c_{iv_k} < c_{iz_j}$$

für alle $i \neq j \in [m]$ und alle $k \in [n]$. Daraus folgt

$$u_i = 0 \Leftrightarrow u_i = c_{iz_i} \Leftrightarrow z_i \in \mathcal{B}_o,$$

für alle $i \in [m]$. Angenommen, der Knoten z_i liegt nicht in der ordinalen Basis. Dann muss mindestens ein Knoten aus V in der ordinalen Basis enthalten sein. Wegen der Ungleichungen für die Matrix C gilt deshalb

$$u_i > 0 \Leftrightarrow u_i = c_{iv_j} \text{ für ein } v_j \in \mathcal{B}_o \cap V.$$

Da $|V| = n$ ist und jede Spalte aus $\mathcal{B}_o$ genau einen Zeilenminimierer enthält, können höchstens n Zeilenminimierer existieren, die größer als null sind. Wenn nun zu jedem Knoten aus $V \cap \mathcal{B}_o$ bekannt ist, in welcher Zeile der Zeilenminimierer liegt, dann können daraus die Zusammensetzung der ordinalen Basis und der Zeilenminimierer hergeleitet werden:
Wenn eine Zeile C_i' mit dem Knoten v_j verbunden ist, dann liegt dieser Knoten in der ordinalen Basis und $c_{iv_j} = u_i$. Andernfalls ist z_i in der ordinalen Basis, und der Zeilenminimierer u_i ist gleich null, also gleich c_{iz_i}. □

Beispiel 5.6. Es seien $m = 4$ und die folgenden Daten gegeben (in der Zeile ZM stehen die zu den Knoten gehörenden Zeilen, die den Zeilenminimierer (ZM) enthalten):

Knoten	v_1	v_2	v_3	v_4	v_5
ZM	C_4'	C_2'	C_3'	C_1'	-

Dann ist die ordinale Basis $\mathcal{B}_o = \{v_1, v_2, v_3, v_4\}$ und enthält keinen Zusatzknoten. Und die Zeilenminimierer sind $u_1 = c_{1v_4}, u_2 = c_{2v_2}, u_3 = c_{3v_3}$ sowie $u_4 = c_{4v_1}$.

Lemma 5.7. *Wenn zu jedem Knoten $v_j \in \mathcal{B}_z \cap V$ gespeichert wird, mit welcher Pivotzeile er in die zulässige Basis aufgenommen wurde, dann kann daraus die zulässige Basis $\mathcal{B}_z$ rekonstruiert werden. Das heißt, aus höchstens $2n$ Daten können die m Elemente in der zulässigen Basis hergeleitet werden.*

Beweis. Dies ist möglich, weil die zulässige Basis zu Beginn des Algorithmus ausschließlich Zusatzknoten enthält. Die zulässige Basis enthält m Elemente, und es gibt m Zusatzknoten. Es wurde also tatsächlich ein Pivotschritt durchgeführt, wenn sich ein Knoten v_j in der zulässigen Basis befindet. Beim ersten Pivotschritt kann nur ein Knoten aus V in die zulässige Basis aufgenommen werden. Dabei muss zwangsweise ein Zusatzknoten aus der Basis entfernt werden. Wenn bei einem späteren Pivotschritt v_j in die Basis aufgenommen wird, dann wird entweder ein Zusatzknoten oder ein anderer Knoten v_k aus der Basis entfernt. Im zweiten Fall übernimmt v_j die Pivotzeile von v_k. Der Zusatzknoten, der mit dieser Zeile korrespondiert, befand sich zu diesem Zeitpunkt nicht mehr in der Basis, weil er zuvor schon entfernt wurde. Analog zum ordinalen Fall kann die zulässige Basis höchstens n Knoten enthalten, die keine Zusatzknoten sind. Das heißt, es sind höchstens $2n$ Daten zu speichern. Wenn die Spalten aus $\mathcal{B}_z \cap V$ und die zugehörigen Pivotzeilen bekannt sind, dann kann anhand der nicht gemerkten Zeilen herausgefunden werden, welche Zusatzknoten in der zulässigen Basis liegen. □

Beispiel 5.8. Gegeben seien $m = 4$ und die folgende Tabelle (mit Pivot ist die Pivotzeile gemeint):

Knoten	v_1	v_2	v_3	v_4	v_5
Pivot	C'_2	-	C'_3	C'_4	-

Daraus kann die zulässige Basis $\mathcal{B}_z = \{v_1, v_3, v_4, z_1\}$ abgeleitet werden (die Clique C'_1 kommt in der Zeile Pivot nicht vor, deshalb muss z_1 in der Basis sein).

Vereinfachung des ordinalen Pivotschritts

Lemma 5.9. *Für die Durchführung des ordinalen Pivotschritts genügt es, eine höchstens $(n \times n)$ - Teilmatrix $\hat{C}$ von C zu kennen. Diese Matrix muss nicht bei jedem ordinalen Pivotschritt neu gebildet werden, sondern wird peu à peu aufgebaut. Da der Pivotschritt nur ordinale Vergleiche umfasst, ist er folglich in Polynomialzeit durchführbar.*

Beweis. Es wird wieder davon ausgegangen, dass es n Knoten und m Zusatzknoten sind.
Vor Beginn des eigentlichen Beweises wird der Ablauf des ordinalen Pivotschritts in groben Zügen wiederholt: Gegeben seien eine ordinale Basis $\mathcal{B}_o$ mit der Kardinalität m, sowie die zugehörigen Zeilenminimierer $u_1, \ldots, u_m$, wobei natürlich bekannt ist, welcher Zeilenminimierer in welcher Spalte steht. Bei einem ordinalen Pivotschritt wird zunächst ein zuvor bestimmter Knoten y aus der ordinalen Basis $\mathcal{B}_o$ entfernt. Hierbei kann y sowohl ein Knoten $v_j \in V$ als auch ein Zusatzknoten z_i sein. Beides ist möglich. Anschließend werden die Zeilenminimierer $u'_i = \min\{c_{ij} \mid j \in \mathcal{B}'_o\}$ der in $\mathcal{B}_o$ verbleibenden Spalten berechnet und die Spalte identifiziert, die nun zwei Zeilenminimierer enthält, einen alten $u'_{i_a} = u_{i_a}$, der zuvor bereits ein Zeilenminimierer war, und einen neuen $u'_{i_n} \neq u_{i_n}$. In der Zeile i_a des alten Zeilenminimierers wird der Eintrag $c_{i_a r}$ in C gesucht, dessen zugehörige Spalte r die in $\mathcal{B}_o$ verbliebenen Spalten zu einer ordinalen Basis ergänzt. Diese Spalte r wird in die ordinale Basis aufgenommen. Die neue ordinale Basis ist dann $\mathcal{B}_o - y + r$. Bis auf zwei Zeilenminimierer bleiben alle gleich. Der Zeilenminimierer u_{i_n} wird durch u'_{i_n} ersetzt, und $c_{i_a r}$ ersetzt den Zeilenminimierer u_{i_a}. Damit ist der ordinale Pivorschritt abgeschlossen.
Es soll geklärt werden, welche Einträge der Matrix C benötigt werden, um den neuen Zeilenminimierer u'_{i_n} zu finden und den Eintrag $c_{i_a r}$ zu bestimmen:
Es sei y die zu entfernende Spalte, und u_k der Zeilenminimierer in y. Nun wird y aus der ordinalen Basis entfernt, und anschließend werden die Zeilenminimierer u'_i berechnet. Dabei verändern sich die

Zeilenminimierer in den Spalten ungleich y nicht, denn das, was vor dem Entfernen von y minimal war, bleibt minimal. Deshalb muss u'_k der neue Zeilenminimierer sein. Das heißt, es müssen nicht alle Zeilenminimierer u'_i berechnet werden, um den neuen Zeilenminimierer u'_{i_n} zu bestimmen, sondern es braucht nur die Zeile betrachtet zu werden, in welcher der Zeilenminimierer der aus $\mathcal{B}_o$ zu entfernenden Spalte lag. In unserem Fall ist der neue Zeilenminimierer in der Zeile C'_k zu suchen. Aufgrund der Ungleichungen für die Matrix C muss der neue Zeilenminimierer $u'_{i_n} = u'_k$ in $\mathcal{B}_o \cap V$ sein. (Denn sonst würde sich z_k in der Basis befinden. In diesem Fall hätte der aus $\mathcal{B}_o$ entfernte Knoten y nicht den Zeilenminimierer u_k haben können.) Also werden nur die n Einträge in der Zeile C'_k, die mit Knoten aus V korrespondieren, benötigt, um u'_k zu finden. Dies entspricht übrigens gerade der Clique C_k im nicht erweiterten Digraphen D. Damit wurde gezeigt, dass n Einträge von C für das Bestimmen des neuen Zeilenminimierers gebraucht werden. Als Zweites wird geklärt, welche der Einträge der Matrix C für die Berechnung von $c_{i_a r}$ notwendig sind. Angenommen, $u'_k = c_{ks}$ wurde gefunden. Die Spalte s lag zuvor bereits in $\mathcal{B}_o$ und hat folglich einen Zeilenminimierer enthalten, der nicht in der Zeile k liegt. Dieser bleibt auch für $\mathcal{B}_o - y$ ein Zeilenminimierer. Folglich enthält s zwei Zeilenminimierer, den alten u_{i_a} und den neuen $u'_{i_n} = u'_k$. Da der alte Zeilenminimierer bereits bekannt ist, muss nichts gerechnet werden, um i_a zu identifizieren. Im Algorithmus wird zunächst die Menge

$$\mathcal{M} = \{k \in V' \mid c_{ik} > u'_i \; \forall i \in [m] - i_a\}$$

gebildet. Das ist die Menge aller Spalten, deren $m-1$ Einträge in den Zeilen $i \neq i_a$ größer als die jeweiligen Zeilenminimierer u_i sind. Aus diesen Spalten wird diejenige ausgewählt, deren Eintrag in der Zeile i_a maximal ist. Das heißt, es wird $\max\{c_{i_a j} \mid j \in \mathcal{M}\} = c_{i_a r}$ bestimmt. So wird sichergestellt, dass $\mathcal{B}_o + r$ unterordnend, also eine ordinale Basis, ist. Aus den Ungleichungen für die Matrix C folgt, dass der Zusatzknoten z_{i_a} immer in $\mathcal{M}$ enthalten ist, und alle anderen Zusatzknoten $z_i \neq z_{i_a}$ nicht in $\mathcal{M}$ liegen können. Also ist

$$\mathcal{M} = \{z_{i_a}\} \dot\cup (\mathcal{M} \cap V) =: \{z_{i_a}\} \dot\cup \hat{\mathcal{M}}.$$

Das heißt, es sind höchstens n Spalten der Matrix C zu untersuchen, um die Menge $\hat{\mathcal{M}}$ zu finden. Aus den Ungleichungen für die Matrix C folgt, dass die Einträge aller Spalten in $\hat{\mathcal{M}}$ größer als Null sind. Mit anderen Worten, die Zeilenminimierer u_i mit $i \neq i_a$, die gleich Null sind (also mit Zusatzknoten korrespondieren), spielen bei der Bildung der Menge $\hat{\mathcal{M}}$ keine Rolle. Sie haben keine einschränkende Wirkung, weil sowieso alle Einträge größer als Null sind. Darum können diese vernachlässigt werden. Deshalb ist

$$\hat{\mathcal{M}} = \{v_k \in V \mid c_{ik} > u'_i \; \forall i \in [m] - i_a \text{ mit } u'_i > 0\}$$

Dies schränkt die Menge der zu betrachtenden Zeilen auf die Zeilen mit echt positiven Zeilenminimierern ein. Aus dem Beweis des Lemmas 5.5 ist bekannt, dass es höchstens n Zeilenminimierer geben kann, die echt größer als Null sind. (Denn jede Spalte der ordinalen Basis enthält genau einen Zeilenminimierer. Ein Zeilenminimierer ist genau dann größer als Null, wenn er mit einem Knoten v_j korrespondiert, welcher in der ordinalen Basis enthalten ist.) Damit wurde gezeigt, dass höchstens n Zeilen bekannt sein müssen, um die Menge $\hat{\mathcal{M}}$ zu bestimmen. Dies lässt sich noch weiter einschränken: Im Abschnitt 3.5 wurde gezeigt, dass immer $u'_{i_n} > u_{i_n}$ und $u'_{i_a} = u_{i_a} > c_{i_a r}$ gilt. Das heißt, $u'_{i_a} > 0$. Folglich gibt es höchstens $n-1$ Zeilenminimierer u'_i ungleich u_{i_a}, die zur Bestimmung von $\hat{\mathcal{M}}$ herangezogen werden können. Da der Zeilenminimierer u'_{i_n} ebenfalls positiv sein muss, ist er einer davon. Also wird $\hat{\mathcal{M}}$ durch eine höchstens $(n-1) \times n$-Matrix bestimmt. Die n Einträge in der Matrix C'_{i_n} (das ist gerade die obige Clique C_k) sind darin enthalten.

