Martina PRESSNIG

Inverse Engpass Zuordnungsprobleme

DIPLOMARBEIT

zur Erlangung des akademischen Grades einer Diplom-Ingenieurin

Diplomstudium Technische Mathematik

Graz University of Technology
Technische Universität Graz

Betreuer:
O.Univ.-Prof. Dr. Rainer BURKARD

Institut für Optimierung und Diskrete Mathematik (Math B)

Graz, im Mai 2011

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während des Studiums und der Erarbeitung der vorliegenden Diplomarbeit begleitet und unterstützt haben.
Ein besonderer Dank gilt Herrn Prof. Rainer Burkard, der mit sehr viel Engagement, guten Ideen und Geduld meine Diplomarbeit betreut hat.
Zudem möchte ich mich bei meiner Familie bedanken, die mich nicht nur finanziell, sondern auch moralisch immer unterstützt und mir den Rücken gestärkt hat.

Für meine Mutter

Abstract

Diese Arbeit beschäftigt sich mit inversen Engpass-Zuordnungsproblemen. Bei einem inversen Engpass-Zuordnungsproblem ist eine Permutation φ gegeben. Das Ziel ist es die Kosten so wenig wie möglich zu verändern, sodass φ zu einer optimalen Lösung für das Engpass-Optimierungsproblem wird. Die Veränderung kann mit Hilfe verschiedener Normen erfasst werden, zum Beispiel mittels der l_1 Norm, der l_∞ Norm und der Hamming Distanz.
Im ersten Abschnitt wird das inverse Engpass-Zuordnungsproblem unter der Summen-Hamming-Distanz mit oberen und unteren Schranken untersucht und ein Lösungsalgorithmus für dieses Problem präsentiert. Später wird das inverse Engpass-Zuordnungsproblem mit der Engpass-Hamming-Distanz betrachtet und gezeigt, dass dieses Problem polynomiell lösbar ist. Sodann werden noch einige weitere inverse Engpass-Zuordnungsprobleme mit der l_1 und der l_∞ Norm vorgestellt. Schließlich wird überprüft, ob ein Zusammenhang zwischen einem Summen-Zuordnungsproblem und einem Engpass-Zuordnungsproblem existiert.

In this thesis the inverse bottleneck assignment problem is considered. Given is a feasible permutation φ, the aim is to modify the cost function as little as possible that φ becomes an optimal solution to the bottleneck assignment problem. The modification can be measured by different norms, for example by the l_1 norm, by the l_∞ norm and by the Hamming-distance.
In the first part the inverse bottleneck assignment problem under the sum-Hamming-distance with lower and upper bounds is examined and an algorithm to solve this problem is presented. The second part deals with the inverse bottleneck assignment problem under the bottleneck-Hamming-distance and it is shown that the problem is solvable in polynomial time. Furthermore some other inverse bottleneck problems with the l_1 and the l_∞ norm are presented. Finally, it is discussed, if a relationship between inverse bottleneck assignment problems and inverse sum assignment problems exists.

Inhaltsverzeichnis

1 Überblick

1.1 Einleitung

Inverse Optimierungsprobleme sind ein relativ neues und ständig wachsendes Forschungsgebiet im Bereich der Optimierung. Während man in der Standardversion von Optimierungsaufgaben nach der bestmöglichen Lösung sucht, beschäftigt man sich bei der inversen Optimierung mit der folgenden Fragestellung: Gegeben ist ein Optimierungsproblem und eine zulässige Lösung für dieses Problem. Es wird nun nach einer Lösung gesucht, bei der die Problemdaten in möglichst geringem Ausmaß modifiziert werden, sodass die vorgegebene Lösung optimal für das Optimierungsproblem mit modifizierten Daten wird.
Auf dem ersten Blick scheinen inverse Optimierungsprobleme ein sehr theoretisches Gebiet zu sein, allerdings sind solche Probleme auch häufig in der praktischen Anwendung zu finden.
Ein Beispiel für ein inverses Optimierungsproblem ist das inverse Standortproblem [14]. Das Standortproblem beschäftigt sich mit der Suche nach einem oder mehreren optimalen Standorten für neue Lagerhäuser, sodass der Abstand zwischen dem Kunden zu dem nächsten Lagerhaus minimal wird. In der Wirklichkeit existieren die Lagerhäuser bereits und eine Neuerrichtung wäre zu teuer. Deshalb will man durch eine Verbesserung der Verkehrsbedingungen die Distanzen von dem Lagerhaus zum Kunden verändern, sodass die gegenwärtige Position der Lagerhäuser optimal wird.
Ein weiteres Beispiel ist das inverse Engpass-Zuordnungsproblem, das in dieser Arbeit genauer untersucht wird. Bei einem Engpass-Zuordnungsproblem sollen n Kunden von n Lagerhäusern beliefert werden. d_{ij} ist dabei die Transportzeit, die benötigt wird, um vom i-ten Lagerhaus den j-ten Kunden zu beliefern. Es wird nach einer Zuordnung gesucht, bei der jedes Lagerhaus genau einen Kunden beliefert und die längste Transportzeit soll dabei minimal werden. Bei dem zugehörigen inversen Problem ist eine zulässige Zuordnung vorgegeben, die optimal werden soll. Dies kann man wiederum durch eine Verbesserung der Verkehrsbedingungen (z.B. Sanierung der Straßen) erreichen.

Als erstes wurden inverse Probleme von Geophysikern untersucht. Die Arbeit von Tarantola [25] liefert eine umfassende Auseinandersetzung mit inversen Problemen. Erstmalig wurden inverse Optimierungsprobleme vom mathematischen Standpunkt aus von Burton und Toint [9] im Jahr 1992 gelöst. In der Folge wurden zahlreiche weitere inverse Fragestellungen auf dem Gebiet der Optimierung erforscht. In der angeführten Liste sind einige Typen von inversen Optimierungsproblemen angegeben und es werden auch einige Beispiele für Arbeiten, in denen die Probleme genauer untersucht werden, aufgelistet:

- Inverse lineare Optimierungsprobleme (Inverse linear programming problems), Ahuja und Orlin [2]
- Inverse minimale Kosten-Fluss Probleme (Inverse minimum cost flow problems), Ciurea und Deaconu [11]

- Inverse Kürzeste-Wege Probleme (Inverse shortest path problems), Burton und Toint [9]
- Inverse minimale spannende Baum Probleme (Inverse spanning tree problems), Zhang, Liu und Ma [29], Sokkalingam, Ahuja und Orlin [24] und Hochbaum [21]
- Inverse Standortprobleme (Inverse location problems), Burkard, Pleschiutschnig und Zhang [7], [8], Alizadeh und Burkard [4], Baroughi Bonab, Burkard und Gassner [5], Cai, Yang und Zhang [10],
- Inverse maximale Flußprobleme (Inverse maximum flow problems), Yang, Zhang und Ma [30]
- Inverse minimale Schnitt Probleme (Inverse minimum cut problems), Zhang und Cai [28]
- Inverse Zuordnungsprobleme (Inverse Assignment Problems), Ahuja und Orlin [2]

Eine gute Übersicht über inverse Optimierungsprobleme findet man bei Heuberger [20] und auch bei Ahuja und Orlin [2].

In dieser Arbeit wird nun das inverse Engpass-Zuordnungsproblem (engl. Bottleneck Assignment Problem) genauer untersucht und nach Lösungsverfahren dafür gesucht. Zuordnungsprobleme können verschiedene Zielfunktionen besitzen, deshalb wird in einer kurzen Einführung das lineare Summen-Zuordnungsproblem und das Engpass-Zuordnungsproblem erklärt. Weiters wird erläutert, worum es sich bei inversen Problemen handelt.
Es werden einige verschiedene Versionen des inversen Bottleneck-Assignment Problems näher untersucht und für diese nach einer Lösungsmethode, die eine Optimallösung auffindet, gesucht.

1.2 Zuordnungsprobleme

Unter Zuordnungs- oder Assignment-Problemen versteht man Probleme, bei denen n Jobs n Maschinen zugeordnet werden, sodass alle Jobs bearbeitet werden und jede Maschine genau einen Job zugeteilt bekommt.
Typische Zuordnungsprobleme sind:

- Zuordnung von Personen auf Arbeitsplätze, Räume, Fahrzeuge...
- Zuordnung von Lehrpersonen auf Personengruppen z.B. Stundenpläne
- Zuordnung von Kränen auf Baustellen, Piloten auf Flugzeuge usw.

Zuordnungen können auf einige verschiedene Arten darstellt werden [6].
Eine Zuordnung φ ist eine bijektive Abbildung zwischen der Menge der Jobs und der Menge der Maschinen und kann als Permutation dargestellt werden. Eine Permutation φ kann wie folgt

präsentiert werden: $\begin{pmatrix} 1 & 2 & ... & n \\ \varphi(1) & \varphi(2) & ... & \varphi(n) \end{pmatrix}$, das heißt, 1 wird $\varphi(1)$ zugeordnet, 2 wird $\varphi(2)$ zugeordnet, ... und n wird $\varphi(n)$ zugeordnet. Mit Hilfe einer Permutationsmatrix $X_\varphi = (x_{ij})$ mit $x_{ij} = \begin{cases} 1, & \text{wenn } j = \varphi(i) \\ 0, & \text{sonst} \end{cases}$ kann nun eine Lösung für das Zuordnungsproblem dargestellt werden.

Ein Zuordnungsproblem besteht aus einer Zielfunktion, die minimiert oder maximiert werden soll, und aus der Menge der zulässigen Zuordnungen.
Um zu garantierten, dass bei einem Zuordnungsproblem alle Jobs bearbeitet werden und jeder Maschine genau ein Job zugewiesen wird, werden noch Nebenbedingungen eingeführt:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, .., n) \tag{1.2.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, .., n) \tag{1.2.2}$$

$$x_{ij} \in \{0, 1\} \quad \text{für alle} \quad i, j = 1, 2, .., n \tag{1.2.3}$$

Die ersten zwei Gleichungen garantieren, dass in jeder Zeile bzw. jeder Spalte der Permutationsmatrix genau ein Eintrag gleich Eins ist. (1.2.3) besagt, dass die Einträge in der Permutationsmatrix nur 0 oder 1 sind.

Schließlich ist es noch möglich, ein Zuordnungsproblem mit Hilfe von Graphen darzustellen. Dazu werden folgende Definitionen benötigt:

Definition 1 *Ein Graph $G = (U, V; E)$ heißt bipartit, wenn die Knotenmengen U und V disjunkt sind, sodass zwischen den Knoten innerhalb beider Teilmengen keine Kanten verlaufen.*

Gegeben sei ein bipartiter Graph $G = (U, V, E)$ mit $|V| = |U| = n$, U ist die Menge der Jobs, V die Menge der Maschinen und $|E| = m$. Die Kante $(i, j) \in E$ existiert genau dann, wenn der Job i der Maschine j zugeordnet werden kann, dh. $j = \varphi(i)$ vgl. Abbildung

Definition 2 *$M \subseteq E$ heißt Matching, falls alle Knoten des Graphens mit höchstens einer Kante des Matchings verbunden sind.*
Ist ein Knoten $i \in V$ von G inzident zu einer Matchingkante, so nennt man i einen gematchten Knoten.
Die Kardinalität eines Matchings M wird mit $|M|$ bezeichnet.
M heißt maximales Matching, wenn man keine weitere Kante e aus E zu M hinzufügen kann, sodass M zusammen mit e ein Matching ist.
Eine Matching M von G nennt man Matching maximaler Kardinalität, falls $|M|$ maximal ist.
Ein perfektes Matching M ist ein Matching, bei dem jeder Knoten $i \in V$ von M gematcht ist.

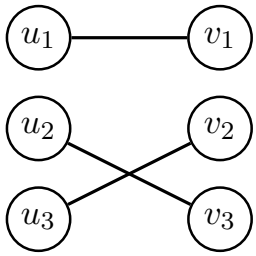


Abbildung 1.1: Zulässige Zuordnung bei einem bipartiten Graphen

1.3 Lineare Zuordnungsprobleme

Bei einem linearen Zuordnungsproblem liegt das Interesse nicht an einer beliebigen zulässigen Lösung des Problems, sondern es soll die Zielfunktion optimiert werden. Bei der Zielfunktion wird die Summe der Kosten entweder minimiert oder maximiert. Gegeben ist eine endliche Menge $I = \{1, .., n\}$ (z.B. Jobs), $J = \{1, .., n\}$ (z.B. Maschinen) und eine Kostenmatrix $C = (c_{ij})$. c_{ij} sind die Kosten, die entstehen, wenn Job $i \in I$ der Maschine $j \in J$ zugeteilt wird. Gesucht wird nach einer optimalen Zuordnung der Jobs zu den Maschinen, sodass jeder Job von genau einer Maschine ausgeführt wird und die Kosten dafür minimal sind.

Das lineare Zuordnungsproblem (LZP) lässt sich wie folgt definieren:
Gegeben ist ein bipartiter Graph $G = (V, E)$ und eine Kostenfunktion c_{ij} für alle Kanten $(i, j) \in E$. Sei $F = \{e \in E \mid F \text{ ist zulässige Zuordnung}\}$ und $\mathcal{F}$ die Menge aller zulässigen Zuordnung in G.

$$\min_{F \in \mathcal{F}} \sum_{e \in F} c(e) \tag{1.3.1}$$

Um eine Lösung für das lineare Zuordnungsproblem zu finden, wird nach einem perfekten Matching im bipartiten Graph $G = (V, E)$ gesucht, sodass die Summe der Kosten der Kanten im Matching so klein wie möglich ist.

Ein lineares Zuordnungsproblem kann auch als ganzzahliges lineares Programm formuliert

werden:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n \\ & x_{ij} \in \{0,1\} \end{aligned} \tag{1.3.2}$$

Diese Darstellung ist äquivalent zu der Darstellung 1.3.1, da eine zulässige Zuordnung jeder Maschine genau einen Job zuweist.

Bemerkung Falls G bipartit ist, dann ist die Restriktionsmatrix vollständig unimodular und man kann statt $x_{ij} \in \{0,1\}$ einfach fordern, dass $x_{ij} \geq 0$ für alle $(i,j) \in E$ gilt.

Beispiel *[31] Eine Transportfirma hat in 5 Städten je einen leeren Lastwagen übrig und benötigt in 5 anderen Städten je einen solchen Lastwagen. Wie muß man die Fahrzeuge dirigieren, damit die gesamte Kilometeranzahl und damit der Transportkostenaufwand möglichst klein bleiben?*

Tabelle 1.1: Entfernung der Orte in km

Ausgangsorte i	Bestimmungsorte j				
	1	2	3	4	5
1	8	3	11	13	16
2	2	8	17	2	7
3	12	9	4	4	6
4	5	11	9	7	14
5	6	8	9	3	13

Die optimale Lösung für dieses Beispiel liefert als Zielfunktionswert 22. Folgende Zuordnung ist optimal:

- $1 \rightarrow 2$
- $2 \rightarrow 5$
- $3 \rightarrow 3$
- $4 \rightarrow 1$
- $5 \rightarrow 4$

1.4 Engpass-Zuordnungsprobleme

Beispiel *Nehmen wir an, dass in der Tabelle des vorhergehenden Beispiels nicht die Entfernungen in km, sondern die Zeit, die benötigt wird, um vom Ausgangsort zum Bestimmungsort zu kommen, angegeben ist. Unter der Voraussetzung, dass alle Fahrzeuge gleichzeitig wegfahren, wird jetzt nach dem frühestmöglichen Zeitpunkt, an dem alle Lastwagen in ihrem Bestimmungsort ankommen, gesucht.*

Bei diesem Problem handelt es sich nun um ein lineares Engpass-Zuordnungsproblem (LEPZP) oder lineares Bottleneck-Assignment Problem. Bei dem in Abschnitt 1.3 betrachteten linearen Zuordnungsproblem wurde die Summe der Einzelkosen gebildet und optimiert. Bei einem Engpass-Zuordnungsproblem wird nun die Summenzielfunktion des linearen Zuordnungsproblems durch eine Engpass-Zielfunktion ersetzt.

Mathematisch lässt sich das (LEPZP) nun wie folgt definieren:
Gegeben ist ein bipartiter Graph $G = (V, E)$ und eine Kostenfunktion c_{ij} für alle Kanten $(i, j) \in E$. Sei $F = \{e \in E \mid F \text{ ist zulässige Zuordnung}\}$ und $\mathcal{F}$ die Menge aller zulässigen Zuordnung in G.

$$\min_{F \in \mathcal{F}} \max_{e \in F} c(e) \tag{1.4.1}$$

Das lineare Engpass-Zuordnungsproblem kann auch mit Hilfe einer Permutationsmatrix $X = (x_{ij})$ formuliert werden:

$$\begin{aligned}
\min \quad & \max \ c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, .., n \\
& \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, .., n \\
& x_{ij} \in \{0, 1\} \quad i, j = 1, .., n
\end{aligned}$$

In diesem Kapitel wird beschrieben, welche Möglichkeiten es gibt um ein lineares Engpass-Zuordnungsproblem zu lösen. Die folgenden Überlegungen und Lösungsmethoden lehnen sich an Burkard, Dell'Amico und Martello [6] an.

Bemerkung: Die Funktion $\max_{e \in F} c(e)$ ist nicht linear und eine spezielle konkave Funktion, die nur endlich viele verschiedene Funktionswerte annehmen kann und daher auch beschränkt ist. Daher nimmt die Funktion ihr globales Minimum in mindestens einem Eckpunkt des zulässigen Bereiches an. Aus diesem Grund genügt es, bei einem Engpass-Zuordnungsproblem nur

zulässige Basislösungen zu betrachten, um eine optimale Lösung zu bestimmen [12].

Diese Behauptung führt zu einem wichtigen Lemma, welches sehr hilfreich ist, um ein lineares Engpass-Zuordnungsproblem zu lösen.

Lemma 1 *Sei $C = (c_{ij})$ die Kostenmatrix eines linearen Engpass-Zuordnungsproblems. Dann gelten folgende zwei Aussagen:*

1. *Der optimale Wert des linearen Engpass-Zuordnungsproblems wird von einem der Kostenkoeffizienten c_{ij} angenommen.*
2. *Die optimale Lösung φ^* hängt nur von der relativen Reihenfolge der Kostenkoeffizienten ab, nicht von deren nummerischem Wert.*

Beispiel *Gegeben ist ein lineares Engpass-Zuordnungsproblem mit der Kostenmatrix*

$$C = \begin{pmatrix} \pi/2 & 0 & \sqrt{99} \\ \pi^{10} & -501 & 23 \\ 5 & -7 & 0.5 \end{pmatrix}$$

Durch Ordnen der Kostenkoeffizienten erhält man folgende Reihenfolge:

$$-501 < -7 < 0 < 0.5 < \pi/2 < 5 < \sqrt{99} < 23 < \pi^{10}$$

Das kleinste Element ist -501 und kann durch den Wert 0 ersetzt werden. Das zweitkleinste Element ist -7 und kann durch 1 ausgetauscht werden, usw. Aus diesem Grund kann die Kostenmatrix C durch folgende Matrix ersetzt werden:

$$\tilde{C} = \begin{pmatrix} 4 & 2 & 6 \\ 8 & 0 & 7 \\ 5 & 1 & 3 \end{pmatrix}$$

Mit anderen Worten heißt das, wenn die Kostenmatrix eines linearen Engpass-Zuordnungsproblems d verschiedene Einträgen besitzt, können diese Einträge durch die Werte $0, 1, 2, .., d-1$ ersetzt werden. Das lineare Engpass-Zuordnungsproblem mit der Kostenmatrix $\tilde{C}$ besitzt die Optimallösung $\tilde{c}_{11}, \tilde{c}_{22}, \tilde{c}_{33}$. Der größte Wert in dieser Lösung ist $\tilde{c}_{11} = 4$ und entspricht dem Wert $\pi/2$ in der ursprünglichen Kostenmatrix C.

Um ein lineares Engpass-Zuordnungsproblem zu lösen, existieren zwei Strategien, die häufig verwendet werden: Die Shortest-Augmenting-Path Methode und die Thresholder Methode, die hier kurz beschrieben werden.

Threshold Algorithmus Dieser Algorithmus basiert auf der Tatsache, dass der optimale Lösungswert identisch mit einem der n^2 Kostenkoeffizienten ist. Die Idee des Algorithmus ist, dass in der ersten Phase ein Grenzwert oder Threshold-Wert c^* gewählt wird und für diesen Grenzwert c^* eine Matrix $\overline{C}$ folgendermaßen definiert werden kann:

$$\overline{C} = \begin{cases} 1, & \text{wenn } c_{ij} > c^* \\ 0, & \text{sonst} \end{cases}$$

In der zweiten Phase wird überprüft, ob es in der Kostenmatrix $\overline{C}$ eine Zuordnung, deren Kosten 0 sind, gibt. Dazu wird ein bipartiter Graph $G = (U, V; E)$ mit $|U| = |V| = n$ und den Kanten $[i, j] \in E$, genau dann wenn $\overline{c} = 0$ ist, konstruiert.
Der kleinste Wert c^*, in dessen zugehörigem bipartiten Graphen es ein perfektes Matching gibt, ist dann der optimale Wert für das lineare Engpass-Zuordnungsproblem.
Es existieren einige Möglichkeiten diesen Algorithmus zu implementieren. Eine Möglichkeit ist die Kostenkoeffizienten in aufsteigender Reihenfolge zu ordnen oder es wird eine Binär-Suche in der ersten Phase des Algorithmus angewandt, um den Wert c^* zu finden. Dies führt dann zu einer Laufzeit von $O(T(n) \log n)$. $T(n)$ ist dabei die Laufzeit, die benötigt wird, um zu überprüfen, ob die Kostenmatrix $\overline{C}$ ein perfektes Matching besitzt.

Shortest Augmenting-Path Methode Eine weitere Möglichkeit ein lineares Engpass-Zuordnungsproblem zu lösen funktioniert mit Hilfe der Shortest-Augmenting-Path Methode. Dazu wird eine essentielle Definition benötigt:

Definition 3 *Ein Weg heißt augmentierend oder matchingvergrößernd bezüglich eines Matchings M, wenn die Kanten des Weges abwechselnd ungematcht (frei) bzw. Matchingkanten sind, beginnend und endend mit einer ungematchten Kante.*

Bei der Shortest-Augmenting-Path Methode wird mit einem Start-Matching M in $G = (U, V; E)$ begonnen. $U(M)$ sei die Menge der Kanten aus U, die von M gematcht werden. Dabei soll das Start-Matching M so gewählt werden, dass es die kleinsten Engpasskosten $c(M) = \max_{(i,j)\in M}$ unter allen Matchings, die genau dieselbe Knotenmenge $U(M)$ besitzen, hat. Mit Hilfe von matchingvergrößernder Wege soll nun das Matching M zu einem Matching M' vergrößert werden, sodass das vergrößerte Matching M' wiederum minimale Kosten unter allen Matchings, die die Menge $U(M')$ matchen, besitzt. Diese Vergrößerung wird erreicht mittels matchingvergrößernder Wege, die eine minimale Länge besitzen [13].

