Vergleich und Synthese von Matching Algorithmen

Eva Hennefeld & Anaïs Siebers

12. März 2020

Zusammenfassung

Thema dieser Arbeit ist eine Erweiterung des in der Graphentheorie bekannten Heiratsproblem, das sich mit dem Matching eines bipartiten Graphen beschäftigt. Das ursprüngliche Heiratsproblem beschäftigt sich mit der Frage: "Wie lassen sich zwei gleichgroße Gruppen an Männern und Frauen glücklich verheiraten?". Die Personengruppen lassen sich dabei auf beliebige andere Gruppen übertragen.
Als Erweiterung werden weitere Parameter, die das Matching beeinflussen hinzugefügt. Zunächst werden bekannte Verfahren zur Lösung des Heiratsproblems unter Berücksichtigung eines einzelnen Parameters herangezogen, im Anschluss wird mit Kombinationsverfahren eine Lösung gesucht, die mehrere Parameter einbezieht.

Diese Arbeit enstand im Rahmen der Veranstaltung
Graphentheorie und -algorithmen im WS 19/20

Inhaltsverzeichnis

1 Definitionen

1.1 Symbole

$=$	Wertzuweisung, Gleichsetzung
$:=$	Identitäts- / Definitionszuweisung
$\widehat{=}$	Wert- / Identitätsgleichheit
$\|X\|$	Mächtigkeit von X
G	ein allgemeiner Graph
V	Menge aller Knoten eines Graphen
E	Menge aller Kanten eines Graphen
v_i	i-ter Knoten des Graphen
e_i	i-te Kante des Graphen
$\overline{v_i v_j}$	gerichtete Kante von v_i zu v_j
$\delta^+(v)$	Menge der ausgehenden Kanten von v
$\delta^-(v)$	Menge der eingehenden Kanten von v
W_G	Menge der Kantengewichte des gesamten Graphen
$w_{\overline{v_i v_j}}$	Gewicht der gerichteten Kante zwischen v_i und v_j
P	Menge der Priorisierungen des gesamten Graphen
p_i	geordnete Liste der Priorisierung von Knoten
$()$	Liste mit festgelegter Reihenfolge
G_B	ein bipartiter Graph
V_X	Untermenge X der Knotenmenge V
M_G	ein Matching in einem Graphen
m_i	das Paar i in M_G, entsprechend einer Kante $\overline{v_i v_j}$
M_{max}	ein maximales Matching in einem Graphen
M_{per}	ein perfektes Matching in einem Graphen
M_{sta}	ein stabiles Matching in einem Graphen
M_{ide}	ein ideales Matching in einem Graphen
M_{gut}	Optimum zwischen M_{sta} und M_{ide} unter der Voraussetzung M_{per}

$pos(liste, x)$	die Position / der Index von x in der Liste $liste$
$max(pos(v_i))$	der Index der höchsten Position in der Liste p_i von v_i
$max(pos(w_{\overline{v_i}})$	das maximale Gewicht der Kanten, die von v_i ausgehen
$min(pos(w_{\overline{v_i}})$	das minimale Gewicht der Kanten, die von v_i ausgehen
M^*sta	Stabilitätsmaß eines Matching
M^*ide	Idealitätsmaß eines Matching

1.2 Graph

Sei $G := (V, E)$ ein Graph mit $V := \{v_1, \ldots, v_n\}$, $E := \{e_1, \ldots, e_r\}$ und $V \cap E = \emptyset$. Außerdem gilt: $V \subseteq E \times E$ und $0 \leq r \leq \binom{n}{2}$.

Formal wird eine Kante e_x zwischen einem Knoten v_i und einem Knoten v_j als $\overline{v_i v_j}$ bezeichnet. Die Kante hat immer eine Richtung von v_i nach v_j, wobei als v_i Quelle und v_j Senke bezeichnet wird. Die Menge der eingehenden (bzw. ausgehenden) wird geschrieben als $\delta^-(v)$ (bzw. $\delta^+(v)$). Folglich hat v_i eine ausgehende Kante, notiert als $|\delta^+(v_i)| = 1$ und v_j eine eingehende Kante $|\delta^-(v_j)| = 1$. $\forall e \in E \quad \exists w \in W$, wobei $w_{\overline{v_i v_j}}$ das Gewicht der bestimmten Kante $\overline{v_i v_j}$ und W Menge aller Gewichte im Graphen darstellt.

Daraus folgend ist $G_B := (V, E)$ ein bipartiter Graph, mit $v_1, \ldots, v_{\frac{n}{2}} \in V_A$ und $v_{\frac{n}{2}+1}, \ldots, v_n \in V_B$ Knoten, wobei V_A und $V_B \subset V$ und $V_A \cap V_B = \emptyset$. Die Knotenmenge V ist somit in zwei gleich große Teilmengen unterteilt, die keine gemeinsamen Elemente haben.

Zudem hat der Graph eine Kantenmenge E für die gilt, dass jede Kante von V_A nach V_B gerichtet ist. $\forall e \in E \quad e := \overline{ab}$.

Außerdem sei definiert: $\forall v \in V \quad \exists p \in P$, wobei p_i die Priorisierungsliste des Knoten v_i ist, wenn $V \in G_B$. p_i sei ein geordnete Liste, die alle Knoten der anderen Knotenmenge enthält. Wenn beispielsweise $v_i \in V_A$, gilt $\forall b \in V_B \quad b \in p_i$ und $\forall a \in V_A \quad a \notin p_i$. Die Indizes der Knoten in der Priorisierungsliste folgen der Fibonacci-Reihe. Folglich ist der erste Knoten an Position 1, der zweite an Position 2, der fünfte an Position 8, etc. Alternativ könnte auch eine andere schnell steigende Reihe verwendet werden. P sei die Menge aller Priorisierungslisten.

1.3 Matching

Sei $M := \{m_1, \ldots, m_n\}$ ein Matching, bestehend aus einer Menge an Paaren $m_1, \ldots, m_k$, wobei $M \subseteq E$. Daraus folgt, dass jedes Paar m_i einer Kante $\overline{v_i v_j} \in E$ entspricht. Ziel ist eine Zuordnung über G der Form: $\forall v \quad |\delta^+ \cup \delta^-| = 1$. Dies bedeutet, dass jeder Knoten entweder eine eingehenden oder eine ausgehende Kante hat. Außerdem hat

ein Paar $m_x := \overline{v_i v_j}$ das Gewicht der Kante $w_{m_x} \widehat{=} w_{\overline{v_i v_j}}$.

Für ein Matching in einem bipartiten Graphen gelten folgende Einschränkungen:

- Im Gegensatz zu einem Knoten in einem klassischen Graphen, darf ein Knoten in einem bipartiten Graphen in maximal einem Paar erscheinen.
- Für alle Paare gilt, dass V_A ihre Quelle und V_B ihre Senke ist. Daher ist $v_i \in V_A$ und $v_j \in V_B$. Es gibt deshalb kein Paar innerhalb von V_A oder V_B.

Es gibt verschiedene Kategorien, denen ein Matching zugeordnet werden kann:

1. **Einfaches Matching** (M) Bei einem einfachen oder klassischen Matching werden Paare gebildet. Die Anzahl der gebildeten Paare $|M|$ muss dabei > 1 sein, kann jedoch alle Paare enthalten: $1 \leq |M| \leq \lfloor \frac{|V|}{2} \rfloor$. In einem bipartiten Graphen sei die maximal mögliche Anzahl an Paare $min(|V_A|, |V_B|)$.
2. **Maximales Matching** (M_{max}) Ein maximales Matching lässt sich dadurch beschreiben, dass es die maximale Anzahl an möglichen Paaren enthält.
3. **Perfektes Matching** (M_{per}) Bei einem perfekten Matching hat jeder Knoten in V_A einen Partner: $\forall v \in V_A$ gibt es ein $v \in V_B$. In einem bipartiten Graphen sei ein Matching perfekt, wenn jeder Knoten von V_A einen Partner in V_B hat: $|M| \widehat{=} |V_A|$ unter der Voraussetzung, dass $|V_A| \leq |V_B|$.
4. **Stabiles Matching** (M_{sta}) Für ein stabiles Matching sei ein Graph mit einer Priorisierung P vorausgesetzt: $\forall v \quad \exists p$. Das gefundene Matching ist stabil, wenn über alle Paare hinweg, die Senke eines Paares von ihrer Quelle hoch priorisiert wird. Die Summe der Abweichungen vom höchst priorisierten Knoten der Quelle im Matching ($\sum_{i=1}^{|B|} pos(p_i, t_j)$ mit $m_{b_i t_j}$) sei möglichst gering. Je niedriger $pos(p_i, v_j)$ bei einem Paar $mat_{\overline{v_i v_j}}$ ist, desto zufriedener ist die Quelle. Ziel eines stabilen Matching ist es, die maximale Zufriedenheit der Quellen zu erreichen.
5. **Ideales Matching** (M_{ide}) Bei einem idealen Matching wird eine Gewichtung W vorausgesetzt. Das heißt, jede Kante besitzt ein Gewicht w. Bei einem idealen Matching sei nun die Summe der Kantengewichte aller Paare möglichst gering, sodass die Kosten des Matching insgesamt minimal sind.

2 Problemstellung

Zentraler Gegenstand dieser Arbeit soll ein bisher nicht graphentheoretisch erfasstes Problemgebiet sein. Analog zu dem bekannten und gelösten "Heiratsproblem"geht es um zwei unterschiedliche Gruppen an Persona, zwischen denen ein Matching gefunden werden sollen, das **alle** oben genannten Matchingkategorien abdeckt.