Die Matrix $\hat{C}$ soll aus der $(n-1) \times n$-Teilmatrix von C bestehen, mit deren Hilfe $\hat{\mathcal{M}}$ berechnet wird, sowie aus der Zeile C_{i_a}, falls diese nicht in den $n-1$ anderen Zeilen bereits enthalten ist. Diese Teilmatrix $\hat{C}$ von C besteht also aus n Spalten und höchstens n Zeilen und enthält alle Daten, die für die Durchführung des

ordinalen Pivotschritts benötigt werden.
Nun ist noch der sukzessive Aufbau von $\hat{C}$ zu beschreiben: Die ordinale Basis besteht im ersten Schritt aus allen Zusatzknoten ungleich z_1 sowie einem Knoten aus V. Also ist der Zeilenminimierer $u_1 > 0$ und alle anderen sind gleich Null. Und $\hat{C}$ ist die $(1 \times n)$-Teilmatrix, die aus der Zeile C_1 besteht (das sind die letzten n Einträge der ersten Clique C_1'). Bei einem ordinalen Pivotschritt wird ein Knoten w aus der ordinalen Basis entfernt. Es werden zwei Fälle unterschieden. (a) Wenn w ein Zusatzknoten z_i ist, dann wird dabei die Teilmatrix $\hat{C}$ um die zugehörige Clique C_i erweitert. Wenn dafür ein anderer Zusatzknoten in die Basis aufgenommen wird, dann wird dabei die Clique, die diesen Zusatzknoten enthält, aus $\hat{C}$ entfernt (vergleiche Beispiel **??**), weil der entsprechende Zeilenminimierer gleich Null wird. Wenn stattdessen ein Knoten aus V aufgenommen wird, dann verändert sich $\hat{C}$ nicht noch einmal. (b) Wenn w ein Knoten aus V ist, und ein Knoten aus V für diesen in die ordinale Basis aufgenommen wird, dann wird die Teilmatrix dabei nicht verändert. Wenn aber ein Zusatzknoten neu in die Basis kommt, dann wird die entsprechende Zeile aus der Matrix gelöscht. So kann die Teilmatrix $\hat{C}$ schrittweise aufgebaut werden. □

Vereinfachung des zulässigen Pivotschritts

Ausgangspunkt für diesen Abschnitt ist die Frage, wie die Wahl der Pivotzeile so gestaltet werden kann, dass nicht die gesamte Matrix B gebraucht wird, und kein Gedanke daran verschwendet werden muss, wie sich das Tableau bei jedem Pivotschritt verändert. Zusätzlich sollte die Wahl des Pivotelements in Polynomialzeit durchführbar sein. Bekanntlich terminiert der Algorithmus, wenn entweder der Zusatzknoten z_1 mit einem ordinalen Pivotschritt in die ordinale Basis aufgenommen oder mit einem zulässigen Pivotschritt aus der zulässigen Basis entfernt wird. Dieser Zusatzknoten ist ausschließlich in der ersten maximalen Clique C_1' enthalten. Deshalb hat die erste Clique eine besondere Rolle. Daraus ist die Überlegung entstanden, die erste Clique so zu wählen und die Regel für die Wahl des Pivotelements so zu gestalten, dass die Clique C_1' erst dann angesprochen werden kann, wenn der Zusatzknoten z_1 entfernt werden muss (der ordinale Fall interessiert gerade nicht). Es wird ein Ansatz entwickelt und überprüft, ob er alle gewünschten Eigenschaften hat. Dafür wird zunächst eine lexikographische Ordnung auf der Knotenmenge eingeführt. Anschließend wird geklärt, wie eine lexikographisch kleinste maximale Clique, die einen bestimmten Knoten enthält, sowie die lexikographisch größte maximale Clique in einem perfekten Graphen gefunden werden können. Anschließend wird dies bei der Vereinfachung des zulässigen Pivotschritts verwendet.

Definition 5.10. Gegeben sei ein totalgeordnetes Alphabet. Dann kann eine *lexikographische Ordnung* wie folgt beschrieben werden: Eine Zeichenkette a ist kleiner als eine Zeichenkette b (d. h. a liegt in der Sortierung vor b), wenn

> entweder das erste Zeichen von a, in dem sich beide Zeichenketten unterscheiden, kleiner ist als das entsprechende Zeichen von b,
> oder wenn a den Anfang von b bildet, aber kürzer ist.

Diese Bezeichnung leitet sich aus der Sortierung im Wörterbuch oder Lexikon ab.

Definition 5.11. Gegeben sei ein Graph $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$. Für diesen wird die lexikographische Ordnung für Knotenmengen $U, W \subseteq V$ definiert, indem die natürliche Ordnung von [n] verwendet wird, um die Knoten innerhalb von U und W anhand ihrer Indizes aufsteigend zu ordnen und anschließend die Indexfolgen von U und W lexikographisch miteinander zu vergleichen.

Beispiel 5.12. Die Knotenmenge $U = \{v_1, v_3, v_7, v_9\}$ ist lexikographisch größer als $W = \{v_1, v_2, v_{10}, v_{12}, v_{15}\}$, weil die Indexmenge $(1, 3, 7, 9)$ lexikographisch größer als $(1, 2, 10, 12, 15)$ ist.

Lemma 5.13. *In einem Graphen kann eine lexikographisch kleinste maximale Clique, die einen bestimmten Knoten v enthält, in Polynomialzeit gefunden werden. Eine solche Clique wird im Folgenden auch kurz als* (lexikographisch) kleinste Clique von v *bezeichnet.*

Beweis. Es seien $G = (V, E)$ und für alle Knoten v bezeichne $N(v) = v + \{w \mid (v, w) \in G\}$ die Menge, die alle mit v adjazenten Knoten und v selbst enthält. Es wird die lexikographisch kleinste Clique gesucht, die den Knoten $v_{i_0} \in V$ enthält. Offensichtlich gilt für jede maximale Clique

$$C = \bigcap_{v \in C} N(v).$$

Das wird ausgenutzt, wenn die gesuchte Clique C gebildet wird. Zunächst wird v_{i_0} in C aufgenommen, anschließend wird der kleinste Knoten v_{i_1} aus der Menge $N(v) \setminus C$ in C aufgenommen. Im dritten Schritt wird als Erstes geklärt, ob der Schnitt der beiden Nachbarschaften $N(v_{i_0})$ und $N(v_{i_1})$ abzüglich der Menge C nichtleer ist. In diesem Fall wird der Knoten mit dem kleinsten Index aus dieser Menge aufgenommen etc. Als Algorithmus sieht das so aus:

$C = v_{i_0}$;
$j = 1$;
While $\left(\bigcap_{v \in C} N(v) \setminus C\right) \neq \emptyset$
$\{$ $i_j = \min\{i \mid v_i \in \left(\cap_{v \in C} N(v)\right) \setminus C\}$;
$C = C + v_{i_j}$;
$j++$;
$\}$

Es lässt sich leicht nachprüfen, dass dies tatsächlich eine lexikographisch kleinste Clique liefert, die v_{i_0} enthält. Der Algorithmus endet nach höchstens $|N(v_{i_0})|$ Schritten (wenn der Start $C = v_{i_0}$ als extra Schritt gezählt wird). Er besteht nur aus Suchen in Mengen mit der Kardinalität höchstens n sowie Schnittmengenbildung. □

Lemma 5.14. *Es ist möglich in einem Graphen eine lexikographisch größte maximale Clique in Polynomialzeit zu finden, wenn es erlaubt ist, die Knoten umzunummerieren.*

Beweis. Wenn zu einer gegebenen Knotennummerierung eine lexikographisch größte maximale Clique gefunden werden soll, dann ist das nicht immer in Polynomialzeit möglich ([22] gibt dazu einen guten Überblick). Glücklicherweise gilt diese Bedingung in diesem Fall nicht. Deshalb wird zunächst eine beliebige maximale Clique C im Graphen G gesucht. Das kann auch so eine wie in Lemma 5.13 sein. Es wird angenommen, dass C aus k Knoten besteht, und der Graph n Knoten hat. Nun werden die Knoten in G so umnummeriert, dass die Knoten in C die m größten Indizes erhalten (das sind alle Indizes in $[n] \setminus [n-k]$). Dann ist die Clique C im umnummerierten Graphen immer noch maximal, und es ist leicht zu sehen, dass es keine andere maximale Clique gibt, die lexikographisch größer ist. □

Nun steht das notwendige Rüstzeug bereit, um den zulässigen Pivotschritt zu vereinfachen.