Eine Verbesserung der Laufzeit kann erzielt werden, indem man die Threshold Methode mit der Augmenting-Path Methode kombiniert. Diese Idee stammt von Gabow und Tarjan [16], die einen Algorithmus für das lineare Engpass-Zuordnungsproblem mit der Laufzeit $O(m\sqrt{n\ log\ n})$ für einen bipartiten Graphen G mit $2n$ Knoten und m Kanten entwickelten. Für dichte Graphen wurde diese Schranke von Punnen und Nair [23] verbessert.
Einen Überblick über die Zeitkomplexität der verschiedenen Algorithmen für das lineare Engpass-Zuordnungsproblem findet man in der folgenden Tabelle:

Methode	Komplexität	$m = \mathcal{O}(n)$	$m = \mathcal{O}(n^2)$
Threshold	$\mathcal{O}(n^2\sqrt{n/\log n})$	$\mathcal{O}(n^2\sqrt{n/\log n})$	$\mathcal{O}(n^2\sqrt{n/\log n})$
Gabow-Tarja	$\mathcal{O}(m\sqrt{n\log n})$	$\mathcal{O}(n\sqrt{n\log n})$	$\mathcal{O}(n^2\sqrt{n\log n})$
Punnen-Nair	$\mathcal{O}(n\sqrt{mn})$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2\sqrt{n})$

Tabelle 1.2: [6] Laufzeit der verschiedenen Algorithmen für das lineare Engpass-Zuordnungsproblem

Die Tabelle zeigt, dass der Threshold Algorithmus die beste Laufzeit für dichte Graphen besitzt. Die Methode von Gabow-Tarjan weist die beste Zeitkomplexität bei Graphen mit nur wenigen Kanten auf. Für alle anderen Graphen verfügt die Methode von Punnen-Nair über die beste Laufzeit.

1.5 Inverse Fragestellung

Wenn ein Optimierungsproblem gelöst werden soll, ist man normalerweise daran interessiert, die beste zulässige Lösung für die gestellte Aufgabe zu finden. In der Praxis jedoch treten häufig Probleme auf, bei denen man zwar in der Lage ist die optimale Lösung herauszufinden, jedoch diese nicht realisierbar ist. Ein Beispiel: Gegeben sei ein Straßennetzwerk und eine Fabrik darin. Das Ziel ist es, die Fabrik an einem Standort zu platzieren, sodass die Distanz zu allen Kunden minimal ist. Es könnte eine neue Fabrik an dem optimalen Standort neu errichtet werden, doch würden dadurch enorme Kosten entstehen. Deshalb wird das Straßennetzwerk modifiziert, sodass die Änderungen so gering wie möglich sind. Z.B. könnte durch eine Sanierung einiger Straßen die Fahrzeit auf diesen Strecken verringert werden.
Bei einem inversen Optimierungsproblem handelt es sich um die Aufgabe, bei der eine zulässige Lösung des zugrunde liegenden Optimierungsproblems vorgegeben ist. Man möchte bestimmte Daten in möglichst geringem Ausmaß modifizieren, sodass die vorgegebene zulässige Lösung optimal für das Optimierungsproblem mit den modifizierten Daten wird.

Formal lässt sich ein inverses Optimierungsproblem nun wie folgt formulieren [20]: Sei $\mathcal{F} \subseteq \mathbb{R}^n$ die Menge aller zulässigen Lösungen, $F \in \mathcal{F}$ eine beliebige zulässige Lösung, $c \in \mathbb{R}^n$ ein Kostenvektor und $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ die Zielfunktion. Weiters sind obere und untere Schranken $l, u \in \mathbb{R}^n$ für den Kostenvektor gegeben. Das 'Standard'-Optimierungsproblem kann dann wie folgt geschrieben werden:

$$\begin{aligned} \min_{F} \quad & f(c, F) \\ \text{s.t.} \quad & F \in \mathcal{F} \end{aligned} \tag{1.5.1}$$

Bei einem inversen Problem ist nun zusätzlich eine zulässig Lösung $F^* \in \mathcal{F}$ gegeben. Gesucht wird ein Kostenvektor $c^* \in \mathbb{R}^n$, sodass die Änderung zum ursprünglichen Kostenvektor c so gering wie möglich ist, und der Kostenvektor c^* muß innerhalb der vorgegebenen Schranken l, u liegen.
Somit kann ein inverses Optimierungsproblem folgendermaßen formuliert werden:

$$\begin{aligned} \min \quad & \|c^* - c\| \\ \text{s.t.} \quad & f(c^*, F^*) = \min\{f(c^*, F) : F \in \mathcal{F}\} \\ & l \leq c^* \leq u \\ & c^* \in \mathbb{R}^n \end{aligned} \tag{1.5.2}$$

Mit $\|.\|$ wird hier eine beliebige Vektornorm definiert. Als Maß für den Grad der Veränderung der Kosten werden üblicherweise die l_1, l_2, l_∞ Norm als auch der Hamming-Abstand, der die Anzahl der modifizierten Kanten angibt, verwendet.

- Die gewichtete l_1 Norm mit den Gewichten $w_i^+, w_i^- \geq 0$ für $i = 1, .., n$:

$$\|z\| = \sum_{i=1}^{n} (w_i^+ \max\{0, z_i\} + w_i^- \max\{0, -z_i\})$$

- Die gewichtete l_∞ Norm mit den Gewichten $w_i^+, w_i^- \geq 0$ für $i = 1, .., n$:

$$\|z\| = \max_{i=1..n} (w_i^+ \max\{0, z_i\} + w_i^- \max\{0, -z_i\})$$

- Die l_2 Norm:

$$\|z\| = \sqrt{\sum_{i=1}^{n} z_i^2}$$

- Eine weitere Möglichkeit ein inverses Optimierungsproblme zu lösen funktioniert mit Hilfe der Hamming-Distanz.

  Definition 4 *Die Hamming-Distanz $H(u, v)$ zwischen zwei Vektoren $u = (u_1, .., u_n)$ und $v = (v_1, .., v_n)$ ist gleich der Anzahl der Stellen, in denen sich die beiden Vektoren unterscheiden.*
  D.h. setzt man

$$h(u_i, v_i) = \begin{cases} 0, & \text{wenn } u_i = v_i \\ 1, & \text{sonst} \end{cases}$$

  dann folgt, dass

$$H(u, v) = |\{i \mid u_i \neq v_i, i = 1, .., n\}| = \sum_{i=1}^{n} h(u_i, v_i)$$

  ist.

In vielen Arbeiten, die zum Thema inverse Optimierung verfasst wurden, wird die l_1 Norm verwendet, da mit Hilfe dieser Norm die ganze Menge der Veränderungen modellierbar ist.
Wenn $w_i^+ = w_i^-$ gilt, wird die Norm symmetrisch gewichtet genannt.
Wenn $w_i^+ = w_i^- = 1$ gilt, so wird die Norm einheitlich gewichtet genannt (unit weight norm).
Ein unbeschränktes inverses Optimierungsproblem (unconstrained inverse problem) liegt vor wenn $l_i = -\infty$ und $u_i = \infty$ gilt. Wird ein Problem mit endlichen oberen und unteren Schranken betrachtet, so wird dieses Problem beschränktes inverses Optimierungsproblem (constrained inverse problem) genannt.

2 Inverses Engpass-Zuordnungsproblem

Im Folgenden wird das inverse Engpass-Zuordnungsproblem (inverses Bottleneck-Assignment Problem) näher betrachtet. Bei einem inversen Engpass-Zuordnungsproblem ist eine zulässige Lösung F^* gegeben. Das Ziel ist es die Kosten so wenig wie möglich zu verändern, sodass F^* zu einer optimalen Lösung für das Engpass Optimierungsproblem wird. In diesem Abschnitt werden einige verschiedene Varianten für das Problem formuliert. Durch die unterschiedlichen Maße für den Grad der Veränderungen der Kosten können verschiedene Probleme formuliert werden:

- Die Summe der Veränderungen der Kosten soll minimal werden
- Die größte Änderung soll minimal werden
- Die Anzahl der Kanten, die sich verändern, soll minimal werden
- Das größte Gewicht der veränderten Kanten soll minimal werden

Weiter wird noch überlegt, welche Veränderungen der Kosten zugelassen sind.

- Die Kosten auf den einzelnen Kanten dürfen nur erhöht werden
- Die Kosten auf den einzelnen Kanten dürfen nur verringert werden
- Verringerungen und Erhöhungen der Kosten sind erlaubt

Problem 1: Die Summe der Veränderungen der Kosten soll minimal werden
Das inverse Engpass-Zuordnungsproblem unter der l_1-Norm

Gegeben ist ein Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, obere und untere Schranken $l(e) \geq 0$, $u(e) \geq 0$ für alle $e \in E$, die angeben um wie viel die Kosten verändert werden dürfen, und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_{l_1}(c^*) := \sum_{e \in E} |c^*(e) - c(e)| \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Problem 2: Die größte Änderung soll minimal werden
Das inverse Engpass-Zuordnungsproblem unter der l_∞-Norm

Bei diesem Problem soll die größte Abweichung von den ursprünglichen Kosten zu den neuen Kosten, dh. $\max |c^*(e) - c(e)|$, minimal werden.
Es sind die gleichen Voraussetzungen wie bei dem vorangegangenen Problem gegeben. Die Aufgabe ist das folgende Problem zu lösen:

$$\begin{aligned} \min \quad & f_{l_\infty}(c^*) := \max_{e \in E} |c^*(e) - c(e)| \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Problem 3: Die Anzahl der Kanten, die sich verändern, soll minimal werden
Das inverse Engpass-Zuordnungsproblem unter Hamming Distanz

Bei diesem Problem soll die Anzahl der Kanten, die sich verändern, minimal werden, egal wie groß die Veränderung ist.

Wieder sind die gleiche Voraussetzungen gegeben wie bei den vorhergehenden Problemen und es soll folgendes Problem gelöst werden:

$$\begin{aligned} \min \quad & f_H(c^*) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Problem 4: Das größte Gewicht der veränderten Kanten soll minimal werden Das inverse Engpass-Zuordnungsproblem unter der Engpass-Hamming Distanz

Bei diesem Problem ist zusätzlich eine Gewichtsfunktion $w(e) \geq 0$ für alle $e \in E$ gegeben und es soll das größte Gewicht der veränderten Kosten minimal werden, dh. $\max_{e \in E} w(e)\ h(c^*(e), c(e))$ soll minimal werden.
Es soll folgendes Problem gelöst werden:

$$\begin{aligned} \min \quad & f_{bH}(c^*) := \max_{e \in E} w(e)\ h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Im nächsten Kapitel wird das Inverse Engpass-Zuordnungsproblem mit Hamming Distanz genau analysiert und nach einer bestmöglichen Lösung für dieses Problem gesucht. Weiters wird ein Algorithmus präsentiert, der diese Aufgabe lösen kann. In Kapitel 4 wird dann das inverse Engpass-Zuordnungsproblem mit einer Engpass-Hamming Distanz genau untersucht und ein Lösungsverfahren für dieses Problem präsentiert.

3 Inverses Engpass-Zuordnungsproblem mit Hamming-Distanz

In diesem Abschnitt wird das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz genauer betrachtet. Bei diesem Problem soll die Anzahl der Kanten, die sich verändern, minimal werden.
Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$ und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_H(c^*) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\, F \in \mathcal{F} \end{aligned}$$

Bemerkung Im Folgenden wird angenommen, dass die Lösung, die optimal werden soll, immer in der Hauptdiagonale der Kostenmatrix steht. Dies kann durch Spalten- oder Zeilenvertauschung erreicht werden.

Als erstes werden einige wichtige Eigenschaften des inversen Engpass-Zuordnungsproblems untersucht. Dazu wird durch $z(F, c) := \max_{e \in F} c(e)$ für $F \in \mathcal{F}$ die Kosten für die Engpass-Funktion oder auch Engpass-Kosten definiert.

Lemma 2 *[18] Sei c^* eine optimale Lösung für das inverse Engpass-Zuordnungsproblem mit $\max_{e \in F^*} c^*(e) = p^*$. Dann gilt für jede Kante $e \in E$:*

- *alle Kosten der Kanten $e \in F^*$, die größer p^* sind, werden verringert, sodass $c^*(e) = p^*$ gilt*
- *für alle zulässigen Lösungen $F \in \mathcal{F}$ mit $\max_{e \in F} c(e) < p^*$ werden die Kosten der Kanten, die sich verändern, auf p^* erhöht, sodass $\max_{e \in F} c^*(e) \geq p^*$ für alle zulässigen Lösungen gilt, dh. wenn $c(e) \neq c^*(e)$, dann gilt $c^*(e) = p^*$.*

Lemma 2 besagt, dass alle Kosten, die sich ändern, auf den Wert p^* gesetzt werden und für eine optimale Lösung c^* gilt dann:

$$c^*(e) = \begin{cases} p^*, & \text{wenn } c^*(e) \neq c(e) \\ c(e), & \text{sonst.} \end{cases}$$

Beweis: Es gilt für jede optimale Lösung $\bar{c}$, dass $\bar{c}(e) \leq p^*$ für jedes $e \in F^*$ und $\bar{c}(e) = c(e)$ für jede Kante $e \notin F^*$, für die $c(e) \geq p^*$ gilt. Somit ist es offensichtlich, dass c^*, definiert durch Lemma 2, eine Lösung für das inverse Engpass-Zuordnungsproblem ist.
Es wird die Behauptung aufgestellt, dass eine optimale Lösung $\bar{c}$ existiert, sodass für eine Kante e' gilt: $\bar{c}(e') \neq c(e')$, $\bar{c}(e') \neq p^* = c^*(e')$, und $\bar{c}(e) = c^*(e)$ für jede Kante $e \neq e'$, dabei muss $e' \in F^*$ und $\bar{c}(e') < p^*$ gelten.
Wenn $c(e') \geq p^*$ ist, dann gilt $c(e') \geq c^*(e') = p^* > \bar{c}(e')$ und deshalb
$f_H(c^*) = \sum_{e \in E} h(c^*(e), c(e)) =$
$\sum_{e \in E} h(\bar{c}(e), c(e)) - h(\bar{c}(e'), p^*) < \sum_{e \in E} h(\bar{c}(e), c(e)) = f_H(\bar{c})$. Dies ist ein Widerspruch zur Optimalität von $\bar{c}$.
Andererseits, wenn $c(e') < p^*$ ist, dann definiere

$$\tilde{c}(e) = \begin{cases} c(e), & \text{wenn } e = e' \\ c^*(e), & \text{sonst} \end{cases}$$

Dann ist $\tilde{c}$ auch eine zulässige Lösung. Außerdem gilt $\tilde{c}(e') = c(e') < p^* = c^*(e')$. Damit kann gezeigt werden, dass $f_H(\tilde{c}) = \sum_{e \in E} h(\tilde{c}(e), c(e)) = \sum_{e \in E} h(c^*(e), c(e)) - h(c(e'), p^*) < \sum_{e \in E} h(c^*(e), c(e)) = f_H(c^*)$ gilt. Dies ist ein Widerspruch zur Optimalität von c^* und der Beweis ist damit beendet.

□

Beispiel *Gegeben ist ein Graph, die zugehörige Kostenmatrix*

$$C = \begin{pmatrix} \boxed{1} & 2 & 3 \\ 2 & \boxed{3} & 2 \\ 1 & 2 & \boxed{7} \end{pmatrix}$$

und eine zulässige Lösung $F^* = \{c_{11}, c_{22}, c_{33}\}$*, die in der Matrix durch die Kästchen gekennzeichnet ist.*
Weiters sei bereits bekannt, dass $\max_{e \in F^*} c^*(e) = p^* = 3$ *gilt.*

Um zu erreichen, dass die vorgegeben Lösung F^* *optimal mit* $\max_{e \in F^*} c^*(e) = 3$ *wird, sollen möglichst wenig Elemente verändert werden. Deshalb wird das Element* c_{33} *auf den Wert* $p^* = 3$ *verringert. Weiters wird das Element* c_{23} *auf den Wert* p^* *erhöht und man erhält* $\max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e)$ *für alle* $F \in \mathcal{F}$.

Definition 5 *Ein Blocker in einem bipartiten Graphen G ist eine Kantenmenge, nach deren Entfernung aus dem Graphen kein perfektes Matching mehr übrig ist.*
Ein minimaler Blocker in einem bipartiten Graphen G ist die kleinste Kantenmenge, nach deren Entfernung aus dem Graphen kein perfektes Matching mehr übrig ist, d.h. der Blocker hat minimale Kardinalität.

Lemma 3 *Gegegeben ist ein bipartiter Graph $G = (U, V; E)$ mit $|U| = |V| = n$. Es gilt, dass ein minimaler Blocker B höchstens aus n Elemente besteht.*

Definition 6 *F_c mit $z(F_c, c) = \max_{e \in F_c} c(e)$ wird eine optimale Lösung für das Engpass-Zuordnungsproblem $\min_{F \in \mathcal{F}} z(F, c)$ genannt.*

Lemma 4 *[19] Für eine zulässige Lösung F_c gilt: F_c ist eine optimale Lösung für das Engpass Optimierungsproblem $\min_{F \in \mathcal{F}} \max_{e \in F} c(e)$ genau dann wenn $E^+ = \{e \in E \mid c(e) \geq z(F_c, c)\}$ ein Blocker ist.*

Beweis: Es wird angenommen, dass $z(F_c, c) \leq z(F, c)$ für $F \in \mathcal{F}$ ist.
Wenn E^+ kein Blocker ist, dann existiert eine zulässige Lösung $F' \in \mathcal{F}$, für die gilt $E^+ \cap F' = \emptyset$ und daher $c(e) < z(F_c, c)$ für jedes $e \in F'$. Es folgt $z(F', c) < z(F_c, c)$, welches ein Widerspruch zur Optimalität von F_c ist.
Nun wird davon ausgegangen, dass E^+ ein Blocker ist. Dann gilt für jedes $F \in \mathcal{F}$, $E^+ \cap F \neq \emptyset$. Daher existiert für jedes $F \in \mathcal{F}$ eine Kante $e' \in F$, sodass $c(e') \geq z(F_c, c)$. Deshalb gilt für jedes $F \in \mathcal{F}$ $z(F, c) \geq c(e') \geq z(F_c, c)$ und F_c ist eine optimale Lösung.

□

Beispiel *Gegeben ist ein Graph und die zugehörige Kostenmatrix*

$$C = \begin{pmatrix} 4 & 3 & 7 \\ 1 & 3 & 5 \\ 1 & 2 & 5 \end{pmatrix}$$

Dann ist $F_c = \{c_{21}, c_{12}, c_{33}\}$ eine optimale Lösung mit $z(F_c, c) = 5$. Werden alle Kanten, deren Kosten größer als 5 sind, aus dem Graphen entfernt, dann erhält man den Graphen von Abbildung 3.1.

In diesem Graphen kann kein perfektes Matching gefunden werden und $E^+ = \{e \in E \mid c(e) \geq z(F_c, c) = 5\}$ ist ein Blocker.

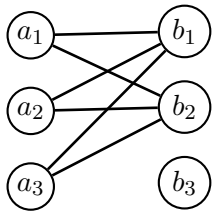


Abbildung 3.1: In diesem Graphen existiert kein perfektes Matching

Um nun ein inverses Engpass-Zuordnungsproblem zu lösen, werden nun zuerst inverse Engpass-Zuordnungsprobleme mit vorgegebenem Engpass-Funktionswert betrachtet. Bei diesem Problem wird ein inverses Engpass-Zuordnungsproblem gelöst, bei dem ein fester Wert p für die Engpass-Funktion vorgegeben ist, dh. es soll $\max_{e \in F^*} c^*(e) = p$ gelten. Es wird nach einer Lösung c^* gesucht, sodass Engpass-Kosten p sind und F^* soll optimal werden.
Für einen vorgegebenen Wert p wird das inverse Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert IEPZP(p) wie folgt definiert:

$$\begin{aligned} \min \quad & f_H(c^*, p) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & \max_{e \in F^*} c^*(e) = p \end{aligned}$$

Um nun solche Probleme zu lösen, werden zuerst einige wichtige Definitionen und Fakten präsentiert:

Definition 7 *Es sei ein fester Wert p für die Engpass-Funktion gegeben. Dann können folgende Mengen definiert werden:*
K_p ist die Menge aller Elemente aus E, für die $c(e) < p$ gilt, dh. $K_p = \{e \in E \mid c(e) < p\}$.
L_p ist die Menge aller Elemente aus der vorgegebenen Lösung F^*, für die $c(e) > p$ gilt, dh.

$L_p = \{e \in F^* \mid c(e) > p\}$.

Lemma 5 *Für $p_1 < p_2$ gilt, dass*

- $K_{p_1} \subseteq K_{p_2}$ *und*
- $L_{p_1} \supseteq L_{p_2}$.

Mit anderen Worten bedeutet das, dass für ein wachsendes p_k die Kardinalität der Menge K_{p_k} immer größer wird und die Menge K_{p_k} eine Folge von monoton wachsender Mengen ist. Die Kardinalität der Menge L_{p_k} wird immer kleiner und L_{p_k} ist eine Folge von monoton fallender Mengen.

Beweis: Es ist zu zeigen, dass $K_{p_1} \subseteq K_{p_2}$ gilt.
Angenommen es existiert ein Element $\bar{a} \in K_{p_1} \backslash K_{p_2}$. Dann gilt, dass $\bar{a} < p_1$ und $\bar{a} \geq p_2$ ist. Dies ist ein Widerspruch zu $p_1 < p_2$.
Der Beweis, das $L_{p_1} \supseteq L_{p_2}$ gilt, funktioniert analog.

□

Beispiel *Gegeben ist die Kostenmatrix*

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 7 \end{pmatrix}$$

Dann haben die Mengen K_p und L_p folgende Gestalt:

$p = 7$ $K_7 = \{c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23}, c_{31}, c_{32}\}$
$L_7 = \{\}$

$p = 3$ $K_3 = \{c_{11}, c_{12}, c_{21}, c_{23}, c_{31}, c_{32}\}$
$L_3 = \{c_{33}\}$

$p = 2$ $K_2 = \{c_{11}, c_{31}\}$
$L_2 = \{c_{22}, c_{33}\}$

$p = 1$ $K_1 = \{\}$
$L_1 = \{c_{22}, c_{33}\}$

und es gilt $K_7 \supseteq K_3 \supseteq K_2 \supseteq K_1$ und $L_7 \subseteq L_3 \subseteq L_2 \subseteq L_1$.

