Diss. ETH 6620

GANZZAHLIGE POLYMATROID-INTERSEKTIONS-ALGORITHMEN

ABHANDLUNG

zur Erlangung des Titels eines

Doktors der Mathematik

der

EIDGENOESSISCHEN TECHNISCHEN HOCHSCHULE ZUERICH

vorgelegt von

Paul Schönsleben

dipl. Math. ETH

geboren am 12. Mai 1952

von Bronschhofen/SG

Angenommen auf Antrag von

Prof. Dr. E. Specker, Referent

PD Dr. Th.M. Liebling, Korreferent

1980

Integral Polymatroid Intersection Algorithms

Let E be a finite set and let $\mathbb{R}^E$ denote the space of real valued vectors $x = [x_e: e \in E]$. Let $\mathbb{R}^E_+ := \{x \in \mathbb{R}^E: 0 \leq x\}$ and $[0,k]^E := \{x \in \mathbb{R}^E_+: x \leq k \in \mathbb{R}\}$. Let P_1, P_2 be two integral polymatroids in $\mathbb{R}^E$.

Making use of theorems of J.Edmonds and R.Giles we present algorithms relative to a subroutine for recognizing the members of the polymatroids to solve the following integral polymatroid intersection problems:

a) Find an x among all $z \in P_1 \cap P_2$ which has a maximal sum of components (i.e. for which $\sum\{x_e: e \in E\}$ is maximal).

b) To a given vector $c \in \mathbb{R}^E$ find a vector x among all $z \in P_1 \cap P_2$, for which c.x , the weighted sum of components, is maximal.

c) To a given vector $c \in \mathbb{R}^E$ and a number $m \in \mathbb{N}$, either find a vector x among all $z \in P_1 \cap P_2$, $\sum\{z_e: e \in E\} = m$, for which c.x is maximal, or prove that no such vector exists.

Let $n := |E|$, $K := \min\{k: k \in \mathbb{N}, P_1 \cap P_2 \subseteq [0,k]^E\}$, Z_k the complexity of the subroutine of P_k mentioned above, $Z := \max\{Z_1, Z_2\}$.

As a key to the problems we found a so called Transitivity Property of polymatroids. This property helps us to determine augmenting paths. And it is very important in order to estimate the complexity of the algorithms. It turns out that the main dificulty is to compute the number of augmentations of the primal vector in the case of a) to c) and the number of revisions of the dual vector in the case of b) and c). For the algorithm of a) we finally have a complexity of $O(\min\{n^3, K\}.n^2.Z.(\log K+n))$, for the algorithms of b) and c) a complexity of $O(n^3.K.(Z+n^2))$.

Ganzzahlige Polymatroid-Intersektions-Algorithmen

Zusammenfassung

Auf dem Gebiet der kombinatorischen Optimierung gibt es einige Probleme mit bekannten Algorithmen, in welchen die Technik der Suche nach vergrössernden Wegen angewandt wird.

Beispiele sind etwa das Zuordnungsproblem, das SDR-Problem (System von distinkten Repräsentanten), das unabhängige SDR-Problem, das Branching-Problem. Alle diese Probleme können als sogenannte Matroid-Intersektions-Probleme formuliert werden und mit einem Algorithmus von J.Edmonds oder E.L.Lawler gelöst werden. Andere Beispiele sind Netzwerk-Fluss-Probleme, etwa das Maximalflussproblem, das Finden eines Flusses mit minimalen Kosten oder das Hitchcock-Problem. Zur Lösung dieser Probleme gibt es ebenso bekannte Algorithmen.

Die ganzzahligen Polymatroid-Intersektions-Algorithmen sind Verallgemeinerungen der Matroid-Intersektions-Algorithmen. Sie können aber neben Matroid-Intersektions-Problemen auch die oben erwähnten Netzwerk-Fluss-Probleme im ganzzahligen Fall lösen.

Sei E eine endliche Menge und sei $\mathbb{R}^E$ der Raum der reellwertigen Vektoren $x = [x_e: e\in E]$. Sei $\mathbb{R}_+^E := \{x\in \mathbb{R}^E: 0\le x\}$ und $[0,k]^E := \{x\in \mathbb{R}_+^E: x\le k\in \mathbb{R}\}$.

Ein Polymatroid im Raum $\mathbb{R}^E$ ist eine kompakte, nichtleere Teilmenge von $\mathbb{R}^E$, so dass

(i) falls $0\le x^0\le x^1\in P$, so $x^0\in P$.

(ii) für jedes $a\in \mathbb{R}_+^E$ hat jeder maximale Vektor x aus der Menge aller Vektoren $z\in P$, $z\le a$, d.h. jede P-Basis x von a, die gleiche Summe $\sum\{x_e: e\in E\}$, genannt Rang von a, r(a) (bezüglich P). "Maximal" heisst, dass es kein $z>x$ mit den Eigenschaften von x gibt.

Ein Polymatroid heisst ganzzahlig, falls (ii) auch erfüllt ist, falls a und x auf ganzzahlige Werte beschränkt sind.

J.Edmonds und R.Giles bewiesen, dass alle Polymatroide Polyeder sind, und dass, falls P_1 und P_2 ganzzahlige Polymatroide in $\mathbb{R}^E$ sind, die Eckpunkte von $P_1\cap P_2$ ganzzahlig sind.

Durch Anwendung dieser Sätze entwickeln wir Rahmenalgorithmen, relativ zu einer Subroutine, welche die Elemente der Polymatroide erkennt, zur Lösung der folgenden gazzahligen Polymatroid-Intersektions-Probleme:

a) Finde einen Vektor x unter allen Vektoren $z \in P_1 \cap P_2$, welcher eine maximale Komponentensumme hat (d.h. für welchen $\sum\{x_e: e \in E\}$ maximal ist).
b) Zu einem gegebenen Vektor $c \in \mathbb{R}^E$ finde einen Vektor x unter allen Vektoren $z \in P_1 \cap P_2$, welcher c.x, die gewichtete Komponentensumme, maximiert.
c) Zu einem gegebenen Vektor $c \in \mathbb{R}^E$ und einer Zahl $m \in \mathbb{N}$ finde entweder einen Vektor x unter allen Vektoren $z \in P_1 \cap P_2$, $\sum\{z_e: e \in E\} = m$, für welchen c.x maximal ist, oder beweise, dass es keinen solchen Vektor gibt.

Sei $n := |E|$, $K := \min\{k: k \in \mathbb{N}, P_1 \cap P_2 \subseteq [0,k]^E\}$, Z_k der Rechenaufwand der oben erwähnten Subroutine von P_k, $Z := \max\{Z_1, Z_2\}$.

Vorerst geht es darum, überhaupt direkte Algorithmen zur Lösung von a) bis c) zu finden. Für a) wird ein primaler Algorithmus entwickelt, für b) und c) ein primal-dualer, der den ersteren als Teilalgorithmus enthält. Als Grundlage werden die Matroid-Intersektions-Algorithmen von Edmonds benützt. Es werden ähnliche oder sogar gleiche Strukturen aufgebaut und ähnliche Lemmas bewiesen.

Ein eigentlicher Schlüssel zum Problem war das Auffinden einer sogenannten Transitivitätseigenschaft von Polymatroiden. Dies ist eine Beziehung zwischen den Elementen des Spanns eines Vektors $x \in P$, d.h. derjenigen Elemente $e \in E$, deren in x entsprechende Komponenten x_e ohne gleichzeitige Verkleinerung einer anderen Komponente x_h, $h \in E$, nicht vergrössert werden kann, ansonsten der veränderte Vektor $\hat{x}$ nicht in $P_1 \cap P_2$ ist. Die Transitivitätseigenschaft sagt nun, dass, falls ein drittes Element $g \in E$ existiert und durch Verkleinerung von x_g die Komponente x_h vergrössert werden kann, so kann anstelle von x_h auch x_e vergrössert werden.

Die Transitivitätseigenschaft hilft nun einerseits, einen vergrössernden Weg zu bestimmen. Sie ist ebenso wesentlich beteiligt bei der Abschätzung der Komplexität der Algorithmen. Es stellt sich als eigentliche Hauptschwierigkeit heraus, die Anzahl Augmentationen des primalen Vektors der Algorithmen für a) bis c) und die Anzahl Revisionen des dualen Vektors für b) und c) zu berechnen. Für den Algorithmus für Problem a) ergibt sich schliesslich eine Komplexität von $O(\min\{n^3, K\}.n^2.Z.(\log K+n))$, für die Algorithmen für die Probleme b) und c) eine solche von $O(n^3.K.(Z+n^2))$.

Integral Polymatroid Intersection Algorithms

Abstract

In the field of combinatorial optimization there are some problems with well known algorithms unsing the technique of constructing so called 'augmenting paths'.

Examples are a series of problems, such as the assignement problem, the SDR (system of distinct representatives) problem, the independent SDR problem, the branching problem. All these problems can be formulated as so called matroid intersection problems and can be solved by an algorithm of J.Edmonds or E.L.Lawler. Other examples are network flow problems, such as the maximal flow problem, the minimal cost flow problem or the hitchcock problem. There are well known algorithms to solve the latter problems.

The integral polymatroid intersection algorithms are generalizations of the matroid intersection algorithms. In particuliar, they can solve the above mentioned matroid intersection problems and also the above mentioned network flow problems in the integral case.

Let E be a finite set and let $\mathbb{R}^E$ denote the space of real valued vectors $x = [x_e: e\in E]$. Let $\mathbb{R}^E_+ := \{x\in \mathbb{R}^E: 0\le x\}$ and $[0,k]^E := \{x\in \mathbb{R}^E_+: x\le k\in\mathbb{R}\}$.

A polymatroid P in the space $\mathbb{R}^E$ is a compact nonempty subset of $\mathbb{R}^E_+$ such that

(i) if $0\le x^0\le x^1\in P$, then $x^0\in P$.

(ii) for every $a\in \mathbb{R}^E_+$, every maximal x among all vectors $z\in P$, $z\le a$, i.e. every P-basis x of a, has the same sum $\sum\{x_e: e\in E\}$, called the rank r(a) of a (with respect to P). "Maximal x" means that there is no $z>x$ having the properties of x.

A polymatroid is called integral, if (ii) holds also when a and x are restricted to be integer valued.

J.Edmonds and R.Giles proved that all polymatroids are polyhedra and, if P_1 and P_2 were integral polymatroids in $\mathbb{R}^E$, the vertices of $P_1\cap P_2$ are integral.

Making use of these theorems we present algorithms relative to a subroutine for recognizing the members of the polymatroids to solve the following Integral Polymatroid Intersection problems:

a) Find an x among all $z \in P_1 \cap P_2$ which has a maximal sum of components (i.e. for which $\sum\{x_e: e \in E\}$ is maximal).

b) To a given vector $c \in \mathbb{R}^E$ find a vector x among all $z \in P_1 \cap P_2$, for which c.x, the weighted sum of components, is maximal.

c) To a given vector $c \in \mathbb{R}^E$ and a number $m \in \mathbb{N}$, either find a vector x among all $z \in P_1 \cap P_2$, $\sum\{z_e: e \in E\} = m$, for which c.x is maximal, or prove that no such vector exists.

Let $n := |E|$, $K := \min\{k: k \in \mathbb{N}, P_1 \cap P_2 \subseteq [0,k]^E\}$, Z_k the complexity of the subroutine of P_k mentioned above, $Z := \max\{Z_1, Z_2\}$.

First we have to actually find direct algorithms to solve a) to c). For a) we construct a primal algorithm, for b) and c) a primal-dual algorithm, which uses the first algorithm as a subalgorithm. The algorithms are based on the matroid intersection algorithms of J.Edmonds. We develop similar or equivalent structures and prove similar lemmas.

As a key to the problem we found a so called Transitivity Property of polymatroids. This is a relation between the elements of the span of a vector $x \in P$, i.e. of those elements $e \in E$, whose components x_e of x can not be increased without decreasing simultaneously annother component x_h, $h \in E$, otherwise the changed vector $\hat{x}$ would not be in $P_1 \cap P_2$. The transitivity property says now that if there was a third element $g \in E$, and by decreasing x_g we could increase x_h, then we can as well increase x_e instead of x_h.

The transitivity property helps us to determine augmenting paths. And it is very important in order to estimate the complexity of the algorithms. It turns out that the main difficulty is to compute the number of augmentations of the primal vector in the case of a) to c) and the number of revisions of the dual vector in the case of b) and c). For the algorithm of a) we finally have a complexity of $O(\min\{n^3, K\}.n^2.Z.(\log K + n))$, for the algorithms of b) and c) a complexity of $O(n^3.K.(Z+n^2))$.

Inhaltsverzeichnis

	Seite
Zusammenfassung	i
Abstract	iii
Inhaltsverzeichnis	v
Literaturverzeichnis	vi
Kapitel 1 - Einleitung und Grundlagen	1
Kapitel 2 - Vektoren mit maximaler Komponentensumme in ganzzahligen Polymatroid-Intersektions-Polyedern	16
Kapitel 3 - Vektoren mit maximaler Gewichtssumme in ganzzahligen Polymatroid-Intersektions-Polyedern	42
Kapitel 4 - Vektoren mit maximaler Gewichtssumme unter Vektoren mit einer vorgegebenen Komponentensumme in ganzzahligen Polymatroid-Intersektions-Polyedern	70
Kapitel 5 - Einige spezielle Polymatroide	73
Kapitel 6 - Implementation und offene Probleme	82

Dankadresse

Neben vielen anderen, die zum Gelingen dieser Arbeit beigetragen haben, möchte ich an dieser Stelle besonders danken:

- meiner lieben Frau für ihre tägliche Geduld.
- Prof.Dr.Th.M.Liebling für seine kritische Leitung der Arbeit und die Uebernahme des Korreferates, Prof.Dr.E.Specker für seine Anregungen und die Uebernahme des Referates.
- Prof.Dr.F.Weinberg, dem Leiter des IFOR, für seine wohlwollende Unterstützung der Arbeit, Prof.Dr.R.Giles für eine inspirierende Besprechung, und Dr.H.Gröflin für viele wichtige Ideen und Diskussionen.

Literaturverzeichnis

E1 J.Edmonds: "Submodular Functions, Matroids and Certain Polyhedra", Combinatorial Structures and their Applications, Gordon and Breach (1970), 69-78.

E2 J.Edmonds: "Matroid Intersection", Ann. of Discr. Math. 4(1979), 39-49

E3 J.Edmonds, R.Karp: "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", Journal of the Ass. for Comp. Machinery, Band 19, No.2, April 1972, S. 248-264.

E4 J.Edmonds, D.R.Fulkerson: "Transversals and Matroid Partition", J. Res. NBS 69B, S.147-153 .

E5 J.Edmonds: "Paths, Trees and Flowers", Can. J. Math. 17, No.3, S. 449-467 (1965).

E6 J.Edmonds: "Optimum Branchings", Math. of Decision Sciences, AMS 1968, S. 325-346.

F1 L.R.Ford, Jr., D.R.Fulkerson: "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem", Nav.Res.Logist. Quart. 4(1957) S. 47-55.

F2 L.R.Ford, Jr., D.R.Fulkerson: "Flows in Networks", Princeton Univ. Press, Princeton, N.J. (1962), S. 93-129.

G1 R.Giles: "Submodular Functions, Graphs and Integer Polyhedra", Doctoral Thesis, Univ. of Waterloo, Waterloo, Ontario (1975).

K1 R.Karp: "On the Computational Complexity of Combinatorial Problems", Networks 5(1975), S. 45-68.

K2 A.V.Karzanov: "Determining the Maximal Flow in a Network by the Method of Preflows", Sov.Math.Dokl., Band 15(1974), S. 434-437.

L1 E.L.Lawler: "Combinatorial Optimization, Networks and Matroids", Rinehart & Winston (1976), S.109-120, 169-173, 201-207, 300-355.

S1 P.Schönsleben: "An Implementation of Edmonds' Primal-Dual Weighted Two Matroid Intersection Algorithm Using Separability Properties of Matroids", IFOR Studienberichte, Inst. für Operations Research, ETH Zürich, 7(1978)

1 Einleitung und Grundlagen

1.1 In dieser Arbeit entwickeln wir Algorithmen zur Lösung von sogenanten ganzzahligen Polymatroid-Intersektions-Problemen.

1.2 Grundlegende algebraische Resultate bezüglich Polymatroid-Intersektions-Problemen sind bei J.Edmonds in [E2] und bei R.Giles in [G1] zu finden. Edmonds und Giles studieren die Polymatroiden und Polymatroid-Intersektionen zugeordneten Polyeder. Durch Gebrauch der Theorie der Polyeder und der linearen Programmierung erhielten sie Resultate bezüglich der Seiten dieser Polyeder, und insbesondere bezüglich der nulldimensionalen Seiten, das heisst der Eckpunkte dieser Polyeder. Diese Resultate sind von besonderem Interesse im Fall von ganzzahligen Polymatroiden, wie wir in den Sätzen (1.12), (1.15), (1.20) und (1.21) sehen werden. Diese Sätze sind bereits in [E1] enthalten. Um die Kompatibilität zu wahren, entnehmen wir sie ohen Beweis aus [G1], ebenso auch die Notation und die Definitionen.

1.3 Sowohl [E1] als auch [G1] entahlten keine Algorithmen zur Lösung von Polymatroid-Intersektions-Problemen. Dagegen existieren Algorithmen von J.Edmonds in [E2] und E.L.Lawler in [L1] zur Lösung von Matroid-Intersektions-Problemen, welche einen Spezialfall von Polymatroid-Intersektions-Problemen bilden. Als Matroid-Intersektions-Probleme können, wie wir noch sehen werden, bekannte kombinatorische Probleme formuliert werden.

Von Interesse ist nun, dass die Algorithmen in [E2] und [L1] grosse Aehnlichkeit zu Algorithmen zur Lösung von Netzwerk-Fluss-Problemen aufweisen, so zum Beispiel zum Algorithmus zur Bestimmung eines maximalen Flusses in Netzwerken oder zum Algorithmus zur Lösung des Hitchcock-Problems (vgl. [F1], [F2], [L1]). Dies soll nun kurz erläutert werden.

Wenn es darum geht, ausgehend von einer gegebenen unabhängigen Menge in einem Matroid-Intersektions-Problem eine unabhängige Menge mit

grösserer Kardinalität zu finden, arbeiten die Algorithmen in [E2] und [L1] mit dem Prinzip des Aufsuchens eines sogenannten vergrössernden Weges. Ebenso suchen die Algorithmen in [F1] und [F2] zur Lösung des Hitchcock-Problems oder des Maximalflussproblems in Netzwerken einen vergrössernden Weg, um ausgehend von einem zulässigen Fluss einen zulässigen Fluss mit grösserem Wert zu finden.

Ein vergrössernder Weg ist eine Teilmenge von Elementen eines Matroides (bzw. eine Teilmenge von Kanten in einem Netzwerk).

Im Falle der Matroid-Intersektions-Algorithmen ist die Kardinalität der erwähnten Teilmenge ungerade. Das zweite, vierte, .. , zweitletzte Element des vergrössernden Weges ist aus der bereits bestimmten unabhängigen Menge und das erste, dritte, .. , letzte Element nicht aus der bereits bestimmten unabhängigen Menge. Die Elemente des vergrössernden Weges, welche zur unabhängigen Menge gehörten, werden daraus entfernt, und die anderen Elemente des vergrössernden Weges zur unabhängigen Menge hinzugefügt. Der vergrössernde Weg ist nun derart gewählt, dass die so entstehende Menge wiederum unabhängig in beiden Matroiden ist, aber nun natürlich mit einer um eins grösseren Kardinalität.

Aehnlich werden in Netzwerkproblemen die den Elementen des vergrössernden Weges entsprechenden Komponenten eines zulässigen Flusses um einen Wert $\delta>0$ vergrössert oder verkleinert. Der vergrössernde Weg ist derart gewählt, dass dadurch ein neuer zulässiger Fluss mit einem um δ grösseren Wert entsteht.

Falls kein vergrössernder Weg gefunden werden kann, wird daraus die Maximalität der Kardinalität der unabhängigen Menge in einem Matroid-Intersektions-Problem (bzw. die Maximalität des Flusses in einem Netzwerkproblem) hergeleitet.

Wenn es darum geht, eine unabhängige Menge mit maximalem Gewicht in einem Matroid-Intersektionsproblem oder einen Fluss mit vorgeschriebenem Wert und minimalen Kosten in einem Hitchcock-Problem zu finden, bedienen sich sowohl [E2]/[L1] als auch [F1] eines primal-dualen Algorithmus, welcher den betreffenden im vorherigen Abschnitt erwähnten Algorithmus zur Suche nach einem vergrössernden Weg als Teilalgorithmus benützt. Letzterer wird mehrmals auf verschiedene vom

Originalproblem abweichend konstruierte Probleme angewandt. Die Optimalität wird dann durch ein Paar von Vektoren (x,y) garantiert, wobei x zulässig im primalen Problem, y zulässig im dualen Problem sind, und (x,y) die sogenannten Komplementärschlupfbedingungen erfüllt.

Es war nun bald einzusehen, dass das (ganzzahlige) Hitchcock-Problem und auch das (gazzahlige) Maximalflussproblem als (ganzzahlige) Polymatroid-Intersektions-Probleme formuliert werden können. Die Matroid-Intersektions-Probleme sind sowieso Spezialfälle von ganzzahligen Polymatroid-Intersektions-Problemen. So war denn die Grundlage zur Entwicklung von ganzzahligen Polymatroid-Intersektions-Algorithmen ein genaues Studium der Algorithmen in [E2], [L1], [F1] und [F2].

Dazu kam das Auffinden einer allgemeinen Eigenschaft von Polymatroiden, einer "Transitivitätseigenschaft", welche erst die Formulierung eines direkten Algorithmus für ganzzahlige Polymatroid-Intersektions-Probleme möglich machte. Sie wird in Kapitel 2 beschrieben, besonders in (2.12).

Eine eigentliche Schwierigkeit war sodann die Berechnung der Komplexität der entwickelten Polymatroid-Intersektions-Algorithmen. Als Inspiration wurden Beweise in einer Arbeit von Edmonds und Karp [E3] verwendet (vgl. auch die Behandlung von [E3] in [L1]), in welcher die Komplexität eines Algorithmus zur Lösung des Maximalflussproblems in Netzwerken mit Bestimmung von "kürzesten" vergrössernden Wegen berechnet wird. Andererseits wurden auch unbewiesene Aussagen in [E2] und [L1] bezüglich der Berechnung des Rechenaufwandes für den Algorithmus zur Lösung des gewichteten Matroid-Intersektions-Problems als wesentliche Denkanstösse verwendet.

1.4 Kapitel 1 behandelt im weiteren folgende Grundlagen:

1.5 bis 1.6 :	Notation
1.7 bis 1.8 :	Definition und Beispiele von Matroiden
1.9 bis 1.12:	Definition von Polymatroiden und β_0-Funktionen, eins-zu-eins Beziehung von Polymatroiden und β_0-Funktionen.
1.13 :	Transformation eines Polymatroides in ein Matroid, Beispiele von Polymatroiden
1.14 bis 1.15:	Eckpunkte von ganzzahligen Polymatroiden
1.16 bis 1.17:	Definition und Beispiele von Polymatroid-Intersektions-Problemen
1.18 bis 1.21:	Duales Polymatroid-Intersektions-Problem, das 'Schwache LP-Dualitäts-Theorem', Eckpunkte von ganzzahligen Polymatroid-Intersektionen
1.22 bis 1.23:	Definitionen aus der Komplexitätstheorie.
1.24 bis 1.30:	Kurzvorstellung des Inhalts der Kapitel 2 bis 6.

1.5 Notation

Wir werden das Symbol ":=" für "ist definiert als" gebrauchen, "=" für "ist gleich", "$\subseteq$" für "ist eine Teilmenge von" und "$\subset$" für "ist eine echte Teilmenge von". Die leere Menge wird mit "$\emptyset$" bezeichnet. "$\square$" bezeichnet das Ende oder das Auslassen eines Beweises. Falls ein Ausdruck eine Menge einschliesst, die aus einem einzigen Element e besteht, werden wir die Klammern um e überall dort weglassen, wo keine Missverständnisse entstehen können.

1.6 Sei E eine endliche Menge. Sei L_E die Menge aller Teilmengen von E und sei $K_E := L_E - \emptyset$. Sei $\mathbb{R}^E$ die Menge aller Vektoren $\{[x_e: e \in E]: x_e \in \mathbb{R}$ für alle $e \in E\}$. Für einen Vektor $x = [x_e: e \in E] \in \mathbb{R}^E$ und für ein $e \in E$ wird x_e eine Komponente von x genannt. Der Vektor von $\mathbb{R}^E$, in welchem jede Komponente den Wert k hat, $k \in \mathbb{R}$, wird ebenfalls mit k bezeichnet, wo immer keine Missverständnisse entstehen.
Sei $\mathbb{Z}^E$ die Menge aller Vektoren $\{[x_e: e \in E]: x_e \in \mathbb{Z}$ für alle $e \in E\}$. Falls $x \in \mathbb{Z}^E$, so heisst x ganzzahlig. Sei $\mathbb{R}^E_+ := \{x \in \mathbb{R}^E: 0 \leq x\}$, sei $[0,k]^E := \{x \in \mathbb{R}^E_+: x \leq k \in \mathbb{R}\}$ und sei $\mathbb{Z}^E_+ := \{x \in \mathbb{Z}^E: 0 \leq x\}$.

Für $x,y \in \mathbb{R}^E$ schreiben wir $x \leq y$, falls $x_e \leq y_e$ für alle $e \in E$, und $x < y$, falls $x_e < y_e$ für alle $e \in E$. Klarerweise ist $\mathbb{R}^E$ somit partiell geordnet durch $\leq$ und wir werden den Ausdruck "Maximaler" Vektor in Bezug auf diese partielle Ordnung gebrauchen.

Für ein $x \in \mathbb{R}^E$ sei $x(\emptyset) := 0$ und für $A \in K_E$ sei $x(A) := \sum\{x_e : e \in A\}$. Für ein $x \in \mathbb{R}^E_+$ und ein $A \in L_E$ sei $x|A := [(x|A)_j] \in \mathbb{R}^E$, wobei $(x|A)_j = x_j$, falls $j \in A$, und $(x|A)_j = 0$, falls $j \notin A$.

1.7 Ein Unabhängigkeitssystem $M = (E,F)$ ist eine endliche Menge E zusammen mit einer Familie F von unabhängigen Teilmengen von E, so dass, falls $Y \subseteq Z \in F$, $Y \in F$. Für eine Menge $S \subseteq E$ definieren wir den Rang von S, $r(S)$, in M als die maximale Kardinalität einer unabhängigen Teilmenge von S, und eine Basis von S ist eine unabhängige Teilmengen von S mit Kardinalität $r(S)$. Ein Matroid ist ein Unabhängigkeitssystem, in welchem für alle $S \subseteq E$ jede maximale unabhängige Teilmenge von S eine Basis von S ist.

1.8 Beispiele von Matroiden sind:

(a) Das Matrix-Matroid: Gegeben eine Matrix $A = \{\underline{A}_1, \ldots, \underline{A}_n\}$, wobei $\underline{A}_1, \ldots, \underline{A}_n$ die Kolonnenvektoren von A sind.
Sei $E_A := \{\underline{A}_i : i = 1, \ldots, n\}$ und
sei $F_A := \{$linear unabhängige Teilmengen von $E_A\}$.
Dann ist $M_A = (E_A, F_A)$ ein Matroid (vgl. [L1]).

(b) Das graphische Matroid: Gegeben ein Graph G.
Sei $E_G := \{$Kanten von $G\}$, und
sei $F_G := \{$Kantenmengen von Wäldern in G, d.h. Teilmengen von E_G ohne Kreise$\}$.
Dann ist $M_G = (E_G, F_G)$ ein Matroid (vgl. [L1]).

(c) Das Partitions-Matroid: Gegeben eine endliche Menge $E_p = \dot{\bigcup} \{E_i : i = 1, \ldots, m\}$.

und eine Menge von nichtnegativen ganzen Zahlen
$\{d_i: i=1,..,m\}$.
Sei $F_P := \{J \subseteq E_P: |J \cap E_i| \leq d_i$ für alle $i \in \{1,..,m\}\}$.
Dann ist $M_P = (E_P, F_P)$ ein Matroid (vgl. [L1]).

(d) Das Transversal-Matroid: Sei $Q=\{q_i: i=1,2,..,m\}$ eine Familie von (nicht notwendigerweise verschiedenen) Teilmengen von einer endlichen Menge $E=\{e_j: j=1,2,..,n\}$. Eine Menge $T=\{e_{j(1)},..,e_{j(t)}\}$, $0 \leq t \leq n$, wird ein "partielles Transversal" von Q genannt, falls T aus verschiedenen Elementen in E besteht und verschiedene natürliche Zahlen $i(1),..,i(t)$ gibt, so dass $e_{j(k)} \in q_{i(k)}$ für $k=1,..,t$. So eine Menge wird ein "Transversal" oder ein "System von distinkten (verschiedenen) Repräsentanten (oder SDR) "von Q" genannt, falls $t=m$.

Sei F_X eine Familie von partiellen Transversalen von Q.
Dann ist $M_X = (E, F_X)$ ein Matroid (vgl. [E4]).
Sei F_Y eine Kollektion von Teilfamilien von Q welche Transversale haben.
Dann ist $M_Y = (Q, F_Y)$ ein Matroid (vgl. [E4]).

(e) Das Matching-Matroid: Sei $G=(N,A)$ ein Graph.
Sei $E_C :=$ eine beliebige Teilmenge von N, und
sei $F_C :=$ die Familie aller Teilmengen $I \subseteq E_C$, so dass es ein Matching
(d.h. eine Teilmenge von A so dass keine zwei Kanten inzident zu demselben Knoten sind) gibt, welches alle Knoten in I überdeckt.
Dann ist $M_C = (E_C, F_C)$ ein Matroid (vgl. [E4]).