Lemma 5.15. *Wenn in D als Startclique C_1 die lexikographisch größte Clique und als Pivotzeile der Spalte y immer die lexikographisch kleinste maximale Clique, die y enthält, gewählt werden, dann ist der Pivotschritt in Polynomialzeit durchführbar. Für die Durchführung des zulässigen Pivotschritts genügt es, eine höchstens $(n \times n)$-Teilmatrix $\hat{B}$ der Cliquenmatrix B zu kennen. Diese Teilmatrix muss nicht bei jedem zulässigen Pivotschritt neu gebildet werden, sondern wird schrittweise aufgebaut.*

Beweis. Mit der Wahl der Startclique und der Pivotzeile wird begonnen und gezeigt, dass diese Wahl sinnvoll ist. Anschließend wird die Anzahl der notwendigen Daten abgeschätzt.
Alle Cliquen sollen in dem Digraphen D berechnet werden. Auf diesen beziehen sich auch die lexikographischen Vergleiche. Einerseits ist so das Finden der Cliquen polynomial in n sicher gestellt, andererseits reicht das auch völlig aus, denn zwei Cliquen C'_i und C'_j müssen gleich sein, wenn sie auf der Knotenmenge V übereinstimmen. Sonst würden sie sich nur in dem Zusatzknoten unterscheiden, was nach der Konstruktion des erweiterten Digraphen D' nicht möglich ist. Außerdem wird der Zusatzknoten z_i durch die Clique C_i bestimmt und nicht umgekehrt.
Es sei C_1 die lexikographisch größte Clique in D (die Ergänzung des Zusatzknotens z_1 liefert C'_1). Die Pivotzeile sei immer die lexikographisch kleinste maximale Clique in D, die y enthält. Dies ist möglich, da der Algorithmus unabhängig von der gewählten Knotennummerierung und Reihenfolge der Cliquen funktioniert. Diese Cliquen können polynomial in n gefunden werden (Lemmata 5.13 und 5.14).
Falls die kleinste Clique von y mit der Startclique C_1 übereinstimmt, dann wird beim zulässigen Pivot der Zusatzknoten z_1 aus der Basis $\mathcal{B}_z$ zu entfernt, und der Algorithmus terminiert. Eine andere Pivotwahl wäre für den Knoten y in diesem Fall gar nicht möglich. Angenommen, y wäre in einer weiteren Clique enthalten, dann wäre diese Clique nicht die lexikographisch größte Clique. Folglich wäre C'_1 nicht die lexikographisch kleinste Clique, die y enthält, was der Wahl von C'_1 widerspräche.
Es ist noch zu klären, ob sich das Pivotelement überhaupt in der lexikographisch kleinsten maximalen Clique von y befinden kann, oder ob der entsprechende Eintrag im aktuellen Tableau kleiner oder gleich Null sein kann. Da alle Einträge der Cliquenmatrix B aus $\{0, 1\}$ sind, ist nur der Fall gleich Null zu betrachten. Dies wäre genau dann der Fall, wenn es einen Knoten $w \in \mathcal{B}_z$ gäbe, so dass einerseits dessen kleinste Clique auch den Knoten y enthält, und der andererseits wiederum in der kleinsten Clique von y enthalten ist. Denn dann entstünde beim Herstellen des Einheitsvektors in der Spalte w sowohl für w als auch für y in der kleinsten Clique für y eine Null. Angenommen, es gibt zwei solche Cliquen. Das heißt, es seien C'_k die kleinste Clique von w, die auch y enthält, und C'_l sei die kleinste Clique von y, die auch w enthält. Wenn diese zwei Cliquen verschieden sind, dann ist eine der beiden lexikographisch kleiner als die andere. Ohne Beschränkung der Allgemeinheit sei $C'_k < C'_l$. Das heißt C'_k ist kleiner als C'_l und enthält y. Das ist ein Widerspruch zur Wahl von C'_l. Also müssen die zwei Cliquen übereinstimmen. Folglich ist es nicht möglich, dass in der kleinsten Clique von y der Eintrag in der Spalte y gleich Null ist.
Gegeben sei nun eine zulässige Basis $\mathcal{B}_z$ mit der Kardinalität m. Analog zur Vereinfachung der zulässigen Basis (vgl. Lemma 5.7) wird davon ausgegangen, dass zu jedem Knoten $v_j \in \mathcal{B}_z$ die Pivotzeile bekannt ist, mit der er in die zulässige Basis aufgenommen wurde. Die Pivotzeilen sind gerade deren lexikographisch kleinsten maximalen Cliquen. Es ist bekannt, dass jeder Zusatzknoten z_i in der Basis nur mit der Clique C'_i korrespondieren kann. Dieser Zusammenhang zwischen Cliquenindex und Knotenindex besteht für die Knoten aus V nicht.
Die Teilmatrix $\hat{B}$ bestehe aus den höchstens n Zeilen, die mit den kleinsten Cliquen der Knoten aus $V \cap \mathcal{B}_z$ korrespondieren, und enthalte von diesen jeweils die letzten n Spalten $v_1, \ldots, v_n$. Das heißt, $\hat{B}$ ist höchstens eine $(n \times n)$-Matrix. Wenn nun ein Knoten y in die zulässige Basis aufgenommen werden soll, dann wird überprüft, ob dessen kleinste Clique C'_i bereits in $\hat{B}$ enthalten ist. Wie oben dargelegt wurde, genügt für diesen Vergleich die Kenntnis der Spalten $v_1, \ldots, v_n$. (a) Wenn sich C'_i bereits in $\hat{B}$ befindet, dann wird die zugehörige Spalte aus der zulässigen Basis entfernt. Falls der aufzunehmende Knoten y ein Zusatzknoten ist, dann ist auch die Clique C'_i aus $\hat{B}$ zu entfernen, weil sie nicht mehr mit einem Knoten aus V korrespondiert. (b) Wenn die Clique sich noch nicht in $\hat{B}$ befindet, dann korrespondiert sie mit einem Zusatzknoten $z_i \in \mathcal{B}_z$. In diesem Fall, wird der Zusatzknoten aus der Basis entfernt, und die Clique in der Matrix $\hat{B}$ ergänzt.
Wenn die zulässigen Pivotschritte auf diese Art durchgeführt werden, sind die Zusammensetzung der

zulässigen Basis und die Teilmatrix $\hat{B}$ bekannt. Allerdings ist unklar, wie das aktuelle Tableau tatsächlich aussieht. □

Berechnung des Kernels mit verkleinertem Gleichungssystem

Lemma 5.16. *Die für die Kernelberechnung notwendigen Komponenten des Lösungsvektors x können mit einem Gleichungssystem berechnet werden, das aus höchstens n Gleichungen und n Unbekannten besteht.*

Beweis. Angenommen, der Algorithmus wurde durchlaufen und mit $\mathcal{B}_z = \mathcal{B}_o =: \mathcal{B}$ beendet. Die zugehörigen Teilmatrizen $\hat{B}$ und $\hat{C}$ stimmen dahingehend überein, dass sie dieselben Zeilenindizes (also Cliquen) und Spaltenindizes (die Knoten aus V) haben. Da kein aktuelles Tableau zur Verfügung steht, kann die zugehörige Lösung nicht einfach abgelesen werden, sondern wird durch Einsetzen des Vektors x mit

$$x_i = \begin{cases} 0 & \text{, für } i \notin \mathcal{B} \\ x_i & \text{, für } i \in \mathcal{B} \end{cases}$$

in das Startgleichungssystem $Bx = \mathbf{1}$ und das anschließende Lösen desselben gefunden. Wenn, wie oben beschrieben, beim Durchlaufen des Algorithmus nur die notwendigen Daten gespeichert werden, dann sind am Ende zwar die Struktur der Lösung sowie die Teilmatrizen $\hat{B}$ und $\hat{C}$, die mit den Knoten in $\mathcal{B} \cap V$ zusammenhängen, bekannt. Aber das vollständige Tableau ist nicht bekannt, sondern nur ein Teil desselben. Das ist das System $\hat{B}\hat{x} = \mathbf{1}$, wobei $\hat{x} = (x_{v_1}, \ldots, x_{v_n})$ ist. Die Frage ist, ob dieses ausreicht, um den Kernel zu finden. Dafür wird das vollständige Tableau $Bx = \mathbf{1}$ näher betrachtet. Dieses besteht aus m Gleichungen und m Unbekannten. Nun sind zwei Fälle zu unterscheiden. Als Erstes wird angenommen, dass sich der Zusatzknoten z_i nicht in der Basis $\mathcal{B}$ befindet. Dann ist die entsprechende Komponente x_{z_i} des Lösungsvektors gleich Null. Das heißt, es spielt für die Lösung des Gleichungssystems überhaupt keine Rolle, dass z_i in der Clique C_i' enthalten ist. Stattdessen kann einfach mit der Gleichung $C_i\hat{x} = 1$ gerechnet werden, ohne dadurch irgendwelche Informationen zu verlieren. Wenn andererseits angenommen wird, dass der Zusatzknoten z_i in der Basis liegt, dann wird der Wert der Komponente x_{z_i} durch die Gleichung $C_i'x = 1$ bestimmt (der Zusatzknoten ist ja nur in einer einzigen Clique enthalten). Wenn es eine andere Komponente x_{v_i} gäbe, deren Wert ebenfalls von dieser Gleichung abhinge, dann gäbe es sozusagen eine Gleichung mit zwei Unbekannten, womit das Ergebnis nicht mehr eindeutig wäre. Ergo haben die Gleichungen $C_i'x = 1$, die mit Zusatzknoten $z_i \in \mathcal{B}$ korrespondieren, keine Bedeutung für die Bestimmung von $\hat{x}$. Der Kernel K wurde als die Menge definiert, die alle Komponenten des Lösungsvektors x enthält, die mit Knoten in V korrespondieren und ungleich null sind (vgl. Beweis des Satzes 3.10). Das heißt,

$$K = \{v_i \mid i \in [n] \text{ und } x_{m+i} \neq 0\} \subseteq V.$$

Folglich reicht das Gleichungssystem $\hat{B}\hat{x} = \mathbf{1}$ für das Finden des Kernels aus. Dieses Gleichungsystem besteht aus höchstens n Gleichungen und n Unbekannten, ist also quadratisch in n. □

Bestimmen der Ordnung $>_i$

Bisher wurde nur gezeigt, dass in jeder maximalen Clique eine Ordnung $>_i$ existiert, aber nicht wie diese bestimmt werden kann. Dieser Algorithmus ist aber für die Betrachtung der PPAD-Zugehörigkeit von KERNEL notwendig. Ein solcher wird jetzt entwickelt und gezeigt, dass er polynomial in der Anzahl der Knoten ist.
Gegeben sei eine Clique C mit der Mächtigkeit k, die sich im Digraphen D befindet. Diese Clique hat mindestens $\frac{k(k-1)}{2}$ Kanten, weil sie vollständig ist, und maximal $k(k-1)$ Kanten, weil Paare entgegengesetzt gerichteter Kanten erlaubt sind. Wenn zwei Knoten durch ein Paar entgegengesetzt gerichteter Kanten verbunden ist, dann geht daraus nicht hervor, welcher Knoten in der Ordnung vor dem anderen steht. Dies geht entweder aus dem Zusammenhang mit den anderen Kanten hervor oder aber die Wahlmöglichkeit bleibt bis zum Ende bestehen. Das heißt, es kann frei entschieden werden, welcher der beiden Knoten in der Ordnung vor dem anderen steht. Deshalb sind die Paare entgegengesetzter Kanten kein bestimmendes Element bei der Bildung der Ordnung, und können vernachlässigt werden. Also sind nur die „einfachen" Kanten zu betrachten. Das sind höchstens $n(n-1)/2$ viele. Gegeben sei eine Liste L, die alle einfachen Kanten enthält. Diese wird mit einem Algorithmus abgearbeitet, der im Prinzip wie Bubblesort funktioniert. Die einfachen Kanten können einen zusammenhängenden Digraphen bilden. In diesem Fall ist für alle beteiligten Knoten klar, auf welchem Platz sie stehen, weil sie über mindestens einen Knoten mit allen anderen Knoten in Beziehung stehen. Hierbei können sich unter Umständen auch zwei Knoten einen Platz teilen, wie weiter unten gezeigt wird. Wenn der Digraph, der aus den einfachen Kanten besteht, nicht zusammenhängend ist, dann können die „Blöcke" mit den geordneten Knoten der verschiedenen Zusammenhangskomponenten beliebig angeordnet werden, weil diese Komponenten nur durch Paare entgegengesetzt gerichteter Kanten miteinander verbunden sind.
Die Ordnung $>_i$ der Knoten wird sukzessive aufgebaut. Gegeben sei die Liste L. Als Erstes werden die zwei Knoten der ersten Kante aufgenommen, anschließend alle Knoten eingeordnet, die mit den bereits vorhandenen Knoten adjazent sind. Wenn es keine adjazenten Knoten mehr gibt und die Liste noch nicht leer ist, werden die Knoten der erste Kante aus der aktuellen Liste eingeordnet, anschließend wird wieder nach adjazenten Knoten gesucht usw. Dieser Teil des Algorithmus stoppt, wenn die Liste leer ist. Wenn in der geordneten Menge Knoten existieren, die sich einen Platz teilen, dann werden diese Stapel so aufgelöst, dass ein Knoten diesen Platz behält und alle anderen benachbarte Plätze erhalten (die übrigen Knoten rutschen entsprechend weiter). Jetzt wird überprüft, ob es Knoten aus V gibt, die noch nicht in der Ordnung vorkommen. Das sind Knoten, die mit allen anderen Knoten durch Paare entgegengesetzt gerichteter Kanten verbunden sind. Diese können beliebig in die Menge geordneter Knoten eingefügt werden. Diese Ordnung der Knoten der Clique C wird ausgegeben. Für die Ordnung der erweiterten Clique $C_i' = C_i \cup \{z_i\}$ ist nur der Zusatzknoten z_i zu ergänzen. Dieser muss bekanntlich der kleinste Knoten in der Ordnung sein.
Bei der Untersuchung des aktuellen Listeneintrags können verschiedene Fälle eintreten. Diese werden mit einem möglichen Verlauf der ersten drei Schritte sowie einer weiteren Variante exemplarisch dargestellt.
Es sei $w_1 = (v, w)$ die Kante auf dem ersten Listenplatz. Es muss also $v >_i w$ gelten. Es wird deshalb die absteigend sortierte Menge $O = [v, w]$[1] gemerkt und anschließend w_1 aus der Liste entfernt.
Es sei w_2 das nächste Listenelement. Wenn dieses inzident mit einem Knoten aus O ist, beispielsweise $w_2 = (w, q)$, dann wird es aus der Liste L entfernt und q in der Menge O hinter w ergänzt, weil $w >_i q$ erfüllt sein muss. Es ist dann $O = [v, w, q]$. Wenn $w_2 = (p, q)$ mit keinem Knoten aus O inzident ist, bleibt alles so, wie es ist, und es wird das nächste Listenelement betrachtet. Der Fall $w_2 = (w, v)$ ist nicht möglich,