Lemma 6 *Der Zielfunktionswert* $\min f_H(c^*, p) := \sum_{e \in E} h(c^*(e), c(e))$ *ergibt sich aus der Summation zweier Faktoren* $z_L(p)$ *und* $z_K(p)$*. Dabei ist* $z_K(p)$ *die Anzahl der 1-Elemente im minimalen Blocker und* $z_L(p)$ *ist die Anzahl der Elemente aus* F^**, die größer als* p *sind, dh. die Kardinalität der Menge* L_p*.*

Definition 8 *Eine Funktion* $f : [a, b] \to \mathbb{R}$ *auf einem reellen Intervall* $I = [a, b] \to \mathbb{R}$ *heißt unimodal, falls sie auf* I *genau ein lokales Minimum* t^* *besitzt.*

Theorem 1 *Die Funktion* $f_H(c^*) := \sum_{e \in E} h(c^*(e), c(e))$ *ist eine unimodale Funktion.*

Beweis: Für $p_2 > p_1$ gilt, dass $z_K(p_2) \geq z_K(p_2)$ und $z_L(p_2) \leq z_L(p_1)$. Mit anderen Worten bedeutet dies, dass für ein fallendes p_k $z_K(p_k)$ eine monoton wachsende Folge ist und $z_L(p_k)$ eine monoton fallende Folge. Addiert man diese zwei Faktoren, so erhält man, dass die Funktion $f_H(c^*) := \sum_{e \in E} h(c^*(e), c(e))$ ist eine unimodale Funktion ist.

□

Um nun ein inverses Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert p lösen zu können, wird zuerst eine Kapazitätsmatrix W_p folgendermaßen konstruiert:

$$w_p(e) = \begin{cases} 1, & \text{wenn } \ e \in K_p \\ 0, & \text{sonst} \end{cases}$$

Die Aufgabe ist dann in dieser Kapazitätsmatrix einen minimalen Blocker zu finden. Dh. es wird nach einer Menge gesucht, deren Anzahl der 1-Elemente möglichst klein ist und nach Entfernung dieser Menge aus dem Graphen existiert kein perfektes Matching mehr.

Definition 9 *Sei* A *eine* $n \times n$ *Adjazenzmatrix. Dann ist ein Blocker* B *in der Matrix* A *eine* $(n - k) \times (k + 1), k \in \{0, 1, .., n - 1\}$ *Untermatrix. Der Wert eines Blockers kann durch* $w(B) = \sum_{(i,j) \in B} a_{ij}$ *definiert werden. Ein minimaler Blocker ist dann ein Blocker* B*, dessen Wert* $w(B)$ *minimal ist.*

Für die veränderten Kosten C^* gilt dann, dass alle 1-Elemente im minimalen Blocker und alle Werte der Menge L_p auf den Wert p gesetzt werden.

Beispiel *Gegeben ist eine Kostenmatrix, ein Engpass-Funktionswert $p = 3$ und eine Lösung F^**

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 7 \end{pmatrix}$$

Dann hat die Kapazitätsmatrix W_3 folgende Gestalt:

$$W_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Dabei ist W_3 die Adjazenzmatrix eines bipartiten Graphen, der eine Kante (i, j) besitzt, genau dann wenn $w_{ij} = 1$ ist. Ein minimaler Blocker in dieser Kapazitätsmatrix ist dann die Menge $B(p) = \{c_{13}, c_{23}, c_{33}\}$, die ein 1-Element entält. Weiters muss noch das Element c_{33} aus der Menge L_p verändert werden. Insgesamt werden also zwei Elemente verändert, dh. auf den Wert 3 gesetzt, um zu erreichen, dass $\max_{e \in F^*} c^*(e) = 3$ gilt.
Die veränderte Kostenmatrix hat dann folgende Gestalt:

$$C^* = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

An dieser Stelle werden einige Eigenschaften des Engpass-Funktionswerts p präsentiert:

Als erstes soll gezeigt werden, in welchem Bereich die Werte der Engpass-Funktion $z(F^*, c^*) = \max_{e \in F^*} c^*(e)$ liegen dürfen.

Korollar 1 *Sei F_c eine optimale Lösung für das Engpass-Zuordnungsproblem, F^* eine vorgegebene Lösung, die optimal werden soll und c^* der gesuchte Kostenvektor, dann gilt:*

$$z(F_c, c) \leq z(F^*, c^*) = p \leq z(F^*, c)$$

oder

$$\max_{e \in F_c} c(e) \leq \max_{e \in F^*} c^*(e) = p \leq \max_{e \in F^*} c(e)$$

Beweis: Die erste Ungleichung gilt, da F_c eine optimale Lösung mit den ursprünglichen Kosten ist. Würde $\max_{e \in F_c} c(e) > \max_{e \in F^*} c^*(e)$ gelten, dann wäre F_c keine Optimallösung. Die zweite Ungleichung gilt, da die Kosten der Kanten in F^*, die größer als p sind, auf den Wert p verringert werden.

□

Als nächstes soll gezeigt werden, dass nur die Elemente $c(e)$, $e \in F^*$, und der Wert $z(F_c, c)$ als Engpass-Funktionswert p in Frage kommen.

Lemma 7 *Hat das inverse Engpass-Zuordnungsproblem eine optimale Lösung c^*, dann nimmt $p^* = \max_{e \in F^*} c^*(e)$ einen Wert aus der Menge*

$$(\{c(e)|e \in F^*\} \cup z(F_c, c)) \cap [z(F_c, c), z(F^*, c)]$$

an.

Beweis: Es wird angenommen, dass c^* eine optimale Lösung des inversen Engpass-Zuordnungsproblems mit $q_{i-1} < p^* = \max_{e \in F^*} c^*(e) < q_i$ und $q_{i-1}, q_i \in \{\{c(e)|e \in F^*\} \cup z(F_c, c)\}$ ist.
Dann gilt für den Zielfunktionswert $\min f_H(c^*, p) := \sum_{e \in E} h(c^*(e), c(e))$:
$f_H(c^*, p^*) = z_K(p^*) + z_L(p^*)$
$f_H(c^*, q_{i-1}) = z_K(q_{i-1}) + z_L(q_{i-1})$.
Es folgt, dass $z_L(p^*) = z_L(q_{i-1})$ da die Mengen L_{p^*} und $L_{q_{i-1}}$ identisch sind.
Weiters gilt, dass $z_K(p^*) \geq z_K(q_{i-1})$ ist, da $A^*_p \supseteq A_{q_{i-1}}$ gilt.
Daraus folgt, dass $f_H(c^*, p^*) \geq f_H(c^*, q_{i-1})$ ist und deshalb ist es ausreichend nur die Elemente aus F^* als Engpass-Funktionswert zu betrachten.

□

Beispiel *Gegeben ist eine Kostenmatrix*

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 7 \end{pmatrix}$$

Für $p = 4$ hat die Kapazitätsmatrix folgende Gestalt:

$$W_4 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Der zugehörige Zielfunktionswert ist dann $f_H(c^*, 4) = z_K(4) + z_L(4) = 2 + 1 = 3$.
Für $p = 3$ hat die Kapazitätsmatrix folgende Gestalt:

$$W_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Zielfunktionswert: $f_H(c^*, 3) = z_K(3) + z_L(3) = 1 + 1 = 2$.
Es folgt daraus $f_H(c^*, 4) \geq f_H(c^*, 3)$.

Proposition 1 *Für ein inverses Engpass-Zuordnungsproblem kann eine obere Schranke für den Zielfunktionswert angegeben werden. Es gilt:*

$$\min f_H(c^*) = \sum_{e \in E} h(c^*(e), c(e)) \leq n - 1$$

Beweis: Aus dem vorhergehenden Lemma ist bekannt, dass nur die Kosten $c(e)$ aus F^* und $z(F_c, c)$ als Engpass-Funktionswert in Frage kommen.
Fall 1: $p = c(e)$ mit $e \in F^*$. Dann ist es im ungünstigsten Fall ausreichend die Kostenmatrix so zu verändern, dass in der Zeile (oder Spalte) in der $c(e)$ steht, die Kosten auf den Wert p gesetzt werden. Damit gilt $\max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e)$ für alle $F \in \mathcal{F}$ und der Zielfunktionswert ist kleiner gleich $n - 1$.
Fall 2: $p = z(F_c, c)$. Dann können alle Elemente $c(e) > p$, $e \in F^*$ auf den Wert p geändert werden. Dies sind im ungünstigsten Fall $n - 1$ Elemente, die geändert werden müssen. Angenommen es wären mehr als $n - 1$ Elemente zu ändern, dh. für alle $e \in F^*$ ist $c(e) > z(F_c, c)$. Dann wäre es besser $p = c(e)$ mit $e \in F^*$ zu wählen, da in diesem Fall sicher weniger als n Elemente zu ändern wären, vgl. Fall 1.

□

Beispiel *Gegeben ist eine Kostenmatrix*

$$C = \begin{pmatrix} 7 & 2 & 3 & 4 \\ 2 & 7 & 3 & 2 \\ 3 & 4 & 7 & 3 \\ 3 & 6 & 4 & 7 \end{pmatrix}$$

Es gilt dann, dass $z(F^*, c) = 7$ *und* $z(F_c, c) = 3$.
Dann ist es sinnvoller $p = 7$ *zu betrachten. In diesem Fall können beispielsweise die Kosten der ersten Zeile auf den Wert 7 gesetzt werden und man erhält, dass* $\max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e)$ *für alle* $F \in \mathcal{F}$ *gilt. Es sind also* $n - 1 = 3$ *Elemente zu ändern, um* $\max_{e \in F^*} c^*(e) = 7$ *zu erreichen.*
Der Fall $p = z(F_c, c) = 3$ *kann vernachlässigt werden, da in diesem Fall sicherlich die vier Elemente* $c(e)$ *mit* $e \in F^*$ *zu verändern wären um* $\max_{e \in F^*} c^*(e) = 3$ *zu erreichen.*

Mit Hilfe dieser Überlegungen wird nun ein inverses Engpass- Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert gelöst und es wird ein Algorithmus für dieses Problem präsentiert.
Als erstes werden die Mengen $K_p := \{e \in E \mid c(e) < p\}$ und $L_p := \{e \in F^* \mid c(e) > p\}$ definiert. Nun wird der Graph $G = (V, K_p)$ betrachtet und untersucht, ob er ein perfektes Matching besitzt.

Fall 1: Der Graph $G = (V, K_p)$ besitzt kein perfektes Matching. D.h. für jede Lösung $F \in \mathcal{F}$ gilt, dass die Engpass-Funktion $z(F, c) \geq p$ ist und deshalb müssen nur jene Werte der vorgegebenen Lösung F^* geändert werden, die größer als p sind. Dann ist

$$c^*(e) = \begin{cases} p, & \forall e \in L_p \\ c(e), & \text{sonst} \end{cases}$$

eine optimale Lösung für das inverse Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert und der Zielfunktionswert ist:

$$f_H(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) = z_L(p)$$

Fall 2: Der Graph $G = (V, K_p)$ besitzt ein perfektes Matching.

In diesem Fall muß ein minimaler Blocker gefunden werden, sodass nach der Entfernung dieser Kantenmenge aus dem Graphen $G = (V, E_p)$ kein perfektes Matching mehr möglich ist.

Subproblem: Suche einen minimalen Blocker $B(p)$. Erstelle dazu eine Kapazitätsmatrix:

$$w_p(e) = \begin{cases} 1, & \text{wenn } e \in K_p \\ 0, & \text{sonst} \end{cases}$$

Suche dann innerhalb dieser Kapazitätsmatrix einen Blocker, d.h. eine Menge $B(p)$, deren Anzahl der 1-Elemente möglichst klein ist. Dieses Problem wird in Kapitel 3.1 noch genauer untersucht.

Im Graphen $G = (V, K_p/B(p))$ kann nun kein perfektes Matching mehr gefunden werden und die Kostenfunktion kann wie folgt definiert werden:

$$c^*(e) = \begin{cases} p, & \forall\ e \in L_p \text{ und } \forall\ e \in \{e \in B(p) \mid w(e) = 1\} \\ c(e), & \text{sonst} \end{cases}$$

Der optimale Zielfunktionswert ist:

$$f_H(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) = z_L(p) + z_K(p)$$

Theorem 2 *Das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz kann zu $\mathcal{O}(n)$ Problemen einen minimalen Blocker zu finden reduziert werden.*

Beweis: Dieses Theorem gilt aufgrund der Tatsache, dass der Engpass-Funktionswert p einen Wert aus der Menge $(\{c(e)|e \in F^*\} \cup z(F_c, c)) \cap [z(F_c, c), z(F^*, c)]$ annimmt, vgl. dazu Lemma 7.

3.1 Lösung des SubProblems: Finde minimalen Blocker

In diesem Abschnitt wird das Problem einen minimalen Blocker zu finden, näher untersucht. Dazu werden einige wichtige Grundlagen benötigt:

Sei $i \in U$, dann kann mit $N(i)$ die Menge aller Nachbarn von i definiert werden, d.h. die Menge aller Knoten $j \in V$, die mit i durch eine Kante aus E verbunden sind.

Theorem 3 (Hall) *Sei $G = (U, V; E)$ ein bipartiter Graph. Es ist möglich jeden Knoten aus U mit einem Knoten aus V zu verbinden genau dann wenn für alle Untermengen U' von U gilt:*

$$|U'| \leq |N(U')| \qquad \text{(Bedingung von Hall)}$$

Theorem 4 (Heiratssatz) *Sei $G = (U, V; E)$ ein bipartiter Graph mit $|U| = |V|$. Es existiert ein perfektes Matching in G genau dann, wenn G die Bedingung von Hall erfüllt.*

Es kann die Bedingung von Hall auch mit Hilfe einer Adjazenzmatrix ausgedrückt werden [6]. Wenn die Adjazenzmatrix A genau n Eins-Elemente enthält, eine Eins in jeder Reihe und in jeder Spalte, dann beinhaltet die Matrix A eine Permutationsmatrix. Die Bedingung von Hall besagt nun, dass für $k = 0, 1, .., n-1$ die Matrix A keine $(k+1) \times (n-k)$ Submatrix aus Null-Elementen enthält, ansonsten würden die $k+1$ Knoten, die den Reihen der Submatrix entsprechen, weniger als $k+1$ Nachbarn haben. Deshalb ist das folgende Theorem äquivalent zum Heiratssatz.

Theorem 5 (Frobenius) *Sei A eine beliebige $n \times n$ Matrix, deren Einträge Null oder Eins sind. Die Matrix A enthält eine Permutationsmatrix genau dann wenn für $k = 0, 1, .., n-1$ die Matrix A keine $(k+1) \times (n-k)$ Submatrix, deren Einträge alle Null sind, enthält.*

Nach Frobenius genügt es, einen minimalen Blocker B in der Klasse aller $(k+1) \times (n-k)$, $k \in \{0, 1, .., n-1\}$, Untermatrizen zu suchen.

Für das inverse Engpass-Zuordnungsproblem mit vorgegebenem Engpass-Funktionswert bedeutet dies nun Folgendes: Gegeben ist ein bipartiter Graph und eine Kostenfunktion $c(e)$ für alle $e \in E$. Sei $K_p := \{e \in E \mid c(e) < p\}$. Die Aufgabe besteht darin, dass das Sub-Problem, einen minimalen Blocker zu finden, gelöst werden soll. Deshalb wird davon ausgegangen, dass der Graph $G = (V, K_p)$ ein perfektes Matching besitzt. In der zugehörigen Adjazenzmatrix bzw. Kapazitätsmatrix wird nach einer $(k+1) \times (n-k)$ $k = 0, 1, .., n-1$ Submatrix B gesucht, deren Anzahl der 1 Elemente minimal ist. Wird diese Matrix B gefunden, dann werden die Kosten für jene Kanten aus B, die 1 sind, auf p gesetzt. Dadurch erhält man eine neue Kostenfunktion $\tilde{c}$ für alle $e \in E$ und eine neue Menge $\tilde{K_p} := \{e \in E \mid \tilde{c}(e) < p\}$, die kein perfektes Matching mehr enthält. Die Aufgabe, einen minimalen Blocker zu finden, ist $\mathcal{NP}$-vollständig. Dies wird in Kapitel 3.2 gezeigt.

Beispiel *Gegeben ist eine Kostenmatrix*

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 7 \end{pmatrix}$$

Für $p = 3$ hat die Kapazitätsmatrix folgende Gestalt:

$$W(3) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Gesucht ist ein minimaler Blocker B.
Wird der Eintrag c_{23} auf Null gesetzt, dann erhält man so eine 1×3 Matrix, die nur aus Nullen besteht. Der zugehörige Graph $G' = (V, K_p/B)$ wird in Abbildung 3.2 dargestellt.

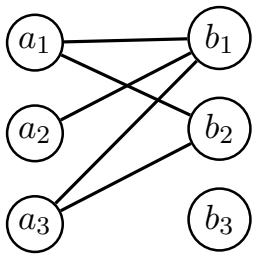


Abbildung 3.2: Graph $G' = (V, K_p \backslash B)$

In diesem Graphen ist es nicht möglich, ein perfektes Matching zu finden und $B = \{(a_1, b_3), (a_2, b_3), (a_3, b_3)\}$ ist somit ein minimaler Blocker.

Weitere Überlegungen werden angestellt, wie viele $(k+1) \times (n-k)$, $k = 0, 1, .., n-1$ Submatrizen in einer $n \times n$ Matrix untersucht werden müssen: Für eine 3×3 Matrix $A = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}$ müssen also 3 (1×3), 9 (2×2) und 3 (3×1) Matrizen untersucht werden. Für $n = 3$ müssen also 15 verschiedene Matrizen betrachtet werden. Für eine 7×7 Matrix müssen bereits 3003 Submatrizen ausgewertet werden.

Für beliebiges n müssen

$$\sum_{k=0}^{n-1} \binom{n}{k+1} * \binom{n}{n-k}$$

Matrizen untersucht werden.

Algorithmus um einen minimalen Blocker zu finden
Algorithmus 1:
Gegeben ist eine $n \times n$ Adjazenzmatrix, die ein perfektes Matching besitzt.

1. For $i = 1$ to $\sum_{k=0}^{n-1} \binom{n}{k+1} * \binom{n}{n-k}$
   Berechne für die Submatrix S_i die Summe U_i über alle Einträge in dieser Matrix.
   Wenn $U_i = 1$, dann ist S_i ein minimaler Blocker und der Algorithmus stoppt.
   Ansonsten i++.

2. Finde das Minimum unter allen U_i. Die zugehörige Submatrix S_i ist dann der gesuchte minimale Blocker.
   Wenn alle Summen $U_i = \infty$ sind, dann existiert keine zulässige Lösung für dieses Problem.

Matlab-Algorithmus um minimalen Blocker in einer $n \times n$ Matrix zu finden

```
clear all;
X = [11 12 13; 21 22 23; 31 32 33];


n = size(X,1)
matrixtestcell = {}
totalsum =100000

for k = 0:(n-1)
    dim1 = k+1
    dim2 = n-k

    for j = 1:n
        matrix = nchoosek(X(j,:),dim2);
        matrixtestcell{k+1,j} = matrix;
     end

     if size(matrix,1) > dim1
         getidx = 1:1:n
         getidx = [nchoosek(getidx,dim1)]'

        for j = 1:size(matrix,1)

            for l = 1:size(getidx,2)

                matrixvalid1 = [];
                idx = getidx(:,l);
                for i = 1:size(getidx,1)

                    matrixtestcell{k+1,idx(i)};
                    matrixvalid1 = [matrixvalid1; matrixtestcell{k+1,idx(i)}(j,:)];
                end
                matrixvalid{k+1,j,l} = matrixvalid1 ;

                if totalsum > sum(matrixvalid{k+1,j,l}(:))
                    totalsum = sum(matrixvalid{k+1,j,l}(:))
                end
                if totalsum == 1
                    return;
                end

            end
```

```
43          end
44      else
45          for f = 1:n
46              matrixvalid{k+1,f,1} = matrixtestcell{k+1,f} ;
47              if totalsum > sum(matrixvalid{k+1,f,1}(:))
48                  totalsum = sum(matrixvalid{k+1,f,1}(:))
49                  if totalsum == 1
50                      return;
51                  end
52              end
53          end
54      end
55  end
56  totalsum
```

Laufzeitanalyse:
Da für eine beliebige $n \times n$ Matrix im ungünstigsten Fall $\sum_{k=0}^{n-1} \binom{n}{k+1} * \binom{n}{n-k}$ Matrizen untersucht werden müssen, resultiert eine sehr große Laufzeit für den Algorithmus.
Für den Algorithmus 1, der einen minimalen Blocker findet, wird eine Laufzeit von $\mathcal{O}(n\ 2^{2n})$ erreicht auf Grund folgender Überlegungen:
Die erste for-Schleife (Zeile 9) und die vierte for-Schleife (Zeile 28) werden n mal durchlaufen. Die zweite und dritte for-Schleife (Zeile 22 und 24) wird im ungünstigsten Fall $\binom{n}{k}$ mal durchlaufen, da der Matlab-Befehl $nchoosek(v, k)$ für einen Vektor v mit n Elementen und ein Skalar k alle möglichen Kombinationen von k Elementen aus v ohne Wiederholungen liefert. Aus der Konstruktionsvorschrift des Pascalschen Dreiecks ergibt sich, dass der Binomialkoeffizient für ein festes n den maximalen Wert annimmt, wenn $k = \lfloor \frac{n}{2} \rfloor$ oder $k = \lceil \frac{n}{2} \rceil$ gewählt wird. Für gerade n fallen beide Fälle zusammen und es kann vereinfacht $k = \frac{n}{2}$ geschrieben werden. Daraus ergibt sich, dass $\binom{n}{k}$ keinen Wert annehmen kann, der größer als $\binom{n}{\lfloor \frac{n}{2} \rfloor} = \frac{n}{\lfloor \frac{n}{2} \rfloor} \cdot \frac{n-1}{\lfloor \frac{n}{2} \rfloor - 1} \cdots \frac{n-(\lfloor \frac{n}{2} \rfloor - 1)}{1}$ ist.

Mit Hilfe der Stirling-Formel $n! \approx \sqrt{2\pi n}\ (\frac{n}{e})^n$ kann $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ abgeschätzt werden durch $\frac{2^n}{\sqrt{n}}$.

Algorithmus 1 besitzt also eine Laufzeit von $\mathcal{O}(n\ 2^{2n})$.