Es ist bekannt und auch nicht schwierig einzusehen, dass (c) und (b) Spezialfälle von (a) sind, dass (c) ebenso ein Spezialfall von (d) ist und dass (d) ein Spezialfall von (e) ist (vgl. [E4], [L1]).

1.9 Wir können die Definitionen des Unabhängigkeitssystems und des Matroids in die Sprache von Vektoren übertragen. Ein Polyideal P ist eine kompakte Teilmenge von $\mathbb{R}_+^E$, so dass, falls $x^1 \in P$ und $0 \leq x^0 \leq x^1$, auch $x^0 \in P$. Für alle Vektoren $a \in \mathbb{R}_+^E$, ist der Rang von a, $r(a)$, das Maximum von $x(E)$ über alle Vektoren x aus der Menge der Vektoren y, $y \in P$, $y \leq a$. Ein Vektor x aus der Menge aller Vektoren y, $y \in P$, $y \leq a$, welcher $x(E)$ maximiert, heisst eine P-Basis von a. Ein Polymatroid ist ein Polyideal $P \subseteq \mathbb{R}_+^E$, so dass für alle $a \in \mathbb{R}_+^E$ jeder maximale Vektor x aus der Menge aller Vektoren y, $y \in P$, $y \leq a$, eine P-Basis von a ist. Mit anderen Worten, für alle $a \in \mathbb{R}_+^E$ und für alle $x^0 \in P$ mit $x^0 \leq a$, gibt es eine P-Basis x^1 von a so dass $x^0 \leq x^1$ und wir sagen, dass x^0 zu einer P-Basis x^1 von a erweitert werden kann. Ein Polymatroid wird ganzzahlig genannt, falls für alle $a \in \mathbb{Z}_+^E$ jeder maximale ganzzahlige Vektor x aus der Menge aller ganzzahligen Vektoren $y \in P$, $y \leq a$, eine P-Basis von a ist.

1.10 Für eine Funktion $f: L_E \to \mathbb{R}$ sei $P(K_E, f)$ die Lösungsmenge des linearen Systems

$$x_e \geq 0 \quad \text{für alle } e \in E,$$

$$x(S) \leq f(S) \text{ für alle } S \in K_E.$$

1.11 Die Rangfunktion $r: L_E \to \mathbb{R}$ eines Matroids ist derart, dass

- $r(\emptyset) = 0$.
- Falls $A, B \in L_E$ und $A \subseteq B$, so ist $r(A) \leq r(B)$.
- $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$, für alle $A, B \in L_E$ (d.h. r ist submodular auf L_E).
- $r(A \cup e) = r(A)$ oder $r(A \cup e) = r(A)+1$, für alle $e \in E$, $A \in L_E$.

Eine β_0-Funktion $f: L_E \to \mathbb{R}$ ist wie folgt definiert:

- $f(\emptyset) = 0$.
- Falls $A, B \in L_E$ und $A \subseteq B$, so ist $f(A) \leq f(B)$ (d.h. f ist nicht abnehmend).
- $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$, für alle $A, B \in L_E$ (f ist submodular.

Somit ist eine Rangfunktion eines Matroids eine β_0-Funktion f mit $f(e)=0$ oder $f(e)=1$ für alle $e\in E$.

1.12 Satz: Falls $f: L_E \to \mathbb{R}$ eine (ganzzahlige) β_0-Funktion ist, so ist $P(K_E,f)$ ein (ganzzahliges) Polymatroid.
Umgekehrt, falls $P\in\mathbb{R}_+^E$ ein (ganzzahliges) Polymatroid mit Rangfunktion r ist, dann gibt es eine (ganzzahlige) β_0-Funktion $f: L_E \to \mathbb{R}$, so dass $P = P(K_E,f)$; dabei gilt für ein genügend grosses $a\in\mathbb{R}_+^E$ (d.h. $x<a$ für alle $x\in P$): $f(A)=r(a\restriction A)$ für alle $A\in K_E$. □

Diese Beziehung zwischen β_0-Funktionen und Polymatroiden ist fundamental. Ein Matroid ist somit ein Polymatroid enthalten in $[0,1]^E$. Falls das P_M entsprechende Matroid $M=(E,F)$ ist, dann gilt

$$J\in F \iff x := [x_j^J: j\in E,\ x_j^J=1 \text{ falls } j\in J,\ x_j^J=0 \text{ falls } j\in E-J] \in P_M.$$

1.13 Klarerweise gilt, dass alle Matroide Beispiele von Polymatroiden sind. Umgekehrt kann jedes ganzzahlige Polymatroid in ein Matroid transponiert werden, und zwar in der folgenden Art (Giles):

Gegeben ein ganzzahliges Polymatroid $P\subseteq\mathbb{R}_+^E$, $P=P(K_E,f)$. Sei $n:=|E|$. Für ein $e_i\in E$, $i=1,2,..n$ sei $q_i := f(e_i)$. Sei $\overline{K}:=\max\{q_i: i\in\{1,2,..,n\}\}$.
Sei E' aus E und f abgeleitet durch q_i-fache 'Multiplikation' eines Elementes e_i, $i\in\{1,..,n\}$:

$$E' := \{e_{11},e_{12},..,e_{1q_1},e_{21},e_{22},..,e_{2q_2},..,e_{n1},e_{n2},..,e_{nq_n}\}.$$

Somit gilt $|E'| = \sum\{q_i: i\in\{1,..,n\}\} \le \overline{K}.n$.
Für ein $J'\subseteq E'$ sei $x^{J'} := [x_j^{J'}: j\in E', x_j^{J'}=1$, falls $j\in J', x_j^{J'}=0$, falls $j\in E'-J']$.
Sei F' die Familie aller Teilmengen J' von E' so dass

$$\hat{x}^{J'} := [\sum\{x_{1j}^{J'}: j\in\{1,..,q_1\}\}, \sum\{x_{2j}^{J'}: j\in\{1,..,q_2\}\},.., \sum\{x_{nj}^{J'}: j\in\{1,..,q_n\}\}]$$

ein Vektor $\hat{x}^{J'}\in P$ ist. Dann ist $M=(E',F')$ ein Matroid.

In Kapitel 5 werden wir Beispiele von Polymatroiden diskutieren, welche nicht "reine" Matroiden sind, welche aber im ganzzahligen

Fall in Matroide transponiert werden können (mit obiger Methode)

(a) Das Partitions-Polymatroid: Gegeben eine endliche Menge $E = \dot{\cup}\{E_i: i=1,..,m\}$, eine Menge von nichtnegativen ganzen Zahlen $\{b_e: e\in E\}$ und eine Menge von nichtnegativen ganzen Zahlen $\{d_i: i\in\{1,..,m\}\}$. Sei $b: L_E \to \mathbb{N}$ definiert wie folgt:

$$b(A) := \begin{cases} 0 & , \text{falls } A=\emptyset \\ \sum\{b_e: e\in A\}, & \text{sonst} \end{cases} .$$

Sei $f: L_E \to \mathbb{N}$ definiert wie folgt:

$$f(A) := \sum\{\min\{b(A\cap E_i), d_i\}: i\in\{1,..,m\}\} \text{ für alle } A\in L_E .$$

Dann ist $P = P(K_E, f)$ ein ganzzahliges Polymatroid.

(b) Das Kardinalitäts-Polymatroid (Edmonds und Giles): Gegeben eine endliche Menge E und eine Funktion $g: \{0,1,..,|E|\} \to \mathbb{R}$, g nichtnegativ, nicht abnehmend und konkav, also

$$g(i) := 0 + h(1) + h(2) + \ldots + h(i) ,$$

wobei $h: \{1,2,..,|E|\} \to \mathbb{R}$ eine Funktion ist, so dass

$$h(1) \geq h(2) \geq \ldots \geq h(|E|) \geq 0$$

Für $S\in L_E$ sei $f(S) := g(|S|)$. Dann ist $P = P(K_E, f)$ ein Polymatroid.

1.14 Die Eckpunkte eines Polyeders P sind seine nulldimensionalen Seiten. Eine wichtige Eigenschaft der Eckpunkte eines Polyeders P ist folgende: Falls P einen Eckpunkt enthält und die lineare Zielfunktion c.x ein Maximum über $x\in P$ hat, so wird dieses Maximum in einem Eckpunkt von P angenommen.
Deshalb gilt, dass, falls P einen Eckpunkt enthält, das Maximum von c.x über $x\in P$ gleich dem Maximum von c.x über alle Eckpunkte von P ist.
Ebenso gilt, dass x^0 genau dann ein Eckpunkt von P ist, wenn eine lineare Zielfunktion $c^0.x$ existiert, so dass x^0 das einzige Element in P ist, welches $c^0.x$ maximiert.

1.15 Satz: Falls $f: L_E \to \mathbb{R}$ eine ganzzahlige β_0-Funktion ist, dann sind die Eckpunkte von $P(K_E,f)$ ganzzahlig.

Dieser Satz wird von Edmonds und Giles bewiesen durch Lösen des linearen Programms

"Maximiere $c.x$, $c \in \mathbb{R}^E$, über $x \in P(K_E,f)$"

mit dem sogenannten "Greedy Algorithmus" für Polymatroide, indem für jedes $c \in \mathbb{R}^E$ eine ganzzahlige Optimallösung $x^0 \in P(K_E,f)$ gefunden wird.

□

1.16 Seien $f_1,f_2: L_E \to \mathbb{R}$ zwei β_0-Funktionen. Für eine lineare Zielfunktion $c.x$ betrachte das lineare Programm

Maximiere $c.x$, wobei $x \in R^E$ derart, dass

$$\begin{aligned} x_e &\geq 0 && \text{für alle } e \in E \\ x(S) &\leq f_1(S) && \text{für alle } S \in K_E \\ x(S) &\leq f_2(S) && \text{für alle } S \in K_E \quad . \end{aligned}$$

Das ist das sogenannte Polymatroid-Intersektions-Problem. Falls f_1,f_2 ganzzahlige β_0-Funktionen sind, so sprechen wir von einem ganzzahligen Polymatroid-Intersektions-Problem. Der Polyeder, der durch die Lösungsmenge des obigen linearen Systems gebildet wird, heisst (ganzzahliger) Polymatroid-Intersektions-Polyeder. Ein Algorithmus zur Lösung des obigen linearen Programms ist ein (ganzzahliger) Polymatroid-Intersektions-Algorithmus.

1.17 Klarerweise sind alle Matroid-Intersektionsprobleme (wo also f_1,f_2 Rangfunktionen von zwei Matroiden sind) Beispiele von (ganzzahligen) Polymatroid-Intersektions-Problemen.

Zum Beispiel:

- Die Bestimmung einer optimalen Arboreszenz (Optimal Branching) in einem gerichteten Graphen. Dieses Problem wurde mit einem Algorithmus von Edmonds [E6] gelöst. Das Matroid M_1 ist ein unitäres Partitionsmatroid und das Matroid M_2 ist ein graphisches Matroid.

- das Zuordnungsproblem, gelöst durch den 'Ungarischen' Algorithmus von Kuhn (vgl. [L1]). Beide Matroide sind unitäre Partitionsmatroide.

Wir werden in Kapitel 5 das sogenannte Hitchcock-Problem, ein Netzwerk-Flussproblem, als ein Polymatroid-Intersektions-Problem formulieren (beide Polymatroide sind Partitionspolymatroide). Wegen der bekannten Aequivalenz von Hitchcock-Problem und dem 'Minimal Cost Flow'-Problem (d.h. unter den Flüssen mit einem bestimmten Wert denjenigen mit den kleinsten Kosten zu finden, falls es überhaupt einen Fluss mit diesem Wert gibt, (vgl. Kap. 5 und [L1], [F2]), kann das letztere Problem, falls ganzzahlig, ebenfalls mit den in dieser Arbeit präsentierten Algorithmen gelöst werden.

1.18 Das zu (1.16) duale lineare Programm ist das folgende lineare Programm:

$$\text{Minimiere } f.y := f_1 \cdot y^1 + f_2 \cdot y^2, \text{ wobei } y^1, y^2 \in \mathbb{R}^{K_E}$$

derart, dass

$$y_k(S) \geq 0 \quad \text{für alle } S \in K_E, k=1,2 \ ,$$

$$y^1(K_E,e) + y^2(K_E,e) \geq c_e \quad \text{für alle } e \in E \ ,$$

wobei für k=1,2,

$$y^k := [y_k(S): S \in K_E],$$

$$y^k(K_E,e) := \sum\{y_k(S): S \in K_E, e \in S\} \ ,$$

sowie $y := [y^1, y^2]$ gilt.

1.19 Das sogenannte "Schwache Dualitäts-Theorem" der linearen Programmierung sagt, dass für jede primal zulässige Lösung x und jede dual zulässige Lösung y gilt, dass $c.x \leq f.y$.
Darüberhinaus gilt, falls x^0 eine primal zulässige Lösung ist, eine dual zulässige Lösung ist, und $c.x^0 = f.y^0$ gilt, dass eine optimale primale Lösung ist und y^0 eine optimale duale Lösung ist.

Ein primal-dualer Algorithmus konstruiert so ein Paar (x^0,y^0) von Vektoren. Auch die Optimalität des Vektors x^0 im Greedy-Algorithmus, der in (1.15) erwähnt wurde, wurde durch die Berechnung eines dual zulässigen Vektors y^0 mit $c.x^0=f.y^0$ garantiert.

1.20 Satz: Sei (x^0,y^0) ein Paar von Optimallösungen von (1.16), (1.18). Falls c ganzzahlig ist, so kann y^0 ganzzahlig gewählt werden. Falls f ganzzahlig ist, so kann x^0 ganzzahlig gewählt werden. Dieser Satz wird von Edmonds in [E1] und von Giles in [G1] algebraisch bewiesen, ohne Algorithmen dafür zu entwickeln.

1.21 Satz: $\max\{x(T): x\in P_1\cap P_2\} = \min\{f_1(S)+f_2(T-S): S\subseteq T\}$ für $T\subseteq E$
Dieser Satz resultiert als ein Spezialfall von (1.20), wobei $c = 1$.

1.22 Ein Rechenschritt ist eine elementare Computeroperation, für unsere Zwecke eine Addition, eine Subtraktion oder eine Vergleichsoperation.
Die Komplexitätsfunktion eines Algorithmus ist die Anzahl von benötigten Rechenschritten zur Lösung eines gegebenen Problems durch diesen Algorithmus. Eine Funktion $f(n)$, $n\in\mathbb{N}$, sei "$O(g(n))$", falls eine Konstante c existiert, so dass $|f(n)|\leq c.|g(n)|$ für alle Werte von n. Sei nun n die Länge des Input-Strings, d.h. die Anzahl Bits um das Problem mit einer binären Codierung in einem digitalen Computer, zu beschreiben.

Ein polynomialer Algorithmus oder ein 'guter' Algorithmus ist ein Algorithmus, dessen Komplexitätsfunktion $O(p(n))$ für ein Polynom p in n ist. Ein Algorithmus, dessen Komplexitätsfunktion nicht derart beschränkt werden kann, wird ein exponentieller Algorithmus genannt.
Wir nehmen somit an, dass alle elementaren Computeroperationen eine einheitliche Rechenzeit benötigen, ungeachtet der Länge der Operanden. Das dürfen wir tun, weil diese Operationen immer

mit einer Anzahl von Bitoperationen durchgeführt werden können, die polynomial zur Länge des Inputs ist. Falls die Länge des Inputs n ist und f(n) und g(n) zwei Polynome in n sind, bleibt ein Algorithmus, welcher O(f(n).g(n)) Rechenschritte benötigt, immer noch ein guter Algorithmus.

1.23 Ein Problem, für welches es einen guten Algorithmus gibt, gehört zur Klasse der sogenannten P-Probleme.
Es gibt eine allgemeinere Klasse von Problemen, z.B. mit Inputlänge n, welche durch implizite Enumeration von möglicherweise 2^n Lösungskandidaten gelöst werden können, wobei aber mit einem guten Algorithmus erkannt werden kann, ob ein vorgeschlagener Kandidat wirklich eine Lösung ist oder nicht. Ein solcher Algorithmus würde $O(2^n)$ Rechenschritte benötigen. Solche Probleme gehören zur Klasse der sogenannten NP-Probleme. Für eine detailliertere Diskussion der Klassen P und NP vergleiche [K1]. Es gilt $P \subseteq NP$ und konsequenterweise, dass für alle Probleme in NP-P bis heute kein guter Algorithmus bekannt ist.

1.24 Wir werden annehmen, dass für jedes Polymatroid P, auf welches sich ein Polymatroid-Intersektions-Algorithmus als Input bezieht, eine Subroutine gegeben ist, welche für einen Vektor $x \in \mathbb{R}^E$ berechnet, ob $x \in P$ oder nicht. Die Effizienz der Algorithmen, welche in dieser Arbeit entwickelt werden, ist relativ zur Effizienz dieser postulierten Subroutinen.
Die Subroutinen für die Beispiele in (1.8) sind wohlbekannt, diejenigen von (1.13) werden in Kapitel 5 gezeigt.

1.25 In Kapitel 2 entwickeln wir einen guten Algorithmus zur Berechnung eines Vektors mit maximaler Komponentensumme in einem ganzzahligen Polymatroid-Intersektions-Polyeder (also für Problem (1.16), wobei c=1). Sei
$n:=|E|$, $K:=\max\{\min\{f_1(e),f_2(e)\}:e\in E\}$, $Z:=\max\{Z_1,Z_2\}$, wobei für k=1,2, Z_k die Schranke der Subroutine von P_k ist, welche in (1.24) erwähnt wurde. Der Alg. benötigt $O(\min\{n^3,K\}.n^2.Z.(\log K+n))$ Rechenschritte. Er ist 'direkt', d.h. wir führen nicht eine Transforma-

tion in ein Matroid mit n.K Elementen durch, wie dies in (1.13) erwähnt wurde, da ein solcher Algorithmus dann sicher $O(K^d.g(n))$ Rechenschritte benötigen würde, wobei $d \in \mathbb{N}$ und g ein Polynom in n ist. Ein solcher Algorithmus wäre dann kein guter Algorithmus, da die Werte $f_k(e)$, $e \in E$, $k=1,2$, mit $O(\log K.n)$ Bits gespeichert werden können.

Das Prinzip des Aufbaus ist aus [E2] entnommen, die entwickelten Strukturen ähnlich oder gleich und viele der Lemmas ähnlich wie entsprechende Strukturen oder Lemmas in [E2].

1.26 In Kapitel 3 entwickeln wir einen direkten Algorithmus zur Lösung von Problem (1.16). Dieser Algorithmus benützt den Algorithmus in Kapitel 2 als Teil- oder Unteralgorithmus. Seine Komplexität ist $O(n^3.K.(Z+n^2))$.

Auch für diesen Algorithmus entnehmen wir das Prinzip aus [E2]. Wir verwenden wiederum ähnliche Strukturen und Lemmas wie in [E2].

1.27 In Kapitel 4 fügen wir zu Problem (1.16) noch die folgende Gleichung hinzu:

$$x(E) = m \quad , \text{ wobei } m \in \mathbb{N} \quad .$$

Das heisst, wir geben die Komponentensumme des Vektors vor. Wir zeigen dann, dass wir für die Lösung dieses Problems lediglich den Algorithmus in Kapitel 3 etwas abändern müssen. Der neue Algorithmus liefert dann mit der gleichen Komplexität wie jener in Kapitel 3 entweder die gewünschte Lösung oder aber beweist, dass es keinen Vektor mit Komponentensumme m gibt, indem ein Vektor mit kleinerer Komponentensumme angegeben wird, welcher aber maximale Komponentensumme hat.

1.28 Wir werden ohne Beschränkung der Allgemeinheit annehmen, dass $f_k(e)>0$ für alle $e\in E$, $k=1,2$.

Falls nämlich $f_k(e)=0$ für ein $e\in E$, $k=1$ oder $k=2$, so gilt für alle zulässigen Vektoren $x\in P_k$, dass $x_e=0$. Also kann man das Element e aus der Betrachtung weglassen und Problem (1.16) in $\mathbb{R}^{(E-e)}$ lösen.

1.29 In Kapitel 5 präsentieren wir spezielle Polymatroide und ihre Subroutinen. Wir diskutieren die Reduktion des "Minimum Cost Flow"-Problems in Netzwerken auf ein Polymatroid-Intersektions-Problem.

1.30 In Kapitel 6 diskutieren wir schliesslich einige Aspekte einer Implementation der Polymatroid-Intersektions-Algorithmen und erwähnen einige offene Probleme.

2 Vektoren mit maximaler Komponentensumme in ganzzahligen Polymatroid-Intersektions-Polyedern

2.1 Definitionen:

Gegeben eine endliche Menge E, $n:=|E|$. Sei L_E die Menge aller Teilmengen von E, und sei $K_E := L_E - \emptyset$.
Sei $q: E \to \{1,2,..,n\}$ eine Indizierung der Elemente in E.

Für einen Vektor $x \in \mathbb{R}^E$ sei $x(\emptyset) := 0$ und für ein $A \in K_E$ sei $x(A) := \sum\{x_e : e \in A\}$.

Für alle $A \in L_E$ und $x \in \mathbb{R}^E$ sei $x|A$ der Vektor definiert wie folgt:

$$(x|A)_j := \begin{cases} x_j & , \text{ falls } j \in A \\ 0 & , \text{ falls } j \notin A \end{cases} .$$

Für alle $e \in E$, $x \in \mathbb{R}^E$ und $r \in \mathbb{N}$, sei $x^{e,r}$ der Vektor definiert wie folgt:

$$(x^{e,r})_j := \begin{cases} x_j + r & , \text{ falls } j = e \\ x_j & , \text{ falls } j \neq e \end{cases} .$$

Für die folgenden Betrachtungen seien P, P_1, P_2 ganzzahlige Polymatroide in $\mathbb{R}^E$. f, f_1, f_2 seien ihre entsprechenden ganzzahligen β_0-Funktionen. Beachte, dass wegen (1.10) und (1.12) gilt

$$P = P(K_E, f) = \{x \in \mathbb{R}_+^E : x(S) \leq f(S) \text{ für alle } S \in K_E\},$$

beziehungsweise die analogen Beziehungen zwischen P_k und f_k für $k=1,2$.

Sei $\overline{K} := \max\{f(e) : e \in E\}$ und sei $K := \max\{\min\{f_1(e), f_2(e)\} : e \in E\}$.

2.2 Betrachte nun das folgende lineare Programm :

$$\text{Maximiere } x(E), \text{ wobei } x \in \mathbb{R}^E \text{ und}$$

$$\begin{aligned} x_e &\geq 0 && \text{, für alle } e \in E \\ x(S) &\leq f_1(S) && \text{, für alle } S \in K_E \\ x(S) &\leq f_2(S) && \text{, für alle } S \in K_E \end{aligned} .$$

2.3 In [G1] und [E1] finden wir folgende Resultate bezüglich (2.2) (vergleiche dazu Kapitel 1, besonders (1.16) bis (1.21)):

(i) (2.2) hat eine ganzzahlige Optimallösung.

(ii) Für $S \subseteq T \subseteq E$ und $x \in P_1 \cap P_2$ gilt $x(T) \leq f_1(S)+f_2(T-S)$.

(iii) $\text{Max}\{x(T): x \in P_1 \cap P_2\} = \min\{f_1(S)+f_2(T-S): S \subseteq T\}$, $T \subseteq E$.

Beachte, dass (ii) und (iii) auch für nicht-ganzzahlige β_0-Funktionen gültig sind.

2.4 Definition: Gegeben $x \in P$, $A \in L_E$. Falls $x(A)=f(A)$, so ist x eine f-Basis von A, und A ist f-kritisch bezüglich x.

2.5 Lemma (Edmonds in [E1]): Gegeben $x \in P$, $A,B \in L_E$, A,B f-kritisch bezüglich x. Dann sind $A \cup B$ und $A \cap B$ f-kritisch bezüglich x.

Beweis: Wegen der Definition von f gilt $x(\emptyset)=0=f(\emptyset)$.
Wegen der Submodularität von f gilt

$$x(A \cup B) = x(A)+x(B)-x(A \cap B) = f(A)+f(B)-x(A \cap B)$$
$$\geq f(A)+f(B)-f(A \cap B) \geq f(A \cup B) \quad .$$

Da $x \in P$, gilt $x(A \cup B) \leq f(A \cup B)$ und damit Gleichheit. Somit sind $A \cup B$ und $A \cap B$ f-kritisch bezüglich x.

□

2.6 Proposition: Es gibt eine eindeutige maximale f-kritische Menge bezüglich x, für alle $x \in P$.

Beweis: Sei $S := \cup\{A \in L_E: x(A)=f(A)\}$. Dann ist S wegen (2.5) die eindeutige maximale f-kritsche Menge bezüglich x.

□

2.7 Lemma: Gegeben $e \in E$, $x \in P$. Falls es eine f-kritische Menge gibt, welche e enthält, dann gibt es eine eindeutige minimale f-kritische Menge, welche e enthält. Sie heisst $C_e(x)$. Darüberhinaus gilt $x_g > 0$ für alle $g \in C_e(x)$, $g \neq e$.

Beweis: Sei $e \in A \cap B$, wobei A,B f-kritisch bezüglich x, $A,B \in L_E$. Wegen (2.5) ist $A \cap B$ f-kritisch bezüglich x und damit ist

$$C_e(x) := \cap\{A \in K_E : e \in A,\ x(A)=f(A)\}$$

die eindeutige minimale f-kritische Menge bezüglich x, welche e enthält.

Falls $x_g = 0$ für ein $g \in C_e(x)$, $g \neq e$, so gilt

$$f(C_e(x)) = x(C_e(x)) = x(C_e(x)-g) \leq f(C_e(x)-g) \leq f(C_e(x)) ,$$

und damit Gleichheit. Aber dann ist $(C_e(x)-g)$ auch f-kritisch im Widerspruch zur Minimalität von $C_e(x)$.

□

2.8 Definition: Sei $e \in E$, $x \in P$, x ganzzahlig. Falls $x^{e,1} \notin P$, so wird e f-abhängig von x genannt. Die Menge aller f-abhängigen Elemente von x heisst f-Spann des Vektors x oder SV(x).

2.9 Proposition: Sei $x \in P$, x ganzzahlig. Dann sind die Elemente des f-Spanns des Vektors x genau die Elemente der (eindeutigen) maximalen f-kritischen Menge bezüglich x.

Beweis: Sei S die maximale f-kritische Menge bezüglich x. Falls $e \in S$, so gilt $x^{e,1}(S) = f(S)+1 > f(S)$ und damit $x^{e,1} \notin P$. Somit ist e f-abhängig.
Umgekehrt sei $g \in E$ f-abhängig von x, somit $x^{g,1} \notin P$. Weil aber $x \in P$, gilt $x(A) \leq f(A)$ für alle $A \in L_E$. Somit gibt es eine Menge B mit $x^{g,1}(B) > f(B)$, wobei $x^{g,1}(B) = f(B)+1$ sein muss, und damit $x(B) = f(B)$. Aus der Definition von $x^{g,1}$ folgt $g \in B \subseteq S$.

2.10 Für das Folgende nehmen wir an, dass es für jedes Polymatroid, auf welches sich ein Algorithmus als Input bezieht, eine Subroutine gibt, welche für ein $x \in \mathbb{Z}^E$ entscheidet, ob $x \in P$ oder nicht. Die Effizienz der folgenden Algorithmen ist relativ zur Effizienz dieser vorausgesetzten Subroutinen (vgl. [E2]).

2.11 Gegeben ein ganzzahliges Polymatroid $P \subseteq \mathbb{R}^E$. Sei Z der Rechenaufwand der Subroutine, also der Aufwand zur Berechnung, ob $x \in P$ oder nicht. Dann gibt es gute Algorithmen relativ zu Z zur Lösung folgender Probleme:

- Für ein $x \in P$, x ganzzahlig, bestimme SV(x):
  Wegen (2.9) gilt, dass $e \in SV(x)$ genau dann, wenn $x^{e,1} \notin P$.
  Somit können wir SV(x) in $O(|E|.Z)$ Rechenschritten erhalten.

- Für ein $S \in L_E$ berechne f(S) durch die Bestimmung einer f-Basis von S:
  (Ein Polymatroid ist eigentlich eine Struktur, wo dies auf eine einfache Art möglich ist.)
  - Sei $x := \underline{0}$. Bearbeite alle $e \in S$ in einer beliebigen Reihenfolge:
    - Vergrössere x_e mit dem grössten Inkrement δ so dass $\hat{x} := x^{e,\delta}$ ein Vektor in P ist. Definiere $x := \hat{x}$.
    - Beachte, dass jeder maximale Vektor aus der Menge aller ganzzahligen Vektoren y, $y \in P$, $y_e = 0$ für alle $e \notin S$, wegen der Definition eines ganzzahligen Polymatroides und (1.12) eine f-Basis von S ist.
    - Mit binärer Suche des Wertes von δ erhalten wir einen Vektor mit maximaler Komponentensumme in S nach $O(|E|.\log\overline{K}.Z)$ Rechenschritten, wobei $\overline{K} := \max\{f(e): e \in E\}$.

- Für ein $x \in P$, x ganzzahlig, und ein $e \in SV(x)$ berechne $C_e(x)$, die minimale f-kritische Menge von e bezüglich x.
  Sei zur Abkürzung $C := C_e(x)$. Es gilt nun folgende Beziehung:

  $$j \in C \quad <=> \quad x^{e,1}|_{E-j} \in P \quad .$$

  Beweis: Es genügt, die Aussage für ein $j \neq e$ nachzuweisen.
  =>: Sei $j \in C$, $j \neq e$. Wegen (2.7) gilt $x_j \geq 1$. Weil $C_e(x)$ Teilmenge aller f-kritischen Mengen bezüglich x ist, welche e enthalten, gilt deshalb $x^{e,1}|_{E-j} \in P$.
  <=: Falls $j \notin C$, so gilt $(x^{e,1}|_{E-j})(C) = f(C)+1 > f(C)$ und damit $x^{e,1}|_{E-j} \notin P$.