[1] Für eine bessere Unterscheidung von der Kante (v, w) werden eckige Klammern verwendet

weil Paare entgegengesetzt gerichteter Kanten ausgeschlossen wurden.
Es seien $O = [v, w, q]$ und w_3 das dritte Listenelement. Dieses Mal werden alle möglichen Fälle betrachtet:

(a) Die Kante w_3 ist mit keinem Knoten aus O inzident. Dann wird zum nächsten Listenelement weiter gegangen.

(b) Die Kante w_3 ist mit genau einem Knoten aus O inzident. Dann können prinzipiell folgende Fälle eintreten:

Kante w_3	O
(r, v)	$[r, v, w, q]$
(q, r)	$[v, w, q, r]$
(v, r)	$[v, \{w, r\}, q]$

,

wobei $\{w, r\}$ bedeutet, dass sich w und r auf derselben Position befinden, sie werden sozusagen gestapelt. Wichtig ist in einem Fall wie (v, r), dass sich der neue Knoten immer einen Platz mit dem direkt benachbarten Knoten teilt. Das heißt $[v, w, \{q, r\}]$ wäre in diesem Fall nicht zulässig. Die verbleibenden Fälle folgen analog.

(c) Die Kante ist mit zwei Knoten aus O inzident. Da nur einfache Kanten betrachtet werden, sind nur zwei Fälle möglich. Im Fall $w_3 = (q, v)$ bricht der Algorithmus ab, weil q und v nicht direkt nebeneinander stehen und deshalb nicht vertauscht werden können. Es handelt sich dann nämlich um einen echten Kreis. Im Fall $w_3 = (v, q)$ ändert sich die Menge O nicht, weil bereits beide Knoten in der korrekten Reihenfolge vorhanden sind.

Jetzt bleibt nur noch ein Fall zu betrachten, der eintreten kann, bevor zur Formulierung des Algorithmus übergegangen werden kann. Es seien $O = [v, \{w, r, z\}, q]$ und (w, r) das aktuelle Listenelement. Dann wird der Stapel $\{w, r, z\}$ so aufgelöst, dass $O = [v, \{w, z\}, r, q]$ ist. Der Fall (r, w) folgt analog.
Im Folgenden bezeichne $w \cap O$ für eine Kante $w = (u, v)$ den Schnitt der mit w bzw. O korrespondierenden Knotenmengen. Zunächst wird die folgende Routine definiert, mit der die Liste auf der Suche nach adjazenten Knoten einmal durchlaufen wird:

Routine LISTE

For $i = 1$ to $|L|$
{ Nimm $w_i \in L$
If $|w_i \cap O| = 0$ Then tue nichts;
If $|w_i \cap O| = 1$ Then
{Füge den neuen Knoten entweder direkt vor dem bereits enthaltenen Knoten ein oder direkt danach (wenn diese Positionen noch leer sind) bzw. stapel sie auf dem direkten Nachbarn davor oder danach und entferne w_i aus L;
}
If $|w_i \cap O| = 2$ Then
{Lass alles so, wie es ist, oder vertausche die beiden Knoten, falls sie direkte Nachbarn sind, oder weise zwei gestapelten Knoten benachbarte Plätze zu und entferne w_i aus L. Brich ab, wenn ein Tausch erforderlich aber nicht möglich ist;
}
}

Es sei $w = (u, v)$. Es bezeichne $O + w$ die Verlängerung von O um u, v. Das heißt diese zwei Knoten werden (in dieser Reihenfolge) hinten dran gehängt. Das darf nur geschehen, wenn $O \cap w = \emptyset$ ist.

Algorithmus Ordnung

Eingabe: Liste L;
$O = \emptyset$;
Nimm w_1 aus L;
$O = O + w_1$;
Entferne w_1 aus L;
While L sich verändert
    {Repeat Routine LISTE;
    }
If $L = \emptyset$ Then
    {Noch bestehende Stapel auflösen, indem diese Knoten nebeneinanderliegende Plätze erhalten;
    Wenn O noch nicht alle Knoten aus V enthält, dann die noch fehlenden Knoten (das sind die Knoten, die mit den entgegengesetzt gerichteten Kanten zusammenhängen) irgendwo einfügen;
    Ausgabe O (Knoten sind absteigend sortiert);
    }
Else
{Nimm w_1 aus L;
$O = O + w_1$;
Entferne w_1 aus L;
Gehe zum Beginn der While-Schleife;
}

Welche Laufzeit hat dieser Algorithmus? Die Liste L enthält höchstens $k(k-1)/2$ Kanten, ist also quadratisch in k. Bubblesort ist quadratisch in der Länge der Eingabe, in diesem Fall also $\mathcal{O}(k^4)$. Bei jedem Durchlauf der for-Schleife in der Routine LISTE werden $w \cap O$ gebildet und gegebenenfalls die Plätze von u und v bestimmt. Das ist im allerschlimmsten Fall auch quadratisch in k. Also ist dieser Algorithmus insgesamt in $\mathcal{O}(k^6)$. Wenn n die Anzahl aller Knoten im Digraphen ist, dann muss für jede Clique $k \leq n$ gelten. Also ist der ganze Algorithmus in $\mathcal{O}(n^6)$. Das ist eine ziemlich grobe Abschätzung. Für die Polynomialität der Laufzeit reicht sie aber aus.

Damit ist der Beweis der PPAD-Zugehörigkeit von KERNEL abgeschlossen. □

5.3 Ein Beispiel für den veränderten Algorithmus zur Kernelberechnung

Die Veränderungen an dem Algorithmus zur Kernelberechnung sollen an einem Beispiel verdeutlicht werden. Dafür wird auf das Beispiel im Abschnitt 3.4 zurückgegriffen. Es soll wieder mit derselben Clique gestartet werden. Deshalb werden die Knoten so umnummeriert, dass die lexikographisch größte Clique mit der ursprünglichen Startclique identisch ist (bis auf die Indizes). Der Algorithmus verläuft trotz identischer Startclique anders als beim ursprünglichen Beispiel und endet bereits nach fünf Schritten. Die Wahl des Pivotelements ist ja auch eine andere. Gegeben sei der folgende Digraph:

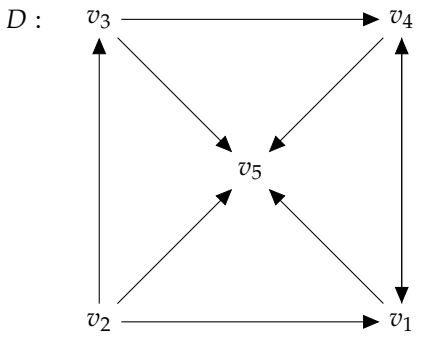


Abb. 5.2: Der Digraph D wie in Beispiel 3.4 nach der Knotenumnummerierung

Bemerkung 5.17. Beim Durchlaufen des Algorithmus wird bei jedem Schritt die gemerkte Teilmatrix $\hat{C}$ hingeschrieben. Für die Platzersparnis (und damit Übersichtlichkeit), und weil die darin enthaltenen Informationen erst bei der Berechnung des Lösungsvektors am Ende des Algorithmus benötigt werden, werden für die Teilmatrix $\hat{B}$ der Cliquenmatrix B nur die darin enthaltenen Zeilen vermerkt. Die Matrix $\hat{B}$ wird nicht ausgeschrieben.

Nun wird der veränderte Algorithmus, der mit polynomialem Speicherplatz für die Daten sowie polynomialer Laufzeit für die einzelnen Schritte auskommt, durchlaufen:

(1) *Bestimmen der Startmengen:* Als Erstes wird die lexikographisch größte Clique $C_1 = \{v_3, v_4, v_5\}$ in dem Digraphen gesucht. Für diese wird die Ordnung $>_1$ bestimmt und die entsprechende Zeile C_1 in der Matrix $\hat{C}$ erstellt. (Um Entartung zu vermeiden wird wieder die Konvention verwendet, dass in der Spalte v_i die Zahl M_i steht, falls der Knoten v_i nicht in der Clique enthalten ist. Für die M_i gelte wieder $M_1 > \ldots > M_5 > 5$.):

$$\hat{C} = \begin{array}{c} \\ C_1 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} M_1 & M_2 & 3 & 2 & 1 \end{pmatrix} \end{array}$$

Es wird der Eintrag $\max\{c_{1v_k} \mid k \in [5]\} = M_1 = c_{1v_1}$ bestimmt. Also wird der Knoten v_1 in die ordinale Basis aufgenommen. Dieser stellt den einzigen Zeilenminimierer (ZM) $u_1 = c_{1v_1}$ ungleich Null. Die Startmengen sind also:

Knoten	v_1	v_2	v_3	v_4	v_5	Zeilen der Teilmatrix
ZM	C_1	-	-	-	-	$\hat{C} : C_1$
Pivot	-	-	-	-	-	$\hat{B} : -$

Der Tabelle kann entnommen werden, dass in der Spalte v_1 der Zeilenminimierer liegt, der sich in der Zeile C_1 befindet, d.h. $u_1 = c_{1v_1}$. Also enthält die ordinale Basis den Knoten v_1 sowie alle Zusatzknoten ungleich z_1. Zu keinem Knoten v_i wurde eine zugehörige Pivotzeile gespeichert. Also enthält die zulässige Basis $\mathcal{B}_z$ alle Zusatzknoten und keine weiteren Knoten. Für die Matrix C wird die Teilmatrix $\hat{C}$ gemerkt, die aus der Zeile C_1 besteht. Die für B gemerkte Teilmatrix $\hat{B}$ ist leer. Der Zeilenminimierer u_1 ist gleich $c_{1v_1} = M_1$, alle anderen Zeilenminimierer sind gleich Null.