2.1 Motivation

Als Verbildlichung des Matching soll folgende Situation dienen: In einer Versteigerung sollen n Tiere an n Bauernhöfe vermittelt werden. Jeder Bauernhof hat dabei Präferenzen, welches Tier er aufnehmen möchte. Ziel ist es bei der Zuordnung verschiedene Aspekte einzubeziehen. Jedes Tier soll das bestmögliche Zuhause finden. Dies kann nur ermöglicht werden, wenn ...

- ... jeder Bauernhof Präferenzen angibt, damit das Tier entsprechend verpflegt werden kann. Nähere Betrachtung des Aspektes in Abschnitt 2.2.1.
- ... ein Tier nicht zu lange transportiert werden muss, damit es möglichst wenig Stress ausgesetzt wird. Deshalb wird der Transportweg bei der Versteigerung berücksichtigt. Nähere Betrachtung des Aspektes in Abschnitt 2.2.2.

Das Beispiel erfasst nur einen abstrakten Anwendungsfall, der auf viele praktische Anwendungsfälle projiziert werden kann. Im Folgenden werden alternative Beispielszenarien aufgeführt:

- Die Verteilung von Schüler auf Schulen: Momentan wird bei der Verteilung der Grundschulplätze die Nähe zum Wohnort als Faktor herangezogen. Bei der Auswahl der weiterführenden Schule wird nach der Priorität der Schüler/Eltern oder nach dem Lernniveau verteilt. Eine Kombination der Verfahren lässt sich durch die Problemstellung abbilden.
- Ein weiterer Anwendungsfall könnte die Zuordnung von Carsharing-Partnern sein. Durch die Gewichte kann die Distanz des Wohnortes zum Abfahrtsort bzw. die Distanz vom Zielort zum Ankunftsort dargestellt werden. Anhand der Prioritäten werden die zu zahlenden Preise gegeneinander aufgewogen.
- Dasselbe Schema lässt sich auch auf Bahnfahrten oder Flüge übertragen. Anstelle der Abweichung vom Start- und Zielort kann die Dauer der Reise als Kantengewicht verwendet werden. Die Prioritäten können individuell gewählt werden: für den einen werden die möglichen Fahrten/Flüge nach dem Preis

absteigend, für den anderen wiederum nach der Uhrzeit oder der bisherigen/-durchschnittlichen/zu erwartenden Auslastung sortiert.

- Das Problem kann auch auf die Zuordnung von Mitarbeiter auf Projektteams innerhalb eines Unternehmens übertragen werden. Auf der einen Seite können die Mitarbeiter ihre Prioritäten, auf der anderen Seite können die Führungskräfte ihre Vorstellung der Projektbesetzung angeben. Mit den Kantengewichten können die Kosten für einen Einsatz des Mitarbeiters in dem entsprechenden Projekt (Einarbeitungsaufwand, eventuelle Reisekosten, etc.) festgehalten werden.

Allgemein kann das Problem auf jede Art von Ressourcenzuordnung übertragen werden, bei der zwei verschiedene Aspekte, wie z.B. Transportweg und Preis gleichstark berücksichtigt werden sollen. Der Fantasie sind keine Grenzen gesetzt. Für die Ausführungen des Papers soll weiterhin das Beispiel der Tiere und Bauernhöfe verwendet werden.

2.2 Informelle Problemdefinition

Eine 1 zu 1 Zuordnung eines Tieres zu einem Bauernhof ist die Basis für die Lösung des Problems. Allerdings sollen darüber hinaus weitere Aspekte betrachtet werden.

2.2.1 Priorisierung

Als erster Faktor werden Prioritäten einbezogen. Jeder Bauernhof muss seine Priorisierung der Tiere angeben, damit diese für das Matching-Verfahren verwendet werden können. Dadurch werden nicht mehr nur Paare gefunden, sondern der Fokus liegt auch auf der Zufriedenheit der Paare.

2.2.2 Gewichte der Kanten

Dieser Abschnitt befasst sich mit der Gewichtung von Kanten in einem bereits gematchten Graphen. Jede Zuteilung bzw. jeder Kauf eines Tieres ist mit Kosten für den Bauernhof verbunden. Das Gewicht der Kanten im Beispiel symbolisiert die Entfernung des Tieres vom Bauernhof. Damit die Transportwege und somit die Kosten für den Käufer und der Stress für das Tier möglichst gering sind, soll die Summe aller Gewichte des Matchings so klein wie möglich sein.

2.3 Formelle Problemdefinition

Im Folgenden befassen wir uns mit einem Graphen G_P. Der Graph hat folgende Eigenschaften:

Der Graph des Beispiels ist bipartit und wie folgt definiert: $G_P = (V_B \cup V_T, E)$, $V_B \cap V_T = \emptyset$ und $V_B = b_1, \ldots, b_n$ bzw. $V_T = t_1, \ldots, t_n$. Dabei bezeichnet V_B die Menge der Bauernhöfe und V_T die Menge der Tiere.

Zudem hat der Graph eine Kantenmenge E für die gilt, dass jede Kante von V_B nach V_T gerichtet ist: $\forall e \in E \quad e := \overline{bt}$. Außerdem hat jeder Knoten b mindestens eine ausgehende Kante, da der Knoten sonst keinem weiteren Knoten zugeteilt werden kann.

Jede Kante hat ein Gewicht, das die Distanz beschreibt, die überwunden werden muss, damit der Bauernhof das Tier erhalten kann. Des Weiteren besitzt jeder Knoten b_i eine Priorisierung p_i. Die Länge der Priorisierungsliste eines Knoten entspricht der Anzahl der ausgehenden Kanten $|p_i| \mathrel{\widehat{=}} |\delta^+(b_i)|$ und die Liste enthält immer die Senken der Kanten: $\forall t \in p_i \quad \exists e_{\overline{b_i t}}$. Jede Senke der Kanten, die von b_i ausgehen, muss in p_i enthalten sein.

Eine Zuordnung eines Bauernhofes b_i zu einem Tier t_j im Graphen, wird als $m_x := \overline{b_i t_j}$ definiert.

Gesucht ist eine Zuordnung aller Bauernhöfe zu Tieren, die folgende Eigenschaften erfüllt:

1. Das Matching soll perfekt sein: Alle Bauernhöfe erhalten ein Tier.
2. Außerdem soll es stabil sein: Alle Bauernhöfe erhalten ein Tier, das sie möglichst hoch priorisieren ($\sum_{i=1}^{|B|} pos(p_i, t_j)$ mit $m_{b_i t_j}$)
3. Und das Matching soll ideal sein: Die Summe aller Kosten ($\sum_{i=1}^{|B|} w_{m_i}$) soll minimal sein.

2.4 Konflikte

Zunächst gilt es ein perfektes Matching zu finden. Dies ist nicht zwingend möglich (siehe 6.2.1).

Wenn eine Zuordnung gefunden wurde, muss eine Kombination aus einem stabilen und idealen Matching gefunden werden. Die Restriktionen bezüglich des Matching lassen darauf schließen, dass dies nur mit einer Gewichtung der Faktoren oder einem Kompromiss zwischen den verschiedenen Matchingkategorien realisierbar ist:

- Ein perfektes Matching ist nicht zwingend stabil oder ideal (siehe 6.2.1). Gegeben sei ein bipartiter Graph G_B mit $U := \{1, 2, 3, 4\}$, $V := \{A, B, C, D\}$ und $E = \{\overline{1A}, \overline{1B}, \overline{1C}, \overline{2A}, \overline{2B}, \overline{2C}, \overline{2D}, \overline{3C}, \overline{3D}, \overline{4C}\}$.

  Wenn das Matching perfekt ist, gibt es **ein** hoch priorisiertes Paar ($m_{\overline{4C}}$) und **drei** niedrig priorisierte Paare ($m_{\overline{1A}}$, $m_{\overline{2D}}$, $m_{\overline{3B}}$).
  Wesentlich zufriedener wäre die Mehrheit der Paare mit **drei** hoch priorisierten Paaren (für u_i mit $1 \leq i \leq 3$ gilt: $pos(p_i, v_j) \mathrel{\widehat{=}} 1$). Dann gäbe es die Paare $m_{\overline{1B}}$, $t_{\overline{2A}}$, $m_{\overline{3C}}$.
  Die Summe der Prioritäten ist im ersten Fall ($\sum_{i=1}^{|U|} pos(p_i, v_j) \mathrel{\widehat{=}} 10$ mit $v_j \in m_{\overline{u_i v_j}}$) größer als beim zweiten Fall ($\sum_{i=1}^{|U|} pos(p_i, v_j) \mathrel{\widehat{=}} 3$ mit $v_j \in m_{\overline{u_i v_j}}$), selbst wenn für jeden nicht zugeordneten Knoten eine Strafe von $|V|+1$ addiert wird ($(\sum_{i=1}^{|U|} pos(p_i, v_j))+5 \mathrel{\widehat{=}} 8$). Allerdings ist der Graph dann nicht perfekt. (siehe Abbildung 6.2.1)

- Ein stabiles Matching muss nicht unbedingt ideal sein und umgekehrt. Gegeben sei ein G_B mit $V := \{1, 2, 3, A, B, C\}$ und $E := \{\overline{1A}, \overline{1B}, \overline{2A}, \overline{2B}, \overline{3C}\}$. Die Kanten haben Gewichte: $W := \{w_{\overline{1A}} = 0.7, \ldots, w_{\overline{3C}} = 0.8\}$.
  In diesem Beispiel geht es darum, das **niedrigste** Gesamtgewicht zu erlangen. Mit maximalem Kantengewicht ist das Matching M nicht mehr ein stabil. (siehe Abbildung 6.2.1)

Es gibt somit Graphen, die nicht alle drei Einschränkungen erfüllen können. Daher sei an dieser Stelle ein gutes Matching M_{gut}, das ein Optimum zwischen einem stabilen und idealen Matching darstellt und gleichzeitig perfekt ist.

