Algorithmen zur Berechnung optimaler Teameinteilungen unter Fairnessbedingungen

Bachelorarbeit
von

Mathieu Schanz

An der Fakultät für Informatik und Mathematik
Lehrstuhl für Theoretische Informatik

Erstgutachter:	Prof. Dr. Ignaz Rutter
Betreuender Mitarbeiter:	Prof. Dr. Ignaz Rutter

Bearbeitungszeit: 22. November 2021 – 03. März 2022

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Passau, 3. März 2022

Zusammenfassung

Bei einer Teameinteilung will man die teilnehmende Personen in Teams mit verschiedenen Themen unterteilen. Dabei haben die Personen eine ungefähre Vorstellung, mit wem sie ein Team bilden wollen und welche Themen sie bearbeiten möchten. Daraus ergibt sich das Ziel eine faire Lösung für alle zu schaffen. In dieser Arbeit wird sich deshalb beschäftigt, wie man all diese Wünsche algorithmisch in eine optimale Lösung umwandeln kann. Dazu wird analysiert, welche Bedingungen zu einer Fairness führen. Auf dieser Grundlage kann die Komplexität des Problems ermittelt und einen Algorithmus zu diesem Problem entwickelt werden. Dessen Laufzeit und Ergebnisse werden anschließend ausgewertet. Damit wird gezeigt, dass dieses Problem für kleine Werte in akzeptabler Zeit optimal gelöst werden kann und dass ein beachtlicher Großteil der Teilnehmer und Teilnehmerinnen zufrieden mit den Einteilungen sein würden.

Inhaltsverzeichnis

1. Einleitung

Seit der Schule werden Kinder und junge Erwachsene trainiert, mit ihren gleichaltrigen Mitschülerinnen und Mitschülern in Teams Projekte zu bearbeiten. Dadurch wird den Schülerinnen und Schülern die Zusammenarbeit mit ihren Mitmenschen gelehrt, was in vielen Berufen essentiell ist. Diese Projekte können einfache Referate oder auch etwas Praktisches sein, wie ein Vulkan aus Pappmaché zu bauen oder eine Anwendung zu programmieren. Schon hier gibt es eine gewisse Auswahl an verschiedensten Themen. Man möchte auch auf die Wünsche der Schülerinnen und Schüler eingehen, mit wem sie ein Team bilden wollen oder welches Thema sie gerne bearbeiten würden. Auch im Studium und in der Arbeit gibt es derartige Teamprojekte. Das prominenteste Beispiel an der Universität Passau ist wohl das Software Engineering Praktikum, in dem ein Team aus bis zu sechs Studenten eine Software von Grund auf entwickeln.

Also besteht eine Teameinteilung aus Teilnehmerinnen und Teilnehmer, die jeweils bestimmte Themen bearbeiten wollen und andere nicht. Außerdem muss darauf geachtet werden, welche Teilnehmer zusammen in ein Team kommen wollen. Daraus ergibt sich die Frage, ob algorithmisch eine faire Einteilung der Teams und Zuteilung der Themen berechnet werden kann. Eine Lösung, mit der ein Großteil der Personen zufrieden sein würde. Unter anderem muss dazu festgestellt werden, an welchen Metriken man eine faire Teameinteilung misst.

Unter Zunahme eines solchen Algorithmus müssten Lehrende, Dozierende, sowie Teamleiterinnen und Teamleiter potentiell nicht mehr per Hand alle in Teams einteilen, sondern können sich dies automatisiert berechnen lassen. Beispielsweise könnten die Teilnehmerinnen und Teilnehmer über eine Webseite ihre Daten und Wünsche angeben, die der Algorithmus für seine Lösung in Betracht zieht. Bei vielen Teilnehmerinnen und Teilnehmern wird somit auch viel effektiver auf die individuellen Wünsche geachtet, da ein Computer mehr Daten verarbeiten kann. Ein Mensch wäre nicht ohne Weiteres in der Lage aus vielen individuellen Wünschen eine optimale Lösung zu errechnen.

Für die Lesbarkeit werden Teilnehmerinnen und Teilnehmer durch das Wort *Teilnehmer* repräsentiert. Außerdem wird zwischen *Gruppen* und *Teams* unterschieden. Eine *Gruppe* ist ein Zusammenschluss aus Teilnehmern, die gemeinsam ein Projekt bearbeiten wollen. Ein *Team* sind die Teilnehmer, die das Thema am Ende wirklich bearbeiten. Deshalb sind *Gruppen* Teil der Eingabe, während die *Teams* ein Teil des Ergebnisses sind. Die Teilnehmer, die keiner *Gruppe* angehören, werden im Folgenden auch *Einzelpersonen* genannt.

Einen ähnlichen Fall behandelt Katz et. al [BIBD11] mit dem Speed Dating-Problem. Dieses Problem findet im Fall einer Speed Dating-Veranstaltung statt. In so einer Veranstaltung teilen sich die Teilnehmer jeweils in Paaren mit einer weiblichen und einem männlichen Teilnehmer auf, um sich für kurze Zeit kennenzulernen. Pro Runde wird eine neue Kombination an Paaren geschaffen. Da die Zeit der Veranstaltung begrenzt ist, ist jede Kombination selten möglich. Damit die Teilnehmern zufrieden mit der Veranstaltung und ihren Paaren sind, müssen die Teilnehmer Fragebögen zu sich selbst und ihrem Wunschpartner ausfüllen. Auf dieser Grundlage untersucht die Arbeit, wie man diese Kombinationen an Paaren fair berechnet, sodass jeder mit der Zuweisung der Paare und somit mit der Veranstaltung zufrieden ist. Die Arbeit beachtet auch das Ungleichgewicht zwischen den weiblichen und den männlichen Teilnehmern. So müssen manche Teilnehmer Runden aussetzen, weil vom anderen Geschlecht nicht genug Teilnehmer dabei sind. Die Arbeit stellt zudem ein allgemeineres Problem zum Speed Dating-Problem vor, welches seine Teilnehmer nicht in zwei disjunkten Teilmengen unterteilt.

Das nächste Kapitel 2 enthält eine Einführung in die Begriffe, die im Laufe der Arbeit relevant werden. Vor allem die *ganzzahlige, lineare Programmierung* wird beschrieben. Der Hauptteil startet mit dem 3. Kapitel. In diesem wird das Problem modelliert. Im Vordergrund steht, wie die Ein- und Ausgabe aussehen soll und an welchen Kriterien die Fairness gemessen wird. Danach wird in einem eigenen Kapitel 4 die Komplexität des Problems ermittelt. Das folgende Kapitel 5 enthält eine Formalisierung. Dies beschreibt den Aufbau von dem *ganzzahligen, linearen Programm*, welches unser Problem berechnet, und wie das Ergebnis ausgewertet wird. Im 6. Kapitel wird diese Formalisierung implementiert und mit generierten Daten ausgewertet. Das Ende der Arbeit rundet diese mit weiteren Ideen zum Thema im Kapitel 7 ab.

2. Grundlagen

Diese Arbeit benötigt ein wenig Vorwissen über Konzepte der theoretischen Informatik und der mathematischen Optimierung. Hierfür wird in diesem Kapitel unter anderem die `3-Partition` vorgestellt, die zum Aufstellen eines Beweises für die NP-Vollständigkeit verwendet wird. Außerdem wird geklärt, was man unter *ganzzahliger, linearer Programmierung* versteht.

2.1. Wichtige Begriffe

Definition 2.1 (Partition)**.** *Eine* Partition *P über einer Menge S ist eine Menge bestehend aus disjunkten Mengen. Jedes Element aus S ist in genau einer dieser Mengen enthalten.*

Beispiel 2.2. *Ein Beispiel für eine Partition P über der Menge $S = \{1, 2, 3, 4, 5, 6\}$ könnte diese Menge $\{\{1, 5\}, \{3\}, \{2, 4, 6\}\}$ sein.*

Definition 2.3 (Bipartit)**.** *Ein Graph $G = (V, E)$ ist* bipartit*, wenn V aus zwei disjunkte Mengen an Knoten K und L besteht, sodass jede Kante in E einen Knoten aus K mit einem Knoten aus L verbindet.*

Für die Reduktion im Kapitel 4 brauchen wir außerdem ein NP-schweres Problem. Dazu bietet sich das Problem `3-Partition` gut an.

Definition 2.4 (`3-Partition`)**.** *Das Problem `3-Partition` erhält eine Liste $L = [a_1, ..., a_n]$. Diese Liste will man in Listen mit jeweils 3 Elementen aufteilen. Von dieser Aufteilung fordert man, dass die Summe der drei Elemente t in jeder Liste gleich ist. Das Problem ermittelt nun, ob eine solche Aufteilung existiert oder nicht. Dieses Problem ist NP-vollständig [ZFC13].*

Beispiel 2.5. *Ein Beispiel dazu wäre die Liste $L = [4, 3, 3, 1, 1, 2, 5, 7, 1]$. So gibt das Problem `3-Partition` zurück, dass eine solche Aufteilung existiert. Nämlich die drei Listen $[4, 3, 2], [1, 1, 7], [3, 5, 1]$. Bei einem anderen Beispiel mit der Liste $L = [4, 3, 1, 9, 3, 2]$, ergibt aber `3-Partition`, dass keine solche Aufteilung möglich ist.*

2.2. Ganzzahlige, lineare Programmierung

Für die Entwicklung der Algorithmen wird die *ganzzahlige, lineare Programmierung* hinzugezogen. Indem man das Problem in ein *ganzzahliges, lineares Programm* übersetzt, kann man es mit Hilfe von Algorithmen lösen, die für die *ganzzahlige, lineare Programmierung* entwickelt wurden. Die Software Gurobi[1] hat sich beispielsweise darauf spezialisiert, solche Programme auszurechnen.

Definition 2.6. *(Lineare Programmierung) Ein* lineares Programm *erhält zwei Matrizen* $A \in \mathbb{R}^{m \times n}$ *und* $C \in \mathbb{R}^{p \times n}$ *als Eingabe. Zudem bekommt es drei Vektoren* $b \in \mathbb{R}^m$, $d \in \mathbb{R}^p$, $z \in \mathbb{R}^n$. *Das lineare Programm hat eine* Zielfunktion*, die man mit dem Setzen des Vektors* $x \in \mathbb{R}^n$ *maximieren bzw. minimieren will.*

$$\max \quad z^T \; x \tag{2.1}$$

Zudem existiert ein Gleichungssystem, sowie ein Ungleichungssystem, nach denen man auflösen muss.

$$Ax = b \tag{2.2}$$
$$Cx \leq d \tag{2.3}$$

Die Variablen aus A und C heißen Koeffizienten *und die Variablen im Vektor x werden* Entscheidungsvariablen *genannt. Für die die* Entscheidungsvariablen *kann man durch C auch untere und obere Schranken definieren. In dieser Arbeit wird eine Zeile aus den Gleichungssystemen kurz* Zeile *genannt.*

Das Ziel der linearen Programmierung *ist, dass die* Zielfunktion *so weit optimiert wird, dass das Gleichungssystem sowie das Ungleichungssystem erfüllt bleibt [W.92a, H.91].*

Die *ganzzahlige, lineare Programmierung* fordert nun, dass die *Entscheidungsvariablen* im Vektor x nur ganzzahlige Werte annehmen können. Es gilt also, dass $x \in \mathbb{Z}^n$ ist [W.92b].

Eigentlich erlaubt *lineare Programmierung* nur Ungleichungen mit dem Zeichen $\leq$ und die *Entscheidungsvariablen* auf der linken Seite. Die Software Gurobi lässt aber auch Ungleichungen der Form $\sum_{i=1}^{n} a_i \cdot x_i \geq b$ zu. Weil man diese durch die einfache Umformung (2.4) erhalten kann, werden in dieser Arbeit auch *Zeilen* mit einem $\geq$ auftauchen. Dies dient lediglich zum Verständnis der Funktionsweise der *Zeilen*.

$$\sum_{i=1}^{n} a_i \cdot x_i \geq b \quad \Longleftrightarrow \quad \sum_{i=1}^{n} -a_i \cdot x_i \leq -b \tag{2.4}$$

[1] `https://www.gurobi.com/`

Um dieses Konzept zu veranschaulichen wird dazu ein Beispiel formuliert.

Beispiel 2.7. *Es wird die* Zielfunktion *als Minimum definiert.*

$$\min 3x_1 + (-1)x_2$$

Für das Gleichungssystem und das Ungleichungssystem werden die folgenden Zeilen *benutzt.*

$$\begin{array}{rcrcrcr} 2x_1 & + & & & 2x_3 & = & 10 \\ (-1)x_1 & + & 2x_2 & + & x_3 & = & 3 \end{array}$$

$$\begin{array}{rcrcrcr} & & 2x_2 & + & x_3 & \leq & 6 \\ (-2)x_1 & + & x_2 & + & (-1)x_3 & \leq & -2 \end{array}$$

Und hinzu kommen die Schranken für die Entscheidungsvariablen.

$$\begin{array}{rcr} x_1 & \geq & -2 \\ x_2 & \geq & 0 \\ x_2 & \leq & 1 \\ x_3 & \leq & 5 \end{array}$$

Die Eingabe sieht dann, wie folgt, aus. Dabei wurden die unteren Schranken der Form $x \geq y$ in $-x \leq -y$ umgewandelt.

$$A = \begin{pmatrix} 2 & 0 & 2 \\ -1 & 2 & 1 \end{pmatrix},\ b = \begin{pmatrix} 10 \\ 3 \end{pmatrix},\ C = \begin{pmatrix} 0 & 2 & 1 \\ -2 & 1 & -1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},\ d = \begin{pmatrix} 6 \\ -2 \\ 2 \\ 0 \\ 1 \\ 5 \end{pmatrix},\ z = \begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix}$$

Die Ausgabe wird mit der Software Gurobi berechnet. Man erhält die Lösung $x_1 = 1$, $x_2 = 0$ und $x_3 = 4$.

3. Modellierung

Das Problem der fairen Team- und Themeneinteilung nennen wir im Folgenden kurz `FairTeams`. Die Modellierung dieses Problems stellt den Grundstein der Arbeit dar. Erst mit der Definition der Eingabe, der entwickelten Maße zur Gewährleistung der Fairness und der gewünschten Ausgabe kann man darauf Algorithmen und Heuristiken entwickeln. Pro Sektion wird erst beschrieben und argumentiert, welche Variablen, Maßen, Mengen benötigt werden. Darauf aufbauend wird dies in eine mathematische Formulierung umgesetzt.

3.1. Eingabe

Am Anfang erhält `FairTeams` eine Anzahl an verschiedenen Teilnehmern, die beispielsweise über eine Webseite Themen bewerten können. Diese Bewertung könnte so ähnlich umgesetzt werden, wie das beliebte Fünf-Sterne-System, das man aus bekannten Online-Shops und App-Stores kennt. Also haben die Teilnehmer die Möglichkeit, das Thema in fünf verschiedenen Abstufungen zu bewerten. Die einzelnen Stufen der Bewertung werden im Folgenden „sehr uninteressant“, „uninteressant“, „neutral“, „interessant“ und „sehr interessant“ genannt. Damit haben die Teilnehmer genügend Auswahl für die Bewertung der Themen.

Außerdem wird den Teilnehmern die Funktion bereitgestellt, sich in Gruppen zu versammeln. So haben Freunde und Kollegen die Möglichkeit, gemeinsam an den Themen zu arbeiten. Demnach wird eine Menge benötigt, die die einzelnen Gruppen beschreibt.

Nun fehlt noch eine Variable, um die Größe der Teams zu bestimmen. Diese Größe spielt eine große Rolle für die Fairness, weil jeder Teilnehmer in einem Team das Arbeitspensum am Projekt senkt. Jedenfalls in der Theorie. Es ist aber nicht immer möglich, dass alle Teams die gleiche Anzahl an Teilnehmern besitzen. Wenn diese Anzahl nicht durch die der Teams teilbar ist, kann man diese nicht gleich aufteilen. Die einzige Ausnahme wäre, man bildet nur ein einziges Team. Das ist aber selten wünschenswert und dafür bräuchte man keinen Algorithmus. Demzufolge sollten sich die Größen der Teams um eins unterscheiden dürfen. Eine Möglichkeit für diese Variable wäre nun die maximale Größe der Teams anzugeben. Aber so gibt es einige Größen, die nicht möglich wären. Bei 14 Teilnehmern kann man kein Team mit 6 Teilnehmern unter unserer Bedingung bilden. Deshalb müsste man stets überprüfen, ob diese Größe möglich wäre. Wegen dieser Komplikation wird die Anzahl der Teams als Variable für die Größe der Teams benutzt. Aus der kann leicht die minimale und maximale Größe der Teams ermittelt werden. Außerdem kann diese überprüfen, ob es genug Themen zur Auswahl gibt.

Jedes Thema darf von höchstens einem Team belegt werden. Sonst müsste die Modellierung beachten, von wie vielen Teams ein Thema bearbeitet werden darf. Beim Wunsch ein Thema an mehrere Teams zu verteilen, könnte diese Funktion leicht durch eine Erweiterung der Software implementiert werden. Die Software muss dann nur das Thema und dessen Bewertungen für die Eingabe mehrfach kopieren. So oft, wie die Administratorin oder Administrator der Software das Thema vergeben möchte.

Um das Problem `FairTeams` mathematisch zu formulieren, erhält es das Quadrupel (G, q, H, σ) als Eingabe. Der Graph $G = (\text{TE} \dot\cup \text{TH},\ E)$ ist ein bipartiter, gewichteter Graph. Die Menge TE enthält die Teilnehmer und die Menge TH die Themen. Die Gewichtung des Graphen G stellen wir als Abbildung $q : E \to \{-2, -1, 0, 1, 2\}$ dar. Dabei stehen die Werte -2 bis 2 für die verschiedenen Bewertungen, die ein Thema von einem Teilnehmer erhalten kann. So wird die -2 als „sehr uninteressant“, die -1 als „uninteressant“, die 0 als „neutral“, die 1 als „interessant“ und die 2 als „sehr interessant“ interpretiert. Zudem ist von der Eingabe gefordert, dass der Graph G vollständig ist. Sonst würde es zu Schwierigkeiten kommen, wenn später die durchschnittliche Bewertung von Themen berechnet wird.

Die Menge H steht für die Menge der Gruppen und Einzelpersonen. Die Menge ist eine Partition über TE. Außerdem wird die Menge der Gruppen als $H_G = \{S \in H \mid |S| \geq 2\}$ definiert, die später bei der Ausgabe benötigt wird.