(2) *Zulässiger Pivot:* Der Knoten v_1 soll in die zulässige Basis pivotiert werden. Hierfür wird die kleinste Clique von v_1 berechnet. Das ist die Menge $\{v_1, v_2, v_5\}$. Diese wurde bisher nicht verwendet. Also wird sie mit C_2 bezeichnet, ihr charakteristischer Vektor wird als Zeile C_2 in $\hat{B}$ ergänzt, und der Zusatzknoten z_2 wird aus $\mathcal{B}_z$ entfernt.
Ordinaler Pivot: Mit einem ordinalen Pivotschritt wird z_2 aus der ordinalen Basis entfernt. Hierfür werden zunächst die Ordnung $>_2$ bestimmt und die Zeile C_2 in der Matrix $\hat{C}$ ergänzt:

$$\hat{C} = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} M_1 & M_2 & 3 & 2 & 1 \\ 2 & 3 & M_3 & M_4 & 1 \end{pmatrix} \end{array}$$

Die Spalte z_2 hat bisher den Zeilenminimierer u_2 enthalten. Somit ist $u_2' = \min\{c_{2k} \mid k \in \mathcal{B}_o - z_2\} = c_{2v_1} = 2$ der neue Zeilenminimierer. Die Spalte v_1 stellt also zwei Zeilenminimierer. Der alte ist $u_1' = c_{1v_1} = M_1$. Nun wird die Menge $\hat{\mathcal{M}} = \{v_k \mid k \in [5], c_{2v_k} > 2\} = \{v_2, v_3, v_4\}$ bestimmt ($\mathcal{M} = \hat{\mathcal{M}} \cup \{z_1\}$ wäre die vollständige Menge, aber z_1 wird nur im Hinterkopf behalten, falls $\hat{\mathcal{M}}$ leer sein sollte). Mit dieser wird $\max\{c_{1k} \mid k \in \hat{\mathcal{M}}\} = c_{1v_2} = M_2$ gefunden. Folglich wird v_2 in die ordinale Basis aufgenommen.
Gespeicherte Daten: Nach diesen beiden Pivotschritten hat die Tabelle die folgende Gestalt:

Knoten	v_1	v_2	v_3	v_4	v_5	Zeilen der Teilmatrix
ZM	C_2	C_1	-	-	-	$\hat{C} : C_1, C_2$
Pivot	C_2	-	-	-	-	$\hat{B} : C_2$

(3) *Zulässiger Pivot:* Der Knoten v_2 soll in die zulässige Basis aufgenommen werden. Die lexikographisch kleinste Clique, die v_2 enthält, ist die Menge $\{v_1, v_2, v_5\}$. Das ist gerade die Clique C_2. Diese korrespondiert bisher mit dem Knoten $v_1 \in \mathcal{B}_z$. Also wird v_1 aus der zulässigen Basis entfernt. Die Matrix $\hat{B}$ verändert sich nicht.
Ordinaler Pivot: Die Spalte v_1 wird aus der ordinalen Basis entfernt. Da dies kein Zusatzknoten ist, verändert sich die Teilmatrix $\hat{C}$ nicht. Die Spalte v_1 hat bisher den Zeilenminimierer u_2 gestellt. Also ist der neue Zeilenminimierer $u_2' = \min\{c_{2k} \mid k \in \mathcal{B}_o - v_1\} = 3 = c_{2v_2}$, und die Spalte v_2 ist die Spalte mit zwei Zeilenminimierern. Der alte ist $u_1' = c_{1v_2} = M_2$. Es werden $\hat{\mathcal{M}} = \{v_k \mid k \in [5], c_{2v_k} > 3\} = \{v_3, v_4\}$ und anschließend $\max\{c_{1k} \mid k \in \hat{\mathcal{M}}\} = 3 = c_{1v_3}$ bestimmt. Also wird v_3 in die ordinale Basis aufgenommen.
Gespeicherte Daten: Die neuen Mengen sind:

Knoten	v_1	v_2	v_3	v_4	v_5	Zeilen der Teilmatrix
ZM	-	C_2	C_1	-	-	$\hat{C} : C_1, C_2$
Pivot	-	C_2	-	-	-	$\hat{B} : C_2$

(4) *Zulässiger Pivot:* Die Spalte v_3 wird in die zulässige Basis aufgenommen. Dafür wird die lexikographisch kleinste maximale Clique, die v_3 enthält, berechnet. Das ist die Clique $C_3 = \{v_2, v_3, v_5\}$. Diese ist neu. Also werden die Zeile C_3 in $\hat{B}$ ergänzt und z_3 aus $\mathcal{B}_z$ entfernt.
Ordinaler Pivot: Nun wird der Knoten z_3 aus der ordinalen Basis entfernt. Dieser stellt bisher den Zeilenminimierer $u_3 = 0$. Die Zeile C_3 wird in der Matrix $\hat{C}$ ergänzt:

$$\hat{C} = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} M_1 & M_2 & 3 & 2 & 1 \\ 2 & 3 & M_3 & M_4 & 1 \\ M_1 & 3 & 2 & M_4 & 1 \end{pmatrix} \end{array}$$

Der neue Zeilenminimierer ist $u_3' = \min\{c_{3k} \mid k \in \mathcal{B}_o - z_3\} = 2 = c_{4v_3}$. Also enthält die Spalte v_3 jetzt zwei Zeilenminimierer. Der alte ist $u_1' = c_{1v_3} = 3$. Nun wird wieder die Menge $\hat{\mathcal{M}} = \{v_k \mid c_{2v_k} > 3, c_{3v_k} > 2\} = \{v_4\}$ gebildet. Dann ist offensichtlich $\max\{c_{1k} \mid k \in \hat{\mathcal{M}}\} = c_{1v_4} = 2$. Und v_4 wird in $\mathcal{B}_o$ aufgenommen.
Gespeicherte Daten:

Knoten	v_1	v_2	v_3	v_4	v_5	Zeilen der Teilmatrix
ZM	-	C_2	C_3	C_1	-	$\hat{C} : C_1, C_2, C_3$
Pivot	-	C_2	C_3	-	-	$\hat{B} : C_2, C_3$

(5) *Zulässiger Pivot:* Die Spalte v_4 wird in die zulässige Basis aufgenommen. Die zugehörige lexikographisch kleinste Clique ist die Menge $\{v_1, v_4, v_5\}$. Diese Clique ist neu. Also werden der charakteristische Vektor von $C_4 = \{v_1, v_4, v_5\}$in $\hat{B}$ ergänzt und z_4 aus der zulässigen Basis $\mathcal{B}_z$ entfernt.
Ordinaler Pivot: Nun wird z_4 aus der ordinalen Basis entfernt. Hierfür wird zunächst die Zeile C_4 in der Matrix $\hat{C}$ ergänzt. Für diese ist die Ordnung $>_4$ nicht eindeutig. Es wird die folgende gewählt:

$$\hat{C} = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} M_1 & M_2 & 3 & 2 & 1 \\ 2 & 3 & M_3 & M_4 & 1 \\ M_1 & 3 & 2 & M_4 & 1 \\ 2 & M_2 & M_3 & 3 & 1) \end{pmatrix} \end{array}$$

Diese Spalte z_4 hat bisher den Zeilenminimierer $u_4 = 0$ gestellt. Also ist $u_4' = \min\{c_{4k} \mid k \in \mathcal{B}_o - z_4\} = c_{4v_4} = 3$ der neue Zeilenminimierer, und die Spalte v_4 enthält jetzt zwei Zeilenminimierer. Der alte ist $u_1' = c_{1v_4} = 2$. Nun wird die Menge $\hat{\mathcal{M}} = \{v_k \mid c_{2v_k} > 3, c_{3v_k} > 2, c_{4v_k} > 3\} = \emptyset$ gebildet. Dieses Mal greift das Wissen, dass z_1 diese Kriterien erfüllt und in $\mathcal{M}$ ist. Folglich ist $c_{1z_1} = 0$ der neue Zeilenminimierer u_1, und z_1 wird in die ordinale Basis aufgenommen. Deshalb wird die zugehörige Zeile C_1 wieder aus der Matrix $\hat{C}$ entfernt (es werden ja nur die Cliquen gemerkt, die mit Knoten $v_k \in \mathcal{B}_o \cap V$ zusammenhängen). Jetzt terminiert der Algorithmus, weil die beiden Basen übereinstimmen.
Gespeicherte Daten:

Knoten	v_1	v_2	v_3	v_4	v_5	Zeilen der Teilmatrix
ZM	-	C_2	C_3	C_4	-	$\hat{C} : C_2, C_3, C_4$
Pivot	-	C_2	C_3	C_4	-	$\hat{B} : C_2, C_3, C_4$

Bei diesem Beispiel stimmt in jeder Spalte der Index des Zeilenminimierers mit der Pivotzeile überein. Dies muss nicht immer so sein, es ist auch eine „Durchmischung“ möglich .
Jetzt ist noch das Gleichungssystem $\hat{B}x = \mathbf{1}$ zu lösen, wobei $x_1 = x_5 = 0$ sind, da v_1 und v_5 nicht in den Basen enthalten sind, das heißt, $x^\top = (0, x_2, x_3, x_4, 0)$. Und die charakteristischen Vektoren der Cliquen sind die Zeilen von $\hat{B}$. Das Tableau hat also die Gestalt

$$\begin{array}{c} \\ C_2 \\ C_3 \\ C_4 \end{array} \begin{array}{c} \begin{array}{ccccc|c} v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{ccccc||c} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{array}\right) \end{array}$$

und liefert den Lösungsvektor $x^\top = (0, 1, 0, 1, 0)$. Dessen positive Komponenten bilden den Kernel $K = \{v_2, v_4\}$. (Wenn die zu Beginn vorgenommene Knotenumnummerierung zurückgenommen wird, dann stimmt dieser Kernel mit dem Kernel überein, der mit dem ursprünglichen Algorithmus zur Kernelberechnung gefunden wurde. Das muss aber nicht immer so sein. Der Digraph könnte ja mehrere Kernel haben.)

Bemerkung 5.18. An den aktuellen vollständigen Tableaus des Gleichungssystems $Bx = \mathbf{1}$ kann veranschaulicht werden, dass das Pivotelement tatsächlich nicht von den vorhergehenden Schritten berührt wurde, und dass durch die vorgenommene Perturbation des Vektors b die Einträge auf der rechten Seite auch negativ werden können. Allerdings sind sie nie gleich null. Wenn die Perturbation zurückgenommen wird, dann steht auf der rechten Seite ein Vektor aus $\{0,1\}^4$. Das muss auch so sein, weil nach dem Satz von Chvátal alle Lösungen in der Menge $\{0,1\}^4$ liegen müssen. Stellvertretend werden die Tableaus der ersten drei Schritte angegeben, wobei das Pivotelement fett geschrieben wurde:

(1)
$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc|c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 1 & 1+\epsilon_2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_3 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1+\epsilon_4 \end{array}\right) \end{array}$$

(2)
$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc|c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 1 & 1+\epsilon_2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1+\epsilon_3 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & \epsilon_4-\epsilon_2 \end{array}\right) \end{array}$$

(3)
$$\begin{array}{c} \\ C_1' \\ C_2' \\ C_3' \\ C_4' \end{array} \begin{array}{c} \begin{array}{cccc|ccccc|c} z_1 & z_2 & z_3 & z_4 & v_1 & v_2 & v_3 & v_4 & v_5 & b \end{array} \\ \left(\begin{array}{cccc|ccccc||c} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1+\epsilon_1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1+\epsilon_2 \\ 0 & -1 & 1 & 0 & -1 & 0 & \mathbf{1} & 0 & 0 & \epsilon_3-\epsilon_2 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1+\epsilon_4 \end{array}\right) \end{array}$$

5.4 Verhalten des Algorithmus bei Eingabe eines nicht clique-azyklischen Digraphen

Die Perfektheit des Graphen kann in Polynomialzeit getestet werden [9]. Anders verhält es sich mit der Überprüfung, ob ein Digraph clique-azyklisch ist. Denn sowohl die Anzahl der Cliquen als auch die Anzahl der echten Kreise in einem einfachen Digraphen können exponentiell sein [23]. Was geschieht, wenn in den Algorithmus zur Kernelberechnung ein Digraph eingeben wird, der nicht clique-azyklisch ist? Der Algorithmus bricht ab, sobald die Clique angesprochen wird, die einen echten Kreis enthält, weil es nicht möglich ist, die Ordnung $>_i$ zu bestimmen. Wenn diese Clique nicht angesprochen wird, dann läuft der Algorithmus durch. Das folgende Beispiel zeigt, dass dieser Fall tatsächlich eintreten kann. Hierbei wird benutzt, dass die Matrizen $\hat{B}$ und $\hat{C}$ sukzessive aufgebaut werden. Andernfalls würde der Algorithmus bereits beim Erstellen der Matrix C abbrechen.

