FACHHOCHSCHULE **TRIER**
Hochschule für Technik, Wirtschaft und Gestaltung
University of Applied Sciences
Informatik

Untersuchung der parametrisierten Komplexität des Student Sectioning Problems

Analysis of the parameterized complexity of the student sectioning problem

Maria Dostert

Master Abschlussarbeit

Betreuer: Prof. Dr. Heinz Schmitz,
Prof. Dr. Peter Rossmanith

Trier, 01. Januar 2013

Danksagung

An dieser Stelle möchte ich mich bei allen Bedanken, die mich während meiner Arbeit unterstützten und somit ein erfolgreiches Abschließen ermöglichten. Diese Arbeit fand in Kooperation der Hochschule Trier mit dem Lehrstuhl für Theoretische Informatik der RWTH Aachen statt und daher möchte ich mich bei denjenigen bedanken, die diese Zusammenarbeit ermöglichten.
Mein Dank gebührt unter anderem Prof. Dr. Heinz Schmitz der Hochschule Trier, da er mein Interesse für die Theoretische Informatik weckte und mir fachlich im Laufe der letzten Jahre und auch während meiner Masterarbeit immer zur Seite stand.
Des Weiteren möchte ich mich bei Prof. Dr. Peter Rossmanith der RWTH Aachen bedanken, da ohne ihn diese Zusammenarbeit nicht möglich gewesen wäre und er mir die Gelegenheit gab die Masterarbeit bei ihm zu schreiben und mich dabei unterstützte.
Zudem möchte ich mich auch bei Alexander Langer, einem ehemaligen Doktoranten der RWTH Aachen, für die gute Betreuung bedanken, dass er mir bei Fragen immer zur Seite stand und somit einen wesentlichen Beitrag zum Gelingen dieser Arbeit geleistet hat.
Abschließend gilt mein Dank auch Felix Reidl, einem Doktoranten der RWTH Aachen, denn er setzte die gute Betreuung von Alexander Langer fort und half mir diese Arbeit erfolgreich abzuschließen.

Kurzfassung

Die Studierenden einer Hochschule nehmen üblicherweise jedes Semester an einer Reihe von Lehrveranstaltungen und zugehörigen Übungsgruppen teil. Für die Hochschulen ergibt sich dadurch unter anderem das Problem, dass die Studierenden auf die verfügbaren Übungsgruppen aufzuteilen sind und zwar am besten so, dass die Studierenden problemlos an sämtlichen Übungen teilnehmen können. Restringiert wird dieses Problem durch verschiedene Bedingungen, da zum Beispiel die Teilnehmerzahl der Übungsgruppen beschränkt ist oder manche Übungen zeitgleich stattfinden. Das *Student Sectioning Problem (SSP)* adressiert genau diese Problemstellung, denn es liefert eine Antwort auf die Frage, ob alle Studierenden an den Übungen ihrer gewünschten Module konfliktfrei teilnehmen können.
In dieser Arbeit werden unterschiedliche parametrisierte Varianten des *SSP* untersucht. Aus der Analyse der parametrisierten Komplexität der verschiedenen Varianten ergibt sich unmittelbar, welche Eingabedaten die exponentielle Größe des Lösungsraums und somit auch die Schwierigkeit dieses Problems beeinflussen. So wird gezeigt, dass sich das parametrisierte *SSP* in der Komplexitätklasse *FPT* befindet, falls die Gesamtzahl der Übungen in der Parametrisierung enthalten ist und das es bei verschiedenen anderen Parametern *para-NP-vollständig* ist.

Abstract

Students at universities usually attend lots of different courses and take part in corresponding sections each semester. For the universities this is accompanied by the problem that students have to be assigned to available sections. In best case this assignment is done in such a way that all students are capable of attending the corresponding sections. In order to be able to perform this assignment, several restrictions have to be taken into account, because e.g. the maximal number of participants of a section is restricted or certain sections take place concurrently. The Student Sectioning Problem (SSP) addresses this problem as it concerns the question whether all students can participate in all sections of their desired courses without any conflicts.
In this work several different parameterized variants of SSP are analysed. Based on the analysis of the parameterized complexity it can be derived which input data primarily influence the exponential size of the solution set and hence the difficulty of the problem. More concretely we show that the parameterized SSP is in the complexity class *FPT* in case the total number of sections is part of the parameterization. Moreover we show that it is *para-NP complete* using some other kinds of parameterizations.

Inhaltsverzeichnis

Abbildungsverzeichnis

Definitionsverzeichnis

1

Einleitung

In dieser Arbeit wird das *Student Sectioning Problem (SSP)* bezüglich seiner parametrisierten Komplexität untersucht. Im nachfolgenden Abschnitt wird zunächst die Praxisrelevanz von *SSP* verdeutlicht. Im Anschluss daran wird der Aufbau der Arbeit vorgestellt.

1.1 Hintergrund der Arbeit

Eines der Ziele der Hochschulleitungen ist es, möglichst viele Studieninteressierte für ihre Hochschule zu begeistern und daher mit einer vielfältigen Auswahl an Veranstaltungen zu werben. Wenn die Hochschulen es schaffen, eine große Anzahl an Studienanfängern zu erreichen, möchten sie diese anschließend aufrecht erhalten und die Studierenden weiterhin mit ihrem Lehrangebot überzeugen. Dies ist nicht nur für den Ruf der Hochschule relevant, sondern es ist auch aus finanzieller Sicht ein wichtiger Aspekt, da die Höhe der finanziellen Zuschüsse von der Anzahl der Studienanfänger abhängt [BGvS09]. Je mehr Studierende eine Hochschule aufnimmt, desto höher ist potenziell der Bedarf an Übungsgruppen, Räumen und Lehrpersonal. Falls möglichst wenige dieser Ressourcen benötigt werden, ist dies nicht nur aus finanzieller Sicht vorteilhaft, sondern kann auch die Stundenplanung verbessern, wodurch die Zufriedenheit der Studierenden und des Lehrpersonals erhöht wird. Neben der Stundenplanung ist ein weiteres wichtiges Problem die möglichst optimale Zuteilung der Studierenden zu den Übungsgruppen der gewünschten Module (Lehrveranstaltungen). Wenn ein Studierender häufig nicht an den gewünschten Modulen teilnehmen kann, da sonst Termin- oder Kapazitätskonflikte entstehen würden, steigt die Unzufriedenheit der Studierenden. Demnach ist eine optimale Studierendenverteilung sowohl aus Sicht der Hochschulleitung als auch aus Sicht der Studierenden sehr wichtig. Das *SSP* adressiert genau dieses Problem, da es, auf Basis eines bereits bestehenden Stundenplans, die Frage behandelt ob die Studierenden an ihren gewünschten Modulen konfliktfrei teilnehmen können.

1.2 Aufbau der Arbeit

Diese Arbeit untersucht das Entscheidungsproblem *SSP* bezüglich seiner parametrisierten Komplexität. Es gibt viele verschiedene Varianten dieses Problems, die sich einerseits in der grundlegenden Problemstruktur und andererseits in der Auswahl der weichen und harten Kriterien unterscheiden. Die genaue Definition des hier analysierten *SSP* wird in Kapitel 2 vorgestellt. Da sich aus den Ergebnissen der Komplexität des SSP im Sinne der klassischen, nicht-parametrisierten, Komplexitätstheorie Erkenntnisse bezüglich der parametrisierten Komplexität ziehen lassen, wird in Kapitel 3 zunächst die Komplexität des nicht-parametrisierten *SSP* betrachtet.
In dem Kapitel zur Analyse der parametrisierten Komplexität 4 wird einleitend in Abschnitt 4.1 der Zusammenhang zwischen der klassischen und der parametrisierten Komplexitätstheorie vorgestellt und allgemein auf die parametrisierte Komplexitätstheorie mit ihren Komplexitätsklassen eingegangen. Da in der parametrisierten Komplexitätstheorie Problem-Parameter-Kombinationen untersucht werden, ist für die Analyse eine Parametrisierung des *SSP* erforderlich. Dieses Problem enthält zahlreiche Eingabedaten, somit bieten sich viele unterschiedliche Parametrisierungen an, welche in Abschnitt 4.2 vorgestellt werden. In den weiteren Abschnitten (4.3 - 4.8) werden eine Auswahl von Problem-Parameter-Kombinationen des *SSP* bezüglich der jeweiligen parametrisierten Komplexität untersucht und miteinander verglichen. Bei der Parametrisierung ist die Wahl des Parameters von hoher Bedeutung, da dies die Komplexität beeinflusst. Somit wird in jedem dieser Unterkapitel das *SSP* jeweils mit unterschiedlichen Parametern untersucht.
Zusammenfassend werden die Resultate dieser Arbeit in Kapitel 5 dargestellt und ein Ausblick auf weitere mögliche Arbeiten gegeben.

2

Problemstellung SSP

Das *Student-Sectioning-Problem (SSP)*, auch als *Student-Scheduling-Problem* bekannt, ist ein Entscheidungsproblem, das in dieser Arbeit untersucht wird. Die Studierenden einer Hochschule erhalten pro Semester einen Stundenplan, in dem alle Ereignisse, an denen sie teilnehmen dürfen, auf Zeitfenster verteilt sind. Es gibt verschiedene Arten von Ereignissen, zum Beispiel Übungsgruppen, Vorlesungen, Seminare oder Tutorien. Jedes dieser Ereignisse ist einer Lehrveranstaltung, hier als Modul bezeichnet, zugeordnet. Der Unterschied zwischen den verschiedenen Ereignissen liegt zum Beispiel darin, dass Studierende, die ein Modul wählen, an allen zugehörigen Vorlesungen teilnehmen sollen, aber die Anwesenheit in Tutorien freiwillig ist. Es gibt noch weitere Unterschiede, welche die verschiedenen Ereignisse voneinander unterscheiden, beispielsweise sollen die Studierenden an allen Vorlesungen und Seminaren des gewählten Moduls teilnehmen und können auch alle zugehörigen Tutorien besuchen. Die Übungsgruppen grenzen sich diesbezüglich von den anderen Ereignissen ab, da hier eine Aufteilung stattfinden muss, denn jeder Studierende sollte nur an einer Übungsgruppe pro Modul teilnehmen. Die Variante des hier untersuchten *SSP* beschäftigt sich ausschließlich mit der Zuordnung von Studierenden zu Übungsgruppen.
Hierbei sind einige wichtige Restriktionen zu beachten, wie zum Beispiel, dass die Studierenden nicht Übungsgruppen zugewiesen werden dürfen, die zeitgleich stattfinden. Des Weiteren dürfen einer Übungsgruppe nicht mehr Studierende zugeordnet werden als der zugehörige Raum aufnehmen kann. Somit besteht eine Abhängigkeit zwischen der Studierendenzuordnung und der Stundenplanung und daher muss diesbezüglich eine Reihenfolge festgelegt werden. Hierzu gibt es ebenfalls verschiedene Varianten. Eine Möglichkeit wäre, dass zuerst die Studierendenzuordnung stattfindet und anschließend die Stundenplanung. Für die Studierendenzuordnung wäre dies einfacher, da Terminkonflikte nicht berücksichtigt werden müssten, jedoch wäre es für die Stundenplanung schwieriger, da diese die Studierendenzuordnung mit einbeziehen muss und somit Einschränkungen bei der Verteilung der Übungsgruppen zu den Zeitfenstern und Räumen hat. Alternativ kann die Stundenplanung und die Studierendenzuordnung zeitgleich stattfinden. Dies bedeutet, dass abwechselnd eine Übungsgruppe einem Zeifenster-Raum-Paar und ein Studierender einer Übungsgruppe zugeordnet wird. Das hier definierte *SSP* baut auf einem existierenden Stundenplan auf, das heißt die Stundenplanung

findet vor der Studierendenverteilung statt und ist abgeschlossen, bevor die Studierendenverteilung beginnt.
Nachfolgend wird zunächst die in dieser Arbeit betrachtete Variante des *SSP* formal definiert. Hierzu wird der Begriff *Studierenden/Modul-Kombination* verwendet, welcher ein Tupel beschreibt, das von einem Studierenden und einem seiner gewählten Module gebildet wird. Im Anschluss daran wird näher auf die Definition des *SSP* eingegangen.

Definition 1. *Student-Sectioning-Problem (SSP)*

Eingabe: *Module* $M = \{m_1, ..., m_v\}$
Studierende $S = \{s_1, ..., s_r\}$
Übungsgruppen $U = \{u_{11}, ..., u_{vj}\}$
Übungsgruppenkapazität $\tau \colon U \to \mathbb{N}^+$
Modulwahl $\beta \colon S \to 2^M \backslash \emptyset$
Terminkonflikte $\delta \colon U \times U \to \{0, 1\}$
Grenze unverteilter Studierenden/Modul-Kombinationen $k \in \mathbb{N}$

Frage: *Existiert eine Verteilung der Studierenden zu Übungsgruppen, so dass alle, bis auf maximal k, Studierenden/Modul-Kombinationen genau eine Übungsgruppe des Moduls zugewiesen bekommen, ohne die Kapazität einer Übungsgruppe zu überschreiten und Terminkonflikte zu erzeugen?*

Die Frage des *SSP* wird formal wie folgt definiert:

Gibt es eine mindestens $(\#A - k)$-elementige Relation $R \subseteq A \times U$ mit
$A = \{(s_i, m_j) | s_i \in S \wedge m_j \in \beta(s_i)\}$, so dass $\forall (a_d, u_{ld}) \in R$ mit
$a_d = (s_i, m_j)$ die folgenden Bedingungen erfüllt werden?

1. $l = j$ (u_{ld} muss eine Übungsgruppe von m_j sein)
2. $\#S' \leq \tau(u_{ld})$ mit $S' = \{s_i | s_i \in S \wedge ((s_i, m_l), u_{ld}) \in R\}$
   (Einhaltung der Kapazität pro Übungsgruppe)
3. $\forall ((s_i, m), u) \in R : m \neq m_j \Rightarrow \delta(u, u_{ld}) = 0$
   (ein Studierender darf nicht mehreren Übungsgruppen, die mindestens einen gemeinsamen Termin haben, zugeteilt werden)
4. $\forall (a_d, u_{mf}) \in R : u_{ld} = u_{mf}$
   (kein Tupel darf mehreren Übungsgruppen zugewiesen werden)

Die Eingabe des *SSP* beinhaltet zum einen eine Menge von Modulen M. Jedes in M enthaltene Modul ist eine Lehrveranstaltung, welche die Studierenden wählen können. Des Weiteren ist die Menge der Studierenden S ein Teil der Eingabe,

sowie die Übungsgruppen U. Jede Übungsgruppe wird durch zwei Indizes bezeichnet, wobei der erste Index gleich dem Index des zugehörigen Moduls ist und der zweite Index für die Nummer der Übungsgruppe steht, so ist beispielsweise u_{ij} die j. Übungsgruppe des Moduls m_i.
Bei der Verteiltung der Studierenden zu den Übungsgruppen ist die Kapazität der, von der Stundenplanung zugeteilten, Räume zu beachten. Des Weiteren werden in manchen Übungsgruppen Ressourcen, wie zum Beispiel Computer, benötigt, deren Anzahl beschränkt ist und somit kann eine Übungsgruppe nur maximal so viele Teilnehmer aufnehmen wie es benötigte Ressourcen im zugehörigen Raum gibt. Folglich existiert zu jeder Übungsgruppe eine Kapazitätsgrenze, die mit Hilfe der Funktion $\tau\colon U \to \mathbb{N}$ dargestellt wird.
In der Einleitung wurde erwähnt, dass sich die verschiedenen Varianten des *SSP* unter anderem bezüglich ihrer grundlegenden Struktur unterscheiden. Ein Beispiel hierfür ist die Zuteilung der Studierenden zu Modulen. Es gibt einerseits die Möglichkeit das *SSP* curriculum-basiert zu definieren; das bedeutet, dass jeder Studierende an allen Modulen teilnimmt, die für das Semester, in dem sich der Studierende befindet, angeboten werden. In der Realität ist es jedoch oft nicht so, da einige Studierende aufgrund von privaten Randbedingungen, nicht die Zeit haben an allen vorgesehenen Modulen teilzunehmen. In den höheren Semester ist es oft der Fall, dass es sich bei den vorgesehenen Modulen um Wahlpflichtfächer handelt, das heißt, die Studierenden können sich aus einer Menge vorgegebener Wahlpflichtfächer diejenigen aussuchen, an denen sie teilnehmen möchten. In Folge dessen wurde das *SSP* nicht curriculum-basiert definiert, so dass jeder Studierende die Möglichkeit hat seine gewünschten Module frei zu wählen. In der Eingabe wird dies mit der Funktion $\beta\colon S \to 2^M$ angegeben.
Die vorher stattgefundene Stundenplanung hat den Übungsgruppen neben den Räumen auch Zeitfenster zugeordnet. Da es in der Praxis häufig Module gibt, deren Übungsgruppen an mehreren Terminen stattfinden, wird in dieser Variante des *SSP* die Zuordnung einer Übungsgruppe nicht auf ein Zeitfenster beschränkt. Für die Studierendenverteilung sind die genauen Zeitfenster der Übungsgruppen irrelevant, denn sie benötigt ausschließlich die Information welche Übungsgruppen sich zeitlich überschneiden. Daher existiert in der Eingabe eine Funktion $\delta\colon U \times U \to \{0, 1\}$, die für je zwei Module angibt, ob diese ein gemeinsames Zeitfenster haben (1, falls ja, sonst 0). Da jede Übungsgruppe mehrere Termine haben darf, ist diese Funktion nicht transitiv.
In großen Universitäten mit einer sehr großen Anzahl von Studierenden ist es oft unmöglich alle Studierende auf alle gewählten Modulen zu verteilen, ohne dabei einen Konflikt zu erzeugen. Daher ist es sinnvoll eine Grenze angeben zu können, so dass die Anzahl der unerfüllten Studierendenwünsche begrenzt wird. Falls das *SSP* eine derartige Grenze enthält und zu einer Instanz x keine konfliktfreie Zuordnung der Studierenden zu Übungsgruppen findet, die diese Grenze einhält, ist die Lösung $x \notin SSP$. In diesem Fall haben die zuständigen Personen der Hochschulen die Möglichkeit die Grenze k etwas abzuschwächen oder beispielsweise zu prüfen, ob das *SSP* basierend auf einem neuen Stundenplan *true* liefert. Zur Repräsentation der Grenze wird in der Eingabe die Zahl $k \in \mathbb{N}$ verwendet, das heißt k gibt die

Anzahl der Studierenden/Modul-Kombinationen, die unverteilt bleiben dürfen, an. Eine Studierende/Modul-Kombination ist ein Tupel, das einerseits einen Studierenden enthält und andererseits ein Modul beinhaltet, welches dieser Studierende gewählt hat. Eine solche Kombination gilt als unverteilt, falls einem Studierenden keine Übungsgruppe des zugehörigen Moduls konfliktfrei zugeordnet werden kann. Die in der Definition enthaltene Menge A ist die Menge aller Studierenden/Modul-Kombinationen.
Wie schon erwähnt existieren Bedingungen für die Zuordnung von Studierenden zu Übungsgruppen, wie zum Beispiel, dass keine Terminkonflikte existieren dürfen. Die Vielfalt an unterschiedlichen Bedingungen für die Studierendenverteilung bildet einen weiteren Aspekt in dem sich die Varianten des *SSP* unterscheiden. Sie werden oft unterschieden in weiche und harte Bedingungen bzw. Kriterien. Weiche Kriterien sind solche, die nicht erfüllt werden müssen, damit eine Lösung zulässig ist. Eine Lösung ist jedoch besser, je mehr weiche Kriterien sie erfüllt. Die Lösungen werden somit hinsichtlich der weichen Kriterien optimiert. Für die Studierenden ist es üblicherweise angenehmer, wenn sich die Anzahl ihrer Übungsgruppen, welche sie pro Tag belegen, ausbalanciert ist. Daher eignet sich beispielsweise diese Anforderung als ein weiches Kriterium. Es gibt auch harte Bedingungen, das sind solche, die zwangsläufig erfüllt werden müssen, ansonsten ist eine Lösung unzulässig. Das hier analysierte *SSP* beinhaltet ausschließlich harte Kriterien.
Die Zuordnung wird als eine Relation dargestellt, die bis auf maximal k Studierende/Modul-Kombinationen jeder dieser Kombinationen genau eine Übungsgruppe zuweist. Bei der Zuweisung muss beachtet werden, dass die zugewiesene Übungsgruppe dem Modul der entsprechenden Studierenden/Modul-Kombination zugehört. Dieses harte Kriterium wird in der Definition des *SSP* als erste Bedingung aufgeführt.
Die zweite Bedingung besagt, dass nicht mehr Studierende/Modul-Kombinationen einer Übungsgruppe zugewiesen werden dürfen, als deren Kapazitätsgrenze zulässt. Es wurde bereits erwähnt, dass bei der Zuweisung auf Terminkonflikte zu achten ist. Es dürfen nicht mehrere Studierende/Modul-Kombinationen eines Studierenden Übungsgruppen zugewiesen werden, die sich zeitlich überschneiden. Dieses Kriterium wird durch die dritte Bedingung repräsentiert. Die vierte Bedingung besagt, dass jeder Studierende maximal einer Übungsgruppe pro gewähltes Modul zugewiesen werden darf, daher ist es nicht erlaubt, dass eine Studierende/Modul-Kombination mehreren Übungsgruppen eines Moduls zugeordnet wird.
Eine Zuteilung ist somit zulässig, wenn alle der soeben beschriebenen Bedingungen erfüllt werden.