Die Variable σ steht für die Anzahl der Teams. Die Bedingung $\sigma \leq |\text{TH}|$ muss erfüllt sein, damit es mindestens so viele Themen gibt wie Teams. Auch muss $\sigma \leq |\text{TE}|$ gelten, denn sonst gibt es Teams ohne Teilnehmer. Aus der Variable kann man nun die Größe der Teams berechnen. Die Formel für die minimale Teamgröße ist $g_{\min} = \lfloor \frac{|\text{TE}|}{\sigma} \rfloor$ und die der maximalen Teamgröße $g_{\max} = \lceil \frac{|\text{TE}|}{\sigma} \rceil$. Des Weiteren wird gefordert, dass kein $S \in H$ mächtiger ist als die maximale Größe der Teams $g_{\max}$. Sonst würde solch eine Gruppe mit Gewissheit aufgeteilt werden.

3.2. Maße zur Aufteilung von Gruppen

Leider ist es nicht immer möglich, jede Gruppe in einem Team aufzunehmen. Zum Beispiel wenn H aus fünf Gruppen mit jeweils vier Teilnehmern besteht, aber die Anzahl der Teams $\sigma = 4$ beträgt. So ist die Größe der Teams auf $g_{\max} = 5 = g_{\min}$ beschränkt. Dann muss mindestens eine Gruppe aufgeteilt werden.

Für solche Fälle werden im Folgenden Beispiele betrachtet, die verschiedene Möglichkeiten zur Aufteilung geben. Aus den besten Aufteilungen der Gruppen werden Maße ausgearbeitet, die ihre Fairness beschreiben können.

Im Beispiel 3.1 besteht Gruppe 1 und 2 aus jeweils vier Teilnehmern und Gruppe 3 und 4 aus fünf. Die vier Gruppen werden auf drei Teams aufgeteilt. Bei der 1. Möglichkeit wird dazu Gruppe 1 in drei Teile zerschlagen. Zwei Teilnehmer aus der ehemaligen Gruppe bleiben zusammen, die anderen Zwei sind jeweils alleine in einem anderen Team. Die 2. Möglichkeit ist schon harmloser. Hier wird Gruppe 3 aufgeteilt. Es entsteht nur eine Person, die allein mit einer anderen Gruppe im Team arbeiten muss. Die dritte Möglichkeit scheint jedoch im Sinne der Teilnehmer die beste Lösung zu sein. Zwar werden dann Gruppe 3 und 4 aufgeteilt, aber jeder Teilnehmer hat immer noch mindestens eine Partnerin oder einen Partner aus der alten Gruppe mit im Team.

Daraus können wir zwei Maße herleiten. Das 1. Maß soll die Anzahl an einzelnen Personen beschreiben, die von ihrer Gruppe getrennt wurden. Das 2. Maß beschreibt hingegen, wie viele Gruppen aufgeteilt werden müssen. Zwar ist in diesem Beispiel das 1. Maß sinnvoller als das 2., jedoch sehen wir das Gegenteil in der nächsten Situation.

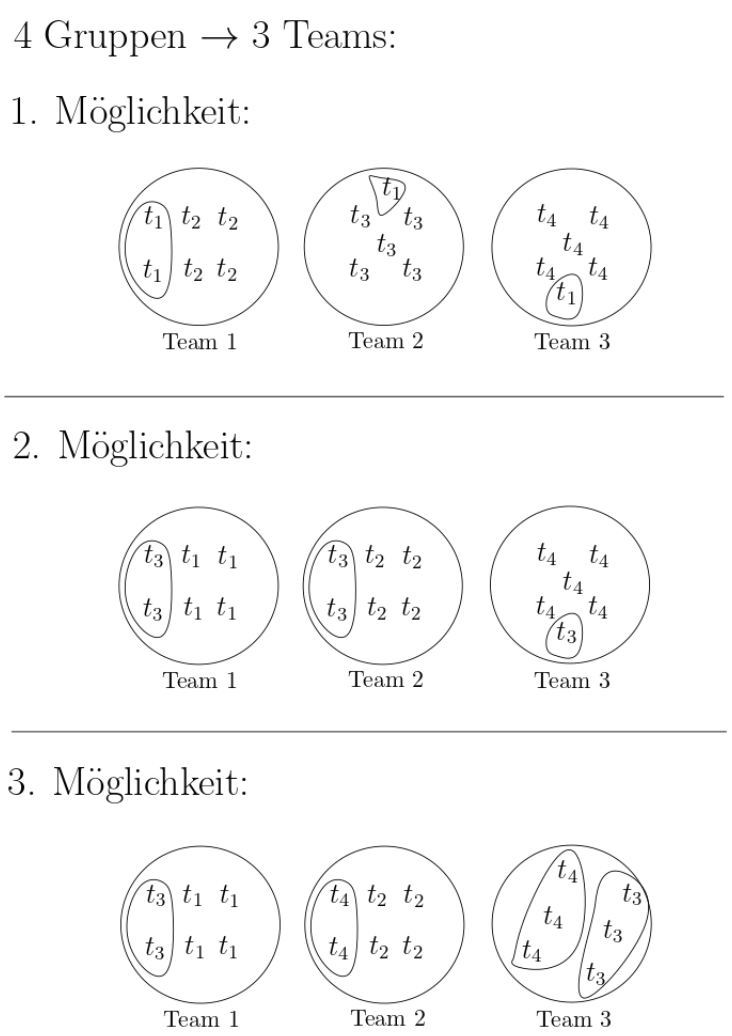

Abbildung 3.1.: Beispiel: 2 Gruppen der Größe 4, 2 Gruppen der Größe 5 → 3 Teams

Im Beispiel 3.2 gibt es unter anderem diese zwei möglichen Aufteilungen. Entweder man teilt eine Gruppe auf, wie in der ersten Möglichkeit. Man verteilt diese vier Teilnehmer. Jeweils ein Teilnehmer wird einem anderen Team zugeteilt. Die anderen Gruppen lässt man bestehen. Die andere Möglichkeit sieht vor, dass man alle Gruppen aufteilt. Das Resultat wäre eine Person weniger, die komplett aus ihrer alten Gruppe isoliert wird. Dafür wäre keiner mit dieser Aufteilung zufrieden. In diesem Fall wäre die Anzahl der Aufteilungen wichtiger als das erstere Maß.

Nun könnte man das ganze noch weiterführen, indem man aufgeteilte Gruppen nach der verbliebenen Größe der Teile zählt. Bei unserem `FairTeams` Problem wird aber von einer verhältnismäßig kleinen Gruppengröße ausgegangen. Deswegen ist für diese Szenarien kein genaueres Maß notwendig.

Aus der Partition H wollen wir nun eine Partition H_T bestimmen. Diese Partition soll die Menge der fertigen Teams darstellen. So muss gelten, dass $\forall T \in H_T : g_{\min} \leq |T| \leq g_{\max}$.

Definition 3.1 (Gruppenverstoß)**.** *Die Anzahl der aufgelösten Gruppen wird als* Gruppenverstoß δ_G *definiert und die Gruppen, die bei dieser Auflösung entstehen, werden* Teilgruppen *genannt.*

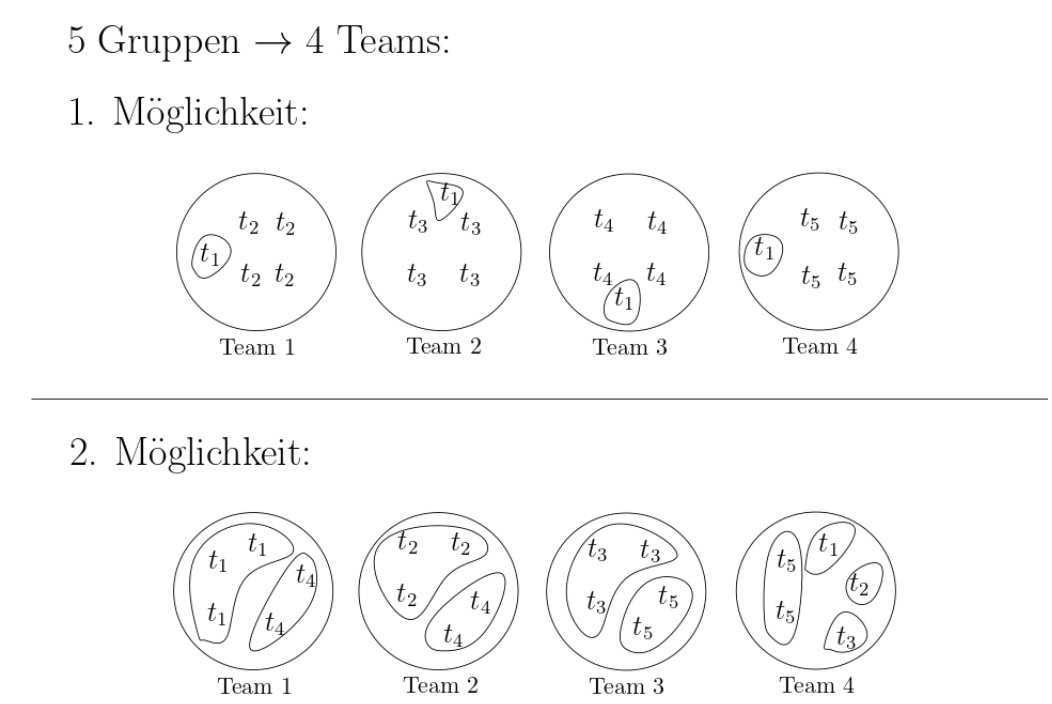

Abbildung 3.2.: Beispiel: 5 Gruppen der Größe 4 → 4 Teams

Definition 3.2 (Teilgruppe)**.** *Wenn eine Gruppe aufgeteilt wird, landen diese Teile in verschiedenen Teams. Diese Teile werden* Teilgruppen *genannt.*

(i) Für eine Teilgruppe $S' \subsetneq S$ *mit* $S \in H$ *gilt, dass genau ein* $T \in H_T$ *existiert mit* $S' \subseteq T$*. Für dieses* T *gilt auch, dass kein* $x \in S \setminus S'$ *existiert mit* $S \cup \{x\} \subseteq T$*.*

(ii) Eine Teilgruppe T *ist* leicht*, wenn* $|T| > 1$ *ist. Die Menge aller leichten* Teilgruppen *wird* leichte Aufteilung A_L *genannt.*

(iii) Interessanter sind jedoch die Teilgruppen *mit der Eigenschaft* $|T| = 1$*. Diese werden* strikt *genannt und die Menge dieser* Teilgruppen *ist die* strikte Aufteilung A_S*.*

Die beschriebenen Maße werden mittels des Beispiels 3.1 in der unteren Aufzählung veranschaulicht.

1. Möglichkeit: Gruppe 1 wird aufgeteilt
   $\implies \delta_G = 1, |A_L| = 1, |A_S| = 2$
2. Möglichkeit: Gruppe 3 wird aufgeteilt
   $\implies \delta_G = 1, |A_L| = 2, |A_S| = 1$
3. Möglichkeit: Gruppen 3 & 4 werden aufgeteilt
   $\implies \delta_G = 2, |A_L| = 4, |A_S| = 0$

Im Beispiel 3.2 würden diese, wie folgt, aussehen.

1. Möglichkeit: Gruppe 1 wird aufgeteilt
   $\implies \delta_G = 1, |A_L| = 0, |A_S| = 4$
2. Möglichkeit: Alle Gruppen werden aufgeteilt
   $\implies \delta_G = 5, |A_L| = 7, |A_S| = 3$

3.3. Maß für Einzelpersonen

Es gibt nun zwei wichtige Maße für Gruppen, aber die Einteilung muss auch fair für Teilnehmer sein, die keiner Gruppe angehören. Auch die Teilnehmer aus *strikten Teilgruppen* müssen dabei beachtet werden. Bei einer Gruppe kann man nämlich davon ausgehen, dass einige der Teilnehmer schon öfter zusammengearbeitet haben. Bei verschiedenen einzelnen Teilnehmern jedoch nicht. So haben Teams, die aus keiner Gruppe oder *leichten Teilgruppen* bestehen, gegenüber diesen Teams einen Nachteil.

Ein sinnvolles Maß könnte dafür sorgen, dass in jedem Team mindestens eine leichte Teilgruppe enthalten ist. Für den Fall $|H_G| < \sigma$ ist dies zwar nicht möglich, aber dieses Maß könnte hier verhindern, dass zwei Gruppen einem einzigen Team zugewiesen wird. Dieses Maß könnte die Anzahl an Teams zählen, die aus keiner Gruppe und keiner *leichten Teilgruppe* bestehen.

Das beschriebene Maß wird γ genannt.
Das Maß wird mit dieser Formel $\gamma = |\{T \in H_T \mid \forall S \in H_G \cup A_L : \neg(S \subseteq T)\}|$ ausgerechnet. Dies ist die Anzahl an Teams, die nur aus Einzelpersonen und *strikten Teilgruppen* besteht. Damit sind wir am Ende der Modellierung für die Bildung der Teams.

3.4. Maße zur Themenvergabe

Es fehlen nur noch Maße, um die Qualität und Fairness der Zuweisungen von Themen zu bewerten. Die Zufriedenheit der zugewiesenen Themen kann an der durchschnittlichen Bewertung festgemacht werden, die das Team seinem Thema gegeben hat. Demnach können wir ein Maß definieren, das aus der Summe der durchschnittlichen Bewertungen der Zuweisungen besteht. Aber dieses Maß ist allein nicht für eine faire Themenzuweisung geeignet. Dazu kann man beispielsweise zwei Teams betrachten. Das 1. Team bewertet Thema 1 mit „sehr interessant“ (2) und Thema 2 mit „neutral“ (0). Das 2. Team hingegen bewertet Thema 1 mit „neutral“ (0) und Thema 2 mit „sehr uninteressant“ (-2). So gibt es die folgenden zwei Fälle.

1. Fall: 1. Team nimmt Thema 2 (Bewertung: 0)
   und 2. Team das Thema 1 (Bewertung: 0)
2. Fall: 1. Team nimmt Thema 1 (Bewertung: 2)
   und 2. Team das Thema 2 (Bewertung: -2)

In beiden Fällen würde die Summe null ergeben. Nur ist der 1. Fall offensichtlich fairer, weil beide Teams diese Zuteilung als „neutral“ betrachten. Beim anderen Fall sieht das ganz anders aus. Das 1. Team ist zwar sehr zufrieden mit seinem Thema, das 2. Team mit seinem dagegen überhaupt nicht.

Deshalb brauchen wir ein besseres Maß, um die Fairness hier zu bestimmen. Dafür bietet sich die Anzahl der Themenzuweisungen an, die auf Unzufriedenheit stoßen. Mit dem durchschnittlichen Wert der Bewertung kommt man dabei auf drei verschiedene Grade. Die Themenzuweisungen, die durchschnittlich schlechter als „uninteressant“, „neutral“ und „interessant“ bewertet wurden. Ein weiterer Grad schlechter als „sehr interessant“ wird dabei nicht benötigt, da alle Themenzuweisungen ab einer durchschnittlichen Bewertung von „interessant“ ein Erfolg sind. Schlechter als „sehr uninteressant“ ist zudem nicht möglich. Sobald diese drei Maße minimal sind, ist das obere Maß aussagekräftiger.

Die Themenvergabe bilden wir als eine Menge Z von Tupeln ab. Für diese Menge soll gelten $Z \subseteq H_T \times \mathrm{TH}$, sodass in der ersten Koordinate ein Team aus H_T steht und in der Zweiten ein Thema aus TH. So bedeutet das Tupel (te, th), dass dem Team te das Thema

th zugeteilt wurde. Es muss zudem gelten, dass kein Team und kein Thema mehrfach in Z vorkommen und jedes Team in der 1. Koordinate der Tupeln von Z enthalten ist.
Die Qualität einer Zuweisung können wir anhand des Durchschnitts berechnen. Dafür können wird die Abbildung $\text{avgVal} : (H_T, \text{TH}) \to [-2, 2], (T, \text{th}) \mapsto \frac{1}{|T|} \cdot \sum_{\text{te} \in T} q((\text{te}, \text{th}))$ nutzen.

Daraus kann nun eine Qualität aller Themenzuweisungen μ bestimmt werden. Dieses Maß μ soll dabei die Summe von den Bewertungen sein, die die Teams ihrem zugewiesenen Thema gegeben haben. So gilt $\mu = \sum_{(T,\text{th}) \in Z} \text{avgVal}(T, \text{th})$ und $\mu \in [-2 \cdot \sigma, 2 \cdot \sigma]$.

Aus dem informellen Teil des Kapitels wissen wir aber, dass dieses Maß nicht viel über die Fairness der Zuteilung aussagt. Vor diesem Hintergrund wurden drei Maße definiert, die darauf zielen, dass alle Teams zufrieden mit ihren Themen sind. Diese heißen „*Themenverstoß des Grades i*“ δ_{T_i} und werden mit der Formel (3.1) berechnet.

$$\delta_{T_i} = |\{(T, \text{th}) \in Z \mid \text{avgVal}(T, \text{th}) < i\}| \text{ für } i \in \{-1, 0, 1\} \tag{3.1}$$

3.5. Ausgabe

Als Ausgabe wird eine Menge gebraucht, die die Teams enthält. Außerdem muss für jedes Team ersichtlich sein, welches Thema ihm zugewiesen wurde. Auf diese Menge kann man dann Bedingungen mittels der Maße definieren.

Die Maße für die Einteilung der Teams priorisieren wir über die der Themenvergabe. Es wird angenommen, dass den Teilnehmern der Zusammenhalt der Gruppe wichtiger ist, als dass ihnen ein interessantes Thema zugewiesen wird. Wenn man eine Gruppe gebildet hat, liegt schließlich der Fokus auf den Personen, mit denen man schon öfter Kontakt hatte und mit denen man ein Projekt starten will. Bei Themen hingegen hat man am Anfang nur eine ungefähre Vorstellung wie interessant die Arbeit daran werden könnte.
Aus dem Beispiel 3.2 können wir darauf schließen, dass mindestens die Hälfte der Gruppen bestehen bleiben soll. Das ist leider in einem Ausnahmefall nicht möglich. Wenn mehr Gruppen mit maximaler Größe existieren, als Teams mit maximaler Größe erlaubt sind, dann müssten in manchen Fällen mehr Gruppen aufgeteilt werden. Dabei muss die Differenz von der Anzahl der Teams mit maximaler Größe und der Gruppen mit maximaler Größe betrachtet werden. Falls das Ergebnis größer als die Hälfte der Gruppen ist, so müssen auch mehr Gruppen aufgeteilt werden als nur die Hälfte.