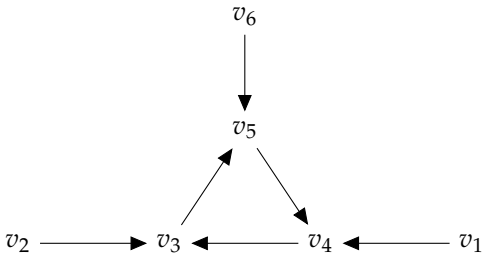


Abb. 5.3: Beispiel für einen nicht clique-azyklischen Digraphen, für den der Algorithmus einen Kernel berechnen kann

Beispiel 5.19. In Abbildung 5.3 ist der gegebene Digraph dargestellt, der eine Clique besitzt, die einen echten Kreis enthält, also nicht clique-azyklisch ist. Der Algorithmus, welcher mit der Clique $C_1 = \{v_1, v_4\}$ startet, nimmt den folgenden Verlauf, ohne dabei die Clique C_4, die den Kreis enthält, einzubeziehen:

Schritt	Pivotzeile	$\mathcal{B}_z$	$\mathcal{B}_o$	Zeilenminimierer u
Start			v_1	$(M_1, 0, 0, 0)$
(1)	C_2	v_1	v_1, v_2	$(M_2, 2, 0, 0)$
(2)	C_3	v_1, v_2	v_1, v_2, v_6	$(2, 2, 2, 0)$
(3)	C_1	v_1, v_2, v_6		

Am Ende ist die vollständige Basis $\mathcal{B} = \{v_1, v_2, v_6, z_4\}$, und die Teilmatrizen haben die folgende Gestalt, wobei $M_1 > \ldots > M_6 > 6$ gewählt wurden:

$$\hat{B} = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \hat{C} = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} M_1 & M_2 & M_3 & M_4 & 1 & 2 \\ 2 & M_2 & M_3 & 1 & M_5 & M_6 \\ M_1 & 2 & 1 & M_4 & M_5 & M_6 \end{pmatrix} \end{array}$$

Ausgegeben wird die Menge $K = \{v_1, v_2, v_6\}$, die ein Kernel ist, obwohl der Digraph nicht clique-azyklisch ist.

Es sind noch zwei Dinge zu klären: Unter welchen Umständen ist es möglich, dass der Algorithmus nicht abbricht, obwohl der eingegebene Digraph nicht clique-azyklisch ist? Ist bei Nicht-Abbruch die Ausgabe

ein Kernel?
Gegeben sei ein Digraph D, der nicht clique-azyklisch ist. Es bezeichne C'_j eine Clique, die einen echten Kreis enthält. Zunächst wird angenommen, dass in der Clique C'_j ein Knoten y existiert, der in keiner weiteren Clique enthalten ist. In diesem Fall ist $c_{iy} > n$ für alle $i \neq j$ (beispielsweise könnten diese Einträge alle gleich M_y sein). Im Abschnitt 3.5 wurde bewiesen, dass für alle Zeilenminimierer $u_i \neq u_1$ immer $u_i < n$ gilt. Also ist bei der Durchführung des ordinalen Pivotschritts y immer in der Menge $\mathcal{M} = \{k \mid c_{ik} > u_i \; \forall \, i \in [m] \setminus i_a\}$ enthalten, solange die Zeile C'_j nicht berührt wurde. Das heißt wiederum, dass die Spalte y bei der Bestimmung der neu aufzunehmenden Spalte durch $\max\{c_{i_a k} \mid k \in \mathcal{M}\}$ zu beachten ist. Insbesondere gilt das, wenn der Index des alten Zeilenminimierers $i_a = 1$ ist. Das heißt, solange die Zeile C'_j nicht angesprochen wurde, muss $u_1 \geq c_{1y} > n$ sein und kann der Algorithmus nicht terminieren (vgl. Abschnitt 3.5). Ergo muss diese Zeile irgendwann die Pivotzeile sein, und der Algorithmus bricht beim anschließenden ordinalen Pivotschritt ab, weil die Ordnung $>_j$ nicht bestimmt werden kann. Hiermit wurde gezeigt, dass es nur dann möglich ist, dass der Algorithmus nicht abbricht, wenn jeder Knoten der Clique, die einen echten Kreis enthält, in mindestens einer weiteren Clique enthalten ist. In diesem Fall ist die ausgegebene Menge K unabhängig, weil die Unabhängigkeit von K nur von dem Gleichungssystems $Bx = b$ abhängt (Abschn. 3.5), welches autonom von der Orientierung der Kanten ist. Die Basis des Lösungsvektors ist $\mathcal{B} = \mathcal{B}_z = \mathcal{B}_o$, wobei $\mathcal{B}_o$ eine ordinale Basis ist. Damit lässt sich identisch zum letzten Schritt des Beweises von Satz 3.10 zeigen, dass K auch dominierend ist.

5.5 Versuch, analog zu STRONG KERNEL zu zeigen, dass KERNEL PPAD-schwer ist

Es wurde bereits gezeigt, dass KERNEL in PPAD liegt. Wenn noch nachgewiesen werden kann, dass KERNEL PPAD-schwer ist, dann wäre KERNEL PPAD-vollständig. Hierfür wird der folgende Ausschnitt aus der Reduktionskette von Kintali herangezogen:

$$\text{PREFERENCE GAME} \leq_p \text{DEGREE 3 PREFERENCE GAME} \leq_p \text{STRONG KERNEL}.$$

Im dritten Kapitel wurde gezeigt, dass die Probleme PREFERENCE GAME und CONSTANT DEGREE PREFERENCE GAME PPAD-vollständig sind, und dass das Problem STRONG KERNEL eine Verallgemeinerung von KERNEL ist (Lemma 4.19). Der algorithmische Beweis, dass jede clique-azyklische Orientierung eines Digraphen einen stark fraktionalen Kernel besitzt, ist im Prinzip derselbe wie der für Kernel, nur dass in den Bedingungen auf die Perfektheit des unterliegenden Graphen verzichtet wurde. Deshalb liegt es nahe, zu versuchen den Beweis für die PPAD-Schwere des Problems STRONG KERNEL auf KERNEL zu übertragen.
Kintali u.a. haben die Reduktion von allgemeinen Präferenzspielen auf Präferenzspiele vom Grad 3 vorgenommen, weil so sicher gestellt ist, dass in der anschließenden Reduktion auf STRONG KERNEL die Bedingung an die Cliquengröße erfüllt wird. Diese sorgt dafür, dass nur polynomial viele maximale Cliquen auftreten. Deshalb sind die Daten und die einzelnen Schritte in Scarf polynomial in n, und das Problem STRONG KERNEL liegt in PPAD. Für den Beweis, dass das Problem KERNEL in PPAD liegt, war eine solche Einschränkung der Anzahl der Cliquen nicht nötig. Deshalb kann auf die erste Reduktion auf Präferenzspiele vom Grad 3 verzichtet werden. Das heißt, es wird versucht, den Beweis des Satzes 4.22 auf PREFERENCE GAME $\leq_p$ KERNEL zu übertragen. Das ist einfach möglich, da dem Grad des Präferenzspiels in dem Beweis keine Funktion zukommt.

Gegeben sei ein Präferenzspiel mit $[n]$ Spielern. Zu diesem wird ein Digraph $D = (V, A)$ konstruiert, indem für jeden Spieler i ein Knoten $\langle i, i \rangle$ und für jeden Spieler $j \in out(i)$ ein Knoten $\langle i, j \rangle$ eingeführt wird. Es gibt eine Kante von $\langle i, j \rangle$ nach $\langle i, k \rangle$, wenn j von i lieber gespielt wird als k, d.h. j steht in der Präferenzliste von i vor k. Für jeden Knoten $\langle i, j \rangle$ mit $i \neq j$ gibt es zusätzlichen Knoten $J(i, j)$, der mit genau zwei Knoten verbunden ist, einmal durch eine von $\langle j, j \rangle$ kommende Kante, sowie durch eine nach $\langle i, j \rangle$ führende Kante. Dieser Digraph ist natürlich wieder clique-azyklisch (vgl. Beweis zu 4.22), und da die charakteristische Funktion eines Kernels ein stark fraktionaler Kernel ist (Lemma 4.19), gibt es es eine Abbildung, die einen Kernel in Polynomialzeit auf ein Gleichgewicht des Präferenzspiels abbildet. Jetzt bleibt nur nur noch zu zeigen, dass der unterliegende Graph perfekt ist, und der Beweis ist fertig. Dies ist leider nicht immer der Fall, wie das folgende Gegenbeispiel zeigt:

Beispiel 5.20. Gegeben sei das folgende Präferenzspiel mit fünf Spielern:

Spieler i	Präferenzliste	$out(i)$	$in(i)$	Grad
1	(3,4,1)	3,4	2	3
2	(5,1,2)	5,1	3	3
3	(2,4,3)	2,4	1	3
4	(5,4)	5	1,3	3
5	(4,5)	4	2,4	3

Zu diesem Präferenzspiel wird wie im Beweis der Digraph konstruiert, welcher in Abbildung 5.4 dargestellt ist. Dort ist zu sehen, dass dieser Digraph zwei ungerade Löcher größer als drei besitzt. Das eine besteht aus neun Knoten, das andere aus elf. Deshalb ist der unterliegende Graph nach dem starken Satz über perfekte Graphen 2.11 nicht perfekt.
Für den Nachweis, dass ein Präferenzspiel zu einem nicht perfekten Graphen führen kann, hätte das durch die Spieler 1,2 und 3 definierte Spiel ausgereicht. Es sollte an diesem Beispiel aber gleichzeitig gezeigt werden, dass es keine Rolle spielt, ob die Reduktion mit PREFERENCE GAME oder DEGREE 3 PREFERENCE GAME wie in Satz 4.16 begonnen wird. Deshalb wurde das Spiel um die Spieler 4 und 5 so erweitert, dass jeder Spieler den Grad drei hat, und sowohl der Ausgangsgrad als auch der Eingangsgrad eines jeden Spielers höchstens zwei sind.