  Um somit C zu bestimmen, initialisieren wir C := e und bearbeiten in einer beliebigen Reihenfolge alle $j \in E-e$ wie folgt:
  - Falls $x^{e,1}|_{E-j} \in P$, so setze $C := C \cup j$.

  Somit kann C in $O(|E|.Z)$ Rechenschritten erhalten werden.

2.12 Lemma: Gegeben $x \in P$, $j,h \in E$, $j \in SV(x)$, $h \in C_j(x)$.
Dann gilt für alle $g \in C_h(x)$, dass $g \in C_j(x)$, somit $C_h(x) \subseteq C_j(x)$.
(Das ist eine Transitivitätseigenschaft von Polymatroiden).

Beweis: Falls g=h, g=j oder h=j, so gibt es nichts zu beweisen. Wir nehmen deshalb an, dass $g \neq h$, $g \neq j$ und $h \neq j$.

Es gilt wegen (2.5), dass $C_j(x) \cap C_h(x)$ eine f-kritische Menge bezüglich x ist. Wegen der Definition gilt, dass $h \in C_j(x) \cap C_h(x)$.

Falls nun $g \in C_j(x)$, so wäre $C_j(x) \cap C_h(x)$ eine echte Teilmenge von $C_h(x)$. Dies wäre im Widerspruch zur Definition von $C_h(x)$, welche die minimale f-kritische Menge bezüglich x ist, welche h enthält.

□

Es folgt eine Illustration dieser Situation. Die minimalen f-kritischen Mengen sind jeweils als "Tropfen" gezeichnet. Punkte auf der Randlinie des Tropfens sind Elemente der Menge und der Punkt auf der Spitze des Tropfens ist das Element, auf welches sich die minimale f-kritische Menge bezieht. Die Transitivitätseigenschaft erlaubt nun folgende Regel in dieser Darstellung: Liegt die Spitze eines Tropfens auf der Randlinie eines zweiten Tropfens, so ist der erste Tropfen ein Bestandteil des zweiten Tropfens. Ein Beispiel soll dies veranschaulichen:

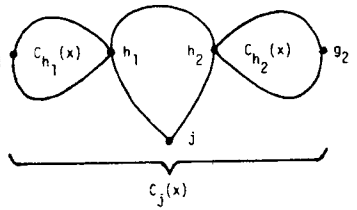


Fig. 2.1
Transitivitätseigenschaft in einem Polymatroid: $h_1, h_2 \in C_j(x)$, somit $C_{h_1}(x) \subseteq C_j(x)$ und $C_{h_2}(x) \subseteq C_j(x)$.

Im Spezialfall des Matroides gilt für alle $h \in E$ mit $x_h > 0$, dass $x_h = 1$ und somit $C_h(x) = \{h\}$. Somit gilt trivialerweise $C_h(x) \subseteq C_j(x)$ für alle $h \in C_j(x)$, $j \in SV(x)$. Die Transitivitätseigenschaft erlaubt uns, wie wir sehen werden, einen guten Algorithmus für das ganzzahlige Polymatroid-Intersektions-Problem zu formulieren. Im Fall von Matroiden hat sie keine Bedeutung.

2.13 Definitionen:

Für das Polymatroid-Intersektions-Problem definieren wir analog

- $SV_k(x)$:= Der f_k-Spann des Vektors x, für k=1,2, $x \in P$.
- $C_e^k(x)$:= Die minimale f_k-kritische Menge bezüglich x, welche e enthält, für $e \in SV_k(x)$, $x \in P$, k=1,2.

2.14 Lemma: Die folgenden Aussagen sind äquivalent: (Vgl.[E2])

(1) $\max\{x(E): x \in P_1 \cap P_2\} = \min\{f_1(T)+f_2(E-T): T \subseteq E\}$.

(2) Es existieren ein $S \subseteq E$ und ein $x \in P_1 \cap P_2$ mit $SV_1(x|S) \cup SV_2(x|E-S)=E$.

(1)=>(2): Für die (1) minimierende Menge T gilt:

$$x(E) = (x|T)(E)+(x|E-T)(E) = (x|T)(T)+(x|E-T)(E-T) = f_1(T)+f_2(E-T).$$

Da $x \in P_1 \cap P_2$, gilt auch $x|T \in P_1 \cap P_2$ und $x|E-T \in P_1 \cap P_2$, und damit $(x|T)(T)=f_1(T)$ und $(x|E-T)(E-T)=f_2(E-T)$.

Aber dann gilt auch $SV_1(x|T) \supseteq T$ und $SV_2(x|E-T) \supseteq E-T$, somit $SV_1(x|T) \cup SV_2(x|E-T)=E$, und somit ist $S:=T$ die gesuchte Menge für (2).

(2)=>(1): Sei $\overline{S}:=SV_1(x|S)$ und $\overline{E-S}:=SV_2(x|E-S)$. Somit gilt $\overline{E-S} \supseteq E-\overline{S}$ $f_1(\overline{S})=(x|S)(\overline{S})$ und $f_2(\overline{E-S})=(x|E-S)(\overline{E-S})$ wegen der Voraussetzung und der Definition von $\overline{S}$ und $\overline{E-S}$.

Dann gelten folgende Beziehungen:

$$x(E) = x(S)+x(E-S) = (x|S)(S)+(x|E-S)(E-S) \geq (x|S)(\overline{S})+(x|E-S)(\overline{E-S}) = f_1(\overline{S})+f_2(\overline{E-S}) \geq f_1(\overline{S})+f_2(E-\overline{S}).$$

Weil $x(E) \leq f_1(A)+f_2(E-A)$ für alle $A \subseteq E$ (vgl. (2.3)), gilt Gleichheit und damit ist $T:=\overline{S}$ die gesuchte Menge für (1).

2.15 Im folgenden wird x jeweils als ganzzahlig angenommen. Wir sind nun bereit, (2.2) zu lösen, und zwar in der folgenden Art:

Gegeben ein $x \in P_1 \cap P_2$ (z.B. x=0). Wir entwickeln einen direkten Algorithmus um entweder einen Vektor $\hat{x} \in P_1 \cap P_2$ mit $\hat{x}(E)>x(E)$ zu finden oder eine Partition von E in S und E-S so dass $SV_1(x|S) \cup SV_2(x|E-S)=E$.
Im ersten Fall benützen wir $\hat{x}$ als neuen Input-Vektor und lassen den Algorithmus ein weiteres Mal laufen, im zweiten Fall wissen wir, dass x maximale Komponentensumme hat.

2.16 Konzept für den Algorithmus

Wir möchten ausgehend vom ganzzahligen Vektor $x \in P$ einen vergrössernden Weg W bezüglich x suchen, der intuitiv etwa fogendermassen konstruiert werden soll:

1) Falls nur P_1 betrachtet wird, so wären alle $e \notin SV_1(x)$ Kandidaten für eine Vergrösserung von x_e um einen Wert $\delta := 1$.
Gibt es kein solches e, so ist S:=E bereits die gesuchte Menge für den Nachweis der maximalen Komponentensumme von x.

2) Andernfalls betrachten wir ein solches Element $e \notin SV_1(x)$, es sei dies v_1. Falls $v_1 \notin SV_2(x)$, so ist bereits ein vergrössernder Weg gefunden, er besteht nur aus dem Element v_1. x_{v_1} wird um δ vergrössert und der dadurch entstehende Vektor $\hat{x}$ ist in P. Ebenfalls gilt $\hat{x}(E)=x(E)+\delta$.

3) Falls $v_1 \in SV_2(x)$, so betrachte $C^2_{v_1}(x)$, die minimale f_2-kritische Menge bezüglich x, welche v_1 enthält. Aus der Definition von $C^2_{v_1}(x)$ wird klar, dass, falls x_{v_1} um den Wert δ vergrössert würde, ein x_h um δ verkleinert werden müsste, wobei $h \neq v_1$, $h \in C^2_{v_1}(x)$, da die Komponentensumme aller Elemente in $C^2_{v_1}(x)$ bereits maximal ist, also $f_2(C^2_{v_1}(x))$ ausmacht. Gibt es kein solches h, so können wir offenbar keinen Weg über v_1 finden und wir müssen im vorhergehenden Schritt ein anderes Element als v_1 wählen. Andernfalls wählen wir ein solches h als Fortsetzung des Weges über v_1 und nennen dieses Element h nun v_2.

4) Falls nun zu v_1 eine um δ vergrösserte und zu v_2 eine um δ ver kleinerte Komponente von x gehörten, so wäre wohl der so veränderte Vektor $\hat{x}$ aus $P_1 \cap P_2$, seine Komponentensumme wäre jedoch gegenüber derjenigen von x nicht vergrössert. Wir dürften jetzt aber, wenn wir wiederum nur P_1 betrachteten, die Komponente eines Elementes $j \neq v_2$, für welches $v_2 \in C^1_j(x)$ gilt, um δ vergrössern. Dies wegen der Definition von $C^1_j(x)$ und weil $v_1 \notin C^1_j(x)$ wegen $v_1 \notin SV_1(x)$. Der so aus $\hat{x}$ entstehende Vektor wäre dann immer noch in P_1.
Gibt es kein solches j, dann können wir offenbar keinen Weg über v_2 finden und wir müssen im vorhergehenden Schritt ein anderes

Element als v_2 wählen. Andernfalls wählen wir ein solches j als Fortsetzung des Weges über v_2 und nennen dieses Element j nun v_3.

5) Falls nun $v_3 \notin SV_2(x)$, so ist der veränderte Vektor auch in P_2 und ein vergrössernder Weg W wäre gefunden, nämlich $W=\{v_1,v_2,v_3\}$. Falls aber $v_3 \in SV_2(x)$, so fahren wir sinngemäss mit Schritt 3 fort, mit "v_3" anstelle von "v_1" und "v_4" anstelle von "v_2", und in Schritt 4 mit "v_5" anstelle von "v_3". So erhalten wir eventuell einen vergrössernden Weg $W=\{v_1,v_2,v_3,v_4,v_5\}$, wobei die zu v_1,v_3,v_5 gehörenden Komponenten vom neuen Vektor $\hat{x}$ gegenüber jenen vom Vektor x um δ vergössert sind, jene zu v_2,v_4 gehörenden um δ verkleinert.

   Das geht so weiter, bis wir unter Umständen einen vergrössernden Weg finden können.

Für Wege mit mehr als drei Elementen ist aber nicht unbedingt sofort einzusehen, dass der veränderte Vektor $\hat{x}$ in $P_1 \cap P_2$ ist. Zudem könnte eventuell sogar ein $\delta \geq 2$ anstelle von $\delta=1$ gewählt werden.

Es ist ebenso nicht sofort intuitiv einzusehen, dass und wie wir, falls wir auf diese Weise keinen vergrössernden Weg finden, daraus die Maximalität der Komponentensumme von x herleiten können.

Es folgt nun ein Beispiel für einen vergrössernden Weg mit fünf Elementen:

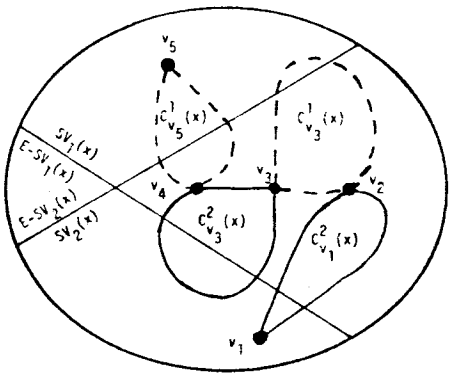


Fig. 2.2 Beispiel für einen vergrössernden Weg W mit fünf Elementen, $W=\{v_1,v_2,v_3,v_4,v_5\}$.

2.17 Die vergrössernde Strunktur $\mathcal{A}$ und der vergrössernde Weg W

Das Mittel zum Auffinden eines vergrössernden Weges ist die Konstruktion einer "vergrössernden Struktur $\mathcal{A}$ bezüglich x" gemäss der in (2.16) intuitiv erläuterten Idee. Das ist eine Partition von E in n+3 (eventuell auch leere) Teilmengen, $E = \dot{\cup}\{E_i: i\in\{1,2,..,n+3\}\}$. Den Index i, $i\in\{1,2,..,n+3\}$ dieser Teilmengen nennen wir "Niveau". $\ell: E \to \{1,2,..,n+3\}$ ist dann die Abbildung, die jedem Element in E sein Niveau in $\mathcal{A}$ zuordnet. Ferner sind noch für alle $i\in\{1,..,n+3\}$ die partiellen Mengensummen $S_i := \dot{\cup}\{E_{i'}: 1\le i'\le i\}$ gegeben und dazu gelte $S_{-1} := \emptyset$, $S_0 := \emptyset$ und $E_0 := \emptyset$.

Die Bestimmung der Mengen E_i erfolgt rekursiv. Die nun folgenden Schritte 1 / 2b / 3 / 4 entsprechen im wesentlichen den Schritten 1 / 2 oder 5 / 3 / 4 in (2.16). Die gegenüber (2.16) erweiterte Idee ist hier diejenige, dass alle möglichen vergrössernden Wege parallel zueinander aufgebaut werden.

1 Sei i:=1. In E_1 gehören alle Elemente $e\notin SV_1(x)$. Jedes dieser Elemente ist Kandidat für den Anfang eines vergrössernden Weges. Also: $E_1 := E - SV_1(x)$.

2a Abbruchkriterium: Falls $S_i = S_{i-2}$, so sei m:=i (beachte: m ungerade), setze $E_{i'} := \emptyset$ für alle $i'\in\{m+1,..,n+2\}$, $E_{n+3} := E-S_m$; $\mathcal{A}$ ist bestimmt; stop. Andernfalls gehe nach 2b.

2b Beachte, dass i ungerade ist. Alle Elemente $e\in E_i$, $e\notin SV_2(x)$, sind Kandidaten für das Ende eines vergrössernden Weges. Tue mit ihnen nichts weiteres und gehe nach 3.

3 Für ein $e\in E_i$, $e\in SV_2(x)$ setze in E_{i+1} alle Elemente h, $h\in C^2_e(x)$, $h\notin S_i$. Falls ein solches h zu einem $e\in E_i$ existiert, ist h Kandidat für die Fortsetzung eines Weges über e, wobei x_h verkleinert würde. Also: $E_{i+1} := \{h: h\in C^2_e(x)-S_i \text{ für ein } e\in E_i\cap SV_2(x)\}$.
Setze i:=i+1 und gehe nach 4.

4 Beachte, dass i gerade ist. Behandle alle Elemente $j\in E-S_i$: Setze j in E_{i+1}, falls ein $h\in E_i$ existiert mit $h\in C^1_j(x)$. So ein j wäre dann ein Kandidat für die Fortsetzung eines Weges über h, wobei x_j vergrössert würde. Also: $E_{i+1} := \{j: j\in E-S_i, C^1_j(x)\cap E_i \neq \emptyset\}$.
Setze i:=i+1 und gehe nach 2a.

Es folgt nun eine Illustration der so definierten Struktur:

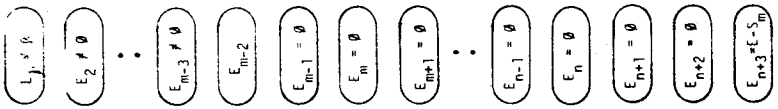


Fig. 2.3 Vergrössernde Struktur $\mathcal{A}$, $E = \dot{\cup}\{E_i : i\in\{1,2,..,n+3\}\}$.

Für alle $i\in\{1,..,n-1\}$ gilt, dass falls $E_i=\emptyset$, so $E_{i+1}=\emptyset$. Damit gilt sofort $m\leq n+2$ und damit $E_i\neq\emptyset$ für alle $i\in\{1,..,m-3\}$. Andererseits gilt wegen $S_m = S_{m-2}$, dass $E_{m-1}=E_m=\emptyset$ und wegen der Definition in Schritt 2a gilt $E_i=\emptyset$ für alle $i\in\{m+1,..,n+2\}$.
Wegen der Konstruktion von $\mathcal{A}$ ist ferner klar, dass für jedes Niveau i, $2\leq i\leq m-2$ und für jedes $g\in E_i$ ein $e\in E_{i-1}$ existiert, so dass, falls i ungerade, $e\in C^1_g(x)$, falls i gerade, $g\in C^2_e(x)$.

Ein alternierender Weg bezüglich x ist eine Menge $A=\{v_1,..,v_k\}$, $1\leq k\leq m-2$, $v_i\in E_i$ für alle i, $1\leq i\leq k$, $v_1\notin SV_1(x)$, so dass für alle i, $2\leq i\leq k$ gilt, falls i ungerade, $v_{i-1}\in C^1_{v_i}(x)$, falls i gerade, so $v_i\in C^2_{v_{i-1}}(x)$.
Ein vergrössernder Weg W bezüglich x und $\mathcal{A}$ ist ein alternierender Weg mit einer ungeraden Anzahl w von Elementen, so dass $v_w\notin SV_2(x)$. Man überlegt sich aufgrund der Konstruktion von $\mathcal{A}$, dass W ein kürzester Weg in dem Sinne ist, dass es von v_1 nach v_w keinen vergrössernden Weg mit einer kleineren Anzahl von Elementen gibt. Ebenso gilt natürlich $w\leq n$.

Der nun folgende Algorithmus (2.18) konstruiert

- entweder einen vergrössernden Weg $W=\{v_1,..,v_w\}$ und einen Vektor $\hat{x}\in P_1\cap P_2$, wobei alle zu Elementen in W mit ungeraden Niveaus gehörenden Komponenten von x um einen Wert $\delta>0$ vergrössert werden, und die zu Elementen in W mit geraden Niveaus gehörenden um δ verkleinert werden (somit $\hat{x}(E)=x(E)+\delta>x(E)$), wobei in (2.19) noch bewiesen wird, dass ein Wert $\delta\geq 1$ gewählt werden kann.
- oder eine Menge S für welche wir dann in (2.20) beweisen werden, dass $SV_1(x|S)\cup SV_2(x|E-S)=E$. Damit hat x maximale Komponentensumme.

2.18 ALGORITHMUS

Gegeben eine endliche Menge E. $q: E \to \{1,..,|E|\}$ eine (eindeutige) Indizierung der Elemente in E. Für k=1,2 sei $P_k \subseteq R^E$ ein ganzzahliges Polymatroid, f_k seine entsprechende β_0-Funktion. Gegeben ein ganzzahliger Vektor $x \in P_1 \cap P_2$. Konstruiere entweder einen ganzzahligen Vektor $\hat{x} \in P_1 \cap P_2$ mit $\hat{x}(E) \geq x(E)+1$ oder eine Partition von E in Mengen S und E-S, so dass $SV_1(x|S) \cup SV_2(x|E-S) = E$.

1 Sei i:=1, $S_{-1} := \emptyset$, $S_0 := \emptyset$. Sei $E_1 := E - SV_1(x)$, $S_1 := E_1$.

2 Falls $S_i = S_{i-2}$, so sei m:=i, $S := E - S_m$ und x hat maximale Komponentensumme. Stop.
Falls $E_i \not\subseteq SV_2(x)$, so gehe nach 5.
Sonst gehe nach 3.

3 Bilde $E_{i+1} := \{h: h \in C_j^2(x) - S_i \text{ für ein } j \in E_i\}$, $S_{i+1} := S_i \oplus E_{i+1}$.
Setze i:=i+1 und gehe nach 4.

4 Bilde $E_{i+1} := \{j: j \in E - S_i,\ C_j^1(x) \cap E_i \neq \emptyset\}$, $S_{i+1} := S_i \oplus E_{i+1}$.
Setze i:=i+1 und gehe nach 2.

5 Sei w:=i. Ein vergrössernder Weg $W=\{v_1,..,v_w\}$ kann folgendermassen bestimmt werden:

- Wähle v_w aus E_w. Für i= w,w-1,..,3,2 wähle v_{i-1}, so dass,
  - falls i ungerade, $q(v_{i-1}) = \min\{q(e): e \in C_{v_i}^1(x),\ e \in E_{i-1}\}$.
  - falls i gerade, $q(v_{i-1}) = \min\{q(e): v_i \in C_e^2(x),\ e \in E_{i-1}\}$.

Gehe nach 6.

6 Für ein $r \in N$ sei x^r der Vektor definiert wie folgt:

$$(x^r)_g := \begin{cases} x_g + r, & \text{falls } g \in \{v_1, v_3, .., v_w\} \\ x_g - r, & \text{falls } g \in \{v_2, v_4, .., v_{w-1}\} \\ x_g & \text{sonst} \end{cases} .$$

Für k=1,2 sei $\delta_k := \max\{r: x^r \in P_k\}$.

Sei $\delta := \min\{\delta_1, \delta_2\}$ und sei $\hat{x} := x^\delta$. Stop.

2.19 Lemma: Falls durch Algorithmus (2.18) ein vergrössernder Weg gefunden werden kann, so kann in Schritt 6 für δ ein Wert grösser oder gleich eins gewählt werden (vgl. [E2]).

Beweis: Gemäss Schritt 6 von Algorithmus (2.18) definieren wir den Vektor x^1 und beweisen, dass $x^1 \in P_1 \cap P_2$.

$x^1 \in P_1$: Für i=1,3,..,w-2,w sei $x_{(i)}$ der Vektor definiert wie folgt:

$$(x_{(i)})_g := \begin{cases} x_g+1, & \text{falls } g\in\{v_1,v_3,..,v_i\} \\ x_g-1, & \text{falls } g\in\{v_2,v_4,..,v_{i-1}\} \\ x_g & \text{sonst} \end{cases} .$$

Sicher gilt $x_{(1)} \in P_1$. Nimm nun an, dass $x_{(i)} \in P_1$, für $1 \le i \le w-2$.

Dann gilt $(x_{(i+2)})_g = \begin{cases} (x_{(i)})_g+1, & \text{falls } g=v_{i+2} \\ (x_{(i)})_g-1, & \text{falls } g=v_{i+1} \\ (x_{(i)})_g & \text{sonst} \end{cases}$.

Sei nun $v:=v_{i+2}$. Es gilt nun $\{v_2,v_4,..,v_{i-1}\} \cap C_v^1(x) = \emptyset$, da sonst wegen der Konstruktion von $\mathcal{A}$ $v \in S_i$ wäre. Damit gilt

$$f_1(C_v^1(x)) = x(C_v^1(x)) \le x_{(i)}(C_v^1(x)) \le f_1(C_v^1(x)) ,$$

somit Gleichheit und damit $x|C_v^1(x) = x_{(i)}|C_v^1(x)$ und damit $C_v^1(x)=C_v^1(x_{(i)})$. Damit gilt wegen der Definition von $C_v^1(x_{(i)})$ und weil $v_{i+1} \in C_v^1(x)$ sofort $x_{(i+2)} \in P_1$.

$x^1 \in P_2$: Für i=w,w-2,..,3,1 sei $x_{(i)}$ der Vektor definiert wie folgt:

$$(x_{(i)})_g := \begin{cases} x_g+1, & \text{falls } g\in\{v_i,v_{i+2},..,v_w\} \\ x_g-1, & \text{falls } g\in\{v_{i+1},..,v_{w-1}\} \\ x_g & \text{sonst} \end{cases} .$$

Sicher gilt $x_{(w)} \in P_2$. Nimm nun an, dass $x_{(i)} \in P_2$, für $3 \le i \le w$.

Dann gilt $(x_{(i-2)})_g = \begin{cases} (x_{(i)})_g+1, & \text{falls } g=v_{i-2} \\ (x_{(i)})_g-1, & \text{falls } g=v_{i-1} \\ (x_{(i)})_g & \text{sonst} \end{cases}$.

Sei nun $v:=v_{i-2}$. Es gilt nun $\{v_{i+1},v_{i+3},..,v_{w-1}\} \cap C_v^2(x) = \emptyset$, da wegen der Konstruktion von $\mathcal{A}$ gilt, dass $C_v^2(x) \subseteq S_{i-1}$.

Damit gilt $f_2(C_v^2(x)) = x(C_v^2(x)) \le x_{(i)}(C_v^2(x)) \le f_2(C_v^2(x))$,

somit Gleichheit und damit $x|C_v^2(x) = x_{(i)}|C_v^2(x)$ und damit $C_v^2(x)=C_v^2(x_{(i)})$. Damit gilt wegen der Definition von $C_v^2(x_{(i)})$ und weil $v_{i-1} \in C_v^2(x)$ sofort $x_{(i-2)} \in P_2$.

□

2.20 Lemma Falls $S_m=S_{m-2}$ in Schritt 2 des Algorithmus (2.18), so gilt

a) $S_m \subseteq SV_2(x|S_m)$ und b) $E-S_m = SV_1(x|E-S_m)$.

$E-S_m$ ist somit die gesuchte Menge S in (2.14) für den Nachweis der Maximalität der Komponentensumme von x.

Beweis:

a) Es genügt der Nachweis, dass $C_e^2(x)\subseteq S_m$ für alle $e\in S_m$:
- Es gilt $E_m=E_{m-1}=\emptyset$. Wegen der Konstruktion von $\mathcal{A}$ in (2.18), besonders Schritt 3 von (2.18) gilt für ein $e\in E_i$, $i\in\{1,..,m-2\}$:
  - falls i ungerade, so ist $C_e^2(x)\subseteq S_{i+1}\subseteq S_m$.
  - falls i gerade, so gibt es ein $g\in E_{i-1}$ mit $e\in C_g^2(x)\subseteq S_i$ und damit wegen (2.12) $C_e^2(x)\subseteq C_g^2(x)\subseteq S_i\subseteq S_m$.

b) $\subseteq$: Es genügt der Nachweis, dass $C_e^1(x)\subseteq E-S_m$ für alle $e\in E-S_m$.
- Nimm an, dass $e\in E-S_m$ für ein $e\in E$. Sicher gilt $e\in SV_1(x)$, da sonst $e\in E_1\subseteq S_m$ wäre. Somit ist $C_e^1(x)$ definiert.
- Nimm nun weiter an, dass $C_e^1(x)\cap S_m \neq \emptyset$. Da $E_m=E_{m-1}=\emptyset$ und $E_1\cap SV_1(x) = \emptyset$, gibt es ein $g\in C_e^1(x)$, $g\in E_i$, $i\in\{2,..,m-2\}$ Wegen der Konstruktion von $\mathcal{A}$ in (2.18), besonders Schritt 4 in (2.18), gilt:
  - falls i gerade, so ist $e\in S_{i+1}\subseteq S_m$.
  - falls i ungerade, so gilt $i\geq 3$, und weil $g\in E_i$, so gibt es ein $h\in E_{i-1}$ mit $h\in C_g^1(x)$. Damit ist aber wegen (2.12) $h\in C_e^1(x)$ und somit $e\in S_i\subseteq S_m$.
- In beiden Fällen erfolgt somit ein Widerspruch. Somit gilt $C_e^1(x)\cap S_m= \emptyset$ und damit $C_e^1(x) \subseteq E-S_m$.

$\supseteq$: - Nimm an, dass $e\in SV_1(x|E-S_m)$. Somit ist $e\in SV_1(x)$ und es existiert $C_e^1(x)$. Wegen (2.7) gilt $x_g>0$ für alle $g\in C_e^1(x)$, $g\neq e$. Da $e\in SV_1(x|E-S_m)$, gilt $g\in E-S_m$ für alle $g\in C_e^1(x)$ mit $x_g>0$.
- Somit gilt, dass,falls $C_e^1(x)\cap S_m\neq \emptyset$, so $C_e^1(x)\cap S_m= \{e\}$ und $x_e=0$. Wegen Konstruktion von $\mathcal{A}$ in (2.18) besonders Schritt 3 von (2.18) und wegen (2.7) muss $x_h>0$ für alle $h\in E_i$, i gerade, gelten. Somit wäre $e\in E_i$ für ein ungerades Niveau i, $i\geq 3$. Damit muss es aber ein $h\in E_{i-1}\subseteq S_m$ geben, mit $h\in C_e^1(x)$. Dies ist aber im Widerspruch zu oben, da $g\in E-S_m$ für alle $g\in C_e^1(x)$, $g\neq e$, gilt.
- Somit ist $C_e^1(x)\cap S_m=\emptyset$ und damit auch $e\in E-S_m$.

□

2.21 Komplexität des Algorithmus

Seien n und K wie in (2.1) definiert.
Für k=1,2, sei Z_k der Rechenaufwand der Subroutine von P_k, also der Aufwand zur Berechnung, ob $x \in P_k$ oder nicht (vgl. (2.10)).
Sei $Z := \max\{Z_1, Z_2\}$.

Nach (2.11) können alle Berechnungen der minimalen f_k-kritischen Mengen der Elemente $e \in E$, für k=1,2, in $O(n^2.Z)$ Rechenschritten erfolgen. (Beachte: eigentlich werden nur die Mengen $C_e^k(x)$, k=1,2, für Elemente e mit ungeradem Niveau in $\mathcal{A}$ benötigt).

Die vergrössernde Struktur $\mathcal{A}$ ist $O(n^2)$ Rechenschritten konstruiert, da die Behandlung eines Elementes $e \in E_i$ für ein Niveau i jeweils O(n) Rechenschritte benötigt und es höchstens n zu behandelnde Elemente gibt.
Nach fertig erstellter Struktur $\mathcal{A}$ bricht der Algorithmus entweder ab oder verzweigt nach Schritt 5.
Weil ein vergrössernder Weg höchstens n Elemente hat, benötigt Schritt 5 höchstens n^2 Rechenschritte.
Schritt 6 benötigt O(logK.n.Z) Rechenschritte, weil δ mit binärer Suche gefunden werden kann. Im Verlauf dieser Suche müssen wir für jeden Wert $1 \le r \le K$, welcher Kandidat für den Wert von δ sein soll, zuerst den Vektor x^r berechnen und dann testen, ob $x \in P_1 \cap P_2$ oder nicht.
Somit ist die Komplexität des Algorithmus $O(n^2.Z + n^2 + n.Z.\log K)$ oder $O(n.Z.(\log K + n))$.