Auf dieser Bedingung kann man nun aufbauen, dass man die Anzahl der strikten Teilgruppen minimieren will. Das hat den Grund, dass vor allem diese Personen sich am unfairsten behandelt fühlen würden. Diese Personen sind komplett aus ihrem Team ausgeschlossen worden. Mithilfe dieser zwei Bedingungen will man nun allgemein die Anzahl der *Gruppenverstöße* minimieren. Daraufhin kann man noch das Maß für die Einzelpersonen betrachten. Mit diesem Maß kann man fordern, dass in jedem Team eine Gruppe oder leichte Teilgruppe vorhanden ist, solange genügend Gruppen existieren. Dieses Maß macht erst ab einer maximalen Teamgröße von vier Sinn. Wenn ein Team höchstens aus drei Teilnehmern besteht, so kann nur eine Gruppe in das Team aufgenommen werden. Für eine 2. Gruppe wäre nicht genug Platz. Deshalb wird im sonstigen Fall vorausgesetzt, dass γ kleiner oder gleich der Anzahl der Teams ist.

Für die Themenvergabe ist nun wichtig, dass man erst die *Themenverstöße des Grades -1* minimiert, dann die *des Grades 0* und danach die mit *1*. Zuletzt kann man von allen übrig gebliebenen Ergebnissen das mit der höchsten Qualität auswählen.

Zwei weitere Variablen müssen noch definiert werden. Die Variable $h_{\max}$ ist die Anzahl der Gruppen mit maximaler Größe $g_{\max}$.
Sie wird definiert durch $h_{\max} = |\{S \in H_G \mid |S| = g_{\max}\}|$. Die Variable $\sigma_{\max}$ hingegen beschreibt die Anzahl der Teams mit maximaler Größe. Für den Fall $g_{\max} = g_{\min}$ gilt $\sigma = \sigma_{\max}$, da jedes Team maximal groß ist. In sonstigen Fällen kann $\sigma_{\max}$ durch die Formel $\sigma_{\max} = |\text{TE}| \bmod \sigma$ berechnet werden.

Als Ausgabe ist die bereits erstellte Menge Z geeignet, weil sie jedes Team und deren Thema in jeweils einem Tupel enthält. Die beschriebenen Bedingungen muss dann die Menge Z erfüllen.

Die Bedingungen für eine faire Teameinteilung wären dementsprechend die Folgenden.

1. Bedingung: $\delta_G \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor, h_{\max} - \sigma_{\max}\}$
2. Bedingung: Minimaler Wert für $|A_S|$ von allen möglichen Ergebnissen, die die obere Bedingung erfüllen
3. Bedingung: Minimaler Wert für δ_G von allen möglichen Ergebnissen, die die vorherigen Bedingungen erfüllen
4. Bedingung: $\gamma \leq \begin{cases} \max\{\sigma - |H_G|, 0\}, \text{ falls } g_{\max} \geq 4 \\ \sigma, \text{ sonst} \end{cases}$

Danach fehlen noch die Bedingungen für eine faire Themenvergabe.

5. a) Bedingung: Minimaler Wert für $\delta_{T_{-1}}$ von allen möglichen Ergebnissen, die die vorherigen Bedingungen erfüllen

   b) Bedingung: Minimaler Wert für δ_{T_0} von allen möglichen Ergebnissen, die die vorherigen Bedingungen erfüllen

   c) Bedingung: Minimaler Wert für δ_{T_1} von allen möglichen Ergebnissen, die die vorherigen Bedingungen erfüllen

6. Bedingung: Maximaler Wert für μ von allen möglichen Ergebnissen, die die vorherigen Bedingungen erfüllen

3.6. Zusammenfassung

Die einzelnen Variablen sind in der Tabelle 3.1 aufgelistet. Das Problem `FairTeams` erhält die Eingabe (G, q, H, σ). Daraus sollen die Teams gebildet werden, die in der Menge H_T gespeichert sind. Diese Menge ist eine Partition über TE und jede Menge hat eine Mächtigkeit von entweder $g_{\min}$ oder $g_{\max}$. Die Mächtigkeit von H_T beträgt σ. Auf dieser Menge wurden vier Bedingungen definiert. Die Erste und die Dritte enthalten die Variable des *Gruppenverstoßes*. Diese zählt die Anzahl an Gruppen aus der Menge H, die nicht gemeinsam in einem Team sind. Die 2. Bedingung will die Anzahl jener Personen reduzieren, die Teil einer Gruppe waren und mit keinem aus dieser Gruppe ein Team teilen. Die Letzte von den Vier will erreichen, dass möglichst wenige Teams ohne eine Gruppe existieren.

Für die Zuteilung der Themen wurde H_T in Z erweitert. Mit Z hat man ein Tupel mit einem Team in der 1. und einem Thema in der 2. Koordinate. Die Bedingungen für die Zuteilung sind, dass die *Themenverstöße* minimiert werden. Ein *Themenverstoß des Grades* l tritt auf, wenn das zugewiesene Thema von seinem Team durchschnittlich unter dem Wert von l bewertet wurde. Die Variable l steht für die drei Werte -1, 0 und 1. Am Ende will man noch von allen restlichen Ergebnissen das beste Ergebnis erhalten. Dazu wird die Qualität der Zuweisungen μ maximiert.

Variable	*Bedeutung*
TE	Menge der Teilnehmer
TH	Menge der Themen
G	Bipartiter, vollständiger Graph mit den Mengen TE und TH als Knoten
$q(e)$	Gewicht der Kante e aus dem Graphen G
H	Menge der Gruppen und Einzelpersonen
H_G	Menge der Gruppen
H_T	Menge der Teams
Z	Menge der Einteilung von Team und Thema
$g_{\min}$	Minimale Größe der Teams
$g_{\max}$	Maximale Größe der Teams
σ	Anzahl der Teams
$\sigma_{\max}$	Anzahl der Teams mit maximaler Größe
$h_{\max}$	Anzahl der Gruppen mit maximaler Größe
A_S	Menge der Teilgruppen, die genau aus einem Teilnehmer bestehen
A_L	Menge der Teilgruppen, die aus mehreren Teilnehmern bestehen
δ_G	*Gruppenverstoß*, Anzahl der aufgeteilten Gruppen
γ	Anzahl der Teams, die nur aus Einzelpersonen und strikten Teilgruppen bestehen
μ	Qualität der Vergabe von den Themen
avgVal(T, th)	Durchschnittliche Bewertung des Themas th vom Team T
δ_{T_i}	*Themenverstoß des Grades i*, Anzahl der Themen für die avgVal(T, th) $< i$ gilt

Tabelle 3.1.: Definierte Variablen, Maße, Mengen und Abbildungen

4. Komplexität

Nachdem dieses Problem modelliert ist, kann gezeigt werden, dass `FairTeams` NP-vollständig ist. Dazu wird das Problem noch einmal kompakt definiert. Danach wird der Beweis mittels einer Reduktion ausgeführt. Für die Reduktion wird das Problem `3-Partition` verwendet, das im 2. Kapitel erklärt wird.

4.1. FairTeams als Entscheidungsproblem

Mit der Eingabe eines bipartiten, vollständigen Graphen G, der Gewichtung q, der Partition H über der Menge TE und der Variable σ wird eine Team- und Themenzuteilung Z berechnet, die folgende Fairnessbedingungen erfüllt:

1. $\delta_G \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor, h_{\max} - \sigma_{\max}\}$
2. $|A_S|$ ist minimal
3. δ_G ist minimal
4. $\gamma \leq \begin{cases} \max\{\sigma - |H_G|, 0\}, \text{ falls } g_{\max} \geq 4 \\ \sigma, \text{ sonst} \end{cases}$
5. a) $\delta_{T_{-1}}$ ist minimal
   b) δ_{T_0} ist minimal
   c) δ_{T_1} ist minimal
6. μ ist maximal

Anhand der *Zeilen* 2, 3, 5a), 5b), 5c) und 6 sieht man, dass `FairTeams` ein Optimierungsproblem ist. Aus der Theoretischen Informatik wissen wir, dass ein Optimierungsproblem meistens NP-vollständig ist, wenn das dazugehörige Entscheidungsproblem NP-vollständig ist. Deshalb wird `FairTeams` im Folgenden in ein Entscheidungsproblem umgewandelt. Dazu werden sechs Variablen $k_1, k_2, k_3, k_4, k_5, k_6 \in \mathbb{N}_0$ definiert und mit diesen die Bedingungen angepasst. Nun gelten die folgenden Bedingungen.

1. $\delta_G \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor, h_{\max} - \sigma_{\max}\}$
2. $|A_S| \leq k_1$
3. $\delta_G \leq k_2$
4. $\gamma \leq \begin{cases} \max\{\sigma - |H_G|, 0\}, \text{ falls } g_{\max} \geq 4 \\ \sigma, \text{ sonst} \end{cases}$
5. a) $\delta_{T_{-1}} \leq k_3$
   b) $\delta_{T_0} \leq k_4$
   c) $\delta_{T_1} \leq k_5$
6. $\mu \geq k_6$

4.2. NP-Vollständigkeit

Nachdem das Optimierungsproblem `FairTeams` kompakt beschrieben und zu einem Entscheidungsproblem umgewandelt wurde, kann folgendes Lemma endlich bewiesen werden.

Lemma 4.1. *`FairTeams` ist NP-vollständig.*

Der Beweis wird in zwei weitere Lemmas in den nächsten Sektionen unterteilt.

4.2.1. Existenz in NP

Lemma 4.2. *`FairTeams` liegt in NP*

Beweis. Rate eine Belegung von Z. Seien $k_1, k_2, k_3, k_4, k_5, k_6 \in \mathbb{N}_0$. Zudem sei $n = |TE|$, um die Laufzeit zu berechnen.

1. Initialisiere einen Zähler $z \coloneqq 0$. Die Menge Z kann in die Menge H_T umgewandelt werden. Das wird gemacht, indem man Tupel von Z nimmt. In der 1. Koordinate ist das Team gespeichert. Nun vereinigt man diese Teams in eine Menge. Diese Menge ist äquivalent zu H_T.
   Nun wird durch die Menge H iteriert. Für jede Menge $A \in H$, vergleiche, ob in H_T eine Menge T liegt, die alle Elemente aus A enthält. Dazu iteriert man durch jede Menge und schaut, ob darin das 1. Element aus A enthalten ist. Wenn man dieses in einer Menge gefunden hat, wird verglichen, ob jedes andere Element auch in dieser Menge ist.
   Ist es das nicht, so inkrementiert man den Zähler mit $z \coloneqq z + 1$. Im anderen Fall passiert nichts. In beiden Fällen geht man weiter zur nächsten Menge aus H.
   Auch wenn in H Mengen mit Einzelpersonen vorhanden sind, so sind diese immer ganz in einer Menge vorhanden. Man kann eine einzelne Person schließlich nicht aufteilen. Deshalb erhöht sich bei jenen nie der Zähler. Nur bei Gruppen, die aufgeteilt wurden.
   Nachdem man alle Mengen aus H überprüft hat, müssen $\frac{|H_G|}{2}, h_{\max}$ und $\sigma_{\max}$ berechnet werden. Die Lösung zu $\max\{\frac{|H_G|}{2}, h_{\max} - \sigma_{\max}\}$ wird dann mit z verglichen. Falls z größer als das Ergebnis ist, muss abgebrochen werden. Das Abbrechen bedeutet hier, sowie in den folgenden Schritten, dass die Belegung die Bedingung nicht erfüllt. Dies kann in $O(n^2)$ Laufzeit berechnet werden, weil für jede Menge aus H alle Elemente aus den Mengen von H_T durchlaufen werden müssen. Beides sind Partitionen über |TE|. Daraus resultiert diese Laufzeit.

2. Ein Zähler $z \coloneqq 0$ wird initialisiert und wie im obigen Schritt, wird Z in H_T umgewandelt.
   Es wird über die Mengen von H iteriert. Jede dieser Mengen wird in diesen Abschnitt M genannt. Für jedes Element aus F wird in H_T die Menge gesucht, die dieses Element enthält. Da H_T eine Partition ist, muss genau eine solche Menge existieren. Sobald diese Menge gefunden ist, wird nachgesehen, ob ein anderes Element aus M enthalten ist.
   Falls nicht, wird der Zähler mit $z \coloneqq z + 1$ aktualisiert und das Element wird in der Menge A_S für den 4. Schritt abgespeichert. Für beide Fälle geht es weiter zum nächsten Element oder zur nächsten Menge.
   Nun wird z mit k_1 verglichen. Falls z größer ist, muss abgebrochen werden.
   Auch dieser Schritt ist in $O(n^3)$ Laufzeit möglich. Man läuft hier durch die Partition H, die n verschiedene Elemente in Mengen aufgeteilt hat. Für jedes Element durchläuft man die n Elemente aus den Mengen der Partition H_T, bis man die Menge mit dem Element gefunden hat. Danach wird für jedes Element aus M verglichen, ob es auch in der Menge enthalten ist.

3. Ein Zähler $z \coloneqq 0$ wird abermals initialisiert. Es wird der selbe Ablauf wie im 1. Schritt gemacht. Danach wird abermals z der Variable k_2 gegenübergestellt. Falls z größer ist, wird abgebrochen. Der Ablauf ist auch in $O(n^2)$ Zeit möglich.

4. Der Zähler mit $z \coloneqq 0$ wird initialisiert. Aus H werden die Einzelpersonen $H \setminus H_G$ berechnet. Die Menge A_S wurde schon im 2. Schritt bestimmt. Auch hier wird wieder H_T aus Z gefiltert.
   Es wird über die Mengen aus H_T iteriert. Diese Mengen werden wieder F genannt. Jedes Element aus F wird mit den Elementen aus $H \setminus H_G$ und A_S verglichen. Kommt jedes Element in einer der zwei Mengen vor, so wird der Zähler mit $z \coloneqq z + 1$ angepasst.
   Am Ende wird der Zähler z mit dem Ergebnis von $\max\{\sigma - |H_G|, 0\}$ verglichen. Wenn z größer ist, muss abgebrochen werden.
   Auch dieser Schritt benötigt höchstens $O(n^2)$ Zeit. Es wird über H_T iteriert. Somit muss für höchstens n Elemente überprüft werden, ob diese in $(H \setminus H_G) \cup A_S$ liegen. Diese Menge hat auch höchstens n Elemente.

5. Nun werden ganze drei Zähler z_{-1}, z_0, z_1 jeweils mit 0 initialisiert.
   Diesmal geht es über die Tupel (h, th) von Z. Dabei wird $\mathrm{avgVal}(h, \mathrm{th})$ berechnet. Falls $\mathrm{avgVal}(h, \mathrm{th}) < 1$ ist, so wird jeweils eine 1 auf die Zähler z_{-1}, z_0, z_1 addiert. Falls das Ergebnis nur kleiner als 0 ist, so wird jeweils eine 1 auf die Zähler z_{-1}, z_0 addiert. Nun wenn das Ergebnis kleiner als -1 ist, dann wird nur $z_{-1} \coloneqq z_{-1} + 1$ berechnet. Am Ende muss nur noch überprüft werden, ob $z_{-1} \leq k_3$, $z_0 \leq k_4$ und $z_1 \leq k_5$ ist. Falls eine dieser Bedingungen nicht erfüllt ist, wird hier abgebrochen.
   Da insgesamt in allen Teams h jeder Teilnehmer genau einmal vorkommt, ist diese Prozedur in $O(n)$ Zeit berechenbar.

6. Im letzten Schritt wird auch ein Zähler initialisiert. Hier ist es $\mu \coloneqq 0$.
   Es wird wieder über die Tupel (h, th) von Z iteriert und dabei $\mathrm{avgVal}(h, \mathrm{th})$ berechnet. Das Ergebnis wird auf μ addiert. Am Ende muss überprüft werden, ob $\mu \geq k_6$ ist. Falls ja, wird die Belegung akzeptiert. Im anderen Fall wird abgebrochen und die Belegung wird abgelehnt.
   So wie der 5. Schritt, ist diese Prozedur auch in $O(n)$ Laufzeit abgehandelt.

Alle Bedingungen können somit in polynomieller Zeit überprüft werden. Nämlich in $O(n^3)$, da von den sechs Schritten der Zweite solange braucht. Die anderen laufen schneller ab. Damit ist gezeigt, dass `FairTeams` in NP liegt. □

4.2.2. Reduktion

Lemma 4.3. *Das Problem* `FairTeams` *ist NP-schwer.*

Beweis. Man kann `3-Partition` auf `FairTeams` reduzieren. Dieser Vorgang der Reduktion wird im Folgenden beschrieben und durch ein Beispiel veranschaulicht.

Sei eine Instanz I eine Liste aus natürlichen Zahlen $[a_1, ..., a_m]$.
Sei $m \in \mathbb{N}$. Wähle $t = \frac{3}{m} \cdot \sum_{i=1}^{m} a_i$.
Für die Reduktion, wähle $\text{TE} = \{1, ..., m\}$ für die Menge der Teilnehmer. Die Teilnehmer symbolisieren hier die Stellen von der Instanz I.
Außerdem sei $H = \{\{i\} \mid i \in \text{TE}\}$. So existieren nur Einzelpersonen dieser Instanz von `FairTeams`.
Setze die Anzahl der Teams $\sigma = \lfloor \frac{m}{3} \rfloor$. So gilt $3 \leq g_{\max} \leq 4$ und $g_{\min} = 3$.
Sei nun die Menge der Themen $\text{TH} = \{(i, j, k) \mid a_i, a_j, a_k \in I,\ \ i, < j, < k,\ \ a_i + a_j + a_k = t\}$.
Falls $|\text{TH}| < \sigma$, so füge noch $\sigma - |\text{TH}|$ Dummyelemente in TH hinzu. Diese Dummys werden als Tripel $(-i, -i, -i)$ für alle $i \in \{1, ..., \sigma - |\text{TE}|\}$ notiert.
Nun setze für jeden Teilnehmer $\text{te} \in \text{TE}$ und jedes Thema $\text{th} = (\text{th}_1, \text{th}_2, \text{th}_3) \in \text{TH}$ $q((\text{te}, \text{th})) = 1$, falls $\text{te} = \text{th}_1 \vee \text{te} = \text{th}_2 \vee \text{te} = \text{th}_3$ gilt. Sonst setze $q((\text{te}, \text{th})) = -2$. Erstelle daraus den bipartiten, gewichteten und vollständigen Graphen G.