3 Algorithmen

Im folgenden Abschnitt werden verschiedene Algorithmen vorgestellt, die die genannten Matchings finden. Die Algorithmen sind kategorisiert nach den Definitionen aus 1.3.

3.1 Lösung für perfektes und stabiles Matching

Eine Möglichkeit eine Lösung für ein perfektes und stabiles Matching zu finden bietet der 'Gale-Shapley Algorithmus'. Der 'Gale-Shapley Algorithmus' wurde erstmals 1962 in einem Paper von David Gale und Lloyd Shapley veröffentlicht. Gale und Shapley bewiesen, dass mit ihrem Algorithmus immer ein stabiles Machting gefunden wird, sofern die beiden Knotenmengen des bipartiten Graphen gleich groß sind. Der Algorithmus wird heute von der amerikanischen Organisation 'National Resident Matching Program' genutzt, um Medizinstudenten auf Lehrkrankenhäuser zu verteilen.

3.1.1 Vorgehensweise

Der Algorithmus erhält als **Input** zwei gleich große Mengen (im Beispiel: Bauernhöfe und Tiere). Für die Ausführung des Algorithmus benötigen sowohl die Bauernhöfe, als auch die Tiere eine Liste mit Präferenzen.

Voraussetzung für den Algorithmus ist, dass die Bauernhöfe und die Tiere in einer geordneten Liste gespeichert sind.

Ablauf des Algorithmus
Zunächst darf der erste Bauernhof das von ihm am höchsten priorisierte Tier anfragen. Da dieses Tier zu Beginn niemandem zugeordnet ist, kann aus dem Bauernhof und seinem gewählten Tier ein Paar gebildet werden.

Dann dürfen der Reihe nach die anderen Bauernhöfe versuchen mit ihrem am höchsten priorisierten Tier ein Paar zu bilden. Sollte ein Tier bereits vergeben sein, wenn ein Bauernhof es anfragt, entscheidet das Tier, aufgrund seiner Prioritätenliste, mit welchem Bauernhof es ein Paar bildet:

1. wenn das Tier den Bauernhof B_u bevorzugt, dem es bereits ein Paar bildet, muss der neu anfragende Bauernhof B_a das Tier prüfen, das an nächster Stelle in seiner Priorisierungsliste steht und das Paar bleibt bestehen.

2. wenn das Tier den anfragenden Bauernhof B_a bevorzugt, wird das Paar aufgelöst und es wird ein neues Paar aus B_a und dem Tier gebildet. Der Bauernhof

des ursprünglichen Paares B_u fragt das nächste Tier auf seiner Priorisierungsliste an.

Dieses Verfahren wird solange wiederholt, bis alle Bauernhöfe Teil eines Paares sind.

3.2 Lösung für perfektes und ideales Matching

Um ein perfektes Matching zu erstellen, dass gleichzeitig ideal ist, kann der 'Hungarian Algorithm' verwendet werden, der 1955 veröffentlicht wurde. Im Gegensaz zum 'Gale-Shapley-Algorithm' wird bei diesem Algorithmus ein Matching gefunden, bei dem die Kantengewichte möglichst gering sind. Es werden keine Prioritäten beachtet. Der 'Hungarian Algorithm' wird anhand eines Beispiels erklärt: Gegeben sei eine Auktion mit drei Tieren, die an drei Bauernhöfe vermittelt werden sollen. Die Kantengewichte des entstehenden Graphen stellen die Wege dar, die die Tiere während des Transports zu dem jeweiligen Bauernhof im Falle eines Kaufes, zurücklegen müssten. Ziel ist es, die Transportkosten für alle Bauernhof-Tier-Paare zu minimieren.

3.2.1 Vorgehensweise

Zur Vorbereitung werden die Gewichte des Graphen in eine nxn - Matrix übertragen. Für das in der Graphik 6.2.2 dargestellte Beispiel entspricht n der Anzahl der Bauernhöfe und der Tiere. Existieren nicht ebenso viele Bauernhöfe wie Tiere, kann der Algorithmus keine Lösung finden.

Im ersten Schritt wird aus jeder Zeile der kleinste Wert identifiziert und von allen anderen Werten der entsprechenden Zeile subtrahiert. Danach wird von jeder Spalte der Matrix der kleinste Wert bestimmt und von den anderen Werten der Spalte abgezogen: Im Anschluss kann geprüft werden, ob sich mit n Linien alle Nullen in der Matrix abdecken lassen:

- Wenn, wie in Abbildung 6.2.2 Teilbild 5 dargestellt, diese Bedingung nicht erfüllt ist, wird der kleinste der nicht abgedeckten Werten der Matrix, von allen nicht abgedeckten Werten abgezogen. An den Schnittpunkten der Linien wird derselbe Wert addiert (Teilbild 6). Danach wird wieder versucht mit n Linien alle Nullen abzudecken. Dieser Schritt wiederholt sich bis mit n Linien alle Nullen abgedeckt werden können.
- Sonst können aus dieser Darstellung alle möglichen Matchings des Graphens abgelesen werden. Auf das Beispiel bezogen: jede Kombination von drei Nullen (eine Null je Linie) stellt eine mögliche Zuordnung der drei Tiere zu Bauernhöfen dar (Teilbild 10). Sollte es mehrere Möglichkeiten geben, haben diese eine identische Summe an Gewichten und sind demnach gleich ideal.

3.3 Lösung für stabiles und ideales Matching

Ziel ist es **beide**, voneinander unabhängigen Faktoren (Priorisierung und Gewichte), in das Matching einzubeziehen. Dafür gibt es drei naheliegende Ansätze:

- Der erste Ansatz (**Prio on Weight**) ist, die Priorität auf die Gewichte der Kanten zu addieren. Danach wird regulär der 'Hungarian Algorithm' durchgeführt. Das Gewicht einer Kante $\overline{b_i t_j}$ wird um die Position von t_j in der Priorisierungsliste von b_i erhöht. Da die Kantengewichte sehr groß sein können, kann statt der reinen Indizes, die Werte der Fibonacci Reihe verwendet werden, damit der Einfluss der Prioritäten nicht verloren geht. Indirekt werden so hoch priorisierte Tiere gegenüber Tieren, die niedriger priorisiert sind, aber auch niedrigere Gewicht haben, bevorzugt.
  Das Ergebnis ist ein ideales Matching unter Berücksichtigung der Priorisierung des Graphen.

- Die zweite Abwandlung (**Weight on Prio**) bekannter Methoden, um ein möglichst ideales und stabiles Matching zu erhalten, soll die Priorisierung nicht den Gewichten vorziehen. Stattdessen werden diesmal die Gewichte in die Priorisierungslisten einbezogen. Danach wird der 'Gale-Shapley-Algorithmus' angewendet.

  Um die Gewichte zu berücksichtigen, wird aus der Priorisierungsliste p_i und den Gewichten der Kanten zu den jeweiligen Tieren eine neue Liste mit derselben Länge abgeleitet:

  Das bedeutet für einen Bauernhof b_i wird über alle Gewichte seiner ausgehenden Kanten ein Durchschnitt d gebildet. Dann wird für jedes Tier $\in p_{b_i}$ der Reihe nach ermittelt, ob das Gewicht der Kante größer oder kleiner d ist.

  - Ist das Gewicht größer als d, wird das Tier wegen des hohen Kantengewichts bestraft, indem zunächst eine leere Stelle in die neue Liste eingefügt wird, bevor es in die neue Liste übertragen wird.
  - Ist das Gewicht kleiner als d, wird das Tier direkt an die nächste freie Stelle in der neue Liste übertragen.
  - Wenn der maximale Index der neuen Liste erreicht ist, wird das Tier an der nächsten freien Stelle in der neuen Liste eingetragen.

  Mit den modifizierten Priorisierungslisten für alle Bauernhöfe wird nun der 'Gale-Shapley-Algorithmus' ausgeführt.
  Das Ergebnis ist ein stabiles Matching, unter Berücksichtigung der Kantengewichte des Graphen.

- Der dritte Ansatz (**Awesome Algorithm**) bevorzugt keinen der beiden Faktoren. Es werden zunächst der 'Gale-Shapley-Algorithmus' und der 'Hungarian Algorithmus' für den Graphen durchgeführt. Daraus ergeben sich zwei Matchings, die evtl. widersprüchlich sind. Aus diesen beiden Zwischenlösungen werden alle Paare, die in beiden Versionen vorkommen direkt übernommen. Eine Präferenz (Priorisierung oder Gewicht) muss nun angeben werden. Angenommen die Präferenz läge auf der Priorisierung, dann würde das erste Paare des stabilen Matchings übernommen werden. Die übrigen Paaren werden neu gemacht. Durch das Wegfallen des gewählten Paares, können sich neue Überschneidungen von idealem und stabilen Matching ergeben. Das Verfahren wird solange wiederholt, bis es keine Möglichkeit der Paarbildung mehr gibt. Eine Visualisierung findet sich im Anhang: siehe 6.2.3).