3.2 Das Problem einen minimalen Blocker zu finden ist $\mathcal{NP}$-vollständig

In diesem Abschnitt wird gezeigt, dass das Problem, einen minimalen Blocker in einem bipartiten Graphen zu finden, $\mathcal{NP}$-vollständig ist. In der Arbeit 'Blockers und Transversals' von R. Zenklusen, B. Ries, C. Picouleau, D. de Werra, M.C. Costa und C. Bentz [27] wird diese Problemstellung genau analysiert.

Definition 10 *Für einen Graphen $G = (V, E)$ wird durch $\nu(G)$ die maximale Kardinalität eines Matchings in G definiert.*

Definition 11 *Ein d-Blocker B in einem Graphen $G = (V, E)$ ist eine Kantenmenge, nach deren Entfernung aus dem Graphen, die Kardinalität eines Matchings mit maximaler Kardinalität um mindestens d verringert wird. Eine Teilmenge $B \subseteq E$ wird dann d-Blocker genannt, wenn $\nu(G') \leq \nu(G) - d$ gilt, dabei ist $G' = (V, E \backslash B)$.*

Beispiel *Gegeben ist folgender Graph $G = (V, E)$:*

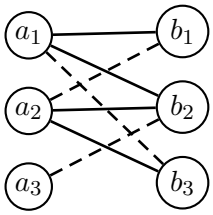


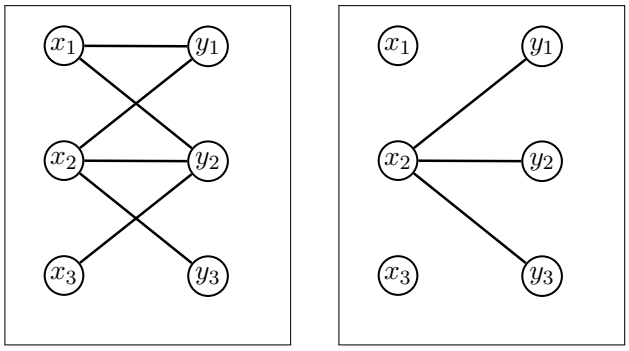
Abbildung 3.3: Graph G mit $\nu(G) = 3$

In diesem Graphen wird das perfekte Matching durch die strichlierten Linien gekennzeichnet und es gilt $\nu(G) = 3$. Entfernt man die Kante (a_3, b_2) aus dem Graphen, dann wird die Kardinalität des Matchings um eins verringert. Es gilt dann $\nu(G') = 2$ und die Menge $B = \{(a_3, b_2)\}$ ist ein 1-Blocker.

Problem: $BLOCK(G, d, k)$
Gegeben ist ein ungerichteter Graph $G = (V, E)$ und zwei natürliche Zahlen $0 \leq d \leq \nu(G)$, $0 \leq k \leq |E|$.
Frage: Existiert eine Menge $B \subseteq E$ mit $|B| \leq k$ sodass $\nu(G') \leq \nu(G) - d$ gilt mit $G' = (V, E \backslash B)$?

Beispiel *Gegeben ist ein ungerichteter Graph $G = (V, E)$, $d = 2$ und $k = 4$.*
Gefragt ist nun, ob eine Menge $B \subseteq E$ existiert mit $|B| \leq 4$ sodass $\nu(G') \leq \nu(G) - 2$ gilt.

Im Graphen G ist die maximale Kardinalität eines Matchings $\nu(G) = 3$. Werden die Kanten $B = \{(x_1, y_1), (x_1, y_2), (x_3, y_2)\}$ aus dem Graphen entfernt, dann erhält man den Graphen $G' = (V, E \backslash B)$ mit $\nu(G') = 1$. Somit ist B ein Blocker mit $|B| = 3 \leq 4$ und $\nu(G') = 1 \leq \nu(G) - 2 = 3 - 2 = 1$



(a) Graph $G = (V, E)$ mit $\nu(G) = 3$ (b) Graph $G' = (V, E \backslash B)$ mit $\nu(G') = 1$

Abbildung 3.4: Beispiel für $BLOCK(G, 2, 4)$

Es wird zuerst der Fall einen d-Blocker in einem bipartiten Graphen zu finden betrachtet und danach wird der Spezialfall einen 1-Blocker zu finden analysiert.

Theorem 6 *In einem bipartiten Graph G ist das Problem $BLOCK(G, d, k)$ $\mathcal{NP}$-vollständig.*

Für den Beweis dieses Theorems werden zuerst einige essentielle Fakten präsentiert:

Definition 12 *Mit C_4 wird ein Graph bezeichnet, der nur aus einem Kreis der Länge 4 besteht.*

Proposition 2 *Sei $k \geq 4$ eine natürliche Zahl und sei $G = (X, Y, E)$ ein einfacher bipartiter Graph, für den gilt:*

1. $|X| > k$
2. $|Y| = \binom{k}{2}$
3. $d(y) = 2, \forall y \in Y$ *und* $d(x) \geq 1, \forall x \in X$
4. *G beinhaltet keinen Kreis C_4.*

Dann gilt $\nu(G) \geq k + 1$.
Der Beweis dieser Proposition ist in [27] zu finden.

Beispiel *Gegeben ist ein bipartiter Graph $G = (X, Y, E)$ und eine Zahl $k = 4$:*

Bei diesem Graphen sind alle Bedingungen der vorhergehenden Proposition erfüllt und es gilt $\nu(G) = 5$.

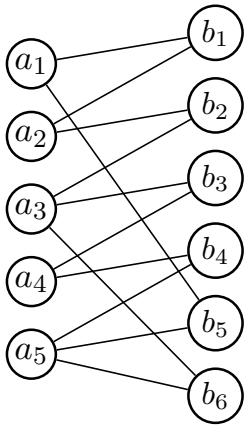


Abbildung 3.5: Graph mit $\nu(G) = 5$

Definition 13 *Sei $G = (V, E)$ ein einfacher Graph und U eine Teilmenge von V. Dann ist U eine Clique von G, wenn für je zwei beliebige verschiedene Knoten v und w aus U gilt, dass sie durch eine Kante miteinander verbunden sind.*

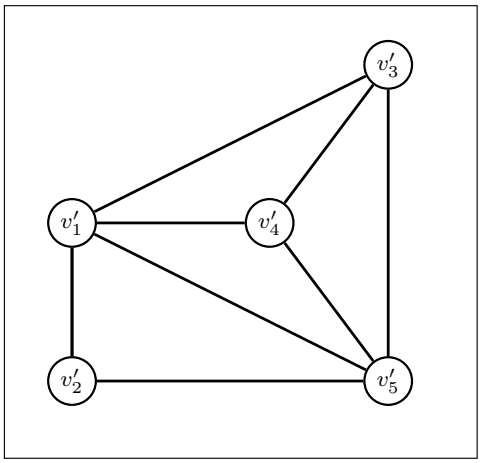
Definition 14 (Cliquenproblem) *Gegeben ist ein einfacher Graph $G = (V, E)$ und eine Zahl k. Das Entscheidungsproblem zu einem Graphen G und einer natürlichen Zahl k zu entscheiden, ob G eine Clique der Größe mindestens k enthält, wird Cliquenproblem genannt,dh. ob G mindestens k Knoten enthält, die untereinander paarweise verbunden sind.*

Theorem 7 *Das Cliquenproblem ist $\mathcal{NP}$-vollständig.*

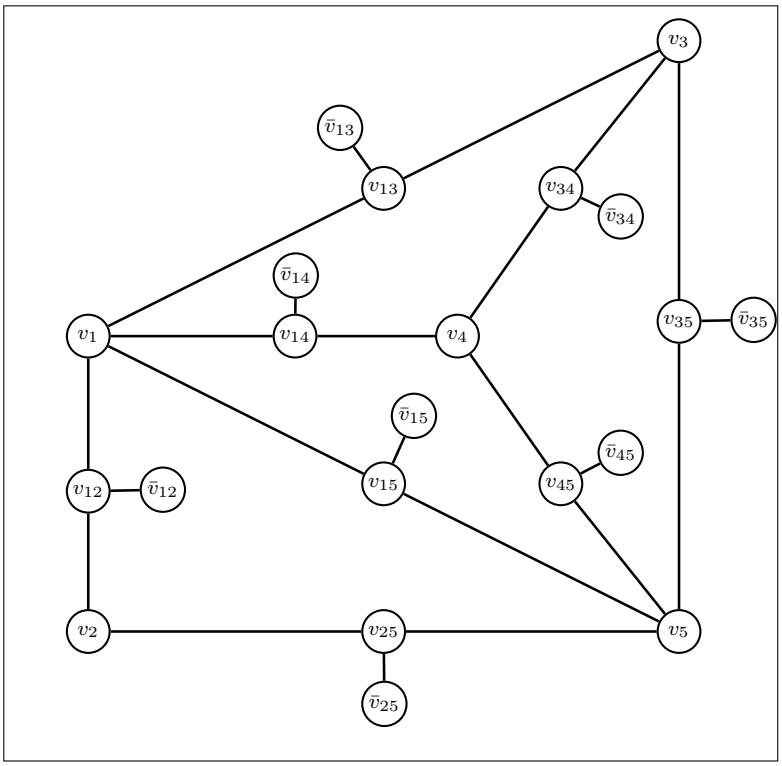
Der Beweis zu diesem Theorem kann in [17] gefunden werden.

Beweis zu Theorem 6:
Indem das Problem $BLOCK(G, d, k)$ auf ein Cliquenproblem zurückgeführt wird, soll gezeigt werden, dass das Problem $BLOCK(G, d, k)$ $\mathcal{NP}$-vollständig ist.

Ausgehend von einem ungerichteten einfachen Graphen $G' = (V', E')$ wird ein neuer Graph $G = (V, E)$ konstruiert und es wird dann bewiesen, wenn G' eine Clique beinhaltet, dann enthält der Graph G einen Blocker.
Sei $G' = (V', E')$ ein ungerichteter einfacher Graph und sei $r \leq |V'|$ eine positive ganze Zahl. Dann wird ein bipartiter Graphen $G = (V, E)$ folgendermaßen konstruiert: für jeden Knoten $v'_i \in V'$ wird im neuen Graphen ein Knoten $v_i \in V$ eingefügt und für jede Kante $e'_{ij} = [v'_i, v'_j] \in E'$ ein Knoten $v_{ij} \in V$. Für jeden Knoten $v_{ij} \in V$ fügt man einen Knoten $v_{\bar{ij}}$ und auch eine Kante $[v_{ij}, v_{\bar{ij}}]$ ein. Schließlich wird für jede Kante $[v'_i, v'_j] \in E'$ eine Kante $[v_i, v_{ij}]$ und eine Kante $[v_{ij}, v_j]$ eingefügt, vgl. dazu Abbildung 3.6.



(a) ungerichteter einfacher Graph $G' = (V', E')$ mit einer Clique C der Größe 4



(b) Konstruierter Graph $G' = (V', E')$

Abbildung 3.6: Beispiel für einen konstruierten Graphen

Zu beachten ist, dass die Kardinalität eines Maximum Matchings M in G ist $|M| = m$, dabei ist m die Anzahl der Kanten in G'. Man kann so ein Matching finden indem man alle Kanten $[v_{ij}, \bar{v_{ij}}]$ nimmt.
Es soll folgende Aussage bewiesen werden: G' beinhaltet eine Clique der Größe r genau dann wenn ein $\frac{r(r-3)}{2}$-Blocker in G mit $|B| = \frac{r(r-1)}{2}$ existiert und keine Kante von U benutzt wird,

mit $U = \{[v_i, v_{ij}] \mid [v'_i, v'_j] \in E'\}$.

Als erstes wird angenommen, dass G' eine Clique C der Größe r beinhaltet und seien $E'_C \subseteq E'$ die Kanten dieser Clique. Wenn nun $B = \{[v_{ij}, v'_{ij}] \mid e'_{ij} \in E'_C\}$ angenommen wird, erhält man einen $\frac{r(r-3)}{2}$-Blocker. Dies gilt, da ein Maximum Matching im Graphen $G^* = (V, E \backslash B)$ gefunden werden kann, indem man alle übrigen Kanten $[v_{ij}, \bar{v_{ij}}]$ (es gibt genau $m - \frac{r(r-1)}{2}$ solcher Kanten) und alle Kanten des Maximum Matchings im Subgraphen, der von den Knoten v_{ij} mit $e'_{ij} \in E'_C$ und den Knoten v_i mit $v'_i \in C$ aufgespannt wird, nimmt. Die Kardinalität eines Maximum Matchings in diesem Subgraphen ist höchstens r. Deshalb gilt dann $\nu(G^*) \leq m - \frac{r(r-1)}{2} + r = m - \frac{r(r-3)}{2}$.

Auf der anderen Seite wird nun angenommen, dass es einen $\frac{r(r-3)}{2}$-Blocker B in G mit $|B| = \frac{r(r-1)}{2}$ gibt und keine Kante von U benutzt wird, aber keine Clique der Größe r in G' existiert. Dies bedeutet, dass der Subgraph, der von den Knoten v_{ij} und den Knoten v_i, v_j mit $[v_{ij}, \bar{v_{ij}}] \in B$ erzeugt wird, ein einfacher bipartiter Graph $\tilde{G} = (X, Y, \tilde{E})$ mit folgenden Eigenschaften ist:

1. $|X| > r$ (da keine Clique der Größe r in G' existiert)
2. $|Y| = \frac{r(r-1)}{2} = \binom{r}{2}$
3. $d(v_i) = 2, \forall v_i \in Y$ und $d(v_{ij}) \geq 1, \forall v_{ij} \in X$
4. $\tilde{G}$ beinhaltet keinen Kreis C_4
5. $\nu(\tilde{G}) \leq r$.

Dies ist ein Widerspruch zur Proposition 2. Deshalb existiert eine Clique der Größe r in G'.

□

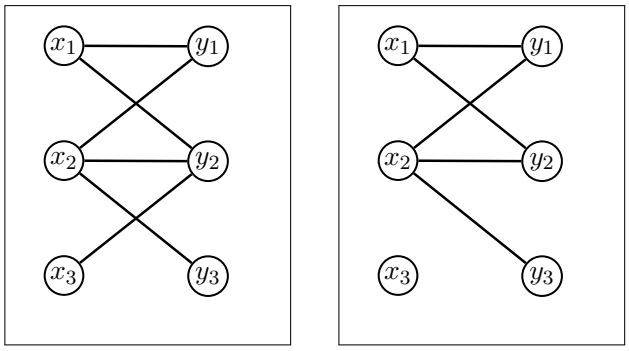
Im Folgenden soll untersucht werden, ob das Problem einen minimalen 1-Blocker zu finden, auch $\mathcal{NP}$-vollständig ist.

Theorem 8 *In einem bipartiten Graph G ist das Problem $BLOCK(G, d = 1, k)$ $\mathcal{NP}$-vollständig.*

Beispiel *Gegeben ist ein ungerichteter Graph $G = (V, E)$, $d = 1$ und $k = 4$.*
Gefragt ist nun, ob eine Menge $B \subseteq E$ existiert mit $|B| \leq 4$ sodass $\nu(G') \leq \nu(G) - 1$ gilt.

Wird die Kante $B = \{(x_3, y_2)\}$ aus dem Graphen entfernt, dann erhält man den Graphen $G' = (V, E \backslash B)$ mit $\nu(G') = 2$.

Beweis zu Theorem 8:
Der Beweis dieses Theorems wird geführt, indem das Problem $BLOCK(G, d, k)$ auf das Problem $BLOCK(G, d = 1, k)$ zurückgeführt wird.

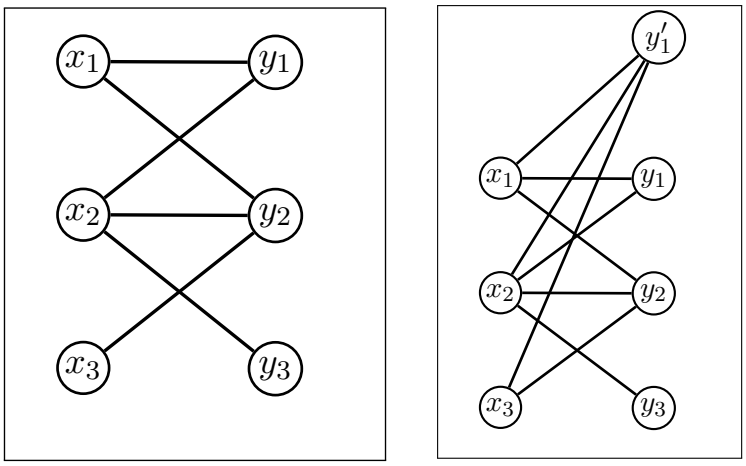


(a) Graph $G = (V, E)$ mit $\nu(G) = 3$

(b) Graph $G' = (V, E \backslash B)$ mit $\nu(G') = 2$

Abbildung 3.7: Beispiel für $BLOCK(G, 1, 4)$

Sei $G' = (X', Y', E')$ ein bipartiter Graph, $d \in \{1, ..., \nu(G')\}$ und $k \in \{0, 1, ..., |E'|\}$. Der Graph $G = (X, Y, E)$ wird dann folgendermaßen definiert: $X = X'$ und $Y = Y' \cup Y'_a$ ist die Menge Y' und $|X'| - \nu(G') + d - 1$ zusätzliche Kanten $Y'_a = \{y'_1, ..., y'_{|X'|-\nu(G')+d-1}\}$. Die Menge $E = E' \cup U$ besteht aus den Kanten E' und für jedes Paar von Knoten $x \in X$ und $y' \in Y'_a$ wird eine Kante $[x, y]$ hinzugefügt. Diese Kanten werden mit U bezeichnet.



(a) bipartiter Graph $G' = (V', E')$ mit $d = 2$

(b) Konstruierter Graph $G = (V, E)$ mit einem zusätzlichen Knoten y'_1, da $|X'| - \nu(G') + d - 1 = 3 - 3 + 2 - 1$

Abbildung 3.8: Beispiel für einen konstruierten Graphen

Es gilt dann $\nu(G) = |X'|$. Diese Aussage gilt aufgrund folgender Überlegung: Sei M ein Maximum Matching in G'. Im Graphen G kann das Matching M leicht zu einem Matching mit der Kardinalität $|X'|$ durch die Kanten aus der Menge U erweitert werden, da $|Y'_a| \geq |X'| - \nu(G')$. Im Folgenden soll bewiesen werden, dass es in G' einen $d - Blocker$ mit einer Kardinalität von höchstens k gibt, genau dann wenn in G ein $1 - Blocker$ mit einer Kardinalität von höchstens

k existiert, der keine Kanten von U beinhaltet.
Zuerst wird angenommen, dass ein $d - Blocker$ B in G' mit einer Kardinalität von höchstens k existiert, der keine Kante von U benutzt und es soll gezeigt werden, dass B dann auch ein $1 - Blocker$ in G ist. Durch einen Widerspruch wird angenommen, dass es ein Matching M in $G\backslash B$ mit Kardinalität $|X'|$ gibt. Dies impliziert, dass die Menge $M\backslash U$ ein Matching in $G'\backslash B$ mit einer Kardinalität von mindestens $|X'| - |Y'_a| = \nu(G') - d + 1$, da jedes Matching in G mindestens $|Y'_a|$ Kanten von U enthält, da U aus $|Y'_a|$ Kantenbündel besteht. Dies widerspricht dann der Tatsache, dass B ein $d - Blocker$ in G' ist.
Auf der anderen Seite wird angenommen, dass es in G' keinen $d-Blocker$ mit einer Kardinalität von höchstens k gibt. Sei $B \subseteq E'$ mit $|B| \leq k$. Da es in G' keinen $d - Blocker$ mit einer Kardinalität von höchstens k gibt, existiert in G' ein Matching $M \subseteq E'\backslash B$ mit einer Kardinalität von $\nu(G') - d + 1$. Das Matching M kann in G mit Hilfe der Kanten aus der Menge U zu einem Matching M' mit einer Kardinalität von X' erweitert werden, da $|Y'_a| = |X'| - \nu(G') + d - 1$. Dies bedeutet, dass B kein $1 - Blocker$ in G ist. Da B beliebig gewählt wurde, impliziert dies, dass es keinen $1 - Blocker$ in G mit Kardinalität k gibt, der Kanten von U beinhaltet.

□

3.3 Algorithmus um das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz zu lösen

Gegeben ist:

- bipartiter Graph $G = (V, E)$,
- eine zulässige Lösung F^*, die optimal werden soll,
- eine Kostenfunktion $c(e)$ für alle $e \in E$.

Gesucht ist eine Kostenfunktion c^*, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_H(c^*) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \end{aligned}$$

Schritt 1:
Suche nach den möglichen Werten für die Engpass-Funktion innerhalb der Menge

$$(\{c(e) \mid e \in F^*\} \cup z(F_c, c)) \cap [z(F_c, c), z(F^*, c)]$$

Sortiere diese Werte in einer aufsteigenden Reihenfolge, $z(F_c, c) = q_1 < q_2 < ... < q_t$.

Schritt 2:
Für jedes $p = (q_1, .., q_t)$ löse ein inverses Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert p, beginnend bei $p = q_t$:
Definiere für jedes p folgende Mengen:
$K_p := \{e \in E \mid c(e) < p\}$
$L_p := \{e \in F^* \mid c(e) > p\}$

Erstelle eine Kapazitätsmatrix:

$$w_p(e) = \begin{cases} 1, & \text{wenn } e \in K_p \\ 0, & \text{sonst} \end{cases}$$

Schritt 3: Löse innerhalb der Kapazitätsmatrix ein Engpass-Zuordnungsproblem

Fall 1: Der Zielfunktionswert des Engpass-Zuordnungsproblemes ist 0:
Dann erhält man als Kostenmatrix:

$$c^*(e) = \begin{cases} p, & \forall e \in L_p \\ c(e), & \text{sonst} \end{cases}$$

und der Zielfunktionswert lautet:

$$f_H(c^*, p) = \sum_{e \in L_p} h(c^*(e), c(e)) = z_L(p)$$

Fall 2: Der Zielfunktionswert des Engpass-Zuordnungsproblems ist größer als 0:
Dann wird mit Hilfe des Algorithmus aus 3.1 ein minimaler Blocker $B(p)$ gesucht. Für den Blocker gilt, dass $\sum_{e \in B(p)} w_p(e)$ minimal werden muss.