Damit hat man die Eingabe für `FairTeams` geschaffen. So übergibt man dem Problem `FairTeams` die Eingabe (G, q, H, σ) und man erhält als Ausgabe Z. Nun wäre eine JA-Instanz, wenn für die Themenzuweisung Z $\delta_{T_1} = 0$ gilt. Außerdem kann man die Teilnehmer der Teams durch die Variablen aus I ersetzen. So wäre jedes Team eine Liste aus drei Variablen, deren Summe t ergibt. Falls aber $\delta_{T_1} \neq 0$ ist, so ist Z eine NEIN-Instanz und keine 3-Partition ist möglich. □

Beispiel 4.4. *Dieses kleine Beispiel soll zum Verständnis der Reduktion helfen. Dazu wird die Instanz* $I = [7, 3, 6, 7, 5, 2]$ *genutzt.*
Für die Variable t *gilt* $t = \frac{3}{m} \cdot \sum_{i=1}^{m} a_i = \frac{3}{6} \cdot (7 + 3 + 6 + 7 + 5 + 2) = \frac{1}{2} \cdot 30 = 15$.
So erstellt die Reduktion die Teilnehmer $\text{TE} = \{1, 2, 3, 4, 5, 6\}$,
$H = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$ *und* $\sigma = \lfloor \frac{m}{3} \rfloor = \lfloor \frac{6}{3} \rfloor = 2$.
Außerdem werden die Themen $\text{TH} = \{(1, 2, 5), (1, 3, 6), (2, 4, 5)\}$ *erstellt. Die Gewichtung* q *wird in der Tabelle 4.1 veranschaulicht. Aus TE, TH und den Kanten definiert in* q *wird der Graph* G *erstellt.*

	Thema (1, 2, 5)	*Thema (1, 3, 6)*	*Thema (2, 4, 5)*
Teilnehmer 1	1	1	−2
Teilnehmer 2	1	−2	1
Teilnehmer 3	−2	1	−2
Teilnehmer 4	−2	−2	1
Teilnehmer 5	1	−2	1
Teilnehmer 6	−2	1	−2

Tabelle 4.1.: Gewichtung q

Nun übergibt man (G, q, H, σ) *dem Problem* `FairTeams`. *Das Entscheidungsproblem* `FairTeams` *berechnet eine Einteilung der Teams und eine Vergabe der Themen, sodass* $\delta_{T_1} = 0$ *gilt. Das Ergebnis ist, dass das Thema* $(1, 3, 6)$ *dem Team* $\{1, 3, 6\}$ *und das Thema* $(2, 4, 5)$ *dem Team* $\{2, 4, 5\}$ *zugewiesen wird. Dies wird mittels der Menge* $Z = \{(\{1, 3, 6\}, (1, 3, 6)), (\{2, 4, 5\}, (2, 4, 5))\}$ *übergeben. Für das Maß gilt dann auch* $\delta_{T_1} = 0$, *weil* $\text{avgVal}(\{1, 3, 6\}, (1, 3, 6)) = \frac{1}{3} \cdot (1 +$

$1+1)=1$ *und* $\text{avgVal}(\{2,4,5\},(2,4,5)) = \frac{1}{3}\cdot(1+1+1)=1$ *gilt.*
Andere Zuweisungen würden dazuführen, dass $\delta_{T_1} > 0$ *ist. Würde zum Beispiel das Thema* $(1,2,5)$ *dem Team* $\{1,2,5\}$ *zugewiesen, so hätte man für Team* $\{3,4,6\}$ *nur die Auswahl zwischen Thema* $(1,3,6)$ *und* $(2,4,5)$*. Bei beiden gilt* $\text{avgVal}(\{3,4,6\},(1,3,6)) = \frac{1}{3}\cdot(1-2+1) = 0 < 1$, $\text{avgVal}(\{3,4,6\},(2,4,5)) = \frac{1}{3}\cdot(-2+1-2) = -1 < 1$*. Es resultiert, dass* $\delta_{T_1} = 1$ *ist und nicht gleich 0. Für weitere Zuweisungen, beispielsweise* $(1,2,5)$ *zu* $\{1,3,6\}$ *und* $(2,4,5)$ *zu* $\{2,4,5\}$*, gilt das ebenfalls. Man sieht auch in der Tabelle, dass nur Thema* $(1,3,6)$ *und* $(2,4,5)$ *komplementär zueinander sind. Deswegen ist nur die Kombination mit beiden möglich und den Teilnehmern, die diese mit 1 gewählt haben.*
Daraus folgt, dass `FairTeams` *eine JA-Instanz berechnet hat. Also existiert eine 3-Partition, die so* $\{1,3,6\} = [7,6,2]$ *und* $\{2,4,5\} = [3,7,5]$ *aussehen würde.*

Lemma 4.5. *Die Reduktion* `3-Partition` $\leq_p$ `FairTeams` *ist korrekt.*

Beweis. Zunächst wird gezeigt, dass die Reduktion in polynomieller Zeit erstellt werden kann. Sei dazu $n = |\text{TE}|$. Die Variablen TE und H können in $O(n)$ erstellt werden, da nur der Index von der Instanz I in TE und H kopiert wird. Die Anzahl der Teams σ kann sogar in konstanter Zeit berechnet werden. Das Erstellen der Themen ist auch in polynomieller Zeit möglich. Genauer gesagt, braucht man für die Themen $\binom{n}{3} = \frac{n!}{3!\cdot(n-3)!} = \frac{n\cdot(n-1)\cdot(n-2)}{6} < n^3 = O(n^3)$ Zeit. Da man im Worst-case $O(n^3)$ Themen hat, benötigt die Erstellung der Abbildung q $O(\text{n})\cdot O(n^3) = O(n^4)$ Zeit. Das liegt daran, dass n Teilnehmer alle Themen bewerten müssen. Dementsprechend ist die Erstellung der Reduktion in polynomieller Zeit möglich.

Nun muss bewiesen werden, dass eine Lösung mit $\delta_{T_1} = 0$ immer für eine JA-Instanz in `3-Partition` steht. Genauso muss bewiesen werden, dass im sonstigen Fall immer eine NEIN-Instanz vorliegt. Dies wird kurz als die Äquivalenz `FairTeams` $\iff$ `3-Partition` geschrieben, die im nächsten Abschnitt gezeigt wird.

- `FairTeams` $\implies$ `3-Partition`
  Wenn das Entscheidungsproblem für `FairTeams` erfüllt ist, gilt $\delta_{T_1} = 0$. Dieser Wert bedeutet, dass jedes zugewiesene Thema von dem Team durchschnittlich mit 1 oder besser bewertet wurde. Wegen der *Zeile* $\text{te} = \text{th}_1 \vee \text{te} = \text{th}_2 \vee \text{te} = \text{th}_3$ bewerten nur jeweils drei Teilnehmer ein Thema mit 1. Keiner gibt einem Thema eine 2. Die restlichen Themen werden mit -2 bewertet. So muss jedes Thema (i,j,k) zu einem Team $\{i,j,k\}$ zugewiesen werden, um $\delta_{T_1} = 0$ zu erreichen. Die Teams sind zudem disjunkt. Wenn man nun die Teams $\{i,j,k\}$ in die Listen $[a_i,a_j,a_k]$ umwandelt, so hat man $\frac{m}{3}$ disjunkte Mengen. Die *Zeile* für die Themen $a_i+a_j+a_k = t$ ergibt zudem, dass die Summe der Elemente in jeder Liste gleich t ist. Somit folgt, dass bei $\delta_{T_1} = 0$ immer eine JA-Instanz für `3-Partition` heraus kommt.

  Für den Fall, dass $t \notin \mathbb{N}$, gibt die Reduktion immer eine NEIN-Instanz aus. In diesem Fall stehen nämlich nur Dummythemen zur Auswahl. Das folgt aus der Instanz I, deren Liste nur aus natürlichen Zahlen besteht. So gilt für die Summe $a_i+a_j+a_k \in \mathbb{N}$. Demnach ist diese Bedingung $a_i+a_j+a_k = t$ zur Erstellung der Themen nie erfüllt. Die Dummys werden zudem von allen Teilnehmern mit -2 bewertet. So folgt, dass $\delta_{T_1} = \sigma \neq 0$ ist.

  Für den Fall, dass m nicht durch 3 teilbar ist, kommt bei der Reduktion auch immer eine NEIN-Instanz raus. So ist $g_{\max} = 4$ und es gibt mindestens ein Team mit 4 Teilnehmern. Ein Thema kann jedoch nur von 3 Teilnehmern mit 1 bewertet werden. Das ergibt die *Zeile* $\text{te} = \text{th}_1 \vee \text{te} = \text{th}_2 \vee \text{te} = \text{th}_3$ bei der Gewichtung q. Der Rest bewertet das Thema mit -2. Daraus folgt für das zugewiesene Thema für ein 4er

Team, dass die durchschnittliche Bewertung höchstens $\frac{1+1+1-2}{4} = \frac{1}{4} < 1$ ist. Demnach muss in diesem Fall $\delta_{T_1} > 0$ gelten und die Reduktion gibt eine NEIN-Instanz aus.

- `FairTeams` $\Longleftarrow$ `3-Partition`
  Nun setzt man voraus, dass `3-Partition` eine JA-Instanz enthält. So existieren $\frac{m}{3}$ Listen mit jeweils 3 Elementen. Die Summe der 3 Elementen ist gleich t. Deshalb existieren für jede dieser Listen $[a_i, a_j, a_k]$ ein Tripel (i, j, k) in den Themen. Für dieses Tripel gibt es drei Teilnehmer i, j und k, die es mit 1 bewerten. Der Rest bewertet es mit -2. Weil es $\frac{m}{3}$ dieser Listen gibt, existieren $\frac{m}{3}$ Themen. Da diese Listen disjunkt sind, sind auch die einzelnen Mengen an Teilnehmer disjunkt, die ein bestimmtes Thema mit 1 bewertet haben. Das Problem `FairTeams` hat auch genauso viele Teilnehmer, wie die Liste aus `3-Partition` Elemente besitzt. Wenn jedem dieser Mengen das Thema übergeben wird, für welches die drei Teilnehmer mit „interessant“ bewertet haben, so hat man am Ende eine Themenzuweisung Z mit $\delta_{T_1} = 0$. Schließlich gilt für jedes dieser Themen $\text{avgVal}(T, \text{th}) = \frac{1+1+1}{3} = 1$.

Aus der Korrektheit der Reduktion folgt, dass ein NP-schweres Problem auf `FairTeams` reduziert werden kann. Demzufolge ist `FairTeams` auch NP-schwer und weil `FairTeams` in NP liegt, ist unser Problem NP-vollständig. □

Eine große Schwäche dieser Reduktion ist das Hinzufügen von den Themen. Es können bis zu $\binom{n}{3} = O(n^3)$ Themen erstellt werden. Also die NP-Vollständigkeit hängt bei dieser Reduktion von einer hohen Anzahl an Themen ab. Interessant wäre hingegen, ob das Problem NP-vollständig mit einer konstanten, kleinen Anzahl an Themen ist. Dies würde eher der Realität entsprechen.

5. Formalisierung als ganzzahliges Programm

Die Modellierung ermöglicht nun die Erstellung einer Formalisierung für `FairTeams`. Dies ist eine Beschreibung eines *ganzzahligen, linearen Programms*, welches unser Problem löst. Mit diesem Programm kann dann ein Ergebnis berechnet werden. Dies geschieht mittels optimierter Algorithmen, die bereits für *ganzzahlige, lineare Programmierung* entwickelt wurden.

5.1. Funktionsweise

Diese Formalisierung wird in 3 Schritten durchgeführt. Im 1. Schritt werden die Variablen aus der Eingabe berechnet, die für das weitere Vorgehen benötigt werden. Im 2. Schritt werden alle *Zeilen* und Variablen definiert, ihr Nutzen erklärt und die Korrektheit bestimmt. Im letzten Schritt wird die Ausgabe aus den definierten *Entscheidungsvariablen* herausgezogen.

Im *1. Schritt* bekommt man die Eingabe des Problems `FairTeams`. Daraus lassen sich sofort $g_{\min}$ und $g_{\max}$ berechnen. Auch die Menge H_G lässt sich bestimmen, sowie die Variablen $\sigma_{\max}$ und $h_{\max}$.

Nachdem alle nötigen Variablen berechnet wurden, können wir uns im *2. Schritt* dem Programm widmen. Die *Zielfunktion* wird am Ende definiert. Zunächst wird sich auf den Aufbau des Gleichungssystems konzentriert. Die dazu definierten *Entscheidungsvariablen* und neue *Koeffizienten* werden noch einmal in der Tabelle 5.1 aufgelistet.

Die ersten und wichtigsten *Entscheidungsvariablen* sind $x_{(i,\ j)}$. Diese stehen dafür, ob ein Thema i dem Teilnehmer j zugewiesen wurde. Es ist gefordert, dass jeder Teilnehmer nur genau einem Team zugewiesen wird. Jedes Team bearbeitet genau ein Thema. Um diesen Sachverhalt in ein Programm zu übersetzen, gibt es die 1. *Zeile*. Diese sagt aus, dass ein Teilnehmer genau ein Thema erhält.

$$\sum_{i=1}^{|\mathrm{TH}|} x_{(i,\ j)} = 1, \quad j \in \{1, ..., |\mathrm{TE}|\} \tag{5.1}$$

$$x_{(i,\ j)} \in \{0, 1\} \tag{5.2}$$

Bei einem Thema i gibt es gewöhnlicherweise mehr als eine Person, der es zugewiesen wird. Aus diesen Teilnehmern wird dann das Team gebildet. Ein Team muss immer aus mindestens $g_{\min}$ und höchstens $g_{\max}$ Teilnehmern bestehen. Aber ein Thema kann auch keinem Team zugeteilt werden. Für diese beiden Fälle führen wir die Entscheidungsvariablen t_i und s_i ein. Die Variable t_i enthält die Information über die Größe des Teams, welches Thema i bearbeitet.

$$t_i = \begin{cases} 1, \text{im Team mit dem Thema i sind } g_{\min} \text{ Teilnehmer, } \mathrm{g_{min}} \neq \mathrm{g_{max}} \\ 0, \text{sonst} \end{cases} \tag{5.3}$$

Auf der anderen Seite beschreibt s_i, ob das Thema überhaupt einem Team zugewiesen wird.

$$s_i = \begin{cases} 0, \text{Thema } i \text{ wird einem Team zugewiesen} \\ 1, \text{sonst} \end{cases} \tag{5.4}$$

Diese beiden Variablen werden im Folgenden für paar *Zeilen* vom Gleichungssystems noch relevant. Beispielsweise dienen sie späteren Berechnungen, wie für die *Zeilen* zur Bestimmung der *Themenverstöße*.

$$\sum_{j=1}^{|\mathrm{TE}|} x_{(i,\ j)} \ + \ g_{\max} \cdot s_i \ + \ t_i = g_{\max}, \quad i \in \{1, ..., |\mathrm{TH}|\} \tag{5.5}$$

$$t_i \leq g_{\max} - g_{\min} \tag{5.6}$$

$$s_i \in \{0, 1\} \tag{5.7}$$

$$t_i \in \{0, 1\} \tag{5.8}$$

Die *Zeile* 5.5 erfüllt nun die drei Funktionen.

1. Sie setzt den Wert für s_i. Aus der Gleichung geht hervor, dass $s_i = 0$ sein muss, wenn mindestens ein $x_{(i,\ j)}$ den Wert 1 annimmt. Schließlich wäre das Ergebnis sonst größer als $g_{\max}$. Falls $\sum_{j=1}^{|\mathrm{TE}|} x_{(i,\ j)} = 0$ ist, so muss hingegen $s_i = 1$ sein. Anders kann die Gleichung nicht erfüllt sein. Zu letzterem gäbe es den Ausnahmefall $g_{\max} = 1$. Dann könnte man auch t_i auf 1 setzen anstatt s_i, doch die *Zeile* 5.10 und $g_{\max} = 1 = g_{\min}$ verhindern dies und setzen t_i auf 0. Schließlich kann $g_{\min}$ nicht gleich 0 sein, sonst gäbe es Teams ohne Teilnehmer.

2. Sie setzt außerdem t_i. Falls $g_{\max} - 1 = g_{\min}$ gilt und es für ein bestimmtes i $g_{\min}$ Variablen von $x_{(i,\ j)}$ den Wert ein 1 annehmen, so muss t_i auf 1 gesetzt werden, um die Gleichung zu erfüllen. In sonstigen Fällen kann diese Variable 0 ergeben, um die Gleichungen zu erfüllen.

3. Die *Zeile* stellt auch sicher, dass jedes Team aus $g_{\min}$ oder $g_{\max}$ Teilnehmern besteht. Es gibt keine Möglichkeit die Gleichung zu erfüllen, wenn für mehr als $g_{\max}$ bzw. weniger als $g_{\min}$ Variablen $x_{(i,\ j)} = 1$ mit $i \in \{1, ..., |\mathrm{TH}|\}$ gilt.

Eine weitere Gleichung wird gebraucht, damit am Ende auch σ Teams heraus kommen. Dafür müssen genau σ Themen verteilt werden. Dazu existiert schon eine *Entscheidungsvariable* s_i, die das zählt. Jedoch hat diese den Wert 1, wenn das Thema nicht genommen wurde.

Deshalb muss die Summe aller s_i der Subtraktion von den Themen und der Anzahl der Teams gleichen.

$$\sum_{i=1}^{|\mathrm{TH}|} s_i = |\mathrm{TH}| - \sigma \tag{5.9}$$

Darüber hinaus ist es wichtig die Gruppen zu filtern, welche nicht aufgeteilt wurden. Mit jener Anzahl wird der *Gruppenverstoß* berechnet. Um die strikten Aufteilungen zu messen, muss man außerdem wissen, wie viele Teilnehmer einer aufgeteilten Gruppe in den jeweiligen Teams sind. Für ein $k \in \{1, ..., |H_G|\}$ können daher Variablen $f_{(i,\ k)}$ und $a_{(i,\ k)}$ definiert werden. So definiert $f_{(i,\ k)}$, ob die gesamte Gruppe k in einem Team enthalten ist. Dieses Team würde das Thema i bearbeiten. Die Subtraktion von der Anzahl der Gruppen und der Summe aller $f_{(i,\ k)}$ ist demnach der *Gruppenverstoß* δ_G. Die Variable $a_{(i,\ k)}$ zählt auf der anderen Seite, wie viele Teilnehmer einer aufgeteilten Gruppe k in jedem Team sind. Des Weiteren wird der Koeffizient g_k gebraucht. Dieser beinhaltet die Größe der ursprünglichen Gruppe k. Deshalb kann auch $a_{(i,\ k)}$ nicht größer als $g_k - 1$ sein, da sonst die gesamte Gruppe in dem Team wäre.