Die Definition des *SSP* wurde so gewählt, dass sie sehr praxisnah ist, aber auf nicht zwangsläufig erforderliche Bedingungen verzichtet, denn je mehr Kriterien zu berücksichtigen sind, desto schwieriger wird das Lösen dieses Problems. Ein Studierender kann nicht zeitgleich an mehreren Übungsgruppen teilnehmen, daher ist die dritte Bedingung zwingend erforderlich, sowie auch die zweite Bedingung zur Einhaltung der Kapazitätsgrenzen und die anderen beiden Bedingungen unumgänglich sind.

Es gibt Varianten mit einer zusätzlichen Bedingung, die besagt, dass ein Studierender nur Übungsgruppen zugeteilt werden darf, an deren Terminen der Studierende verfügbar ist. Hierzu enthält die Eingabe weiterhin die verfügbaren Termine jedes Studierenden. Dieses Kriterium ist zum Beispiel nicht zwingend erforderlich, da ein Studium eine Vollzeit-Beschäftigung ist und man daher erwarten kann, dass die Studierenden zeitlich immer verfügar sind. Die Bedingungen, welche nicht zwangsläufig die Zulässigkeit der Zuordnungen beeinflussen, sondern nur die Zuordnungen verbessern, wie beispielsweise eine ausbalancierte Anzahl an täglichen Übungen pro Studierender, werden hier außen vor gelassen.
In dieser Arbeit wird die parametrisierte Komplexität des soeben definierten *SSP* analysiert und das nachfolgende Theorem bewiesen. Der Parameter p steht für die Anzahl insgesamter Übungsgruppe, c für die maximale Kapazität der Übungsgruppen, u gibt die maximale Anzahl von Übungsgruppen pro Modul an, t ist die maximale Anzahl Termine jeder Übungsgruppe und s steht für die insgesamte Anzahl von Studierenden.

Theorem 1. *Das parametrisierte SSP ist in FPT, falls die Parametrisierung p enthält und para-NP vollständig, falls die Parameter in $\{k, c, u, t\}$ enthalten sind. Des Weiteren gilt für $k > 0$, dass die parametrisierten SSP, deren Parameter eine Teilmenge aus $\{c, u, t, s\}$ bilden, para-NP vollständig sind.*

3

Komplexität

Der Hauptaspekt dieser Arbeit ist die Analyse der parametrisierten Komplexität des *SSP*. Auf Basis der Erkenntnisse der klassischen, nicht-parametrisierten Komplexität des *SSP* lassen sich Rückschlüsse auf die parametrisierte Komplexität ziehen. Aus diesem Grund wird in diesem Kapitel die klassische Komplexität des *SSP* betrachtet.

Lemma 3.1. SSP *ist NP-schwer.*

Beweis. Hierzu wird eine *Polynomialzeit-Many-ony-Reduktion* von *3-SAT* auf *SSP* angegeben. Da das bekannte *3-SAT* ein NP-vollständiges Problem ist, folgt daraus die NP-Härte des *SSP*. Das zu reduzierende Entscheidungsproblem ist wie folgt definiert:

Definition 2. *3-Satisfiability(3-SAT):*

Eingabe: $x = (C, L)$ *mit* $\forall c_i \in C : c_i$ *ist in 3-KNF mit den Literalen aus* L

Frage: *Existiert eine Belegung* $\beta \colon L \to \{0, 1\}$*, so dass alle Klauseln* $c_i \in C$ *erfüllt werden?*

3.1 Beispiel für eine Reduktion

Einleitend wird die *Polynomialzeit-Many-one-Reduktion* von *3-SAT* auf *SSP* mit einem Beispiel vorgestellt, bevor die Reduktion im nächsten Abschnitt allgemein und formal definiert wird. Zur einfachen Veranschaulichung dieser Reduktion wird eine kleine *3-SAT*-Instanz gewählt:

$$\text{Sei } x = (C, L) \text{ mit } C = \{v \vee y \vee \overline{z}, v \vee \overline{y} \vee z\} \text{ und } L = \{v, y, z\}.$$

Hieraus lässt sich folgende Instanz für SSP formulieren:
Für jedes Literal und für jede Klausel wird ein Modul erzeugt. Somit gilt: $M = \{m_1, m_2, m_3, m_4, m_5\}$, wobei das Modul m_1 für das Literal v steht, m_2 für

das Literal y, m_3 für das Literal z, m_4 für die Klausel $(v \vee y \vee \overline{z})$ und m_5 für die Klausel $(v \vee \overline{y} \vee z)$.
Die Menge der Studierenden S beinhaltet für jedes Literal einen Studierenden, so dass gilt $S = \{s_1, s_2, s_3\}$ mit s_1 für das Literal v, s_2 für das Literal y und s_3 für das Literal z.
Jedes Modul besitzt zwei Übungsgruppen und demnach gilt $U = \bigcup_{1 \leq i \leq 5} \{u_{i1}, u_{i2}\}$.
Die Übungsgruppe u_{i1} repräsentiert das Literal bzw. die Klausel, für die das Modul m_i steht und u_{i2} repräsentiert das entsprechende negierte Literal bzw. die entsprechende negierte Klausel.
In der nachfolgenden Abbildung 3.1 wird die Erstellung von M, S, U und k basierend auf C und L graphisch dargestellt.

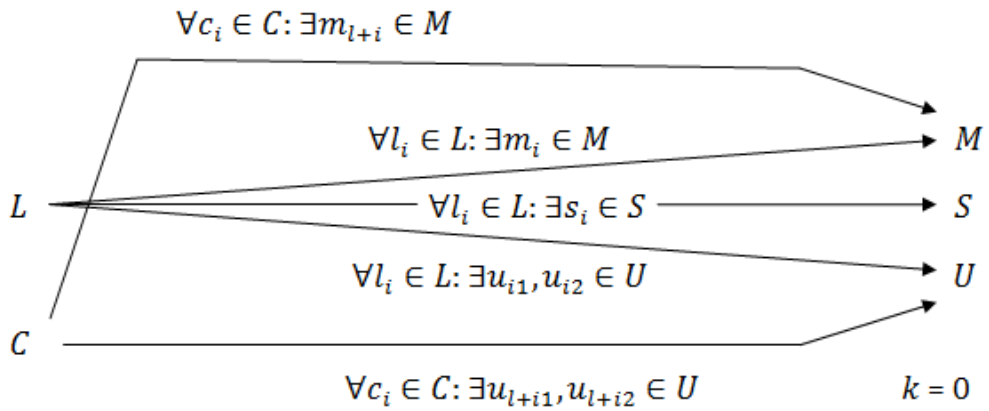

Abbildung 3.1. Die teilweise erstellte Instanz für *SSP*

Die Kapazität der Übungsgruppen, deren Modul für ein Literal steht, ist 1. Somit gilt $\forall u_{ij} \in U : i \in \{1, 2, 3\} \rightarrow \tau(u_{ij}) = 1$. Für die restlichen Übungsgruppen gilt, dass pro Modul die erste Übungsgruppe die Kapazität 3 hat und die zweite Übungsgruppe die Kapazität 2 hat. Demnach ist τ wie folgt definiert:

$$\forall u_{ij} \in U : \tau(u_{ij}) = \begin{cases} 1, & \text{wenn } 1 \leq i \leq 3 \\ 3, & \text{wenn } i > 3 \text{ und } j = 1 \\ 2, & \text{sonst} \end{cases}$$

Gemäß den bisherigen Zuordnungen gilt, dass jeder Studierende $s_i \in S$ das Literal $l_i \in L$ repräsentiert. Jeder Studierende $s_i \in S$ wählt das Modul, welches ebenfalls für l_i steht (m_i) und jedes Modul, deren zugehörige Klausel l_i oder $\overline{l_i}$ enthält. Demnach gilt $\forall s_i \in S : \beta(s_i) = \{m_i\} \cup \{m_{l+j} | m_{l+j} \in M \wedge (l_i \in c_j \vee \overline{l_i} \in c_j)\}$.
Die Terminkonflikte werden in der nachfolgenden Abbilung 3.2 als ein Graph dargestellt, dessen Knoten jeweils eine Übungsgruppe aus U repräsentieren.

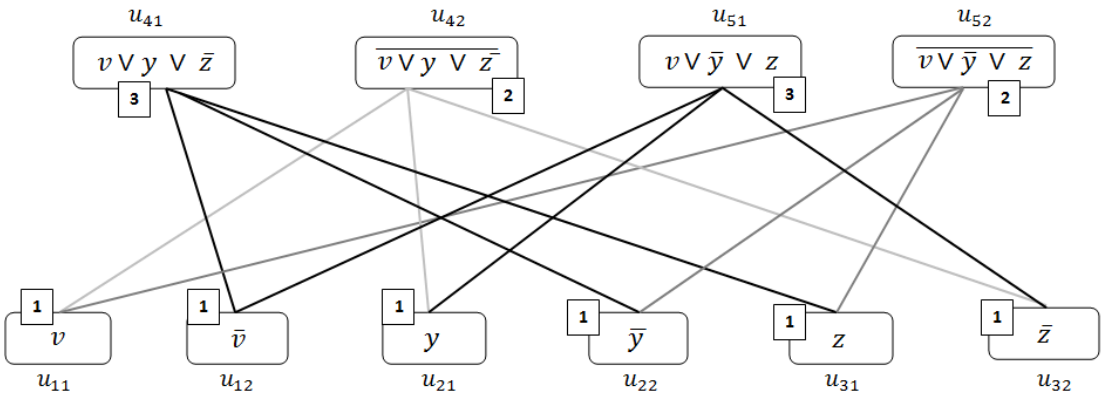

Abbildung 3.2. Beispiel einer Reduktion von *3-SAT* auf *SSP*

Für je zwei Übungsgruppen u_{ij}, u_{rs} gilt, dass sie mindestens einen gemeinsamen Termin haben, wenn ihre zugehörigen Knoten durch eine Kante verbunden sind, ansonsten $\delta(u_{ij}, u_{rs}) = 0$.
Weiterhin wird für diese Reduktion festgelegt: $k = 0$, das heißt es sind alle Studierenden/Modul-Kombinationen zu verteilen.
$f(x)$ ist die aus x resultierende Instanz für *SSP*.
Laut der Definition der *Polynomialzeit-Many-one-Reduzierbarkeit*[Sch11] muss unter anderem gelten $x \in$ *3-SAT* $\Leftrightarrow f(x) \in SSP$.

Die Intuition, auf welcher diese Reduktion basiert, wird nachfolgend erklärt. Die Übungsgruppen, die für ein Literal stehen, haben als mögliche Teilnehmer nur den Studierenden, welcher das gleiche Literal repräsentiert. Wird ein Studierender der Übungsgruppe v zugeordnet, so bedeutet dies für das *3-SAT*, dass v mit *wahr* bzw. 1 belegt wird. Analog dazu wird einem Literal v die 0 zugewiesen, falls ein Studierender der Übungsgruppe $\overline{v}$ zugeteilt wird.
Neben diesen Übungsgruppen für die Literale existieren zwei Übungsgruppen für jede Klausel. Eine dieser Übungsgruppen steht genau für die entsprechende Klausel und die andere für die negierte Klausel. Die möglichen Teilnehmer einer solchen Übungsgruppe sind die Studierenden, deren zugehöriges Literal in der entsprechenden Klausel enthalten ist. Somit gibt es für jede derartige Übungsgruppe genau drei Studierende, die als Teilnehmer in Frage kommen. Die Terminkonflikte sind so verteilt, dass einer derartigen Übungsgruppe nur Studierende zugeteilt werden dürfen, deren Belegung des Literals für das Erfüllen der zugehörigen Klausel beiträgt. Somit darf beispielsweise der Übungsgruppe für die Klausel $v \vee y \vee \overline{z}$ der Studierende des Literals v nur dann zugeordnet werden, wenn die Belegung von v 1 ist bzw. der Studierende der Übung v zugeordnet wurde. Dementsprechend kann der Studierende des Literals z der Übung u_{41} nur dann zugeordnet werden, wenn die Belegung von z 0 ist bzw. der Studierende ein Teilnehmer der Übung $\overline{z}$ ist.
Falls $x \in$ *3-SAT* müssen alle Klauseln erfüllt werden und somit müssen die Übungsgruppen die eine Klausel aus C repräsentieren mindestens einen Teilnehmer haben.

Hieraus ergibt sich, dass jede Übung, welche eine negierte Klausel aus C repräsentiert, nur maximal zwei Teilnehmer haben darf. Hätte beispielsweise die Übung u_{42} drei Teilnehmer, würde dies bedeuten, dass an der Übungsgruppe u_{41} kein Studierender teilnimmt und somit wären v und y mit 0 und z mit 1 belegt. Folglich wäre die Klausel von u_{41} nicht erfüllt und es würde gelten $x \notin$ *3-SAT*.

Für dieses Beispiel wäre $\varphi(v) = 1, \varphi(y) = 0, \varphi(z) = 0$ eine zulässige Belegung, die alle Klauseln aus C erfüllt. Aus der obigen Definition der Reduktionsfunktion entsteht die Relation $R = \{((s_1, m_1), u_{11}), ((s_1, m_4), u_{41}), ((s_1, m_5), u_{51}), ((s_2, m_2), u_{22}), ((s_2, m_4), u_{41}), ((s_2, m_5), u_{52}), ((s_3, m_3), u_{32}), ((s_3, m_4), u_{42}), ((s_3, m_5), u_{51})\}$, welche alle Bedingungen des *SSP* erfüllt.
Somit gilt für dieses Beispiel $x \in$ *3-SAT* und $f(x) \in$ *SSP*.

3.2 Reduktion

Die in dem vorherigen Beispiel beschriebene Reduktion wird nachfolgend allgemein und formal angegeben:

Lemma 3.2. *3-SAT* $\leq_m^p$ *SSP*.

Beweis. Sei x eine Instanz für *3-SAT* und $c = \#C$ und $l = \#L$.

Die Reduktionsfunktion f ist wie folgt definiert:

Sei $f(x) =_{def} (M, S, U, \tau, \beta, \delta, k)$ mit

$M = \{m_1, ..., m_{c+l}\}$

$S = \{s_1, ..., s_l\}$

$U = \bigcup_{m_i \in M} \{u_{i1}, u_{i2}\}$

$\tau(u_{ij})$ gibt die Kapazität der Übungsgruppe u_{ij} an.

$$\tau(u_{ij}) = \begin{cases} 1, & \text{wenn } 1 \leq i \leq l \\ 3, & \text{wenn } i > l \text{ und } j = 1 \\ 2, & \text{sonst} \end{cases} \qquad \text{für } \forall u_{ij} \in U$$

$\beta(s_i)$ ist die Menge der Module, die der Studierende s_i wählte.

$\beta(s_i) = \{m_i\} \cup \{m_{l+j} | m_{l+j} \in M \wedge (l_i \in c_j \vee \overline{l_i} \in c_j)\}$ für $s_i \in S$

$\forall u_{ij} \in U : p_{ij}$ enthält die Termine von u_{ij}, die in Kollision mit den Terminen einer anderen Übungsgruppe stehen.

$T = \{t_1, ..., t_{3 \cdot c}\}$ ist eine Menge von Zeitfenstern.

Im Folgenden wird für alle $1 \leq i \leq \#M$ und $j \in \{1, 2\}$ p_{ij} berechnet:

Sei $q := 0$

Sei $p_{11} = ... = p_{\#M2} = \emptyset$

$\forall 1 \leq i \leq l :$

$\forall 1 \leq j \leq c :$

falls $\overline{l_i} \in c_j$ dann

$p_{i1} := p_{i1} \cup \{t_{q+1}\}$

$p_{(l+j)1} := p_{(l+j)1} \cup \{t_{q+1}\}$

$p_{i2} := p_{i2} \cup \{t_{q+2}\}$

$p_{(l+j)2} := p_{(l+j)2} \cup \{t_{q+2}\}$

$q := q + 2$

sonst falls $l_i \in c_j$ dann

$p_{i2} := p_{i2} \cup \{t_{q+1}\}$

$p_{(l+j)1} := p_{(l+j)1} \cup \{t_{q+1}\}$

$p_{i1} := p_{i1} \cup \{t_{q+2}\}$

$p_{(l+j)2} := p_{(l+j)2} \cup \{t_{q+2}\}$

$q := q + 2$

$$\forall u_{ib}, u_{jd} \in U : \delta(u_{ib}, u_{jd}) = \begin{cases} 0, & \text{wenn } p_{ib} \cap p_{jd} = \emptyset \\ 1, & \text{sonst} \end{cases}$$

$k := 0$

3.2.1 Korrektheitsbeweis

Es existiert eine *Polynomialzeit-Many-one-Reduktion* von *3-SAT* auf *SSP*, falls die soeben definierte Reduktionsfunktion f in Polynomialzeit berechenbar ist und für alle x gilt $x \in$ *3-SAT* $\Leftrightarrow f(x) \in$ *SSP*. Laut obiger Definition der Funktion f gilt, dass f in Polynomialzeit berechenbar ist. Es ist noch zu zeigen, dass für alle x gilt $x \in$ *3-SAT* $\Leftrightarrow f(x) \in$ *SSP*, was nachfolgend bewiesen wird.

1. Beweis für $x \in$ *3-SAT* $\Rightarrow f(x) \in$ *SSP*.

   Sei $x \in$ *3-SAT*.

   $\Rightarrow$ es existiert eine Belegungsfunktion $\varphi\colon L \to \{0, 1\}$, so dass alle $c_i \in C$ erfüllt werden.

   Basierend auf φ wird für die *SSP*-Instanz $f(x)$ eine Relation R erstellt:

$R = \emptyset$
foreach $s_j \in S$
  foreach $m_k \in \beta(s_j)$
    if $l_j \in c_{k-l} \vee k = j$ **then**
      if $\varphi(l_j) = 1$ **then**
        $R := R \cup \{((s_j, m_k), u_{k1})\}$
      else
        $R := R \cup \{((s_j, m_k), u_{k2})\}$
    else $\overline{l_j} \in c_{k-l}$
      if $\varphi(l_j) = 0$ **then**
        $R := R \cup \{((s_j, m_k), u_{k1})\}$
      else
        $R := R \cup \{((s_j, m_j), u_{k2})\}$
  endfor
endfor

Da für alle $((s_j, m_k), u_{ki}) \in R$ gilt, dass $m_k \in \beta(s_j)$, folgt daraus, dass $(s_j, m_k) \in A$. Weiterhin gilt, dass für alle $m_q \in M$ die Übungsgruppen u_{q1} und u_{q2} in U existieren, woraus folgt $R \subseteq A \times U$.