2.22 Definitionen

Sei eine Augmentation definiert als eine Anwendung von Algorithmus (2.20). Aufeinanderfolgende Augmentationen sind Augmentationen, wo der Output-Vektor einer Augmentation jeweils der Input-Vektor der nächsten Augmentation ist.

Sei x der Input-Vektor einer beliebigen Augmentation. $\mathcal{A}$ sei dann die während dieser Augmentation aufgebaute vergrössernde Struktur bezüglich x, die Partition $E = \bigcup\{E_i : i\in\{1,..,n+3\}\}$ und die Funktion $\ell: E \to \{1,..,n+3\}$ sind dann bezüglich x und $\mathcal{A}$ definiert.

Falls ein vergrössernder Weg W bezüglich x gefunden wird, so ist $W=\{v_1,..,v_w\}$ (beachte: w ist ungerade). $\hat{x}$ ist dann der Output-Vektor dieser Augmentation und gleichzeitig der Input-Vektor der nächsten Augmentation, während der die vergrössernde Struktur $\hat{\mathcal{A}}$ bezüglich $\hat{x}$ aufgebaut wird.
$E = \bigcup\{\hat{E}_i : i\in\{1,..,n+3\}\}$ ist dann die betreffende Partition und $\hat{\ell}: E \to \{1,..,n+3\}$ ist die Abbildung, die jedem Element in E sein Niveau in $\hat{\mathcal{A}}$ zuordnet.
Falls nochmals ein vergrössernder Weg in $\hat{\mathcal{A}}$ gefunden werden kann, so heisst er $\hat{W}$, seine Länge $\hat{w}$, und $\hat{W}=\{\hat{v}_1,..,\hat{v}_{\hat{w}}\}$.

Sei $i\geq 2$. v_i heisst '<u>kritisch auf dem Niveau i</u>', wenn,
- falls i gerade, $v_i \notin C^2_{v_{i-1}}(\hat{x})$ (beachte, dass $v_i \in C^2_{v_{i-1}}(x)$ wegen Konstruktion von $\mathcal{A}$ und W in (2.17)).
- falls i ungerade, $v_{i-1} \notin C^1_{v_i}(\hat{x})$ (beachte, dass $v_{i-1} \in C^1_{v_i}(x)$ wegen Konstruktion von $\mathcal{A}$ und W in (2.17)).

2.23 Satz: Ein Vektor $x \in P_1 \cap P_2$ mit maximaler Komponentensumme kann mit $O(\min\{n^3, K\}.n^2.Z.(\log K+n))$ Rechenschritten erhalten werden.

Beweis:

- Wegen (2.21) ist die Komplexität von Alg.(2.18) $O(n.Z.(\log K+n))$.
- Es gibt höchstens n.K Augmentationen, da jede Augmentation die Kompnenentensumme des Vektors um mindestens 1 erhöht.
- Der schwierige Teil des Beweises wird mit Lemma (2.33) gezeigt: Ein Vektor mit max. Komponentensumme wird unabh. von K nach höchstens $\frac{n^4}{2} - \frac{n^3}{2} + 2n$ aufeinanderfolgenden Augmentationen erhalten.
- (2.24) bis (2.32) sind Vorbereitungen für den Beweis von (2.33).
  - (2.24) zeigt, dass für ein $e \in SV_k(x)$, k=1,2, e mit Niveau i, die Elemente von $C_e^k(x)$ für k=1 mindestens das Niveau i-1 haben, und für k=2 höchstens das Niveau i+1.
  - (2.25) definiert für jedes $e \in SV_k(x)$, k=1,2, eine Menge $B_e^k(x)$, von welcher gezeigt wird, dass sie die gleiche Eigenschaft wie die vorher für $C_e^k(x)$ erwähnte hat, und welche zudem f_k-kritisch bezüglich sowohl x als auch $\hat{x}$ ist. Die entsprechenden Beweise werden in (2.26) und (2.27) geführt.
  - (2.28) formt dann die Aussagen etwas um für den Gebrauch in den folgenden Lemmas. Zum Beispiel resultieren sofort die Aussagen dass $C_e^k(\hat{x}) \subseteq B_e^k(x)$ für $e \in SV_k(x)$, k=1,2, und dass der f_k-Spann von $\hat{x}$ denjenigen von x enthält, für k=1,2.
  - (2.29) zeigt, dass, falls $SV_2(\hat{x})=SV_2(x)$, bei aufeinanderfolgenden Augmentationen das Niveau eines Elementes nicht sinkt.
  - (2.30) und (2.31) sind Aussagen für den Fall $SV_2(\hat{x})=SV_2(x)$ und ein Element $e \in E$, das in zwei aufeinanderfolgenden Augmentationen das gleiche Niveau i, $2 \le i \le n$, beibehält. (2.30) zeigt, dass die Vorgänger von e in $\hat{\mathcal{A}}$, also diejenigen Elemente in $\hat{E}_{i-1}$, von welchen jedes genügte, um e in $\hat{E}_i$ zu bringen, auch in $\mathcal{A}$ das Niveau i-1 gehabt haben müssen. (2.31) zeigt, dass, falls jene Elemente in E_{i-1} alle einen grösseren Index aufweisen als ein weiteres Element $d \in E_{i-1}$, welches nicht zu jenen Elementen gehört, dieses Element d nicht Vorgänger von e in $\hat{\mathcal{A}}$ werden kann, auch wenn es in $\hat{\mathcal{A}}$ das Niveau i-1 beibehält.
  - Aus (2.29) bis (2.31) wird in (2.32) hergeleitet, dass, solange der f_2-Spann des Inputvektors von aufeinanderfolgenden Augmentationen nicht ändert, ein Element höchstens n-i+1 Male kritisch auf einem Niveau i, $2 \le i \le n$, sein kann.

□

2.24 Lemma: 1) Für alle $e \in SV_1(x)$ gilt, dass $C_e^1(x)$ nur Elemente $g \in E$ enthält mit

$$\ell(g) \geq \begin{cases} \ell(e) & \text{, falls } \ell(e) \text{ gerade} \\ \ell(e)-1 & \text{, falls } \ell(e) \text{ ungerade} \end{cases} .$$

2) Für alle $e \in SV_2(x)$ gilt, dass $C_e^2(x)$ nur Elemente $g \in E$ enthält mit

$$\ell(g) \leq \begin{cases} \ell(e) & \text{, falls } \ell(e) \text{ gerade} \\ \ell(e)+1 & \text{, falls } \ell(e) \text{ ungerade} \end{cases} .$$

Beweis:

1): Falls $\ell(e)$ gerade, so sei $i := \ell(e)$, falls $\ell(e)$ ungerade, so sei $i := \ell(e)-1$. Betrachte ein $g \in C_e^1(x)$. Es gilt $\ell(e) \geq 2$ und $\ell(g) \geq 2$, da sonst $e, g \notin SV_1(x)$ und damit $e, g \notin C_e^1(x)$. Nimm nun an, dass $\ell(g) < i$.

Wegen der Konstruktion von $\mathcal{A}$ in (2.17) und besonders aus dem Schritt 4 von (2.17) folgt, dass

- falls $\ell(g)$ gerade, so gilt $\ell(e) \leq \ell(g)+1 \leq i-2+1 < i$,
- falls $\ell(g)$ ungerade, so ist $\ell(g) \geq 3$ und es gibt ein h mit $\ell(h) = \ell(g)-1$, $h \in C_g^1(x)$. Dann gilt aber wegen (2.12) $h \in C_e^1(x)$ und somit $\ell(e) \leq \ell(h)+1 = \ell(g) < i$.

In beiden Fällen folgt ein Widerspruch und somit gilt $\ell(g) \geq i$.

2): Betrachte ein Element g, $g \in C_e^2(x)$.

Aus der Konstruktion von $\mathcal{A}$ in (2.17), besonders aus dem Schritt 3 von (2.17) folgt, dass

- falls $\ell(e)$ ungerade, so gilt $\ell(g) \leq \ell(e)+1$,
- falls $\ell(e)$ gerade, so gibt es ein j mit $\ell(j) = \ell(e)-1$, $e \in C_j^2(x)$, und somit gilt wegen (2.12) $g \in C_j^2(x)$. Damit gilt aber $\ell(g) \leq \ell(j)+1 = \ell(e)$.

□

2.25 Definition: Gegeben x, $\mathcal{A}$, ℓ und $W = \{v_1,..,v_w\}$.
Wir definieren für k=1,2 und für alle $e \in SV_k(x)$ eine Menge $B_e^k(x)$ rekursiv wie folgt:

1 Sei $B_e^k(x) := C_e^k(x)$.

2 Suche ein $v_i \in B_e^k(x)$, $i \in \{2,4,..,w-1\}$, für welches das Element j, $j:=v_{i+1}$ falls k=1, $j:=v_{i-1}$ falls k=2, nicht in $B_e^k(x)$ enthalten ist.
Falls kein solches v_i existiert, stop. Sonst gehe nach 3.

3 Wähle ein v_i gemäss Schritt 2 und bestimme das Element j gemäss Schritt 2. Setze $B_e^k(x) := B_e^k(x) \cup C_j^k(x)$ und gehe nach 2.

Die Bedeutung dieser Mengen $B_e^k(x)$ ist nun die folgende (alle nun folgenden Aussagen werden in den nächsten Lemmas bewiesen werden): $B_e^k(x)$ ist f_k-kritisch bezüglich x. Darüber hinaus ist $B_e^k(x)$ aber auch f_k-kritisch bezüglich $\hat{x}$, und wegen $e \in B_e^k(x)$ gilt, dass $C_e^k(\hat{x})$ existiert und eine Teilmenge von $B_e^k(x)$ ist. Somit erhalten wir eine Beziehung zwischen den minimalen f_k-kritischen Mengen von e bezüglich x und $\hat{x}$, und, da eine vergrössernde Struktur mit Beziehung auf solche minimale f_k-kritische Mengen gebaut wird, erhalten wir auch Beziehungen zwichen $\mathcal{A}$ und $\hat{\mathcal{A}}$.

Die folgende Illustration gibt ein Beispiel für ein $e \in E$ mit Niveau $\ell(e)=i$, i=5. Das Beispiel gebraucht eine in 2.26 bewiesene Aussage, dass nämlich $B_e^1(x)$ nur Elemente g mit $\ell(g) \geq 4$ enthalten kann.

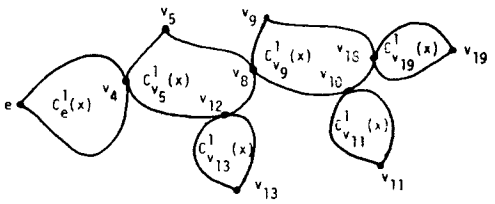


Fig. 2.5 Beispiel einer Menge $B_e^1(x)$ für ein $e \in E_i$, $i \leq 5$

2.26 Lemma: 1) Für alle $e \in SV_1(x)$ enthält $B_e^1(x)$ das Element e und daneben nur Elemente $g \in E$ mit

$$\ell(g) \geq \begin{cases} \ell(e) & \text{, falls } \ell(e) \text{ gerade} \\ \ell(e)-1 & \text{, falls } \ell(e) \text{ ungerade} \end{cases} .$$

2) Für alle $e \in SV_2(x)$ enthält $B_e^2(x)$ das Element e und daneben nur Elemente $g \in E$ mit

$$\ell(g) \leq \begin{cases} \ell(e) & \text{, falls } \ell(e) \text{ gerade} \\ \ell(e)+1 & \text{, falls } \ell(e) \text{ ungerade} \end{cases} .$$

Beweis:

1): Der Beweis wird induktiv geführt durch eine Nachverfolgung der Konstruktion von $B_e^1(x)$ in (2.25). Dabei bildet ein einmaliges Durchlaufen der Schritte 2 und 3 einen Induktionsschritt. Die Induktionsvoraussetzung ist jeweils, dass zu Beginn des Schrittes 2 die Menge $B_e^1(x)$ die Aussage (2.26.1) erfüllt. Falls Schritt 3 durchlaufen werden muss, so wird die Induktionsbehauptung bewiesen, dass die dort erweiterte Menge $B_e^1(x)$ wieder (2.26.1) erfüllt.

Die Induktionsverankerung geschieht mit dem einmaligen Durchlaufen von Schritt 1 . Wegen $B_e^1(x)=C_e^1(x)$ gilt $e \in B_e^1(x)$ und wegen (2.24.1) hat $B_e^1(x)$ die erforderlichen Eigenschaften.

Beweis des Induktionsschrittes: Sei ein $v_i \in B_e^1(x)$ gemäss Schritt 2 von (2.25) gefunden und sei j das dort zu v_i definierte Element. Es gilt dann

$$\ell(j) = \ell(v_i)+1 \geq \begin{cases} \ell(e)+1, & \text{falls } \ell(e) \text{ gerade} \\ \ell(e) & \text{, falls } \ell(e) \text{ ungerade} . \end{cases}$$

Wegen (2.24.1), angewendet auf j und $C_j^1(x)$, gilt nun für jedes $g \in C_j^1(x)$, dass $\ell(g) \geq \ell(j)-1$. Damit gilt aber für $B_e^1(x) \cup C_j^1(x)$ sofort die Induktionsbehauptung .

2): Analog zum Beweis von (2.26.1). Anstelle von "$B_e^1(x)$" steht "$B_e^2(x)$", anstelle von "$C_e^1(x)$" oder "$C_j^1(x)$" stehen "$C_e^2(x)$" bzw. "$C_j^2(x)$" , anstelle von "$\geq$" steht "$\leq$" in den Ungleichungen, und anstelle von "+" steht "-" und umgekehrt.

□

2.27 Proposition: 1) Für alle $e \in SV_1(x)$ ist $B_e^1(x)$ f_1-kritisch bezüglich sowohl x als auch $\hat{x}$.

2) Für alle $e \in SV_2(x)$ ist $B_e^2(x)$ f_2-kritisch bezüglich sowohl x als auch $\hat{x}$.

Beweis:

1): - Wegen der Konstruktion in (2.25) gilt $x(B_e^1(x)) = f_1(B_e^1(x))$, weil $B_e^1(x)$ die Vereinigung von f_1-kritischen Mengen bezüglich x ist und (2.5) gilt.

- Wegen der Konstruktion in (2.25) ist ebenso klar, dass zu jedem $v_i \in B_e^1(x)$, $i \in \{2,4,..,w-1\}$ auch $v_{i+1} \in B_e^1(x)$.

- Umgekehrt gilt, dass es zu jedem $v_i \in B_e^1(x)$, $i \in \{1,3,..,w\}$, das Element v_{i-1} gibt, da $v_1 \notin SV_1(x)$ und damit $v_1 \notin B_e^1(x)$. Ist nun $v_i \in B_e^1(x)$, so ist v_i Element einer der $B_e^1(x)$ bildenden minimalen f_1-kritischen Mengen bezüglich x, es sei $v_i \in C_g^1(x)$ für ein $g \in B_e^1(x)$. Da $v_{i-1} \in C^1_{v_i}$, gilt wegen (2.12) auch $v_{i-1} \in C_g^1(x) \subseteq B_e^1(x)$.

- Somit ist die Anzahl der zu Elementen in $B_e^1(x)$ gehörenden Komponenten von x, welche durch die Augmentation vergrössert werden, gleich der Anzahl der zu Elementen in $B_e^1(x)$ gehörenden Komponenten von x, welche durch die Augmentation verkleinert werden. Damit gilt $\hat{x}(B_e^1(x)) = x(B_e^1(x)) = f_1(B_e^1(x))$.

2): Analog zum Beweis von (2.27.1) Anstelle von "$B_e^1(x)$", "$SV_1(x)$", "f_1" und "$C_g^1(x)$" stehen "$B_e^2(x)$", "$SV_2(x)$", "f_2" und "$C_g^2(x)$", anstelle von "v_{i-1}", "v_{i+1}" und "v_1" stehen "v_{i+1}", "v_{i-1}" und "v_w".

□

2.28 Proposition: 1) Für alle $e \in SV_1(x)$ gelten die Aussagen 1a) bis 1d).
2) Für alle $e \in SV_2(x)$ gelten die Aussagen 2a) bis 2d).

1a) $e \in SV_1(\tilde{x})$, somit $SV_1(\tilde{x}) \supseteq SV_1(x)$.

1b) $C_e^1(\tilde{x}) \subseteq B_e^1(x)$.

Sei $i := \ell(e)$, falls $\ell(e)$ ungerade, $i := \ell(e)+1$, falls $\ell(e)$ gerade.

1c) Falls $v_{i-1} \notin C_e^1(x)$, so enthält $C_e^1(\tilde{x}) - C_e^1(x)$ nur Elemente g mit $\ell(g) \geq i+1$.

1d) Falls $v_{i-1} \in C_e^1(x)$, so enthält $C_e^1(\tilde{x}) - (C_e^1(x) \cup C_{v_i}^1(x))$ nur Elemente g mit $\ell(g) \geq i+1$.

2a) $e \in SV_2(\tilde{x})$, somit $SV_2(\tilde{x}) \supseteq SV_2(x)$.

2b) $C_e^2(\tilde{x}) \subseteq B_e^2(x)$.

Sei $i := \ell(e)$, falls $\ell(e)$ gerade, $i := \ell(e)+1$, falls $\ell(e)$ ungerade.

2c) Falls $v_i \notin C_e^2(x)$, so enthält $C_e^2(\tilde{x}) - C_e^2(x)$ nur Elemente g mit $\ell(g) \leq i-2$.

2d) Falls $v_i \in C_e^2(x)$, so enthält $C_e^2(\tilde{x}) - (C_e^2(x) \cup C_{v_{i-1}}^2(x))$ nur Elemente g mit $\ell(g) \leq i-2$.

Beweise für 1a) bis 1d): (die Beweise für 2a) bis 2d) sind ähnlich)

1a): Wegen (2.26.1) und (2.27.1) folgt die Aussage sofort.

1b): Wegen (2.26.1), (2.27.1) und (2.7) folgt die Aussage sofort.

1c): Wegen 1b), (2.26.1) und der Konstruktion von $B_e^1(x)$ in (2.25) sowie Anwendung von (2.24.1) auf die Elemente in $B_e^1(x) - C_e^1(x)$ folgt die Aussage sofort. Betrachte als Beispiel die Situation in Fig. 2.5 von (2.25) mit $\ell(e)=3$, d.h. $i=3$.

1d): Wegen 1b), (2.26.1) und der Konstruktion von $B_e^1(x)$ in (2.25) sowie Anwendung von (2.24.1) auf die Elemente in $B_e^1(x) - (C_e^1(x) \cup C_{v_i}^1(x))$ folgt die Aussage sofort. Betrachte als Beispiel die Situation in Fig.2.5 von (2.25) mit $\ell(e)=5$, d.h. $i=5$.

□

2.29 Lemma: Falls $SV_2(\hat{x})=SV_2(x)$, so gilt $\hat{\ell}(e) \geq \ell(e)$ für alle $e \in E$.

Beweis: Durch Induktion nach den Niveaus i in $\hat{\mathcal{A}}$, $i \in \{1,..,n+3\}$.

Zuerst wird das Lemma für alle Elemente mit Niveau i=1 bewiesen. Dann wird angenommen, dass das Lemma für alle Elemente $g \in E$ mit Niveau $\ell(g) \leq i-1$ gilt für ein i, $2 \leq i \leq n$. Daraus wird die Gültigkeit des Lemmas für alle Elemente mit Niveau i hergeleitet. Die Niveaus n+1, n+2 sind leer, und für das Niveau n+3 ist dann nichts mehr zu beweisen.

Verankerung: Für alle Elemente $e \in \hat{E}_1$ gilt $e \notin SV_1(\hat{x})$ und somit wegen (2.28.1a) auch $e \notin SV_1(x)$ und damit $\ell(e)=\hat{\ell}(e)=1$.

Induktionsvoraussetzung: Sei $\hat{\ell}(e) \geq \ell(e)$ für alle $e \in E$ mit $\hat{\ell}(e) \leq i-1$ für ein i, $2 \leq i \leq n$.

Induktionsbehauptung: Dann gilt $\hat{\ell}(e) \geq \ell(e)$ für alle $e \in \hat{E}_i$.

Beweis: Sei $e \in \hat{E}_i$.

- Fall 1: i ist gerade. Dann gibt es wegen der Konstruktion von $\hat{\mathcal{A}}$ ein $j \in \hat{E}_{i-1}$ mit $e \in C_j^2(\hat{x})$. Somit ist $j \in SV_2(\hat{x})$ und wegen der Voraussetzung des Lemmas gilt $j \in SV_2(x)$. Für j gilt die Induktionsvoraussetzung und damit $\ell(j) \leq \hat{\ell}(j)=i-1$.
  $C_j^2(\hat{x})$ enthält aber wegen (2.28.2b) und (2.26.2) nur Elemente g mit $\ell(g) \leq \ell(j)+1 \leq \hat{\ell}(j)+1=i$. Somit gilt $\ell(e) \leq i=\hat{\ell}(e)$.
- Fall 2: i ist ungerade. Dann gibt es wegen der Konstruktion von $\hat{\mathcal{A}}$ ein $h \in \hat{E}_{i-1}$ mit $h \in C_e^1(\hat{x})$. Für h gilt die Induktionsvoraussetzung und damit $\ell(h) \leq \hat{\ell}(h)=i-1$.
  Falls $e \notin SV_1(x)$, so gilt $\ell(e)=1<i=\hat{\ell}(e)$. Andernfalls enthält $C_e^1(\hat{x})$ wegen (2.28.1b) und (2.26.1) nur Elemente g mit $\ell(g) \geq \ell(e)-1$. Somit gilt $\ell(e)-1 \leq \ell(h) \leq \hat{\ell}(h)=i-1$ und damit $\ell(e) \leq i=\hat{\ell}(e)$.

□

2.30 Lemma: Sei $SV_2(\hat{x})=SV_2(x)$. Sei $e\in E_i\cap\hat{E}_i$ für ein Niveau i, $2\le i\le n$. Sei $g\in\hat{E}_{i-1}$. Dann gilt

1) Falls i ungerade und $g\in C_e^1(\hat{x})$, so ist $g\in E_{i-1}$.
2) Falls i gerade und $e\in C_g^2(\hat{x})$, so ist $g\in E_{i-1}$.

Beweis:

1) Da $i>1$, gilt $e\in SV_1(x)$. Wegen (2.28.1b) und (2.26.1) enthält $C_e^1(\hat{x})$ nur Elemente $d\in E$ mit $\ell(d)\ge i-1$. Wegen (2.29) gilt andererseits für alle d mit $\ell(d)>i-1$, dass $\hat{\ell}(d)>i-1$. Da $\hat{\ell}(g)=i-1$, gilt somit $\ell(g)=i-1$.

2) Da $g\in SV_2(\hat{x})$, gilt wegen der Voraussetzung $g\in SV_2(x)$. Wegen (2.28.1b) und (2.26.2) enthält $C_g^2(\hat{x})$ somit nur Elemente $d\in E$ mit $\ell(d)\le\ell(g)+1$. Wegen (2.29) gilt andererseits $\ell(g)\le\hat{\ell}(g)=i-1$. Somit gilt $i=\ell(e)\le\ell(g)+1\le\hat{\ell}(g)+1=i$, damit Gleichheit und somit $\ell(g)=i-1$.

□

2.31 Lemma: Sei $e\in E_i\cap\hat{E}_i$, $d\in E_{i-1}\cap\hat{E}_{i-1}$, $2\le i\le n$. Sei $SV_2(\hat{x})=SV_2(x)$.

1) i ungerade: Falls $q(g)>q(d)$ für alle $g\in E_{i-1}$, $g\in C_e^1(x)$, so gilt $q(g)>q(d)$ für alle $g\in\hat{E}_{i-1}$, $g\in C_e^1(\hat{x})$.
2) i gerade: Falls $q(g)>q(d)$ für alle $g\in E_{i-1}$, $e\in C_g^2(x)$, so gilt $q(g)>q(d)$ für alle $g\in\hat{E}_{i-1}$, $e\in C_g^2(\hat{x})$.

Beweis:

1) -Sei $g\in\hat{E}_{i-1}$, $g\in C_e^1(\hat{x})$. Wegen (2.30.1) gilt $g\in E_{i-1}$.
-Falls $v_{i-1}\notin C_e^1(x)$ oder $e=v_i$, so gilt wegen (2.28.1c) oder (2.28.1d) und wegen $g\in E_{i-1}$, dass $g\in C_e^1(\hat{x})$ nur wenn $g\in C_e^1(x)$. Dann gilt aber $q(g)>q(d)$ wegen der Voraussetzung.
-Falls $v_{i-1}\in C_e^1(x)$ und $e\neq v_i$, so gilt wegen (2.28.1d) und wegen $g\in E_{i-1}$, dass $g\in C_e^1(\hat{x})$ nur wenn $g\in C_e^1(x)$ oder $g\in C_{v_i}^1(x)$. Im ersten Fall gilt $q(g)>q(d)$ wegen der Voraussetzung und im zweiten Fall gilt $q(v_{i-1})<q(g)$, andernfalls wäre g in W und nicht v_{i-1} wegen der Wahl von v_{i-1} in Schritt 5 von Algorithmus (2.18). Aber wegen $v_{i-1}\in C_e^1(x)$ gilt $q(v_{i-1})>q(d)$ und somit $q(g)>q(v_{i-1})>q(d)$.

2) -Sei $g\in\hat{E}_{i-1}$, $e\in C_g^2(\hat{x})$. Wegen (2.30.2) gilt $g\in E_{i-1}$.
-Falls $v_i\notin C_g^2(x)$ oder $g=v_{i-1}$, dann gilt wegen (2.28.2c) oder (2.28.2d) und wegen $v_i\in E_i$, dass $e\in C_g^2(\hat{x})$ nur wenn $e\in C_g^2(x)$. Aber dann gilt $q(g)>q(d)$ wegen der Voraussetzung.
-Falls $v_i\in C_g^2(x)$ und $g\neq v_{i-1}$, dann gilt wegen (2.28.2d) und wegen $v_i\in E_i$, dass $e\in C_g^2(\hat{x})$ nur wenn $e\in C_g^2(x)$ oder $e\in C_{v_{i-1}}^2(x)$. Im ersten Fall gilt $q(g)>q(d)$ wegen der Voraussetzung und im zweiten Fall gilt $q(v_{i-1})<q(g)$, andernfalls wäre g in W und nicht v_{i-1} wegen der Wahl von v_{i-1} in Schritt 5 von Algorithmus (2.18). Aber wegen $e\in C_{v_{i-1}}^2(x)$ gilt $q(v_{i-1})>q(d)$ und somit $q(g)>q(v_{i-1})>q(d)$.

□

2.32 Lemma: Sei $e \in E$, $2 \leq i \leq n$. Solange der f_2-Spann des Inputvektors sich nicht verändert, ist e höchstens in n-i+1 aufeinanderfolg. Augmentationen kritisch auf dem Niveau i (vgl. (2.22))

Beweis:

Wir nehmen an, dass sich der f_2-Spann des Inputvektors nicht ändert und e auf dem Niveau i mehrmals erscheint. Wegen (2.29) ist dies nur in einer Serie von aufeinanderfolgenden Augmentationen der Fall.

Sei nun e das erste Mal kritisch auf dem Niveau i während einer bestimmten Augmentation. x sei der Input-Vektor und $\mathcal{A}$ sei die vergrössernde Struktur, W der vergrössernde Weg. Somit ist $e=v_i$. Sei nun $d:=v_{i-1}$. Es gilt, weil e kritisch auf dem Niveau i ist, dass,

- falls i ungerade, so $d \notin C_e^1(\hat{x})$. Wegen (2.28.1d) gilt dann für ein $g \in \hat{E}_{i-1}$, dass $g \in C_e^1(\hat{x})$ nur wenn $g \in C_e^1(x)$. Wegen (2.30.1) gilt auch $g \in E_{i-1}$ für so ein g, und somit gilt wegen der Wahl von d in Schritt 5 des Algorithmus (2.18), dass $q(g)>q(d)$ für alle $g \in \hat{E}_{i-1}$, $g \in C_e^1(\hat{x})$.
- falls i gerade, so $e \notin C_d^2(\hat{x})$. Für alle $g \in \hat{E}_{i-1}$ mit $e \in C_g^2(\hat{x})$ gilt dann wegen (2.30.2) $g \in E_{i-1}$. Wegen (2.28.2c) folgt, falls $e \notin C_g^2(x)$, sofort $e \notin C_g^2(\hat{x})$. Somit gilt $e \in C_g^2(x)$. Somit gilt wegen der Wahl von d in Schritt 5 von Algorithmus (2.18), dass $q(g)>q(d)$ für alle $g \in \hat{E}_{i-1}$, mit $e \in C_g^2(\hat{x})$.

Für i ungerade (bzw. i gerade) gilt nun durch wiederholte Anwendung von (2.31.1) (bzw. (2.31.2)), solange $e \in E_i$ und $d \in E_{i-1}$ in späteren (aufeinanderfolgenden) Augmentationen, dass $d \notin C_e^1(y)$ (bzw. $e \notin C_d^2(y)$), wobei y jeweils der Input-Vektor der betreffenden Augmentation ist. Wegen (2.29) kehrt d nie mehr auf das Niveau i-1 zurück, sobald es dieses einmal verlassen hat.