Nun wird eine *Zeile* benötigt, um diese Werte zu setzen. Darum wird die Menge der Gruppen $H_G = \{h_1, ..., h_{|H_G|}\}$ benötigt. Sei nun $h_k = \{\mathrm{te}_1, ..., \mathrm{te}_{g_k}\} \in H_G$ eine Gruppe. Für jede Gruppe und jedes Thema gibt es dann diese *Zeile*.

$$\sum_{\mathrm{te} \in h_k} x_{(i,\ \mathrm{te})} \ - \ a_{(i,\ k)} \ - \ g_k \cdot f_{(i,\ k)} = 0, \quad i \in \{1, ..., |\mathrm{TH}|\} \tag{5.10}$$

$$a_{(i,\ k)} \in \{0, ..., g_k - 1\} \tag{5.11}$$

$$f_{(i,\ k)} \in \{0, 1\} \tag{5.12}$$

Mittlerweile sind alle *Zeilen* erstellt, die eine valide Belegung von der Menge Z liefern. Es fehlen nur noch die, die die Bedingungen einer fairen Teameinteilung erfüllen. Aus bereits bekannten Variablen kann jetzt eine *Zeile* für die erste Bedingung geschaffen werden. Die Bedingung $\delta_G \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor\}$ muss dabei umgeformt werden.

$$\begin{aligned}
&|H_G| - \sum_{i=1}^{|\mathrm{TH}|} \sum_{k=1}^{|H_G|} f_{(i,\ k)} = \delta_G \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor,\ h_{\max} - \sigma_{\max}\} \\
\iff &|H_G| \leq \max\{\lfloor \frac{|H_G|}{2} \rfloor,\ h_{\max} - \sigma_{\max}\} + \sum_{i=1}^{|\mathrm{TH}|} \sum_{k=1}^{|H_G|} f_{(i,\ k)} \\
\iff &|H_G| - \max\{\lfloor \frac{|H_G|}{2} \rfloor,\ h_{\max} - \sigma_{\max}\} \leq \sum_{i=1}^{|\mathrm{TH}|} \sum_{k=1}^{|H_G|} f_{(i,\ k)}
\end{aligned} \tag{5.13}$$

Die resultierende Ungleichung kann gleich für das Programm verwendet werden.

$$\sum_{i=1}^{|\mathrm{TH}|} \sum_{k=1}^{|H_G|} f_{(i,\ k)} \geq |H_G| - \max\{\lfloor \frac{|H_G|}{2} \rfloor,\ h_{\max} - \sigma_{\max}\} \tag{5.14}$$

Um die zweite Bedingung zu überprüfen, muss gezählt werden, wie viele Variablen $a_{(i,\ k)}$ den Wert 1 annehmen. Dies ist nämlich die Anzahl der strikten Aufteilungen. Dafür werden extra zwei weitere Entscheidungsvariablen benötigt.

$$c_{(i,\ k)} = \begin{cases} 1, \text{ falls } \mathrm{a}_{(\mathrm{i},\ \mathrm{k})} = 0 \\ 0, \text{ sonst} \end{cases} \tag{5.15}$$

$$\tag{5.16}$$

$$y_{(i,\ k)} = \begin{cases} 1, \text{ falls } \mathrm{a}_{(\mathrm{i},\ \mathrm{k})} = 1 \\ 0, \text{ sonst} \end{cases} \tag{5.17}$$

Dabei beschreibt $y_{(i,\ k)}$, ob im Team mit dem Thema i eine strikte Teilgruppe von Gruppe h_k existiert. Die Variable $c_{(i,\ k)}$ wird nur gebraucht, um $y_{(i,\ k)}$ zu berechnen. Um nun $c_{(i,\ k)}$ zu berechnen, werden zwei *Zeilen* benötigt. Die nächsten drei *Zeilen* (5.18), (5.19) und (5.21) müssen für jedes $i \in \{1, ..., |\mathrm{TE}|\}$ und $k \in \{1, ..., |H_G|\}$ existieren.

$$a_{(i,\ k)} + c_{(i,\ k)} \geq 1 \tag{5.18}$$

$$a_{(i,\ k)} + g_{\max} \cdot c_{(i,\ k)} \leq g_{\max} \tag{5.19}$$

$$c_{(i,\ k)} \in \{0, 1\} \tag{5.20}$$

Aus der *Zeile* (5.18) folgt nun $a_{(i,\ k)} = 0 \implies c_{(i,\ k)} = 1$. Für den sonstigen Fall, kann $c_{(i,\ k)}$ den Wert 0 oder 1 annehmen. Damit im sonstigen Fall $c_{(i,\ k)}$ gleich 0 ist, gibt es die andere *Zeile* (5.19). Diese setzt $c_{(i,\ k)}$ auf 0, falls $a_{(i,\ k)} \geq 1$ ist. Sonst kann $c_{(i,\ k)}$ beliebig gesetzt werden. Damit kann man $y_{(i,\ k)}$ berechnen.

$$a_{(i,\ k)} + 2c_{(i,\ k)} + y_{(i,\ k)} \geq 2 \tag{5.21}$$

$$y_{(i,\ k)} \in \{0, 1\} \tag{5.22}$$

So gibt es drei verschiedene Fälle für $a_{(i,\ k)}$.

1. Fall $a_{(i,\ k)} = 0$:
   $\implies c_{(i,\ k)} = 1$
   $\implies y_{(i,\ k)} \in \{0, 1\} \implies y_{(i,\ k)} = 0$
2. Fall $a_{(i,\ k)} = 1$:
   $\implies c_{(i,\ k)} = 0$
   $\implies y_{(i,\ k)} = 1$
3. Fall $a_{(i,\ k)} \geq 2$:
   $\implies c_{(i,\ k)} = 0$
   $\implies y_{(i,\ k)} \in \{0, 1\} \implies y_{(i,\ k)} = 0$

Hier muss beachtet werden, dass in der *Zielfunktion* (5.33) die Summe aller $y_{(i,\ k)}$ minimiert wird. Wenn es egal ist, ob $y_{(i,\ k)}$ die Werte 0 oder 1 annehmen kann, so wird dieser immer den Wert 0 annehmen.

Auch für die 3. Bedingung, dass δ_{G} minimiert wird, muss auf die *Zielfunktion* gewartet werden.

Für die 4. Bedingung werden erneut neue *Entscheidungsvariablen* benötigt. Hierfür wird die Variable e_i benutzt. Sie soll den Wert 1 annehmen, wenn das Team nur aus Einzelpersonen und Teilnehmer aus strikten Aufteilungen besteht. Die Menge der Einzelpersonen ist $H \setminus H_G = \{\{te_1\}, ..., \{te_{|H \setminus H_G|}\}\}$. Diese wird zur Berechnung von der Variable e_i gebraucht, die mit den beiden *Zeilen* (5.23) und (5.24) bestimmt wird. Das muss für jedes $i \in \{1, ..., |TH|\}$ gemacht werden.

$$\sum_{\{te\} \in H \setminus H_G} x_{(i,\ te)} + \sum_{k=1}^{|H_G|} y_{(i,\ k)} + t_i - e_i \leq g_{\max} - 1 \tag{5.23}$$

$$\sum_{\{te\} \in H \setminus H_G} x_{(i,\ te)} + \sum_{k=1}^{|H_G|} y_{(i,\ k)} + t_i - g_{\max} \cdot e_i \geq 0 \tag{5.24}$$

$$e_i \in \{0, 1\} \tag{5.25}$$

Für die nächste Erklärung wird dieser Teil der Ungleichung als Formel betrachtet.

$$\sum_{\{te\} \in H \setminus H_G} x_{(i,\ te)} + \sum_{k=1}^{|H_G|} y_{(i,\ k)} + t_i \tag{5.26}$$

Besteht eine Gruppe nur aus Teilnehmern von $H \setminus H_G$ oder $|A_S|$, so ergibt die Formel (5.26) $g_{\max}$. Dann ist die Ungleichung (5.23) nicht erfüllt, außer e_i wird auf 1 gesetzt. Die Ungleichung (5.24) ist in dem Fall auch erfüllt.

Wenn nun eine Gruppe in dem Team ist, so gilt für die Formel (5.26), dass sie kleiner als $g_{\max} - 2$ ist. Das hat den Grund, dass eine Gruppe mindestens 2 Teilnehmer hat. Also ist die 1. Ungleichung (5.23) schon erfüllt und die 2. Ungleichung stellt sicher, dass $e_i = 0$ gilt.

Der letzte Fall wäre, dass das Thema i nicht genommen wurde. Hier wäre das Ergebnis von der Formel (5.26) die 0. Damit ist die Ungleichung (5.23) für beide Werte von e_i erfüllt, aber für die andere Ungleichung (5.24) muss $e_i = 0$ gelten.

$$\sum_{i=1}^{|TH|} e_i \leq \begin{cases} \max\{\sigma - |H_G|, 0\}, \text{ falls } g_{\max} \geq 4 \\ \sigma, \text{ sonst} \end{cases} \tag{5.27}$$

Die darauf folgende Bedingung behandelt die drei verschiedene Arten von *Themenverstößen*. Für alle drei gibt es eine einzige Formel. Dazu wird wieder einmal eine weitere *Entscheidungsvariable* eingeführt. Die Variable $v_{(i,\ l)}$ sagt aus, ob bei Thema i ein *Themenverstoß des Grades* l zutrifft.

$$\begin{aligned} &\sum_{j=1}^{|TE|} q((j,i)) \cdot x_{(i,\ j)} + g_{\max} \cdot s_i + 3 \cdot g_{\max} \cdot v_{(i,\ l)} \geq (g_{\max} - t_i) \cdot l \iff \\ &\quad \sum_{j=1}^{|TE|} q((j,i)) \cdot x_{(i,\ j)} + g_{\max} \cdot s_i + 3 \cdot g_{\max} \cdot v_{(i,\ l)} + l \cdot t_i \geq l \cdot g_{\max} \\ &\qquad\qquad l \in \{-1, 0, 1\},\ i \in \{1, ..., |TH|\} \end{aligned} \tag{5.28}$$

Die Formel rechnet jetzt die Summe aller Bewertungen von Teilnehmern aus, die das Thema i erhalten. Die Summe muss größer sein als die Multiplikation der Teamgröße mit l. Das geht aus der nächsten Formel hervor.

$$\frac{\sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)}}{\sum_{j=1}^{|\mathrm{TE}|} x_{(i,\ j)}} < l \iff \sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)} < l \cdot \sum_{j=1}^{|\mathrm{TE}|} x_{(i,\ j)} \iff \sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)} < l \cdot (g_{\max} - t_i) \tag{5.29}$$

Wenn nun ein *Themenverstoß des Grades l* auftaucht, muss man die Variable $v_{(i,\ l)}$ auf 1 setzen, damit die Ungleichung von (5.28) erfüllt ist. Dafür benötigt $v_{(i,\ l)}$ den *Koeffizienten* $3 \cdot g_{\max}$. Das minimale Ergebnis von $\sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)}$ ist schließlich $-2 \cdot g_{\max}$. Dies kann durch den *Koeffizienten* ausgeglichen werden und selbst für den höchsten Wert von l, die 1, wäre die Ungleichung erfüllt. Damit Themen, die nicht zugewiesen wurden, nicht als *Themenverstoß* gewertet werden, wird auch hier s_i benutzt. Durch die Multiplikation mit $g_{\max}$ ist die Ungleichung für ein nicht zugewiesenes Thema immer gegeben. Eine Gleichung, die extra $v_{(i,\ l)}$ auf 0 setzt, wird nicht benötigt. Die Variable wird in der zukünftigen *Zielfunktion* minimiert.

Nun fehlt nur noch eine *Entscheidungsvariable* für μ. Da μ jedoch eine reelle Zahl ist, kann diese nicht ohne Weiteres für die *ganzzahlige, lineare Programmierung* übernommen werden. Deshalb wird die Variable p genutzt, die die Summe aller Bewertungen, von einem Teilnehmer zu seinem zugewiesenen Thema, ist.

$$\sum_{i=1}^{|\mathrm{TH}|} \sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)} - p = 0 \tag{5.30}$$

Es gilt, dass die Maximalität von p die Maximalität von μ impliziert. Das stellt sich aus den zwei Umformungen heraus.

$$\mu = \sum_{(T,\mathrm{th}) \in Z} \mathrm{avgVal}(T, \mathrm{th}) = \sum_{(T,\mathrm{th}) \in Z} \frac{1}{|T|} \cdot \sum_{\mathrm{te} \in T} q((\mathrm{te}, \mathrm{th})) \tag{5.31}$$

$$p = \sum_{i=1}^{|\mathrm{TH}|} \sum_{j=1}^{|\mathrm{TE}|} q((j,i)) \cdot x_{(i,\ j)} = \sum_{i=1}^{|\mathrm{TH}|} \sum_{T \in H_T} \sum_{\mathrm{te} \in T} q((j, \mathrm{te})) \cdot x_{(i,\ \mathrm{te})} = \sum_{(T,\mathrm{th}) \in Z} \sum_{\mathrm{te} \in T} q((\mathrm{th}, \mathrm{te})) \cdot x_{(\mathrm{th},\ \mathrm{te})} = \sum_{(T,\mathrm{th}) \in Z} \sum_{\mathrm{te} \in T} q((\mathrm{th}, \mathrm{te})) \tag{5.32}$$

Beide sind nun maximal, wenn die Summe der Belegungen $\sum_{\mathrm{te} \in T} q((\mathrm{th}, \mathrm{te}))$ maximal ist.

Nach all diesen *Zeilen* kann endlich die *Zielfunktion* aufgestellt werden. Diese stellt alle Bedingungen dar, die minimiert oder maximiert werden müssen. Die Bedingungen sehen

vor, dass eine Sache vor einer anderen minimiert bzw. maximiert wird. Beispielsweise muss der *Themenverstoß des Grades -1* vor dem *des Grades 0* minimiert werden. Um das zu erreichen, werden die Variablen unterschiedlich gewichtet. Außerdem wird in der Funktion ein Minimum genutzt, da der Großteil minimiert werden muss. Um nun von den restlichen Variablen das Maximum zu berechnen, werden diese einfach abgezogen, an Stelle von einer Addition.

$$\begin{aligned}\min \sum_{i=1}^{|\mathrm{TH}|}\sum_{k=1}^{|H_G|} 2|\mathrm{TE}||H_G|\sigma^3 y_{(i,\ k)} - \sum_{i=1}^{|\mathrm{TH}|}\sum_{k=1}^{|H_G|} 2|\mathrm{TE}|\sigma^3 f_{(i,\ k)} + \sum_{i=1}^{|\mathrm{TH}|} 2|\mathrm{TE}|\sigma^2 v_{(i,\ -1)} \\ + \sum_{i=1}^{|\mathrm{TH}|} 2|\mathrm{TE}|\sigma v_{(i,\ 0)} + \sum_{i=1}^{|\mathrm{TH}|} 2|\mathrm{TE}| v_{(i,\ 1)} - p\end{aligned} \tag{5.33}$$

Dabei soll als letztes p maximiert werden, da μ zu maximieren die letzte Bedingung ist. Deshalb wird diese Variable nur vom restlichen Minimum abgezogen. Die Variablen $v_{(i,\ 1)}$ sollen vor p minimiert werden. Um das zu erreichen, wird jede so viel gewertet, wie der maximale Wert, den p annehmen kann. Nämlich $2|\mathrm{TE}|$. Es können höchstens σ *Themenverstöße* passieren, weil nicht mehr Themen zugewiesen werden. Also werden $v_{(i,\ 0)}$ mit $2|\mathrm{TE}|\sigma$ und $v_{(i,\ -1)}$ mit $2|\mathrm{TE}|\sigma^2$ gewichtet. So ist ein *Themenverstoß des Grades -1* so viel Wert, wie wenn für alle Themen ein *Themenverstoß des Grades 0* gilt.

Auch die nächsten Variablen bekommen eine stärkere Gewichtung mit der Multiplikation von σ und der Gewichtung von $v_{(i,\ -1)}$. Da die Summe von allen $f_{(i,\ k)}$ das Gegenteil vom *Gruppenverstoß* δ_G ist, muss die Summe maximiert statt minimiert werden. Deshalb wird diese Summe von der Gleichung abgezogen.

Am Ende wird noch das Minimum von der strikten Aufteilung kalkuliert. Dazu müssen diese mit $2|\mathrm{TE}||H_G|\sigma^3$ gewichtet werden, da es höchstens $|H_G|$ *Gruppenverstöße* geben kann.

Somit sind alle *Zeilen* und die *Zielfunktion* unseres *ganzzahligen, linearen Programms* definiert und ihr Nutzen wurde umfangreich erklärt.

Zusammenfassend kann gesagt werden, dass die *Zeile* (5.1) jedem Teilnehmer genau ein Thema vergibt. Die *Zeilen* (5.5) und (5.6) berechnen, ob ein Thema vergeben wird und wie groß das Team mit dem Thema ist. Durch (5.9) wird versichert, dass es σ Teams am Ende gibt. Die *Zeile* (5.10) bestimmt, ob eine Gruppe aufgeteilt wurde oder nicht. Daraus kann nun mit (5.14) entschieden werden, ob die 1. Bedingung erfüllt ist. Auch kann bestimmt werden, ob eine Aufteilung strikt ist. Dazu sind die drei *Zeilen* (5.18), (5.19) und (5.21) da. Für die 4. Bedingung zählen die *Zeilen* (5.23) und (5.24), wie viele Teams ohne Gruppen existieren. Die Summe kann durch (5.27) mit der 4. Bedingung verglichen werden. Die *Themenverstöße* werden mit der *Zeile* (5.28) berechnet und die Qualität mittels der *Zeile* (5.30). Zum Ende hin werden mit der *Zielfunktion* (5.33) die Bedingungen 2, 3, 5a), 5b), 5c) und 6 erreicht.

Nachdem das Programm erstellt wurde, können Algorithmen für *ganzzahlige, lineare Programmierung* angewendet werden. Das Resultat kann ohne weitere Probleme in eine Lösung für `FairTeams` umgewandelt werden. Wie schon erwähnt, gilt für die Entscheidungsvariablen $x_{(i,\ j)}$ mit $i \in \{1, ..., |\mathrm{TH}|\}$, $j \in \{1, ..., |\mathrm{TE}|\}$, dass sie den Wert 1 annehmen, wenn das Thema i von Teilnehmer j bearbeitet wird.

Die Menge Z wird als leere Menge instantiiert. Dann wird ein Tupel (T, i) für jedes i erstellt. In die Menge T fügt man dann alle Teilnehmer j ein, für die $x_{(i,\ j)} = 1$ gilt. Falls $T \neq \emptyset$, füge T in Z ein. Nachdem Ablauf kann man Z als Ergebnis für `FairTeams` ausgeben.