Die Kostenmatrix wird dann wie folgt definiert:

$$c^*(e) = \begin{cases} p, & \forall e \in \{e \in B(p) \mid w(e) = 1\} \cup L_p \\ c(e), & \text{sonst} \end{cases}$$

und der Zielfunktionswert lautet:

$$f_H(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) = z_K(p) + z_L(p)$$

Abbruchkriterien:

An dieser Stelle ist zu überprüfen, ob es bereits eine optimale Lösung gibt und der Algorithmus stoppen kann oder ob der Algorithmus weiterlaufen soll:

- Der Algorithmus kann abgebrochen werden, wenn der Zielfunktionswert $f_H(c^*, p) = 1$ ist. Dann kann keine bessere Lösung mehr gefunden werden.

- Weiters kann der Algorithmus abgebrochen werden, wenn für $p_2 > p_1$ gilt, dass $f_H(c^*, p_2) \leq f_H(c^*, p_1)$, da $f_H(c^*)$ eine unimodale Funktion ist.
- Der Algorithmus kann gestoppt werden, wenn für $p = q_i > q_{i-1}$ mit $i \in \{1, .., t\}$ gilt, dass

$$f_H(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) \leq \sum_{e \in L_{q_{i-1}}} h(c^*(e), c(e))$$

ist.
Das heißt mit anderen Worten: wenn in der vorgegebenen Lösung F^* die Anzahl der Einträge, die größer als p sind, größer gleich der Anzahl der geänderten Kosten $f_H(c^*, p) = \sum_{e \in E} h(c^*(e), c(e))$ ist, kann der Algorithmus beendet werden, da mit p bereits eine optimale Lösung gefunden wurde. Alle weiteren Lösungen können nur mehr gleich oder schlechter werden, als die bereits gefundene Lösung.

3.4 Beispiele für das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz

Beispiel:
Gegeben ist ein bipartiter Graph und eine Kostenmatrix C

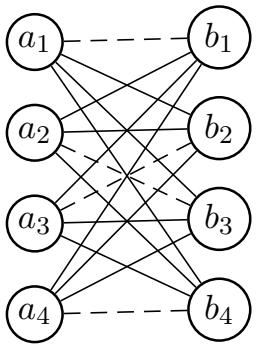


Abbildung 3.9: Zur Kostenmatrix C zugehöriger Graph. Die strichlierten Linien kennzeichnen die optimale Lösung F_c

$$C = \begin{pmatrix} \boxed{1} & 2 & 3 & 4 \\ 4 & 7 & \boxed{1} & 8 \\ 3 & \boxed{1} & 5 & 8 \\ 5 & 2 & 2 & \boxed{1} \end{pmatrix}$$

Weiters ist eine zulässige Lösung F^*, die optimal werden soll, gegeben. Die Einträge von F^* stehen in der Hauptdiagonalen der Kostenmatrix. Die Zahlen, die in der Kostenmatrix eingerahmt sind, zeigen die optimale Lösung des Engpass-Zuordnungsproblems F_c.

Gesucht ist nun eine Kostenfunktion c^*, sodass die gegebene Lösung F^* optimal wird.

Als erstes wird nach der Kandidatenmege für die möglichen Engpass-Kosten gesucht

$$(\{c(e) \mid e \in F^*\} \cup z(F_c, c)) \cap [z(F_c, c), z(F^*, c)]$$

Also die Menge

$$(\{1, 5, 7\} \cup \{1\}) \cap [1, 7]$$

und es ist ersichtlich, dass p nur die Werte 7, 5 oder 1 annehmen kann.

Fall $p = 7$:
$L_p = \{e \in F^* \mid c(e) > p\}$
$L_7 = \emptyset$

$K_p = \{e \in E \mid c(e) < p\}$
$K_7 = \{c_{11}, c_{12}, c_{13}, c_{14}, c_{21}, c_{23}, c_{31}, c_{32}, c_{33}, c_{41}, c_{42}, c_{43}, c_{44}\}$

Erstelle eine Kapazitätsmatrix:

$$w_p(e) = \begin{cases} 1, & \text{wenn } e \in K_7 \\ 0, & \text{sonst} \end{cases}$$

$$W_7 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Suche nun mit Hilfe des Algorithmus aus Kapitel 3.1 in der Kapazitätsmatrix nach einem minimalen Blocker $B(p)$.

Es existiert ein minimaler Blocker: $B(7) = \{c_{14}, c_{24}, c_{34}, c_{44}\}$

Definiere nun die geänderten Kosten:

$$c^*(e) = \begin{cases} p, & \forall e \in \{e \in B(p) \mid w(e) = 1\} \cup L_p \\ c(e), & \text{sonst} \end{cases}$$

Nun wird eine neue Kostenmatrix C^* erstellt:

$$C^* = \begin{pmatrix} 1 & 2 & 3 & \textcircled{7} \\ 4 & 7 & 1 & 8 \\ 3 & 1 & 5 & 8 \\ 5 & 2 & 2 & \textcircled{7} \end{pmatrix}$$

Die Werte, die in der Kostenmatrix C^* gekennzeichnet sind, müssen geändert werden, damit die vorgegebene Lösung F^* optimal wird.
Als Zielfunktionswert erhält man folgenden Wert: $f_H(c^*, p) = z_K(7) + z_L(7) = 2 + 0 = 2$.

Fall $p = 5$:
$L_5 = \{c_{22}\}$

$K_5 = \{c_{11}, c_{12}, c_{13}, c_{14}, c_{21}, c_{23}, c_{31}, c_{32}, c_{42}, c_{43}, c_{44}\}$

Die Kapazitätsmatrix hat dann folgende Gestalt:

$$W_5 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Es existiert ein minimaler Blocker: $B(5) = \{c_{14}, c_{24}, c_{34}, c_{44}\}$

Neue Kostenmatrix C^*:

$$C^* = \begin{pmatrix} 1 & 2 & 3 & \textcircled{5} \\ 4 & \textcircled{5} & 1 & 8 \\ 3 & 1 & 5 & 8 \\ 5 & 2 & 2 & \textcircled{5} \end{pmatrix}$$

Für den Zielfunktionswert gilt: $f_H(c^*, 5) = z_K(5) + z_L(5) = 2 + 1 = 3$.

An dieser Stelle kann der Algorithmus bereits abgebrochen werden, da schon eine optimale Lösung für das inverse Engpass-Zuordnungsproblem gefunden wurde. Da $f_H(c^*)$ eine unimodale Funktion ist und $f_H(c^*, 5) > f_H(c^*, 7)$ ist, kann der Algorithmus gestoppt werden. Alle weiteren Lösungen können nur mehr gleich oder schlechter werden.

Für $p = 7$ besitzt der Algorithmus eine optimale Lösung mit dem Zielfunktionswert $f_H(c^*) = 2$. Die geänderte Kostenmatrix hat folgende Gestalt:

$$C^* = \begin{pmatrix} 1 & 2 & 3 & 7 \\ 4 & 7 & 1 & 8 \\ 3 & 1 & 5 & 8 \\ 5 & 2 & 2 & 7 \end{pmatrix}$$

3.5 Inverse Engpass-Zuordnungsprobleme mit Hamming-Distanz mit oberen und unteren Schranken

In diesem Abschnitt soll das inverse Engpass-Zuordnungsproblem, bei dem obere und untere Schranken gegeben sind, gelöst werden.

Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, obere und untere Schranken $l(e) \geq 0$, $u(e) \geq 0$ für alle $e \in E$, die angeben um wie viel die Kosten verändert werden dürfen, und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_{Hb}(c^*) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall e \in E \end{aligned}$$

Die Prozedur um dieses Problem zu lösen funktioniert ähnlich dem im vorhergehenden Kapitel beschriebenen Algorithmus. Es wird wieder für jedes p aus einer Kandidatenmenge ein inverses Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert p gelöst und unter diesen Lösungen wird dann nach jener Lösung gesucht, bei der die Anzahl der geänderten Kosten minimal ist.

Als erstes wird der Engpass-Funktionswert untersucht und einige Veränderungen zum vorhergehenden Kapitel präsentiert:

Definition 15 *Durch* $\underline{c} = \max\{z(F_c, c), z(F^*, c-l)\}$ *wird eine größte untere Schranke für die Kosten der Engpass-Funktion* $z(F^*, c^*)$ *definiert.*

Lemma 8 *[18] Sei* F_c *eine optimale Lösung für das Engpass-Zuordnungsproblem,* F^* *eine vorgegebene Lösung, die optimal werden soll und* c^* *der gesuchte Kostenvektor, dann gilt:*

$$\underline{c} \leq z(F^*, c^*) = p \leq z(F^*, c)$$

Beweis:
Die Ungleichung $z(F^*, c-l) \leq z(F^*, c^*) \leq z(F^*, c)$ gilt auf Grund der Tatsache, dass $c(e) - l(e) \leq c^*(e)$ für alle $e \in E$ gilt.

Sei $z(F_c, c) = p_c$ und $z(F^*, c^*) = p^*$. Es wird gezeigt, dass $p_c \leq z(F^*, c^*)$ gilt, wenn $p_c \geq z(F^*, c-l)$ ist.

Für ein p wird eine Menge $L_p := \{e \in F^* \mid c(e) > p\}$ und einen Kostenvektor $\tilde{c}$ folgendermaßen definiert:

$$\tilde{c}(e) = \begin{cases} p_c, & \forall e \in L_{p_c} \\ c(e), & \text{sonst.} \end{cases}$$

Für jede zulässige Lösung F gilt, dass $z(F, c) \geq p_c$ und $z(F, \tilde{c}) \geq p_c = z(F^*, \tilde{c})$ ist.
Aus diesem Grund ist F^* eine optimale Engpass-Lösung mit dem Kostenvektor $\tilde{c}$ und $\tilde{c}$ ist eine zulässige Lösung für das inverse Engpass-Zuordnungsproblem.
Aufgrund der Optimalität von c^* gilt dann:

$$\sum_{e \in L_{p_c}} h(\tilde{c}(e), c(e)) = \sum_{e \in E} h(\tilde{c}(e), c(e)) \geq \sum_{e \in E} h(c^*(e), c(e)) \geq \sum_{e \in L_{p^*}} h(c^*(e), c(e))$$

und dies bedeutet, dass $p_c \leq p^*$ und damit $z(F_c, c) \leq z(F^*, c^*)$ ist.

□

Beispiel *Gegeben ist ein Graph, die zugehörige Kostenmatrix*

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 7 \end{pmatrix}$$

und eine untere Schranke $l(e) = 4$ *für alle* $e \in E$ *und eine zulässige Lösung* F^*.
Dann gilt:

$$z(F_c, c) = 2 \leq z(F^*, c^*) = 3 \leq z(F^*, c) = 7$$
$$z(F^*, c-l) = 3 \leq z(F^*, c^*) = 3 \leq z(F^*, c) = 7$$

Auf Grund dieser Tatsachen kommt als Lösung für das Engpass-Zuordnungsproblem nur jene Kosten $c(e)$ mit $e \in F^*$ in Frage, die im Intervall $[\underline{c}, z(F^*, c)]$ liegen.

Daher nimmt der Engpass-Funktionswert einen Wert an, der in der folgenden Menge liegt (vgl. dazu Lemma 7):

$$(\{c(e)|e \in F^*\} \cup (\{c(e) + u(e)|e \in F^*\} \cup \{\underline{c}\}) \cap [\underline{c}, z(F^*, c)]$$

Nun wird ein Kriterium präsentiert, mit dessen Hilfe überprüft werden kann, ob ein inverses Engpass-Zuordnungsproblem überhaupt eine zulässige Lösung besitzt:

Lemma 9 *Gegeben ist eine Kostenmatrix C, obere und untere Schranken $l(e)$ und $u(e)$ für alle $e \in E$ und eine zulässige Lösung F^*, die optimal werden soll. Es wird dann jede Kante $e \in E$ auf ihre obere Schranke gesetzt, dh. es wird eine Kostenfunktion $\overline{c}$ folgendermaßen definiert:*

$$\overline{c}(e) = c(e) + u(e), \ \forall e \in E$$

Wenn eine Optimallösung F_0 existiert mit $z(F_0, \overline{c}) < z(F^*, c - l)$, dann existiert keine zulässige Lösung für das inverse Engpass-Zuordnungsproblem.

Beweis:
Da die Kostenelemente $c(e) \in E \backslash F^*$ nur erhöht werden, sind die unteren Schranken für $e \in E \backslash F^*$ irrelevant. Weiters ist aus dem vorhergehenden Lemma bekannt, dass $\underline{c} \leq z(F^*, c^*)$ gilt. Aus diesem Grund können auch die unteren Schranken für $e \in F^*$ außer Acht gelassen werden. Wenn es dann eine Lösung F_0 gibt mit $z(F_0, \overline{c}) < z(F^*, c - l)$, dann kann es keine zulässige Lösung geben, deren Engpass-Funktionswert größer als $z(F^*, c - l)$. Aus diesem Grund existiert keine zulässige Lösung für das inverse Engpass-Zuordnungsproblem.

□

Beispiel *Gegeben ist ein Graph, die zugehörige Kostenmatrix*

$$C = \begin{pmatrix} 5 & 11 & 9 & 6 & 14 \\ 8 & 3 & 11 & 13 & 16 \\ 6 & 8 & 3 & 9 & 13 \\ 2 & 8 & 17 & 7 & 6 \\ 12 & 9 & 6 & 7 & 12 \end{pmatrix}$$

obere und untere Schranke mit $l(e) = u(e) = 2$ für alle $e \in E$ und eine zulässige Lösung F^*.

Die veränderte Kostenmatrix $\overline{C}$ hat dann folgende Gestalt:

$$\overline{C} = \begin{pmatrix} 7 & 13 & 11 & \boxed{8} & 16 \\ 10 & \boxed{5} & 13 & 15 & 18 \\ \boxed{8} & 10 & 5 & 11 & 15 \\ 4 & 10 & 19 & 9 & \boxed{8} \\ 14 & 11 & 8 & \boxed{9} & 14 \end{pmatrix}$$

In dieser Matrix existiert eine Optimallösung F_0 mit $z(F_0, \overline{c}) = 8$, die in der Matrix $\overline{C}$ durch die eingerahmten Werten gekennzeichnet ist. Da aber $z(F^*, c - l) = 10$ ist, existiert keine zulässige Lösung für das inverse Engpass-Zuordnungsproblem.

3.6 Algorithmus um das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz mit oberen und unteren Schranken zu lösen

In diesem Abschnitt wird nun eine Prozedur präsentiert, um ein inverses Engpass-Zuordnungsproblem mit oberen und unteren Schranken zu lösen.

Gegeben ist:

- bipartiter Graph $G = (V, E)$,
- eine zulässige Lösung F^*, die optimal werden soll,
- eine Kostenfunktion $c(e)$ für alle $e \in E$,
- obere und untere Schranken $l(e), u(e)$ für alle $e \in E$.

Gesucht ist eine Kostenfunktion c^*, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_{Hb}(c^*) := \sum_{e \in E} h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Das inverse Engpass-Zuordnungsproblem mit Schranken wird in ähnlicher Weise wie das unbeschränkte Problem gelöst. Zu beachten ist lediglich, dass bei der Definition der Kapazitätsmatrix überprüft werden muß, ob die Kante zulässig ist [18].

Schritt 1:
Überprüfe ob es eine Optimallösung F_0 mit $z(F_0, c+u) < z(F^*, c-l)$ gibt. Existiert eine Lösung F_0, dann besitzt das inverse Engpass-Zuordnungsproblem keine zulässige Lösung.

Schritt 2:
Suche nach den möglichen Werten für die Engpass-Funktion innerhalb der Menge

$$(\{c(e) \mid e \in F^*\} \cup (\{c(e) + u(e) \mid e \in F^*\} \cup \{\underline{c}\}) \cap [\underline{c}, z(F^*, c)]$$

und sortiere diese Werte in einer aufsteigenden Reihenfolge, $\underline{c} = q_1 < q_2 < ... < q_t$.

Schritt 3:
Für jedes $p = (q_1, .., q_t)$ löse ein inverses Engpass-Zuordnungsproblem mit vorgegebenen Engpass-Funktionswert p, beginnend bei $p = g_t$:
Definiere für jedes p folgende Mengen:
$K_p := \{e \in E \mid c(e) < p\}$
$L_p := \{e \in F^* \mid c(e) > p\}$
$\bar{K}_p := \{e \in K_p \mid p \leq c(e) + u(e)\}$

Erstelle eine Kapazitätsmatrix:

$$w_p(e) = \begin{cases} 1, & \text{wenn } e \in \bar{K}_p \\ \infty, & \forall e \in K_p \backslash \bar{K}_p \\ 0, & \text{sonst} \end{cases}$$

Schritt 4: Löse innerhalb der Kapazitätsmatrix ein Engpass-Zuordnungsproblem:

Fall 1: Der Zielfunktionswert des Engpass-Zuordnungsproblemes ist 0:

$$c^*(e) = \begin{cases} p, & \forall e \in L_p \\ c(e), & \text{sonst} \end{cases}$$

und der Zielfunktionswert lautet:

$$f_{Hb}(c^*, p) = \sum_{e \in L_p} h(c^*(e), c(e))$$

Fall 2: Der Zielfunktionswert des Engpass-Zuordnungsproblemes ist größer als 0:
Dann wird mit Hilfe des Algorithmus aus 3.1 ein minimaler Blocker $B(p)$ gesucht. Für den Blocker gilt, dass $\sum_{e \in B(p)} w_p(e)$ minimal werden muss. Existiert kein Blocker mit $\sum_{e \in B(p)} w_p(e) < \infty$, dann ist das Problem unlösbar.

Die Kostenmatrix wird dann wie folgt definiert:

$$c^*(e) = \begin{cases} p, & \forall e \in \{e \in B(p) \mid w(e) \neq 0\} \cup L_p \\ c(e), & \text{sonst} \end{cases}$$

und der Zielfunktionswert lautet:

$$f_{Hb}(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) = z_K(p) + z_L(p)$$

Abbruchkriterien:
An dieser Stelle ist zu überprüfen, ob es bereits eine optimale Lösung gibt und der Algorithmus stoppen kann oder ob der Algorithmus weiterlaufen soll:

- Der Algorithmus kann abgebrochen werden, wenn der Zielfunktionswert $f_{Hb}(c^*, p) = 1$ ist. Dann kann keine bessere Lösung mehr gefunden werden.

- Weiters kann der Algorithmus abgebrochen werden, wenn für $p_2 > p_1$ gilt, dass $f_H(c^*, p_2) \leq f_H(c^*, p_1)$, da $f_H(c^*)$ eine unimodale Funktion ist.

- Der Algorithmus kann gestoppt werden, wenn für $p = q_i > q_{i-1}$ mit $i \in \{1, .., t\}$ gilt, dass

  $$f_{Hb}(c^*, p) = \sum_{e \in E} h(c^*(e), c(e)) \leq \sum_{e \in L_{q_{i-1}}} h(c^*(e), c(e))$$

  ist.
  Das heißt mit anderen Worten: wenn in der vorgegebenen Lösung F^* die Anzahl der Einträge, die größer als p sind, größer gleich der Anzahl der geänderten Kosten $f_{Hb}(c^*, p) = \sum_{e \in E} h(c^*(e), c(e))$ ist, kann der Algorithmus beendet werden, da mit p bereits eine optimale Lösung gefunden wurde. Alle weiteren Lösungen können nur mehr gleich oder schlechter werden, als die bereits gefundene Lösung.

3.7 Beispiel für das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz mit oberen und unteren Schranken

Beispiel:
Gegeben ist ein bipartiter Graph und eine Kostenmatrix C,

$$C = \begin{pmatrix} 7 & \boxed{2} & 2 & 1 \\ \boxed{2} & 3 & 5 & 9 \\ 8 & 4 & 3 & \boxed{1} \\ 3 & 8 & \boxed{2} & 1 \end{pmatrix}$$

obere und untere Schranken mit $l(e) = 5$ und $u(e) = 2$ für alle $e \in E$ und eine zulässige Lösung F^*, die optimal werden soll. Die Zahlen, die in der Kostenmatrix eingerahmt sind, zeigen die optimale Lösung des Engpass-Zuordnungsproblems F_c.

Gesucht ist nun eine Kostenfunktion c^*, sodass die gegebene Lösung F^* optimal wird.

Als erstes wird überprüft, ob das inverse Engpass-Problem überhaupt eine zulässige Lösung besitzt. Dazu muß getestet werden, ob es eine Optimallösung F_0 mit $z(F_0, c+u) < z(F^*, c-l)$ gibt. Bei diesem Beispiel ist $z(F_0, c+u) = 4$ und $z(F^*, c-l) = 2$. Es existiert also eine zulässige Lösung.

Als nächstes wird nach der Kandidatenmenge für die möglichen Engpass-Kosten gesucht

$$(\{c(e) \mid e \in F^*\} \cup (\{c(e) + u(e) \mid e \in F^*\} \cup \underline{c}) \cap [\underline{c}, z(F^*, c)]$$

Also die Menge

$$(\{1, 3, 7\} \cup \{3, 5, 9\} \cup \{2\}) \cap [2, 7]$$

und es ist ersichtlich, dass p nur die Werte 7, 5, 3 oder 2 annehmen kann.

Fall $p = 7$:
$L_p = \{e \in F^* \mid c(e) > p\}$
$L_7 = \emptyset$

$K_p = \{e \in E \mid c(e) < p\}$
$K_7 = \{c_{12}, c_{13}, c_{14}, c_{21}, c_{22}, c_{23}, c_{32}, c_{33}, c_{34}, c_{41}, c_{43}, c_{44}\}$

$\bar{K}_p = \{e \in K_p \mid p \leq c(e) + u(e)\}$
$\bar{K}_7 = \{c_{23}\}$

Erstelle eine Kapazitätsmatrix:

$$w_p(e) = \begin{cases} 1, & \text{wenn } e \in \bar{K}_p \\ \infty, & \forall e \in K_p \backslash \bar{K}_p \\ 0, & \text{sonst} \end{cases}$$

$$W_7 = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & \infty & 1 & 0 \\ 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \end{pmatrix}$$

Suche nun mit Hilfe des Algorithmus aus Kapitel 3.1 in der Kapazitätsmatrix nach einem minimalen Blocker $B(p)$.

Es existiert kein minimaler Blocker mit $\sum_{e \in B(p)} w_p(e) < \infty$. Für $p = 7$ kann deshalb keine Lösung gefunden werden.

Fall $p = 5$:
$L_5 = \{c_{11}\}$
$K_5 = \{c_{12}, c_{13}, c_{14}, c_{21}, c_{22}, c_{32}, c_{33}, c_{34}, c_{41}, c_{43}, c_{44}\}$
$\bar{K}_7 = \{c_{22}, c_{32}, c_{33}, c_{41}\}$

$$W_5 = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 1 & 0 & 0 \\ 0 & 1 & 1 & \infty \\ 1 & 0 & \infty & \infty \end{pmatrix}$$

Es existiert kein minimaler Blocker mit $\sum_{e \in B(p)} w_p(e) < \infty$. Für $p = 5$ kann deshalb keine Lösung gefunden werden.