Für alle $m_k \in \beta(s_j)$ gilt, dass $l_j \in c_{k-l} \vee \overline{l_j} \in c_{k-l} \vee k = j$ laut Reduktionsfunktion. Somit gilt für alle $m_k \in \beta(s_j)$, dass entweder $((s_j, m_k), u_{k1})$ oder $((s_j, m_k), u_{k2})$ in R hinzugefügt wird. Hieraus ergibt sich, dass für jedes $m_k \in \beta(s_j)$ genau ein Element in R existiert und folglich gilt $\#R = \#A$ und somit auch $\#R \geq \#A - k$.
Falls die erstellte Relation R alle vier Bedingungen des *SSP* erfüllt, dann gilt $f(x) \in SSP$. Somit ist zu zeigen, dass R eine zulässige Relation für das *SSP* ist.

Beweis für R erfüllt alle vier Bedingungen des *SSP*:

1. Bedingung: $\forall((s_i, m_k), u_{jn}) \in R : j = k$

   (u_{jn} muss Übungsgruppe von m_k sein)

   Direkter Beweis:

   Laut obiger Zuweisung gilt: $\forall((s_i, m_k), u_{jn}) \in R : j = k \wedge (n = 1 \vee n = 2)$

   $\Rightarrow \forall((s_i, m_k), u_{jn}) \in R : j = k$

   Somit gilt, dass alle Elemente aus R die erste Bedingung erfüllen.

2. Bedingung:

   $\#S' \leq \tau(u_{kj})$ mit $S' = \{s_i | s_i \in S \wedge ((s_i, m_k), u_{kj}) \in R\}$ für $u_{kj} \in U$

   Widerspruchsbeweis:

   Angenommen Bedingung 2 wird nicht erfüllt.

   $\Rightarrow \exists u_{kj} \in U : \#S' > \tau(u_{kj})$

   i. Sei $k \leq l$.

      $\Rightarrow \tau(u_{kj}) = 1$

      Da $k \leq l$ gilt $\{s_i | s_i \in S \wedge m_k \in \beta(s_i)\} = \{s_k\}$ laut Reduktionsfunktion.

      $$\begin{aligned} &\Rightarrow \#\{s_i | s_i \in S \wedge (s_i, m_k) \in A\} && = 1 \\ &\Rightarrow \#\{s_i | s_i \in S \wedge ((s_i, m_k), u_{kj}) \in R\} && \leq 1 \\ &\Rightarrow \#S && \leq 1 \\ &\Rightarrow \#S && \leq \tau(u_{kj}) \end{aligned}$$

      Dies ist ein Widerspruch zur Behauptung $\#S > \tau(u_{kj})$ und daher gilt, dass für alle $u_{kj} \in U$ mit $k \leq l$ die 2. Bedingung des *SSP* erfüllt wird.

   ii. Sei $k > l$.

      A. Angenommen $j = 2$:

         $\Rightarrow \tau(u_{kj}) = 2$

         Laut Definition von β gilt: $m_k \in \beta(s_i) \Rightarrow l_i \in c_{k-l} \vee \overline{l_i} \in c_{k-l} \vee k = i$

         Da $\#S \leq l$ und $k > l$ gilt $m_k \in \beta(s_i) \Rightarrow l_i \in c_{k-l} \vee \overline{l_i} \in c_{k-l}$

         $\Rightarrow \forall (s_i, m_k) \in A \backslash \{(s_i, m_i)\} : l_i \in c_{k-l} \vee \overline{l_i} \in c_{k-l}$

            (laut Definition von A)

         Da $j = 2$ gilt laut Zuordnung von R:

         $\forall ((s_i, m_k), u_{kj}) \in R \backslash \{((s_i, m_i), u) | u \in \{u_{i1}, u_{i2}\}\}$:

            $(l_i \in c_{k-l} \Rightarrow \varphi(l_i) = 0) \vee (\overline{l_i} \in c_{k-l} \Rightarrow \varphi(l_i) = 1)$

         Laut der Beweisvoraussetzung des Widerspruchsbeweises gilt:

$\#S' > \tau(u_{kj})$:

$\#\{s_i | s_i \in S \wedge ((s_i, m_k), u_{kj}) \in R\} > 2$

$\Rightarrow \#\{s_i | (s_i, m_k) \in A \wedge ((l_i \in c_{k-l} \Rightarrow \varphi(l_i) = 0) \vee (\overline{l_i} \in c_{k-l} \Rightarrow \varphi(l_i) = 1))\} \geq 3$

$\Rightarrow \#\{s_i | (l_i \in c_{k-l} \Rightarrow \varphi(l_i) = 0) \vee (\overline{l_i} \in c_{k-l} \Rightarrow \varphi(l_i) = 1)\} \geq 3$

$\Rightarrow \#\{l_i | (l_i \in c_{k-l} \Rightarrow \varphi(l_i) = 0) \vee (\overline{l_i} \in c_{k-l} \Rightarrow \varphi(l_i) = 1)\} \geq 3$

Da c_i in *3-KNF* ist, gilt $\#\{l_j | l_j \in c_i \vee \overline{l_j} \in c_i\} = 3$

$\Rightarrow \forall l_i \in c_{k-l} : \varphi(l_i) = 0 \wedge \forall \overline{l_i} \in c_{k-l} : \varphi(l_i) = 1$

$\Rightarrow c_{k-l}$ wird nicht erfüllt.

Da $k > l$ gilt $k - l > 0$ und da $m_k \in M$ gilt $k \leq l + c$.
Hieraus folgt für $k - l$ das gilt $0 < k - l \leq c$ und somit gilt $c_{k-l} \in C$.
Dies führt zu einem Widerspruch, denn für alle c_i aus C gilt, dass c_i erfüllt wird und daher erfüllen alle Elemente aus R mit
$u_{kj} \in U \wedge k > l \wedge k$ ist $j = 2$ die zweite Bedingung.

B. Angenommen $j = 1$:
$\Rightarrow \tau(u_{kj}) = 3$
Da Bedingung 2 für u_{kj} nicht erfüllt wird, gilt:

$\#\{s_i | s_i \in S \wedge ((s_i, m_k), u_{kj}) \in R\} > 3$

$\Rightarrow \#\{s_i | s_i \in S \wedge m_k \in \beta(s_i)\} > 3$

(sonst wäre Bedingung 4 des *SSP* verletzt)

$\Rightarrow \#\{s_i | s_i \in S \wedge (l_i \in c_{k-l} \vee \overline{l_i} \in c_{k-l})\} > 3$

$\Rightarrow \#\{l_i | l_i \in c_{k-l} \vee \overline{l_i} \in c_{k-l}\} > 3$

Dies führt zu einem Widerspruch, da laut Definition 2 des *3-SAT* alle c_i aus C in *3-KNF* sind.
Somit wird die 2. Bedingung für alle Elemente $((s_i, m_k), u_{kj}) \in R$ mit $k > l$ und $j = 1$ erfüllt.
Folglich erfüllen alle Elemente aus R die 2. Bedingung.

3. Bedingung:
$\forall((s_i, m_k), u_{kj}), ((s_i, m_n), u_{nj'}) \in R : m_k \neq m_n \Rightarrow \delta(u_{kj}, u_{nj'}) = 0$

Angenommen Bedingung 3 wird nicht erfüllt.

$\Rightarrow \exists((s_i, m_k), u_{kj}), ((s_i, m_n), u_{nj'}) \in R : m_k \neq m_n \wedge \delta(u_{kj}, u_{nj'}) = 1$

Angenommen für $((s_i, m_k), u_{kj}), ((s_i, m_n), u_{nj'}) \in R$ wird die dritte Bedingung verletzt.

$$\Rightarrow \delta(u_{kj}, u_{nj'}) = 1$$
$$\Rightarrow p_{kj} \cap p_{nj'} \neq \emptyset$$
$$\Rightarrow \exists t \in T : t \in p_{kj} \wedge t \in p_{nj'}$$
$$\Rightarrow (k = i \wedge n > l) \vee (n = i \wedge k > l)$$

Die letzte Implikation erfolgt auf Basis der Belegung von p.

oBdA: Sei $k = i \wedge n > l$.

Es gilt $((s_i, m_n), u_{nj'}) \in R$.

$$\Rightarrow (s_i, m_n) \in A$$
$$\Rightarrow m_n \in \beta(s_i)$$
$$\Rightarrow n = i \vee l_i \in c_{n-l} \vee \overline{l_i} \in c_{n-l}$$

Da $i \leq l$ und $n > l$ gilt $n \neq i$, woraus folgt $l_i \in c_{n-l} \vee \overline{l_i} \in c_{n-l}$.

i. Sei $l_i \in c_{n-l}$.
   $\Rightarrow p_{i2} \cap p_{n1} \neq \emptyset \wedge p_{i1} \cap p_{n2} \neq \emptyset \wedge p_{i1} \cap p_{n1} = \emptyset \wedge p_{i2} \cap p_{n2} = \emptyset$
   (laut Belegung von p)

   A. Sei $j = 1$.
      Da $p_{kj} \cap p_{nj'} \neq \emptyset \wedge j = 1 \wedge k = i \wedge p_{i1} \cap p_{n2} \neq \emptyset \wedge p_{i1} \cap p_{n1} = \emptyset$ gilt $j' = 2$.
      Weil $((s_i, m_k), u_{k1}) \in R \wedge k = i$ gilt $\varphi(l_i) = 1$.
      Da $((s_i, m_n), u_{n2}) \in R \wedge l_i \in c_{n-l}$ gilt $\varphi(l_i) = 0$.
      Dies führt zu einem Widerspruch und daher gilt $p_{kj} \cap p_{nj'} = \emptyset$, woraus folgt $\delta(u_{kj}, u_{nj'}) = 0$.

   B. Sei $j = 2$.
      Da $p_{kj} \cap p_{nj'} \neq \emptyset \wedge j = 2 \wedge k = i \wedge p_{i2} \cap p_{n1} \neq \emptyset \wedge p_{i2} \cap p_{n2} = \emptyset$ gilt $j' = 1$.
      Weil $((s_i, m_k), u_{k2}) \in R \wedge k = i$ gilt $\varphi(l_i) = 0$.
      Da $((s_i, m_n), u_{n1}) \in R \wedge l_i \in c_{n-l}$ gilt $\varphi(l_i) = 1$.

Dies führt zu einem Widerspruch und daher gilt $p_{kj} \cap p_{nj'} = \emptyset$, woraus folgt $\delta(u_{kj}, u_{nj'}) = 0$.

ii. Sei $\overline{l_i} \in c_{n-l}$.
$\Rightarrow p_{i1} \cap p_{n1} \neq \emptyset \wedge p_{i2} \cap p_{n2} \neq \emptyset \wedge p_{i1} \cap p_{n2} = \emptyset \wedge p_{i2} \cap p_{n1} = \emptyset$
(laut Belegung von p)

A. Sei $j = 1$.
Da $p_{kj} \cap p_{nj'} \neq \emptyset \wedge j = 1 \wedge k = i \wedge p_{i1} \cap p_{n1} \neq \emptyset \wedge p_{i1} \cap p_{n2} = \emptyset$ gilt $j' = 1$.
Weil $((s_i, m_k), u_{k1}) \in R \wedge k = i$ gilt $\varphi(l_i) = 1$.
Da $((s_i, m_n), u_{n1}) \in R \wedge \overline{l_i} \in c_{n-l}$ gilt $\varphi(l_i) = 0$.
Dies führt zu einem Widerspruch und daher gilt $p_{kj} \cap p_{nj'} = \emptyset$, woraus folgt $\delta(u_{kj}, u_{nj'}) = 0$.

B. Sei $j = 2$.
Da $p_{kj} \cap p_{nj'} \neq \emptyset \wedge j = 2 \wedge k = i \wedge p_{i2} \cap p_{n2} \neq \emptyset \wedge p_{i2} \cap p_{n1} = \emptyset$ gilt $j' = 2$.
Weil $((s_i, m_k), u_{k2}) \in R \wedge k = i$ gilt $\varphi(l_i) = 0$.
Da $((s_i, m_n), u_{n2}) \in R \wedge \overline{l_i} \in c_{n-l}$ gilt $\varphi(l_i) = 1$.
Dies führt zu einem Widerspruch und daher gilt $p_{kj} \cap p_{nj'} = \emptyset$, woraus folgt $\delta(u_{kj}, u_{nj'}) = 0$.

Folglich gilt, dass R die 3. Bedingung ebenfalls erfüllt.

4. Bedingung: $\forall (a_b, u_{kj}), (a_b, u_{k'j'}) \in R : u_{kj} = u_{k'j'}$ (kein Element aus A darf mehreren Übungsgruppen zugewiesen werden)
Da beim Erstellen von R pro Element aus A genau ein Durchlauf der inneren Schleife erfolgt und in diesem Durchlauf genau ein Element in R hinzugefügt wird, erfüllt R diese Bedingung.
Da gezeigt wurde, dass R die Bedingungen $1, 2, 3$ und 4 des *SSP* erfüllt und $R \subseteq A \times U$ ist mit $\#R \geq \#A - k$ gilt $x \in SSP$.

2. Beweis für $f(x) \in SSP \Rightarrow x \in$ *3-SAT*

Sei $f(x) \in SSP$.

$\Rightarrow$ es existiert eine zulässige Relation R mit $\#R \geq \#A - k$.

$\Rightarrow$ es existiert eine zulässige Relation R mit $\#R = \#A$ (da $k = 0$).

Basierend auf der Relation R wird für die *3-SAT*-Instanz x eine Belegung

$\varphi\colon L \to \{0,1\}$ erstellt:

$$\forall l_i \in L : \varphi(l_i) = \begin{cases} 1, & \text{wenn } ((s_i, m_i), u_{i1}) \in R \\ 0, & \text{sonst } ((s_i, m_i), u_{i2}) \in R \end{cases}$$

Angenommen, es gilt $\exists l_i \in L : ((s_i, m_i), u_{i1}), ((s_i, m_i), u_{i2}) \notin R$

$$\Rightarrow \forall u_{ij} \in U : ((s_i, m_i), u_{ij}) \notin R$$

$$\Rightarrow \forall ((s_e, m_f), u_{fg}) \in R : (s_e, m_f) \neq (s_i, m_i)$$

Da laut Bedingung 4 des *SSP* gilt, dass jedes Element aus A nur maximal einmal in R enthalten ist und jedes Element aus R ein Element aus A enthält, gilt somit $\#R < \#A$, da $(s_i, m_i) \in A$.
Dies führt zu einem Widerspruch, da für alle l_i aus L gilt:

$$(((s_i, m_i), u_{i1}) \in R \wedge ((s_i, m_i), u_{i2}) \notin R) \vee (((s_i, m_i), u_{i1}) \notin R \wedge ((s_i, m_i), u_{i2}) \in R)$$

Es ist zu zeigen, dass alle Klauseln mit dieser Belegung erfüllt werden.

Widerspruchsbeweis:

Angenommen $\exists c_i \in C : c_i$ wird nicht erfüllt.

Sei $c_i = (v \vee y \vee z)$, da c_i in *3-KNF*.

Laut der Definition 2 des *3-SAT* gilt: c_i ist in *3-KNF* mit den Literalen aus L.

Somit gilt:

- $\exists l_b \in L : v = l_b \vee v = \overline{l_b}$
  Falls $v = l_b$ dann $\varphi(l_b) = 0$, da sonst c_i erfüllt wäre, ansonsten $\varphi(l_b) = 1$.
- $\exists l_c \in L : y = l_c \vee y = \overline{l_c}$
  Falls $y = l_c$ dann $\varphi(l_c) = 0$, da sonst c_i erfüllt wäre, ansonsten $\varphi(l_c) = 1$.
- $\exists l_d \in L : z = l_d \vee z = \overline{l_d}$
  Falls $z = l_d$ dann $\varphi(l_d) = 0$, da sonst c_i erfüllt wäre, ansonsten $\varphi(l_d) = 1$.

a) Sei $v = l_b$.

$\Rightarrow ((s_b, m_b), u_{b2}) \in R$ (laut obiger Zuordnung von l_b)

Da $l_b \in c_i$ gilt $(s_b, m_{l+i}) \in A$ und aus $\#R = \#A$

folgt $\exists u_{b'd} \in U : ((s_b, m_{l+i}), u_{b'd}) \in R$.

Auf Grund der Zulässigkeit von R gilt: $b' = l + i$

($u_{b'd}$ ist Übungsgruppe von m_{l+i} laut Bedingung 1 des *SSP*).

Da $\forall m_n \in M : \{u_{nj} | u_{nj} \in U\} = \{u_{n1}, u_{n2}\}$, gilt $d \in \{1, 2\}$.

Weil $((s_b, m_b), u_{b2}) \in R \wedge p_{b2} \cap p_{(l+i)1} \neq \emptyset$ gilt $\delta(u_{b2}, u_{(l+i)1}) = 1$.

Falls $d = 1$ dann existiert ein Terminkonflikt in R.

Dies führt zu einem Widerspruch da R zulässig ist.

Somit gilt $d = 2$ und $((s_b, m_{l+i}), u_{(l+i)2}) \in R$.

b) Sei $v = \overline{l_b}$,

$\Rightarrow ((s_b, m_b), u_{b1}) \in R$ (laut obiger Zuordnung von l_b)

Da $\overline{l_b} \in c_i$ gilt $(s_b, m_{l+i}) \in A$ und aus $\#R = \#A$ folgt

$\exists u_{b'd} \in U : ((s_b, m_{l+i}), u_{b'd}) \in R$.

Auf Grund der Zulässigkeit von R gilt: $b' = l + 1$

($u_{b'd}$ ist Übungsgruppe von m_{l+1} laut Bedingung 1 des *SSP*).

Da $\forall m_n \in M : \{u_{nj} | u_{nj} \in U\} = \{u_{n1}, u_{n2}\}$, gilt $d \in \{1, 2\}$.

Weil $((s_b, m_b), u_{b1}) \in R \wedge p_{b1} \cap p_{(l+i)1} \neq \emptyset$ gilt $\delta(u_{b1}, u_{(l+i)1}) = 1$.

Falls $d = 1$ dann existiert ein Terminkonflikt in R.

Dies führt zu einem Widerspruch da R zulässig ist.

Somit gilt $d = 2$ und $((s_b, m_{l+i}), u_{(l+i)2}) \in R$.

Folglich gilt in beiden Fällen $((s_b, m_{l+i}), u_{(l+i)2}) \in R$.

Für l_c und l_d ist der Beweis analog und daher gilt ebenfalls

$((s_c, m_{l+i}), u_{(l+i)2}) \in R \wedge ((s_d, m_{l+i}), u_{(l+i)2}) \in R$.

Sei $S' = \{s | s \in S \wedge ((s, m_{l+i}), u_{(l+i)2}) \in R\} \Rightarrow \#S' \geq 3$.

Da $\tau(u_{(l+i)2}) = 2$ wird somit die Kapazität der Übungsgruppe $u_{(l+i)2}$ überschritten.

Dies führt zu einem Widerspruch da R zulässig ist und somit gilt für die Belegung φ, dass alle $c_i \in C$ erfüllt werden.

Folglich gilt $x \in$ *3-SAT*.

Es wurde gezeigt, dass die Reduktion $3 - SAT \leq_m^p SSP$ korrekt ist und somit gilt, dass *SSP NP-hart* ist.

Lemma 3.3. SSP $\in$ *NP*.