Variable	*Bedeutung*
$x_{(i,\ j)}$	1, falls Teilnehmer j das Thema i zugewiesen bekommt 0, sonst
g_k	Die Größe der Gruppe h_k
s_i	0, falls Thema i einem Team zugewiesen wurde 1, sonst
t_i	1, das Team mit dem Thema i hat die Größe $g_{\min}$ und es gilt $g_{\min} \neq g_{\max}$ 0, sonst
e_i	1, das Team mit dem Thema i besteht nur aus Einzelpersonen und strikten Teilgruppen 0, sonst
$f_{(i,\ k)}$	Die ganze Gruppe h_k bekommt das Thema i zugeteilt
$a_{(i,\ k)}$	#Teilnehmer aus der aufgeteilten Gruppe h_k, die das Thema i zugeteilt bekommen. Wenn die ganze Gruppe im Team ist, liegt der Wert bei 0.
$c_{(i,\ k)}$	1, falls $a_{(i,\ k)} = 0$ 0, sonst
$y_{(i,\ k)}$	1, falls $a_{(i,\ k)} = 1$. Die Variable zählt demnach, ob im Team vom Thema i eine strikte Teilgruppe von Gruppe h_k existiert. 0, sonst
$v_{(i,\ l)}$	1, falls das Thema i bei der Vergabe zu einem *Themenverstoß des Grades l* führt. Hierbei gilt $l \in \{-1, 0, 1\}$. 0, sonst
p	Qualität der Vergabe von allen Themen

Tabelle 5.1.: Entscheidungsvariablen, Koeffizienten und ihre Bedeutung

5.2. Wachstum der Formalisierung

Die Berechnung der Variablen verläuft innerhalb von linearer Zeit. Für das *ganzzahlige, lineare Programm* werden hingegen die Entscheidungsvariablen und *Zeilen* gezählt. Eine genaue Messung der Laufzeit ist nicht möglich, da die Algorithmen für die *ganzzahlige, lineare Programmierung* nicht bekannt sind, sondern als Blackbox dienen.

Seien im Folgenden $n = |\text{TE}|$ die Anzahl der Teilnehmer und $m = |\text{TH}|$ die Anzahl der Themen. Es gilt, dass es höchstens $\frac{n}{2}$ Gruppen geben kann. Schließlich besteht eine Gruppe aus mindestens 2 Teilnehmern. Sei $o = |H_G|$ die Anzahl der Gruppen und damit gilt $o \leq \frac{n}{2}$.

Somit existieren $n \cdot m$ Variablen von $x_{(i,\ j)}$. Es gibt auch jeweils $o \cdot m \leq \frac{n \cdot m}{2}$ von den Variablen $f_{(i,\ k)}$, $a_{(i,\ k)}$, $c_{(i,\ k)}$ und $y_{(i,\ k)}$. Auch besteht das Programm aus je m *Entscheidungsvariablen* s_i, t_i, e_i, $v_{(i,\ -1)}$, $v_{(i,\ 0)}$ und $v_{(i,\ 1)}$. Zum Ende gibt es noch genau eine Variable, nämlich p. Die Summe ergibt $n \cdot m + 4 \cdot \frac{n \cdot m}{2} + 6m + 1 = 3n \cdot m + 6m + 1$. Dementsprechend wächst die Anzahl der *Entscheidungsvariablen* mit $O(n \cdot m)$.

Die *Zeilen* wachsen ähnlich schnell. Es gibt von der *Zeile* (5.1) für jeden Teilnehmer einen. Das wären n Stück. Bei der *Zeile* (5.5) gibt es m verschiedene Varianten, da für jedes Thema t_i und s_i gesetzt werden muss. Von den *Zeilen* (5.9), (5.14), (5.27) und (5.30) gibt es aber nur eine Einzige jeweils. Die *Zeile* (5.10) ist ein wenig komplizierter. Dort wird für jede Gruppe geprüft, ob sie in einem Team mit dem Thema i vorhanden ist. Dementsprechend gibt es $o \cdot m \leq \frac{n \cdot m}{2}$ von diesen *Zeilen*. Dasselbe gilt jeweils für (5.18), (5.19) und (5.21). Dann existieren noch jeweils m *Zeilen* von (5.6) , (5.23) und (5.24). Zum Schluss müssen noch $3m$ der *Zeile* (5.28) erstellt werden. Für jedes $l \in \{-1, 0, 1\}$ von $v_{(i,\ l)}$ eine. Dazu kommen noch die unteren und oberen Schranken für alle *Entscheidungsvariablen* mit Ausnahme von p. Das führt zu $2 \cdot O(n \cdot m - 1) = O(n \cdot m)$ zusätzlichen *Zeilen*. Insgesamt gibt es höchstens $n + m + 4 + 4 \cdot \frac{n \cdot m}{2} + 3m + 3m + O(n \cdot m) = 2n \cdot m + 7m + n + 4 + O(n \cdot m)$ *Zeilen*. So wächst auch deren Anzahl in $O(n \cdot m)$.

6. Auswertung der Evaluation

Die Formalisierung ist in Java übersetzt worden. Um das *ganzzahlige, lineare Programm* ausführen zu können, wird dabei die Software Gurobi[1] mit der Version 9.5.0 und der kostenlosen Academic License benutzt. Die Laufzeitbedingungen entsprechen einem PC mit dem 64-Bit Betriebssystem Windows 10 Pro. Der Computer stellt außerdem 8GB RAM und einen Intel(R) Core(TM) i5-2400 Prozessor mit 3,10 GHz bereit.

6.1. Auswertung eines Beispiels

Zum Veranschaulichen der Auswertungen wird ein kleines Beispiel durchgeführt. Als Eingabe wird der Graph G genutzt, der für seine Knoten die Menge TE = {„Arnold“, „Beatrix“, „Cersei“, „David“, „Ellen“, „Finn“, „George“, „Harry“, „Isabell“, „Julia“ } und TH = {„Raspberry Pi Programmierung“, „Analyse der Spieltheorie von Risiko“, „Entwicklung eines Jump N’ Run Spiels“, „Capture The Flag - IT Sicherheit“}. Somit existieren 10 Teilnehmer und 4 Themen. Wie die Teilnehmer die Themen bewerten, die Gewichtung von G, wird in der Tabelle 6.1 fest gehalten. Die Werte wurden zufällig mit einem Würfel bestimmt. Außerdem gibt es drei Gruppen und eine Einzelperson in
H = {{„Arnold“, „Beatrix“, „Cersei“}, {„David“, „Ellen“, „Finn“}, {„George“},
{„Harry“, „Isabell“, „Julia“}} und die Anzahl der Teams wird auf 2 festgelegt. Nun wird unserer Formalisierung das Quadrupel $(G, q, H, 2)$ als Eingabe übergeben.

Daraus berechnet der Algorithmus die Lösung, dass das 1. Team aus „Ellen“, „Finn“, „Harry“, „Isabell“ und „Julia“ besteht und sie das Thema „Analyse der Spieltheorie von Risiko“ bearbeiten werden. Das 2. Team enthält „Arnold“, „Beatrix“, „Cersei“, „David“ und „George“. Ihr Thema ist die „Entwicklung eines Jump N’ Run Spiels“. Die Gruppe mit „David“, „Ellen“ und „Finn“ wurde vom Algorithmus aufgeteilt und „David“ ist allein in einem anderen Team. Dafür wurde ihm ein Thema zugeteilt, das ihm am meisten interessiert. Die anderen 2 Gruppen blieben zusammen. Deshalb liegt der *Gruppenverstoß* δ_G bei 1. Auch die Anzahl der strikten Aufteilungen $|A_S|$ ist 1. Da in allen zwei Teams Gruppen enthalten sind, beläuft sich die Anzahl der Teams mit nur Einzelpersonen γ auf 0.

Das 1. Team hat sein Thema durchschnittlich mit 1,0 bewertet. Das 2. Team war noch zufriedener mit ihrem Thema. Der Durchschnitt der Bewertungen liegt hier bei 1,8. Daraus folgt, dass keiner der drei Arten von *Themenverstößen* einmal auftaucht. Die Qualität μ liegt bei 2,8.

[1] `https://www.gurobi.com/`

	Raspberry Pi Programmierung	*Analyse der Spieltheorie von Risiko*	*Entwicklung eines Jump N' Run Spiels*	*Capture The Flag - IT Sicherheit*
Arnold	1	2	2	0
Beatrix	0	1	2	1
Cersei	−2	0	1	−2
David	0	−2	2	−1
Ellen	1	1	−2	−2
Finn	−2	2	−2	1
George	1	−1	2	0
Harry	−2	1	0	2
Isabell	0	−1	−1	−1
Julia	2	2	−2	0

Tabelle 6.1.: Gewichtung von $q((\text{te}, \text{th}))$

Die Lösung wurde innerhalb von 139 Millisekunden berechnet. Es wurden 112 *Entscheidungsvariablen* und 86 *Zeilen* vom *ganzzahligen, linearen Programm* erstellt.

6.2. Generierung der Eingabe

Bevor weitere Auswertungen begutachtet werden können, muss geklärt werden, wie die Eingabe generiert wird. Bei sehr großen Eingaben ist es unmöglich, alle per Hand einzugeben. Deshalb werden die Bewertungen sowie die Größen der Gruppen zufällig generiert. Das führt jedoch zu Ausreißern in der Auswertung, die im nächsten Kapitel veranschaulicht werden. Um eine zufällige Eingabe zu erstellen, werden drei Variablen benötigt. Eine positive Anzahl an Teilnehmern n und eine ebenso positive Anzahl an Themen m. Die 3. Variable l ist eine Liste aus positiven, ganzzahligen Werten. Diese beinhaltet die Gruppengrößen für jede Gruppe, die erstellt werden soll. Demzufolge ist die Größe der Liste auch die Anzahl der Gruppen.

Daraus können gleich n Teilnehmer und m Themen kreiert werden. Die Bewertung der Themen ist jedoch aufwendiger. Im Folgenden wird die Bewertung des Themas j von Teilnehmer i als $q(i, j)$ beschrieben wie in der Modellierung. Die Bewertungen werden mit Hilfe einer binomialverteilten Zufallsvariablen bestimmt. Für den Ergebnisraum Ω werden die Möglichkeiten der Bewertungen verwendet. In unserem Fall wäre $\Omega = \{-2, -1, 0, 1, 2\}$ die Menge der verschiedenen Bewertungen.
Für die Zufallsvariable $X_j : \Omega \to \{0, ..., 4\}, \omega \mapsto \omega + 2$ gilt $X \sim B(4,\ p_j)$. Wenn p_j größer als $1/2$ ist, so ist die Wahrscheinlichkeit höher, dass das Thema j gut bewertet wird und je größer p_j, desto höher ist auch diese Wahrscheinlichkeit. Andererseits wenn p_j kleiner ist, so ist die Wahrscheinlichkeit höher, dass das Thema j schlechter bewertet wird.

Für alle Themen j soll der Parameter p_j berechnet werden. Zudem soll die Vergabe der Werte für die Parameter $p_1, ..., p_m$ ausgeglichen sein. Das heißt, es sollen ähnlich viele Werte größer als $1/2$ geben, wie kleiner als $1/2$. Um das zu erreichen, wird für p_1 eine zufällige Zahl aus [0, 1] gewählt.

Für alle p_j mit $j \in \{2, ..., m\}$ wird dann eine Zufallszahl berechnet, die entweder im Intervall [0, 0,5] oder im Intervall [0,5, 1] liegt. Die Wahrscheinlichkeit, dass Ersteres gewählt wird, liegt beim Durchschnitt von $p_1, ..., p_{j-1}$. Die Wahrscheinlichkeit vom Letzteren ist dementsprechend, wenn man von 1 den Durchschnitt subtrahiert. So ist es bei übermäßig niedrigen Werten von $p_1, ..., p_{j-1}$ wahrscheinlicher, dass p_j einen höheren Wert zugewiesen bekommt. Im anderen Fall, ist es wahrscheinlicher, dass p_j einen kleineren Wert als 0,5 erhält.

Diese Berechnung von $p_1, ..., p_m$ muss für jede Gruppe durchgeführt werden. Jeder Teilnehmer einer Gruppe nutzt dieselben Werte für p_j. Damit ist gewährleistet, dass die Teilnehmer einer Gruppe die Themen ähnlich bewerten. Auch für jede Einzelperson wird diese Kalkulation ausgeführt. Danach kann die Verteilungsfunktion $F_{X_j}(x)$ bestimmt werden. Eine weitere Zufallszahl z wird generiert. Mit der kann die Bewertung des Themas j im Intervall [0, 1] bestimmt werden. Wenn nun $z \in [0, F_{X_j}(0)]$ ist, so folgt q(i, j) = -2. Sonst gilt $z \in (F_{X_j}(x-1), F_{X_j}(x)] \implies q(i,j) = x - 2$. Somit sind die Bewertungen $q(i, j)$ zufällig bestimmt worden.

Die Größen der Gruppen werden bei der *Gruppenintensität* auch zufällig bestimmt. Dabei wird eine zufällige Zahl zwischen 2 und $g_{\max}$ gewählt und daraus eine Gruppe erstellt. Das wird solange gemacht, bis eine bestimmte Anzahl an Personen auf Gruppen verteilt wurden.

6.3. Auswertung

Aus der Generierung können die Tests automatisch ablaufen. Dabei werden auch die Anzahl der Teilnehmer in Gruppen als Parameter benutzt. Diese wird hier durch die *Gruppenintensität* λ beschrieben. Eine *Gruppenintensität* von 0,25 soll aussagen, dass für circa 25% aller Teilnehmer gilt, dass sie Teil von Gruppen sind. Der Rest, 75% der Teilnehmer, sind Einzelpersonen. Bei der Multiplikation mit der Anzahl der Teilnehmer und der *Gruppenintensität* wird abgerundet.

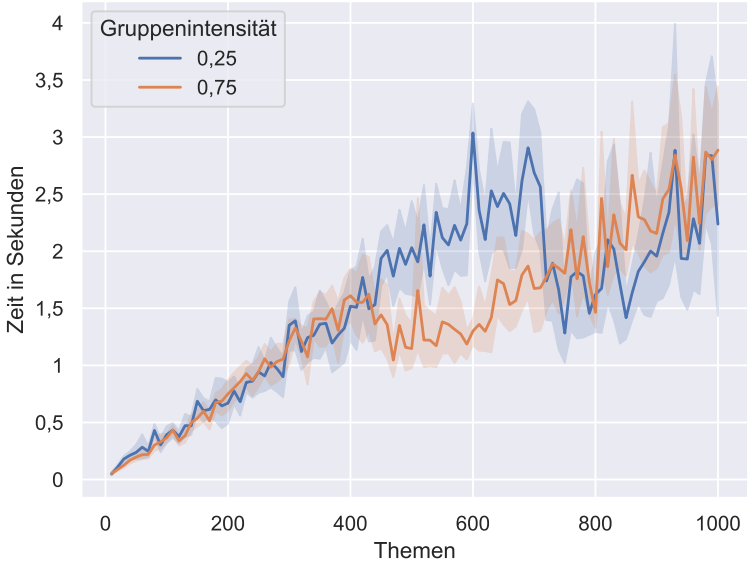

Abbildung 6.1.: Messungen zur Anzahl der Themen mit 25 Teilnehmern, 5 Teams

Der 1. Test betrachtet den Zuwachs der Laufzeit unter der Anzahl der Themen. Als Parameter wurden 25 Teilnehmer und 5 Teams gewählt. Die Zeit wurde für Themen zwischen 10 und 1000 in Zehner-Schritten gemessen. Für jeden Schritt wurden 10 Instanzen generiert und daraus die durchschnittliche Zeit mittels des Diagramms 6.1 ermittelt.

Man sieht in diesem Diagramm 6.1, dass sich die Laufzeit mit der Anzahl der Themen größtenteils steigert. Eine Ausnahme bildet die Stelle von 700 Themen. Hier kommt es zu einem kurzzeitigen Abstieg bei $\lambda = 0,25$. Dieser wird zum Ende hin wieder aufgeholt. Dabei gibt es am Anfang und am Ende keinen großen Unterschied zwischen den zwei

Gruppenintensitäten. Im mittleren Bereich zwischen 450 und 700 Themen ist jedoch $\lambda = 0,25$ etwas aufwendiger zu berechnen. Im Ganzen kann man aber einen leichten Anstieg im Zusammenhang mit der Anzahl der Themen beobachten. Selbst bei einer großen Anzahl kann der Algorithmus eine Lösung in wenigen Sekunden berechnen.

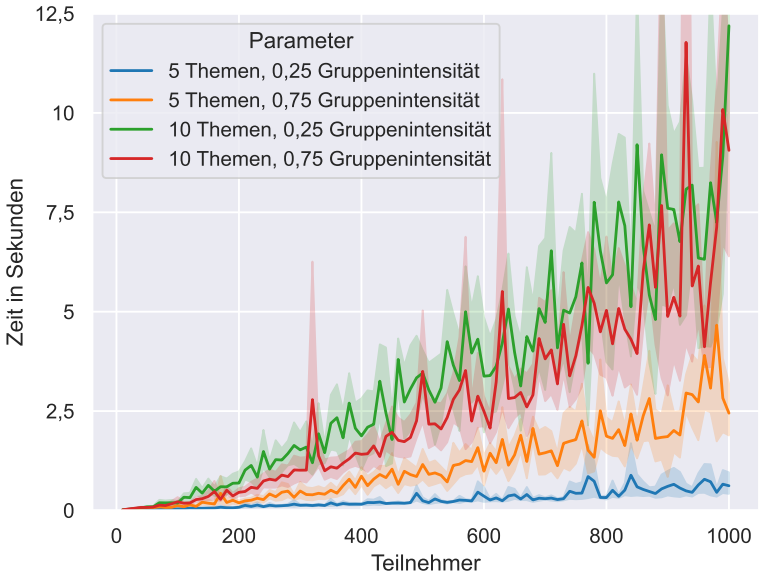

Abbildung 6.2.: Messungen zur Anzahl der Teilnehmer in 5 Teams

Im nächsten Diagramm 6.2 erkennt man, dass der Zusammenhang mit der Anzahl der Teilnehmer und der Themen sich stark auf die Laufzeit auswirken können. Es wurden wieder eine Anzahl von 5 Teams genutzt. Bei 5 Themen und einer niedrigen *Gruppenintensität* waren die meisten Tests nach weniger als einer Sekunde fertig. Bei einer höheren *Gruppenintensität* ändert sich dies schon. Hier dauerten die Tests schon teilweise bis zu 5 Sekunden bei vielen Teilnehmern. Aber die Laufzeit steigert sich mehr mit der Anzahl der Themen. In den zwei anderen Graphen wurde die Themenanzahl verdoppelt. Nicht nur steigerten sich die Werte viel stärker als bei nur 5 Themen. Die Graphen mit $\lambda = 0,25$ waren aufwendiger zu berechnen als die mit $\lambda = 0,75$. Sie benötigten teilweise über 7,5 Sekunden im Bereich von 800 bis 1000 Teilnehmern. In den späteren Diagrammen wird dieses unterschiedliche Verhalten bei den *Gruppenintensitäten* noch genauer betrachtet.