Es könnte versucht werden, dies irgendwie zu fixen und den Beweis zu retten. Es gibt allerdings noch ein zweites Hindernis. Wie bereits gesagt wurde, ist die charakteristische Funktion eines Kernels ein stark fraktionaler Kernel, der polynomial in ein Gleichgewicht des Präferenzspiels überführt werden kann. Da die charakteristische Funktion nur die Werte 0 und 1 annehmen kann, können auch die dadurch definierten Gewichtsfunktionen nur die Werte 0 und 1 annehmen. Anders formuliert würde damit beweisen, dass jedes Präferenzspiel ein Gleichgewicht hat, das nur aus reinen Strategien besteht. Dies ist offensichtlich nicht der Fall, wie beispielsweise das einfache Gadget $\mathcal{G}_{\frac{1}{2}}$ (Lemma 4.8) zeigt, das einen Spieler erzeugt, der sich selbst in jedem Gleichgewicht mit dem Gewicht $1/2$ spielt. Deshalb ist dieser Ansatz, PREFERENCE GAME auf KERNEL zu reduzieren und so nachzuweisen, dass KERNEL PPAD-schwer ist, ungeeignet. Die Frage, ob KERNEL PPAD-schwer ist, ist also noch offen.
Király und Pap haben einen anderen Beweis für die Kernelberechnung [24] geliefert. Dieser basiert auf dem Lemma von Sperner. Da das Suchproblem SPERNER PPAD-vollständig ist, ist es vielleicht möglich, durch eine Reduktion von SPERNER auf diesen Algorithmus nachzuweisen, dass KERNEL PPAD-schwer ist.

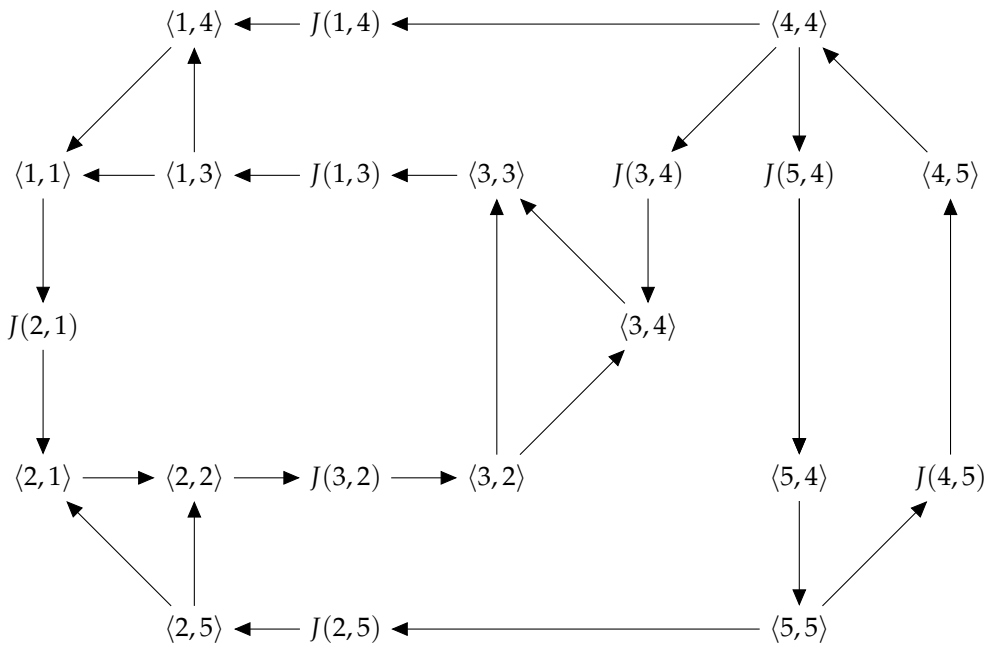


Abb. 5.4: Beispiel, dass ein Präferenzspiel mit einem nicht perfekten Digraphen korrespondieren kann

5.6 Betrachtungen zur Polynomialität von KERNEL und SCARF

Das Problem SCARF ist PPAD-vollständig, also genau so schwer wie die Probleme BROUWER, SPERNER und NASH [13]. Bisher ist von keinem dieser Probleme bekannt, dass es in Polynomialzeit lösbar ist. Es erscheint derzeit unwahrscheinlich, dass das jemals der Fall sein wird. Und falls sich für eins dieser Probleme doch ein polynomialer Algorithmus finden sollte, dann würde dies suggerieren, dass alle anderen ebenfalls polynomial lösbar sind, schließlich sind sie durch Reduktionen miteinander verbunden. All diese Probleme ziehen ihre mögliche exponentielle Gesamtlaufzeit daraus, dass zwar jeder einzelne Schritt in Polynomialzeit ausgeführt werden kann, aber insgesamt exponentiell viel Schritte benötigt werden. Und es lässt sich nicht im Vorhinein abschätzen, wie der Verlauf sein wird.
Das Problem KERNEL verwendet den Algorithmus von Scarf, ist also vermutlich ähnlich schwer. Es sei denn, es ist durch die zusätzlichen Informationen über den Digraphen, die beim normalen Scarf ja nicht zur Verfügung stehen, möglich, einen Verlauf zu definieren, der nach polynomial vielen Schritten endet. Das Beispiel im Abschnitt **??** zeigt, dass es möglich ist, dass ein Knoten, der zuvor aus beiden Basen entfernt wurde, später wieder in diese aufgenommen wird. Der Algorithmus läuft also, salopp ausgedrückt, nicht straight forward durch sondern kann mäandern. Und über dieses mögliche Mäandern liegen keine Informationen vor. Der einzige Fall, in dem die Anzahl der notwendigen Schritte vorhergesagt werden kann, ist der folgende:

Lemma 5.21. KERNEL *endet nach höchstens* $n+1$ *Schritten, wenn bei jedem ordinalen Pivotschritt* u'_1 *der alte Zeilenminimierer ist.*

Beweis. Als Erstes wird gezeigt, dass beim ersten ordinalen Pivotschritt der alte Zeilenminimierer immer in der Zeile C'_1 liegen muss. Der Algorithmus startet mit der zulässigen Basis $\mathcal{B}_z = \{z_1, \ldots, z_m\}$ und der ordinalen Basis $\mathcal{B}_o = \{v_j, z_2, \ldots, z_m\}$, wobei v_j die Spalte ist, in der das Maximum der Menge $\{c_{1k} \mid k \in V\}$ liegt. Also ist $u_1 = c_{1v_j}$ und alle anderen Zeilenminimierer sind gleich Null. Nun wird v_j mit einem zulässigen Pivotschritt in $\mathcal{B}_z$ aufgenommen und dafür ein Zusatzknoten z_i entfernt. Anschließend wird

dieser Zusatzknoten z_i mit einem ordinalen Pivotschritt aus der ordinalen Basis entfernt. Wie bereits im Beweis des Lemmas 5.9 gezeigt wurde, muss u'_i der neue Zeilenminimierer sein, weil die Spalte z_i bisher den Zeilenminimierer u_i gestellt hat. Für die perturbierte Matrix C gelten die folgenden Ungleichungen:

$$0 = c_{iz_i} < c_{iv_k} < c_{iz_j} \text{ für alle } i \neq j \in [m], k \in [n].$$

Deshalb liegt der neue Zeilenminimierer $u'_i = \min\{c_{ik} \mid k \in \mathcal{B}_o - z_i\}$ in der Spalte v_j, weil dies die einzige Spalte aus $V \cap \mathcal{B}_o$ ist. Also enthält die Spalte v_j nun zwei Zeilenminimierer, und u'_1 ist der alte Zeilenminimierer.
Als Zweites wird gezeigt, dass der Algorithmus nach höchstens $n+1$ Schritten (das Bilden der Startmengen als eigenen Schritt gezählt) endet, wenn u'_1 immer der alte Zeilenminimierer ist. Angenommen es wurde ein ordinaler Pivotschritt durchgeführt, dabei die Spalte y aus der ordinalen Basis entfernt, dafür die Spalte r aufgenommen, und u'_1 war der alte Zeilenminimierer (hierbei ist es übrigens egal, ob r und y Zusatzknoten oder Knoten aus V sind). Nach Voraussetzung ist nach dem Durchführen des ordinalen Pivots c_{1r} der neue Zeilenminimierer u_1. (Es wurde vorne schon einmal erläutert, dass sich beim ordinalen Pivotschritt genau zwei Zeilenminimierer ändern. Der neue Zeilenminimierer u'_{i_n} ersetzt u_{i_n}, und der alte $u_{i_a} = u'_{i_a}$ wird durch $c_{i_a r} = \max\{c_{i_a k} \mid k \in \mathcal{M}\}$ ersetzt.) Da die Spalte, die den alten Zeilenminimierer enthält, weiterhin in der ordinalen Basis ist, muss $c_{ir} < u'_1$ gelten. Das heißt der Zeilenminimierer u_1 ändert sich bei jedem ordinalen Pivotschritt und wird kleiner. Bei der Vereinfachung des ordinalen Pivotschritts wurde gezeigt, dass für die Menge $\mathcal{M}$, die für das Finden von $c_{i_a r}$ herangezogen wird, $\mathcal{M} = \{z_{i_a}\} \cup (\mathcal{M} \cap V)$ gilt, und dass der Zusatzknoten erst dann angesprochen wird, wenn die Menge $\mathcal{M} \cap V$ leer ist. Daraus folgt, dass bei jedem ordinalen Pivotschritt nur Knoten angesprochen werden können, die in $\{z_1\} \cup V$ liegen. Diese Menge hat die Kardinalität $n+1$. Da der Zeilenminimierer u_1 bei jedem Schritt kleiner wird, entfallen alle Knoten, die zuvor schon einmal einen Zeilenminimierer in der ersten Zeile gestellt haben. Deshalb muss zwangsweise spätestens beim $(n+1)$-ten Schritt der Zusatzknoten z_1 angesprochen und in die ordinale Basis aufgenommen werden. Bekanntermaßen terminiert der Algorithmus in diesem Fall. □

Lemma 5.22. *Wenn bei jedem Pivotschritt, mit dem ein Knoten w aus $V \cap \mathcal{B}_o$ in die zulässige Basis aufgenommen werden soll, die Pivotzeile C'_i so gewählt werden kann, dass das Minimum der Menge $\{c_{ik} \mid k \in \mathcal{B}_o \cap V\}$ in der aufzunehmenden Spalte liegt, dann ist der Zeilenminimierer u_1 immer in der Spalte $\mathcal{B}_o \setminus \mathcal{B}_z$, beim ordinalen Pivot ist immer u'_1 der alte Zeilenminimierer, und für jeden Knoten $w \in \mathcal{B}_z \cap \mathcal{B}_o \cap V$ liegt der zugehörige Zeilenminimierer in der Pivotzeile, mit der w in die zulässige Basis aufgenommen wurde.*