Beweis. Die Menge der möglichen Lösungen für *SSP* ist exponentiell groß, denn für jede Übungsgruppe existiert eine Teilmenge von Studierenden, die zugewiesen werden könnte und somit existieren pro Übungsgruppe maximal $2^{\#S}$ unterschiedliche Mengen von Studierenden, die als Teilnehmer in Frage kommen. Hieraus ergibt sich das es $(2^{\#S})^{\#U} = 2^{\#S \cdot \#U}$ viele Lösungskandidaten gibt, die mit Hilfe eines nichtdeterministischen Algorithmus parallel erstellt werden könnten. Jeder Lösungskandidat ist auf seine Zulässigkeit zu prüfen, das heißt er muss alle vier Bedingungen des *SSP* erfüllen, was in polynomieller Zeit untersucht werden kann.

Somit existiert ein nichtdeterministischer Algorithmus, der *SSP* in Polynomialzeit entscheidet, und folglich gilt *SSP* ist in der Komplexitätsklasse *NP*.

Korollar 1. SSP *ist NP-vollständig.*

Aus dem Resultat, dass *SSP* in *NP* ist und *SSP NP-hart* ist, folgt die *NP-Vollständigkeit* für *SSP*.

4

Parametrisierte Komplexität

4.1 Die parametrisierte Komplexitätstheorie

In der klassischen Komplexitätstheorie wird die Laufzeit der Entscheidungs- und Optimierungsprobleme bezüglich ihrer Eingabelänge bestimmt. Hierzu existieren unter anderem die Komplexitätsklassen P, NP und EXP. In P befinden sich die Probleme, die in polynomieller Zeit gelöst werden können. In der Komplexitätsklasse EXP sind die Probleme enthalten, deren zugehörige Algorithmen eine exponentielle Laufzeit haben. Eine viel untersuchte Komplexitätsklasse ist die Klasse NP, denn hierin befinden sich Probleme, die exponentiell viele Lösungskandidaten enthalten, aber die Prüfung, ob ein solcher Kandidat tatsächlich eine Lösung für dieses Problem ist, kann in polyomieller Zeit erfolgen. Die Probleme der Klasse NP auf die alle anderen Probleme dieser Klasse *Polynomialzeit-Many-one-reduzierbar* [Sch11] sind, werden als *NP-vollständig* bezeichnet und sind die schwierigsten Probleme dieser Klasse. Diese Komplexitätsklasse enthält Probleme, die in polynomieller Zeit von einem nichtdeterministischen Algorithmus gelöst werden können. Der Nichtdeterminismus ist jedoch nur ein theoretisches Modell und somit ist zu erwarten, dass solange die Frage, ob $P = NP$ nicht beantwortet ist, kein effizienter Algorithmus für *NP-vollständige* Probleme gefunden werden kann. Daher haben derzeit die Algorithmen für *NP-vollständige* Probleme eine exponentielle Laufzeit und sind folglich für große Instanzen praktisch nicht lösbar. Da die Klasse NP praktisch relevante Probleme enthält, werden beispielsweise Approximationsalgorithmen entworfen, das heißt, Algorithmen, deren maximale Abweichung der Lösung vom Optimum begrenzt ist, jedoch für große Instanzen in Polynomialzeit lösbar sind [Sch08].
Eine Alternative zur klassischen Komplexitätstheorie bietet die parametrisierte Komplexitätstheorie, die eine zweidimensionale Komplexitätsanalyse verwendet. Die Laufzeit ist zum Einen, wie in der klassischen Komplexitätstheorie, abhängig von der Größe der Instanz und zum Anderen von einer Parametrisierung κ. Analog dazu sind parametrisierte Probleme ebenfalls zweidimensional. Diese sind wie folgt definiert [FG06]:

Definition 3. *Parametrisierung, Parametrisiertes Problem*

Sei Σ ein endliches Alphabet.

1. *Eine Parametrisierung von Σ^* ist eine Abbildung $\kappa\colon \Sigma^* \to \mathbb{N}$, die in polynomieller Zeit berechenbar ist.*
2. *Ein parametrisiertes Problem über Σ ist ein Paar (Q, κ) mit $Q \subseteq \Sigma^*$ und einer Parametrisierung κ von Σ^*.*

Ein parametrisiertes Problem ist somit eine Problem-Parameter-Kombination, das heißt es besteht zum Einen aus einem Problem und zum Anderen aus einer Parametrisierung bzw. einem Parameter. Falls die Parametrisierung auf eine Konstante abbildet ($\kappa(x) = i$ mit $i \in \mathbb{N}$), dann kann das zugehörige parametrisierte Problem verkürzt mit (Q, i) dargestellt werden. In der Definition 3 ist nur eine Funktion für die Parametrisierung vorhanden, dies bedeutet aber nicht, dass nicht mehrere Daten der Eingabe in die Parametrisierung mit aufgenommen werden können. Falls ein Problem mit einer Menge von Parametern P zu untersuchen ist, so wäre die entsprechende Parametrisierung $\kappa(x) = \sum_{p \in P} p$.
Die Grundidee der parametrisierten Komplexitätstheorie ist, dass man den exponentiellen Teil der Laufzeit auf den Parameter $\kappa(x)$ beschränken möchte, sodass die Laufzeit in der Eingabelänge polynomiell und bezüglich $\kappa(x)$ exponentiell ist. Somit kann das Problem effizient gelöst werden, falls der Parameter klein ist, ohne auf die Optimalität verzichten zu müssen. Am besten sind somit Problem-Parameter-Kombinationen, deren Parameter in der Praxis gewöhnlich einen kleinen Wert hat. Die Wahl der Parametrisierung ist entscheidend, denn die Einordnung einer Problem-Parameter-Kombination in eine parametrisierte Komplexitätsklasse ist immer von der zugehörigen Parametrisierung abhängig.
Die Problem-Parameter-Kombinationen für die gilt, dass der exponentielle Teil der Laufzeit ausschließlich auf den Parameter beschränkt werden kann, werden als *fixed-parameter tractable* bezeichnet. Die Eigenschaft *fixed-parameter tractable* ist wie folgt definiert [FG06]:

Definition 4. *fixed-parameter tractability, FPT*

Sei Σ ein endliches Alphabet und $\kappa\colon \Sigma^* \to \mathbb{N}$ eine Parametrisierung.

1. *Ein Algorithmus $\mathcal{A}$ mit einem Eingabealphabet Σ ist ein FPT-Algorithmus bezüglich κ, falls eine berechenbare Funktion $f\colon \mathbb{N} \to \mathbb{N}$ und ein Polynom $p \in \mathbb{N}_0[X]$ existiert, so dass für jedes $x \in \Sigma^*$ die Laufzeit von $\mathcal{A}$ mit der Eingabe x höchstens $f(\kappa(x)) \cdot p(|x|)$ ist.*
2. *Ein parametrisiertes Problem (Q, κ) ist fixed-parameter tractable, falls ein FPT-Algorithmus bezüglich κ existiert, der Q entscheidet.*

   Die Komplexitätsklasse FPT enthält genau die Probleme, die fixed-parameter tractable sind.

Wie bereits erwähnt wurde, darf die Parametrisierung einer Problem-Parameter-Kombination von mehreren Parametern abhängen. Je mehr Parameter dieses Problem enthält oder je größer deren Werte sind, desto einfacher ist in der Regel der Entwurf eines zugehörigen *FPT-Algorithmus*. Zu einer Problem-Parameter-Kombination, deren Parameter der Eingabelänge entspricht, kann immer ein *FPT-Algorithmus* entworfen werden. Ein solcher Algorithmus ist nur hilfreich, wenn seine zugehörigen Parameter in der Praxis gewöhnlich einen relativ niedrigen Wert enthalten. Wird als Parameter die Eingabelänge gewählt, dann unterscheidet sich die parametrisierte Komplexitätsanalyse nicht von der klassischen Komplexitätsanalyse. In der parametrisierten Komplexitätsanalyse ist das Ziel, die spezielle Struktur des zu untersuchenden Problems dahingehend auszunutzen, dass wenige möglichst kleine Parameter gewählt werden, welche die exponentielle Laufzeit beschränken. Es ist daher bei jedem Resultat der parametrisierten Komplexität auf die Qualität der Parameter zu achten, bevor man beurteilen kann, wie gut das Resultat wirklich ist.
Für die parametrisierten Probleme (Q, κ) mit $Q \in P$ gilt, dass $(Q, \kappa) \in FPT$ für alle Parameter κ, da die Laufzeit von Q ausschließlich aus einem polynomiellen Teil besteht. In der parametrisierten Komplexitätstheorie existieren, neben der Komplexitätsklasse *FPT*, die Komplexitätsklassen *para-NP*, *XP* und noch viele weitere [FG06]. Wenn man die Klasse *FPT* als vergleichbare Klasse zu *P* in der parametrisierten Komplexitätstheorie sieht, dann würde entsprechend die Klasse *para-NP* zu der Klasse *NP* korrespondieren. Für alle Probleme Q gilt, dass Q genau dann in *NP* ist, wenn das parametrisierte Problem $(Q, 1)$ in *para-NP* ist. Die Definition dieser Klasse sieht wie folgt aus [FG06]:

Definition 5. *para-NP*
Ein parametrisiertes Problem (Q, k) über einem Alphabet Σ ist in para-NP, falls eine berechenbare Funktion $f\colon \mathbb{N} \to \mathbb{N}$, ein Polynom $p \in \mathbb{N}_0[X]$ und ein nichtdeterministischer Algorithmus existiert, der für $x \in \Sigma^*$ entscheidet, ob $x \in Q$ in höchstens $f(\kappa(x)) \cdot p(|x|)$ Schritten.

Der Unterschied der Definition 4 (*FPT*) und 5 (*para-NP*) liegt darin, dass in *para-NP* der Algorithmus nichtdeterministisch sein darf. Infolge dessen gilt *FPT* ist eine Teilmenge von *para-NP*. Die Frage, ob *FPT* eine echte Teilmenge von *para-NP* ist oder ob diese beiden Klassen gleich sind, ist noch offen. Diese Frage steht im Zusammenhang mit der Frage, ob $P = NP$, denn es gilt $P = NP$ genau dann, wenn $FPT = \text{para-}NP$ [FG06].

Die nachfolgende Abbildung zeigt noch weitere Klassen der parametrisierten Komplexitätstheorie, wobei in dieser Arbeit einzig die Klassen *FPT* und *para-NP* betrachtet werden.

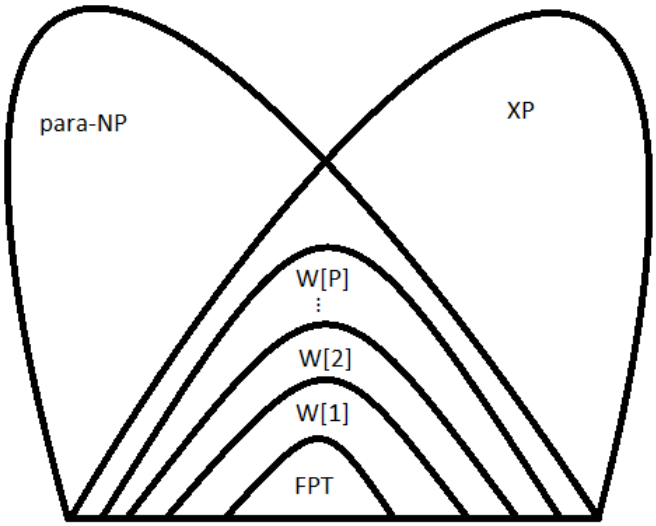

Abbildung 4.1. Überblick einiger Klassen der parametrisierten Komplexitätstheorie [FG06]

In dieser Arbeit wird *SSP* bezüglich seiner parametrisierten Komplexität untersucht. Da diese Analyse eine zweidimensionale Betrachtung vorsieht, werden verschiedene parametrisierte Probleme von *SSP* analysiert. Die Hoffnung besteht darin einen guten Parameter κ zu finden, so dass gilt $(SSP, \kappa) \in FPT$.
Zum Nachweis der *fixed-parameter tractability* eines parametrisierten Problems gibt es unterschiedliche Vorgehensweisen. Beispielsweise kann ein konkreter *FPT-Algorithmus* entworfen werden. Hierfür eignen sich unterschiedliche Entwurfsmuster, wie zum Beispiel *Tiefenbeschränkte Suchbäume*, deren Tiefe mit Hilfe des Parameters beschränkt wird. Ein solcher Suchbaum kann beispielsweise für das (*3-Hitting Set Problem*, k)[Nie02] angegeben werden, welches als Parameter k die Größe des *Hitting Sets* enthält. Zu jeder Hyperkante gibt es maximal 3 Knoten von denen mindestens eine im *Hitting Set* enthalten sein muss. Somit kann mit Hilfe eines *Tiefenbeschränkten Suchbaums* der Tiefe k ein *FPT-Algorithmus* entworfen werden, dessen Laufzeit in $O(3^k \cdot n)$ liegt mit der Eingabelänge n.
Ein weiteres Entwurfsmuster für *FPT-Algorithmen*, welches sich insbesondere für Minimierungsprobleme eignet, ist beispielsweise *Iterative Compression*. Hierbei wird basierend auf einer Lösung der Größe $k + 1$ eine Lösung der Größe k erzeugt und somit wird iterativ zur gewünschten Größe der Lösung hingearbeitet.
Es gibt einige weitere Entwurfsmuster für *FPT-Algorithmen*, die sich für unterschiedliche Problemstellungen eignen [Nie02]. Neben der Variante des Entwurfs eines konkreten *FPT-Algorithmus*, kann die Eigenschaft der *fixed-parameter tractablility* auch mit Hilfe einer Datenreduktion auf den Problemkern, bekannt als *Kernelization* (Definition 6), nachgewiesen werden. Hierbei wird das Problem so

auf sich selbst reduziert, dass die Länge der Instanz ausschließlich von den ursprünglichen Parametern abhängt. Die Laufzeit jedes Algorithmus lässt sich auf die Eingabelänge beschränken und somit in diesem Fall auf die Parameter. Da die Modifizierung der Instanz in polynomieller Zeit berechenbar sein muss, folgt hieraus, dass diese Problem-Parameter-Kombination in FPT ist, falls eine derartige Reduktion existiert. Somit bietet die *Kernelization* eine Alternative zum Nachweis der *fixed-parameter tractability*. In Abschnitt 4.3.1 wird eine *Kernelization* für ein parametrisiertes *SSP* erstellt. Darauffolgend werden verschiedene Parametrisierungen zu *SSP* angegeben und bezüglich ihrer parametrisierten Komplexität untersucht.

4.2 Das parametrisierte *SSP*

Die Analyse eines parametrisierten Problems ist eine zweidimensionale Untersuchung, die zum Einen das Problem und zum Anderen die zugehörige Parametrisierung betrachtet. Ein Problem kann mehrere potenzielle Parameter haben und somit ist es interessant verschiedene Problem-Parameter-Kombinationen zu einem Problem zu untersuchen. Das *SSP* enthält mehrere Eingabedaten und somit gibt es auch viele mögliche Parameter. Wie im vorherigen Kapitel erklärt wurde, ist eine Parametrisierung besser je weniger Parameter sie enthält und desto kleiner deren Werte sind. Ein *FPT-Algorithmus* zu einem parametrisierten Problem ist daher für die Praxis hilfreicher, je kleiner die zugehörige Parametrisierung ist. Im Folgenden werden nicht nur parametrisierte *SSP* mit einer kleinen Parametrisierung untersucht, da es ebenfalls interessant ist, wie sich die großen Eingabedaten auf die Laufzeit eines zugehörigen Algorithmus auswirken. Hierzu wird beispielsweise die Anzahl der Studierenden (s) als Parameter betrachtet. Dieser Parameter hat gewöhnlich den größten Wert der Eingabe.
Die Größe der gesuchten Lösung ist abhängig von k und da k die Anzahl der Studierenden/Modul-Kombinationen ist, welche unverteilt bleiben dürfen, ist dieser Wert normalerweise sehr klein. k wird ebenfalls als Parameter untersucht. Die Kapazität der Übungsgruppen gehört in der Regel ebenfalls nicht zu den größten Werten der Eingabe und wäre somit auch ein guter Parameter. Die Anzahl der insgesamten Übungsgruppen gehört im Allgemeinen zu den größeren Daten in der Eingabe, wird aber trotzdem in dieser Untersuchung als Parameter hinzugenommen. Ein besserer Parameter ist die maximale Anzahl der Übungsgruppen pro Modul (u), daher wird dieser Wert ebenfalls als Parameter betrachtet. Es könnte sein, dass die Schwierigkeit des *SSP* mit der maximalen Anzahl der Termine (t), die eine Übungsgruppe hat, zusammenhängt, daher wird dieser Wert ebenfalls als Parameter analysiert. Im Folgenden werden die verschiedenen parametrisierten *SSP* bezüglich ihrer parametrisierten Komplexität und der Qualität ihrer Parametrisierung untersucht.

4.3 Parameter p, c, k

Eine der hier untersuchten parametrisierten Probleme ist das *SSP* mit den Parametern p (die insgesamte Anzahl von Übungsgruppen), c (die maximale Kapazität einer Übungsgruppe) und k (die maximale Anzahl der Studierenden/Modul-Kombinationen, die unverteilt bleiben dürfen). Somit ist (Q, κ) mit $\kappa(x) = p+c+k$ das zu untersuchende parametrisierte Problem.

Lemma 4.1. *Das parametrisierte SSP mit den Parameter k, p unc c befindet sich in FPT.*

Die Korrektheit des Lemma 4.1 wird im nachfolgenden Abschnitt 4.3.1 anhand einer *Kernelization* gezeigt, sowie in Abschnitt 4.3.2 mittels eines *FPT-Algorithmus* bewiesen.

4.3.1 Kernelization

Wie einleitend bereits erläutert, kann zum Nachweis der *fixed-parameter tractability*, neben der Erstellung eines *FPT-Algorithmus*, eine *Kernelization* erzeugt werden. Falls zu einer Problem-Parameter-Kombination eine Reduktion des Problems auf sich selbst erstellt werden kann, so dass die Instanz durch die Reduktionsfunktion derart geändert wird, dass deren Größe nur von deren ursprünglichen Parameter abhängig ist und die Reduktionsfunktion in Polynomialzeit berechnet werden kann, dann existiert eine *Kernelization* zu der Problem-Parameter-Kombination. Die *Kernelization* ist wie folgt definiert [FG06]:

Definition 6. *Kernelization*

Eine Kernelization eines parametrisierten Problems (Q, κ) über dem Alphabet Σ ist eine in Polynomialzeit berechenbare Funktion $K\colon \Sigma^* \to \Sigma^*$, so dass für alle $x \in \Sigma^*$ gilt:

$$x \in Q \Leftrightarrow K(x) \in Q$$

und

$$|K(x)| \leq g(\kappa(x))$$

für eine berechenbare Funktion g.

Für jede Instanz x ist $K(x)$ der zugehörige *Problemkern*. Auf Grund der Definition 6 gilt, dass sich die Größe des Problemkerns $K(x)$ ausschließlich von der Parametrisierung $\kappa(x)$ begrenzen lässt und daher ist die Laufzeit jedes Algorithmus, der das Problem $K(x) \in Q$ entscheidet, nur von der Parametrisierung $\kappa(x)$ abhängig. Die Vorberechnung zum Erstellen von $K(x)$ findet, laut Definition 6, in Polynomialzeit statt. Es kann somit ein Algorithmus $\mathcal{A}$ für Q erstellt werden, der jede Eingabe $x \in \Sigma^*$ mit Hilfe der Funktion K in Polynomialzeit reduziert und anschließend mit

einem Algorithmus, dessen Laufzeit nur von $\kappa(x)$ abhängt, entscheidet. Hieraus ergibt sich für die Laufzeit von $\mathcal{A}$, dass sich ausschließlich der polynomielle Teil auf die Eingabelänge bezieht und falls ein exponentieller Teil existiert, dieser nur auf den Parameter bezogen ist. Somit gilt für $\mathcal{A}$, dass dieser Algorithmus ein *FPT-Algorithmus* ist. Dies bedeutet, dass sich eine Problem-Parameter-Kombination in *FPT* befindet, falls eine zugehörige *Kernelization* existiert. Somit bietet diese Vorgehensweise eine Alternative zum Nachweis der *fixed-parameter tractability*.