3.4 Fazit

Fazit zum Ansatz 'Prio on Weight'
Dieser Ansatz findet nicht immer das Matching, was für die Kombination beider Faktoren das Optimum ist. Aber er bietet eine Möglichkeit die Priorisierung **und** das Gewicht bei einem Matching zu berücksichtigen. Die erste Abwandlung bevorzugt trotzdem eindeutig die Gewichte eines Graphen, da als Lösungsalgorithmus der 'Hungarian Algorithm' verwendet wird und dieser Algorithmus ein ideales Matching findet. Sind die Gewichte sehr groß (> 100) hat der Index trotz Fibonacci-Reihen wenig Auswirkung.

Fazit zum Ansatz 'Weight on Prio'
Hier wird ebenso wie beim erste Ansatz einer der beiden Faktoren bevorzugt. Da die Priorisierungsliste im Fall eines hohen Gewichts angepasst wird, kann angenommen werden, dass die Summe der Gewichte im endgültigen Matching niedriger ist, als sie ohne 'Neusortierung' im stabilen Matching des Graphen gewesen wäre. Dadurch wird ein Einfluss des Gewichts im Gegensatz zum 'Gale-Shapley-Algorithm' deutlich, allerdings findet der Algorithmus kein Optimum zwischen Gewicht und Präferenz.

Fazit zum Ansatz 'Awesome Algorithm'
Nach dem 'Awesome Algorithm' werden die gemeinsamen Paare ausgewählt. Der Gedanke dahinter ist, dass ein Paar, das sowohl im stabilen als auch im idealen Matching enthalten ist, sehr gut sein muss. Nachdem ein weiteres Paar (stabil oder ideal) gewählt wurde, wird ein neues stabiles und ein neues ideales Matching für die übrigen Paare bestimmt. Hierbei kann sich ein schlechteres Matching bilden als zuvor, da durch gewisse Paare Möglichkeiten wegfallen können oder durch die Präferenz eines Matchings (stabil oder ideal) ein schlechteres Matching entsteht (siehe 6.2.4).

Gesamtfazit Der Nachteil, den alle Algorithmen mit sich bringen ist, dass schlus-

sendlich ein Verfahren (entweder die Priorisierung oder das Gewicht) präferiert wird und ausschlaggebend für das resultierende Matching ist. Beim ersten Ansatz ist es die Priorisierung und beim zweiten Ansatz das Gewicht und auch beim dritten Ansatz wird sich für eines der beiden entschieden. Des Weiteren fällt auf, dass bei allen bisherigen Algorithmen die Priorität der Gegenseite (der Tiere) wenig Beachtung findet. Bei der Entwicklung eines neuen Algorithmus sollte dies, wenn möglich, in Zukunft stärker statt finden.

4 Synthese

Bei den vorherigen Lösungen war immer ein perfektes Matching angestrebt. Allerdings kann sich ein vom perfekten Matching abweichendes Matching für ein Optimum besser eignen (siehe Widerspruch stabiles und perfektes Matching). Bezieht man diese Beobachtung in die Lösungsfindung mit ein, so hat man drei verschiedene Parameter die gleichwertig zu betrachten sind: das perfekte, stabile und ideale Matching.

4.1 Analyse

Die klassischen Matchingverfahren ('Gale-Shapley-Algorithmus' und 'Hungarian Algorithm') finden immer ein perfektes Matching (wenn möglich). Daher muss dies bei der Bewertung der Ergebnisse des Algorithmus mit einbezogen werden. Die Vernachlässigung von Paaren zugunsten einer höheren Stabilität bzw. Idealität des Matching ist nicht vorgesehen. Folglich muss geprüft werden, ob eine höheren Gesamtzufriedenheit erreicht werden kann, wenn Knoten ignoriert werden und dementsprechend nicht im Matching erscheinen.

Die weiterentwickelten Verfahren (3.3) versuchen eine Kombination aus den klassischen Matchingverfahren herzuleiten. Der intuitive Ansatz, den einen Parameter (z.B. das Gewicht) mit dem anderen (die Priorisierung) zu verknüpfen, hat jedoch die Schwäche, dass das Verfahren, das zur Bestimmung des Matchings herangezogen wird, dafür ausschlaggebend ist, welcher Parameter mehr gewichtet wird. Daraus resultiert, dass eine andere Art der Kombination genutzt werden muss. Es bietet sich an, die Art von Matching zu wählen, die keine große Verschlechterung für zukünftig Matchings (sowohl ideal als auch stabil) bewirkt. Eine Funktion zur Bestimmung eines Grenzwertes bietet sich an.

Im 'Awesome Algorithm' werden, wenn es Überschneidungen zwischen dem idealen und stabilen Matching gibt, in einem Schritt mehrere Paare gebildet. Dies kann bei der Suche nach der höchsten Gesamtzufriedenheit hinderlich sein. Wenn ein Paar des guten Matching festgelegt wird und im Anschluss werden für das nächste Paar wieder das stabile und ideale Matching gesucht, dann können sich dort neue Kombinationen ergeben, da ein Paar (also zwei Knoten und mindestens eine Kante) von der Betrachtung wegfallen. Wenn in einem Schritt schon vier Paare gematcht werden, dann wird ignoriert, dass es vielleicht besser gewesen wäre, eines der Paare schlechter zu matchen um die anderen besser matchen zu können. Oder dass durch Iteration des Verfahrens die Quellen der Paare noch bessere Senken finden (höhere Priorität oder geringeres Gewicht), denn aus unterschiedlichen Eingaben resultieren verschiedene Matchings und jede Iteration hat eine andere Eingabe an Knoten und Kanten. Es ist also erforderlich, dass die Zuordnung von Paaren in einzelnen Schritten erfolgt (pro

Iteration nur ein Paar), damit die Auswirkung des einzelnen Paars besser bewertet werden kann.

4.2 Umsetzungsidee

Die Idee besteht darin, die Matchings zu bewerten. Bei dem Versuch der Kombination der Verfahren ist das größte Problem, dass durch Auswahl bestimmter Paare die nachfolgenden Matchings sehr schlecht in Bezug auf Stabilität/Idealität werden können. Um dies zu verhindern, wird nun ein Maß zur Bewertung des Matching eingeführt. Der Wert des Maßes liegt immer zwischen 0 und 1, wobei sich ein gutes Matching durch einen Wert nahe 1 und ein Schlechtes sich durch einen Wert nahe 0 auszeichnet.

Es wird zunächst ein stabiles und ein ideales Matching gebildet. Dann wird bewertet welches Matching besser ist, also welcher Wert näher an der 1 ist. Dieses Matching wird nun gewählt und innerhalb des Matchings wird das *beste Paar* gewählt. Im Falle eines stabilen Matchings, das mit der höchsten Priorisierung - hier können die Tiere einbezogen werden: geringste Summe von der Priorisierung des Bauernhofes und des zugeteilten Tiers. Ist es ein ideales Matching: das geringste Gewicht.

Nach Festlegung eines Paares wird geprüft, wie gut das stabile und das ideale Matching nach einer Neuberechnung sind. Wenn die neuen Maße um $\mathbf{alterWert} - \frac{1}{e^{(\mathbf{alterWert}+1)}}$ abweichen (Abweichungsfunktion), kann das Paar gewählt werden, ansonsten wird es verworfen und das nächste Paar betrachtet werden.

Wenn ein Paar gewählt wurde, wird das Verfahren für die restliche Menge wiederholt.

4.3 Maße

Es werden zwei Maße definiert (siehe 2.3): das **Stabilitätsmaß** ($\sum_{i=1}^{|B|} pos(p_i, t_j)$ **mit** $m_{b_i t_j}$) und das **Idealitätsmaß** ($\sum_{i=1}^{|B|} w_{m_i}$). Diese Funktionen sind ausbaufähig: Sie dienen zur Bestimmung der Stabilität bzw. Idealität des Matching. Jedoch berechnen sie einen Wert, der in keinem Verhältnis steht. Das optimale Matching hat den Wert 1, dass heißt, dass bei einem stabilen Matching bei dem jeder Knoten seine höchste Priorität hat, das Stabilitätsmaß 1 ergeben muss. Das schlechteste Matching (jeder Knoten erhält nur einen Partner seiner geringsten Priorität) geht gegen 0. Folgende

Formel zur Berechnung des Stabilitätsmaß ergibt sich:

$$\begin{aligned} M^*_{sta} = 1 - (&\frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} \\ &+ \frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|}) \\ &+ \frac{\frac{2}{3} * |B|}{\sum_{y=1}^{|B|} max(pos(b_y))} \end{aligned}$$

Die Division $\frac{\sum_{i=1}^{|M_{sta}|} pos(b_i,t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(pos(b_i))+|B|-|M_{sta}|}$ berechnet das Verhältnis der Indizes in der Priorisierungsliste bei dem gefundenen stabilen Matching zu den Indizes des schlechtmöglichsten Matching. Dabei werden auch die nicht zugeordneten Knoten einbezogen. Wenn bei einem Matching jeder Knoten seinem am höchsten priorisierten Knoten zugeordnet wurde (der Index entspricht überall 1), soll das Stabilitätsmaß 1 sein. Daher wird der Wert der Division (je kleiner dieser Wert desto besser das Matching) von 1 abgezogen. Die 1 wird bei der Subtraktion von 1 jedoch nie erreicht, da die Division nicht 0 werden kann. Das kleinstmögliche Ergebnis ist, wenn z.B. ein Matching mit 3 Knoten, von denen jeder Knoten mit dem Knoten seiner höchsten Priorität gematcht ist, es keine alleinestehenden Knoten gibt und jeder Knoten 3 Knoten auf seiner Priorisierungsliste stehen hat, existiert:

$$\begin{aligned} &\frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} \\ &+ \frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} \\ =&\frac{2}{3} * \frac{\sum_{i=1}^{3} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{3} max(pos(b_i)) + 3 - 3} \\ &+ \frac{1}{3} * \frac{\sum_{z=1}^{3-3} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{3} max(pos(b_i)) + 3 - 3} \\ =&\frac{2}{3} * \frac{1+1+1}{3+3+3+3-3} = \frac{2}{3} * \frac{3}{9} = \frac{6}{27} \end{aligned}$$