Falls e ein weiteres Mal kritisch auf dem Niveau i ist, dann wird das dem Element e in jenem vergrössernden Weg vorausgehende Element d' ein anderes Element als d sein. Für d' können wir dann die obigen für d angestellten Ueberlegungen wiederholen. Falls e ein drittes Mal kritisch auf dem Niveau i sein sollte, so muss das dem Element e vorausgehende Element d" derart sein, dass $d'' \notin \{d,d'\}$. Und so weiter.

Da die Niveaus 1,2,..,i-2 immer besetzt sein müssen und e auf dem Niveau i ist, hat e, während es auf dem Niveau e ist, höchstens n-i+1 verschiedene vorausgehende Elemente. Somit kann e höchstens n-i+1 Male kritisch auf dem Niveau i sein.

□

2.33 Lemma: Nach höchstens $\frac{n^4}{2} - \frac{n^3}{2} + 2n$ aufeinanderfolgenden Augmentationen liegt ein Vektor mit maximaler Komponentensumme vor.

Beweis:

- In Schritt 6 von Algorithmus (2.18) wird δ berechnet als die grösste Zahl, so dass $\hat{x}=x^{\delta}$ ein Vektor in $P_1 \cap P_2$ ist.
- Wegen der Konstruktion von $\mathcal{A}$ und $W=\{v_1,..,v_w\}$ gelten die folgenden Beziehungen (vgl. auch (2.17)):
  - $v_1 \notin SV_1(x)$, $v_w \notin SV_2(x)$, und für alle v_i, $2 \le i \le w$, gilt, falls i gerade, so $v_i \in C^2_{v_{i-1}}(x)$, falls i ungerade, so $v_{i-1} \in C^1_{v_i}(x)$.
- Falls nun die gleichen Beziehungen auch in $\hat{\mathcal{A}}$ bezüglich $\hat{x}$ für W gelten würden, so könnte $\hat{W} = W$ gewählt werden. Das heisst, dass δ nicht maximal war: wir hätten $\delta' = \delta+1$ wählen können und der Vektor $x^{\delta'}$ wäre immer noch ein Vektor in $P_1 \cap P_2$ gewesen.
- Aus diesen Ueberlegungen folgt, dass δ bestimmt wird durch folgende Situationen a) oder b):
  a): $v_1 \in SV_1(\hat{x})$ oder $v_w \in SV_2(\hat{x})$. v_1 bzw. v_w werden dann "anfangskritisch" bzw. "endkritisch" genannt. Wegen (2.28.1a) bzw. (2.28.2a) wird v_1 bzw. v_w in einer späteren Augmentation nie mehr anfangs- bzw. endkritisch sein. Fall a) tritt somit total höchstens 2n Male auf.
  b): Für ein i, $2 \le i \le w$, gilt, falls i gerade, $v_i \notin C^2_{v_{i-1}}(\hat{x})$, falls i ungerade, $v_{i-1} \notin C^1_{v_i}(\hat{x})$. Das heisst, v_i ist kritisch auf dem Niveau i. Solange sich der f_2-Spann des Inputvektors nicht ändert, ist wegen (2.32) v_i höchstens in n-i+1 aufeinanderfolgenden Augmentationen kritisch auf dem Niveau i, somit alle Elemente höchstens $n.\sum\{n-i+1: i=2,..,n\} = n^2.(n-1)/2$ Male kritisch auf irgendeinem Niveau (beachte: $w \le n$). Wegen (2.28.2a) kann sich der f_2-Spann des Inputvektors nur vergrössern, und zwar höchstens n Male. Somit sind alle Elemente insgesamt höchstens $n^3.(n-1)/2$ Male kritisch auf irgendeinem Niveau i, $2 \le i \le n$.
- Für jede Augmentation gibt es somit ein Element e (aus dem betr. W) das kritisch auf einem Niveau i, $2 \le i \le n$ (Fall b)) oder anfangs- oder endkritisch ist (Fall a)). Somit gibt es höchstens $n^3.(n-1)/2 + 2.n$ Augmentationen insgesamt.

□

3 Vektoren mit maximaler Gewichtssumme in ganzzahligen Polymatroid-Intersektions-Polyedern

3.1 Gegeben die gleichen Definitionen wie in (2.1).

Zusätzlich ist ein Gewichtsvektor c definiert, $c=[c_e: e\in E,\ c_e\in \mathbb{R}]$.

Sei $S_x := \{e\in E: x_e>0\}$.

Für ein $T\in L_E$ sei $L_T := \{S\in L_E,\ S\subseteq T\}$, $K_T := L_T-\emptyset$; $f|T : L_T \to \mathbb{R}$ ist die Funktion f 'restringiert' auf T: $(f|T)(S):=f(S)$ für alle $S\subseteq T$.

O.B.d.A. gilt $f_k(e)>0$ für alle $e\in E$, k=1,2, (vgl. (1.28)).

3.2 Das "gewichtete Polymatroid-Intersektions-Problem" ist nun, c.x über $P(K_E,f_1)\cap P(K_E,f_2)$ zu maximieren, oder mit anderen Worten:

Maximiere c.x , wobei $x\in \mathbb{R}^E$ derart, dass

$$\begin{aligned} x_e &\geq 0 && \text{für alle } e\in E\\ x(S) &\leq f_1(S) && \text{für alle } S\in K_E\\ x(S) &\leq f_2(S) && \text{für alle } S\in K_E \end{aligned}$$.

3.3 Das duale lineare Programm von (3.2) ist

Minimiere $f_1.y^1+f_2.y^2$, wobei $y^1,y^2\in \mathbb{R}^{K_E}$ und

$$\begin{aligned} y_k(S) &\geq 0 && \text{für alle } S\in K_E,\ k=1,2\\ y^1(K_E,e)+y^2(K_E,e) &\geq c_e && \text{für alle } e\in E \end{aligned}$$,

wobei für k=1,2,

$$y^k := [y_k(S): S\in K_E]$$

$$y^k(K_E,e) := \sum\{y_k(S): e\in S\in K_E\}$$.

3.4 In [E1] finden wir folgende Sätze:

- Falls f_1 und f_2 ganzzahlig sind, dann kann das maximierende x in (3.2) ganzzahlig gewählt werden.
- Falls c ganzzahlig ist, dann kann das minimierende y^1 und y^2 in (3.3) ganzzahlig gewählt werden.

3.5 In diesem Kapitel präsentieren wir nun einen Algorithmus zur Lösung von (3.2).
Wir nehmen dabei natürlich an, dass c auf einem digitalen Computer codierbar ist. Aufgrund der in (1.22) gemachten Betrachtungen werden wir überdies für alle elementaren Computeroperationen, auch in Verbindung mit dem Vektor c, eine einheitliche Rechenzeit annehmen. Ausser der in dieser Einheitszeitannahme versteckten polynomialen Abhängigkeit von c wird die Komplexität des Algorithmus unabhängig von der Wahl von c sein.

Der Algorithmus benützt den Algorithmus (2.20) als Unteralgorithmus. Seine Komplexität ist somit auch abhängig vom Rechenaufwand für die Subroutinen zur Entscheidung, ob ein gegebener Vektor $x \in \mathbb{R}^E$ ein Vektor $x \in P_k$ ist oder nicht, für k=1,2 (vgl. (2.10)). Der Algorithmus beruht auf einem von Edmonds in [E2] gegebenen Algorithmus und benützt ebenso die Arbeit von Giles [G1] und die Arbeit des Autors [S1].

Wie durch (3.4) schon vorgezeichnet, werden wir, da f_1, f_2 ganzzahlig sind, unabhängig von der Wahl von c einen ganzzahligen optimierenden Vektor x finden.

3.6 Proposition (Giles in [G1]):

Sei $T \subseteq E$.
Dann gibt es eine eindeutige maximale Menge $S \subseteq E$, so dass $T \subseteq S$ und $f(T) = f(S)$.
(Diese Aussage gilt auch für nicht-ganzzahlige ε_0-Funktionen).

□

3.7 Definitionen: (gelten auch für nichtganzzahlige β_0-Funktionen)

Sei $T \subseteq E$. Nenne die eindeutige maximale Teilmenge $S \subseteq E$ so dass $T \subseteq S$ und $f(T)=f(S)$ 'f-Spann der Menge T', $S = SP(T)$.
Wir nennen T f-abgeschlossen, falls für alle $S \subseteq E$ mit $T \subset S$ gilt, dass $f(T)<f(S)$. Damit ist T f-abgeschlossen genau dann wenn $T=SP(T)$.

3.8 Proposition:

Gegeben $x \in P$, x ganzzahlig. Dann ist SV(x) f-abgeschlossen.
(Bemerkung: Es kann durchaus ein $x' \in P$, $x' \neq x$, x' ganzzahlig geben, so dass $SV(x)=SV(x')$).

Beweis:

Wegen (2.6) sind alle Elemente von SV(x) Elemente der eindeutigen maximalen f-kritischen Menge T bezüglich x.
Sei nun $e \in SP(T)-T$. Dann gilt $f(T)=x(T) \leq x(T \cup e) \leq f(T \cup e)=f(T)$, somit gilt Gleichheit und $T \cup e$ ist auch f-kritisch bezüglich x. Dies widerspricht der Maximalität von T.

□

3.9 Satz (Giles [G1]):

Sei $T \subseteq E$ eine feste Teilmenge von E, $\overline{T} := E-T$. Sei $fxT: L_T \to \mathbb{R}$, eine Funktion, genannt Kontraktion von f auf T, definiert wie folgt:

$$(fxT)(S) := f(\overline{T} \cup S) - f(\overline{T}) \quad , \quad \text{für alle } S \subseteq T.$$

Dann ist fxT eine β_0-Funktion und für alle $x^0 \in \mathbb{R}_+^T$ gilt

$$x^0 \in P(K_T, fxT) \iff [x^0, x^1] \in P(K_E, f) \text{ für alle } x^1 \in P(K_{\overline{T}}, f|\overline{T}) \quad .$$

(Dieser Satz gilt auch für nichtganzzahlige β_0-Funktionen).

□

3.10 Definitionen: (gelten auch für nicht-ganzzahlige β_0-Funktionen)

Eine Menge $T \in K_E$ wird f-separabel genannt, falls es eine Menge $S \in K_T - \{T\}$ gibt, so dass $f(S)+f(T-S)=f(T)$. Andernfalls wird T f-nichtseparabel genannt. Eine f-Separation von T ist eine Partition F von T in nichtleere Teilmengen mit $f(T)=\sum\{f(S): S \in F\}$. Eine f-Separation F von T wird eine minimale f-Separation genannt, falls jede Menge $S \in F$ f-nichtseparabel ist.

3.11 Lemma: Jede Menge $T \in K_E$ hat eine eindeutige minimale f-Separation.

Dieses Lemma gilt auch für nichtganzzahlige $\mathfrak{S}_0$-Funktionen. Der Beweis folgt demjenigen von Giles [Gi]: Zuerst werden drei Propositionen bewiesen und dann das Lemma.

a) Sei F eine f-Separation von B, $B \in K_E$. Dann gilt für alle $F' \subseteq F$ dass F' eine f-Separation von $\cup\{S: S \in F'\}$ ist, d.h. $f(\cup\{S: S \in F'\}) = \sum \{f(S): S \in F'\}$.

Beweis: Es genügt für alle $T \in F$ zu zeigen, dass $f(\cup\{S: S \in F-\{T\}\}) = \sum \{f(S): S \in F-\{T\}\}$. Wegen der Submodularität von f gilt:

$$\begin{aligned} f(\cup\{S: S \in F-\{T\}\}) + f(T) &\geq f(B) \\ &= \sum \{f(S): S \in F\} \\ &= \sum \{f(S): S \in F-\{T\}\} + f(T) \\ &\geq f(\cup\{S: S \in F-\{T\}\}) + f(T) . \end{aligned}$$

Somit gilt Gleichheit und damit folgt die Proposition. □

b) Falls $f(B) = f(T)+f(\overline{T})$ für ein $T \subseteq B \in K_E$, $\overline{T} := B-T$, dann gilt für alle $S \subseteq B$, dass $f(S) = f(S \cap T)+f(S \cap \overline{T})$.

Beweis: Es gilt folgende Beziehung:

$$\begin{aligned} f(B) &= f(T) + f(\overline{T}) \\ &\geq f(S \cap T) + f(S \cup T) - f(S) + f(S \cap \overline{T}) + f(S \cup \overline{T}) - f(S) \\ &= [f(S \cap T) + f(S \cap \overline{T}) - f(S)] + f(S \cup T) + f(S \cup \overline{T}) - f(S) \\ &\geq f(S \cup T) + f(S \cup \overline{T}) - f(S) \\ &\geq f(B) . \end{aligned}$$

Somit gilt Gleichheit und damit $f(S) = f(S \cap T)+f(S \cap \overline{T})$.

□

c) Jede Menge $T \in K_E$ hat eine minimale f-Separation.

Beweis: Nimm an, dass F eine f-Separation von T ist und dass für ein $R \in F$ eine Menge $V \in K_R - \{R\}$ existiert, so dass $f(R)=f(V)+f(R-V)$. Dann ist $(F-\{R\}) \cup \{V, R-V\}$ eine f-Separation von T, da $f(T)=\sum\{f(S): S \in F-\{R\}\}+f(V)+f(R-V)$ gilt. Somit hat T eine minimale f-Separation.

□

Beweis des Lemmas:

Es genügt zu zeigen, dass E eine minimale f-Separation hat. Nimm an, dass F und H zwei verschiedene f-Separationen von E sind. Da F und H verschiedene Partitionen von E sind, so gibt es ein $S \in F$ und ein $T \in H$ so dass $S \cap T \neq \emptyset$ und $S \neq T$. Wir dürfen annehmen, dass $S \not\subseteq T$. Dann gilt $S \cap (E-T) \neq \emptyset$. Wegen a) gilt $f(E)=f(T)+f(E-T)$. Wegen b) gilt dann $f(S) = f(S \cap T)+f(S \cap (E-T))$. Aber dann ist S f-separabel und somit F keine minimale f-Separation.

□

3.12 Lemma:

Sei die leere Menge f-abgeschlossen (d.h. $f(e)>0$ für alle $e \in E$). Sei $B \in K_E$ eine f-abgeschlossene Teilmenge von E und F eine f-Separation von B.
Dann ist jede Menge $T \in F$ f-abgeschlossen.

(Dieses Lemma gilt auch für nicht-ganzzahlige B_0-Funktionen.)

Beweis:

Sei $\overline{T} := B-T$. Wegen (3.11.a) gilt $f(B)=f(T)+f(\overline{T})$.

Weil B f-abgeschlossen ist, gilt $SP(T) \subseteq B$, $SP(\overline{T}) \subseteq B$ und $SP(T) \cup SP(\overline{T})=B$.

Deswegen und wegen der Submodularität von f gilt

$$f(B) \leq f(B)+f(SP(T) \cap SP(\overline{T})) \leq f(SP(T))+f(SP(\overline{T})) = f(T)+f(\overline{T}) = f(B)$$

Somit gilt Gleichheit und damit $f(SP(T) \cap SP(\overline{T})) = 0$.

Da die leere Menge f-abgeschlossen ist, gilt $SP(T) \cap SP(\overline{T}) = \emptyset$.

Somit gilt $T = SP(T)$ und $\overline{T} = SP(\overline{T})$: Falls $e \in SP(T)-T$, so $e \in \overline{T}$, somit $e \in SP(\overline{T})$ und damit $e \in SP(T) \cap SP(\overline{T})$, aber $SP(T) \cap SP(\overline{T}) = \emptyset$.

□

3.13 Lemma: Sei $x \in P$ und $B \in K_E$, $x(B)=f(B)$. Sei F die (eindeutige) minimale f-Separation von B. Dann gilt:

a) $x(T) = f(T)$ für alle $T \in F$.

b) Falls $e \in B$, dann gibt es eine eindeutige Menge $T \in F$, so dass $C_e(x) \subseteq T$.

Beweis:

a): $x(B) = \sum\{x(T): T \in F\} \leq \sum\{f(T): T \in F\} = f(B) = x(B)$.
Somit gilt Gleichheit. Wegen $x \in P$ gilt aber $x(T) \leq f(T)$ für alle $T \in F$ und damit wegen obiger Ueberlegung $x(T)=f(T)$ für alle $T \in F$.

b): Wegen (2.7) gilt für $e \in B$ sofort $C_e(x) \subseteq B$.

Nimm nun an, dass es ein $T \in B$ gibt mit $U := C_e(x) \cap T$, $V := C_e(x) \cap (B-T)$, $U, V \neq \emptyset$. Somit gilt $e \in U$ oder $e \in V$.
Wegen (3.11.a) gilt $f(B)=f(T)+f(B-T)$.
Wegen (3.11.b) gilt, wo $C_e(x)$ die Menge S in (3.11.b) ist, dass $f(C_e(x)) = f(U) + f(V)$.

Deswegen und weil $x(C_e(x))=f(C_e(x))$ und weil $x \in P$, gilt mit der gleichen Ueberlegung wie in a), dass $x(U)=f(U)$ und $x(V)=f(V)$.
Da $e \in U$ oder $e \in V$, widerspricht dies der Minimalität von $C_e(x)$.
□

Bemerkung: Da alle f-Separationen, die wir berechnen werden, f-Separationen von Mengen $B \in K_E$ mit $x(B)=f(B)$ sind (für den Vektor x, welcher jeweils aktuell ist), zeigt uns b), wie wir die minimale f-Separation F einer dieser Mengen B berechnen:

- Wir initialisieren die 'feinste' mögliche Partition von B, nämlich diejenige der einelementigen Mengen.
- Da nun zwei Elemente $e,e' \in B$ nur dann in zwei verschiedenen Mengen $T \in F$ sein können, wenn $C_e(x) \cap C_{e'}(x)=\emptyset$, führen wir für alle $e \in B$ folgende Operationen durch:
  1 Bestimme $C_e(x)$.
  2 Revidiere die bisher gebildete Partition, indem alle Mengen der Partition, die mit $C_e(x)$ einen nicht-leeren Durchschnitt haben, in eine Menge zusammengefasst werden.

Ein solcher Algorithmus benötigt $O(|B|^2)$ Rechenschritte, relativ zum Rechenaufwand der Subroutine (2.10).

3.14 Familien von verschachtelten Teilmengen von E

a) Sei $\mathcal{A}$ eine Menge von nichtleeren verschiedenen Teilmengen von E.

Wir nennen $\mathcal{A}$ eine Sequenz von verschachtelten Mengen, falls für $A,B \in \mathcal{A}$ gilt: $A \cap B \in \{A,B\}$.

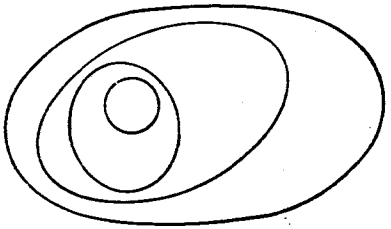


Fig. 3.1: Sequenz von verschachtelten Mengen

b) Sei $\mathfrak{R}$ eine Menge von nichtleeren verschiedenen Teilmengen von E.

Wir nennen $\mathfrak{R}$ eine Familie von verschachtelten Mengen, falls für $A,B \in \mathfrak{R}$ gilt: $A \cap B \in \{\emptyset,A,B\}$.

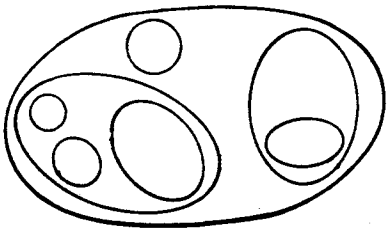
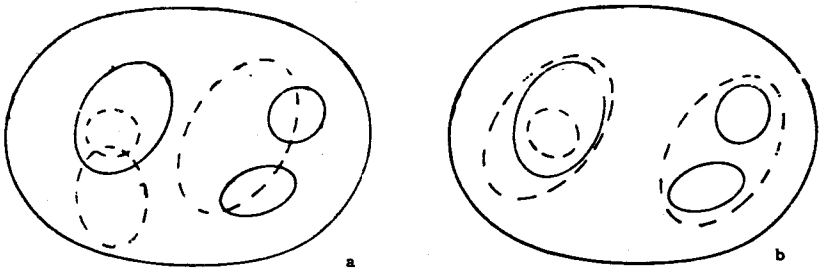


Fig. 3.2: Familie von verschachtelten Mengen

Natürlich ist jede Seqenz von verschachtelten Mengen ein Spezialfall einer Familie von verschachtelten Mengen.

c) Im Verlaufe des Algorithmus werden wir zwei solche Familien von verschachtelten Mengen konstruieren, welche laufend verändert werden durch Hinzufügen neuer Mengen und Streichungen von Mengen. Wir werden zu beweisen haben, dass neue Mengen wirklich in die Familie eingfügt werden können, d.h. dass wir nicht eine Situation wie in

Figur 3.3.a haben sondern eine solche wie in Figur 3.3.b. In der Figur 3.3 sind die bereits bestehenden Mengen der Familie mit durchgehenden Linien gezeichnet, die neu hinzustossenden Mengen mit gestrichenen Linien.



Figur 3.3: Revision der Familien von verschachtelten Mengen

d) Sei $\mathfrak{R}^E := \mathfrak{R} \cup \{E\}$. Für ein $A \in \mathfrak{R}^E$ sei $V_A := \emptyset \cup \{B: B \in \mathfrak{R}, B \subset A\}$ und für ein $A \in \mathfrak{R}$ sei, falls $A \neq E$, $T_A := \cap\{B: B \in \mathfrak{R}^E, B \supset A\}$, falls $A=E$, $T_A := E$. Es gibt somit mindestens eine Menge $A \in \mathfrak{R}^E$ mit $V_A = \emptyset$ und eine Menge $B \in \mathfrak{R}$ mit $T_B = E$. Es gilt für alle $A \in \mathfrak{R}$, $V_A \subset A$ und, falls $A \neq E$, $A \subset T_A$.

e) Die Familien, welche wir im Verlaufe des Algorithmus konstruieren werden, bestehen aus f-abgeschlossenen, f-nichtseparablen Mengen, die überdies bezüglich des jeweils geltenden Primalvektors f-kritisch sind. Auch gilt $f(e)>0$ für alle $e \in E$. Für eine solche Familie $\mathfrak{R}$ gelten nun für alle $A \in \mathfrak{R}$ folgende Beziehungen:
   - $f(V_A) < f(A)$ und, falls $A \neq E$, $f(A) < f(T_A)$:
     Sicher gilt $f(V_A) \leq f(A) \leq f(T_A)$, da f nichtabnehmend ist. Aus $f(A)=f(V_A)$ oder $f(A)=f(T_A)$ folgt aber sofort $A=V_A$ oder $A=T_A$, da die Mengen V_A, A, T_A f-abgeschlossen sind.
   - $x(V_A) < x(A)$ und, falls $A \neq E$, $x(A) < x(T_A)$:
     Dies folgt sofort aus obiger Beziehung, weil $x(A)=f(A)$ für alle $A \in \mathfrak{R} \cup \{\emptyset\}$ gilt
   - Somit gibt es, falls $V_A \subset A \subset T_A$, zwei Elemente j,h, $j \in A - V_A$, $h \in T_A - A$ mit $j,h \in S_x = \{e: x_e > 0\}$.

f) Im Algorithmus werden die zwei Familien von verschachtelten Mengen $\mathfrak{R}_1$ und $\mathfrak{R}_2$ genannt. Für $k=1,2$ sind dann $\mathfrak{R}_k^E$, V_A^k, T_A^k die den Mengen $\mathfrak{R}^E$, V_A, T_A in $\mathfrak{R}$ entsprechenden Mengen.

3.15 Primales LP:

Der Algorithmus wird folgendes lineare Programm lösen:

Maximiere $c.x$, wobei $x \in \mathbb{R}^E$ derart, dass

$$
\begin{aligned}
x_e &\geq 0 && \text{für alle } e \in E \\
x(S) &\leq f_1(S) && \text{für alle } S \in K_E,\ S\ f_k\text{-abgeschlossen} \\
x(S) &\leq f_2(S) && \text{und } f_k\text{-nichtseparabel, für } k=1,2. \\
x(E) &\leq m && m \in \mathbb{N} .
\end{aligned}
$$

Der hier definierte Polyeder ist derselbe wie der Polyeder, der in (3.2) definiert wurde:

- Jede Ungleichung $x(S) \leq f_k(S)$, wo S nicht nicht f_k-abgeschlossen ist, wird 'dominiert' durch die Ungleichung $x(SP_k(S)) \leq f_k(SP_k(S))$, wobei $SP_k(S)$ den f_k-Spann der Menge S bedeutet, für k=1,2. Somit kann sie weggelassen werden:
  $x(S) \leq x(SP_k(S)) \leq f_k(SP_k(S)) = f_k(S)$.

- Falls F eine f_k-Separation von S ist (für Mengen S, die f_k-separabel sind), dann kann die Ungleichung $x(S) \leq f_k(S)$ ebenfalls weggelassen werden. Sie wird dann dominiert durch die Ungelichungen $x(S') \leq f_k(S')$, $S' \in F$:
  $x(S) = \sum\{x(S'): S' \in F\} \leq \sum\{f_k(S'): S' \in F\} = f_k(S)$.

- Die Ungleichung mit m als rechter Seite wurde eingeführt, um auf einfache Art eine duale Initiallösung zu erzeugen. Wählt man m gross genug, z.B. m=n.K, so ist sie redundant.

3.16 Duales LP:

Für k=1,2 sei

$$y^k := [y_k(S) : S \in K_E,\ S\ f_k\text{-abgeschlossen und } f_k\text{-nichtseparabel}]$$

$$t_k(y,e) := \sum\{y_k(S): e \in S \in K_E,\ S\ f_k\text{-abgeschlossen und } f_k\text{-nicht-separabel}\}$$

Sei überdies

$$y := [y_m, y^1, y^2]$$

$$t(y,e) := y_m + t_1(y,e) + t_2(y,e) \quad .$$

Dann ist das duale LP zu (3.15):

$$\text{Minimiere } f.y := m.y_m + f_1.y^1 + f_2.y^2 \text{ , wobei}$$

$$y \geq 0$$

$$t(y,e) \geq c_e \quad \text{für alle } e \in E \quad .$$

3.17 Bedingungen für die Optimalität

Eine Möglichkeit zu zeigen, dass ein x, welches eine Lösung von (3.15) ist, $c.x$ maximiert, ist die Angabe einer Lösung y von (3.16), so dass für (x,y) die 'Komplementärschlupfbedingungen' erfüllt sind:

a) $x_e > 0 \Rightarrow t(y,e) = c_e$.

b) $y_k(S) > 0 \Rightarrow x(S) = f_k(S)$, d.h. x ist eine f_k-Basis von S.

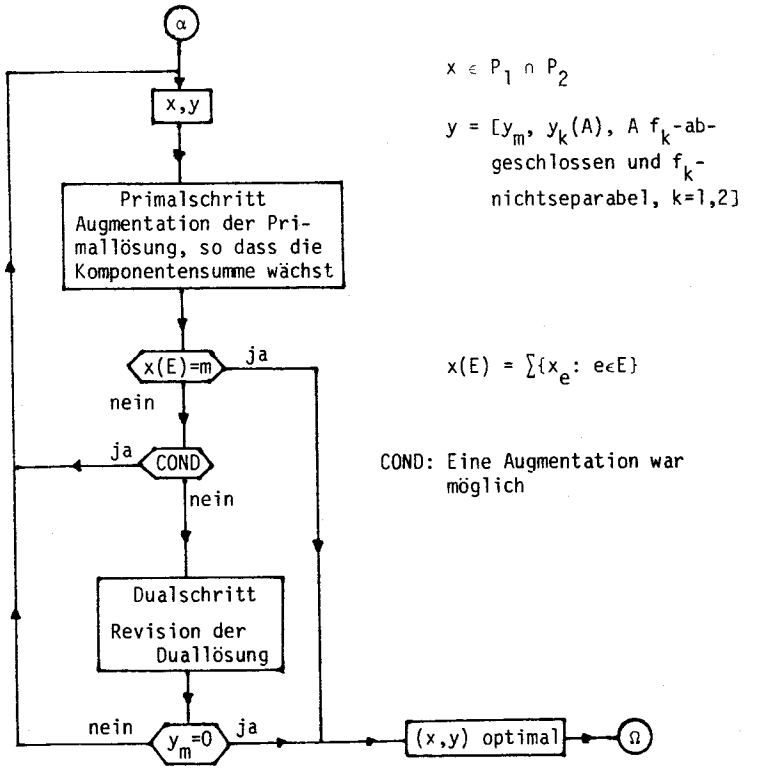
c) $y_m > 0 \Rightarrow x(E) = m$.

Daraus folgt nach einem bekannten Satz der LP-Theorie (vgl. dazu [G1]), dass $c.x = f.y$ und damit die Optimalität des Paares (x,y).

3.18 Algorithmus, generelles Konzept

Der Algorithmus wird mit einer primal und dual zulässigen Initiallösung starten, welche (3.17.a) und (3.17.b), aber im allgemeinen nicht (3.17.c) erfüllt. Er terminiert, sobald (3.17.c) erfüllt ist.

Das folgende Flussdiagramm illustriert dies:



Figur 3.4: Primal-Dualer Polymatroid-Intersektions-Algorithmus. generelles Konzept.

3.19 ALGORITHMUS

Gegeben eine endliche Menge E. Für k=1,2 sei $P_k \subseteq \mathbb{R}^E$ ein ganzzahliges Polymatroid, f_k seine entsprechende ganzzahlige β_0-Funktion. O.B.d.A gilt $f_k(e)>0$ für alle $e\in E$. Gegeben $c=[c_e: e\in E, c_e\in\mathbb{R}]$. Finde einen Vektor x aus der Menge aller ganzzahligen Vektoren z, $z\in P_1\cap P_2$, $z(E)\leq m$, so dass c.x maximal ist.

0 Initialisierung

$x:=0$, $y_k(A):=0$ für alle $A\subseteq E$, $k=1,2$. $y_m := \max\{0, c_e : e\in E\}$.