Kernelization für *SSP* mit den Parameter p, c und k:

In diesem Abschnitt wird eine *Kernelization* für das *SSP* mit den Parametern p, c und k angegeben:

$$\text{Sei } x = (M, S, U, \tau, \beta, \delta, p, c, k) \text{ eine Instanz für } SSP.$$

Für die *Kernelization* wird eine Reduktionfunktion K benötigt, die in polynomieller Zeit die Eingabe x auf ihren Problemkern reduziert, so dass die Eingabelänge ausschließlich von den ursprünglichen Parametern k, p und c abhängig ist.
Falls bei der Studierendenverteilung die Kapazität aller Übungsgruppen voll ausgeschöpft wird, dann werden maximal $p{\cdot}c$ viele Studierende/Modul-Kombinationen verteilt. Dies ist die maximale Anzahl der Studierenden/Modul-Kombinationen, die theoretisch zugewiesen werden können. Laut Definition von *SSP* dürfen maximal k Studierende/Modul-Kombinationen unverteilt bleiben, das bedeutet, dass wenn es mehr als $p \cdot c + k$ viele Studierende/Modul-Kombinationen gibt, dann bleiben mehr als k Studierende/Modul-Kombinationen unverteilt und dann gilt $x \notin SSP$. Laut Definition der Funktion β muss jeder Studierende mindestens ein Modul wählen, das heißt es existieren maximal so viele Studierende wie es Studierende/Modul-Kombinationen gibt. Folglich gilt für $\#S > p \cdot c + k$, dass $x \notin SSP$. Ansonsten gilt $\#S \leq p \cdot c + k$ und somit wird die Anzahl der Studierenden mit den Parametern k, p und c beschränkt. Des Weiteren können keine Studierenden/Modul-Kombinationen, deren Modul keine Übungsgruppen hat, verteilt werden. Daher werden in der Vorberechnung solche Module aus M und β gelöscht und entsprechend der Parameter k angepasst. Mit Hilfe dieser Modifizierungen von M lässt sich die Größe dieser Menge auf den Parameter p beschränken. Die Funktion K ist wie folgt definiert:

$$K(x) = \begin{cases} x^*, & \text{falls } \#S > p \cdot c + k \\ x', & \text{sonst} \end{cases}$$

mit $x^* = (\{m_1\}, \{s_1, ..., s_{k+1}\}, \emptyset, \tau, \beta \colon S \to \{m_1\}, \delta, 0, c, k)$

und $x' = (M', S, U, \tau, \beta', \delta, p, c, k - (\#A - \#A'))$ mit

$M' = \{m_i \mid m_i \in M \wedge \exists u_{ij} \in U\}$,

$\forall s \in S : \beta'(s) = \{m_i \mid m_i \in \beta(s) \wedge m_i \in M'\}$,

$A' = \{(s_i, m_j) \mid (s_i, m_j) \in A \wedge m_j \in M'\}$.

Für die Instanz x^* gilt, unabhängig von der konkreten Eingabe x, dass

$sol_{SSP}(x^*) = false$, da $k+1$ viele Studierende/Modul-Kombinationen zu verteilen sind, aber keine Übungsgruppen existieren. Hieraus ergibt sich, dass alle $k+1$ Studierende/Modul-Kombinationen unverteilt bleiben.

Der Beweis dieser Behauptung sieht wie folgt aus:

Es ist zu zeigen, dass gilt $x^* \notin SSP$.

Direkter Beweis:

x^* ist wie folgt definiert: $x^* = (\{m_1\}, \{s_1, ..., s_{k+1}\}, \emptyset, \tau, \beta \colon S \to \{m_1\}, \delta, k)$

$$\Rightarrow \forall R \subseteq A \times U : \quad R = \emptyset, \text{ da U } = \emptyset.$$

$$\Rightarrow \forall R \subseteq A \times U : \#R = 0$$

Auf Grund der Belegung von β gilt, dass $A = \bigcup_{s_i \in S} \{(s_i, m_1)\}$

$$\Rightarrow \#A = \#S$$

$$\Rightarrow \#A = k + 1$$

Folglich gilt für alle $R \subseteq A \times U : \#R = \#A - (k+1)$ und somit gilt $\#R < \#A - k$.

Es existiert daher keine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$ und folglich

gilt $x^* \notin SSP$.

Laut der Definition 6 muss eine *Kernelization* folgende Äquivalenzbeziehung erfüllen:

$$x \in Q \Leftrightarrow K(x) \in Q$$

Somit ist zu zeigen, dass gilt $x \in SSP \Leftrightarrow K(x) \in SSP$.

1. Direkter Beweis für $x \in SSP \Rightarrow K(x) \in SSP$:

   Sei $x \in SSP$.

   $$\Rightarrow \exists R \subseteq A \times U : \#R \geq \#A - k \wedge R \text{ ist für } SSP \text{ zulässig.}$$

   Sei $R' \subseteq A \times U$ mit $\#R' \geq \#A - k \wedge R'$ ist für SSP zulässig.

   $$\Rightarrow \forall (s_i, m_j) \in A : m_j \notin M' \Rightarrow \forall u_{lq} \in U : ((s_i, m_j), u_{lq}) \notin R'$$

   $$\Rightarrow \forall ((s_i, m_j), u_{jl}) \in R' : m_j \in M'$$

   $$\Rightarrow \forall ((s_i, m_j), u_{jl}) \in R' : (s_i, m_j) \in A'$$

   $$\Rightarrow \exists R \subseteq A' \times U : \#R \geq \#A - k \wedge R \text{ ist für } SSP \text{ zulässig.}$$

   Aus $\#R \geq \#A - k$ folgt:

   $$\Rightarrow \#R \geq \#A' - k + \#A - \#A'$$

   $$\Rightarrow \#R \geq \#A' - (k - (\#A - \#A'))$$

Somit gilt:

$\exists R \subseteq A' \times U : \#R \geq \#A' - (k - (\#A - \#A')) \wedge R$ ist zulässig für SSP

$\Rightarrow x' \in SSP$

$\Rightarrow K(x) \in SSP$ (da $K(x) = x'$ sonst wäre $\#S > p \cdot c + k$ und folglich $x \notin SSP$)

2. Indirekter Beweis für $K(x) \in SSP \Rightarrow x \in SSP$.

Sei $x \notin SSP$.

Widerspruchsbeweis:

Sei $K(x) \in SSP$.

$\Rightarrow K(x) = x' \wedge K(x) \in SSP$ (da für $K(x) = x^*$ gilt $K(x) \notin SSP$)

$\Rightarrow x' \in SSP$

$\Rightarrow \exists R \subseteq A' \times U : \#R \geq \#A' - (k - (\#A - \#A')) \wedge R$ ist zulässig für SSP

$\Rightarrow \exists R \subseteq A \times U : \#R \geq \#A' - (k - (\#A - \#A')) \wedge R$ ist zulässig für SSP

(da $A' \subseteq A$)

$\Rightarrow \exists R \subseteq A \times U : \#R \geq \#A - k \wedge R$ ist zulässig für SSP

$\Rightarrow x \in SSP$

Dies führt zu einem Widerspruch der Beweisvoraussetzung, die besagt, dass gilt $x \notin SSP$ und somit gilt $K(x) \notin SSP$.

Somit gilt $x \in SSP \Leftrightarrow K(x) \in SSP$.

Des Weiteren ist zu zeigen, dass die Länge von $K(x)$ ausschließlich von den Parametern k, p und c abhängt.

1. Fall: Sei $K(x) = x^*$.

$$\begin{aligned}
\text{Somit gilt } |K(x)| &= |\{m_1\}| + |\{s_1, ..., s_{k+1}\}| + |\emptyset| + |\tau| + |\beta| + |\delta| + |p| + |c| + |k| \\
&\in O(1) + O(k+1) + O(1) + O(1) + O(2 \cdot \#S) + O(1) \\
&\quad + O(1 + |c| + |k|) \\
&\in O(k+1) + O(p \cdot c + k + |c| + |k|) \\
&\in O((p+1) \cdot c + k)
\end{aligned}$$

2. Fall: Sei $K(x) = x'$.

$$
\begin{aligned}
\text{Dann gilt } |K(x)| &= |M'| + |S| + |U| + |\tau| + |\beta'| + |\delta| + |p| + |c| \\
&\quad + |k - (\#\beta - \#\beta')| \\
&\in O(p + (p \cdot c + k) + p + 2 \cdot p + \#S \cdot \#M' + 3 \cdot p^2 + |p| + |c| \\
&\quad + |k|) \\
&\in O(p + (p \cdot c + k) + (p \cdot c + k) \cdot p + \cdot p^2 + |p| + |c| + |k|) \\
&\in O((p \cdot c + k) \cdot (p + 1) + |p| + |c| + |k|) \\
&\in O(((p + 1) \cdot c + k) \cdot (p + 1)) \\
&\in O((p + 1)^2 \cdot (c + k))
\end{aligned}
$$

In beiden Fällen ist $|K(x)|$ in $O((p + 1)^2 \cdot (c + k))$. Somit kann mit Hilfe der Parametrisierung $\kappa(x) = p + c + k$ die Größe von $|K(x)|$ auf $O(\kappa(x)^3)$ beschränkt werden.
Es wurde gezeigt, dass K alle erforderlichen Eigenschaften für eine *Kernelization* erfüllt. Da sich die exponentielle Laufzeit eines *FPT-Algorithmus* zu einer Problem-Parameter-Kombination mit Hilfe der Parameter beschränken lässt gilt, dass das Problem für kleine Werte der Parameter effizient lösbar ist. Daher sollte eine gute Kernelization die Parameter eines Problems bei der Reduktion nicht erhöhen, dass heißt, es sollte für K gelten $\kappa(K(x)) \leq \kappa(x)$. Die Funktion K, die in dieser Reduktion verwendet wurde, wird auf diese Eigenschaft untersucht.
Für x^* gilt $\kappa(x^*) = k + c + 0$. Die beiden Parameter k und c werden für x^* durch die Funktion K nicht verändert, jedoch wird der Parameter p auf 0 reduziert. Da $\kappa(x) = k + c + p$ gilt $\kappa(x^*) \leq \kappa(x)$.
Für x' gilt $\kappa(x') = k - (\#A - \#A') + c + p$. Die Funktion K verändert die beiden Parameter c und p nicht. Da A' eine Teilmenge von A ist, gilt $\#A - \#A' \geq 0$. Hieraus ergibt sich, dass $k - (\#A - \#A') \leq k$ ist und somit wird der Parameter k für x' nicht erhöht. Es gilt somit $\kappa(x') \leq \kappa(x)$.
Aus den beiden Resultaten $\kappa(x^*) \leq \kappa(x)$ und $\kappa(x') \leq \kappa(x)$ folgt $\kappa(K(x)) \leq \kappa(x)$. Somit gilt, dass K die Eigenschaft $\kappa(K(x)) \leq \kappa(x)$ erfüllt.
Ein Algorithmus, der mittels erschöpfender Suche alle potenziellen Lösungskandidaten durchläuft und diese bezüglich ihrer Zulässigkeit prüft, hat eine exponentielle Laufzeit bezüglich der Eingabelänge. Dieser Algorithmus ist ein *FPT-Algorithmus* für *SSP*, da nach polynomieller Vorberechnung, die Eingabelänge und somit auch die Laufzeit ausschließlich von den Parametern abhängt.
Daher gilt, dass *SSP* mit den Parametern k, p und c in *FPT* enthalten ist.
Nicht jede *Kernelization* ist gleich gut, denn je mehr das Problem auf seinen Kern reduziert wird, desto besser ist sie. Eine gute *Kernelization* ist eine *polynomielle Kernelization*, die wie folgt definiert ist [FG06]:

Definition 7. *Eine Kernelization K eines parametrisierten Problems (Q, κ) über das Alphabet Σ ist polynomiell, falls ein Polynom $p(X) \in \mathbb{N}_0[X]$ existiert, so dass $|K(x)| \leq p(\kappa(x))$ für alle $x \in \Sigma^*$.*

Es wurde gezeigt, dass gilt $|K(x)| \in O(\kappa(x)^3)$. Da $\kappa(x)^3$ mit $\kappa(x)$ ein Polynom ist, gilt dass sich die Größe von $K(x)$ mittels einer polynomiellen Funktion von $\kappa(x)$ beschränken lässt.

4.3.2 FPT-Algorithmus

Im vorherigen Abschnitt wurde eine *polynomielle Kernelization* für *SSP* mit den Parametern k, p und c angegeben und folglich gilt, dass sich diese Problem-Parameter-Kombination in der Komplexitätsklasse *FPT* befindet.
Die zugehörige *Kernelization* zeigte, dass ein entsprechender *FPT-Algorithmus* existiert, der beispielsweise so aussehen könnte, dass der erste Teil dieses Algorithmus aus der polynomiellen Vorberechnung besteht und anschließend die erschöpfende Suche angewendet wird. Dies bedeutet, falls $\#S > p \cdot c + k$, dann liefert der Algorithmus *false* zurück, ansonsten startet die erschöpfende Suche mit der modifizierten Instanz x'.
Zu jeder Übungsgruppe u_{ij} existiert eine Menge $B_{ij} \subseteq S$, die alle Studierende enthält, die das Modul m_i wählten und bisher noch keiner Übungsgruppe von m_i zugeteilt wurden, sowie an den Terminen der Übungsgruppe u_{ij} noch nicht anderweitig verplant sind. Somit gilt, dass die Studierenden, die der Übungsgruppe u_{ij} zugewiesen werden, eine Teilmenge von B_{ij} bilden, deren Kardinalität maximal $\tau(u_{ij})$ beträgt. Der Algorithmus probiert alle solche Teilmengen von B_{ij} aus, das heißt maximal $2^{\#S}$ viele verschiedene Teilmengen existieren, die der Übungsgruppe u_{ij} zugewiesen werden können. Die nachfolgende Abbildung stellt alle Pfade der erschöpfenden Suche dar, somit ist ein Pfad vom Wurzelknoten bis zu einem Blatt eine mögliche Relation für *SSP*. In der Abbildung 4.2. existieren x viele verschiedene Teilmengen von $B_{11}(B_{111}, ..., B_{11x})$, zur Übungsgruppe u_{12} gibt es y viele Teilmengen von B_{12} und B_{vj} hat z viele verschiedene Teilmengen.

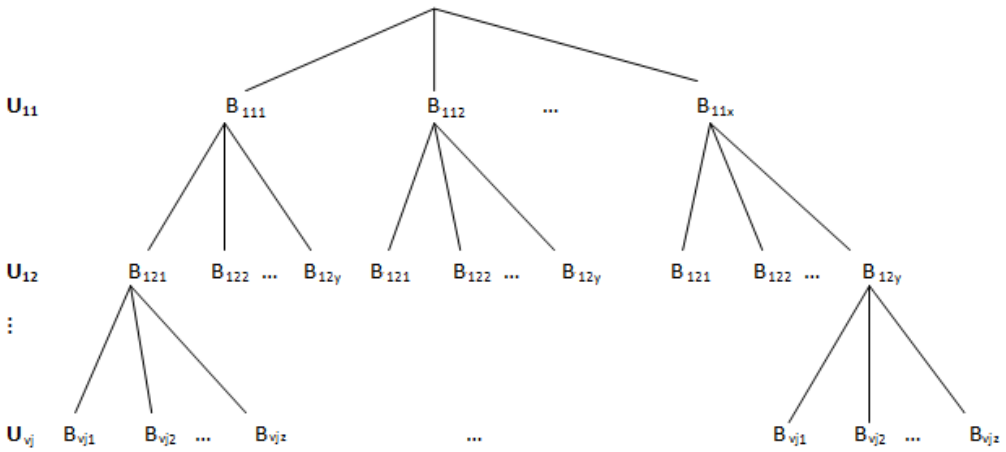

Abbildung 4.2. Suchbaum für erschöpfende Suche des Problemkerns

Der resultierende Suchbaum hat eine Tiefe von p und jeder Knoten hat ma-

ximal $2^{\#S}$ viele Nachfolger. Dies bedeutet, dass in $O((2^{\#S})^p) = O(2^{\#S \cdot p})$ eine erschöpfende Suche möglich ist. Die zu untersuchende Problem-Parameter-Kombination enthält als Parameter k, p und c und somit darf der exponentielle Teil der Laufzeit nur von diesen Parametern abhängen.
Der Algorithmus prüft zuerst in Polynomialzeit, ob $\#S > p \cdot c + k$ und falls dem so ist, gibt der Algorithmus *false* zurück. Andernfalls wird die erschöpfende Suche mit dem, in Polynomialzeit erstellten, Problemkern gestartet. Das Durchlaufen des Lösungsraumes kann in $O(2^{\#S \cdot p})$ und somit auch in $O(2^{(p \cdot c + k) \cdot p})$ erfolgen. Da jede potenzielle Lösung in polynomieller Zeit auf ihre Zulässigkeit geprüft werden kann, gilt, dass ein Polynom b existiert, so dass die Laufzeit des Algorithmus für *SSP* in $O(2^{p \cdot (p \cdot c + k)} \cdot b(n))$ liegt und somit dieser Algorithmus ein *FPT-Algorithmus* ist. Nachfolgend wird der soeben beschriebene Algorithmus als rekursiver Algorithmus angegeben:

Algorithmus 4.7. SSP mit den Parametern k, p, c

Eingabe: $x = (M, S, U, \tau, \beta, \delta, k, R, i)$ $//R$: Liste mit aktuellem Pfad

Ausgabe: y

$//$Sei $u_{jl} \in U$ mit $\alpha^1(u_{jl}) = i$
(1) **if** $i = 1 \wedge \#S > (\sum_{u_{jl} \in U} \tau(u_{jl})) + k$ **then return** *false*;
(2) **endif**;
(3) $Subsets := getStudentSubsets^2(u_{jl}, R, \tau(u_{jl}))$;
(4) **if** $i = \#U$ **then**
(5) **foreach** $subset \in Subsets$ **do**
(6) **if** $getAmountOfUnassignedStudents^3(R, subset) \leq k$ **then**
(7) **return** *true*;
(8) **endif**;
(9) **endfor**;
(10) **else**
(11) **foreach** $subset \in Subsets$ **do**
(12) **if** $SSP(M, S, U, \tau, \beta, \delta, k, R.set(i, subset), i + 1) = true$ **then**
(13) **return** *true*;
(14) **endif**;
(15) **endfor**;
(16) **endif**;
(17) **return** *false*;

---

[1] Die Funktion $\alpha\colon U \to \{1, ..., \#U\}$ weist jeder Übungsgruppe $u_{ij} \in U$ eine eindeutige Zahl zu, so dass gilt $\alpha(u_{11}) = 1, \alpha(u_{12}) = 2, ...\alpha(u_{vj}) = p$, wobei $v = \#M$ und j die Anzahl der Übungsgruppen des Moduls m_v ist.

[2] Die Funktion *getStudentSubsets*$(u_{jl}, R, \tau(u_{jl}))$ bestimmt die Menge aller Studierenden, die m_j wählten, in R noch nicht bezüglich m_j verteilt wurden und die noch keine andere Übungsgruppe in R zugeteilt bekamen, die sich zeitlich mit u_{jl} überschneidet. Aus dieser Studierendenmenge B_{jl} werden alle möglichen Teilmengen erzeugt, die nicht mehr als $\tau(u_{jl})$ viele Elemente enthalten und anschließend zurückgeliefert.