Damit bei dem besten Ergebnis, das ein Matching ergeben kann (im Beispiel oben wäre das Stabilitätsmaß also $1 - \frac{6}{27}$), das Maß 1 wird, muss zum Ausgleich der kleinste Wert addiert werden: $\frac{\frac{2}{3}*|B|}{\sum_{y=1}^{|B|} max(pos(b_y))} + \frac{2}{9}$. Zuletzt werden noch mit einem Gewicht von $\frac{1}{3}$ die nicht gematchten Knoten einbezogen, da sie weniger 'wichtig' für das normale Matching sind als die Summe der gematchten Indizes, da dort die

entscheidene Varianz ist. Es wird der schlechteste Wert der Priorisierungsliste gewählt und 1 addiert, um das Nichtzustandekommen eines Matchings zu bestrafen. Da hier der Wert um 1 erhöht wird, müssen die nicht gemachten Knoten auch im Nenner betrachtet werden - $|B| - |M_{sta}$. Auf das ideale Matching analog übertragen heißt das wiederum, dass sich das folgende Idealitätsmaß mit denselben Ansätzen finden lässt:

$$M^*_{ide} = 1 - (\frac{2}{3} * \frac{\sum_{i=1}^{|M_{ide}|} w_{b_i,t_j} \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} + \frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{ide}|} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|}) + \frac{\frac{2}{3} * \sum_{y=1}^{|B|} min(w_{b_y})}{\sum_{y=1}^{|B|} max(w_{b_y})}$$

Ein passendes Beispiel für die erfolgreiche Anwendung der Maße befindet sich unter 6.2.5. Es handelt sich dabei um ein Gegenbeispiel für den Erfolg des 'Awesome Algorithm' (6.2.3).

4.4 Bemerkungen

- Der Algorithmus terminiert nicht nachdem ein perfektes Matching gefunden wurde, sondern errechnet alle möglichen Matchings um das insgesamt beste festlegen zu können. Dies kann durch Addition beider Maße bei jedem Schritt zur Bestimmung des Matchings erfolgen.
- Es kann jedoch sein, dass der Algorithmus terminiert ohne das ein Matching gefunden wurde. Dies ist der Fall, wenn bei allen Möglichkeiten wegen der Abweichung abgebrochen werden muss.
- Die Abweichungsfunktion erlaubt für niedrige Werte eine große Abweichung und für gute Ergebnisse eine geringe Abweichung. Dadurch wird sichergestellt, dass das Matching sich nicht stark verschlechtert, aber deutlich verbessern kann.

5 Evaluation und Fazit

5.1 Problemlösung

Aus den vorhergehenden Überlegungen kann geschlossen werden, dass es sich bei dem Problem um ein Optimierungsproblem handelt. Nach Möglichkeit soll ein perfektes Matching entstehen. Diese Forderung konkurriert mit zwei Minimierungsproblemen:

- Das erste Minimierungsproblem entspricht dem Anspruch ein ideales Matching zu finden. Daher soll die Summe aller Kantengewichte möglichst gering sein.
- Außerdem ist ein stabiles Matching angestrebt. Dementsprechend besteht das zweite Minimierungsproblem darin, dass die Summe aller gewählten Indizes in den Priorisierungslisten möglichst gering sein soll.

Es gilt zwei Fälle zu unterscheiden: Unter der Voraussetzung, dass ein perfektes Matching gefunden wir, lassen sich die beiden Teilalgorithmen deterministisch in polynomieller Zeit lösen (mithilfe des 'Hungarian Algorithm' oder des 'Gale Shapley-Algorithm'). Darf widerum die Zufriedenheit Einzelner Knoten extrem niedrig sein, um eine möglichst hohe Gesamtzufriedenheit zu erreichen (das Matching ist also nicht mehr perfekt), dann sind schon die Minimierungsprobleme in sich nicht deterministisch lösbar, da es keinen Auslöser gibt, der andeutet, dass ein Knoten aus dem Matching ausgeschlossen werden sollte. Auch dies muss auf gut Glück gelöst werden. Außerdem ist die Kombination der beiden Probleme nicht deterministisch lösbar, wenn nicht die Lösung eines der beiden Minimierungsprobleme bevorzugt wird.

Über ein Berechnen aller Möglichkeiten kann natürlich die beste Lösung (mit dem Maßen, die schon zu Beginn eingeführt wurden) bestimmt werden. Allerdings ist es nicht möglich deterministisch zur Laufzeit zu entscheiden, welches Paar an einer bestimmten Stelle bevorzugt werden muss, um das Optimum sicherzustellen. Durch die Abweichungsfunktion kann dieses Problem zwar gemindert, aber nicht eindeutig gelöst werden. Daraus kann geschlossen werden, dass allein die optimale Lösung in Bezug auf Idealität und Stabilität zu finden, ein Problem ist, das nicht in deterministisch lösbar ist.

Es muss in jedem Fall entweder die Stabilität oder die Idealität bevorzugt werden bzw. einer der beiden Faktoren gewählt werden. Dadurch wird für den benachteiligten Faktor keine optimale Lösung gefunden. Wenn die Stabilität präferiert wird, ist es möglich, dass das bestimmte Lösungs-Matching einen schlechteren Wert für das Idealitätsmaß beinhaltet, als das ideale Matching für den Graphen hätte.

Der Nachweis jedoch, dass ein unter dem Aspekt der Stabilität bestimmtes Matching einen Wert für das Stabilitätsmaß nahe 1 hat oder nicht, ist in polynomieller Zeit

berechenbar. Es lässt sich durch das Stabilitätmaß auch eindeutig feststellen, welches von n Matchings die höchste Stabilität hat. Dadurch lässt sich einfach und eindeutig nachprüfen, ob der Algorithmus die optimale Lösung (in Bezug auf Stabilität) für den Graphen gefunden hat oder nicht. Allerdings werden immer Vergleichswerte benötigt, dementsprechend kann die Überprüfung nur nach Berechnung aller Möglichkeiten stattfinden.

Das Idealitätsmaß ermöglicht dieselbe eindeutige Überprüfung für den Aspekt der Idealität des von einem Algorithmus bestimmten Matching. Deshalb können sie nur zur Validierung des Ergebnisses und nicht während der Ausführung des Algorithmus zur Findung der optimalen Lösung für beide Faktoren genutzt werden. Die Maße für die Lösung des guten Matching (M_{gut}) heranzuziehen war nur möglich, da in jedem Schritt alle Knoten sowohl mit dem ‚Gale-Shapley-Algorithm' als auch mit dem ‚Hungarian Algorithm' gematcht werden.

Abschließend kann konkludiert werden, dass es sich bei dem Problem um ein Problem handelt, das sich aus mehreren Teilproblemen zusammensetzt. Diese Teilprobleme sind nicht deterministisch lösbar, wenn kein perfektes Matching angestrebt ist (die Minimierungsprobleme). Das gesamte Optimierungsproblem lässt sich dementsprechend auch nicht deterministisch lösen, da in jedem Iterationsschritt während der Ausführung des Algorithmus für die Lösung eines guten Matchings ein lokales Optimum (das zum aktuellen Zeitpunkt beste Matching) gewählt wird und nicht ein globales Optimum (die insgesamt beste Möglichkeit).

Das Problem, eine optimale Lösung in Bezug auf Idealität **und** Stabilität zu finden, ist demnach nur annäherungsweise gelöst. Eine immer eindeutige Lösung, die für beide Faktoren optimal ist, gibt es nicht.

5.2 Ergebnis

Für die Lösung des in 2 beschriebenen Problems konnte kein deterministisches Verfahren gefunden werden. Damit die Entwicklung eines entsprechenden Lösungsalgorithmus möglich ist, müssen zunächst deterministische Verfahren zur Lösung der Minimierungsprobleme gefunden werden, die kein perfektes Matching anstreben. Erst dann kann versucht werden für das Optimierungsproblem ein deterministisches Verfahren zu entwickeln.

Die gefundene Lösung (vergleiche 4.2) unter Verwendung der Maße und der Abweichungsfunktion minimiert zuvor analysierte Fehlerquellen, wie z.B. Verschlechterung des gesamten Matchings durch die Auswahl eines einzelnen Paares in einem Iterationsschritt. Jedoch garantiert das Verfahren keine optimale Lösung, da beispielsweise die Abweichungsfunktion in jeder Iteration trotzdem eine kleine Verschlechterung des

Ergebnissen zulässt. Besonders verstärkt kann diese Verschlechterung bei einer höheren Anzahl an Iterationen auftreten, wenn sich die Verschlechterungen pro Schritt aufsummieren.