Fall $p = 3$:
$L_3 = \{c_{11}, c_{23}, c_{24}, c_{31}, c_{42}\}$
$K_3 = \{c_{12}, c_{13}, c_{14}, c_{21}, c_{34}, c_{43}, c_{44}\}$
$\bar{K}_3 = \{c_{12}, c_{13}, c_{14}, c_{21}, c_{34}, c_{43}, c_{44}\}$

Die Kapazitätsmatrix hat dann folgende Gestalt:

$$W_3 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Es existiert ein minimaler Blocker: $B(3) = \{c_{21}, c_{22}, c_{23}, c_{24}\}$

Neue Kostenmatrix C^*:

$$C^* = \begin{pmatrix} \textcircled{3} & 2 & 2 & 1 \\ \textcircled{3} & 3 & 5 & 9 \\ 8 & 4 & 3 & 1 \\ 3 & 8 & 2 & 1 \end{pmatrix}$$

Für den Zielfunktionswert gilt: $f_{Hb}(c^*, 3) = z_K(3) + z_L(3) = 1 + 1 = 2$.

An dieser Stelle kann der Algorithmus bereits abgebrochen werden, da schon eine optimale Lösung für das inverse Engpass-Zuordnungsproblem gefunden wurde. Für den Engpass-Funktionswert $p = 2$ sind zumindest drei Werte zu ändern, da in der vorgegeben Lösung F^* drei Werte größer als zwei sind.

Der Algorithmus liefert eine optimale Lösung für das inverse Engpass-Zuordnungsproblem. Für $p = 3$ ist die Anzahl der Kanten, die zu ändern sind, minimal und der Zielfunktionswert lautet $f_{Hb}(c^*) = 2$.

3.8 Laufzeit

In diesem Abschnitt wird die Gesamtlaufzeit des Algorithmus genauer untersucht. Aus Kapitel 3.2 ist bereits bekannt, dass das Problem einen minimalen Blocker zu finden $\mathcal{NP}$-vollständig ist. Weiter wurde in Kapitel 3.1 bereits festgestellt, dass der Algorithmus, der einen minimalen Blocker zu findet, eine Laufzeit von $\mathcal{O}(n\ 2^{2n})$ besitzt.

Weiters ist bereits aus Theorem 2 bekannt, dass das inverse Engpass-Zuordnungsproblem mit Hamming-Distanz zu $\mathcal{O}(n)$ Problemen einen minimalen Blocker zu finden, reduziert werden kann.

Daraus resultiert eine Gesamtlaufzeit von $\mathcal{O}(n^2\ 2^{2n})$ für den Algorithmus.

4 Inverses Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz

In diesem Abschnitt wird das inverse Engpass-Zuordnungsproblem mit einer gewichteten Engpass-Hamming-Distanzfunktion genau untersucht. Bei diesem Problem soll das größte Gewicht der veränderten Kanten minimal werden.

Das inverse Engpasszuordnungsproblem kann folgendermaßen definiert werden:

Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, eine Gewichtsfunktion $w(e)$ für alle $e \in E$, obere und untere Schranken $l(e) \geq 0$, $u(e) \geq 0$ für alle $e \in E$, die angeben um wie viel die Kosten verändert werden dürfen, und eine zulässige Lösung F^*. Die Aufgabe ist nun eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$
\begin{aligned}
\min \quad & f_{bH}(c^*) := \max_{e \in E} w(e)\ h(c^*(e), c(e)) \\
\text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\
& c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall e \in E
\end{aligned}
$$

Zuerst wird dieses Problem ohne obere und untere Schranken betrachtet und ein Algorithmus dazu präsentiert [15]. Im weiteren Verlauf sind dann zusätzlich obere und untere Schranken gegeben und es wird gezeigt, wie diese Aufgabe gelöst werden kann [18].

4.1 Algorithmus um das inverse Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz zu lösen

Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, eine Gewichtsfunktion $w(e)$ für alle $e \in E$ und eine zulässige Lösung F^*. Der Einfachheit halber

wird die Menge $E = \{e_1, ..e_m\}$ umgeordnet, sodaß $0 < w(e_1) \leq ... \leq w(e_m)$ gilt. Es wird dann nach einer Kostenfunktion c^* gesucht, bei der das größte Gewicht der veränderten Kanten minimal wird.

$$\begin{aligned} \min \quad & \max_{e \in E} w(e)\ h(c^*(e), c(e)) \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall F \in \mathcal{F} \end{aligned}$$

Die Elemente $e_j \in F^*$ $(j = 1, .., \sigma)$ werden Ast (branch) von F^* genannt und die Elemente $e_j \in E \backslash F^*$ $(j = \sigma + 1, .., m)$ Wurzel (chord) genannt.

Anmerkung: Werden die Kosten der Äste erhöht oder die Kosten der Wurzeln verringert, bringt dies F^* nicht näher zur Optimalität. Aus diesem Grund werden nur folgende Veränderungen der Kosten für die Kante e mit $c^* \neq c$ durchgeführt: Wenn die Kante e ein Ast von F^* ist, dann werden die neuen Kosten $c^*(e) = -M$ gesetzt, dabei sei M ein beliebiger sehr großer Wert. Ist hingegen e eine Wurzel von F^*, dann wird $c^*(e) = M$ gesetzt.

Algorithmus für das inverse Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz [15]:

Step 1 Nummeriere die Elemente von E sodass $w(e_1) \leq w(e_2) \leq ... \leq w(e_m)$ gilt
$k := 0$, $w(e_0) := 0$, $c^* = c$,
Löse ein Engpass-Zuordnungsproblem mit Kosten c^* und berechne
$\delta^* = \min\{\max_{e \in F} c^*(e) \mid F \text{ ist zulässig}\}$

Step 2 While $\max_{e \in F^*} c^*(e) > \delta^*$ do
$k := k + 1$
Wenn e_k mit dem Gewicht $w(e_k)$ ein Ast ist, dann setze $c^*(e_k) := -M$, sonst $c^*(e_k) := M$
Löse das Engpass-Zuordnungsproblem mit den neuen Kosten c^* und aktualisiere $\delta*$

Step 3 Stoppe mit dem optimalen Wert $w^* := w(e_k)$

Bei diesem Algorithmus wird mit der Lösung $c^* = c$ gestartet und es wird in jedem Schritt eine Lösung für das Engpass-Zuordnungsproblem berechnet. Der Algorithmus überprüft in jedem Schritt die Optimalität der Werte w_k und endet sobald $\max_{e \in F^*} c^*(e) \leq \delta^*$ gilt.

Laufzeit: Der Algorithmus besitzt eine Laufzeit von $\mathcal{O}(m\ A(m,n))$. $A(m,n)$ ist dabei die Zeit, die benötigt wird um ein Engpass-Zuordnungsproblem zu lösen. Wird für die Lösung eines solchen Problems die Methode von Punnen und Nair verwendet, dann ist $A(m,n) = \mathcal{O}(n\sqrt{mn})$. Durch eine Binärsuche kann der Algorithmus noch beschleunigt werden und man erhält eine Laufzeit von $\mathcal{O}((A(m,n)\log\ m)$. Als Gesamtlaufzeit ergibt sich somit $\mathcal{O}(n\sqrt{mn}\log\ m)$. Der Algorithmus kann noch verbessert werden, da im 2. Schritt nur reoptimiert wird.

4.2 Beispiel für das inverse Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz

Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, eine Gewichtsfunktion $w(e)$ für alle $e \in E$ und eine Lösung F^*, die optimal werden soll.

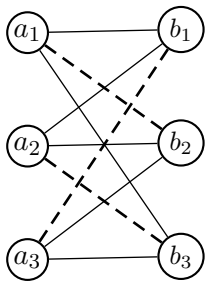


Abbildung 4.1: Zur Kostenmatrix C zugehöriger Graph. Die strichlierten Linien kennzeichnen die Lösung, die optimal werden soll

$$C = \begin{pmatrix} 1 & \boxed{3} & 7 \\ 2 & 5 & \boxed{8} \\ \boxed{4} & 6 & 9 \end{pmatrix} \qquad W = \begin{pmatrix} 3 & 2 & 2 \\ 4 & 5 & 4 \\ 2 & 1 & 5 \end{pmatrix}$$

Die markierten Einträge in der Kostenmatrix C kennzeichnen die gegebene Lösung F^*, die optimal werden soll.

Step 1 Nummeriere die Elemente von E um, sodass $w(e_1) \leq w(e_2) \leq ... \leq w(e_m)$ gilt $k := 0$, $w(e_0) := 0$, $c^* = c$,

$$C^* = \begin{pmatrix} w(e_5) & w(e_3) & w(e_2) \\ w(e_6) & w(e_8) & w(e_7) \\ w(e_4) & w(e_1) & w(e_9) \end{pmatrix}$$

Löse ein Engpass-Zuordnungsproblem mit Kosten c^* und berechne
$\delta^* = \min\{\max_{e \in F} c^*(e) \mid F \text{ ist zulässig}\} = 7$

Step 2 While $\max_{e \in F^*} c^*(e) = 8 > 7 = \delta^*$ do
k=1
$w(e_1)$ mit dem Gewicht 1 ist kein Ast, setze $c^*(w(e_1)) := M$

$$C^* = \begin{pmatrix} 1 & 3 & 7 \\ 2 & 5 & 8 \\ 4 & M & 9 \end{pmatrix}$$

Löse das Engpass-Zuordnungsproblem mit den aktualisierten Kosten c^*, dann ist $\delta^* = 7$

k=2
$w(e_2)$ mit dem Gewicht 2 ist kein Ast, setze $c^*(w(e_2)) := M$

$$C^* = \begin{pmatrix} 1 & 3 & M \\ 2 & 5 & 8 \\ 4 & M & 9 \end{pmatrix}$$

Löse das Engpass-Zuordnungsproblem mit den aktualisierten Kosten c^*, dann ist $\delta^* = 8$

Step 3 Stoppe mit dem optimalen Wert $w^* = w(e_2) = 2$

Als Lösung für dieses Beispiel erhält man den Kostenvektor

$$C^* = \begin{pmatrix} 1 & 3 & M \\ 2 & 5 & 8 \\ 4 & M & 9 \end{pmatrix}$$

und den Zielfunktionswert $f_{bH}(c^*) := \max_{e \in E} w(e)\ h(c^*(e), c(e)) = 2$.

4.3 Inverses Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz mit oberen und unteren Schranken

Bei diesem Problem sind noch zusätzlich obere und untere Schranken gegeben, die angeben, um wieviel sich der gesuchte Kostenvektor c^* ändern darf.

Es soll folgendes Problem genauer untersucht werden:
Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, eine

Gewichtsfunktion $w(e)$ für alle $e \in E$, obere und untere Schranken $l(e) \geq 0$, $u(e) \geq 0$ für alle $e \in E$, die angeben um wieviel die Kosten verändert werden dürfen, und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_{bH}(c^*) := \max_{e \in E} w(e)\ h(c^*(e), c(e)) \\ \text{s.t} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Die folgenden Überlegungen und Lösungsmethoden lehnen sich an die Arbeiten von Guan und Zhang [18] an.

Zuerst werden einige wichtige Definitionen und Tatsachen gezeigt um dann einen Algorithmus zu präsentieren, der dieses Problem löst.
Für ein gegebenes Kantengewicht $w(e_{i_k})$, definiere $E_k := \{e \in E \mid w(e) \leq w(e_{i_k})\}$,

$$c^k(e) = \begin{cases} c(e) - l(e), & \text{wenn } e \in E_k \cap F^*, \\ c(e) + u(e), & \text{wenn } e \in E_k \backslash F^*, \\ c(e), & \text{wenn } e \notin E_k \end{cases}$$

$\max_{e \in F^*} c^k(e) := p_k$, und $E_k^+ := \{e \in E \mid c^k(e) \geq p^k\}$. Sei

$$c^{p_k}(e) = \begin{cases} p_k, & \text{wenn } e \in E_k \cap F^*, p_k \leq c(e) + u(e), \\ p_k, & \text{wenn } e \in E_k \backslash F^*, c(e) - l(e) \leq p_k \leq c(e) + u(e), \\ c^k(e), & \text{sonst} \end{cases}$$

und $E_{p_k}^+ := \{e \in E \mid c^{p_k} \geq p_k\}$.

Dann gelten die folgenden Tatsachen:

Lemma 10 *Wenn E_k^+ ein Blocker von $\mathcal{F}$ ist, dann ist $E_{p_k}^+$ auch ein Blocker von $\mathcal{F}$.*

Beweis:
Bei diesem Beweis genügt es zu zeigen, dass $E_k^+ \subseteq E_{p_k}^+$ gilt. Da $e \in E_k^+$ ist, gilt: wenn $c^{p_k}(e) = p_k \leq c^k(e)$ ist, dann folgt daraus, dass $e \in E_k^+ \cap E_{p_k}^+$ ist. Wenn $c^{p_k}(e) = c^k(e) \geq p_k$ ist, dann folgt, dass $e \in E_k^+ \cap E_{p_k}^+$ ist. Aus diesem Grund gilt für jede Kante e, die in E_k^+ liegt, dass diese Kante auch in $E_{p_k}^+$ liegt.

□

Lemma 11 *Wenn $E^+_{p_k}$ kein Blocker von $\mathcal{F}$ ist, dann gilt für jedes $\tilde{c}$ mit $c - l \leq \tilde{c} \leq c + u$ und $f_{bH}(\tilde{c}) = w(e_{i_k})$, dass $\tilde{E}^+ := \{e \in E \mid \tilde{c}(e) \geq \tilde{p} = \max_{e \in F^*} \tilde{c}(e)\}$ kein Blocker von $\mathcal{F}$ ist.*

Beweis:
Als erstes wird gezeigt, dass $\tilde{p} \geq p_k$ gilt.

$$\tilde{p} = \max_{e \in F^*} \tilde{c}(e) = \max\{\max_{e \in F^* \cap E_k} \tilde{c}(e), \max_{e \in F^* \setminus E_k} \tilde{c}(e)\} = \max\{\max_{e \in F^* \cap E_k} \tilde{c}(e), \max_{e \in F^* \setminus E_k} c(e)\} \geq$$
$$\max\{\max_{e \in F^* \cap E_k} (c(e) - l(e)), \max_{e \in F^* \setminus E_k} c(e)\} = p_k$$

Als nächstes wird gezeigt, dass $\tilde{E}^+ \subseteq E^+_{p_k}$ gilt:
Wenn $\tilde{c}(e) \geq \tilde{p} \geq p_k = c^{p_k}(e)$ gilt, dann ist $e \in \tilde{E}^+ \cap E^+_{p_k}$. Wenn $e \in \tilde{E}^+ \cap (E_k \setminus F^*)$ mit $p_k < c(e) - l(e)$, dann gilt $c^{p_k}(e) = c(e) + u(e) \geq \tilde{c}(e) \geq \tilde{p} \geq p_k$ und deshalb folgt $e \in E^+_{p_k}$. Wenn $e \in \tilde{E}^+ \setminus E_k$, dann gilt $c^{p_k}(e) = \tilde{c}(e) \geq \tilde{p} \geq p_k$ und $e \in E^+_{p_k}$. Wenn $e \in \tilde{E}^+ \cap E_k$ mit $p_k > c(e) + u(e)$, dann gilt $\tilde{c}(e) \geq \tilde{p} \geq p_k > c(e) + u(e)$. Dies widerspricht die Zulässigkeit von $\tilde{c}$ und der Fall kann nicht auftreten.

□

Es folgt von dem vorhergehenden Lemma, wenn $E^+_{p_k}$ kein Blocker von $\mathcal{F}$ ist, dann existiert keine zulässige Lösung $\tilde{c}$ für das inverse Engpass-Zuordnungsproblem mit $f_{bH}(\tilde{c}) \leq w(e_{i_k})$. Aus diesem Grund ist p_k der kleinste Wert, für den das inverse Engpassproblem eine zulässige Kostenfunktion $\tilde{c}$ mit $f_{bH}(\tilde{c}) \leq w(e_{i_k})$ besitzt.

Als nächstes wird ein Algorithmus vorgestellt, der das inverse Engpass-Zuordnungsproblem mit einer Engpass-Hamming-Distanz löst. Mit Hilfe einer Binärsuche unter den Gewichten $w(e)$ für $e \in E$ wird in jeder Iteration überprüft, ob die Menge $E^+_{p_k}$ ein Blocker von $\mathcal{F}$ ist oder nicht. Insbesondere wenn $w(e_{i_k}) = \max_{e \in E} w(e)$ und $E^+_{p_k}$ kein Blocker von $\mathcal{F}$ ist, dann ist das Problem nicht lösbar.
Sei $\lceil x \rceil$ die kleinste ganze Zahl, die größer oder gleich x ist.

Algorithmus für das inverse Engpass-Zuordnungsproblem mit oberen und unteren Schranken:

Step 1 Sortiere die Elemente von E sodass $w(e_{i_1}) < w(e_{i_2}) < ... < w(e_{i_\tau})$ gilt,
Setze $a := 1$, $b := \tau$,

Step 2 Wenn $E^+_{p_\tau}$ kein Blocker von $\mathcal{F}$ ist, dann ist das Problem unlösbar.

Step 3 Wenn $b - a = 1$, dann gebe die veränderten Kosten c^b und den Zielfunktionswert $f_{bH}(c^b) = w(e_{i_b})$ aus, stop. Ansonsten gehe zu Step 4.

Step 4 Sei $k := \lceil \frac{a+b}{2} \rceil$, $E_k := \{e \in E \mid w(e) \leq w(e_{i_k})\}$, $p_k := \max_{e \in F^*} c^k(e)$ und $E^+_{p_k} := \{e \in E \mid c^{p_k}(e) \geq p_k\}$, mit c^k und c^{p_k} wie vorhin definiert. Wenn $E^+_{p_k}$ ein Blocker von $\mathcal{F}$ ist, dann setze $b := k$, ansonsten setzte $a := k$. Kehre zu Step 3 zurück.

Theorem 9 *[18]*
Das inverse Engpass-Zuordnungsproblem unter der gewichteten Engpass-Hamming-Distanz kann zu $\mathcal{O}(log\ m)$ Problemen, die überprüfen ob $E^+_{p_k}$ ein Blocker ist, reduziert werden.

Laufzeit: Das Theorem besagt, dass der Algorithmus $\mathcal{O}(log\ m)$ Probleme in $E^+_{p_k}$ einen Blocker zu finden, löst. Das Problem ob es in $E^+_{p_k}$ einen Blocker gibt, ist äquivalent zu dem Problem ob in der Menge $E \backslash E^+_{p_k}$ ein perfektes Matching gefunden werden kann. Der Algorithmus, der mit Hilfe von augmentierenden Wegen überprüft, ob es in einem Graphen ein perfektes Matching gibt, benötigt eine Laufzeit von $\mathcal{O}(nm)$ [22]. Also erhält man eine Gesamtlaufzeit von $\mathcal{O}(nm\ log\ m)$.

4.4 Beispiel für das inverse Engpass-Zuordnungsproblem unter der Engpass-Hamming-Distanz mit oberen und unteren Schranken

Gegeben sind dieselben Werte wie im vorhergehenden Beispiel. Zusätzlich werden jetzt noch obere und untere Schranken $l(e) = 5$ für alle $e \in E$, $u(e) = 10$ für alle $e \in E$ definiert.

$$C = \begin{pmatrix} 1 & \boxed{3} & 7 \\ 2 & 5 & \boxed{8} \\ \boxed{4} & 6 & 9 \end{pmatrix} \qquad W = \begin{pmatrix} 3 & 2 & 2 \\ 4 & 5 & 4 \\ 2 & 1 & 5 \end{pmatrix}$$

Die markierten Einträge in der Kostenmatrix C kennzeichnen die gegebene Lösung F^*, die optimal werden soll.

Step 1 Sortiere die Elemente von E sodass $w(e_1) < w(e_2) < ... < w(e_9)$ gilt,
$a := 1$, $b := 9$,

$$\begin{pmatrix} w(e_5) & w(e_3) & w(e_2) \\ w(e_6) & w(e_8) & w(e_7) \\ w(e_4) & w(e_1) & w(e_9) \end{pmatrix}$$

Step 2 Überprüfe ob $E^+_{p_9}$ ein Blocker von $\mathcal{F}$ ist.
Wenn $E^+_{p_9}$ kein Blocker von $\mathcal{F}$ ist, dann ist das Problem unlösbar.
$E_9 = \{e \in E \mid w(e) \leq w(e_9)\}$ = alle Kanten $e \in E$.

$$c^9(e) = \begin{pmatrix} 11 & 0 & 17 \\ 12 & 15 & 3 \\ 0 & 16 & 19 \end{pmatrix}$$

$\max_{e \in F^*} c^9(e) = 3$

$$c^{p_9}(e) = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 19 \end{pmatrix}$$

$E^+_{p_9} = \{e \in E \mid c^{p_9} \geq 3\}$ = alle Kanten $e \in E$.
$E^+_{p_9}$ ist ein Blocker und deshalb existiert eine Lösung für dieses Problem.