[3] Die Methode *getAmountOfUnassignedStudents(R, subset)* bestimmt die Anzahl der Studierenden/Modul-Kombinationen, die in R und u_{vj} nicht verteilt werden konnten. Die Menge der Kombinationen (A) kann in $O(\#S \cdot \#M)$ erstellt werden. Anschließend werden alle diese Elemente durchlaufen und geprüft, ob sie in R enthalten sind. Somit gibt es maximal $\#S \cdot \#M$ viele Schleifendurchläufe, in denen diese Prüfung stattfindet. Da R maximal $p-1$ viele Mengen von Studierenden beinhaltet, befinden sich in R nicht mehr als $p \cdot \#S$ bzw. $p \cdot (p \cdot c + k)$ viele Elemente. Die Prüfung, ob eine Studierende/Modul-Kombination in R enthalten ist, kann somit in $O(p \cdot (p \cdot c + k))$ erfolgen. Demnach gilt, dass sich die Laufzeit dieser Methode in der Ordnungsklasse $O(\#S \cdot \#M \cdot p \cdot (p \cdot c + k))$ und somit in $O((p \cdot c + k)^2 \cdot p^2)$ befindet.

Laufzeitanalyse:

Die Funktion *getStudentSubsets* aus Zeile (3) liefert maximal $2^{\#S}$ viele Teilmengen und somit ist die Laufzeit dieser Funktion in $O(2^{\#S})$. Da laut nicht-erfüllter *if-Bedingung* gilt, dass $\#S \leq p \cdot c + k$, folgt daraus, dass *getStudentSubsets* in $O(2^{p \cdot c + k})$ liegt.
Im *if-Rumpf* ab Zeile (5), sowie im *else-Rumpf* in Zeile (11), durchläuft die *foreach-Schleife* alle Teilnehmer aus *Subsets* und daher gibt es maximal $2^{p \cdot c + k}$ viele Durchläufe.
Die Funktion *getAmountOfUnassignedStudents(R, subset)* aus Zeile (6) bestimmt die Anzahl nicht-verteilter Studierender/Modul-Kombinationen in $O((p \cdot c + k)^2 \cdot p^2)$.
Für jeden Rumpf der *foreach-Schleife* aus Zeile (11) gilt, dass der rekursive Algorithmus für *SSP* erneut aufgerufen wird, wobei i um eins erhöht wird. Für die Laufzeitanalyse des Algorithmus für *SSP* wird eine Funktion L_{SSP} angegeben, die neben den Eingabedaten für den Algorithmus, die Parameter p und c beinhaltet.

$$L_{SSP}(M, S, U, \tau, \beta, \delta, k, R, i, p, c) = \begin{cases} O(2^{p \cdot c + k} \cdot L_{SSP}(M, S, U, \tau, \beta, \delta, k, R, i + 1, p, c)), & \text{wenn } 1 \leq i < p \\ O(2^{p \cdot c + k} \cdot (p \cdot c + k)^2 \cdot p^2), & \text{sonst} \end{cases}$$

Da der erste Aufruf dieses Algorithmus mit $i = 1$ startet und der letzte mit $i = \#U$

endet, wird diese Methode $p-1$ mal in Zeile (12) aufgerufen und anschließend mit $i = p$ der *if-Rumpf* ab Zeile (5) ausgeführt.

Somit befindet sich die Laufzeit des Algorithmus für *SSP* in der Ordnungsklasse $O(2^{p\cdot(p\cdot c+k)} \cdot (p \cdot c + k)^2 \cdot p^2)$ und daher auch in $O(2^{\kappa(x)^3} \cdot b(n))$, wobei b ein Polynom ist und n die Eingabelänge. Da der exponentielle Teil der Laufzeit ausschließlich von $\kappa(x)$ abhängt, ist dieser Algorithmus ein *FPT-Algorithmus* für das parametrisierte *SSP* mit den Parametern k, p und c.
Dies gilt unter der Annahme, dass dieser Algorithmus korrekt ist, das heißt, dass er genau dann *true* zurückliefert, wenn für die eingegebene Instanz x gilt, dass $x \in$ *SSP*. Nachfolgend wird die Korrektheit dieses Algorithmus bewiesen.

Korrektheitsbeweis:

Es ist zu zeigen, dass folgende Äquivalenzbeziehung gilt:

$(M, S, U, \tau, \beta, \delta, k) \in SSP \Leftrightarrow f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true$

Beweis:

1. Es ist zu zeigen, dass gilt:

   $(M, S, U, \tau, \beta, \delta, k) \in SSP \Rightarrow f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true$

   Beweis:

   Sei $(M, S, U, \tau, \beta, \delta, k) \in SSP$

   $\Rightarrow$ es existiert eine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$, die alle Bedingungen des *SSP* erfüllt.

   Sei $M_{\alpha(u_{bc})} = \{s_a \mid ((s_a, m_b), u_{bc}) \in R\}$ für $u_{bc} \in U$.

   Es wird gezeigt: $M_{\alpha(u_{bc})} \in getStudentSubsets(u_{bc}, (M_1, ..., M_{\alpha(u_{bc})-1}), \tau(u_{bc}))$.

   Es gilt $m_b \in \beta(s_a)$, da R zulässig ist und daher die erste Bedingung des *SSP* erfüllt wird. Des Weiteren gilt, dass $\#M_{\alpha(u_{bc})} \leq \tau(u_{bc})$, da R ansonsten gegen die zweite Bedingung des *SSP* verstößt. Falls ein Studierender aus M_i bisher schon an den Terminen von u_{bc} verplant worden wäre, dann würde ein Terminkonflikt entstehen und R würde gegen Bedingung 3 des *SSP* verstoßen. Somit gilt für alle $u_{bc} \in U$, dass die Menge $M_{\alpha(u_{bc})} = \{s_a \mid ((s_a, m_b), u_{bc}) \in R\}$ in $getStudentSubsets(u_{bc}, (M_1, ..., M_{\alpha(u_{bc})-1}), \tau(u_{bc}))$ enthalten ist.

   Falls $getAmountOfUnassignedStudents((M_1, ..., M_{p-1}), M_p) > k$, dann gilt laut Funktionsdefinition $\#R < \#A - k$ und somit gäbe es ein Widerspruch zu $\#R \geq \#A - k$.

   Demnach gilt:

$getAmountOfUnassignedStudents((M_1, ..., M_{p-1}), M_p) \leq k$

Für $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-1}), p)$ gilt, dass neben der *if*-Bedingung aus Zeile (4) ($i = \#U$), auch die *if*-Bedingung aus Zeile (6) für $M_p \in getStudentSubsets(u_{vj}, (M_1, ..., M_{p-1}), \tau(u_{vj}))$ mit $\alpha(u_{vj}) = p$ erfüllt wird.

Somit gilt: $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-1}), p) = true$.

Für $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-2}), p-1) = true$ gilt, dass zwar die *if-Bedingung* aus Zeile (4) nicht erfüllt wird (denn $i < \#U$), aber die *if-Bedingung* aus Zeile (12) für $M_{p-1} \in getStudentSubsets(u_{fg}, (M_1, ..., M_{p-2}), \tau(u_{fg}))$ mit $\alpha(u_{fg}) = p - 1$ erfüllt wird.

Demnach gilt: $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-2}), p-1) = true$.

Für $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-3}), p-2) = true$ gilt, dass die *if-Bedingung* aus Zeile (4) nicht erfüllt wird (denn $i < \#U$), aber die *if-Bedingung* aus Zeile (12) für $M_{p-2} \in getStudentSubsets(u_{jl}, (M_1, ..., M_{p-3}), \tau(u_{jl}))$ mit $\alpha(u_{jl}) = p - 2$ erfüllt wird.

Es wurde gezeigt: $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{p-3}), p-2) = true$.

Für $1 \leq i \leq p - 3$ kann analog gezeigt werden, dass für die Berechnung von $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{i-1}), i)$ die Zeile (13) aufgeführt wird und somit gilt $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (M_1, ..., M_{i-1}), i) = true$.

Demnach gilt ebenfalls $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true$.

2. Es ist zu zeigen, dass gilt:

   $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true \Rightarrow (M, S, U, \tau, \beta, \delta, k) \in SSP$

   Beweis:

   Sei $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true$.

   Angenommen $\#S > (\sum_{u_{jl} \in U} \tau(u_{jl})) + k$, dann wird die *if-Bedingung* aus Zeile (1) erfüllt und es würde gelten $f_{SSP}(M, S, U, \tau, \beta, \delta, c, (), 1) = false$. Da dies ein Widerspruch zur Voraussetzung ist, gilt $\#S \leq (\sum_{u_{jl} \in U} \tau(u_{jl})) + k$.

1. Fall : $p = 1$

   Dann gilt $U = \{u_{11}\}$.

   Aus $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = true$ und $1 = \#U$ folgt, dass die Zeile (7) ausgeführt wird.

   Demnach gilt: $\exists a_1 \in getStudentSubsets(u_{11}, (), \tau(u_{11}))$:

   $$getAmountOfUnassignedStudents((), a_1) \leq k$$

   Sei $R = \{((s_i, m_1), u_{11}) | s_i \in a_1\}$, dann gilt $\#R \geq \#A - k$.

Auf Grund der Definition von $getAmountOfUnassignedStudents$ gilt für alle $s_i \in a_1$, dass $m_1 \in \beta(s_i)$. Somit gilt $R \subseteq A \times U$.
Nachfolgend wird geprüft, ob R alle vier Bedingungen des SSP erfüllt.

i. 1. Bedingung: $\forall((s_i, m_j), u_{ml}) \in R : j = m$.

Direkter Beweis:

Laut der obigen Zuordnung von R gilt für alle $((s_i, m_j), u_{ml}) \in R$, dass $m_j = m_1$ und $u_{ml} = u_{11}$. Somit wird die erste Bedingung des SSP erfüllt.

ii. 2. Bedingung: $\forall u_{ij} \in U : \#\{s_l | s_l \in S \wedge ((s_l, m_i), u_{ij}) \in R\} \leq \tau(u_{ij})$

Direkter Beweis:

$\#\{s_l | s_l \in S \wedge ((s_l, m_1), u_{11}) \in R\} = \#a_1$

Da $a_1 \in getStudentSubsets(u_{11}, (), \tau(u_{11}))$ gilt $\#a_1 \leq \tau(u_{11})$, woraus folgt $\#\{s_l | s_l \in S \wedge ((s_l, m_1), u_{11}) \in R\} \leq \tau(u_{11})$. Somit wird die zweite Bedingung für u_{11} erfüllt. Da $U = \{u_{11}\}$ wird diese Bedingung für alle Elemente aus U erfüllt.

iii. 3. Bedingung:

$\forall((s_i, m_d), u_{ef}), ((s_i, m_g), u_{gh}) \in R : m_d \neq m_g \Rightarrow \delta(u_{ef}, u_{gh}) = 0$

Direkter Beweis:

Da $U = \{u_{11}\}$ wird diese Bedingung erfüllt.

iv. 4. Bedingung: $\forall((s_i, m_d), u_{ef}), ((s_i, m_d), u_{gh}) \in R : u_{ef} = u_{gh}$

Direkter Beweis:

Da $U = \{u_{11}\}$ wird diese Bedingung erfüllt.

Demnach existiert eine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$, welche alle vier Bedingungen des SSP erfüllt. Folglich gilt $(M, S, U, \tau, \beta, \delta, k) \in SSP$.

2. Fall : $p > 1$

- Da $i = 1$ und $\#U = p$ gilt $i \neq \#U$, somit wird der *else*-Fall ab Zeile (10) ausgeführt.

  Angenommen für alle $a_1 \in getStudentSubsets(u_{11}, (), \tau(u_{11}))$ gilt $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (a_1), 2) = false$, dann wird Zeile (13) nie ausgeführt und es gilt $f_{SSP}(M, S, U, \tau, \beta, \delta, k, (), 1) = false$.

  Dies ist ein Widerspruch, daher gilt:

$\exists a_1 \in getStudentSubsets(u_{11}, (), \tau(u_{11})) :$
$f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1), 2) = true.$

- $i = 2$: Bei der Berechnung von $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1), 2)$ wird für $p > 2$ wieder der *else*-Fall ab Zeile (10) aufgeführt und da $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1), 2) = true$, muss die Zeile (13) erreicht werden. Somit gilt: $\exists a_2 \in getStudentSubsets(u_{ef}, (a_1), \tau(u_{ef})) :$ $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1, a_2), 3) = true$ mit $\alpha(u_{ef}) = 2$.

- Das kann bis $i = p - 1$ wiederholt werden. Somit gilt für die Berechnung von $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1, ..., a_{p-2}), p - 1)$: $\exists a_{p-1} \in getStudentSubsets(u_{pq}, (a_1, ..., a_{p-2}), \tau(u_{pq})) :$ $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1, ..., a_{p-1}), p) = true$ mit $\alpha(u_{pq}) = p - 1$.

- $i = p$: Da $f_{SSP}(M, S, U, \alpha, \tau, \beta, \delta, k, (a_1, ..., a_{p-1}), p) = true$ und $i = p = \#U$ wird die Zeile (7) ausgeführt, woraus folgt $getAmountOfUnassignedStudents((a_1, ..., a_{p-1}), a_p) \leq k$

Sei $R = \bigcup_{u_{jl} \in U} \{((s_i, m_j), u_{jl}) | s_i \in a_{\alpha(u_{jl})}\}$.
Da $\#U = p$ enhält R alle Studierendenverteilungen von $(a_1, ..., a_p)$ und somit folgt aus $getAmountOfUnassignedStudents((a_1, ..., a_{p-1}), a_p) \leq k$, dass gilt $\#R \geq \#A - k$.
Für alle $((s_i, m_j), u_{jl}) \in R$ gilt laut obiger Zuordnung, dass $s_i \in a_{\alpha(u_{jl})}$ und somit auch, dass $s_i \in getStudentSubsets(u_{jl}, (a_1, ..., a_{\alpha(u_{jl})}), \tau(u_{jl}))$.
Auf Grund der Definition von $getStudentSubsets$ gilt für alle $s \in a_i$, dass $m_j \in \beta(s)$. Somit gilt $m_j \in \beta(s_i)$, woraus folgt $(s_i, m_j) \in A$. Da die Übungsgruppenmenge U der Definitionsmenge der Funktion α entspricht, gilt $u_{jl} \in U$. Demnach gilt $R \subseteq A \times U$.

Es ist zu zeigen, dass R alle vier Bedingungen des *SSP* erfüllt.

i. 1. Bedingung: es ist zu zeigen, dass gilt: $\forall((s_x, m_y), u_{wz}) \in R : y = w$
Laut obiger Zuordnung von R wird diese Bedingung erfüllt.

ii. 2. Bedingung: es ist zu zeigen, dass gilt:

$$\forall u_{jl} \in U : \#\{s_i | s_i \in S \wedge ((s_i, m_j), u_{jl}) \in R\} \leq \tau(u_{jl})$$

Widerspruchsbeweis:

Angenommen R verstößt gegen die zweite Bedingung des *SSP*.

$$\Rightarrow \exists u_{jl} \in U : \#\{s_i | s_i \in S \wedge ((s_i, m_j), u_{jl}) \in R\} > \tau(u_{jl})$$

$$\Rightarrow \exists u_{jl} \in U : \#\{s_i | s_i \in a_{\alpha(u_{jl})}\} > \tau(u_{jl})$$

$$\Rightarrow \exists u_{jl} \in U : \#a_{\alpha(u_{jl})} > \tau(u_{jl})$$

$$\Rightarrow \exists a_{\alpha(u_{jl})} \in getStudentSubsets(u_{jl}, (a_1, ..., a_{\alpha(u_{jl})-1}), \tau(u_{jl})) :$$

$$\#a_{\alpha(u_{jl})} > \tau(u_{jl})$$

Dies widerspricht der Definition von *getStudentSubsets* und daher gilt, dass R die zweite Bedingung des *SSP* erfüllt.

iii. 3. Bedingung: es ist zu zeigen, dass gilt:

$$\forall((s_i, m_j), u_{jl}), ((s_i, m_{j'}), u_{j'l'}) \in R : m_j \neq m_{j'} \Rightarrow \delta(u_{jl}, u_{j'l'}) = 0$$

Widerspruchsbeweis:

Angenommen R erfüllt die dritte Bedingung des *SSP* nicht.

$$\Rightarrow \exists((s_i, m_j), u_{jl}), ((s_i, m_{j'}), u_{j'l'}) \in R : m_j \neq m_{j'} \wedge \delta(u_{jl}, u_{j'l'}) = 1$$

$$\Rightarrow s_i \in a_{\alpha(u_{jl})} \wedge s_i \in a_{\alpha(u_{j'l'})} \wedge \delta(u_{jl}, u_{j'l'}) = 1$$

$$\Rightarrow s_i \in getStudentSubsets(u_{jl}, R, \tau(u_{jl})) \wedge$$

$$s_i \in getStudentSubsets(u_{j'l'}, R', \tau(u_{j'l'})) \wedge \delta(u_{jl}, u_{j'l'}) = 1$$

oBdA gilt: $\alpha(u_{jl}) < \alpha(u_{j'l'})$.

Da $\alpha(u_{jl}) < \alpha(u_{j'l'})$ wurde s_i der Übung u_{jl} in R' bereits vor dem Aufruf von $getStudentSubsets(u_{j'l'}, R', \tau(u_{j'l'}))$ zugeteilt.

In $getStudentSubsets(u_{j'l'}, R', \tau(u_{j'l'}))$ ist somit s_i enthalten, obwohl s_i bereits einer anderen Übung zugeteilt wurde, die sich mit $u_{j'l'}$ terminlich überschneidet. Dies widerspricht der Funktionsdefinition von $getStudentSubsets(u_{j'l'}, R', \tau(u_{j'l'}))$ und somit wird die dritte Bedingung des *SSP* erfüllt.

iv. 4. Bedingung: es ist zu zeigen, dass gilt:

$$\forall a_i \in A : \forall(a_i, u_{kl}), (a_i, u_{mf}) \in R : u_{kl} = u_{mf}$$

Widerspruchsbeweis:

Angenommen R erfüllt die vierte Bedingung des *SSP* nicht.

$$\Rightarrow \exists a_i \in A : \exists (a_i, u_{kl}), (a_i, u_{mf}) \in R : u_{kl} \neq u_{mf}$$

Sei $a_i = (s_d, m_e)$.

$$\Rightarrow \exists ((s_d, m_e), u_{kl}), ((s_d, m_e), u_{mf}) \in R : u_{kl} \neq u_{mf}$$

$$\Rightarrow \exists ((s_d, m_e), u_{el}), ((s_d, m_e), u_{ef}) \in R : u_{el} \neq u_{ef}$$

(laut Bedingung (1) des *SSP*)

$$\Rightarrow s_d \in getStudentSubsets(u_{el}, R, \tau(u_{el})) \wedge$$

$$s_d \in getStudentSubsets(u_{ef}, R', \tau(u_{ef})) \wedge u_{el} \neq u_{ef}$$

oBdA gilt: $\alpha(u_{el}) < \alpha(u_{ef})$.

Da $\alpha(u_{el}) < \alpha(u_{ef})$ wurde s_d der Übung u_{el} in R' bereits zugewiesen bevor $getStudentSubsets(u_{ef}, R', \tau(u_{ef}))$ aufgerufen wurde. Der Studierende s_d ist in $getStudentSubsets(u_{ef}, R', \tau(u_{ef}))$ enthalten, obwohl er bereits einer anderen Übung von m_e zugeteilt wurde. Dies widerspricht der Definition von $getStudentSubsets(u_{ef}, R', \tau(u_{ef}))$ und daher wird die vierte Bedingung des *SSP* erfüllt.

Es existiert somit eine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$, welche alle Bedingungen des *SSP* erfüllt.

Demnach gilt: $(M, S, U, \tau, \beta, \delta, k) \in SSP$.