Außerdem wurde bei der Messung eine ausschweifende Instanz entfernt. Diese hatte 560 Teilnehmer, 10 Themen und $\lambda = 0,75$. Die Berechnung für diese Instanz hat insgesamt 864 Sekunden gedauert. Umgerechnet sind das über 14 Minuten. Dies ist ein Vielfaches zu den durchschnittlichen 3 Sekunden, die die anderen Instanzen ausgerechnet haben.

In diesem Diagramm 6.2 kann leider nicht genau erkannt werden, welche Auswirkungen unterschiedliche *Gruppenintensitäten* haben. Es scheint so, dass für wenige Themen eine höhere *Gruppenintensität* zu mehr Laufzeit führt, während bei mehreren Themen eine niedrigere *Gruppenintensität* mehr Zeit verbraucht. Deshalb wurden zwei weitere Tests konzeptioniert, die extra dies untersuchen. Im ersten Test werden n^2 Teilnehmer genutzt, die in n Teams aufgeteilt werden und denen n Themen zugewiesen wird. Dadurch hat jedes Team genau n Teilnehmer.

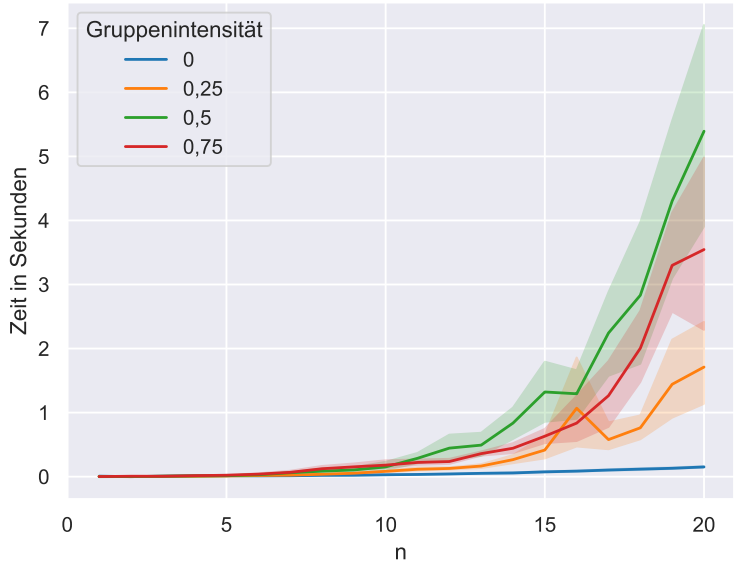

Abbildung 6.3.: n^2 Teilnehmer, n Themen, n Teams, niedrige Gruppenintensität

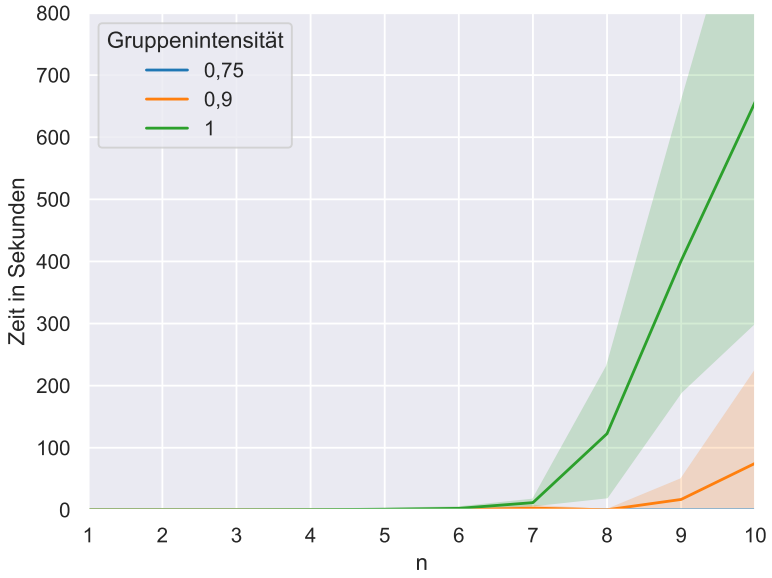

Abbildung 6.4.: n^2 Teilnehmer, n Themen, n Teams, hohe Gruppenintensität

Man erkennt im Diagramm 6.3, dass für $\lambda = 0$ selbst die Berechnung von hohen Werten kein Problem darstellen. Auch bei 400 Teilnehmern, 20 Themen und 20 Teams wird einen winzigen Bruchteil einer Sekunde benötigt zum Ausrechnen. Sobald Gruppen existieren, steigt die Laufzeit aber exponentiell an. Zwar ist die Berechnung von Instanzen für eine mittlere Größe von λ höher als für $\lambda = 0,75$. Im Diagramm 6.4 sieht man jedoch, dass für sehr hohe *Gruppenintensitäten* die Werte schon in kleinerem Rammen stark ansteigen. Für ein $\lambda = 1$ benötigt man schon allein für 100 Teilnehmer, 10 Themen und 10 Teams durchschnittlich 11 Minuten um eine Lösung zu berechnen. Dabei wurden zudem zwei ausschweifende Instanzen entfernt. Beide sind im 2. Durchlauf aufgetreten bei den Stellen $n = 9$ und $n = 10$. Für $n = 9$ hat man knapp eine halbe Stunde gebraucht, für $n = 10$ waren

es sogar über 9 Stunden. Wegen dieser hohen Dauer, wurden bei $\lambda = 1$ nur 5 Instanzen generiert. Auch ohne diese Instanzen sieht man, dass bei $\lambda = 1$ man sehr viel länger braucht, um sich eine Lösung auszurechnen zu lassen. Dies kann daran liegen, dass in diesem Fall oftmals Gruppen aufgeteilt werden müssen. In der Tabelle A.2 sind die Daten zu den einzelnen Instanzen aufgezählt. Ab einem $n = 4$ musste in 82% der Fälle mindestens eine Gruppe aufgeteilt werden. Kleinere Werte für n sind hier uninteressant, da bei höchstens 9 Teilnehmern und 3 Teams es zu selten zu Aufteilungen kommen kann. Bei einem $\lambda = 0,75$ hingegen wurde innerhalb von 10 Instanzen für jedes $n \in \{1, ..., 20\}$ keine einzige Gruppe jemals aufgeteilt. Zum Vergleich wurde ein Durchlauf davon in der Tabelle A.1 festgehalten. Daraus kann gefolgert werden, dass die hohe Zeitdauer für $\lambda = 1$ durch das Suchen nach den besten Aufteilungen entsteht.

Zudem fällt auf, dass sich bei $\lambda = 1$ die Laufzeiten stark unterscheiden. Bei $n = 10$ hat die 1. Instanz nur ungefähr 3 Minuten gebraucht. Beim 2. Durchlauf hat man schon 9 Stunden benötigt. Die 3. dauerte 12 Minuten, die 4. 21 Minuten und die Letzte 7 Minuten. Einen Zusammenhang mit den *Gruppenverstößen* ist nicht zu erkennen. Auch für $n = 8$ und $n = 9$ gibt es größere Schwankungen.

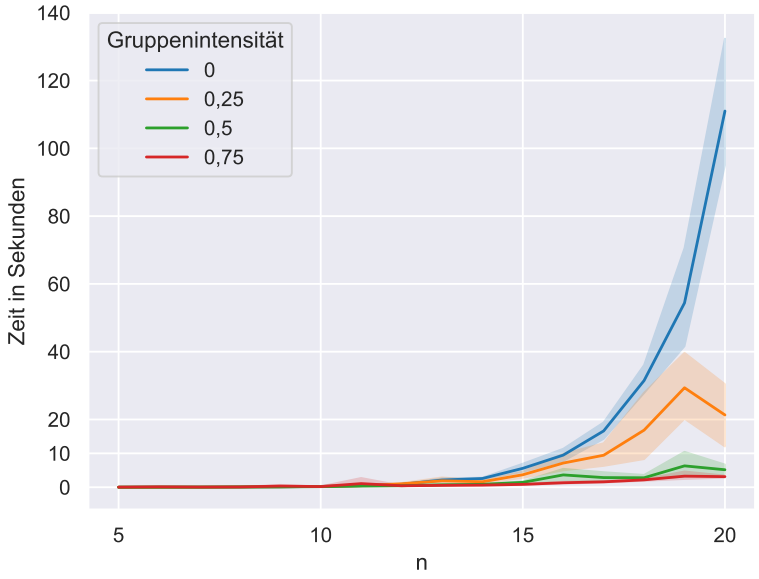

Abbildung 6.5.: n^2 Teilnehmer, n Themen, 5 Teams, niedrige *Gruppenintensität*

Nach diesem Vergleich werden die einzelnen Parameter behalten. Nur die Anzahl der Teams wird auf der Konstante 5 festgelegt. Bei den niedrigeren Werten für die *Gruppenintensität* erkennt man einen gewissen Tausch der Ergebnisse gegenüber dem letzten Test. Für $\lambda = 0$ steigt die Dauer nun exponentiell an. Für 400 Teilnehmern, 20 Themen, 5 Teams und 0 Gruppen benötigte der Rechner durchschnittlich eine Minute und 50 Sekunden. Auch für $\lambda = 0,25$ sind die Zeiten höher als für $\lambda = 0,5$. Letzteres benötigt ebenfalls mehr Zeit als $\lambda = 0,75$. Dies sieht man direkt an den vier Graphen im Diagramm 6.2. Nur für sehr hohe *Gruppenintensitäten* ist die Zeit höher. So zeigt das Diagramm 6.6, dass auch für $\lambda = 0,9$ und $\lambda = 1$ mehr Zeit zum Auswerten benötigt wird. Trotzdem wird weniger Zeit benötigt als bei dem letzten Test mit n Teams. Das kann daran liegen, dass durch die wenigen Teams weniger Gruppen aufgeteilt werden müssen.

Ein weiterer Test betrachtet den Zuwachs der Zeit in Anbetracht, wie viele Teams existieren. Dazu werden 30 Teilnehmer genommen sowie 30 Themen. Das Diagramm 6.7 zeigt, dass bei einer niedrigen Anzahl der Teams die Laufzeit sich zunächst leicht erhöht, bis es bei

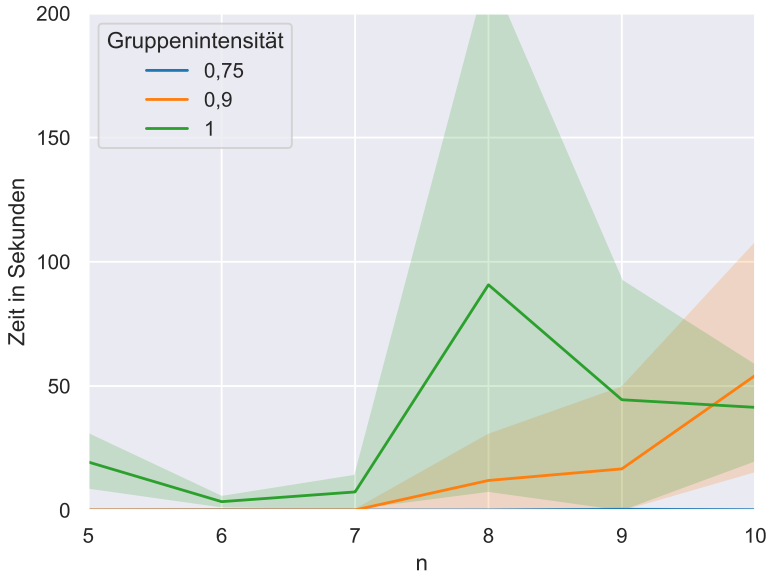

Abbildung 6.6.: n^2 Teilnehmer, n Themen, 5 Teams, hohe *Gruppenintensität*

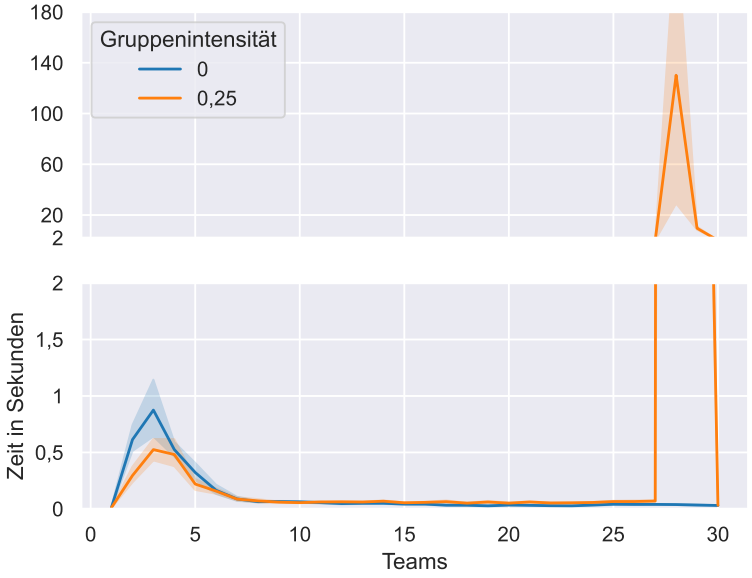

Abbildung 6.7.: Messungen zur Anzahl der Teams mit 30 Teilnehmern und 30 Themen

3 Teams zu einem Höhepunkt von 900 Millisekunden für $\lambda = 0$ bzw. 500 Millisekunden für $\lambda = 0,25$ kommt. Im Bereich von einem Team bis 6 Teams ist $\lambda = 0$ auch ein wenig aufwendiger zu berechnen. Dies ändert sich jedoch für 28 und 29 Teams. Während bei $\lambda = 0$ nur paar Millisekunden für eine Lösung benötigt wird, braucht man für $\lambda = 0,25$ durchschnittlich 10 Sekunden für 29 Teams. Bei 28 Teams sind es sogar durchschnittlich 2 Minuten und 15 Sekunden. Beides ist weit über die restliche Werte hinaus.

Mit dieser *Gruppenintensität* $\lambda = 0,25$ existieren höchstens $\lfloor 30 * 0,25 \rfloor = 7$ Teilnehmer in Gruppen. Da $g_{\max} = 2$ ist für 28 und 29 Teams bei 30 Teilnehmern, gibt es somit 3 Gruppen mit jeweils 2 Teilnehmern. Mit 28 Teams können nur 2 Gruppen zusammen bleiben. Bei 29

sind es nur 1. Trotzdem ist die Dauer bei 28 Teams höher als bei 29. Demzufolge ist es schwerer eine Gruppe zu suchen, die aufgeteilt werden kann, als eine Gruppe, die zusammen bleibt.
Auch hier zeigt sich, dass das Aufteilen von Gruppen zu einer höheren Laufzeit führt.

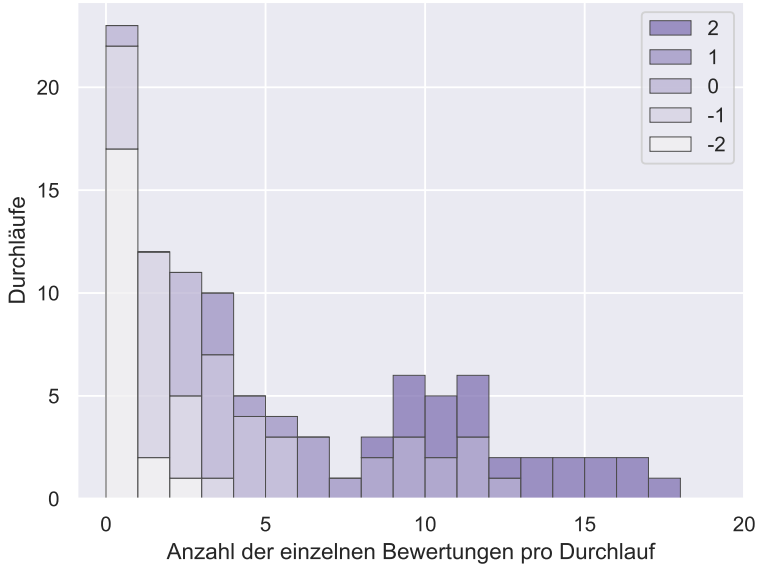

Abbildung 6.8.: Häufigkeit der Bewertungen mit $\lambda = 1$

Weil der Algorithmus eine möglichst faire Teameinteilung errechnen will, muss auch betrachtet werden, wie zufrieden die Teilnehmer mit ihrem Thema sind, wie viel Aufteilungen entstanden sind und welche davon strikt sind. Von 24 Teilnehmern werden in den nächsten Diagrammen die Bewertungen angesehen, die sie ihrem Thema gegeben haben. Dazu wurden 5 Teams, 5 Themen und $\lambda = 1$ als Parameter gewählt. Die Ergebnisse von 20 verschiedenen Instanzen sind in der Tabelle B.3 zu finden. Die Häufigkeit der Bewertungen werden im Diagramm 6.8 veranschaulicht. Die x-Achse steht für die Anzahl der Bewertungen in einem Durchlauf. Die y-Achse misst dann, wie oft es in den 20 Durchläufen eine solche Anzahl aufgetreten ist. Beispielsweise bedeutet im Diagramm 6.8 die zwei Blöcke rechts über der 5, dass in 3 Instanzen 5 Teilnehmer ein Thema bekommen haben, welches sie als „neutral“ bewertet haben. Der zweite, etwas dunklere Block über dem ersten bedeutet, dass in einer Instanz 5 Teilnehmer ein für sie „interessantes“ Thema erhielten.