Step 3 k=5

$E_5 = \{e \in E \mid w(e) \leq 3\} = \{(a_1, b_1), (a_1, b_3), (a_2, b_1), (a_2, b_3), (a_3, b_1)\}$

$$c^5(e) = \begin{pmatrix} 11 & -2 & 17 \\ 2 & 5 & 8 \\ -1 & 16 & 9 \end{pmatrix}$$

$\max_{e \in F^*} c^5(e) = 8$

$$c^{p_5}(e) = \begin{pmatrix} 8 & 8 & 8 \\ 2 & 5 & 8 \\ 8 & 8 & 9 \end{pmatrix}$$

$$E^+_{p_5}(e) = \{e \in E \mid c^{p_5} \geq 8\} = \{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_3), (a_3, b_1), (a_3, b_2), (a_3, b_3)\}$$

In $E \backslash E^+_{p_5}$ kann kein perfektes Matching gefunden werden und deshalb ist $E^+_{p_5}$ ein Blocker.
Setzte $b = 5$.

k=3
$E_3 = \{e \in E \mid w(e) \leq 2\} = \{(a_1, b_2), (a_1, b_3), (a_3, b_2)\}$

$$c^3(e) = \begin{pmatrix} 1 & -2 & 17 \\ 2 & 5 & 8 \\ 4 & 16 & 9 \end{pmatrix}$$

$\max_{e \in F^*} c^3(e) = 8$

$$c^{p_3}(e) = \begin{pmatrix} 1 & 8 & 8 \\ 2 & 5 & 8 \\ 4 & 8 & 9 \end{pmatrix}$$

$E^+_{p_3}(e) = \{e \in E \mid c^{p_3} \geq 8\} = \{(a_1, b_2), (a_1, b_3), (a_2, b_3), (a_3, b_2), (a_3, b_3)\}$.
Der Graph $G = (V, E \backslash E^+_{p_3})$ hat folgende Gestalt:

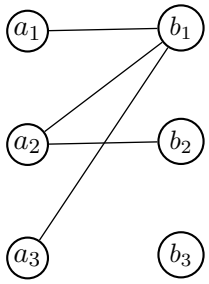


Abbildung 4.2: Graph $G = (V, E \backslash E^+_{p_3})$

In $E \backslash E^+_{p_3}$ kann kein perfektes Matching gefunden werden und deshalb ist $E^+_{p_3}$ ein Blocker.
Setzte $b = 3$.

k=2
$E_2 = \{e \in E \mid w(e) \leq 2\} = \{(a_1, b_3), (a_3, b_2)\}$

$$c^2(e) = \begin{pmatrix} 1 & 3 & 17 \\ 2 & 5 & 8 \\ 4 & 16 & 9 \end{pmatrix}$$

$\max_{e \in F^*} c^2(e) = 8$

$$c^{p_2}(e) = \begin{pmatrix} 1 & 3 & 8 \\ 2 & 5 & 8 \\ 4 & 8 & 9 \end{pmatrix}$$

$E^+_{p_2}(e) = \{e \in E \mid c^{p_2} \geq 8\} = \{(a_1, b_3), (a_2, b_3), (a_3, b_2), (a_3, b_3)\}$.

Der Graph $G = (V, E \backslash E^+_{p_2})$:

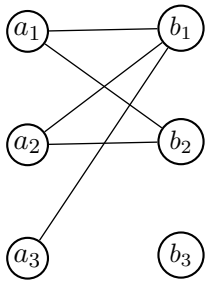


Abbildung 4.3: $G = (V, E \backslash E^+_{p_2})$

In $E \backslash E^+_{p_2}$ kann kein perfektes Matching gefunden werden und deshalb ist $E^+_{p_2}$ ein Blocker. Setzte $b = 2$.

Step 4 Wenn $b - a = 1$, dann gebe die veränderten Kosten c^b und den Zielfunktionswert $f_{bH}(c^b) = w(e_{i_b})$ aus, stop.
Dieses Problem hat als Lösung den Zielfunktionswert $f_{bH}(c^*) = w(e_2) = 2$ und die neue Kostenmatrix C^* hat folgende Gestalt:

$$C^* = \begin{pmatrix} 1 & 3 & \textcircled{17} \\ 2 & 5 & 8 \\ 4 & \textcircled{16} & 9 \end{pmatrix}$$

Im nächsten Kapitel wird eine Zusammenfassung über das inverse Summen-Zuordnungsproblem gegeben und überprüft, ob und welche Unterschiede es bei den Berechnungen des Engpass-Zuordnungsproblems und des Summen-Zuordnungsproblems mit der Engpass-Hamming-Distanz bzw. mit der Summen-Hamming-Distanz gibt.

5 Inverses Summen-Zuordnungsproblem unter der Hamming-Distanz

In diesem Kapitel wird ein Überblick über das inverse Summen-Zuordnungsproblem gegeben. Zuerst wird das Summen-Zuordnungsproblem kurz beschrieben. Später wird das inverse Summen-Zuordnungsproblem mit der Engpass-Hamming-Distanz definiert und Lösungsmethoden für diese Problemstellung präsentiert. Zuletzt wird das inverse Summen-Zuordnungsproblem mit der Summen-Hamming-Distanz betrachtet.

5.1 Das Summen-Zuordnungsproblem

Ein lineares Summen-Zuordnungsproblem kann folgendermaßen definiert werden:

Gegeben ist ein bipartiter Graph $G = (U, V; E)$ mit $|U| = |V| = n$ und Kantenkosten c_{ij}, $(ij) \in E$.
Gesucht werden Variablen x_{ij}, die die folgenden Bedingungen erfüllen:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n \\ & x_{ij} \in \{0,1\} \end{aligned}$$

Lösungsstrategien
Ein Summen-Zuordnungsproblem kann auf mehrere verschiedene Arten gelöst werden. Eine Möglichkeit ist es das Problem direkt als primales Problem zu lösen. Eine andere Möglichkeit ist, das duale Problem zu lösen und schließlich kann man das Summen-Zuordnungsproblem auch als eine Kombination der beiden Methoden (primal-dual Verfahren) lösen [6].

Eine der wichtigsten Methoden um ein Summen-Zuordnungsproblem zu lösen ist die Ungarische Methode. Diese Technik kann das Problem in $\mathcal{O}(n^3)$ Zeit lösen [6].
Weitere Prozeduren um dieses Problem zu lösen sind: Der sukzessive kürzeste Wege Algorithmus (Successive Shortest Path Algorithm), der Relaxations Algorithmus und der Kosten-Skalierungs-Algorithmus.

Es gilt, dass der sukzessive kürzeste Wege Algorithmus und der Relaxations-Algorithmus das Summen-Zuordnungsproblem in $\mathcal{O}(n\ S(n,m,C))$ Zeit lösen. Dabei ist $S(n,m,C)$ die Zeit, die benötigt wird, um ein kürzeste Wege Problem im Residualnetzwerk zu lösen. Eine einfache Implementierung des Kosten-Skalierungs-Algorithmus löst das Summen-Zuordnungsproblem in $\mathcal{O}(nm\ \log(nC))$ Zeit und durch eine Abwandlung des Algorithmus kann diese Laufzeit zu $\mathcal{O}(\sqrt{n}m\ \log(nC))$ verbessert werde. Dabei ist $\mathcal{O}(\log(nC))$ die Anzahl der Skalierungphasen [1].

5.2 Inverses Summen-Zuordnugsproblem unter der gewichteten Engpass-Hamming-Distanz

Dieses Problem kann folgendermaßen definiert werden:
Gegeben ist ein Graph $G = (V, E)$, Kantenkosten $c(e)$, $e \in E$ und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\min \quad f_{sbH}(c^*) := \max_{e \in E} w(e)\ h(c^*(e), c(e))$$

$$\text{s.t.} \quad \sum_{e \in F^*} c^*(e) \leq \sum_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F}$$

In Kapitel 4.1 wurde bereits ein Algorithmus präsentiert, der auch auf diese Aufgabenstellung angewendet werden kann [15].

Algorithmus für das inverse Summen-Zuordnungsproblem unter der Engpass-Hamming-Distanz:

Step 1 Nummeriere die Elemente von E sodass $w(e_1) \leq w(e_2) \leq ... \leq w(e_m)$ gilt
$k := 0$, $w(e_0) := 0$, $c^* = c$,
Löse ein Summen-Zuordnungsproblem mit Kosten c^* und berechne
$\delta^* = \min\{\sum_{e \in F} c^*(e) \mid F \text{ ist zulässig}\}$

Step 2 While $\sum_{e \in F^*} c^*(e) > \delta^*$ do
$k := k + 1$
Wenn e_k mit dem Gewicht $w(e_k)$ eine Kante in F^* ist, dann setze $c^*(e_k) := -M$, sonst $c^*(e_k) := M$
Löse das Summen-Zuordnungsproblem mit den neuen Kosten c^* und aktualisiere δ^*

Step 3 Stoppe mit dem optimalen Wert $w^* := w(e_k)$

Der Algorithmus besitzt eine Laufzeit von $\mathcal{O}(m\, A(m,n))$. $A(m,n)$ ist dabei die Zeit, die benötigt wird um ein Summen-Zuordnungsproblem zu lösen. Ein Summen-Zuordnungsproblem kann in $\mathcal{O}(n^3)$ Zeit gelöst werden [6]. Durch eine Binärsuche kann der Algorithmus noch beschleunigt werden und wir erhalten eine Laufzeit von $\mathcal{O}((A(m,n) \log\ m)$. Als Gesamtlaufzeit ergibt sich somit $\mathcal{O}(n^3 \log\ m)$.

In [15] wird noch ein weiterer Algorithmus für das inverse Summen-Zuordnungsproblem mit der Engpass-Hamming-Distanz präsentiert. Die Idee dieses Algorithmus ist, dass nicht in jeder Iteration δ^* berechnet wird, sondern dass durch andere Mittel überprüft wird, ob F^* mit den geänderten Kosten optimal ist.

Algorithmus für das inverse Summen-Zuordnungsproblem unter der Engpass-Hamming-Distanz ohne wiederholte Berechnung von δ^*:

Step 1 Nummeriere die Elemente von E sodass $w(e_1) \leq w(e_2) \leq ... \leq w(e_m)$ gilt
$k := -1$, $w(e_0) := 0$, $c^* = c$,
Löse ein Summen-Zuordnungsproblem mit Kosten c^* und berechne
$\delta^* = \min\{\sum_{e \in F} c^*(e) \mid F \text{ ist zulässig}\}$

Step 2 Repeat
$k := k + 1$,

Wenn $k > 0$ ist,
wenn e_k eine Kante in F^* ist, dann setze $c^*(e_k) := 0$, sonst $c^*(e_k) := M$;
Wende eine besondere Prozedur P an um zu überprüfen, ob F^* mit den geänderten Kosten c^* eine optimale Lösung ist
until F^* ist optimal

Step 3 Stoppe mit dem optimalen Wert $w^* := w(e_k)$

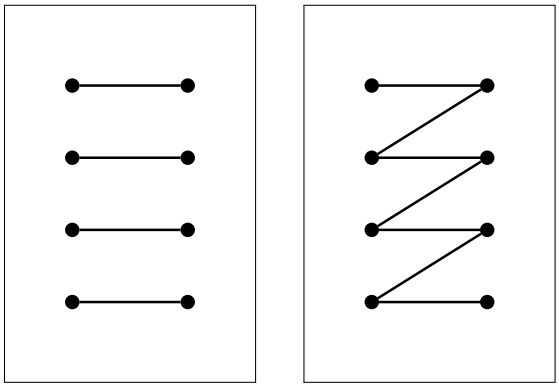
Die Idee ist es nun das lineare Summen-Zuordnungsproblem zu einem Minimalen Kosten Fluß Problem umzuschreiben.
Minimales Kosten Fluß Problem:
Sei $G = (V, E)$ ein gerichtetes Netzwerk mit Kosten c_{ij} und Kapazitäten u_{ij} für alle Kanten $(ij) \in E$. Für jeden Knoten wird ein Bedarf $(b(i) < 0)$ oder ein Vorrat $(b(i) > 0)$ angegeben.

$$\begin{aligned} \min \quad & \sum_{(ij) \in E} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j:(ij) \in E} x_{ij} - \sum_{j:(ji) \in E} x_{ji} = b(i), \quad \forall i \in V \\ & 0 \leq x_{ij} \leq u_{ij} \end{aligned}$$

Um nun ein lineares Summen-Zuordnungsproblem zu erhalten sind folgende Änderungen notwendig: In einem gerichteten bipartiten Graphen mit $2n$ Knoten und Kanten von i nach

j werden die Bedarfs- und Vorratsknoten mit $b(i) = 1$ und $b(j) = -1$ definiert. Eine zulässige Lösung für das Summen-Zuordnungsproblem hat n Elemente, diese Lösung kann zu einer zulässigen Lösung für das Minimale Kosten Fluß Problem erweitert werden. Diese Lösung ist dann ein spannender Baum im bipartiten Graphen mit $2n-1$ Kanten. Dabei ist ein spannender Baum ein Baum, der alle Knoten im Graphen enthält.

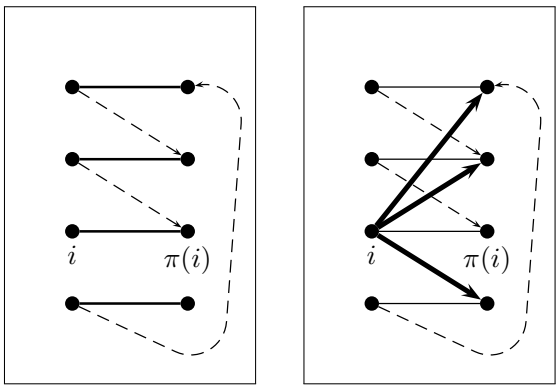
Beispiel



(a) zulässige Lösung für das Summen-Zuordnungsproblem (b) zulässige Zuordnung für das Minimale Kosten Fluß Problem

Abbildung 5.1: Zulässige Lösung des Summen-Zuordnungsproblems zu einer zulässigen Lösung des Minimalen Kosten Fluß Problems erweitert

In [15] wird dann die Prozedur P präsentiert, die überprüft, ob F^* mit den geänderten Kosten c^* eine optimale Lösung ist. Zuerst werden die reduzierten Kosten mit Hilfe der Netzwerk-Simplex Methode berechnet. Bei der Netzwerk-Simplex Methode wird mit der Startlösung $F_B \supset F^*$ begonnen. F_B ist dabei die Basislösung eines spannenden Baumes (Basisvariablen).
Die Optimalität von F_B kann dann überprüft werden, indem man die reduzierten Kosten für alle Kanten aus $E \backslash F_B$ (Nicht-Basisvariablen) berechnet. Eine Lösung F^* ist dann optimal, wenn alle diese Werte nicht-negativ sind.
Um die reduzierten Kosten effizient zu berechnen, wird von der Netzwerk-Simplex Methode auch Knotenpotentiale p_v für alle Knoten $v \in V$ berechnet. Für eine Kante (v, w) in der Basislösung gilt dann: $c^*(v, w) = p_v - p_w$.
Leider treten hin und wieder Schwierigkeiten auf: Wenn alle reduzierten Kosten der Nicht-Basisvariablen nicht-negativ sind, dann ist F^* optimal. Jedoch hat eine Kante (v, w) negative reduzierte Kosten, dann kann daraus nicht mehr gefolgert werden, dass F^* nicht optimal ist. Bei einem linearen Zuordnungsproblem kann dies verhindert werden, indem sich die Basislösungen $F_B \supset F^*$ ständig verändern. In einem bipartiten Graphen kann die Lösung eines linearen Zuordnungsproblems durch eine Permutation π^0 dargestellt werden. F^* wird dann durch die Kanten $(i, \pi^0(i)), i = 1, .., n$, repräsentiert. Für einen Knoten i wird dann eine Basislösung eines spannenden Baumes F_{B_i} folgendermaßen definiert: für jeden Knoten $i' \neq i$ hat F_{B_i} $n-1$ zusätzliche Kanten $(i', \pi^0(i'+1))$ und $\pi^0(n+1) \equiv \pi^0(1)$. Dann werden die reduzierten Kosten für alle von i ausgehenden Kanten (i, j) berechnet und auch alle Knotenpotentiale für den Baum F_{B_i} ermittelt.



(a) konstruierter spannender Baum F_{B_i}

(b) jede von i ausgehende Kante erzeugt im Baum F_{B_i} einen alternierenden Kreis

Abbildung 5.2: Zulässige Lösung des Summen-Zuordnungsproblems zu einer zulässigen Lösung des Minimalen Kosten Fluß Problems erweitert

Der Baum F_{B_i} hat die Eigenschaft, dass jede Kante (i, j), die von i ausgeht, einen alternierenden Kreis erzeugt. Bei einem alternierenden Kreis wechseln sich Kanten aus F^* und freie Kanten ab. Folgt man so einem Kreis, kann eine weitere Basislösung F' mit $\pi(i) = j$ gefunden werden. Diese Lösung F' hat dann die Kosten $\sum_{e \in F'} c^*(e) = \sum_{e \in F^*} c^*(e) + c^*(i, j) - p_i + p_j$. In einer Zeit von $\mathcal{O}(n)$ können also die reduzierten Kosten für alle Kanten ausgehend von i berechnet werden, inklusive die Konstruktion des Graphen F_{B_i} und die Bestimmung der Knotenpotentiale. Wird diese Prozedur für alle Knoten $v \in V$ durchgeführt, dann kann in $\mathcal{O}(n^2)$ Zeit überprüft werden, ob F^* eine optimale Lösung ist. Daraus folgt, dass das inverse Summen-Zuordnungsproblem mit der gewichteten Hamming-Distanz in einer Zeit von $\mathcal{O}(n^2 \log n)$ gelöst werden kann.

Eine weitere Möglichkeit um eine Lösung für das inverse Summen-Zuordnungsproblem zu finden ist direkt ein inverses Minimum Kosten Fluß Problem zu lösen [26].

Das inverse Minimum Kosten Fluß Problem unter einer gewichteten Engpass-Hamming-Distanz kann wie folgt definiert werden:
Gegeben ist ein zusammenhängendes, gerichtetes Netzwerk $N(U, V, E, u, c)$ mit der Knotenmenge $|U| = |V| = n$ und der Kantenmenge E. Es sind Kosten c_{ij} für alle $(i, j) \in E$, Kapazitäten u_{ij} für alle $(i, j) \in E$, Gewichte w_{ij} für alle $(i, j) \in E$ und obere und untere Schranken l_{ij}, q_{ij} für alle $(i, j) \in E$ gegeben. Weiters ist ein zulässiger Fluß x^0 im Netzwerk gegeben, der optimal werden soll. Dann wird nach einem Kostenvektor c^* gesucht, für den gilt:

$$\begin{aligned} \min \quad & \sum_{(ij)\in E} w_{ij}\ h(c_{ij}, c^*_{ij}) \\ \text{s.t.} \quad & \text{Fluß } x^0 \text{ ist ein minimaler Kosten-Fluß in } N(U,V,E,u,c^*) \\ & -l_{ij} \le c^*_{ij} - c_{ij} \le q_{ij} \end{aligned}$$

An dieser Stelle ist es notwendig einige wichtige Tatsachen über das Minimale Kosten Fluß Problem zu präsentieren:

Für das gegebene Netzwerk $N(U,V,E,u,c)$ und den Fluß x^0 kann das Residual-Netzwerk $N = (U,V,E',u',c')$ mit Hilfe des folgenden Algorithmus gebildet werden:
Algorithmus Ahuja, Magnanti und Orlin, [1]

Step 1 Wenn $(i,j) \in E$ und $x^0_{ij} < u_{ij}$, dann ist $(i,j) \in E'$, setze $u'_{ij} = u_{ij} - x^0_{ij}$ und $c'_{ij} = c_{ij}$.

Step 2 Wenn $(i,j) \in E$ und $x^0_{ij} > 0$, dann ist $(j,i) \in E'$, setze $u'_{ji} = x^0_{ij}$ und $c'_{ji} = -c_{ij}$.

Bemerkung: Mit $E(c)_1$ und $E(c)_2$ werden die Kantenmengen definiert, die in Schritt 1 und Schritt 2 gebildet werden.

Theorem 10 *Der Fluß x^0 ist ein minimaler Kosten-Fluß im Netzwerk N genau dann wenn das zugehörige Residual-Netzwerk $N'(U,V,E',u',c')$ keinen Kreis mit negativen Kosten enthält.*

Theorem 11 *Wenn das vorhin beschriebene Problem eine zulässige Lösung besitzt, dann existiert eine optimale Lösung c^*, für die gilt:*

1. $c^*_{ij} \ge c_{ij}$ *für* $(i,j) \in E(c)_1$
2. $c^*_{ij} \le c_{ij}$ *für* $(j,i) \in E(c)_2$

Theorem 12 *Wenn das Problem eine zulässige Lösung besitzt, dann existiert eine optimale Lösung c^*, für die gilt:*

1. *Wenn* $(i,j) \in E(c)_1$ *und* $c^*_{ij} \ne c_{ij}$ *dann setze* $c^*_{ij} = c_{ij} + q_{ij}$
2. *Wenn* $(j,i) \in E(c)_2$ *und* $c^*_{ij} \ne c_{ij}$ *dann setze* $c^*_{ij} = c_{ij} - l_{ij}$

Nun wird ein Algorithmus vorgestellt, der das inverse Minimum Kosten Fluß Problem löst:

Algorithmus

Step 0 Konstruiere ein Residual-Netzwerk $N = (U, V, E', u', c')$ für den Fluss x^0. Sei $W = \Omega = \emptyset$

Step 1 Wähle einen Kreis C mit negativen Kosten im aktuellen Residual-Netzwerk. Wenn kein Kreis mit negativen Kosten existiert, dann gehe zu Step 4, ansonsten gehe zu Step 2.

Step 2 Wenn $C \backslash \Omega = \emptyset$, dann gehe zu Step 5. Ansonsten gehe zu Step 3.

Step 3 Finde eine Kante $(x, y) \in C \backslash \Omega$, für die gilt:
$w_{xy} = \min\{w_{ij} \mid (ij) \in C \backslash \Omega\}$
aktualisiere das aktuelle Netzwerk und das zugehörige Residual-Netzwerk folgendermaßen:
Wenn $(x, y) \in E(c)_1$ ist, dann setze $c_{xy} = c_{xy} + q_{xy}$, $c'_{xy} = c_{xy}$, $W = W \cup \{w_{xy}\}$.
Wenn $(x, y) \in E(c)_2$ ist, dann setze $c_{yx} = c_{yx} - l_{yx}$, $c'_{yx} = c_{yx}$, $W = W \cup \{w_{yx}\}$.
$\Omega = \Omega \cup \{(x, y)\}$.
Gehe zurück zu Step 1.

Step 4 Stop. x^0 ist ein minimaler Kosten-Fluß im aktuellen Netzwerk, die optimale Lösung des inversen Problems ist der Kostenvektor des aktuellen Netzwerks und der Zielfunktionswert $\max\{w_{ij} \mid w_{ij} \in W\}$.

Step 5 Stop. Das inverse Problem besitzt keine zulässige Lösung.

Dieser Algorithmus löst das inverse Minimum Kosten Fluß Problem mit der Engpass-Hamming-Distanz in $\mathcal{O}(nm^2)$.

5.3 Inverses Summen-Zuordnungsproblem unter der gewichteten Summen-Hamming-Distanz

Gegeben ist ein Graph $G = (V, E)$, Kantenkosten $c(e)$, $e \in E$, und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\min \qquad f_{ssH}(c^*) := \sum_{e \in E} h(c^*(e), c(e))$$

$$\text{s.t.} \qquad \sum_{e \in F^*} c^*(e) \leq \sum_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F}$$

Wie im vorhergehenden Kapitel kann das inverse Summen-Zuordnungsproblem unter der gewichteten Summen-Hamming-Distanz zu einem inversen Minimum Kosten Fluss Problem unter der gewichteten Summen-Hamming-Distanz umgeschrieben werden.