Literaturverzeichnis

1. Graph Theory II (Tom Leighton and Ronitt Rubinfeld - Mathematics for Computer Science 2006)
2. Matching Theory (L. Lovász and M. D. Plummer - North Holland 1986) [Basic Terminology & Matchings in Bipartite Graphs]
3. The Hungarian method for the assignment problem (H. W. Kuhn - Bryn Nair College)
4. College Admissions and the Stability of Marriage (D. Gale and L. S. Shapley - The American Mathematical Monthly 1962) [Vol. 69, No. 1, pp. 9-15]
5. Gale-Shapley algorithm simply explained (Alexander Osipenko 2019) [https://towardsdatascience.com/gale-shapley-algorithm-simply-explained-caa344e643c2 - letzter Zugriff am 19.12.2019 - 07:57 Uhr]
6. College admissions and the stability of marriage (D. Gale and L. S. Shapley - Department of Mathematics, Brown University 1962)
7. A distributed auction algorithm for the assignment problem (M. M. Zavlanos and L. Spesivtsev and G. J. Pappas - 47th IEEE Conference on Decision and Control 2008)
8. An Efficient Algorithm for the "Optimal" Stable Marriage (Robert W. Irving and Paul Leather and Dan Gusfield - Association for Computing Machinery (NY USA) 1987)
9. Machiavelli and the Gale-Shapley Algorithm (L. E. Dubins and D. A. Freedman - The American Mathematical Monthly 1981)
10. A Survey of the Stable Marriage Problem and Its Variants (K. Iwama and S. Miyazaki - International Conference on Informatics Education and Research for Knowledge-Circulating Society 2008)
11. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the technion (Nitsan Perach and Julia Polak and Uriel Rothblum - International Journal of Game Theory 2008)
12. How the matching algorithm works (The Match - National Resident Matching Program 2020) [www.nrmp.org/matching-algorithm- letzter Zugriff am 08.01.2020 - 16:32 Uhr]

13. Diverse Weighted Bipartite b-Matching (Faez Ahmed, John P. Dickerson, Mark Fuge - Cornell University 2017) [https://arxiv.org/abs/1702.07134 - letzter Zugriff am 08.01.2020 - 16:44 Uhr]

6 Anhang

6.1 Pseudocode

6.1.1 Gale-Shapley-Algorithm

```
/*** bauernhof b, bauernhoefe B, tier t, Tiere T,
*     matching m, paar p
***/

eingabe B, T:
        solange B nicht leer :
                solange b noch nicht gematcht wurde:
                        nimm den ersten bauernhof b1 aus B
                        nimm das von b1 am meisten priorisierte t1
                        wenn t1 noch nicht gematcht ist:
                                matche b1 und t1 zu p
                                fuege das p zu m hinzu
                                entferne b1 aus B
                        ansonsten:
                                nimm das p1, das aktuell t1 enthaelt
                                wenn der b aus p1 von t1 hoeher priorisiert wird:
                                        loesche t1 aus der prio von b
                                ansonsten:
                                        entferne p1 aus dem m
                                        matche b und t1
                                        fuege das neue p2 hinzu
                                        loesche b aus B
                                        fuege den b aus p1 zu B hinzu
        gib das m zurueck
```

6.1.2 Hungarian Algorithm

```
/*** bauernhof b, bauernhoefe B, tier t, Tiere T,
*     matching m, kanten k (in einer Matrix)
*     zeile z, spalte s, element e
*     horizontal hz, vertikal vk, linie ln, nullen nl
*     anzahl anz, minimum min
***/

eingabe k:
```

```
9         tausche alle  0 gegen unendlich
10
11        fuer alle  z:
12                suche das min der z
13                ziehe  das min von jedem e von z ab
14        fuer alle  s:
15                suche das min der s
16                ziehe  das min von jedem e von s ab
17
18        kopie k1 von k erstellen
19        k1 mit nullen fuellen
20
21        fuer alle  e in k1:
22                wenn e in k 0 ist :
23                        zaehle nl, die durch eine hz ln abgedeckte werden koennen
24                        zaehle nl, die durch eine vk ln abgedeckt werden koennen
25                        wenn die hs ln mehr nl abdeckt:
26                                setze  e in k1 auf -( anz der nl, die durch hz ln abgedeckt werden)
27                sonst:
28                        setze  e in k1 auf anz der nl, die von der vk ln abgedeckt werden
29
30                erstelle  eine kopie k2 von k1
31                fuer jedes e in k1:
32                        wenn e > 0:
33                        zaehle eine ln
34                                fuer jedes e der s:
35                                        setze e1 in k1 = 0
36                                        setze e2 in k2 = 1
37                        sonst, wenn der e < 0:
38                                zaehle eine ln
39                                fuer jedes e der z:
40                                        setze e1 in k1 = 0
41                                        setze e2 in k2 = 1
42
43        solange die anz der gefundenen ln < groesse von k ist:
44                finde das min der nicht von ln abgedeckten e
45                addiere min auf die schnittpunkte der ln
46                subtrahiere min von den nicht von einer ln bedeckte e
47                finde die min anz an ln fuer die neue matrix
48
```

```
49          finde moegliches m (die nl stellen moegliche kombinationen dar)
```

6.1.3 prio auf weight

```
1  /*** bauernhof b, bauernhoefe B, tier t, Tiere T,
2  *       priorisierungsliste    p
3  ***/
4
5  eingabe B, kanten:
6           fuer jeden b:
7                   fuer jedes t aus p:
8                           nimm position in p (fibonacci)
9                           fuege position zum gewicht der kante von b zu t hinzu
10
11          fuehre den hungarian algorithmus durch
```

6.1.4 weight on prio

```
1  /*** bauernhof b, bauernhoefe B, tier t, Tiere T,
2  *       priorisierungsliste    p, durchschnitt d
3  ***/
4
5  eingabe B, T, kanten:
6           fuer jeden b:
7                   berechne den d der ausgehenden kantengewichte eines b
8
9           fuer jeden d:
10                  fuer jedes t in p von b:
11                          wenn das gewicht der kante von b > als der d des b
12                          und die neue p nicht groesser als die jetztige liste - 1 ist:
13                                  fuege erst ein leeres element zur neuen p
14                                  dann t zur neuen p hinzu
15                          ansonsten:
16                                  wenn liste leer:
17                                          fuege das t zu p hinzu
18                          ansonsten:
19                                  ersetze erstes leeres element in der neuen p durch t
20
21          fuer jeden b:
22                  tausche die alte p mit der neuen
23                  fuehre den gale-shapley algorithmus durch
```

6.2 Bilder

6.2.1 Konflikte

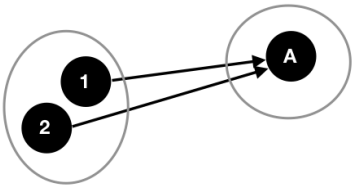
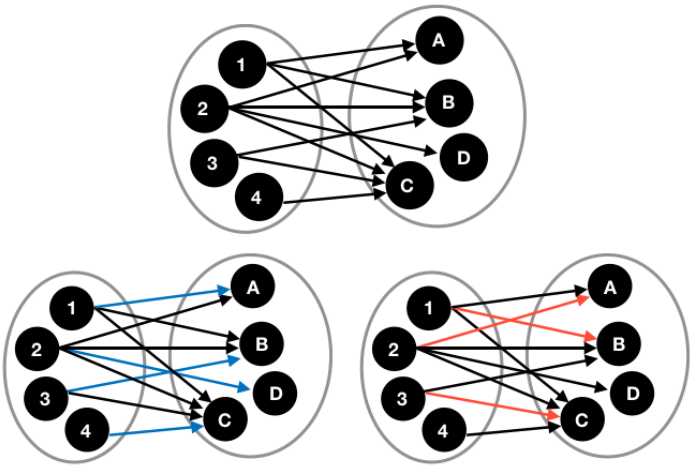


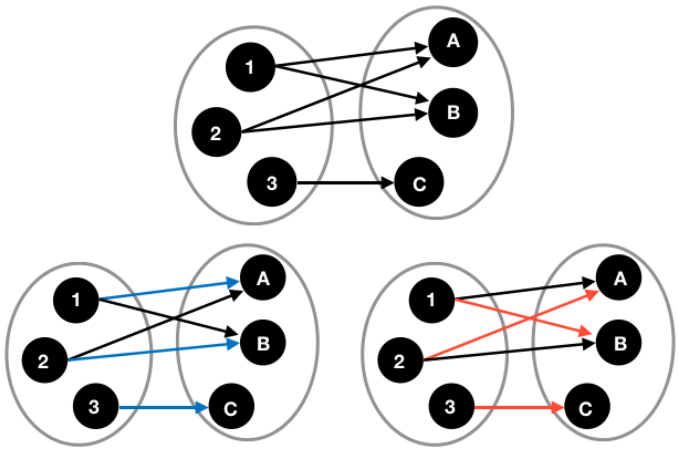
Abbildung 1: Perfektes Matching nicht möglich