1 Primalschritt

Input: (x,y), primal und dual zulässig, so dass

- $y_k(A)>0 \iff A\in \mathfrak{R}_k$, $k=1,2$, wobei $\mathfrak{R}_k$ eine Familie von verschachtelten, f_k-abgeschlossenen, f_k-nichtseparablen, verschiedenen und nichtleeren Teilmengen von E ist.
- $S_x = \{e: x_e>0\} \subseteq E^0 := \{e\in E: t(y,e)=c_e\}$
- $x(A)=f_k(A)$ für alle $A\in \mathfrak{R}_k$, d.h. x ist eine f_k-Basis von A.

1.1 Für alle $A\in \mathfrak{R}_k^E$, $k=1,2$, sei $f_k^A := (f_k|A) \times (A-V_A^k)$ die Kontraktion von $f_k|A$ auf $A-V_A^k$, und sei P_k^A das Polymatroid definiert durch $P_k^A := P(K_{(A-V_A^k)}, f_k^A)$.

Für k=1,2, sei $P_k' := \bigcap\{P_k^A: A\in \mathfrak{R}_k^E\}$ und $f_k': L_E \to \mathbb{R}$ die entsprechende β_0-Funktion, $f_k'(D) = \sum\{f_k^A(D\cap(A-V_A^k)): A\in \mathfrak{R}_k^E\}$, $D\in L_E$.

Für k=1,2, sei $f_k^0 := f_k'|E^0$ und $P_k^0 := P(K_{E^0}, f_k^0)$.

1.2 Wende wiederholt Algorithmus (2.18) bezüglich x und P_k^0, k=1,2 an, bis

- entweder $\hat{x}(E^0)=\hat{x}(E)=m$ (falls $\hat{x}(E)>m$, würden wir das letzte Inkrement δ in Schritt 6 von Algorithmus (2.18) entsprechend reduzieren). Stop. $(\hat{x},y)$ ist ein optimales Paar von Lösungen.
- oder $\hat{x}(E)<m$, aber $\hat{x}(E^0)=\hat{x}(E)$ maximal in $P_1^0 \cap P_2^0$ ist. Es gibt dann eine Menge S_m in Algorithmus (2.18). Definiere $S:=S_m$, $x:=\hat{x}$ und gehe zu Schritt 2.

2 Dualschritt

2.1 Sei $S^1 := SV_1'(x|E-S)$ und $S^2 := SV_2'(x|S)$, wobei für k=1,2, $SV_k'(x)$ die maximale f_k'-kritische Menge bezüglich x bezeichnet.

Für k=1,2 und alle $A \in \mathfrak{R}_k$ finde $B_A^k := (S^k \cap A) \cup V_A^k$,

finde F_A^k, die minimale f_k-Separation von B_A^k, und

füge alle Mengen $B \in F_A^k$, $A \in \mathfrak{R}_k$ entsprechend in die Familie $\mathfrak{R}_k$ dabei die Familie $\bar{\mathfrak{R}}_k$ bildend: $\bar{\mathfrak{R}}_k := \mathfrak{R}_k \cup \{B \in F_A^k : A \in \mathfrak{R}_k\}$.

2.2 Modifiziere die Dualvariablen:

- $\hat{y}_m := y_m - \delta$
- $\hat{y}_k(D) := y_k(D) + \delta . N_k(D)$, für alle $D \subseteq E$, k=1,2, wobei
- $N_k(D)$ die Anzahl der Mengen $B \in \bar{\mathfrak{R}}_k - \mathfrak{R}_k$, so dass B=D, minus die Anzahl der Mengen $A \in \mathfrak{R}_k$, so dass A=D.

$\delta := \min\{ y_m ; t(y,e)-c_e : e \in E-(S^1 \cup S^2) ; y_k(A) : N_k(A)=-1, A \subseteq E\}$

Für k=1,2, sei $\mathfrak{R}_k := \hat{\mathfrak{R}}_k$, wobei $\hat{\mathfrak{R}}_k$ aus $\bar{\mathfrak{R}}_k$ erhalten wird durch Streichung aller Mengen A mit $\hat{y}(A)=0$ und durch Identifizierung aller identischen Mengen.

Definiere $y := \hat{y}$.

2.3 Falls $y_m=0$, dann stop: (x,y) ist ein optimales Paar von Lösungen.

Andernfalls gehe zu Schritt 1.

3.20 Validität des Algorithmus

Wir werden beweisen, dass für jeden Input des Primalschritts des Algorithmus (3.19) folgende Beziehungen gelten:

a) $x \in P_1 \cap P_2$, also x primal zulässig, um (3.15) und damit (3.2) zu erfüllen.

b) y ist dual zulässig, um (3.16) zu erfüllen

c) $y_k(A)>0 \iff A \in \mathfrak{R}_k$, k=1,2, wobei $\mathfrak{R}_k$ eine Familie von verschachtelten, f_k-abgeschlossenen und f_k-nichtseparablen, verschiedenen und nichtleeren Teilmengen von E ist.

d) $S_x = \{e: x_e>0\} \subseteq E^0 = \{e \in E, t(y,e)=c_e\}$ (um (3.17.a) zu erfüllen)

e) $x(A)=f_k(A)$ für alle $A \in \mathfrak{R}_k$, k=1,2, d.h. x ist eine f_k-Basis von A (um (3.17.b) zu erfüllen).

Dies wird sicher bewiesen, indem wir zeigen, dass

- die Initiallösung a) bis e) erfüllt (3.21)
- nach Schritt 1.2 des Algorithmus a), d), e) für den neuen Vektor $\hat{x}$ erfüllt sind (3.22)
- alle Mengen $B \in F_A^k$, $A \in \mathfrak{R}_k$, konstruiert in Schritt 2.1 des Algorithmus, f_k-abgeschlossene, f_k-nichtseparable Teilmengen von E sind, welche überdies f_k-kritisch bezüglich des geltenden Primalvektors x sind (3.23)
- alle Mengen $B \in F_A^k$, $A \in \mathfrak{R}_k$, wirklich in die Familie $\mathfrak{R}_k$ eingefügt werden können (d.h. $\hat{\mathfrak{R}}_k$ ist wieder eine Familie von verschachtelten Mengen) (3.24)
- nach der Revision der Duallösung in Schritt 2.2 der neue Vektor $\hat{y}$ dual zulässig ist (3.25)
- $S_x \subseteq \hat{E}^0 := \{e \in E: t(\hat{y},e)=c_e\}$ gilt (3.26)

3.21 Proposition: Die Initiallösung in Algorithmus (3.19) erfüllt (3.20.a) bis (3.20.e)

Beweis: Dies ist sicher der Fall wegen $x:=0$ und $y_k(A)=0$ für alle $A\subseteq E$, $k=1,2$ und wegen $y_m := \max\{0,c_e : e\in E\}$. □

3.22 Lemma: Nach Schritt 1.2 von Algorithmus (3.19) gilt $\hat{x}\in P_1\cap P_2$ und (3.20.d) und (3.20.e) sind bezüglich $\hat{x}$ erfüllt.

Beweis:

Wegen der Konstruktion von P_k^0 aus P_k und weil x eine f_k-Basis von A für alle $A\in\mathfrak{A}_k$, $k=1,2$ ist, gilt

$$x\in P_k^0 \Rightarrow x\in P_k' \Leftrightarrow x|A-V_A^k \in P_k^A \text{ für alle } A\in\mathfrak{A}_k^E \Leftrightarrow x\in P_k .$$

Der Algorithmus (2.18) wird nun wiederholt auf x und P_k^0, $k=1,2$ angewandt, er terminiert mit einem Vektor $\hat{x}\in P_1^0\cap P_2^0$. Somit gilt sofort $\hat{x}\in P_1\cap P_2$.

(3.20.d) ist bezüglich $\hat{x}$ erfüllt, weil die Augmentationen auf P_k^0, $k=1,2$, durchgeführt wurden.

Sei nun $A\in\mathfrak{A}_k$, für $k=1$ oder $k=2$. Somit $x(A)=f_k(A)$. Wegen Konstruktion von P_k' aus P_k und wegen (3.9) gilt

$$f_k'(A) = \sum\{f_k^A(D-V_D^k) : D\in\mathfrak{A}_k,\ D\subseteq A\} = \sum\{f(D)-f(V_D^k) : D\in\mathfrak{A}_k,\ D\subseteq A\} = f(A) .$$

Somit gilt auch $x(A)=f_k'(A)$. Seien nun $\mathcal{A}$ und $W=\{v_1,..,v_w\}$ die während eines Durchlaufs von Algorithmus aufgebaute vergrössernde Struktur und der vergrössernde Weg. Da die Augmentation in P_k^0 durchgeführt wird, gilt für jedes $v_i\in A$, i gerade, dass, falls $k=1$, so $v_{i+1}\in A$, falls $k=2$, so $v_{i-1}\in A$. Damit ist $f_k'(A)\geq\hat{x}(A)\geq x(A)=f_k(A)=f_k'(A)$. Damit gilt Gleichheit. Durch wiederholte Anwendung dieser Ueberlegungen gilt somit (3.20.e) bezüglich $\hat{x}$.

□

3.23 Lemma: Alle Mengen $B \in F_A^k$, $A \in \mathfrak{A}_k^E$, k=1,2, welche in Schritt 2.1 des Algorithmus (3.19) konstruiert werden, sind f_k-abgeschlossen, f_k-nichtseparabel und f_k-kritisch bezüglich x.

Beweis:

Wegen der Konstruktion der Mengen B, wegen (3.12) und (3.13.a) genügt es, für jede Menge B_A^k, $A \in \mathfrak{A}_k^E$, k=1,2 zu zeigen, dass B_A^k f_k-abgeschlossen ist, d.h. $B_A^k = (S^k \cap A) \cup V_A^k = SP_k((S^k \cap A) \cup V_A^k)$, und zu zeigen, dass $x(B_A^k) = f_k(B_A^k)$ gilt:

Sei für k=1,2 und alle $A \in \mathfrak{A}_k^E$, $S_A^k := S^k \cap (A - V_A^k)$. Wegen der Konstruktion von P_k' aus P_k gilt, dass f_k'-Spannbildung des Vektors x und Schnitt mit $A - V_A^k$ vertauscht werden können, d.h. es gilt

- $S_A^1 = SV_1'(x|(E-S) \cap (A-V_A^1)) = SV_k^A(x|(E-S) \cap (A-V_A^1))$ und
- $S_A^2 = SV_2'(x|S \cap (A-V_A^2)) = SV_k^A(x|S \cap (A-V_A^k))$, wobei für k=1,2, $SV_k^A(x)$ der f_k^A-Spann des Vektors $x|A-V_A^k$ ist. Damit gilt $x(S_A^k) = f_k^A(S_A^k)$.

Wegen Konstruktion von S_A^k gilt nun $B_A^k = S_A^k \oplus V_A^k$. Damit gilt wegen der Konstruktion von P_k' aus P_k und wegen (3.9):

$$\underline{f_k(B_A^k)} = f_k^A(S_A^k) + f_k(V_A^k) = x(S_A^k) + x(V_A^k) = \underline{x(B_A^k)} \quad .$$

Sei nun $f_k(B_A^k \cup e) = f_k(B_A^k)$ für ein $e \in E$. Wenn wir zeigen können, dass $e \in B_A^k$, dann ist B_A^k f_k-abgeschlossen. Es gilt nun:

- $f_k(B_A^k) = x(B_A^k) \leq x(B_A^k \cup e) \leq f_k(B_A^k \cup e) = f_k(B_A^k)$, somit Gleicheit und damit $x_e = 0$ und $x(B_A^k \cup e) = f_k(B_A^k \cup e)$.
- Zuerst beweisen wir, dass $e \in A$. Falls A=E, so gilt dies trivialerweise. Andernfalls ist $A \in \mathfrak{A}_k$; wegen $x(A) = f_k(A)$, (2.5) und $B_A^k \subseteq A$ gilt auch $x(A \cup e) = f_k(A \cup e)$, damit $f_k(A \cup e) = x(A \cup e) = x(A) = f_k(A) \leq f_k(A \cup e)$, somit Gleichheit und also $f_k(A) = f_k(A \cup e)$; da A f_k-abgeschlossen ist, gilt somit $e \in A$.
- Ist nun $e \in V_A^k$, so gilt wegen $V_A^k \subseteq B_A^k$ sofort $e \in B_A^k$.
- Falls aber $e \in A - V_A^k$, so gilt wegen (3.9) und der Voraussetzung

$$f_k(B_A^k) = f_k(V_A^k) + f_k^A(S_A^k) \leq f_k(V_A^k) + f_k^A(S_A^k \cup e) = f_k(B_A^k \cup e) = f_k(B_A^k) \quad ,$$

somit Gleichheit und damit $f_k^A(S_A^k) = f_k^A(S_A^k \cup e)$. Damit gilt aber $x(S_A^k \cup e) = f_k^A(S_A^k \cup e)$, und weil S_A^k die maximale f_k'-kritische Menge bezüglich $x|A-V_A^k$ ist, gilt $e \in S_A^k$, somit $e \in B_A^k$.

□

3.24 Lemma: Alle Mengen $B \in F_A^k$, $A \in \mathfrak{R}_k^E$, k=1,2, konstruiert in Schritt 2.1 des Algorithmus (3.19), können in die Familie $\mathfrak{R}_k$ eingefügt werden, so dass die neue Familie $\hat{\mathfrak{R}}_k$ wieder eine Familie von verschachtelten Mengen ist.

Beweis:

- Wegen der Konstruktion der Menge B_A^k und der Partition F_A^k gilt

  $V_A^k \subseteq \bigcup\{B: B \in F_A^k\} = B_A^k \subseteq A$

- Somit gilt für alle $A' \in \mathfrak{R}_k^E$, $B' \in F_{A'}^k$, k=1,2 :

  falls $A \cap A' = \emptyset$, so $B \cap A' = \emptyset$ und $B \cap B' = \emptyset$,

  falls $A \subset A'$, so $B \cap A' = B$ und $B \cap B' = B$,

  falls $A = A'$, so $B \cap A' = B$ und $B \cap B' = \emptyset$ für $B \neq B'$

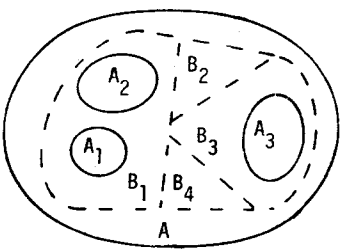
- Der einzige nichttriviale Punkt ist zu zeigen, dass

  falls $A \supset A'$, so $B \cap A' \in \{\emptyset, A'\}$:

  Der Beweis dafür ist ähnlich wie derjenige in (3.11):
  - Nimm an, dass $B \cap A' \neq \emptyset$, aber $A' \not\subseteq B$.
  - Sei $\overline{B} := B_A^k - B$ Wegen der Definition einer f_k-Separation und wegen (3.11.a) gilt $f_k(B_A^k) = f_k(B) + f_k(\overline{B})$, und damit wegen (3.11.b) $f_k(A') = f_k(A' \cap B) + f_k(A' \cap \overline{B})$. Aber dann ist A' f_k-separabel und dies ist ein Widerspruch, da wegen $A' \subset A$ $A' \in \mathfrak{R}_k$ gilt und alle Mengen in $\mathfrak{R}_k$ f_k-nicht-separabel sind.
  - Somit gilt $A' \subseteq B$.

  □

- Die folgende Illustration zeigt die Situation:



Figur 3.5: Einfügen der neuen Mengen $B \in F_A^k$ für ein $A \in \mathfrak{R}_k^E$:
$F_A^k = \{B_1, B_2, B_3, B_4\}$. $A_1 \cup A_2 \cup A_3 = V_A^k \subseteq B_A^k \subseteq A$.

3.25 Lemma: Nach Schritt 2.2 des Algorithmus ist der neue duale Vektor $\hat{y}$ dual zulässig.

Beweis: In den Punkten a) bis e) treffen wir Vorbereitungen für den eigentlichen Beweis in f). Wo nichts anderes angegeben ist steht der Index k jeweils für k=1 oder k=2.

a) $S^k = S^k \cap E = S^k \cap (\theta\{A-V_A^k\colon A\in \mathfrak{R}_k^E\}$

$= \theta\{S_A^k\colon A\in \mathfrak{R}_k^E\} = \theta\{B_A^k - V_A^k\colon A\in \mathfrak{R}_k^E\}$

$= \theta\{(\theta\{B\colon B\in F_A^k\} - V_A^k)\colon A\in \mathfrak{R}_k^E\}$.

b) $E-S^k = \theta\{(A-V_A^k)\colon A\in \mathfrak{R}_k^E\} - \theta\{S_A^k\colon A\in \mathfrak{R}_k^E\} = \theta\{(A-B_A^k)\colon A\in \mathfrak{R}_k^E\}$

$= \theta\{(A - \theta\{B\colon B\in F_A^k\})\colon A\in \mathfrak{R}_k^E\}$

c) Bemerkung: Jede Sequenz von (ev. echt verschachtelten) Mengen der Familie $\bar{\mathfrak{R}}_k$ hat die Form:

$\emptyset \subseteq B_A^k \subseteq A \subseteq B_{A'}^k \subseteq \ldots \subseteq A'' \subseteq B_E^k \subseteq E$, wo $A,A',A''\in \mathfrak{R}_k$, $F_A^k, F_{A'}^k, F_{A''}^k \in \bar{\mathfrak{R}}_k - \mathfrak{R}_k$,
d.h. zwischen zwei Mengen $A,A'\in \mathfrak{R}_k$ gibt es ein $B\in \bar{\mathfrak{R}}_k - \mathfrak{R}_k$
und zwischen zwei Mengen $B,B'\in \bar{\mathfrak{R}}_k - \mathfrak{R}_k$ gibt es ein $A\in \mathfrak{R}_k$.
Dies folgt sofort aus der Definition der Mengen $B_A^k\colon A\in \mathfrak{R}_k^E$.

d) Sei $\bar{\bar{\mathfrak{R}}}_k$ aus $\bar{\mathfrak{R}}_k$ erhalten durch Identifizierung identischer Mengen und sei $\hat{\mathfrak{R}}_k$ aus $\bar{\bar{\mathfrak{R}}}_k$ erhalten durch Streichung aller Mengen $A\in \bar{\bar{\mathfrak{R}}}_k$ mit $y_k(A)=0$.

e) Für jede Funktion $\alpha : L_E \to \mathbb{R}$ gilt nun folgende Beziehung:

$\sum\{\alpha(A).\hat{y}_k(A)\colon A\in \hat{\mathfrak{R}}_k\} = \sum\{\alpha(A).\hat{y}_k(A)\colon A\in \bar{\bar{\mathfrak{R}}}_k\}$

$= \sum\{\alpha(A).[y_k(A)+\delta.N_k(A)]\colon A\in \bar{\bar{\mathfrak{R}}}_k\}$

$= \sum\{\alpha(A).y_k(A)\colon A\in \mathfrak{R}_k\} + \delta.\sum\{\alpha(A).N_k(A)\colon A\in \bar{\bar{\mathfrak{R}}}_k\}$

$= \sum\{\alpha(A).y_k(A)\colon A\in \mathfrak{R}_k\}$

$+\delta.[\sum\{\alpha(B)\colon B\ F_A^k,\ A\in \mathfrak{R}_k^E\} - \sum\{\alpha(A)\colon A\in \mathfrak{R}_k\}]$.

(3.25 Fortsetzung)

f) Um nun die Aussage von (3.25) zu beweisen, führen wir folgende Ueberlegungen durch:

- $\hat{y}_m \geq 0$, da $\delta \leq y_m$
- $\hat{y}_k(A) \geq 0$ für alle $A \in L_E$, da $\delta \leq \min\{y_k(A): N_k(A)=-1, A \in L_E\}$.
- $t(\hat{y},j) \geq c_j$ für alle $j \in E$:
  - Um $t_k(\hat{y},j)$, k=1,2, zu berechnen, müssen wir die Summation der $\hat{y}_k(A)$ über alle $A \in \hat{\mathfrak{R}}_k$, wobei $j \in A$, durchführen.
  - Wegen e), wobei $\alpha(A)=1$ für $j \in A$ und $\alpha(A)=0$ für $j \notin A$, wegen a), b) und c) und weil alle Mengen von $\mathfrak{R}_k$, welche j enthalten, eine Sequenz wie unter c) geschildert bilden, gilt
    - falls $j \in S^k$, so ist die Anzahl der Mengen $B \in F_A^k$: $A \in \mathfrak{R}_k^E$, welche j enthalten, gleich der Anzahl der Mengen $A \in \mathfrak{R}_k$, welche j enthalten, plus 1, somit $t_k(\hat{y},j) = t_k(y,j)+\delta$.
    - falls $j \in E-S^k$, so sind die beiden Anzahlen gleich, also
    $$t_k(\hat{y},j) = t_k(y,j) .$$
  - Somit gilt
    $$t(\hat{y},j) = t(y,j)+\delta \text{ , falls } j \in S^1 \cap S^2 ,$$
    $$t(\hat{y},j) = t(y,j) \text{ , falls } j \in S^1-S^2 \text{ oder } j \in S^2-S^1$$
    $$t(\hat{y},j) = t(y,j)-\delta \text{ , falls } j \in E-(S^1 \cup S^2).$$
  - Somit gilt $t(\hat{y},j) \geq c_j$ für alle $j \in E$, weil
    $$\delta \leq \min\{t(y,j)-c_j: j \in E-(S^1 \cup S^2)\}.$$

□

3.26 Lemma: Nach der Revision der Duallösung in Schritt 2.2 des Algorithmus (3.19) gilt

$$S_x \subseteq \hat{E}^0 := \{e \in E: t(\hat{y},e) = c_e\} \quad .$$

Beweis:

Für alle $e \in S_x$ galt $e \in E^0$. Weil $S^1 \cup S^2 \supseteq E^0$, gilt somit $e \in S^1$ oder $e \in S^2$.

Sobald wir bewiesen haben, dass $e \notin S^1 \cap S^2$, gilt wegen der Resultate in (3.25), besonders den Ueberlegungen in (3.25.f), sofort

$$t(\hat{y},e) = t(y,e) = c_e \quad \text{und damit} \quad e \in \hat{E}^0 \ .$$

Beweis für $e \notin S^1 \cap S^2$:

- Wegen Konstruktion gilt $x \in P_1 \cap P_2$, damit wegen den gleichen Ueberlegungen wie in (3.22) $x \in P_1' \cap P_2'$.
- Es gilt $x^{|S \cup e} \in P_1' \cap P_2'$. Somit gilt wegen $x_e > 0$
  - falls $e \in S_x$ und $e \notin S$, dann $\quad e \notin SV_2'(x|S) = S^2$
  - falls $e \in S_x$ und $e \in S$, dann $\quad e \notin SV_1'(x|E-S) = S^1$
- Somit ist $e \notin S^1 \cap S^2$.

3.27 Termination des Algorithmus

Proposition: Der Wert von δ berechnet in Schritt 2.2 von Algorithmus (3.19) ist grösser als 0.

Beweis:

- Falls $\delta = y_m$, dann gilt $\delta>0$ weil $y_m>0$ (andernfalls hätte der Algorithmus schon vorher terminiert.
- Falls $\delta = y_k(A)$ für ein $A\in L_E$, k=1 oder k=2, dann gilt $N_k(A)=-1$ und somit $A\in \mathfrak{R}_k$. Damit ist $\delta > 0$, wegen $y_k(A)>0$.
- Falls $\delta = t(y,e)-c_e$ für ein $e\in E$, dann ist $e\in E-(S^1\cup S^2)$, somit $e\notin E^0$ und damit $\delta > 0$ wegen $t(y,e)>c_e$.

□

Proposition: Für ganzzahlige Vektoren c terminiert der Algorithmus (3.19) nach höchstens $c_{max} := \max\{0,c_e : e\in E\}$ Iterationen, d.h. Revisionen der Duallösung.

Beweis:

Dies ist klar wegen der Definition von $y_m = c_{max}$ und der Berechnung von δ : y und δ sind ganzzahlig und deshalb gilt $\delta\geq 1$ für jede Revision der Duallösung. Somit ist $y_m=0$ nach spätestens c_{max} Revisionen.

□

Bemerkung: Diese Abschätzung ist natürlich, obwohl leicht einzusehen, nicht das, was wir wünschen. Die Komplexität des Algorithmus sollte nicht von den Gewichten abhängig sein, und somit auch für nichtganzzahlige c gelten (vgl. (3.5)). Im folgenden Satz (3.28) wird aber gezeigt, dass wir für den Algorithmus eine Komplexität angeben können, welche unabhängig von der Wahl von c ist.

3.28 Satz: Für k=1,2, sei Z_k die Komplexität der Subroutine zur Berechnung, ob $x \in P_k$ oder nicht. Sei $Z := \max\{Z_1, Z_2\}$.

Dann kann mit Algorithmus (3.19) ein Paar von optimalen Vektoren (x,y) in $O(n^3.K.(Z+n^2))$ Rechenschritten erhalten werden. (In (3.34) diskutieren wir diese Komplexität).

Beweis:

- In (3.30) bis (3.33) beweisen wir, dass es höchstens $O(n^2)$ Revisionen zwischen zwei Augmentationen geben kann. (3.29) beinhaltet die dazu notwendigen Definitionen.
- Wegen der Definition von n und K gibt es höchstens n.K Augmentationen, da jede Augmentation die Komponentensumme des Primalvektors um mindestens 1 erhöht.
- Für jede Augmentation gibt es einen Durchlauf von Schritt 1.2 des Algorithmus (3.19) mit Berechnung der minimalen f_k-kritischen Mengen bezüglich x. Da wir mit dem schlimmsten Fall von Inkrementen $\delta=1$ für die Vergrösserung der Komponentensumme rechnen, dürfen wir nach (2.22) $O(n^2.Z)$ als Schranke annehmen.
- Schritt 1.1 kann in $O(n^2)$ Rechenschritten realisiert werden: Mit Berücksichtigung der verschachtelten Familien kann bei geeigneter Codierung für jedes $A \in \mathfrak{A}_k^E$, k=1,2, in O(n) Rechenschritten die Menge $C_e^k(x) \cap A - V_A^k$ berechnet werden, für $e \in E$ (vgl. (6.2)).
- 1.2 benötigt $O(n^2)$ Rechenschritte für jeden Durchlauf ohne Augmentation (d.h. vor jeder Revision). Dies wegen (2.22), weil keine Neurechnung der Mengen $C_e^k(x)$, $e \in E$, k=1,2, erfolgen muss.
- Somit benötigt Schritt 1 $O(n^2)$ Rechenschritte für jede Revision.
- Schritt 2.1 benötigt $O(n^2)$ Rechenschritte, um die Mengen S^k und B_A^k zu berechnen, für k=1,2 und $A \in \mathfrak{A}_k$. Wegen (3.13.b) ist leicht einzusehen, dass alle Separationen F_A^k, $A \in \mathfrak{A}_k$ in $O(n^2)$ Rechenschritten erhalten werden können, weil wir für jedes A nur noch die Mengen $C_e(x)$ aller Elemente e, $e \in B_A^k - V_A^k$ betrachten müssen.
- Schritt (2.2) benötigt O(n) Rechenschritte.
- Somit benötigt Schritt 2 $O(n^2)$ Rechenschritte für jede Revision.
- Da es höchstens $O(n^2)$ Revisionen zwischen zwei Augmentationen gibt, gibt es höchstens $O(n.K \cdot n^2)$ Durchläufe von Algorithmus (3.19) ohne den wiederholten Durchlauf von 1.2 wegen Augmentationen.
- Die Komplexität von Algorithmus (3.19) ist somit $O(n.K.(n^2Z+n^4))$.

□

3.29 Definitionen:

Betrachte eine bestimmte Revision. x sei der gültige Primalvektor. Sei $\mathcal{A}$ die während des letzten Durchlaufs von Schritt 2.1 von Algorithmus (3.19) konstruierte vergrössernde Struktur, ℓ die Niveaufunktion der Elemente von E^0 in $\mathcal{A}$.

Seien y und $\hat{y}$ die dualen Vektoren vor und nach der Revision. $\hat{y}$ ist somit der duale Input-Vektor für den nächsten Durchlauf von Schritt 1 von Algorithmus (3.19). Für k=1,2 sind dann $\hat{\mathfrak{R}}_k$, $\hat{V}_A^k$, $\hat{T}_A^k$, $\hat{P}_k'$, $\hat{E}^0$ die $\mathfrak{R}_k$, V_A^k, T_A^k, P_k', E^0 entspr. Familien, Mengen und Polymatroide.

$\hat{x}$ ist dann der primale Vektor, gültig für die nächste Revision (also $\hat{x}=x$, falls dazwischen keine Augmentation stattfand).

$\hat{\mathcal{A}}$ und $\hat{\ell}$ sind dann entsprechend $\mathcal{A}$ und ℓ definiert.

Für k=1,2, seien $\hat{SV}_k'$, $\hat{S}$, $\hat{S}^k$, $\hat{B}_A^k$, $\hat{\mathfrak{R}}_k$ die SV_k', S, S^k, B_A^k, $\bar{\mathfrak{R}}_k$ entsprechenden Familien und Mengen.

3.30 Proposition: Für k=1,2 gilt $SV_k(x) = SV_k'(x)$

Beweis:

$\subseteq$: Sei $e \in SV_k(x)$. Da x eine f_k-Basis aller Mengen $A \in \mathfrak{R}_k$ ist, gilt $SV_k(x) \supseteq V_E^k$.

Sei nun A die kleinste Menge aus $\mathfrak{R}_k \cup \{SV_k(x)\}$, welche e enthält. Wegen der Konstruktion von P_k' aus P_k (vgl. (3.9)) gilt:

$$f_k'(A-V_A^k) = f_k^A(A-V_A^k) = f_k(A)-f_k(V_A^k) = x(A)-x(V_A^k) = x(A-V_A^k) \quad .$$

Da $e \in A-V_A^k$, gilt somit $e \in SV_k'(x)$.