In [26] wird gezeigt, dass das inverse Minimum Kosten Fluß Problem unter der gewichteten Summen-Hamming-Distanz APX-schwer ist. Dabei ist APX die Abkürzung für approximable und deutet an, dass das Optimierungsproblem, zumindest bis zu einem gewissen Grad, effektiv approximierbar ist. APX ist die Menge der $\mathcal{NP}$-Optimierungsprobleme, die sich für ein $\epsilon > 0$ in polynomieller Zeit ϵ-approximativ lösen lassen.

6 Weitere inverse Zuordnungsprobleme

In diesem Kapitel werden einige weitere inverse Zuordnungsprobleme genauer untersucht. Zuerst wird das inverse Summen-Zuordnungsproblem mit der l_1 Norm präsentiert und es werden einige Ergebnisse für diese Probleme aus den Arbeiten von Ahuja und Orlin [2] vorgestellt und auch ein Lösungsalgorithmus gezeigt. Dann wird noch kurz das inverse Engpass-Zuordnungsproblem mit der l_∞ Norm präsentiert.

6.1 Inverses Summen-Zuordnungsproblem mit der l_1 Norm

Ein inverses Summen-Zuordnungsproblem kann folgendermaßen definiert werden:
Gegeben ist ein Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$ und eine zulässige Lösung F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_S(c^*) := \sum_{e \in E} |c^*(e) - c(e)| \\ \text{s.t.} \quad & \sum_{e \in F^*} c^*(e) \leq \sum_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \end{aligned}$$

Ahuja und Orlin haben in ihrer Arbeit [2] Folgendes gezeigt:

- Wenn das Problem P ein lineares Optimierungsproblem ist, dann ist auch das dazu inverse Problem ein lineares Optimierungsproblem.
- Ist das Problem P ein lineares Zuordnungsproblem, dann kann das dazu inverse Problem unter der l_1 Norm mit Einheitsgewicht durch das Lösen eines linearen Zuordnungsproblems berechnet werden. Sind unterschiedliche Gewichte gegeben, dann kann das inverse Problem auf ein Minimales Kosten Fluß Problem reduziert werden.

Ahuja und Orlin geben folgende Methode an, um ein inverses Summen-Zuordnungsproblem zu lösen [3]:

Sei $G = (U, V, E)$ ein bipartiter gerichteter Graph mit der Knotenmenge $|U| = |V|$ und der Kantenmenge E. Bei einem Summen-Zuordnungsproblem wird nach einer Zuordnung M

gesucht, für die gilt, dass der Knoten $i \in U$ einem Knoten $j \in V$ zugeordnet wird, sodass $\sum_{(i,j)\in M} c_{ij}$ minimal ist. Sei $c(M) = \sum_{(i,j)\in M} c_{ij}$. Bei einem inversen Zuordnungsproblem soll eine gegebene Zuordnung M^* optimal werden. Dies geschieht durch eine Modifikation des Kostenvektors c zu einem neuem Kostenvektor c^*, sodass $|c^* - c|$ minimal wird.

Sei M^0 eine optimale Zuordnung in G und durch π werden die optimalen dualen Variablen gekennzeichnet. Durch $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$ werden die reduzierten Kosten einer Kante $(i,j) \in E$ definiert.

Auf Grund der Optimalitätsbedingungen eines Zuordnungsproblems gilt:

- $c_{ij}^{\pi} = 0$ für jede Kante $(i,j) \in M^0$
- $c_{ij}^{\pi} \geq 0$ für jede Kante $(i,j) \notin M^0$

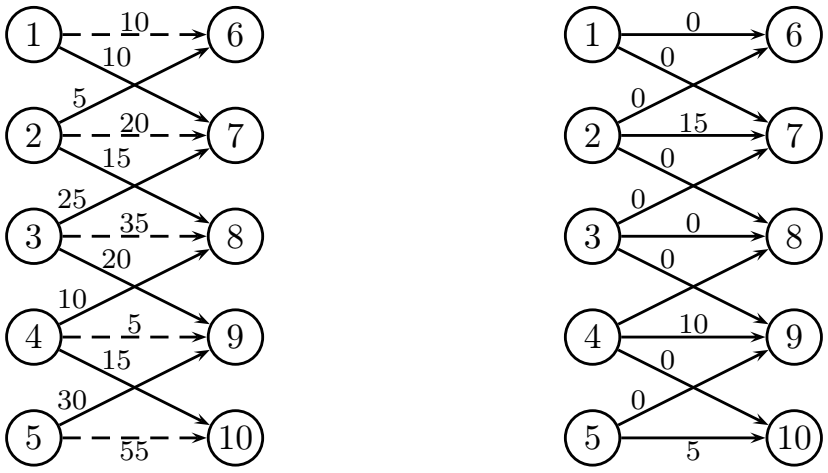
Dann kann der Kostenvektor c^* folgendermaßen definiert werden:

$$c_{ij}^* = \begin{cases} c_{ij} - c_{ij}^{\pi}, & \text{für alle } (i,j) \in M^* \\ c_{ij}, & \text{für alle } (i,j) \notin M^* \end{cases}$$

Dies bedeutet, dass die Kosten der Kanten in M^* um den Wert der reduzierten Kosten verringert werden. Alle anderen Kanten bleiben unverändert. Diese Veränderung reduziert die Kosten der Zuordnung M^* um einen Wert in der Höhe von $\sum_{(i,j)\in M^*} c_{ij}^{\pi}$. Die Kosten der Zuordnung M^0 bleiben unverändert, da die reduzierten Kosten jeder Kante in M^0 null sind. Nach dieser Modifikation werden die veränderten reduzierten Kosten jeder Kante $(i,j) \in M^*$ null und M^* wird zu einer weiteren optimalen Zuordnung in G.
Ahuja und Orlin [3] haben gezeigt, dass diese Prozedur eine Optimallösung für das inverse Summen-Zuordnungsproblem mit der l_1 Norm liefert.

Beispiel *Gegeben ist ein bipartiter Graph, Kantenkosten c_{ij}, $(i,j) \in E$ und eine zulässige Lösung*
$M^* = \{(1,6),(2,7),(3,8),(4,9),(5,10)\}$. *In Abbildung 6.1 wird der Graph und die zulässige Lösung M^* dargestellt.*

Die optimale Zuordnung in diesem Graphen ist $M^0 = \{(1,7),(2,6),(3,8),(4,10),(5,9)\}$.
Die Knotenpotentiale π und die reduzierten Kosten c_{ij}^{π} können mit Hilfe folgender Gleichungen berechnet werden:



(a) Graph mit Kosten c_{ij}, die gekennzeichnete Zu- (b) Graph mit reduzierten Kosten $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$
ordnung soll optimal werden

Abbildung 6.1: Beispiel für das inverse Zuordnungsproblem

$$
\begin{aligned}
10 - \pi_1 + \pi_7 &= 0 \\
5 - \pi_2 + \pi_6 &= 0 \\
35 - \pi_3 + \pi_8 &= 0 \\
15 - \pi_4 + \pi_{10} &= 0 \\
30 - \pi_5 + \pi_9 &= 0
\end{aligned}
$$

$$
\begin{aligned}
10 - \pi_1 + \pi_6 &\geq 0 \\
20 - \pi_2 + \pi_7 &\geq 0 \\
15 - \pi_2 + \pi_8 &\geq 0 \\
25 - \pi_3 + \pi_7 &\geq 0 \\
20 - \pi_3 + \pi_9 &\geq 0 \\
10 - \pi_4 + \pi_8 &\geq 0 \\
5 - \pi_4 + \pi_9 &\geq 0 \\
55 - \pi_5 + \pi_{10} &\geq 0
\end{aligned}
$$

Daraus ergeben sich folgende Werte für die dualen Variablen: $\pi_1 = 0$, $\pi_2 = -5$, $\pi_3 = 15$, $\pi_4 = -10$, $\pi_5 = 25$, $\pi_6 = -10$, $\pi_7 = -10$, $\pi_8 = -20$, $\pi_9 = -5$, $\pi_{10} = -25$.

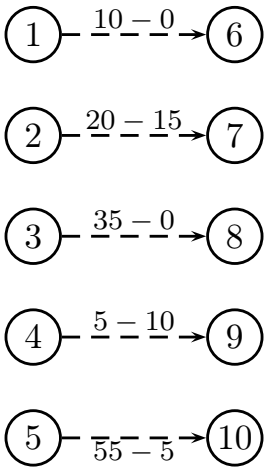


Abbildung 6.2: Graph mit den optimalen Kosten c^*

Man erhält den optimalen Kostenvektor c^* durch eine Verringerung der Kosten auf den Kanten, die in M^* liegen, siehe Abbildung 6.2. Es gilt dann, dass $c^*_{ij} = c_{ij} - c^{\pi}_{ij}$ für alle $(i,j) \in M^*$ ist. $c^*_{ij} \in M^*$ besitzt dann folgende Werte: $c^*_{1,6} = 10$, $c^*_{2,7} = 5$, $c^*_{3,8} = 35$, $c^*_{4,9} = -5$, $c^*_{5,10} = 50$.

Für die Kosten ergibt sind dann $\sum_{(i,j)\in M^*} c^*_{ij} = \sum_{(i,j)\in M^0} c_{ij} = 95$.

Im Fall eines inversen Summen-Zuordnungsproblems mit einer gewichteten l_1 Norm wird das Zuordnungsproblem zu einem Minimalen Kosten Fluß Problem umformuliert und kann durch einen Minimum Kosten Fluß Algorithmus gelöst werden [3].

6.2 Inverses Engpass-Zuordnungsproblem unter der l_∞ Norm

In diesem Kapitel soll nun das inverse Engpass-Zuordnungsproblem mit der l_∞ Norm untersucht werden.
Bei diesem Problem soll die größte Abweichung von den ursprünglichen Kosten zu den neuen Kosten, dh. $\max |c^*(e) - c(e)|$, minimal werden.
Gegeben ist ein bipartiter Graph $G = (V, E)$, eine Kostenfunktion $c(e)$ für alle $e \in E$, eine Gewichtsfunktion $w(e)$ für alle $e \in E$, obere und untere Schranken $l(e) \geq 0$, $u(e) \geq 0$ für alle $e \in E$, die angeben um wie viel die Kosten verändert werden dürfen, und eine zulässige Lösung

F^*. Die Aufgabe ist nun, eine Kostenfunktion c^* zu finden, die das folgende Problem löst:

$$\begin{aligned} \min \quad & f_{l_\infty}(c^*) := \max_{e \in E} |c^*(e) - c(e)| \\ \text{s.t.} \quad & \max_{e \in F^*} c^*(e) \leq \max_{e \in F} c^*(e), \quad \forall\ F \in \mathcal{F} \\ & c(e) - l(e) \leq c^*(e) \leq c(e) + u(e), \quad \forall\ e \in E \end{aligned}$$

Guan und Zhang präsentieren in ihrer Arbeit [19] einen Algorithmus für das inverse Engpass-Zuordnungsproblem mit der l_∞ Norm. Dieser Algorithmus besitzt eine Laufzeit von $\mathcal{O}(m^2 n^3 \log m)$.

7 Zusammenhang zwischen Summen-Zuordnungsproblemen und Engpass-Zuordnungsproblemen

In diesem Abschnitt wird untersucht, ob es einen Zusammenhang zwischen Summen-Zuordnungsproblemen und Engpass-Zuordnungsproblemen bzw. zwischen inversen Summen-Zuordnungsproblemen und inversen Engpass-Zuordnungsproblemen gibt.
Es gilt, dass das Engpass-Zuordnungsproblem durch das Summen-Zuordnungsproblem approximiert werden kann.

Beispiel *Gegeben ist ein bipartiter Graph und eine Kostenmatrix C*

$$C = \begin{pmatrix} 1 & 8 & 7 \\ 3 & 5 & 8 \\ 7 & 8 & 9 \end{pmatrix}$$

Dann erhält man als Lösung für das Summen-Zuordnungsproblem die Zuordnung $\{c_{11}, c_{22}, c_{33}\}$ *und für das Engpass-Zuordnungsproblem* $\{c_{13}, c_{22}, c_{31}\}$.

Wird nun von den Elementen der Kostenmatrix C die 10-te Potenz betrachtet, dann hat die Matrix $\tilde{C}$ *folgende Gestalt:*

$$\tilde{C} = \begin{pmatrix} 1^{10} & 8^{10} & 7^{10} \\ 3^{10} & 5^{10} & 8^{10} \\ 7^{10} & 8^{10} & 9^{10} \end{pmatrix} = \begin{pmatrix} 1 & 1073741824 & 282475249 \\ 59049 & 9765625 & 1073741824 \\ 282475249 & 1073741824 & 3486784401 \end{pmatrix}$$

Es stellt sich heraus, dass in dieser Matrix das Summen-Zuordnungsproblem und das Engpass-Zuordnungsproblem dieselbe Lösung $\{c_{13}, c_{22}, c_{31}\}$ *besitzen.*

Das bedeutet, dass Summen-Zuordnungsprobleme durch Engpass-Probleme approximiert werden können. Jedoch gilt diese Behauptung für inverse Probleme nicht. Die Lösung eines inversen Summen-Zuordnungsproblems liefert nicht die Lösung für ein inverses Engpass-Zuordnungsproblem. Dies wird mit Hilfe eines Beispiels gezeigt.

Beispiel *Gegeben ist wieder folgende Kostenmatrix*

$$\tilde{C} = \begin{pmatrix} 1 & 8^{10} & 7^{10} \\ 3^{10} & 5^{10} & 8^{10} \\ 7^{10} & 8^{10} & 9^{10} \end{pmatrix}$$

Weiters ist eine zulässige Lösung $F^* = \{c_{12}, c_{23}, c_{31}\}$ gegeben, die optimal werden soll. Die Lösung $\{c_{31}, c_{22}, c_{13}\}$ ist eine Optimallösung sowohl für das Summen-Zuordnungsproblem als auch für das Engpass-Zuordnungsproblem.

Lösung eines inversen Summen-Zuordnungsproblems unter der l_1-Norm:

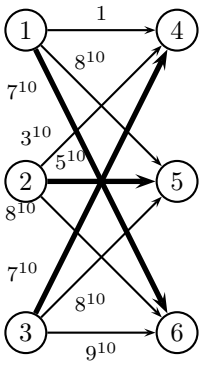


Abbildung 7.1: Graph mit den potenzierten Kosten

Als erstes werden die Knotenpotentiale mit Hilfe folgender Gleichungen berechnet:

$$\begin{aligned} 7^{10} - \pi_1 + \pi_6 &= 0 \\ 5^{10} - \pi_2 + \pi_5 &= 0 \\ 7^{10} - \pi_3 + \pi_6 &= 0 \end{aligned}$$

$$\begin{aligned} 1 - \pi_1 + \pi_4 &\geq 0 \\ 8^{10} - \pi_1 + \pi_5 &\geq 0 \\ 3^{10} - \pi_2 + \pi_4 &\geq 0 \\ 8^{10} - \pi_2 + \pi_6 &\geq 0 \\ 8^{10} - \pi_3 + \pi_5 &\geq 0 \\ 9^{10} - \pi_3 + \pi_6 &\geq 0 \end{aligned}$$

Daraus ergibt sich, dass die dualen Variablen $\pi_1 = 0,\ \pi_2 = 5^{10} - 8^{10},\ \pi_3 = 7^{10} - 1,$ $\pi_4 = -1,\ \pi_5 = -8^{10},\ \pi_6 = -7^{10}$ *sind.*
Die veränderte Kostenmatrix hat dann folgende Gestalt:

$$C* = \begin{pmatrix} 1 & 8^{10} & 7^{10} \\ 3^{10} & 5^{10} & 5^{10} - 8^{10} + 7^{10} \\ 7^{10} & 8^{10} & 9^{10} \end{pmatrix}$$

und als Zielfunktionswert erhält man $\sum_{e \in E} |c^*(e) - c(e)| = |5^{10} - 8^{10} + 7^{10} - 8^{10}|$.

Lösung eines inversen Engpass-Zuordnungsproblems unter der l_1***-Norm****:*

Suche nach den möglichen Werten für die Engpass-Funktion: $p \in \{7^{10}, 8^{10}\}$

$p = 8^{10}$

Kapazitätsmatrix:

$$W_p = \begin{pmatrix} 8^{10} - 1 & 0 & 8^{10} - 7^{10} \\ 8^{10} - 3^{10} & 8^{10} - 5^{10} & 0 \\ 8^{10} - 7^{10} & 0 & 0 \end{pmatrix}$$

In der Kapazitätsmatrix existiert ein minimaler Blocker und die veränderte Kostenmatrix hat folgende Gestalt:

$$C^* = \begin{pmatrix} 1 & 8^{10} & 8^{10} \\ 3^{10} & 5^{10} & 8^{10} \\ 7^{10} & 8^{10} & 9^{10} \end{pmatrix}$$

Es werden die Kosten der Kante c_{13} *geändert und der Zielfunktionswert lautet* $|8^{10} - 7^{10}|$.

$p = 7^{10}$

Kapazitätsmatrix:

$$W_p = \begin{pmatrix} 7^{10}-1 & 0 & 0 \\ 7^{10}-3^{10} & 7^{10}-5^{10} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In dieser Matrix existiert eine 1×3 *Nullmatrix und die veränderte Kostenmatrix hat daher folgende Gestalt:*

$$C^* = \begin{pmatrix} 1 & 7^{10} & 7^{10} \\ 3^{10} & 5^{10} & 7^{10} \\ 7^{10} & 8^{10} & 9^{10} \end{pmatrix}$$

Die Kosten der Kanten c_{12} *und* c_{23} *müssen geändert werden und der Zielfunktionswert lautet* $|8^{10}-7^{10}|+|8^{10}-7^{10}|$.

Für $p = 8^{10}$ *erhält man also den kleinsten Zielfunktionswert mit* $|8^{10}-7^{10}|$.

Das bedeutet, dass bei einem inversen Summen-Zuordnungsproblem nur die Kante c_{23} *geändert wird und man erhält als Zielfunktionswert* $|5^{10}-8^{10}+7^{10}-8^{10}|$. *Bei einem inversen Engpass-Zuordnungsproblem wird die Kante* c_{13} *geändert und der Zielfunktionswert lautet* $|8^{10}-7^{10}|$. *Aus diesem Grund kann ein inverses Summen-Zuordnungsproblem nicht durch ein inverses Engpass-Zuordnungsproblem approximiert werden.*

Literaturverzeichnis

[1] R.K. Ahuja, Th.L. Magnanti, and J.B. Orlin. *Network flows. Theory, algorithms, and applications*. Prentice Hall Inc., NJ, 1993.

[2] R.K. Ahuja and J.B. Orlin. Inverse optimization. *Operations Research*, 49:771–783, 2001.

[3] R.K. Ahuja and J.B. Orlin. Combinatorial algorithms for inverse network flow problems. *Networks*, 40:181–187, 2002.

[4] B. Alizadeh and R.E. Burkard. Inverse center location problems. *Electronic Notes in Discrete Mathematics*, 36:105–110, 2010.

[5] F. Baroughi, R.E. Burkard, and E. Gassner. Inverse p-median problems with variable edge lengths. *Mathematical Methods of Operations Research*, 73:263–280, 2011.

[6] R.E. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. SIAM, Philadelphia, Pa., 2009.

[7] R.E. Burkard, C. Pleschiutschnig, and J. Zhang. Inverse p-median problems. *Discrete Optimization*, 1:23–39, 2004.

[8] R.E. Burkard, C. Pleschiutschnig, and J. Zhang. The inverse 1-median problem on a cycle. *Discrete Optimization*, 5:242–253, 2007.

[9] D. Burton and Ph. L. Toint. On an instance of the inverse shortest path problem. *Mathematical Programming*, 53:45–61, 1992.

[10] M.C. Cai, X.G. Yang, and J. Zhang. The complexity of the inverse center location problem. *J. Global Opt.*, 15:213–218, 1999.

[11] E. Ciurea and A. Deaconu. Inverse minimum flow problem. *Journal of Applied Mathematics and Computing*, 23:193–203, 2007.

[12] S. Dempe and H. Schreier. *Operations Research, Deterministische Modelle und Methoden*. Teubner, 2006.

[13] U. Derigs and U. Zimmermann. An augmenting path methode for solving linear bottleneck assignment problems. *Computing*, 19:285–295, 1978.

[14] R.B. Dial. Minimal-revenue congestion pricing part i: A fast algorithm for the single-origin case. *Transportation Res. Part B*, 33:189–202, 1999.

[15] C.W. Duin and A. Volgenant. Some inverse optimization problems under the Hamming distance. *Operations Research*, 170:887–899, 2006.

[16] H.N. Gabow and R.E. Tarjan. Algorithms for two bottleneck assignment problems. *Journal of Algorithms*, 9:411–417, 1988.

[17] M.R. Garey and D.S. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W. H. Freeman & Co, 1979.

[18] X. Guan and J. Zhang. Inverse bottleneck optimization problems on networks. *Lecture Notes in Computer Science*, 4041:220–230, 2006.

[19] X. Guan and J. Zhang. Inverse constrained bottleneck problems under weighted l_∞ norm. *Computers & Operations Research*, 34:3243–3254, 2007.

[20] C. Heuberger. Inverse optimization, a survey on problems, methods, and results. *J. Com.Opt*, 8:329–361, 2004.

[21] D. Hochbaum. Efficient algorithms for the inverse spanning tree problems. *Operations Research*, 51:785–797, 2003.

[22] S. O. Krumke and H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. Vieweg und Teubner, 2009.

[23] A.P. Punnen and K.P.K. Nair. Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Applied Mathematics*, 55:91–93, 1994.

[24] P.T. Sokkalingam, R. Ahuja, and J.B. Orlin. Solving inverse spanning tree problems through network flow techniques. *Operations Research*, 47:291–298, 1999.

[25] A. Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*. Elsevier, 1987.

[26] J. Yiwei, L. Liu, B. Wu, and E. Yao. Inverse minimum cost flow problems under the weighted Hamming distance. *J. of Operational Research*, 207:50–54, 2010.

[27] R. Zenklusen, B. Ries, C. Picouleau, D. de Werra, M. C. Costa, and C. Bentz. Blockers and transversals. *Discrete Mathematics*, 13:4306–4314, 2009.

[28] J. Zhang and M.C. Cai. Inverse problem of minimum cuts. *Mathematical Methods of Operations Research*, 47:51–58, 1998.

[29] J. Zhang, Z. Liu, and Z. Ma. On the inverse problem of minimum spanning tree with partition constraints. *Mathematical Methods of Operations Research*, 44:171–188, 1996.

[30] J. Zhang, C. Yang, and Z. Ma. Inverse maximum flow and minimum cut problems. *Optimization*, 40:147–170, 1997.

[31] W. Zimmermann and U. Stache. *Operations Research*. Oldenbourg Wissenschaftsverlag, München, 2001.