Es wurde gezeigt, dass der *FPT-Algorithmus* für das parametrisierte *SSP* mit den Parametern k, p und c korrekt ist und somit gilt, wie auch mittel der Kernelization gezeigt wurde, dass sich das *SSP* mit den Parametern k, p und c in der parametrisierten Komplexitätsklasse FPT befindet.
Diese Komplexitätsklasse enthält alle Problem-Parameter-Kombinationen für die gilt, dass das Problem effizient lösbar ist, wenn die Parameter einen kleinen Wert erhalten. Das *SSP* ist somit einfach zu lösen, falls die Parameter k, p und c in der Regel einen kleinen Wert haben. Die Eingabedaten des *SSP* lassen sich alle auf die Parameter k, p und c beschränken und somit gilt, dass die Eingabelänge klein ist, wenn die Parameter einen kleinen Wert haben. In diesem Fall ist die Parametrisierung nicht hilfreich, da ein *NP-vollständiges* Problem mit einer kleinen Eingabelänge effizient lösbar ist. Je weniger Parameter eine Problem-Parameter-Kombination enthält und desto kleiner die Parameter gewöhnlich sind, desto besser ist die Parametrisierung. Zur Verbesserung der Parametrisierung wird im folgenden Abschnitt auf den Parameter p verzichtet und analysiert, ob sich die Problem-Parameter-Kombination mit $\kappa(x) = k + c$ ebenfalls in FPT befindet.

4.4 Parameter k, c

Bisher wurde gezeigt, dass das parametrisierte *SSP* mit den Parametern k, p und c in *FPT* ist. Eine Parametrisung ist hilfreicher, wenn sie wenig Parameter enthält und diese in der Regel einen kleinen Wert zugewiesen bekommen. Der Parameter p hat normalerweise einen größeren Wert als die Parameter k und c und daher wird in dem zu untersuchenden parametrisierten *SSP* auf den Parameter p verzichtet und somit bleiben als Parameter k und c. Falls diese Problem-Parameter-Kombination ebenfalls in *FPT* ist, dann existiert ein Algorithmus für *SSP* dessen exponentieller Teil der Laufzeit nicht von p abhängt und somit wäre dieses Resultat besser als die Erkenntnis der *fixed-parameter tractability* der vorherigen Problem-Parameter-Kombination, die zusätzlich p als Parameter enthält.
In Kapitel Komplexität 3 wurde mit Hilfe einer *Polynomialzeit-Many-one-Reduktion* von *3-SAT* auf *SSP* gezeigt, dass *SSP* ein *NP-vollständiges* Problem ist. In dieser Reduktion wurde $k = 0$ und $c = 3$ gesetzt und somit gilt die *NP-Vollständigkeit* für *SSP*, sogar für konstante Werte dieser Parameter. Falls *SSP* mit den Parametern k und c in *FPT* ist, existiert ein *FPT-Algorithmus*, dessen Laufzeit in $O(f(k, c) \cdot b(n))$ mit einem Polynom b und der Eingabelänge n ist. Dies bedeutet, dass für das Teilproblem *SSP* mit konstantem $k = 0$ und $c = 3$ die Funktion $f(k, c)$ gleich $f(0, 3)$ ist und somit einen konstanten Wert liefert. Dieses Teilproblem ist somit in polynomieller Zeit lösbar. Dies widerspricht der *Polynomialzeit-Many-one-Reduktion* von 3-SAT auf *SSP*, da auf dieses Teilproblem reduziert wird und somit die *NP-Vollständigkeit* gilt. Auf Grund dieses Widerspruchs ist *SSP* mit den Parametern k und c nicht in der Komplexitätsklasse *FPT*.

Korollar 2. *SSP ∈ para-NP.*

Beweis. In Folge des Satzes 4.2 [FG06] gilt, dass sich das parametrisierte *SSP* mit einer beliebigen Parametrisierung in der Komplexitätsklasse *para-NP* befindet.

Satz 4.2. *Sei Q ein Problem, welches nicht parametrisiert ist. Falls $Q \in NP$, dann folgt daraus $(Q, \kappa) \in$ para-NP, für jede Parametrisierung κ.*

Demnach gilt, dass die Komplexitätsklasse *para-NP* unter anderem das parametrisierte *SSP* mit der Parametrisierung $\kappa(x) = k + c$ enthält.

Korollar 3. *Das SSP mit den Parametern k und c ist para-NP-vollständig.*

Beweis. Zum Nachweis der *para-NP-Vollständigkeit* des parametrisierten *SSP* mit den Parametern k und c wird der kommende Satz angewendet [FG06]. Da dieser Satz den Begriff *Slices* verwendet, wird zuvor die zugehörige Definition 8 angegeben [MSTV12].

Definition 8. *Slice*

Der k. Slice eines parametrisierten Problems (Q, κ) ist die Menge

$$(Q, \kappa)_k = \{x = Q \mid \kappa(x) = k\}$$

Satz 4.3. *Sei (Q, κ) ein nichttriviales parametrisiertes Problem in para-NP. Dann sind die folgenden Aussagen äquivalent:*

1. *(Q, κ) ist para-NP-vollständig unter FPT-Reduktionen.*
2. *Die Vereinigung von endlich vielen Slices von (Q, κ) ist NP-vollständig. Das heißt, es gibt $l, m_1, ..., m_l \in \mathbb{N}$, so dass $(Q, \kappa)_{m_1} \cup ... \cup (Q, \kappa)_{m_l}$ unter Polynomialzeit-Many-one-Reduktionen NP-vollständig ist.*

Es gilt somit, dass eine Problem-Parameter-Kombination (Q, κ) genau dann *para-NP vollständig* ist, wenn eine Menge von Werten für den Parameter existiert, so dass die Vereinigung der Instanzen von Q, deren Parameter einen Wert dieser Menge hat, *NP-vollständig* ist.
Sei die Menge der Wert $\kappa(x)$ gleich $\{3\}$. Die Vereinigung der *SSP*-Instanzen für die gilt $\kappa(x) = k + c = 3$ enthält unter anderem die *SSP*-Instanzen mit $k = 0$ und $c = 3$, für die bereits die *NP-Vollständigkeit* gezeigt wurde. In Folge dessen gilt, dass die Vereinigung der *SSP*-Instanzen mit $k + c = 3$ unter *Polynomialzeit-Many-one-Reduktion NP-vollständig* ist und folglich gilt die *para-NP-Vollständigkeit* für das parametrisierte *SSP* mit dem Parameter $k + c$.

4.5 Parameter k,c,u,t

Korollar 4. *Das SSP mit den Parametern k, c, u und t ist para-NP-vollständig.*

Beweis. In der *SSP*-Instanz, welche aus der *Polynomialzeit-Many-one-Reduktion* von *3-SAT* auf *SSP* entstanden ist, hat jedes Modul genau zwei Übungsgruppen, unabhängig von der konkreten *SSP*-Instanz. Des Weiteren hat keine Übungsgruppe der erzeugten Instanz mehr als 3 Termine und somit ist bei der Reduktion der Wert für t ebenfalls fest. Dies bedeutet, dass *SSP* nicht nur mit festen Werten für k und c *NP-vollständig* ist, sondern das *SSP* sogar mit festen Werten für k, c, u und t *NP-vollständig* ist.
Die Menge der *SSP*-Instanzen mit $k + c + u + t = 8$ beinhaltet unter anderem die *SSP*-Instanzen mit $k = 0, c = 3, u = 2$ und $t = 3$ für die bereits die *NP-Vollständigkeit* bewiesen wurde. Auf Grund dieses Beweises gilt die *NP-Vollständigkeit* für die Vereinigung der *SSP*-Instanzen mit $\kappa(x) = k + c + u + t = 8$ unter *Polynomialzeit-Many-one-Reduktion*. Folglich gilt die *para-NP-Vollständigkeit* für das parametrisierte *SSP* mit der Parametrisierung $\kappa(x) = k + c + u + t$.

Korollar 5. *Sei (Q, κ) ein parametrisiertes Problem mit $Q \in NP$ und seien M und N jeweils eine nicht-leere Menge von Eingabedaten von Q, die als Wert eine natürliche Zahl haben, mit $N \subseteq M$. Falls $\kappa(x) = \sum_{p \in M} p$ und (Q, κ) para-NP-vollständig ist, dann gilt die para-NP-Vollständigkeit ebenfalls für alle (Q, ν) mit $\nu(x) = \sum_{p \in N} p$.*

Beweis.

1. Da $Q \in NP$ gilt $(Q, \nu) \in$ *para-NP* (laut Satz 4.2).
2. Laut Voraussetzung gilt: (Q, κ) ist *para-NP-vollständig.*

   $\Rightarrow$ es existiert eine endliche Menge $W \subseteq \mathbb{N}$, so dass die Vereinigung aller Instanzen von Q mit $\kappa(x) \in W$ *NP-vollständig* ist.

   $\Rightarrow$ es existiert mindestens ein *NP-vollständiges* Teilproblem Q' von Q mit konstantem Parameter $\kappa(x) \in W$.

   Sei Q' ein *NP-vollständiges* Teilproblem von Q mit konstantem Parameter $\kappa(x) \in W$.

   $\Rightarrow$ für alle Instanzen x^* von Q' gilt: $\exists w_l \in W : \kappa(x^*) = w_l$.

   $\Rightarrow$ für alle Instanzen x^* von Q' gilt: $\exists w_l \in W : \sum_{p \in M} p = w_l$.

   $\Rightarrow$ für alle Instanzen x^* von Q' gilt: $\exists w_l \in W : \sum_{p \in N} p = w_l - \sum_{p \in M \setminus N}$.

   Sei Q'' ein Teilproblem von Q mit einem konstantem Parameter $\nu(x) = w_l - \sum_{p \in M \setminus N}$

   $\Rightarrow$ für alle Instanzen x^* von Q' gilt: x^* ist eine Instanz von Q''.

   $\Rightarrow$ Q'' ist ein *NP-vollständiges* Teilproblem von Q mit konstantem Parameter $\nu(x) \in W''$ mit $W'' = \{w_l - \sum_{p \in M \setminus N}\}$.

   $\Rightarrow$ es existiert eine endliche Menge $W'' \subseteq \mathbb{N}$, so dass die Vereinigung aller Instanzen von Q mit konstantem Parameter $\nu(x) \in W''$ *NP-vollständig* ist.

   $\Rightarrow$ (Q, ν) ist *para-NP-vollständig.*

Es wurde gezeigt, dass (Q, κ) mit $\kappa(x) = k+c+u+t$ *para-NP-vollständig* ist und auf Grund des Korollars 5 gilt die *para-NP-Vollständigkeit* ebenfalls für alle (Q, ν) mit $\nu(x) = \sum_{p \in N} p$ und $N \subseteq \{k, c, u, t\}$. Demnach gilt, dass alle parametrisierten *SSP*, deren Parameter eine Teilmenge von $\{k, c, u, t\}$ sind, *para-NP-vollständig* sind.

4.6 Parameter p

In Kapitel 4.3 wurde gezeigt, dass sich das parametrisierte *SSP* mit den Paramtern k, p und c in der Komplexitätsklasse *FPT* befindet. Es wird versucht diese Parametrisierung um mindestens einen Parameter zu reduzieren, ohne die Eigenschaft der *fixed-parameter tractability* zu verletzen. Falls dies gelingt, wäre die neue Parametrisierung besser, da der exponentielle Teil der Laufzeit eines zugehörigen *FPT* Algorithmus von weniger Parametern abhängt. Im vorherigen Abschnitt wurde das parametrisierte *SSP* unter anderem mit den Parametern k und c untersucht mit dem Ergebnis, dass diese Problem-Parameter-Kombination nicht in *FPT* liegt, sondern *para-NP-vollständig* ist.
Der eliminierte Parameter p wird in diesem Kapitel betrachtet, indem das *parametrisierte SSP* mit dem Parameter p untersucht wird. Dieser Parameter repräsentiert die Anzahl der Übungsgruppen, die es insgesamt gibt. In der Regel ist dieser Wert nicht sehr klein, aber wenn diese Problem-Parameter-Kombination in *FPT* sein sollte, dann wäre dies eine bessere Erkenntnis als die *fixed-parameter tractability* von *SSP* mit den Parametern p, k und c.

Lemma 4.4. *Das SSP mit dem Parameter p ist in FPT.*

Beweis. Es gibt viele verschiedene Entwurfsmuster zum Erstellen eines *FPT* Algorithmus. Eine dieser Varianten baut auf einem *Integer Linearen Programm (ILP)* auf. Der Grundstein für diese Idee liefert ein Resultat von *Hendrik W. Lenstra*. Es besagt, dass *ILP* mit einer konstanten Anzahl von Variablen in linearer Zeit gelöst werden können [Len83]. Das Resultat basiert auf dem *Integer Lineare Programming Feasibility Problem* und daher wird dieses zuerst definiert bevor näher auf das Ergebnis von *Lenstra* eingegangen wird.

Definition 9. *Integer Linear Programming Feasibility (ILPF)*

Instanz: *Eine Matrix $W^{m \times n}$ und ein Vektor $\vec{b} \in \mathbb{Z}^m$*
Frage: *Existiert eine Belegung der Variablen, die alle Ungleichungen des Integer Linearen Programms erfüllt?*

Ein *ILP* besteht aus einer Menge von Ungleichungen, die linke Seite wird als Matrix dargestellt und die rechte Seite der Ungleichungen als Vektor. Somit repräsentieren die Matrix W und der Vektor b ein *ILP* mit m Ungleichungen und n Variablen.
Wird diesem Problem ein *ILP* für *SSP* übergeben, so gilt, dass die Frage des *ILPF* genau dann mit *wahr* beantwortet wird, wenn die Frage des *SSP* ebenfalls *wahr* ist.
Lenstra hat im Jahr 1983 einen Algorithmus für das *ILPF* publiziert, der bei einer festen Anzahl von Variablen dieses Problem in linearer Zeit löst. Somit gilt folgender Satz [Ros11]:

Satz 4.5. *Das ILPF mit dem Parameter k, welches die Anzahl der Variablen repräsentiert, ist in FPT.*

Falls ein *ILP* für *SSP* existiert, dessen Anzahl von Variablen ausschließlich von dem Parameter p abhängt, dann gilt laut Satz 4.5, dass ein *FPT-Algorithmus* für das parametrisierte *SSP* mit dem Parameter p existiert. Daher wird in diesem Abschnitt ein *ILP* für *SSP* entworfen. Hierfür sind vorab Einführungen von weiteren Variablen notwendig:

Sei $U^* = \{u_{ij} \mid 1 \leq i \leq \#M \wedge j - 1 = max\{q \mid u_{iq} \in U\}\}$

$$\forall u_{ij} \in U^* \cup U^* : c_{ij} = \begin{cases} k, & \text{wenn } u_{ij} \in U^* \\ \tau(u_{ij}), & \text{sonst } (u_{ij} \in U) \end{cases}$$

$\forall u_{ij} \in U^* : \forall u_{rq} \in (U \cup U^*) : u_{ij} \neq u_{rq} \Rightarrow \delta(u_{ij}, u_{rq}) = 0.$

$L \subseteq \mathcal{P}(U \cup U^*) \backslash \{l \mid l \in \mathcal{P}(U \cup U^*) \wedge \exists u_{ir}, u_{js} \in l : (u_{ir} \neq u_{js} \wedge (\delta(u_{ir}, u_{js}) = 1 \vee i = j))\}$

$\forall l \in L : x_l =$ Die Anzahl der Studierenden, die den Übungsgruppen aus l zugewiesen werden können.

Die Menge U^* enthält für jedes Modul eine zusätzliche Übungsgruppe, die sich zeitlich mit keiner anderen Übungsgruppe überschneidet. Eine Übungsgruppe u_{ij} aus U^* beinhaltet alle Studierenden, die keiner Übungsgruppe aus U des gewählten Moduls m_i konfliktfrei zugeordnet werden können. Laut der Definition des *SSP* dürfen maximal k Studierende/Modul-Kombinationen unverteilt bleiben, daher hat jede Übungsgruppe aus U^* die Kapazität k. Neben der Menge U^* wird eine weitere Menge, namens L, eingeführt. Diese Menge enthält Teilmengen von $U \cup U^*$. Für jede dieser Teilmengen gilt, dass deren Übungsgruppen sich zeitlich nicht überschneiden und sie maximal eine Übungsgruppe pro Modul enthalten. Zu jeder Teilmenge l aus L existiert eine Variable x_l, deren Wert die Anzahl der Studierenden angibt, die den Übungsgruppen dieser Teilmenge zugeordnet werden können. Die Variablen $\{x_l | l \in L\}$ entsprechen den Variablen des *ILP* für das *SSP*, welches nachfolgend angegeben wird:

ILP:

$$(1)\ \forall u_{ij} \in U \cup U^* : \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} x_l \leq c_{ij}$$

$$(2)\ \forall D \subseteq M : \sum_{l \in \{g | g \in L \wedge \{m_j | u_{jf} \in g\} = D\}} x_l \geq \#\{s_i \mid s_i \in S \wedge \beta(s_i) = D\}$$

$$(3)\ \sum_{l \in L} x_l \cdot \#(l \cap U^*) \leq k$$

Das *SSP* beinhaltet vier Bedingungen, die eine zulässige Lösung einzuhalten hat. Zum Nachweis der Korrektheit eines *ILP* muss gezeigt werden, dass genau dann eine Belegung der Variablen existiert, so dass die Ungleichungen erfüllt werden, wenn es eine zulässige Relation für *SSP* gibt. Wird eine Relation R für das *SSP* erzeugt, indem in R zu jedem $l \in L$ maximal viele Studierenden/Modul-Kombinationen den Übungen aus l zugeordnet werden (höchsten x_l viele), dann ist sicher zu stellen, dass R alle vier Bedingungen des *SSP* erfüllt und $\#R \geq \#A - k$ ist. Bei der Erstellung der Relation ist zu beachten, dass nur Studierende/Modul-Kombinationen einer Übungsgruppe zugeordnet werden, die Bedingung (1) des *SSP* erfüllen ($\forall((s_i, m_j), u_{fg}) \in R : j = f$). Damit die erstellte Relation R für das *SSP* zulässig ist, existieren die Bedingungen (1), (2) und (3) des *ILP*. Im Folgenden wird jede Bedingung des *ILP* näher erläuert.

1. Zu jeder Übungsgruppe existiert eine Kapazität, das heißt es dürfen nicht mehr Studierende/Modul-Kombinationen einer Übungsgruppe zugeteilt werden als die Kapazität vorgibt. R weist jeder Menge von Übungsgruppen l maximal x_l viele Studierende/Modul-Kombinationen zu. Eine Übungsgruppe hat somit maximal für jedes l, in der die Übungsgruppe enthalten ist, x_l viele Teilnehmer. Zum Einhalten der Kapazität existiert die Bedingung (1): $\forall u_{ij} \in U \cup U^* : \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} x_l \leq c_{ij}$.

2. Die Definition des *SSP* besagt, dass $\#R \geq \#A - k$. Dies bedeutet, dass alle bis auf maximal k Studierende/Modul-Kombinationen einer Übung zuzuteilen sind. Für jedes $l \in L$ gilt, dass maximal x_l viele Studierende/Modul-Kombinationen den Übungen aus l zugeordnet werden dürfen, wobei die Bedingung (1) des *SSP* einzuhalten ist. Für eine Menge von Modulen D gilt, dass die Anzahl der Studierenden, die diese Module wählten, der Kardinalität der Menge von $\{s_i \mid s_i \in S \wedge \beta(s_i) = D\}$ entspricht. Die Summe $\sum_{l \in \{g | g \in L \wedge \{m_j | u_{jf} \in g\} = D\}} x_l$ entspricht der Anzahl der Studierenden, die einer Menge l, welche genau zu jedem der Module aus D eine Übungsgruppe beinhaltet, zugeordnet werden können. Somit stellt Bedingung (2) des *ILP* sicher, dass für jede Menge von Modulen D mindestens so viele Studierende/Modul-Kombinationen den zugehörigen Übungsgruppenmengen l zugewiesen werden können, wie die Anzahl der Studierenden ist, die dieses Menge wählten.