So sticht heraus, dass in 17 von 20 Fällen kein einziges Thema an einen Teilnehmer zugewiesen wurde, welcher es mit „sehr uninteressant“ bewertet hat. In 10% der Fälle wurde ein solches Thema verteilt und nur in einem schlimmsten Fall wurden zwei gleichzeitig vergeben. Von insgesamt $24 \cdot 20 = 480$ Zuweisungen, machen diese Zuweisungen circa 0,8% aus. Abgesehen davon erkennt man, dass die Qualität sich mit den Anzahl der Bewertungen steigt. Pro Durchlauf gibt es mindestens 8 Bewertungen der Klasse „sehr interessant“. Wenn man die Tabelle hinzuzieht, sieht man auch, dass wenn die Bewertungen der Klasse „sehr interessant“ niedrig sind, so sind die Bewertungen der Klasse „interessant“ höher als üblich. Neben dem Durchlauf 9 mit den zwei sehr schlechten Bewertungen, war der schlimmste Fall der Durchlauf 12, der fünf neutrale Themenzuweisungen und zwei schlechte enthielt. Der Beste ist der 14. Durchlauf. In diesem Durchlauf waren die zwei schlechtesten Zuweisungen, die die von ihren Teilnehmern als „neutral“ angesehen wurden. Insgesamt dominiert die Anzahl der Bewertungen der Klasse „interessant“ und „sehr interessant“ mit

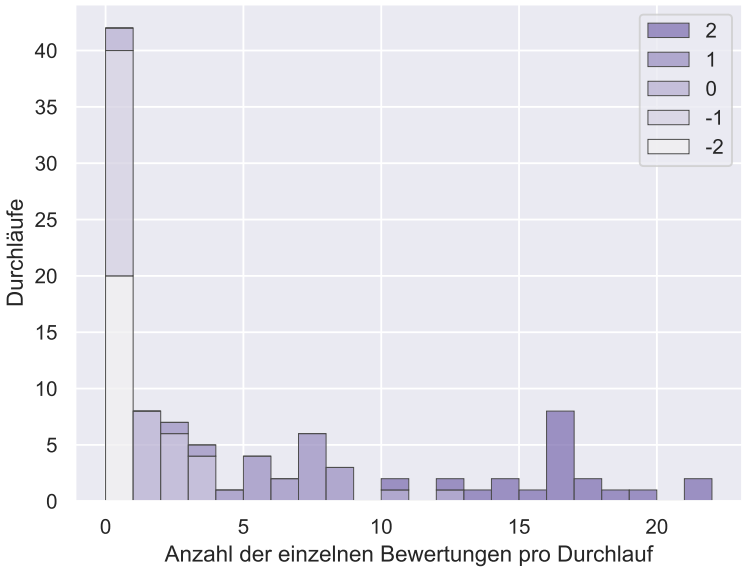

Abbildung 6.9.: Häufigkeit der Bewertungen mit $\lambda = 0$

ungefähr 31,4% und 50,6% der Themenzuweisungen. Somit ist jede 4. Person von 5 mit ihrem Thema zufrieden.

In der Tabelle B.4 kann zudem betrachtet werden, wie viele Gruppen aufgeteilt werden mussten. Von den 6 bis 8 Gruppen musste nie mehr als eine Gruppe in Teilgruppen geteilt werden. In 8 von den 20 Fällen ist jedoch genau eine aufgeteilt worden. Das sind ungefähr 5% aller Gruppen aus allen Instanzen. Insgesamt sind auch 5 Teilnehmer in strikten Aufteilungen geraten. Pro Durchlauf ist es aber nicht mehr als eine Person. Von allen Teilnehmern wären das lediglich 1%. Man kann vermuten, dass sie dafür ein Thema bekamen, dass ihnen sehr interessiert. Schließlich wurde gerade dieser Teilnehmer aus seinem Team gezogen und nicht die Anderen, und es ist gefordert, dass μ maximiert wird. Deshalb ist es sehr wahrscheinlich, dass diese Personen gute Themenzuweisungen erhalten. Des Weiteren ist es interessant zu sehen, wie lange der Algorithmus für diese Beispiele gebraucht hat. In allen acht Fällen, in denen Gruppen aufgeteilt werden mussten, hat der Algorithmus über eine Sekunde zum Berechnen gebraucht. Wenn dabei außerdem eine strikte Teilgruppe entstanden ist, so sind es sogar mindestens 10 Sekunden. Die längste Zeit hat der 1. Algorithmus mit knapp 49 Sekunden gebraucht. Die restlichen Durchläufe sind ziemlich schnell abgelaufen. Das Minimum hierbei betrug 0,017 Sekunden im 6. Durchlauf.

Ohne eine Gruppen sieht die Themenzuteilung noch wesentlich besser aus. Schon allein im Diagramm 6.9 erkennt man, dass in allen Durchläufen exakt null Themen zugewiesen wurden, die mit -2 oder -1 bewertet wurden. Das schlechteste, was die Teilnehmer erhalten, sind Themen, die sie „neutral“ fanden. Selbst diese Anzahl ist viel niedriger als im vorherigen Beispiel. Vor allem wurden viele Themen an Teilnehmern vergeben, die es „sehr interessant“ finden. Von allen Zuweisungen machen diese ganze zwei Drittel aus. Hingegen traten die neutralen Bewertungen nur 6,6% der Fälle auf. So kommt auf eine neutrale Bewertung 10 sehr gute.

Daraus folgt, dass eine niedrigere *Gruppenintensität* zu mehr guten Themenzuweisungen führt. Schließlich müssen für weniger Teilnehmer die Bedingungen der Gruppeneinteilungen erfüllt sein. Auch ein Anstieg an Themen würde die Anzahl der guten Themenzuweisungen steigern, da es mehr Auswahl geben würde.

6.4. Fazit zur Auswertung

Die Diagramme über die Laufzeit haben gezeigt, dass vor allem die *Gruppenintensität* Auswirkungen auf die Laufzeit haben. Während die Zeit mit den Themen und Teilnehmern leicht ansteigt, führt schon der Fall $\lambda = 0$ zu einem exponentiellen Wachstum, wenn man die Anzahl der Teams als Konstante wählt. Jedoch zeigen die Diagramme auch, dass für eine *Gruppenintensität* zwischen 0 und 0,75 im Rahmen von bis zu 225 Teilnehmern, 15 Themen und 15 Teams man die Teameinteilung effizient berechnen lassen kann. All diese Instnazen haben höchstens ein paar Sekunden gedauert, bis eine Lösung berechnet wurde. Für eine höhere *Gruppenintensität* können schon 64 Teilnehmer zu mehreren Minuten Berechnungen führen. Eine Korrelation zwischen einer hohen Laufzeit und dem Aufteilen von Gruppen wurde gefunden. Dies bestätigt auch das Diagramm, welches die Laufzeit unter dem Wachstum der Teams untersucht.

Die Balkendiagramme haben zudem gezeigt, dass ein Großteil der Teilnehmer sehr zufrieden mit der Aufteilung sein wird. Die Zufriedenheit steigert sich sehr, je kleiner die *Gruppenintensität* ist und je höher die Themenauswahl. Bei $\lambda = 1$ gibt es teilweise Teilnehmer, die aus ihrer Gruppe geworfen wurden oder ein Thema bekommen haben, welches sie „uninteressant“ oder sogar „sehr uninteressant“ fanden.

7. Fazit

In dieser Arbeit wurde gezeigt, anhand welcher Fairnessbedingungen eine Teameinteilung gemessen werden kann und wie die Einteilung modelliert wird. Auf dieser Grundlage konnten wir dessen Komplexität, die NP-Vollständigkeit, beweisen. Außerdem wurde eine Formalisierung eines *ganzzahligen, linearen Programms* definiert, welche unser Problem `FairTeams` optimal berechnet. Zwar ist das Problem an sich NP-vollständig, aber für die meisten Instanzen konnte der Algorithmus in kurzer Zeit eine Lösung finden. Eine Ausnahme bildet sich, wenn sich ein sehr hoher Anteil der Teilnehmer in Gruppen befindet. Hier können sich Berechnungen sehr in die Länge ziehen.
Ein weiterer Ausnahme ist, wenn sich die Anzahl der Teams konstant klein bleibt, während die Anzahl der Themen und der Teilnehmer steigt. In diesem Fall steigt die Laufzeit vor allem für sehr niedrige *Gruppenintensitäten* an. Für solche Fälle müssten deshalb Heuristiken entwickelt werden. Auch wurde gezeigt, dass der Großteil der Teilnehmer sehr zufrieden mit der Einteilung sein würden.

Man kann diese Arbeit noch in dem Punkt erweitern, indem ein weiterer Beweis zur Komplexität geführt wird. Dieser sollte die Größe der Themen auf eine Konstante beschränken, um zu sehen, ob das Problem NP-vollständig bleibt. Eine zusätzliche Erweiterung wäre die Entwicklung von Heuristiken, die dann miteinander und mit der Formalisierung verglichen werden. Dabei können auch noch weitere Randfälle bei einer Auswertung betrachtet werden. Ein Beispiel dazu wäre, wenn jede Gruppe nur wenige Themen gut bewertet. Auch könnte man den Themen einen gewissen Beliebtheitsgrad vergeben.

Außerdem wäre es interessant, wie man diesen Algorithmus in eine Software umwandeln würde. Speziell in der Form einer Webseite würde das große Anwendung finden. Man könnte dafür, wie in der Modellierung erwähnt, ein Fünf-Sterne-System einführen, mit dem die Teilnehmer die Themen bewerten können. Man muss diskutieren, was mit den Teilnehmern geschieht, die Themen nicht bewertet haben. Es ist auch wichtig zu betrachten, wie man zu viele schlechte Bewertungen verhindert. Ein Teilnehmer könnte nämlich alle Themen mit „sehr uninteressant“ bewerten, abgesehen von seinem Favoriten. Das sollte von der Software oder der Webseite verhindert werden. Zudem wäre einem interessante Frage, in welchem Kontext die Formalisierung genutzt wird und in welchem eine Heuristik geeigneter wäre.

Literaturverzeichnis

[BIBD11] Katz B., Rutter I., Strasser B. und Wagner D: *Speed Dating*. Experimental Algorithms, 6630:292–303, 2011. https://doi.org/10.1007/978-3-642-20662-7_25.

[H.91] Karloff H.: *Linear Programming*, Kapitel The Basics. Birkhäuser, 1991, ISBN 0-8176-3561-0.

[W.92a] Dinkelbach W.: *Operations Research*, Kapitel Lineare Programmierung. Springer, Berlin, Heidelberg, 1992, ISBN 978-3-540-54926-0. https://link.springer.com/chapter/10.1007/978-3-642-77162-0_4.

[W.92b] Dinkelbach W.: *Operations Research*, Kapitel Ganzzahlige lineare Programmierung. Springer, Berlin, Heidelberg, 1992, ISBN 978-3-540-54926-0. https://link.springer.com/chapter/10.1007/978-3-642-77162-0_4.

[ZFC13] Jiang Z., Chen F. und Wu C.: *The Strong NP-Completeness of 3-PARTITION Problem with $B \geq K^m$*. The Journal of the Operational Research Society, 64(5):786–787, 2013. http://www.jstor.org/stable/23407021.

Anhang

A. Tabellen zu der Gruppenintensität

n	δ_G	$\lvert A_S \rvert$	γ	$\delta_{T_{-1}}$	δ_{T_0}	δ_{T_1}	μ	*Laufzeit*
1	0	0	1	0	0	0	1.0	0s
2	0	0	1	0	1	1	0.0	0,032s
3	0	0	0	0	0	1	3.3	0s
4	0	0	0	0	0	1	4.75	0,016s
5	0	0	0	0	0	0	6.8	0,016s
6	0	0	0	0	0	0	9.67	0,016s
7	0	0	0	0	0	0	10.85	0,047s
8	0	0	0	0	0	0	10.375	0,078s
9	0	0	0	0	0	0	13.2	0,328s
10	0	0	0	0	0	0	15.6	0,172s
11	0	0	0	0	0	1	16.36	0,172s
12	0	0	0	0	0	0	20.25	0,141s
13	0	0	0	0	0	0	23.15	0,344s
14	0	0	0	0	0	1	22.36	0,375s
15	0	0	0	0	0	1	25.2	0,469s
16	0	0	0	0	0	2	23.8125	0,843s
17	0	0	0	0	0	5	19.5	0,407s
18	0	0	0	0	0	0	33.17	1,563s
19	0	0	0	0	0	0	34.21	5,079s
20	0	0	0	0	0	0	37	4,562s

Tabelle A.1.: Eine Instanz pro n mit einer *Gruppenintensität* von 0,75, n^2 Teilnehmer, n Themen, n Teams

Durchlauf	n	δ_G	$\lvert A_S \rvert$	γ	$\delta_{T_{-1}}$	δ_{T_0}	δ_{T_1}	μ	*Laufzeit*
1	1	0	0	1	0	1	1	-1.0	0,015s
	2	0	0	0	0	0	0	2.0	0s
	3	0	0	0	0	0	1	3.4	0,015s
	4	2	2	0	0	0	2	4.25	0,828s
	5	1	1	0	0	0	0	7.8	3,314s
	6	1	0	0	0	0	1	8.2	5,001s
	7	1	0	0	0	0	1	9.7	5,485s
	8	1	0	0	0	0	0	12.75	245,222s
	9	2	0	0	0	0	0	14.1	157,562s
	10	3	0	0	0	0	0	16.4	182,866s
2	1	0	0	1	1	1	1	-2.0	0s
	2	0	0	0	0	0	0	2.5	0,016s
	3	0	0	0	0	0	0	4.0	0s
	4	0	0	0	0	0	2	3.0	0,016s
	5	1	0	0	0	0	1	5.8	0,610s
	6	2	0	0	0	0	0	8.2	3,860s
	7	1	0	0	0	0	1	9.4	6,234s
	8	1	0	0	0	0	0	11.25	26,876s
	9	2	0	0	0	0	1	13.3	1727,867s
	10	2	0	0	0	0	0	16.0	32441,689s
3	1	0	0	1	0	1	1	-1.0	0s
	2	0	0	0	0	1	2	-1.0	0,016s
	3	0	0	0	0	0	2	1.7	0,015s
	4	1	1	0	0	0	1	4.25	0,297s
	5	0	0	0	0	0	3	4.2	0,094s
	6	4	0	0	0	0	1	8.5	3,047s
	7	3	0	0	0	0	1	9.0	21,360s
	8	1	0	0	0	0	0	11.375	295,030s
	9	2	0	0	0	0	0	13.2	799,462s
	10	2	0	0	0	0	0	15.8	741,226s
4	1	0	0	1	1	1	1	-2.0	0s
	2	0	0	0	0	1	2	0.0	0s
	3	0	0	0	0	0	1	3.3	0,016s
	4	2	2	0	0	0	1	5.25	0,406s
	5	0	0	0	0	0	4	2.8	0,047s
	6	0	0	0	0	0	1	5.8	0,156s
	7	2	0	0	0	0	1	9.1	13,502s
	8	0	0	0	0	0	1	9.875	0,297s
	9	3	0	0	0	0	0	12.4	222,657s
	10	1	0	0	0	0	0	15.1	1283,263s
5	1	0	0	1	0	1	1	-1.0	0s
	2	0	0	0	0	0	1	2.0	0s
	3	0	0	0	0	1	1	2.0	0,031s
	4	1	1	0	0	1	2	2.75	0,344s
	5	1	0	0	0	0	2	6.0	0,781s
	6	0	0	0	0	0	2	6.8	0,078s
	7	2	0	0	0	0	2	8.3	11,690s
	8	1	0	0	0	0	2	8.5	46,144s
	9	2	0	0	0	0	0	15.4	424,468s
	10	1	0	0	0	0	0	15.1	420,788s

Tabelle A.2.: Instanzen mit einer *Gruppenintensität* von 1, n^2 Teilnehmer, n Themen, n Teams

B. Tabellen zu der Qualität von den Teameinteilungen

Durchlauf	*Bewertung -2*	*Bewertung -1*	*Bewertung 0*	*Bewertung 1*	*Bewertung 2*
1	1	1	2	10	10
2	0	1	3	11	9
3	1	0	4	3	16
4	0	1	3	8	12
5	0	1	0	8	15
6	0	2	2	6	14
7	0	0	5	5	14
8	0	3	2	9	10
9	2	0	3	6	13
10	0	1	4	6	13
11	0	1	3	9	11
12	0	2	5	7	10
13	0	1	3	3	17
14	0	0	2	11	11
15	0	2	4	9	9
16	0	1	5	3	15
17	0	2	2	12	8
18	0	1	2	10	11
19	0	1	3	4	16
20	0	0	4	11	9

Tabelle B.3.: Einzelne Bewertungen der Teilnehmer zu ihrem zugewiesenen Thema mit einer *Gruppenintensität* von 1. Weitere Parameter sind 24 Teilnehmer, 5 Themen und 5 Teams.

Durchlauf	$\|H_G\|$	δ_G	$\|A_S\|$	γ	$\delta_{T_{-1}}$	δ_{T_0}	δ_{T_1}	μ	*Laufzeit*
1	6	1	1	0	0	0	0	5.65	48,860s
2	7	0	0	0	0	0	1	5.85	0,021s
3	7	1	1	0	0	0	1	7.0	15,023s
4	7	1	1	0	0	0	1	6.5	12,607s
5	6	1	1	0	0	0	0	7.75	11,812s
6	6	0	0	0	0	0	1	6.7	0,017s
7	8	0	0	0	0	0	1	6.75	0,080s
8	7	0	0	0	0	0	2	5.25	0,060s
9	7	0	0	0	0	0	1	5.8	0,380s
10	8	1	0	0	0	0	1	6.4	1,790s
11	7	1	1	0	0	0	0	6.25	31,627s
12	8	0	0	0	0	0	3	5.3	0,066s
13	7	0	0	0	0	0	1	7.25	0,030s
14	6	1	0	0	0	0	0	6.9	1,392s
15	7	0	0	0	0	0	2	5.35	0,060s
16	7	1	0	0	0	0	0	6.6	2,110s
17	8	0	0	0	0	0	1	5.2	0,060s
18	6	0	0	0	0	0	1	6.5	0,300s
19	7	0	0	0	0	0	0	7.2	0,090s
20	7	0	0	0	0	0	1	6.1	0,512s

Tabelle B.4.: Weitere Daten zu den 20 Durchläufen mit den 24 Teilnehmern, 5 Themen, 5 Teams und einer *Gruppenintensität* von 1.

Durchlauf	*Bewertung -2*	*Bewertung -1*	*Bewertung 0*	*Bewertung 1*	*Bewertung 2*
1	0	0	3	6	15
2	0	0	2	6	16
3	0	0	3	5	16
4	0	0	3	8	13
5	0	0	2	12	10
6	0	0	1	5	18
7	0	0	0	8	16
8	0	0	1	7	16
9	0	0	1	2	21
10	0	0	1	7	16
11	0	0	2	10	12
12	0	0	0	3	21
13	0	0	1	7	16
14	0	0	3	7	14
15	0	0	2	5	17
16	0	0	2	8	14
17	0	0	1	7	16
18	0	0	1	4	19
19	0	0	1	7	16
20	0	0	2	5	17

Tabelle B.5.: Einzelne Bewertungen der Teilnehmer zu ihrem zugewiesenen Thema mit einer *Gruppenintensität* von 0. Weitere Parameter sind 24 Teilnehmer, 5 Themen und 5 Teams.