Knoten	1. Prio	2. Prio	3. Prio	4. Prio
1	B	C	A	
2	A	B	C	D
3	C	B		
4	C			

Knoten	1. Prio	2. Prio	3. Prio	4. Prio
1	B	C	A	
2	A	B	C	D
3	C	B		
4	C			

Abbildung 2: Widerspruch stabiles und perfektes Matching



Knoten	1. Prio	2. Prio
1	A	B
2	B	A
3	C	

Knoten	A	B	C
1	0.7	–	0.4
2	0.2	0.9	–
3	–	–	0.8

Abbildung 3: Widerspruch stabiles und ideales Matching

6.2.2 Hungarian Algorithm

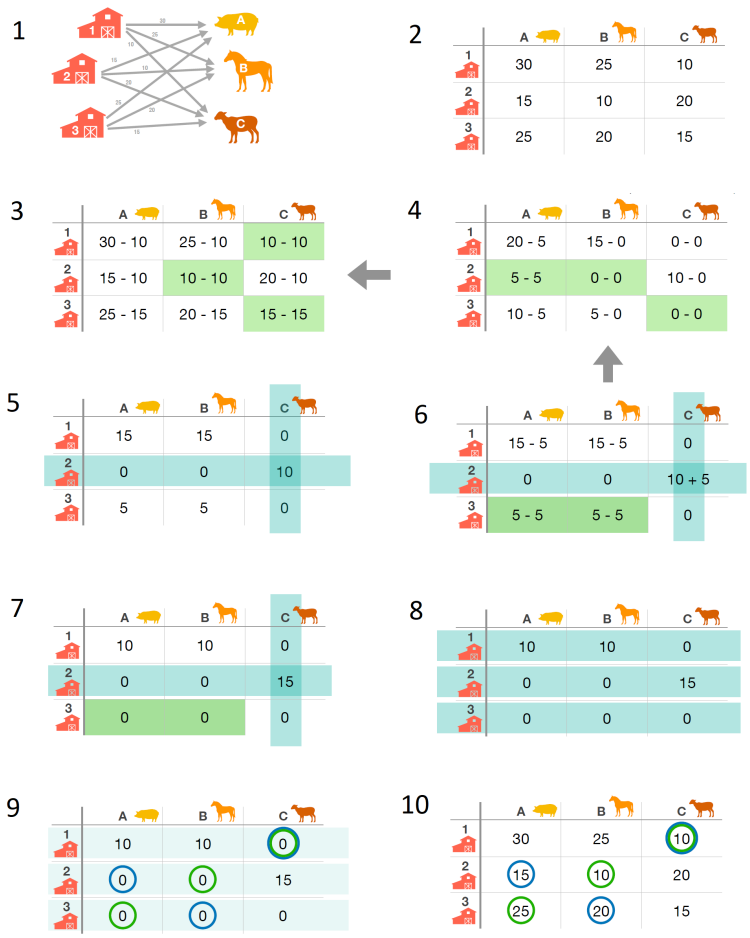
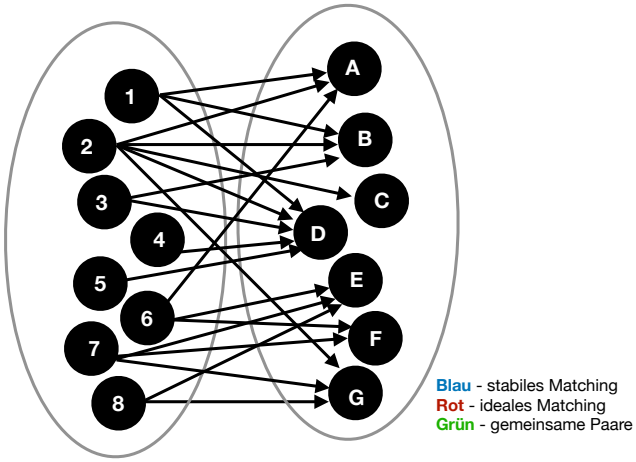


Abbildung 4: Die Gewichte des Beispielgraphen (1) werden in eine 3x3 Matrix (2) übertragen. Der kleinster Wert jeder Zeile (3) wird von jedem Wert der Zeile abgezogen, analog für die Spalten (4). Die Geraden können nicht alle Nullen der Matrix abdecken (5), weshalb der kleinste aller nicht abgedeckten Werte -5- von allen nicht abgedeckten Werten abgezogen werden muss (6). Nach der Subtraktion (7), lassen sich dann alle Nullen mit 3 Geraden abdecken (8). Die möglichen Kombinationen von je einer Null pro Linie, sind mit grünen bzw. blauen Kreisen markiert(9). Bei einer Übertragung der markierten Stellen in die Matrix (10) mit den ursprünglichen Werten, kann die Gesamtzahl der Gewichte abgelesen werden: 45.

6.2.3 Awesome Algorithm

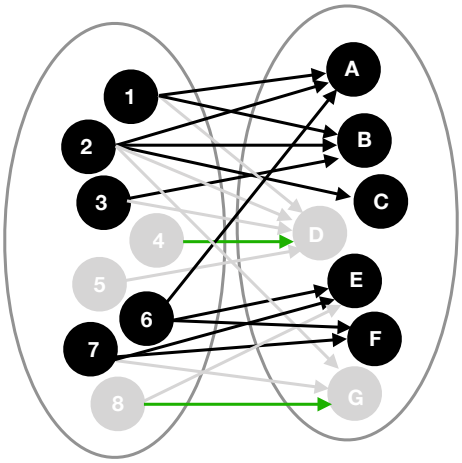
Die nächsten Seiten führen den Awesome Algorithmus vor.



Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1	B	D	A		
2	A	B	C	D	G
3	D	B			
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

Knoten	A	B	C	D	E	F	G
1	0.1	0.5		0.4			
2	0.4	0.5	0.2	0.8			0.4
3		0.3		0.1			
4				0.2			
5				0.6			
6	0.4				0.2	0.1	
7					0.7	0.5	0.9
8					0.8		0.3

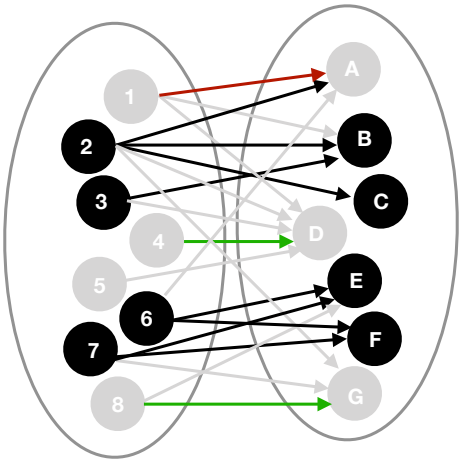
Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							



Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1	B		A		
2	A	B	C		
3		B			
4					
5					
6	F	E			
7	E		F		
8		E			

Knoten	A	B	C	D	E	F	G
1	0.1	0.5					
2	0.4	0.5	0.2				
3		0.3					
4							
5							
6	0.4				0.2	0.1	
7					0.7	0.5	
8							

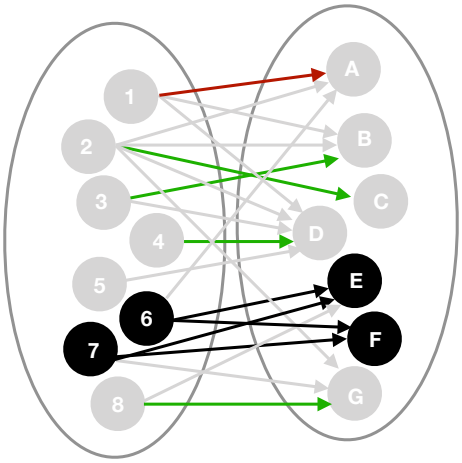
Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							


1
2
3
4
5
6
7
8
A
B
C
D
E
F
G

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2		B	C		
3		B			
4					
5					
6	F	E			
7	E		F		
8					

Knoten	A	B	C	D	E	F	G
1							
2		0.5	0.2				
3		0.3					
4							
5							
6					0.2	0.1	
7					0.7	0.5	
8							

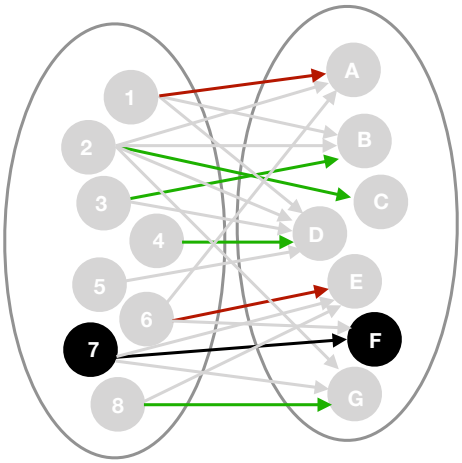
Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							


1
2
3
4
5
6
7
8
A
B
C
D
E
F
G

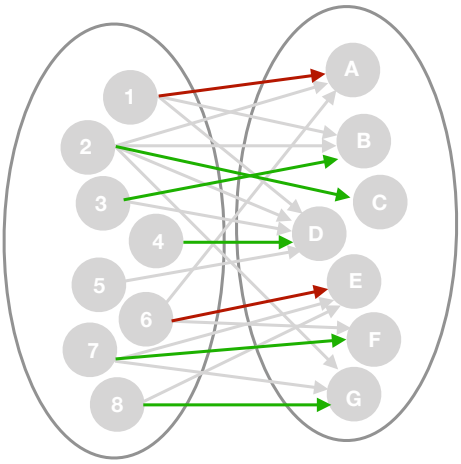
Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2					
3					
4					
5					
6	F	E			
7	E		F		
8		E			

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6					0.2	0.1	
7					0.7	0.5	
8							

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

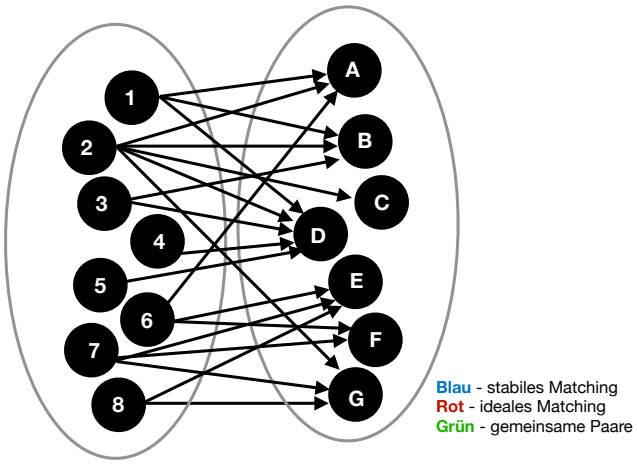

1
2
3
4
5
6
7
8
A
B
C
D
E
F
G

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							



6.2.4 Paare fallen weg

Auf den nächsten Seiten wird Schritt für Schritt ein Situation hergeleitet, bei der ein Paar wegfällt, durch das es eine bessere Lösung gegeben hätte.



Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1	B	D	A		
2	A	B	C	D	G
3	D	B			
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

Knoten	A	B	C	D	E	F	G
1	0.6	0.2		0.8			
2	0.4	0.7	0.3	0.5			0.9
3		0.8		0.5			
4				0.7			
5				0.8			
6	0.1				0.9	0.3	
7					0.5	0.6	0.2
8					0.5		0.3

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2	A		C	D	G
3	D				
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

Knoten	A	B	C	D	E	F	G
1							
2	0.4		0.3	0.5			0.9
3				0.5			
4				0.7			
5				0.8			
6	0.1				0.9	0.3	
7					0.5	0.6	0.2
8					0.5		0.3

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2					
3	D				
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

Knoten	A	B	C	D	E	F	G
1							
2							
3				0.5			
4				0.7			
5				0.8			
6					0.9	0.3	
7					0.5	0.6	0.2
8					0.5		0.3

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2					
3	D				
4	D				
5	D				
6					
7	E	G			
8	G	E			

Knoten	A	B	C	D	E	F	G
1							
2							
3				0.5			
4				0.7			
5				0.8			
6							
7					0.5		0.2
8					0.5		0.3

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2					
3	D				
4	D				
5	D				
6					
7					
8	G				

Knoten	A	B	C	D	E	F	G
1							
2							
3				0.5			
4				0.7			
5				0.8			
6							
7							
8							0.3

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1					
2					
3	D				
4	D				
5	D				
6					
7					
8					

Knoten	A	B	C	D	E	F	G
1							
2							
3				0.5			
4				0.7			
5				0.8			
6							
7							
8							

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Besser wäre Folgendes gewesen:

Knoten	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							

Da sowohl Knoten 3 als auch 4 und 5 den Knoten D an erster Stelle ihrer Priorisierungsliste haben, wäre es sinnvoller, die Kombination 3-D zu wählen, weil dort das Gewicht niedriger ist. Da bei dem Verfahren aber immer ein zuvor festgelegtes Matching präferiert wird (ideal oder stabil) kommt es hier zu einer „schlechteren Entscheidung“.

6.2.5 Maße

Die folgenden Seiten zeigen beispielhafte Bewertungen der Matchings vom vorherigen Beispiel 6.2.4 unter Verwendung der Maße (Stabilitäts- und Idealitätsmaß).

$$M^*_{sta} = 1 - (\frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, l_j) \text{ mit } mat_{b_i,l_j}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} + \frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|}) + \frac{\frac{2}{3} * |B|}{\sum_{y=1}^{|B|} max(pos(b_y))}$$

$$M^*_{ide} = 1 - (\frac{2}{3} * \frac{\sum_{i=1}^{|M_{ide}|} w_{b_i,l_j} \text{ mit } mat_{b_i,l_j}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} + \frac{1}{3} * \frac{\sum_{i=1}^{|B|-|M_{ide}|} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|}) + \frac{\frac{2}{3} * \sum_{y=1}^{|B|} min(w_{b_y})}{\sum_{y=1}^{|B|} max(w_{b_y})}$$

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1	B	D	A		
2	A	B	C	D	G
3	D	B			
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

$$\frac{\frac{2}{3} * |B|}{\sum_{y=1}^{|B|} max(pos(b_y))} = \frac{\frac{2}{3} * 8}{\sum_{y=1}^{8} max(pos(b_y))} = \frac{16}{3} * \frac{1}{2*fib(1) + 3*fib(2) + 2*fib(3) + fib(5)} = \frac{16}{3} * \frac{1}{2*1+3*2+2*3+8} = \frac{16}{3} * \frac{1}{2+6+6+8} = \frac{16}{66} = 0{,}\overline{24}$$

$$\frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} = \frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{22+8-6} = \frac{2}{3} * \frac{6*1}{22+8-6} = \frac{1}{6} = 0{,}1\overline{6}$$

$$\frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} = \frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{24} = \frac{1}{3} * \frac{(1+1)+(2+1)}{24} = \frac{5}{72} \approx 0{,}0694$$

$$M^*_{sta} = 1 - (\frac{16}{66} + \frac{1}{6}) + \frac{5}{72} \approx 0{,}6603535354$$

Knoten	1. Prio	2. Prio	3. Prio	4. Prio	5. Prio
1	B	D	A		
2	A	B	C	D	G
3	D	B			
4	D				
5	D				
6	F	E			
7	E	G	F		
8	G	E			

$$\frac{\frac{2}{3} * |B|}{\sum_{y=1}^{|B|} max(pos(b_y))} = 0{,}\overline{24}$$

$$\frac{2}{3} * \frac{\sum_{i=1}^{|M_{sta}|} pos(b_i, t_j) \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} = \frac{2}{3} * \frac{4*1+2+2*3}{22+8-7} = \frac{2}{3} * \frac{12}{23} = \frac{8}{23} \approx 0{,}3478$$

$$\frac{1}{3} * \frac{\sum_{z=1}^{|B|-|M_{sta}|} 1 + max(pos(b_z)) \text{ mit } b_z \notin M_{sta}}{\sum_{i=1}^{|B|} max(pos(b_i)) + |B| - |M_{sta}|} = \frac{1}{3} * \frac{(1+1)}{23} = \frac{2}{69} \approx 0{,}029$$

$$M^*_{sta} = 1 - (\frac{16}{66} + \frac{8}{23}) + \frac{2}{69} \approx 0{,}4387351779$$

Knoten	A	B	C	D	E	F	G
1	0.6	0.2		0.8			
2	0.4	0.7	0.3	0.5			0.9
3		0.8		0.5			
4				0.7			
5				0.8			
6	0.1				0.9	0.3	
7					0.5	0.6	0.2
8					0.5		0.3

$$\frac{\frac{2}{3} * \sum_{y=1}^{|B|} min(w_{b_y})}{\sum_{y=1}^{|B|} max(w_{b_y})} = \frac{\frac{2}{3} * \sum_{y=1}^{8} min(w_{b_y})}{\sum_{y=1}^{8} max(w_{b_y})} = \frac{\frac{2}{3} * (0{,}2 + 0{,}3 + 0{,}5 + 0{,}7 + 0{,}8 + 0{,}1 + 0{,}2 + 0{,}3}{0{,}8 + 0{,}9 + 0{,}8 + 0{,}7 + 0{,}8 + 0{,}9 + 0{,}6 + 0{,}5} = \frac{2{,}0\overline{6}}{6} = 0{,}3\overline{4}$$

$$\frac{2}{3} * \frac{\sum_{i=1}^{|M_{ide}|} w_{b_i,t_j} \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} = \frac{2}{3} * \frac{\sum_{i=1}^{6} w_{b_i,t_j} \text{ mit } mat_{b_i,t_j}}{6 + 8 - 6} = \frac{2}{3} * \frac{0{,}2 + 0{,}3 + 0{,}5 + 0{,}1 + 0{,}2 + 0{,}5}{6 + 8 - 6} = \frac{2}{3} * \frac{1{,}8}{8} = 0{,}\overline{6}$$

$$\frac{1}{3} * \frac{\sum_{i=1}^{|B|-|M_{ide}|} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} = \frac{1}{3} * \frac{\sum_{i=1}^{8-6} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{6 + 8 - 6} = \frac{1}{3} * \frac{(1 + 0{,}7) + (1 + 0{,}8)}{8} = \frac{1}{3} * \frac{3{,}5}{8} \approx 0{,}1458$$

$$M^*_{ide} = 1 - (0{,}\overline{6} + 0{,}1458) + 0{,}3\overline{4} \approx 0{,}531984444$$

Knoten	A	B	C	D	E	F	G
1	0.6	0.2		0.8			
2	0.4	0.7	0.3	0.5			0.9
3		0.8		0.5			
4				0.7			
5				0.8			
6	0.1				0.9	0.3	
7					0.5	0.6	0.2
8					0.5		0.3

$$\frac{\frac{2}{3} * \sum_{y=1}^{|B|} min(w_{b_y})}{\sum_{y=1}^{|B|} max(w_{b_y})} = 0{,}3\overline{4}$$

$$\frac{2}{3} * \frac{\sum_{i=1}^{|M_{ide}|} w_{b_i,t_j} \text{ mit } mat_{b_i,t_j}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} = \frac{2}{3} * \frac{\sum_{i=1}^{7} w_{b_i,t_j} \text{ mit } mat_{b_i,t_j}}{6+8-7} = \frac{2}{3} * \frac{0{,}6+0{,}3+0{,}8+0{,}7+0{,}1+0{,}2+0{,}5}{6+1} = \frac{2}{3} * \frac{3{,}2}{7} = 0{,}\overline{6}$$

$$\frac{1}{3} * \frac{\sum_{i=1}^{|B|-|M_{ide}|} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{\sum_{i=1}^{|B|} max(w_{b_i}) + |B| - |M_{ide}|} = \frac{1}{3} * \frac{\sum_{i=1}^{8-7} 1 + max(w_{b_z}) \text{ mit } b_z \notin M_{ide}}{6+8-7} = \frac{1}{3} * \frac{(1+0{,}8)}{7} = \frac{1}{3} * \frac{1{,}8}{7} \approx 0{,}2571$$

$$M^*_{ide} = 1 - (0{,}\overline{6} + 0{,}2571) + 0{,}3\overline{4} \approx 0{,}4206777844$$