$\supseteq$: Sei $e \in SV_k'(x)$. Sei $A := SV_k'(x) \cap E-V_E^k$. Da x eine f_k-Basis von allen Mengen in $\mathfrak{R}_k$ ist, dass $x(V_E^k)=f_k(V_E^k)$. Ebenso gilt $x(A)=f_k'(A)$ wegen der Konstruktion von P_k' aus P_k (vgl. 3.9)). Somit gilt:

$$f(A \oplus V_E^k) = f_k^E(A)+f_k(V_E^k) = f_k'(A)+f_k(V_E^k) = x(A)+x(V_E^k) = x(A \oplus V_E^k) \quad .$$

Weil $e \in A \oplus V_E^k$, so gilt somit $e \in SV_k(x)$.

3.31 Proposition: Es gelten folgende Aussagen für die Mengen $S, S^1, S^2, S_x, E^0, \hat{E}^0$:

a) $S^1 \subseteq E-S$

b) $S_x \cap S^2 \subseteq S$

c) $S \subseteq S^2$

d) $S \subseteq \hat{E}^0$

Beweise:

a) und b): Sei $SV_k^0(x)$ der f_k^0-Spann bezüglich x, für k=1,2.

Wegen (2.20) und da $x_j=0$ für alle $j\in E-E^0$, gilt

$$E^0-S = SV_1^0(x|E^0-S) \subseteq SV_1^1(x|E^0-S) = SV_1^1(x|E-S) = S^1 .$$

Weil $S\subseteq E^0$, gilt wegen $S\cap SV_1^0(x|E^0-S)=\emptyset$ sofort auch $S\cap SV_1^1(x|E^0-S)=\emptyset$ und damit $S\cap S^1=\emptyset$, d.h. $S^1\subseteq E-S$.

Wegen der Betrachtungen im Beweis von (3.26) gilt für alle $e\in S_x$, dass entweder $e\in S^1$ oder $e\in S^2$. Andererseits gilt für alle $e\in S_x$, dass entweder $e\in E^0-S$ oder $e\in S$. Wegen obiger Abschätzung von E^0-S gilt für jedes $e\in E^0-S$, dass $e\in S^1$. Durch Bildung der Kontraposition gilt dann für jedes $e\in S_x$ mit $e\in S^2$, dass $e\in S$.

c): Wegen (2.20) gilt sofort

$$S \subseteq SV_2^0(x|S) \subseteq SV_2^1(x|S) = S^2$$

d): Wegen a) gilt $S\cap S^1=\emptyset$. Der Art der Berechnung von $t(\hat{y},j)$ für ein $j\in E$ in (3.25.f) entnehmen wir, dass $t(\hat{y},j)>t(y,j)$ nur dann, wenn $j\in S^1\cap S^2$. Da für ein $e\in S$ sicher $e\in S^1\cup S^2$ gilt (wegen $S\subseteq E^0\subseteq S^1\cup S^2$), gilt somit, wieder wegen der Berechnung in (3.25.f), dass $t(\hat{y},e)=t(y,e)=c_e$. Damit gilt aber $S\subseteq\hat{E}^0$.

□

3.32 Lemma: Sei $\hat{x}=x$. Dann gilt $\hat{S} \supseteq S$.

Beweis: Wir zeigen für alle $e \in S$, dass $\hat{\ell}(e) \leq \ell(e)$ (daraus folgt sofort $e \in \hat{S}$). Diese Aussage beweisen wir durch Induktion nach den Niveaus i in $\mathcal{A}$:

Verankerung für i=1: Sei $\ell(e)=1$ für ein $e \in E^0$. Somit gilt $e \notin SV_1'(x)$ und $e \in S$. Wegen (3.31.d) gilt $e \in \hat{E}^0$. Wegen (3.30) gilt aber $e \notin SV_1(x)$, wegen $\hat{x}=x$ damit auch $e \notin \hat{SV}_1'(x)$ und somit wegen der Konstruktion von $\hat{\mathcal{A}}$ auch $\hat{\ell}(e)=1=\ell(e)$.

Induktionsschritt: Nimm an, dass $\hat{\ell}(j) \leq \ell(j)$ für alle $j \in S$ mit $\ell(j)<i$. Wir beweisen, dass $\hat{\ell}(e) \leq \ell(e)$ für alle $e \in S$ mit $\ell(e)=i$:

Beweis:

- Sei $e \in S$ mit $\ell(e)=i$.
- Wegen (3.31.d) gilt $e \in \hat{E}^0$.
- Sei i gerade. Sei A die kleinste Menge in $\mathfrak{R}_2^E$ mit $e \in A$. Wegen der Konstruktion von P_2' aus P_2 und von $\mathcal{A}$ gibt es ein $g \in E^0$ mit $e \in C_g^2(x)$, $\ell(g)=i-1$, $g \in A-V_A^2$. Für g ist somit die Induktionsvoraussetzung gültig: $\hat{\ell}(g) \leq \ell(g)$. Da $e,g \in S$, gilt wegen (3.31.c) $e,g \in S^2$. Wegen (3.13.b) sind e und g in der gleichen Menge $D \in F_A^2$ und wegen der Konstruktion von $\hat{\mathfrak{R}}_2$ gilt $D \in \hat{\mathfrak{R}}_2$ und $e,g \in D-V_D^2$. Somit gilt wegen der Konstruktion von $\hat{P}_2$ aus P_2 und von $\hat{\mathcal{A}}$:
  - falls $\hat{\ell}(g)$ ungerade, so $\hat{\ell}(e) \leq \hat{\ell}(g)+1 \leq \ell(g)+1 = i$.
  - falls $\hat{\ell}(g)$ gerade, so gibt es ein $j \in D-V_D^2$, $\hat{\ell}(j)=\hat{\ell}(g)-1$, $g \in C_j^2(x)$, somit wegen (2.12) $e \in C_j^2(x)$ und damit $\hat{\ell}(e) \leq \hat{\ell}(j)+1 = \hat{\ell}(g) \leq \ell(g) = i-1$.
- Sei i ungerade. Sei A die kleinste Menge in $\mathfrak{R}_1^E$ mit $e \in A$. Wegen Konstruktion von P_1' aus P_1 und von $\mathcal{A}$ gibt es ein $g \in E^0$, $\ell(g)=i-1$, $g \in C_e^1(x)$, $g \in A-V_A^1$. Für g gilt die Ind.voraussetzung: $\hat{\ell}(g) \leq \ell(g)$. Da $e,g \in S$ gilt wegen (3.31.a) $e,g \notin S^1$ und damit $e,g \in A-B_A^1$. Sei D die kleinste Menge in $\hat{\mathfrak{R}}_1$, die A enthält (ev. D=A). Es gilt $\hat{V}_D^1=B_A^1$ und $e,g \in D-V_D^1$, somit wegen Konstruktion von $\hat{P}_1'$ aus P_1 und von $\hat{\mathcal{A}}$:
  - falls $\hat{\ell}(g)$ gerade, so $\hat{\ell}(e) \leq \hat{\ell}(g)+1 \leq \ell(g)+1 = i$.
  - falls $\hat{\ell}(g)$ ungerade, so gibt es ein $h \in D-V_D^1$, $\hat{\ell}(h)=\hat{\ell}(g)-1$, $h \in C_g^1(x)$, somit wegen (2.12) $h \in C_e^1(x)$ und damit $\hat{\ell}(e) \leq \hat{\ell}(h)+1=\hat{\ell}(g) \leq \ell(g)=i-1$.

□

3.33 Lemma: Es gibt höchstens $O(n^2)$ Revisionen der dualen Lösung zwischen zwei Augmentationen des Primalvektors.

Beweis:

Falls zwischen zwei Revisionen keine Augemntation erfolgt, gilt $\hat{x}=x$. Ferner gibt es wegen der Wahl von δ in Schritt 2.2 von Algorithmus (3.19), falls $\delta \neq y_m$, ein $e \in E-(S^1 \cup S^2)$ mit $e \in \hat{E}^0$ oder ein $A \in \mathfrak{A}_k$, k=1 oder k=2, mit $N_k(A)=-1$ und $A \notin \hat{\mathfrak{A}}_k$. Wir werden zeigen, dass diese beiden Fälle nicht "allzu oft" auftreten können, ohne dass eine Augmentation des Primalvektors erfolgt.

Fall 1: Es gibt ein $e \in E-(S^1 \cup S^2)$, $e \in \hat{E}^0$. Wegen (3.31.c) gilt, da $e \notin S^2$, sofort $e \in S$. Wir werden zeigen, dass $e \in \hat{S}$. Wegen (3.32) gilt dann $\hat{S} \supseteq S_{\hat{x}e} \supset S$. Da es aber höchstens n Elemente in den Mengen $\hat{S}$ bzw. S gibt, tritt Fall 1 somit höchstens n Male ein.

Beweis für $e \in \hat{S}$:

- Wegen (3.30) gilt $SV_1(x)=SV_1'(x)$.
- Falls $e \notin SV_1(x)$, so gilt sofort $\ell(e)=1$ und damit $e \in \hat{S}$.
- Falls $e \in SV_1(x)$, so existiert $C_e^1(x)$. Es gilt aber nach Voraussetzung $e \notin S^1 = SV_1'(x|E-S) = SV_1'(x|E^0-S)$. Sei nun A die kleinste Menge in $\mathfrak{A}_1^E$, welche e enthält. Es gilt, da A f_k-kritisch bezüglich x ist oder A=E gilt, dass $C_e^1(x) \subseteq A$.

  Da $x_e=0$ wegen $e \notin E^0 \subseteq S^1 \cup S^2$, da $C_e^1(x) \ni \{e\}$ wegen $f_1(e)>0$, da $e \in SV_1'(x)$, aber $e \notin S^1$, gibt es ein $g \neq e$, $g \in C_e^1(x) \cap (A-V_A^1)$, $g \in S^2$. Wegen (2.7) gilt $g \in S_x$ und damit wegen (3.31.b) sofort $g \in S$, und somit gilt wegen der Konstruktion von B_A^1, dass $e,g \in A-B_A^1$.

  Sei nun D die kleinste Menge in $\hat{\mathfrak{A}}_1$, welche A enthält (ev. D=A). Es gilt $\hat{V}_D^1=B_A^1$ und $e,g \in D-\hat{V}_D^1$. Somit gilt wegen Konstruktion von $\hat{P}_1'$ aus P_1 und von $\hat{\mathfrak{A}}$, und weil wegen (3.32) $g \in \hat{S}$:
  - falls $\hat{\ell}(g)$ gerade, so $\hat{\ell}(e) \leq \hat{\ell}(g)+1$ und damit $e \in \hat{S}$.
  - falls $\hat{\ell}(g)$ ungerade, so gibt es ein $h \in D-\hat{V}_D^1$, $\hat{\ell}(h)=\hat{\ell}(g)-1$, $h \in C_g^1(x)$, $h \in \hat{S}$, somit wegen (2.12) $h \in C_e^1(x)$ und damit $\hat{\ell}(e) \leq \hat{\ell}(h)+1 = \hat{\ell}(g)$. Somit gilt $e \in \hat{S}$.

□

(Lemma (3.33), Fortsetzung)

Fall 2: Es gibt ein $A \in \mathfrak{R}_k$, k=1 oder k=2, mit $N_k(A)=-1$ und $A \notin \hat{\mathfrak{R}}_k$.

Wir zeigen, dass Fall 2 höchstens $O(n^2)$ Male auftreten kann:

- Sei $T := T_A^k$ (vgl. (3.14.d)) und sei $D \in F_T^k$ die eindeutige Menge in B_T^k, welche A enthält. Da $N_k(A)=-1$, gilt $B_A^k \subset A \subset D$. Somit gibt es mit den gleichen Ueberlegungen wie in (3.14.e) ein $h \in D-A$ und ein $j \in A-B_h^k$, $j,h \in S_x=\{e: x_e>0\}$.
- Sei nun für k=1,2, I_e^k (bzw. $\hat{I}_e^k$) eine kleinste Menge in $\bar{\mathfrak{R}}_k$ (bzw. $\hat{\bar{\mathfrak{R}}}_k$), welche ein Element $e \in E$ enthält (in $\bar{\mathfrak{R}}_k$ und $\hat{\mathfrak{R}}_k$ können auch gleiche Mengen sein).
- Es gilt nun $D = I_h^k \supset I_j^k = A$. Durch Konstruktion von $\hat{\mathfrak{R}}_k$ gilt nun: $D \in \hat{\mathfrak{R}}_k$, $F_A^k \subseteq \hat{\mathfrak{R}}_k$, und nach Voraussetzung $A \notin \hat{\mathfrak{R}}_k$. $\cup\{G: G \in F_A^k\} = \hat{V}_D^k$.
- Wegen (3.26) und wegen (3.25.a,b) gilt $h \in S^k$ und $j \in S^{\bar{k}}$, wobei $\bar{k}:=1$, falls k=2 und $\bar{k}:=2$, falls k=1.
- Nimm nun an, dass $\hat{S}=S$. Dann gilt $h \in \hat{S}^k$ und $j \in \hat{S}^{\bar{k}}$, da sonst $\hat{S} \neq S$ wäre, wegen (3.31.a,b). Dann gilt aber $h \in \hat{B}_D^k - \hat{V}_D^k$ und $j \in D-\hat{B}_D^k$. Damit gilt aber $\hat{B}_D^k = \hat{I}_h^k \subset \hat{I}_j^k = D$, im Unterschied zu $I_h^k \supset I_j^k$. Das heisst, h und j haben in den Familien von verschachtelten Mengen $\mathfrak{R}_k$ und $\hat{\mathfrak{R}}_k$ "ihre Plätze vertauscht".
- Wegen (3.31.a) und (3.31.b) heisst das nun aber, dass jedesmal, wenn Fall 2 eintritt, mindestens ein Element aus E-S und ein Element aus S ihre Plätze vertauschen müssen. Man kann sagen, die Elemente aus S streben nach untergeordneten Mengen in der Familie und die Elemente aus E-S nach übergeordneten.
- Wegen (3.31) gilt $\hat{S} \supseteq S$. Zwischen zwei Augmentationen kann somit nur der Fall eintreten, dass ein Element aus E-S in $\hat{S}$ hinüberwechselt. Es kann dann mit allen in $E-\hat{S}$ verbliebenen Elementen "den Platz wechseln" (dies sind höchstens n-1 Elemente). Daraus folgt aber, dass Fall 2 höchstens $O(n^2)$ Male auftreten kann.

Da für jede Revision zwischen zwei Augmentationen Fall 1 oder Fall 2 eintreten muss, gibt es somit höchstens $O(n^2)$ Revisionen zwischen zwei Augmentationen.

□

3.34 Bemerkungen zur Komplexität des Algorithmus (3.19):

Wir würden gerne eine Komplexität angeben, welche proportional nicht zu K, sondern höchstens zu logK ist. Dies ist im ungewichteten Fall (Kapitel 2) gelungen. Im gewichteten Fall sind aber die Ansätze des Autors bisher nicht erfolgreich gewesen.

Man könnte zum Beispiel versucht sein, eine Methode im Stile der sogenannten "Scaling" Methode zu finden, mit welcher Edmonds und Karp [E3] im Falle vom Hitchcock-Problem erfolgreich eine Komplexität proportional zu logK und nicht zu K angeben konnten. Das Problem ist hier aber, dass, falls $f: L_E \to \mathbb{R}$ eine ganzzahlige β_0-Funktion ist, die Funktionen $g,h : L_E \to \mathbb{R}$ mit $g(A):= \left\lceil \frac{f(A)}{2} \right\rceil$ und $h(A):= \left\lfloor \frac{f(A)}{2} \right\rfloor$ im allgemeinen keine β_0-Funktionen sind ($\lceil x \rceil$ ist die kleinste ganze Zahl grösser oder gleich x, und $\lfloor x \rfloor$ die grösste ganze Zahl kleiner oder gleich x).

Auch Versuche, einen "direkten" Beweis einer Komlexität proportional zu logK für den Algorithmus (3.19) zu finden, scheiterten. Das liegt sehr wahrscheinlich daran, dass die wählbaren Inkremente δ zur Vergrösserung der Komponentensumme des Primalvektors im gewichteten Fall gerade noch genügend gross sind. Vergleiche dazu die Lemmas (2.29) und (3.31), wo gegenteilige "Tendenzen" für das Verhalten der Elemente während einer Augmentation und während einer Revision aufgezeigt werden: Während Augmentationen bewegen sich die Elemente nach höheren Niveaus in der vergrössernden Struktur, während Revisionen nach tieferen Niveaus.

Immerhin ist die angegebene Komplexität von $O(n^3.K.(n^2+Z))$ um einiges kleiner als diejenige eines Algorithmus, welcher die Polymatroide zuerst in Matroide mit n.K Elementen transponiert (vgl. (1.13)) und dann einen Matroid-Intersektions-Algorithmus als Teilalgorithmus benützt. Nach [L1] ist die Komplexität des Matroid-Intersektionsalgorithmus von Lawler $O(m^2R^3+mR^2Z)$, wobei m die Anzahl Elemente und R das Minimum der Ränge der Matroiden ist. Dies ergäbe für R=m=n.K eine Komplexität von $O(n^5.K^5+n^3.K^3.Z)$ oder $O(n^3.K^3.(n^2.K^2 + Z))$.

4 Vektoren mit maximaler Gewichtssumme unter Vektoren mit einer vorgegebenen Komponentensumme in ganzzahligen Polymatroid-Intesektions-Polyedern

4.1 Gegeben die gleichen Definitionen wie in (3.1)

Das Problem, welches wir jetzt lösen wollen, ist folgendes:

$$\begin{aligned} &\text{Maximiere } c.x \text{ , wobei } x \in \mathbb{R}^E \text{ und} \\ &x_e \geq 0 \qquad \text{für alle } e \in E \\ &x(S) \leq f_1(S) \quad \text{für alle } S \in K_E \\ &x(S) \leq f_2(S) \quad \text{für alle } S \in K_E \\ &x(E) = m \qquad m \in N \end{aligned}$$

4.2 Wegen (3.15) ist Problem (4.1) das gleiche Problem wie folgendes:

$$\begin{aligned} &\text{Maximiere } c.x \text{ , wobei } x \in \mathbb{R}^E \text{ und} \\ &x_e \geq 0 \qquad \text{für alle } e \in E \\ &\left.\begin{aligned} x(S) &\leq f_1(S) \\ x(S) &\leq f_2(S) \end{aligned}\right\} \begin{array}{l}\text{für alle } S \in K_E,\ S \text{ } f_k\text{-abgeschlossen und} \\ \quad f_k\text{-nichtseparabel, für } k=1,2\end{array} \\ &x(E) = m \qquad m \in N \end{aligned}$$

4.3 Seien y^1, y^2, y, $t_1(y,e)$, $t_2(y,e)$ und $t(y,e)$ wie in (3.16) definiert. Dann ist das duale lineare Programm zu (4.2) das folgende LP:

$$\begin{aligned} &\text{Minimiere } f.y = m.y_m + f_1.y^1 + f_2.y^2, \text{ wobei} \\ &t(y,e) \geq c_e \qquad \text{für alle } e \in E \\ &y^1, y^2 \geq 0 \\ &y_m \in \mathbb{R} \end{aligned}$$

4.4 Komplementärschlupfbedingungen (vgl. (3.17)):

(A) $x_e > 0 \Rightarrow t(y,e) = c_e$

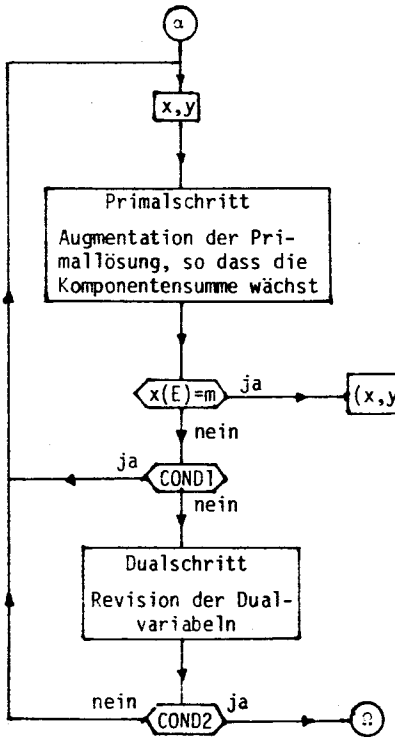
(B) $y_k(S) > 0 \Rightarrow x(S) = f_k(S)$, d.h. x ist eine f_k-Basis von S.

4.5 Algorithmus, generelles Konzept (vgl. (3.18)):

Der Algorithmus startet mit einer Initiallösung (x,y), welche dual zulässig ist und (4.4) erfüllt, ebenso alle Ungleichungen von (4.2) (d.h. $x \in P_1 \cap P_2$). Nur die Gleichung $x(E)=m$ ist nicht eingehalten. Wir können die gleiche Initiallösung wie in Algorithmus (3.19) verwenden.

Der Algorithmus terminiert, sobald entweder $x(E)=m$ oder wenn wir beweisen können, dass es keinen primal zulässigen Vektor mit der geforderten Komponentensumme gibt.

Das folgende Flussdiagramm zeigt dieses Konzept:



$x \in P_1 \cap P_2$

$y = [y_m, y_k(A)$: A f_k-abgeschlossen und f_k-nichtseparabel, für $k=1,2$]

$x(E) = \sum\{x_e : e \in E\}$

COND1: Eine Augmentation war möglich

COND2: Es existiert kein primal zulässiger Vektor mit grösserer Komponentensumme.

4.6 Algorithmus

Gegeben eine endliche Menge E. Für k=1,2 sei P_k ein ganzzahliges Polymatroid, f_k seine entsprechende ganzzahlige β_0-Funktion. Gegeben der Vektor $c=[c_e : e\in E]$, $c_e \in \mathbb{R}$, und eine positive ganze Zahl m. Finde einen Vektor x, x aus der Menge aller Vektoren **z**, $z\in P_1\cap P_2$, $z(E)=m$, so dass c.x maximal.

Der Algorithmus ist im wesentlichen der Algorithmus (3.19), wobei zwei Modifikationen vorgenommen werden:

- Die Bestimmung von δ in Schritt 2.2:
  $\delta := \min\{t(y,e)-c_e : e\in E-(S_1\cup S_2)\ ;\ y_k(A): N_k(A)=-1,\ A\subseteq E,\ k=1,2\}$
  Falls kein $e\in E-(S_1\cup S_2)$ und kein $y_k(A)$ mit $N_k(A)=-1$ existiert, dann sei $\delta := \infty$.

- Schritt 2.3 :
  Falls $\delta=\infty$, dann stop: x hat maximale Komponentensumme. Es gibt dann keinen primal zulässigen Vektor mit grösserer Komponentensumme (wegen (2.15), der Konstruktion der Mengen S_1 und S_2, weil $S_1\cup S_2 = E$ und wegen (3.22), d.h. $z\in P_1\cap P_2 \iff z\in P_1'\cap P_2'$ für alle $z\in \mathbb{R}^E$).
  Andernfalls gehe zu Schritt 1.

4.7 Validität des Algorithmus:

Es ist einfach nachzuprüfen, dass (3.20) bis (3.26) auch für den modifizierten Algorithmus anwendbar sind. Somit ist der Algorithmus gültig.
Es ist auch leicht einzusehen, dass (3.29) bis (3.34) auch auf den modifizierten Algorithmus angewendent werden können. Die Komplexität des modifizierten Algorithmus (4.6) ist somit die gleiche wie die Komplexität des Algorithmus (3.19).

5 Einige spezielle Polymatroide

5.1 Das Partitions-Polymatroid mit Kapazitäten

5.1.1 Gegeben eine endliche Menge E und eine Partition von E in nichtleere Teilmengen $E = \dot{\cup}\{E_i: i\in\{1,..,m\}\}$. Sei $I:=\{1,..,m\}$.

Gegeben eine Menge von nichtnegativen ganzen Zahlen $\{b_e: e\in E\}$ ('Kanten-Kapazitäten') und eine Menge von nichtnegativen ganzen Zahlen $\{d_i: i\in I\}$ ('Knoten-Kapazitäten'), so dass, falls $e\in E_i$, $b_e \le d_i$.

Sei $A_i := A\cap E_i$, für alle $A\in L_E$, $i\in I$.

Sei $b: L_E \to \mathbb{R}$ die Funktion definiert wie folgt:

$$b(A) := \begin{cases} 0 & , \text{ falls } A=\emptyset \\ \sum\{b_e: e\in E\} & , \text{ sonst} \end{cases} .$$

Sicher ist b eine β_0-Funktion.

Sei $f: L_E \to \mathbb{R}$ die Funktion definiert wie folgt:

$$f(A) := \sum\{\min\{b(A_i), d_i\}: i\in I\} .$$

Wir werden beweisen, dass f eine β_0-Funktion ist, und somit $P := P(K_E,f) = \{x\in\mathbb{R}_+^E: x(A)\le f(A) \text{ für alle } A\in K_E\}$ ein Polymatroid, das sogenannte Partitions-Polymatroid mit Kapazitäten

Beweis:

Sicher ist f nichtabnehmend und $f(\emptyset)=0$. Aber f ist auch submodular:

$$\begin{aligned} f(A\cup B)+f(A\cap B) &= \sum\{\min\{b((A\cup B)_i), d_i\}: i\in I\} + \sum\{\min\{b((A\cap B)_i), d_i\}: i\in I\} \\ &= \sum\{[\min\{b(A_i\cup B_i), d_i\} + \min\{b(A_i\cap B_i), d_i\}]: i\in I\} \\ &= \sum\{\min\{b(A_i\cup B_i)+b(A_i\cap B_i)\ ,\ d_i+b(A_i\cap B_i)\ ,\ 2d_i\}: i\in I\} \\ &\le \sum\{\min\{b(A_i)+b(B_i)\ ,\ b(A_i)+d_i\ ,\ b(B_i)+d_i\ ,\ 2d_i\}: i\in I\} \\ &= \sum\{[\min\{b(A_i), d_i\} + \min\{b(B_i), d_i\}]: i\in I\} \\ &= \sum\{\min\{b(A_i), d_i\}: i\in I\} + \sum\{\min\{b(B_i), d_i\}: i\in I\} \\ &= f(A) + f(B) . \end{aligned}$$

5.1.2 Subroutine

Um für ein $x \in \mathbb{R}^E$ zu testen, ob $x \in P$, benötigen wir $O(|E|)$ Schritte:

$$x \in P \quad <=> \quad \begin{cases} 0 \le x_e \le b_e & \text{für alle } e \in E \\ x(E_i) \le d_i & \text{für alle } i \in I \end{cases}$$

5.1.3 Komplexität der Intersektions-Algorithmen für Partitions-Polymatroide

Für $k=1,2$ seien m_k, I_k, $E = \dot{\cup}\{E_i^k : i \in I_k\}$, $\{b_e^k : e \in E\}$, $\{d_i^k : i \in I_k\}$, $b_k, f_k : L_E \to \mathbb{R}$, $P_k = P(E, f_k)$ gemäss (5.1) gegeben.
Sei $m := \min\{m_1, m_2\}$ und sei $M := \max\{m_1, m_2\}$. Sei $b_e := \min\{b_e^1, b_e^2\}$, für $e \in E$.

Betrachten wir nun Satz (2.23), d.h. den Aufwand an Rechenschritten, um einen Vektor mit maximaler Komponentensumme zu erhalten. Es ist nicht schwierig zu beweisen, dass sich die Komplexität auf $O(m^3M^2)$ reduziert, wenn wir berücksichtigen, dass

- es höchstens $|E| \le m.M$ Elemente gibt,
- wir $O(m.M)$ Rechenschritte benötigen, um alle Mengen $C_e^k(x)$, $e \in SV_k(x)$, $k=1,2$, zu berechnen,
- der Wert von δ in Schritt 6 von Algorithmus (2.18) in der folgenden Art berechnet werden kann, unter Annahme von $W=\{v_1,..,v_w\}$ und $v_1 \in E_i^1$ für ein $i \in I_1$ und $v_2 \in E_j^2$ für ein $j \in I_2$:
  $\delta = \min\{d_i^1 - x(E_i^1), b_{v_1} - x_{v_1}, x_{v_2}, b_{v_3} - x_{v_3}, x_{v_4}, .., x_{v_{w-1}}, b_{v_w} - x_{v_w}, d_j^2 - x(E_j^2)\}$,
- jedes Element höchstens einmal auf jedem Niveau 'kritisch' sein kann in dem Sinne, dass es den Wert von δ limitiert, so dass x_e entweder auf 0 reduziert oder auf b_e vergrössert wird (vgl. [E3]),
- es für jede Augmentation ein Element $e \in E$ gibt, welches entweder kritisch auf einem Niveau ist oder kein Element des f_k-Spanns des alten Vektors ist, aber ein Element des f_k-Spanns des neuen Vektors, für $k=1$ oder $k=2$.

Analog betrachten wir Satz (3.28), d.h. die Anzahl Rechenschritte, um einen Vektor mit maximaler Gewichtssumme zu erhalten. Die Komplexität wird auf $O(m^4.M^4.K)$ reduziert, wenn wir obige Gedanken berücksichtigen und ebenso, dass Schritt 1 und 2 von (3.19) ohne Augmentation von x in $O(m.M)$ Rechenschritten durchlaufen werden (die f_k-Separationen der Teilmengen von E sind durch die Partitionen gegeben)

Vergleiche diese Betrachtungen mit denjenigen in [E3] und [F1].