Beweis. Vorab wird bemerkt, dass es für jeden Zusatzknoten nur eine mögliche Pivotzeile gibt und diese Beziehung zwischen Pivotzeile und Zeilenminimierer immer erfüllt ist.
Es wird angenommen, dass es immer möglich ist, die Pivotzeile so zu wählen. Die Startmengen seien $\mathcal{B}_z = \{z_1, \ldots, z_m\}$, $\mathcal{B}_o = \{y, z_2, \ldots, z_m\}$ und $u = (c_{1y}, c_{2z_2}, \ldots, c_{mz_m})$. Nach Konstruktion liegt die Spalte y immer V und enthält immer den Zeilenminimierer u_1. In die zulässige Basis soll y aufgenommen werden. Dafür wird eine Clique C'_i als Pivotzeile gewählt, für welche $c_{iw} = \min\{c_i k \mid k \in \mathcal{B}_o \cap V\}$ ist. Da w der einzige Knoten im Schnitt $\mathcal{B}_o \cap V$ ist, kann eine beliebige Clique gewählt werden, die w enthält. Ohne Einschränkung sei C'_2 diese Clique. Dann muss der Zusatzknoten z_2 erst aus der zulässigen Basis und anschließend aus der ordinalen Basis entfernt werden. Da die Spalte z_2 den Zeilenminimierer u_2 enthalten hat, ist $u'_2 = \min\{c_{2k} \mid k \in \mathcal{B}_o - z_2\} = c_{2w}$ der neue Zeilenminimierer. Das gilt wegen der Ungleichungen für die Matrix C, und weil w der einzige Knoten in $(\mathcal{B}_o - z_2) \cap V$ ist. Also enthält die Spalte w zwei Zeilenminimierer, den neuen u'_2 und den alten u'_1. Nun wird wie immer durch Bilden der Menge $\mathcal{M}$ und anschließendes Bestimmen von $\min\{c_{1k} \mid k \in \mathcal{M}\} = c_{1s}$ die Spalte s bestimmt, die in die ordinale Basis aufgenommen wird. Nach diesem Schritt sind $\mathcal{B}_z = \{z_1, w, z_3, \ldots, z_m\}$, $\mathcal{B}_o = \{w, s, z_3, \ldots, z_m\}$ und

$u = (c_{1s}, c_{2w}, 0, \ldots, 0\}$. Es ist $\mathcal{B}_z \cap \mathcal{B}_o = \{w\}$, und w korrespondiert mit der Pivotzeile C'_2 und dem Zeilenminimierer u_2. Die Spalte s enthält den Zeilenminimierer u_1 und liegt in $\mathcal{B}_o \setminus \mathcal{B}_z$. Damit ist gezeigt, dass nach dem ersten Pivotschritt alle Bedingungen erfüllt sind.
Nun wird angenommen, dass für jeden Knoten aus $\mathcal{B}_o \cap V$ der zugehörige Zeilenminimierer in der Pivotzeile liegt, die mit diesem Knoten korrespondiert, und dass der Zeilenminimierer u_1 in der Spalte $w = \mathcal{B}_o \cap \mathcal{B}_z$ liegt. Die Spalte w soll in die zulässige Basis aufgenommen werden. (Der Knoten w kann kein Zusatzknoten sein, weil er den Zeilenminimierer u_1 stellt, und das wäre nur für z_1 möglich. In diesem Fall würde bereits $\mathcal{B}_z = \mathcal{B}_o$ gelten.) Es wird eine Clique C'_j, für die $\min\{c_{jk} \mid k \in \mathcal{B}_o \cap V\} = c_{jw}$ gilt, als Pivotzeile gewählt. Der Knoten q, der mit dieser Zeile korrespondiert, wird aus der zulässigen Basis entfernt. Nach Voraussetzung liegt der Zeilenminimierer u_j in der Spalte q. Beim anschließenden ordinalen Pivotschritt wird q aus der ordinalen Basis entfernt, und u'_{j_n} ist der neue Zeilenminimierer. Aufgrund der Wahl der Pivotzeile ist $u'_{j_n} = c_{jw}$. Also hat die Spalte w zwei Zeilenminimierer, und u'_1 ist der alte Zeilenminimierer. Deshalb liegt der Zeilenminimierer der q ersetzenden Spalte r in der Zeile C'_1, das heißt $u_1 = c_{1r}$. Damit wurde gezeigt, dass für den Knoten w, der neu in $\mathcal{B}_z \cap \mathcal{B}_o$ ist, der zugehörige Zeilenminimierer u_j in der mit w korrespondierenden Pivotzeile C'_j liegt, und die Spalte $r = \mathcal{B}_o \setminus \mathcal{B}_z$ den Zeilenminimierer u_1 enthält. □

Bemerkung 5.23. Folgende Fragen sind offen:

- Existiert eine Clique wie in Lemma 5.22 in jedem Fall?
- Wie kann eine solche Clique gefunden werden?
- Könnte bei dieser Wahl der Pivotzeile das Pivotelement im aktuellen Tableau eventuell gleich null sein?

Für das Problem SCARF gilt Lemma 5.21 analog. Das Lemma 5.22 kann nicht übernommen werden, weil keine Informationen über irgendwelche Beziehungen zwischen den Matrizen B und C vorliegen. Das heißt die Beziehungen

$$b_{ij} = 1 \Leftrightarrow c_{ij} \leq n,$$
$$b_{ij} = 0 \Leftrightarrow c_{ij} > n,$$

die im Problem KERNEL für alle $j \in V$ gelten, sind nicht gegeben. Dafür wäre das Bestimmen der Pivotzeile analog zur Clique sehr einfach.

6 Zusammenfassung

Das Ziel dieser Arbeit war, zu untersuchen, ob sich Kintalis [1] Beweis für die PPAD-Vollständigkeit des Problems STRONG KERNEL auf KERNEL übertragen lässt.

Für den Nachweis der PPAD-Zugehörigkeit wurden zunächst die notwendigen Informationen zusammengetragen: Das Problem der Kernelberechnung in einer clique-azyklischen Superorientierung eines perfekten Graphen, ist ein totales Suchproblem. Der Algorithmus entspringt dem Beweis von Aharoni und Holzman [4], mit dem die Kernelexistenz gezeigt wurde. Im Prinzip überträgt dieser die Informationen des Digraphen auf zwei Matrizen, die die Voraussetzungen eines Lemmas von Scarf [2] erfüllen. Aus dem Beweis des Lemmas von Scarf ist ein Algorithmus hervorgegangen. Selbiger wird auf die Matrizen angewandt und liefert zu einem linearen Gleichungssystem die Basis eines Lösungsvektors, welcher berechnet wird und das gewünschte Ergebnis hervorbringt. Die Matrizen können exponentiell in der Eingabelänge sein. Das Problem SCARF, welches mit dem Lemma von Scarf korrespondiert, ist bereits in PPAD enthalten.
Es wurde erarbeitet, dass diese Daten ohne Informationsverlust so heruntergebrochen und verarbeitet werden können, dass der Aufwand polynomial in der Eingabelänge ist. Es wurde ein Algorithmus entwickelt, der zu einer gegebenen maximalen Clique in Polynomialzeit die Ordnung $>_i$ findet, und so die bestehende Lücke im Algorithmus zur Kernelberechnung geschlossen. Elegant ist das Umschiffen der Schwierigkeit, eine lexikographisch größte Clique in Polynomialzeit zu finden, indem eine Knotenumnummerierung vorgenommen wird, was in diesem Fall glücklicherweise zulässig ist.

Kintali et al. [1] haben das Problem PREFERENCE GAME eingeführt und anhand einer Reduktionskette gezeigt, dass selbiges PPAD-vollständig ist. Dieses Problem basiert auf dem Finden eines Gleichgewichts in einem Präferenzspiel. Das ist ein Spiel, in dem jeder Spieler abhängig von seiner Präferenzrelation auf der Menge der Strategien, welche mit der Spielermenge identisch ist, jeder Strategie ein Gewicht zuweist. Diese Gewichtsfunktionen unterliegen gewissen Regeln. Ausgehend von diesem Spiel haben sie gezeigt, dass das Problem STRONG KERNEL PPAD-schwer ist. Es wurde getestet, ob sich dieser Beweis auf KERNEL übertragen lässt, und gezeigt, dass dies nicht möglich ist. Ein alternativer Ansatz wäre, den Beweis von Király und Pap [24], der die Existenz eines Kernels in den gewünschten Digraphen nachweist, heranzuziehen. Dieser Beweis basiert auf dem Lemma von Sperner. Bekanntermaßen ist SPERNER ein PPAD-vollständiges Problem. Vielleicht ist es möglich, dieses Problem auf KERNEL zu reduzieren und so zu zeigen, dass KERNEL PPAD-vollständig ist.

Zusätzlich wurde untersucht, wie sich der Algorithmus zur Kernelberechnung verhält, wenn der eingegebene Digraph perfekt, aber nicht clique-azyklisch, ist. Auch wenn es unwahrscheinlich erscheint, dass KERNEL in Polynomialzeit gelöst werden kann, wurde gezeigt, dass es ein Kriterium gibt, welches garantiert, dass der Algorithmus zur Kernelberechnung in polynomialer Zeit terminiert, wenn es erfüllt ist. Offen ist, ob es erfüllt werden kann.

Literaturverzeichnis

[1] Shiva Kintali, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. Reducibility among fractional stability problems. SIAM J. Comput., 42(6):2063–2113, 2013.

[2] Herbert E. Scarf. The core of an n person game. Econometrica, 35, No. 1:50–69, 1967.

[3] Überblick. Finding kernels in special digraphs, January 2015. http://lemon.cs.elte.hu/egres/open/Finding_kernels_in_special_digraphs.

[4] Ron Aharoni and Ron Holzman. Fractional kernels in digraphs. Journal of Combinatorial Theory, Series B, 73:1–6, 1998.

[5] Claude Berge. Färbung von graphen deren sämtliche bzw. ungerade kreise starr sind (zusammenfassung). Wissenschaftliche Zeitschrift der Martin-Luther-Universität Halle-Wittenberg Mathematisch-Naturwissenschafliche Reihe, pages 114–115, 1961.

[6] László Lovász. Normal hypergraphs and the perfect graph conjecture. Discrete Mathematics 2, pages 253–267, 1972.

[7] László Lovász. A characterization of perfect graphs. Journal of Combinatorial Theory, Series B, 13:95–98, 1972.

[8] Delbert Ray Fulkerson. Anti-blocking polyhedra. Journal of Combinatorial Theory, Series B, 12:50–71, 1972.

[9] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. Annals of Mathematics, 164:51–229, 2006.

[10] Vašek Chvátal. On certain polytopes associated with graphs. Journal of combinatorial theory, Series B, 18:138–154, 1975.

[11] Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. Journal of Research of the National Bureau of Standards, Sect. B, 69B:125–130, 1965.

[12] Christios H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. Journal of Computer and system sciences, 48(3):498–532, 1994.

[13] Constantino Daskalakis. Nash equilibria: Complexity, symmetries and approximation, November 2014. http://people.csail.mit.edu/costis/academic.html.

[14] Constantino Daskalakis. 6.896: Topics in algorithmic game theory, spring 2010, November 2014. http://people.csail.mit.edu/costis/6896sp10/.

[15] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. Elsevier, Theoretical Computer Science, 81:317–324, 1991.

[16] C.E. Lemke and J.T. Howson. Equilibrium point of bimatrix games. Journal of the Society for Industrial and Apllied Mathematics, 12(2):413–423, 1964.

[17] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. SIAM J. Comput., 39(1):195–259, 2009.

[18] Jesper Makholm Nielsen. On the number of maximal independent sets in a graph, February 2015. http://www.brics.dk/RS/02/15/.

[19] Michael J. Todd. Orientation in complementary pivot algorithms. Mathematics of Operations Research, 1(1):54–66, 1976.

[20] George B. Dantzig. Linear Programming and Extensions. Princeton University Press, New Jersey, 1998.

[21] Lloyd S. Shapley. A note on the lemke-howson algorithm. Pivoting and Extension: Mathematical Programming Study I, pages 175–189, 1974.

[22] Panos M. Oardalos and Jue Que. The maximum clique problem. Journal of Global Optimization, 4:301–328, 1994.

[23] Donald B. Johnson. Finding all the elementary circuits of a directed graph. SIAM J. Comput., 4(1):77–84, 1975.

[24] Tamás Király and Júlia Pap. A note on kernels and Sperner's lemma. Discrete Applied Mathematics, 157:3327–3331, 2009.

Abbildungsverzeichnis