5.2 Das 'Minimum Cost Flow'-Problem als Intersektions-Problem von zwei Partitions-Polymatroiden mit Kapazitäten

5.2.1 'Minimum Cost Flow'-Problem, Transportprobleme, Hitchcock-Problem

Gegeben ein gerichteter Graph $G(V,E)$, wobei V eine endliche Menge von Knoten und E eine endliche Menge von Kanten, genannt (i,j), sind, so dass (i,j) inzident zu g_i und g_j sind, mit $g_i, g_j \in V$.

Nimm an, dass zu jeder Kante eine nichtnegative ganze Zahl b_{ij} gehört, die 'Kapazität' von (i,j), und ebenso die Kosten $a_{ij} \in \mathbb{R}$.

Seien $s,t \in V$ zwei Knoten, Quelle und Senke genannt. Sei $x_{ij} \in \mathbb{Z}$ die Menge des 'Flusses' durch die Kante (i,j). Dann ist ein (zulässiger) Fluss durch G definiert als eine Menge von Zahlen $x = (x_{ij})$, so dass

$$\text{(i)} \qquad 0 \le x_{ij} \le b_{ij} \text{ , für alle } (i,j) \in E \text{ ,}$$

$$\sum_j x_{ji} - \sum_j x_{ij} = d_i \text{ , wobei } i,j \in V \text{ und } d_i := \begin{cases} -v, & \text{falls } i=s \\ 0, & \text{falls } i \neq s,t \\ +v, & \text{falls } i=t \end{cases} .$$

v wird der Wert des Flusses genannt. Die Gleichungen mit $d_i=0$ heissen Flusserhaltungsgleichungen.

Das 'Minimum Cost Flow'-Problem, d.h. einen Fluss zu einem bestimmten Wert v zu finden, so dass $\sum_{i,j} a_{ij} \cdot x_{ij}$ minimal ist, ist das lineare Programm

'maximiere $-a.x$, wobei (i) für x erfüllt ist'.

Ein Transportproblem mit Kapazitäten ist eine Form des 'Minimum-Cost-Flow'-Problems, in welchem es für jeden Knoten i eine Zahl d_i gibt, wobei anstelle der Flusserhaltungsgleichungen in (i) gilt:

$$\sum_j x_{ij} - \sum_j x_{ji} \ge d_i \quad .$$

Falls $d_i<0$, >0, $=0$, dann ist der Knoten i ein Angebotsknoten, bzw. ein Nachfrageknoten, bzw. ein Umladeknoten.

Ein Hitchcock-(Transport)-Problem mit Kapazitäten ist ein Transportproblem auf einem bipartiten Graphen $G=(S^1,S^2,A)$, wobei alle

Angebotsknoten in S^1, alle Nachfrageknoten in S^2 und alle Kanten von S^1 nach S^2 gerichtet sind. Umladeknoten sind eliminiert.

Es ist klar, dass das allgemeine Transportproblem in ein herkömmliches 'Minimum Cost Flow'-Problem mit einem einzigen Quellen-Senken-Paar s,t transponiert werden kann. Beachte, dass, falls das Problem zulässig sein soll, die Summe aller Angebote nicht kleiner als die Summe aller Nachfragen sein kann, d.h.

$$-\sum_{d_i 0} d_i \geq \sum_{d_i 0} d_i = v \quad .$$

Nimm an, dass die Kosten jedes gerichteten Weges von einem Angebotsknoten zu einem Nachfrageknoten nichtnegativ sind, so dass es eine Optimallösung gibt, wo die Nachfrage-Ungleichungen mit Gleichheit erfüllt sind. Führe einen Knoten s (die Quelle) ein mit je einer Kante (s,i), $b_{si} := -d_i$, $a_{si}=0$, zu jedem Angebotsknoten i, und einen Knoten t (die Senke) mit je einer Kante (j,t), $b_{jt} := +d_i$, $a_{jt}=0$ zu jedem Nachfrageknoten j. So sind denn in allen Knoten ausser s,t die Flusserhaltungsgleichungen erfüllt. Dann liefert ein 'Minimum Cost Flow'-Problem mit Wert v eine Lösung des Transportproblems.

Umgekehrt ist ebenso klar, dass das 'Minimum Cost Flow'-Problem ein Transportproblem mit Kapazitäten ist (für einen gewünschten Flusswert v setze $d_s := -v$, $d_t := +v$.

In (5.2.2) werden wir das 'Minimum Cost Flow'-Problem auf ein Hitchcock-Problem mit Kapazitäten zurückführen. Diese Art der Transformation ist gut bekannt und in unserem Fall [L1] entnommen. Vergleiche dazu auch [F2]. In (5.2.3) geben wir dafür ein Beispiel. In (5.2.4) zeigen wir die Uebersetzung des Hitchcock-Problems in ein Intersektions-Problem von zwei Partitions-Polymatroiden mit Kapazitäten. In (5.2.5) erwähnen wir den Spezialfall des Maximalflussproblems.

5.2.2 Wir transformieren das Originalproblem in die Form des sogenannten 'Hitchcock'-Problems (vgl. [F1] oder [F2]):

Sei $m := |V|$. Wir definieren einen neuen Graphen $\overline{G}(\overline{V},\overline{E})$ mit 2.m Knoten, wobei jeder Knoten $i \in V$ durch zwei Knoten i^1 und i^2, beide in $\overline{V}$, repräsentiert wird.

Ordne zu jedem Knoten $i^k \in \overline{V}$, $k=1,2$, $i \in V$, einen Wert d_i^k zu, so dass

$$d_i^k := \begin{cases} 2v, & \text{falls } (i=s \text{ und } k=1) \text{ oder } (i=t \text{ und } k=2) \\ v & \text{sonst} \end{cases}$$

Beachte: Für alle $i \in V$ gilt $d_i^2 - d_i^1 = d_i$.

In $\overline{G}$ sind nun die folgenden Kanten definiert:

- für jedes $i \in V$ eine Kante (i^1,i^2) mit Kapazität $\overline{b_{i^1 i^2}} := \min\{d_i^1, d_i^2\}$ und Kosten $\overline{a_{i^1 i^2}} := 0$.
- für jede Kante (i,j) in G eine Kante $(i^1 j^2)$ in $\overline{G}$ mit Kapazität $\overline{q_{i^1 j^2}} := \min\{d_i^1, d_j^2, b_{ij}\}$ und Kosten $\overline{a_{i^1 j^2}} := a_{ij}$.

Sicher ist $\overline{G}$ ein bipartiter Graph. Sei S^1 (bzw. S^2) die Knotenmenge der 'linken' (bzw. der 'rechten') Seite von $\overline{V}$, somit $S^1 \dot{\cup} S^2 = \overline{V}$. Ein Beispiel für diese Transformation wird in (5.3.3) gezeigt.

Betrachte nun das folgende lineare Programm, ein 'Hitchcock'-Problem:

$$\text{Maximiere } \sum \{-a_{ij} \cdot x_{ij} : (i,j) \in \overline{E},\ x_{ij} \in \mathbb{Z}\},$$

wobei

$$\sum_{j \in S^2} x_{ij} = d_i^1 \quad \text{für alle } i \in S^1$$

(ii)

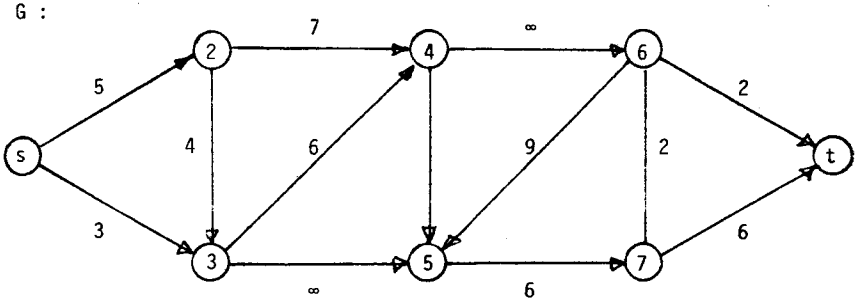
$$\sum_{i \in S^1} x_{ij} = d_j^2 \quad \text{für alle } j \in S^2$$

$$0 \le x_{ij} \le \overline{b_{ij}} \quad \text{für alle } (i,j) \in \overline{E}$$

Es ist leicht nachzuprüfen, dass jede zulässige Lösung von (i) eine zulässige Lösung von (ii) zur Folge hat, und umgekehrt.

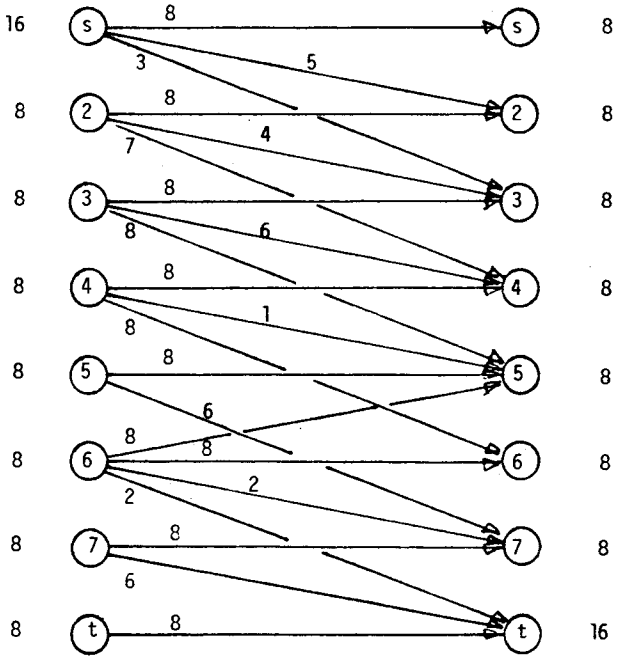
5.2.3 Transformation des Graphen G eines 'Minimal Cost Flow' Problems in den Graphen $\overline{G}$ eines 'Hitchcock'-Problems (die Zahlen auf den Kanten sind jeweils die Kapazitäten, die Zahlen in den Knoten die Nummern der Knoten.

G :



v = 8

$\overline{G}$:



5.2.4 $\overline{E}$ kann in zwei 'natürlichen' Arten partitioniert werden:

$$\overline{E}^1_i := \{(i,j): (i,j)\in\overline{E},\ j\in S^2\} \text{ für alle } i\in S^1$$

$$\overline{E}^2_j := \{(j,i): (j,i)\in\overline{E},\ j\in S^1\} \text{ für alle } i\in S^2$$

Somit gilt $\mathbf{0}\{\overline{E}^1_i: i\in\{1,..,m\}\} = \mathbf{0}\{\overline{E}^2_i: i\in\{1,..,m\}\} = \overline{E}$.

Sei nun $b: L_{\overline{E}} \to \mathbb{R}$ eine Funktion definiert wie folgt:

$$b(A) := \begin{cases} 0 & \text{falls } A=\emptyset \\ \sum\{\overline{b_{ij}}: (i,j)\in A\}, & \text{andernfalls} \end{cases} .$$

Für k=1,2, sei $f_k: L_{\overline{E}} \to \mathbb{R}$ eine Funktion definiert wie folgt:

$$f_k(A) := \sum \{\min\{b(A\cap E^k_i), d^k_i\}: i\in\{1,..,m\}\} .$$

Dann ist f_k eine β_0-Funktion und $P_k := P(K_E, f_k) =$

$= \{x\in R^{\overline{E}}_+: x(A)\le f_k(A) \text{ für alle } A\in K_E\}$ ist ein Partitions-Polymatroid mit ganzzahligen Kapazitäten.

Betrachte nun das folgende lineare Programm:

$$\text{Maximiere } \sum \{-\overline{a_{ij}}.x_{ij} : x_{ij}\in\mathbb{R},\ (i,j)\in\overline{E}\}$$

wobei

(iii)
$$x \in P_1 \cap P_2$$
$$x(E) = \sum \{d^1_i: i\in S^1\} = \sum \{d^2_i: i\in S^2\} .$$

Der Polyeder definiert in (iii) ist sicherlich derselbe wie jener definiert in (ii). Somit ist das 'Minimum Cost Flow' Problem reduzierbar auf ein Hitchcock-Problem,welches als äquivalent zu einem Intersektions-Problem von zwei Partitions-Polymatroiden mit Kapazitäten erkannt wurde. Das letztere Problem kann mit Algorithmus (4.6) mit $O(m^8.K)$ Rechenschritten gelöst werden, wenn wir die in (5.1.3) berechnenten Komplexitäten berücksichtigen.

5.2.5 Wenn wir die letzte Gleichung in (iii) weglassen und für v einen genügend grossen Wert einsetzen, und $a_{ij}=1$ für alle (i,j) setzen, haben wir eine Beschreibung des Maximal-Fluss-Problems als ein Intersektions-Problem von zwei Partitions-Polymatroiden. Mit Algorithmus (2.18) können wir dieses Problem mit $O(m^5)$ Rechenschritten lösen, wie aus (2.23) und (5.1.3) hervorgeht. Diese Schranke ist dieselbe wie diejenige für den Algorithmus in [E3].

5.3 Das Kardinalitäts-Polymatroid (Edmonds und Giles):

5.3.1 Sei $f: L_E \to \mathbb{R}$ eine Funktion so dass $f(S) = g(|S|)$,

wobei $g: \{0,1,..,|E|\} \to \mathbb{R}$ eine nichtnegative, nichtabnehmende, konkave Funktion ist, das heisst

$$g(i) = 0 + h(1) + h(2) + \ldots + h(i) \quad ,$$

wobei $h: \{1,2,..,|E|\} \to \mathbb{R}$ eine Funktion ist, so dass

$$h(1) \geq h(2) \geq \ldots \geq h(|E|) \geq 0 \quad .$$

Sicher ist f nichtabnehmend und $f(\emptyset)=0$.

Seien nun $A,B \in K_E$. Nimm ohne Beschränkung der Allgemeinheit an, dass $|A| \geq |B|$. Dann gilt $|A \cup B| \geq |A| \geq |B| \geq |A \cap B|$, und wegen $|A \cup B| - |A| = |B| - |A \cap B|$ und der Definition von f, g und h gilt

$$f(A \cap B)+f(A \cup B) = \sum_{k=1}^{|A \cap B|} h(k) + \sum_{k=1}^{|A \cup B|} h(k)$$

$$= 2. \sum_{k=1}^{|A \cap B|} h(k) + \sum_{k=|A \cap B|+1}^{|A|} h(k) + \sum_{k=|A|+1}^{|A \cup B|} h(k)$$

$$\leq 2. \sum_{k=1}^{|A \cap B|} h(k) + \sum_{k=|A \cap B|+1}^{|A|} h(k) + \sum_{k=|A \cap B|+1}^{|B|} h(k)$$

$$= \sum_{k=1}^{|A|} h(k) + \sum_{k=1}^{|B|} h(k)$$

$$= f(A) + f(B) \quad .$$

Somit ist f eine β_0-Funktion und $P := P(K_E, f)$ ist ein Polymatroid, genannt ein Kardinalitäts-Polymatroid. Falls h ganzzahlig ist, ist auch f ganzzahlig und somit P ein ganzzahliges Polymatroid.

5.3.2 Subroutine:

Um für ein $x \in \mathbb{R}^E$ zu testen, ob $x \in P$ oder nicht, benötigen wir $O(|E|^2)$ Rechenschritte:

- Gib jedem Element $e \in E$ einen Index, so dass $x_{e_1} \geq x_{e_2} \geq .. \geq x_{e_{|E|}}$.

- Sei $E_i := \{e_1, e_2, .., e_i\}$ für alle $i \in \{1, .., |E|\}$.

- Dann gilt die folgende Aussage:

$$x \in P \iff \begin{cases} x_e \geq 0 & \text{für alle } e \in E \\ x(E_i) \leq f(E_i) & \text{für alle } i \in \{1, .., |E|\}. \end{cases}$$

Beweis:

=>: Dieser Teil des Beweises ist klar.

<=: Wir werden mit Induktion nach dem Index i die folgende Aussage beweisen, aus welcher für $x \geq 0$ sofort $x \in P$ folgt.

(*) Falls $x(E_j) \leq f(E_j)$ für alle $j \leq i$, so gilt $x(A) \leq f(A)$ für alle $A \subseteq E_i$.

Sicher gilt (*) für i=1, da $x(\emptyset)=0$ und $E_1 = e_1$.

Nimm an, dass (*) für ein i gilt. Wir zeigen, dass (*) auch für i+1 gilt :

Sei $A \subseteq E_i$. Da $x_{e_{i+1}} \leq x_{e_j}$ für alle $j \leq i$, gilt

$x(A \cup e_{i+1} - e_j) \leq x(A) \leq f(A)$ für alle $j \leq i$ und $A \subseteq E_i$.

Aber dann gilt $x(A) \leq f(A)$ für alle $A \subset E_{i+1}$ und da $x(E_{i+1}) \leq f(E_{i+1})$, gilt $x(A) \leq f(A)$ für alle $A \subseteq E_{i+1}$.

□

6 Implementation und offene Probleme

6.1 Allgemeine Gedanken zur Implementation

Wie ein Vergleich der Matroid-Intersektions-Algorithmen mit den Polymatroid-Intersektions-Algorithmen zeigt, ist die Implementation der Polymatroid-Intersektionsalgorithmen nicht viel komplizierter als diejenige der entsprechenden Matroid-Intersektions-Algorithmen. Letztere wurden vom Verfasser für zwei Matrix-Matroide implementiert.

Ein Schlüsselproblem ist dabei die Darstellung der Familien von verschachtelten Mengen für den gewichteten Intersektions-Algorithmus. Wir werden uns damit im nächsten Abschnitt eingehender befassen.

Wie in (3.34) erwähnt, ist die Komlexität des gewichteten (und natürlich auch des ungewichteten) ganzzahligen Polymatroid-Intersektionsalgorithmus kleiner als diejenige eines 'Umweg'-Algorithmus, welcher zuerst das Originalproblem in ein Matroid-Intersektions-Problem verwandelt (vgl. (1.13)) und dann als Unteralgorithmus den entsprechenden Matroid-Intersektionsalgorithmus verwendet.
Es wird somit von Interesse sein, ein Polymatroid-Intersektions-Problem mit dem direkten Algorithmus und mit einem Umweg-Algorithmus zu lösen. Wir würden dabei die Anzahl der Augmentationen der Primallösung, der Revisionen der Duallösung, sowie den Speicherbedarf und die Laufzeit beider Algorithmen vergleichen.

Es wird ebenso von Interesse sein, Netzwerk-Fluss-Probleme mit dem entsprechenden Polymatroid-Intersektions-Algorithmus zu lösen und die Effizienz mit derjenigen von bereits verfügbaren Algorithmen zu vergleichen, etwa dem Hitchcock-Algorithmus, dem Maximalfluss-Algorithmus oder dem 'Out of Kilter'-Algorithmus (vgl. [F2])

6.2 Darstellung der Familien von verschachtelten Mengen im Algorithmus (3.19)

Im Verlaufe des Algorithmus (3.19) werden die Elemente von E in zwei Familien von verschachtelten Teilmengen von E zusammengefasst (vgl. (3.14)), je eine Familie $\mathfrak{R}_k$ pro Polymatroid P_k, k=1,2.

Einer Menge $A \in \mathfrak{R}_k$, k=1 oder k=2, sind keine, eine oder mehrere echte Teilmengen untergeordnet, sowie mindestens ein "freistehendes" Element aus E, welches nicht in den untergeordnenten Teilmengen enthalten ist (letzteres folgt daraus, dass die Mengen in $\mathfrak{R}_k$ f_k-nichtseparabel sind (vgl. (3.14)).

Die Menge E muss nicht unbedingt zu $\mathfrak{R}_k$ gehören. Im Verlaufe der Berechnungen wird sie aber benötigt. Deshalb und auch zur Realisierung eines einfachen Initialzustandes wird die Menge E in der Darstellung der Familie $\mathfrak{R}_k$ berücksichtigt.

Auf einer beliebigen Stufe des durch die Verknüpfungen in $\mathfrak{R}_k$ entstehenden Baumes sieht die Situation somit folgendermassen aus, für eine Menge $A \in \mathfrak{R}_k$ (beachte, dass $S_x=\{e: x_e>0\}$):

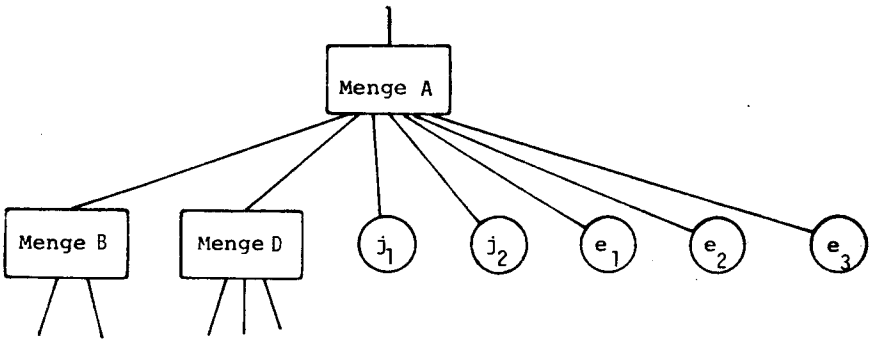


Fig. 6.1: Baumstruktur der Verknüpfungen in der Familie $\mathfrak{R}_k$, k=1,2. Die Mengen sind aus $\mathfrak{R}_k$, die Elemente j_1,j_2 aus S_x und die Elemente e_1,e_2,e_3 aus $E-S_x$. Es gilt ferner (vgl. (3.14)) $V_A^k=B \cup D$ und $A-V_A^k=\{j_1,j_2,e_1,e_2,e_3\}$.

Wir möchten nun möglichst einfach und schnell fplgende Operationen durchführen können:

a) Pro Menge $A \in \mathcal{R}_k$ bestimme V_A^k, d.h. die Vereinigung aller Teilmengen von A in $\mathcal{R}_k$, und bestimme alle Elemente aus $A-V_A^k$. (Vergleiche dazu (3.28): wir berechnen damit das kontrahierte Polymatroid P' aus P, und ebenso die minimale f'-kritische Menge eines Elementes $e \in E$ bezüglich x aus der minimalen f-kritischen Menge von e bezüglich x (vgl.(2.7)).)

b) Einfügen einer Menge in die Familie; entfernen einer Menge aus der Familie. (Vergleiche dazu Schritt 2.2 in Algorithmus (3.19).

Für eine Computerlösung erscheint es nun sinnvoll, folgende Verknüpfungen in diesem Baum zu speichern:

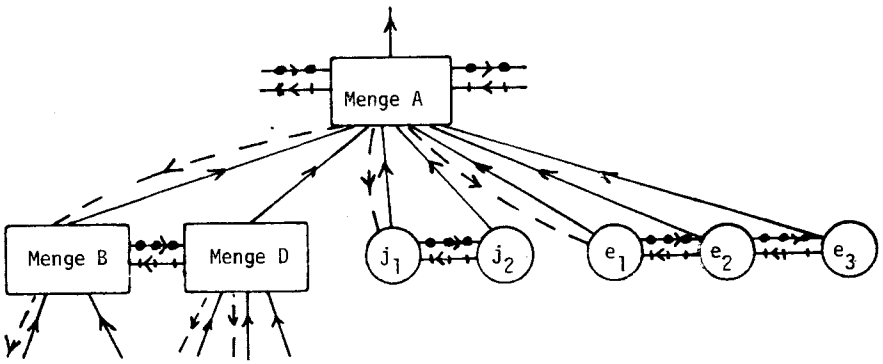


Fig. 6.2: Zu speichernde Verknüpfungen in der Baumstruktur:

——→—— : Uebergeordnete Menge (pro Menge oder Element)

— —→— : Erste untergeordnete Menge / erstes untergeordnetes Element aus S_x / erstes untergeordnetes Element aus $E-S_x$ (pro Menge)

—•—•→•— : Nachfolgende Menge / nachfolgendes Element aus S_x / nachfolgendes Element aus $E-S_x$, auf gleicher Stufe, d.h. der gleichen Menge direkt untergeordnet (pro Menge oder Element)

—←—←— : Wie —•—•→•— , aber 'vorherige' statt 'nachfolgende'

Die Berechnungen für a) können nun über die Verbindungen (eig. 'Pointers' oder Nummern von Elementen oder Mengen) — → — und ⟷ ausgeführt werden.

Die Berechnungen für b) werden durch folgende Subroutinen ausgeführt:

S1M : Lösche die Verbindung einer Menge B zur direkt übergeordneten Menge A: Dabei werden die Verbindungen ⟷ und ⟷ 'links' und 'rechts' von B benötigt: Vorgänger und Nachfolger von B werden 'kurzgeschlossen' und, falls es sich bei B um die erste untergeordnete Menge von A handelte, muss die Verbindung — →— auf die nächste Menge zeigen (ggf. ist die Verbindung dann ein 'Nullzeiger').

S2M : Schaffe eine Unterordnung der Menge B unter eine Menge A: B wird als erste untergeordnete Menge von A eingefügt. Die Verbindung — —>— muss also auf B zeigen und, falls es schon mindestens eine untergeordnete Menge von A gab, werden die Verbindungen ⟷ und ⟷ von B zur bisherigen ersten untergeordneten Menge von A hergestellt.

S1J/S1E : Analog zu S1M mit $j \in S_x / e \in E\text{-}S_x$ anstelle der Menge B.

S2J/S2E : Analog zu S2M mit $j \in S_x / e \in E\text{-}S_x$ anstelle der Menge B.

Bemerkung: b) könnte auch ohne die Verbindung ⟷ realisiert werden, allerdings mit grösserem Rechenaufwand für die Routinen S1M/S1J/S1E .

Der Initialzustand der Baumstruktur ist nun folgender:

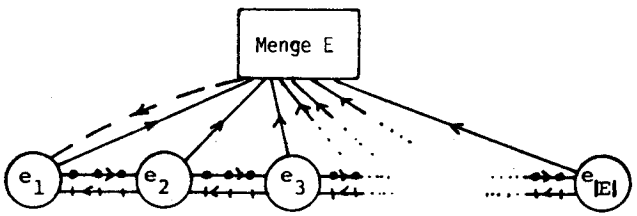


Fig. 6.3: Initialzustand der Familie. E ist die einzige Menge der Baumstruktur, die keine übergeordnete Menge besitzt.

6.3 Offene Probleme

Das erste Problem, welches wir hier erwähnen möchten, ist die Subroutine zur Erkennung, ob ein Vektor Element des Polymatroides ist oder nicht (vgl. 1.24)), bzw. ihre Komplexität. In Kapitel 5 erwähnten wir Beispiele von Polymatroiden mit einer 'gutartigen' Subroutine. Wir suchen nun noch mehr Polymatroide mit einer gutartigen Subroutine. Wir sind aber ebenso interessiert an Polymatroiden mit einem Subroutinen-Problem, welches zur Klasse der NP-vollständigen Probleme gehört. Um Polymatroide beider Arten zu erhalten, scheint es hilfreich, eine Studie über Hypergraphen und der Polymatroide, welche von diesen Strukturen abgeleitet werden können, durchzuführen. Solche Polymatroide wären Verallgemeinerungen des graphischen Matroids.

Eine wichtige Eigenschaft zur Entwicklung von ganzzahligen Polymatroid-Intersektions-Algorithmen war die Transitivitätseigenschaft von Polymatroiden (2.12). Wir hätten gerne mehr Einsicht in die geometrische Bedeutung dieser Eigenschaft.

Für die gewichteten Probleme (Kapitel 3 und 4) ist immer noch die Frage offen, ob es einen Algorithmus gibt, dessen Komplexität höchstens proportional zu logK anstelle von K ist.

Falls $|E|\leq 2$ und P_1, P_2 zwei Polymatroide in $\mathbb{R}^E$ sind, ist leicht einzusehen, dass $P_1 \cap P_2$ immer noch ein Polymatroid ist. Welches ist nun die kleinste Zahl k, so dass, falls $|E|=k$, es zwei Polymatroide P_1, P_2 in $\mathbb{R}^E$ gibt, so dass $P_1 \cap P_2$ kein Polymatroid ist? Es ist bekannt, dass $P_1 \cap P_2$ im allgemeinen kein Polymatroid ist.

Karzanov [K2] löst das Maximalflussproblem in Netzwerken ohne die Konstruktion von vergrössernden Wegen. Gibt es ein ähnliches Konzept für ganzzahlige Polymatroid-Intersektions-Probleme, um einen Vektor mit maximaler Komponentensumme zu bestimmen?

Ein weiteres Problem ist die Suche nach einer Verallgemeinerung der hier präsentierten Algorithmen, um Matching-Probleme (vgl. (1.6.e)) und Polymatroid-Intersektions-Probleme mit dem gleichen Algorithmus lösen zu können. Dies ist von Interesse, weil auch ein Matching maximaler Kardinalität in einem Graphen mit dem Konzept der Suche nach vergrössernden Wegen erhalten werden kann (vgl. [E5]).

Lebenslauf

Ich wurde am 12. Mai 1952 in Rüti/ZH geboren.

Nach Absolvierung der Primalschule und des Gymnasiums erhielt ich 1971 das Maturitätszeugnis Typ B von der Kantonsschule Zürcher Oberland in Wetzikon/ZH.

Ab 1971 studierte ich Mathematik an der ETH in Zürich und erwarb im Herbst 1977 das Diplom als Mathematiker. Die Diplomarbeit befasste sich mit Matroid-Intersektions-Algorithmen.

Im Frühjahr 1978 bekam ich die Gelegenheit, im Rahmen einer Forschungsarbeit am Institut für Operations Research der ETH den Themenkreis der Matroide und Polymatroide eingehend zu studieren. Diese Arbeit ist das Hauptresultat der Untersuchungen.

Von 1971 bis 1974 war ich als Werkstudent bei IBM Zürich auf dem Gebiet des Systems Engineering tätig. Seit 1974 arbeite ich als selbständiger Berater auf diesem Gebiet, vor allem in Verbindung mit der Firma HESCO in Rüti/ZH.

Seit 1973 bin ich verheiratet und wir haben seit 1979 ein Kind.