3. Wenn es zu dem *ILP* eine Lösung gibt, die alle Bedingungen erfüllt, bedeutet dies das alle Studierende/Modul-Kombinationen zu Übungsgruppen verteilt werden können. Jedoch ist zu beachten, dass l nicht nur Übungsgruppen des *SSP* enthält, sondern die Übungen aus U^* hinzugefügt wurden. Da diese Übungsgruppen in *SSP* nicht enthalten sind, ist die Menge der Studierenden/Modul-Kombinationen, die nicht verteilt werden können, die Anzahl der Teilnehmer der Übungsgruppen aus U^*. Die Definition des *SSP* besagt, dass insgesamt maximal k Studierende/Modul-Kombinationen unverteilt bleiben dürfen, daher beinhaltet das *ILP* die Bedingung (3) die dies sicher stellt.

Genau dann, wenn es eine Belegung des *ILP* gibt, welche die Bedingungen (1), (2) und (3) erfüllt, existiert, laut dem nachfolgenden Beweis, eine zulässige Relation R für *SSP*. $sol_{ILP}(x) = true$ bedeutet, dass zu dem *ILP*, welches aus der *SSP*-Instanz x erzeugt wurde, eine Belegung existiert, die alle Bedingungen des *ILP* erfüllt. Falls keine solche Belegung der Variablen $\{x_l \mid l \in L\}$ existiert, dann gilt $sol_{ILP}(x) = false$.

Korrektheitsbeweis:

Das *SSP* kann mit Hilfe des *ILP* genau dann gelöst werden, falls gilt:
$sol_{ILP}(x) \neq \emptyset \Leftrightarrow x \in SSP$.

1. Es ist zu zeigen, dass gilt: $x \in SSP \Rightarrow sol_{ILP}(x) \neq \emptyset$.

   Direkter Beweis:

   Sei $x \in SSP$.

   $\Rightarrow$ es existiert eine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$, so dass R alle Bedingungen des *SSP* erfüllt.

   Für jeden Studierenden s_i wird eine Menge B_i erstellt, die alle Übungen enthält, die dem Studierenden in R zugeteilt wurden. Somit ist B_i folgendermaßen definiert: $B_i = \{u_{jq} \mid u_{jq} \in U \wedge ((s_i, m_j), u_{jq}) \in R\}$ für $1 \leq i \leq \#S$.

   Laut Definition von L gilt:

   $B_i \in L \Leftrightarrow (B_i \subseteq (U \cup U^*) \wedge \forall u_{jl}, u_{qs} \in B_i : u_{jl} \neq u_{qs} \Rightarrow (j \neq q \wedge \delta(u_{jl}, u_{qs}) = 0))$.

   Ob B_i diese Eigenschaften erfüllt, wird nachfolgend überprüft:

   a) Laut der Definition von B_i gilt $B_i \subseteq U$, woraus folgt: $B_i \subseteq (U \cup U^*)$.

   b) Angenommen $\exists 1 \leq i \leq \#S : \exists u_{jl}, u_{qs} \in B_i : u_{jl} \neq u_{qs} \wedge j = q$.

   $$\Rightarrow \exists((s_i, m_j), u_{jl}), ((s_i, m_q), u_{qs}) \in R : u_{jl} \neq u_{qs}$$
   $$\Rightarrow \exists((s_i, m_j), u_{jl}), ((s_i, m_j), u_{js}) \in R : u_{jl} \neq u_{js} (\text{da } j = q)$$
   $$\Rightarrow \text{ Bedingung 4 des } SSP \text{ wird verletzt.}$$

   Dies führt zu einem Widerspruch da R zulässig ist und daher gilt:

   $$\forall 1 \leq i \leq \#S : \forall u_{jl}, u_{qs} \in B_i : u_{jl} \neq u_{qs} \Rightarrow j \neq q$$

   c) Angenommen $\exists 1 \leq i \leq \#S : \exists u_{jl}, u_{qs} \in B_i : u_{jl} \neq u_{qs} \wedge \delta(u_{jl}, u_{qs}) = 1$.

   $$\Rightarrow \exists((s_i, m_j), u_{jl}), ((s_i, m_q), u_{qs}) \in R : u_{jl} \neq u_{qs} \wedge \delta(u_{jl}, u_{qs}) = 1$$
   $$\Rightarrow \exists((s_i, m_j), u_{jl}), ((s_i, m_q), u_{qs}) \in R : m_j \neq m_q \wedge \delta(u_{jl}, u_{qs}) = 1$$

Die letzte Implikation basiert auf dem vorherigen Beweis. Somit wurde gezeigt, dass die Bedingung 3 des *SSP* verletzt wird.
Dies führt zu einem Widerspruch daher gilt:

$$\forall 1 \leq i \leq \#S : \forall u_{jl}, u_{qs} \in B_i : u_{jl} \neq u_{qs} \Rightarrow \delta(u_{jl}, u_{qs}) = 0$$

Somit gilt $\forall 1 \leq i \leq \#S : B_i \in L$ bzw. $\exists l \in L : B_i = l$.

Die Belegungen der Variablen des *ILP* sind wie folgt:

$$\forall l \in L : x_l = \#\{s_r \mid s_r \in S \wedge B_r = l \backslash U^* \wedge \beta(s_r) = \{m_j \mid u_{jr} \in l\}\} \tag{4.1}$$

Demnach entspricht der Wert einer Variablen x_l der Anzahl Studierender, die einerseits in R genau den Übungen aus l, die sich nicht in U^* befinden, zugeteilt wurden und andererseits genau die Menge von Modulen wählten, die eine zugehörige Übung in l haben.

Es ist zu prüfen, ob diese Belegung zulässig ist, das heißt, ob alle Bedingungen des *ILP* mit dieser Belegungen erfüllt werden:

a) Es ist zu zeigen, dass Bedingung (1) erfüllt wird und somit gilt:

$\forall u_{ij} \in U \cup U^* : \sum_{l \in \{f \mid f \in L \wedge u_{ij} \in f\}} x_l \leq c_{ij}$

Widerspruchsbeweis:

Angenommen Bedingung (1) wird nicht erfüllt.

$\Rightarrow \exists u_{ij} \in U \cup U^* : \sum_{l \in \{f \mid f \in L \wedge u_{ij} \in f\}} x_l > c_{ij}$

1. Fall : $u_{ij} \in U$

$$\begin{aligned}
\tau(u_{ij}) &= c_{ij} && (4.2)\\
&< \sum_{l \in \{f \mid f \in L \wedge u_{ij} \in f\}} x_l && (4.3)\\
&= \sum_{l \in \{f \mid f \in L \wedge u_{ij} \in f\}} \#\{s_r \mid s_r \in S \wedge B_r = l \backslash U^* \wedge && (4.4)\\
&\qquad \beta(s_r) = \{m_j \mid u_{jq} \in l\}\} && (4.5)\\
&\leq \sum_{l \in \{f \mid f \in L \wedge u_{ij} \in f\}} \#\{s_r \mid s_r \in S \wedge Br = l \backslash U^*\} && (4.6)\\
&\leq \#\{s_r \mid s_r \in S \wedge u_{ij} \in B_r\} && (4.7)\\
&= \#\{s_r \mid s_r \in S \wedge ((s_r, m_i), u_{ij}) \in R\} && (4.8)
\end{aligned}$$

Erläuterung:
Laut der Definition des *ILP* gilt die Zeile (4.2), woraus sich auf Grund der Beweisannahme die Zeile (4.3) schließen lässt. Unter Verwendung

der Gleichung (4.1) ergeben sich die Zeilen (4.4) und (4.5). Die Bedingungen der Menge aus Zeile (4.4) und (4.5) werden im nächsten Schritt abgewächt, daher kann sich die Kardinalität der Mengen aus (4.6) höchstens erhöhen. Für jedes $l \in L$ mit $u_{ij} \in L$ und $B_r = l \backslash U^*$ gilt, dass $u_{ij} \in B_r$, da $u_{ij} \in U$ und somit $u_{ij} \notin U^*$. Somit ist jedes Element der Menge aus (4.6) in der Menge (4.7) enthalten. Da s_r nicht mehrfach in (4.6) gezählt wird, ist der Wert aus Zeile (4.7) nicht größer als der Wert der Zeile (4.6). Laut der Definition von B_r gilt für alle u_{ij} aus B_r, dass $((s_r, m_i), u_{ij}) \in R$, somit lässt sich die Zeile (4.7) durch die Zeile (4.8) ersetzen.
Es wurde somit gezeigt, dass für u_{ij} gilt: $\tau(u_{ij}) < \#\{s_r | s_r \in S \wedge ((s_r, m_i), u_{ij}) \in R\}$. Da dies gegen die zweite Bedingung des *SSP* verstößt, verletzt R die zweite Bedingung.
Dies führt zu einem Widerspruch, da R zulässig ist.

2. Fall : $u_{ij} \in U^*$

$$k = c_{ij} \tag{4.9}$$

$$< \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} x_l \tag{4.10}$$

$$= \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} \#\{s_r \mid s_r \in S \wedge B_r = l \backslash U^* \wedge \beta(s_r) = \{m_y \mid u_{yq} \in l\} \tag{4.11}$$

$$\leq \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} \#\{s_r \mid s_r \in S \wedge m_i \in \beta(s_r) \wedge B_r = l \backslash U^*\} \tag{4.12}$$

$$= \sum_{l \in \{f | f \in L \wedge u_{ij} \in f\}} \#\{s_r \mid s_r \in S \wedge (s_r, m_i) \in A \wedge B_r = l \backslash U^*\} \tag{4.13}$$

$$= \#\{s_r \mid s_r \in S \wedge (s_r, m_i) \in A \wedge \forall u_{ij} \in U : ((s_r, m_i), u_{ij}) \notin R\} \tag{4.14}$$

$$= \#\{(s_r, m_i) \mid (s_r, m_i) \in A \wedge \forall u_{ij} \in U : ((s_r, m_i), u_{ij}) \notin R\} \tag{4.15}$$

$$\leq \#A - \#R \tag{4.16}$$

Erläuterung:
Laut der Definition des *ILP* gilt die Gleichung aus Zeile (4.9), woraus sich auf Grund der Beweisvoraussetzung die Zeile (4.10) schließen lässt. Wendet man darauf die Zeile (4.1) an, so entsteht die Zeile (4.11). Für jeden Studierenden s_r mit $\beta(s_r) = \{m_y | u_{yq} \in l\}$ gilt, dass $m_i \in \beta(s_r)$, da $u_{ij} \in l$. Somit befinden sich alle Elemente der Menge aus Zeile (4.11) ebenfalls in der Menge aus Zeile (4.12). Dies bedeutet, dass die Kardinalität der Menge aus Zeile (4.12) mindestens so groß ist wie die Kardinalität der Menge aus Zeile (4.11).

Laut der Definition von A gilt für alle $m_i \in \beta(s_r)$, dass (s_r, m_i) in A, daher lässt sich die Zeile (4.12) zur Zeile (4.13) umformen. Für jeden Studierenden s_r mit $u_{ij} \in l$ und $B_r = l \backslash U^*$ gilt, dass keine Übung von m_i in B_r existiert und somit s_r keiner Übung von m_i in R zugewiesen werde konnte. Da in Zeile (4.13) kein Studierender mehrfach gezählt wird, denn zu jedem Studierenden s_r existiert genau eine Menge B_r, ist der Wert von Zeile (4.13) und Zeile (4.14) gleich. In Zeile (4.14) wird die Anzahl der Studierenden, die keiner Übung des Moduls m_i zugeordnet werden konnten, bestimmt. Da R eine Übung immer einer Studierenden/Modul-Kombination zuordnet, entspricht (4.14) der Anzahl Studierenden/Modul-Kombinationen, die als Modul m_i enthalten und in R nicht verteilt werden konnten. Somit entspricht der Wert der Zeile (4.14) dem Wert der Zeile (4.15). Die Anzahl der Studierenden/Modul-Kombinationen mit m_i, die nicht in R enthalten sind, ist maximal so groß wie die Gesamtzahl der Studierenden/Modul-Kombinationen, die in R nicht verteilt werden konnten. Da jedes Element aus R genau eine Studierenden/Modul-Kombination enthält und keine Studierenden/Modul-Kombination mehrfach in R enthalten ist, entspricht die Kardinalität von R gleich der Anzahl Studierenden/Modul-Kombinationen, die in R verteilt werden konnten. Somit ist $\#A - \#R$ die Anzahl der Studierenden/Modul-Kombinationen, die keine Übung zugeteilt bekommen konnten. Demnach ist der Wert aus Zeile (4.16.) mindestens so groß wie der Wert aus Zeile (4.15).
Es wurde gezeigt, dass gilt $\#A - \#R > k$, das heißt $\#R < \#A - k$. Dies führt zu einem Widerspruch, da $\#R \geq \#A - k$.

Somit gilt, dass Bedingung (1) des *ILP* erfüllt wird.

b) Es ist zu zeigen, dass die Bedingung (2) des *ILP* erfüllt wird und somit gilt:
$\forall D \subseteq M : \sum_{l \in \{g | g \in L \wedge \{m_j | u_{jf} \in g\} = D\}} x_l \geq \#\{s_i \mid s_i \in S \wedge \beta(s_i) = D\}$
Direkter Beweis:

Für alle $D \subseteq M$ gilt:

$\sum_{l \in \{g | g \in L \wedge \{m_j | u_{jf} \in g\} = D\}} x_l$
$= \sum_{l \in \{g | g \in L \wedge \{m_j | u_{jf} \in g\} = D\}} \#\{s_p \mid s_p \in S \wedge B_p = l \backslash U^* \wedge \beta(s_p) = D\}$
$= \#\{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l \in L : \{m_j \mid u_{ji} \in l\} = D \wedge B_p = l \backslash U^*\}$

Es wird gezeigt, dass für alle $s_r \in S$ mit $\beta(s_r) = D$ gilt :
$s_r \in \{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l \in L : \{m_j \mid u_{ji} \in l\} = D \wedge B_p = l \backslash U^*\}$
Beweis:

$\forall s_r \in S$ mit $\beta(s_r) = D$ gilt:

1. Fall : Sei $\#B_r = \#\beta(s_r)$ und $l_q = B_r$ mit $l_q \in L$:

   Da $\{m_j \mid u_{ji} \in B_r\} = \{m_j \mid ((s_r, m_j), u_{ji}) \in R\}$ gilt

   $\{m_j \mid u_{ji} \in B_r\} \subseteq \beta(s_r)$.

   Aus der Voraussetzung $\#B_r = \#\beta(s_r)$ folgt $\{m_j \mid u_{ji} \in B_r\} = \beta(s_r)$.

   Somit gilt $\exists l_q \in L : \{m_j \mid u_{ji} \in l_q\} = D \wedge B_r = l_q$.

   Da $B_r \cap U^* = \emptyset \wedge l_q = B_r$ gilt $l_q \cap U^* = \emptyset$ und somit existiert ein $l_q \in L$ mit $\{m_j \mid u_{ji} \in l_q\} = D \wedge B_r = l_q \backslash U^*$.

   Demnach gilt:

   $s_r \in \{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l_q \in L : \{m_j \mid u_{ji} \in l_q\} = D \wedge$
   $B_p = l_q \backslash U^*\}$

2. Fall : Sei $\#B_r > \#\beta(s_r)$, dann wurden dem Studierenden s_r mehr Übungsgruppen zugewiesen als die Anzahl seiner gewählten Module. Da einem Studierenden nur Übungsgruppen eines gewählten Moduls zugewiesen werden dürfen (Bedingung 1 des *SSP*) gilt somit, dass einem Studierenden mehrere Übungsgruppen eines Moduls zugewiesen wurden. Dies verstößt gegen Bedingung 4 des *SSP*, woraus sich ein Widerspruch ergibt, da R alle Bedingungen des *SSP* erfüllt.

3. Fall : Sei $\#B_r < \#\beta(s_r)$ und $l_{q'} = B_r$ mit $l_{q'} \in L$.

   Weil für alle $u_{ij} \in B_r$ gilt $m_i \in \beta(s_r)$ folgt $\{m_j \mid u_{ji} \in l_{q'}\} \subset \beta(s_r)$.

   Da $B_r \cap U^* = \emptyset \wedge l_{q'} = B_r$ gilt $l_{q'} \cap U^* = \emptyset$.

   Sei $\beta'(s_r) = \beta(s_r) \backslash \{m_j \mid u_{ji} \in l_{q'}\}$ und $l_{q''} = \{u_{ji} \mid u_{ji} \in U^* \wedge$
   $m_j \in \beta'(s_r)\}$.

   Somit ist $l_{q''}$ eine Teilmenge von U^*, woraus folgt $l_{q''} \in U \cup U^*$.

   Zu keinem Modul existieren mehrere Übungen in U^*, somit enthält $l_{q''}$ maximal eine Übung pro Modul.

   Des Weiteren gilt für alle $\forall u_{ji} \in U^* : \forall u_{qw} \in U \cup U^* : \delta(u_{ji}, u_{qw}) = 0$, woraus folgt $\forall u_{ji}, u_{qw} \in l_{q''} : \delta(u_{ji}, u_{qw}) = 0$.

   Demnach gilt $l_{q''} \in L$.

   Sei $l_q = l_{q'} \cup l_{q''}$.

   Es ist zu prüfen, ob $l_q \in L$:

   Aus $l_{q'} \subseteq U$ und $l_{q''} \subseteq U^*$ folgt $l_q \subseteq U \cup U^*$.

Aus $l_{q'} \in L$ folgt $\forall u_{ij}, u_{qw} \in l_{q'} : u_{ij} \neq u_{qw} \Rightarrow (\delta(u_{ij}, u_{qw}) = 0 \wedge i \neq q)$.
Ebenso folgt aus $l_{q''} \in L$: $\forall u_{ij}, u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \Rightarrow (\delta(u_{ij}, u_{qw}) = 0 \wedge i \neq q)$.
Angenommen $l_q \notin L$, dann gilt $\exists u_{ij} \in l_{q'} : \exists u_{rw} \in l_{q''} : u_{ij} \neq u_{qw} \wedge (\delta(u_{ij}, u_{qw}) = 1 \vee i = q)$.

1. Fall Angenommen es gilt:

$$\exists u_{ij} \in l_{q'} : \exists u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \wedge \delta(u_{ij}, u_{qw}) = 1$$

Da $u_{qw} \in l_{q''}$ gilt $u_{qw} \in U^*$.
Weiterhin gilt:

$$\forall u_{qw} \in U^* : \forall u_{xy} \in U \cup U^* : u_{qw} \neq u_{xy} \Rightarrow \delta(u_{qw}, u_{xy}) = 0$$

Da $u_{ij} \in l_{q'}$ gilt $u_{ij} \in U$ und somit auch $u_{ij} \in U \cup U^*$.

$$\Rightarrow (u_{qw} \neq u_{ij} \Rightarrow \delta(u_{qw}, u_{ij}) = 0)$$

Da $u_{qw} \in U^*$ und $u_{ij} \in U$ und $U \cap U^* = \emptyset$ gilt $u_{qw} \neq u_{ij}$.
Somit gilt $\delta(u_{qw}, u_{ij}) = 0$.
Dies widerspricht der Beweisannahme und somit gilt:

$$\forall u_{ij} \in l_{q'} : \forall u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \Rightarrow \delta(u_{ij}, u_{qw}) = 0$$

2. Fall Angenommen es gilt:

$$\exists u_{ij} \in l_{q'} : \exists u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \wedge i = r$$

Dann gilt:

$$\exists u_{qw} \in l_{q''} : m_q \in \{m_y \mid u_{yz} \in l_{q'}\}$$

Dies widerspricht der Definition von $l_{q''}$ und somit gilt:

$$\forall u_{ij} \in l_{q'} : \forall u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \Rightarrow i \neq q$$

Es wurde gezeigt, dass gilt:

$$\forall u_{ij} \in l_{q'} : \forall u_{qw} \in l_{q''} : u_{ij} \neq u_{qw} \Rightarrow (\delta(u_{ij}, u_{qw}) = 0 \wedge i \neq q)$$

Somit gilt $l_q \in L$.

Des Weiteren gilt: $B_r = l_{q'} \Rightarrow l_{q'} \cap U^* = \emptyset$ und aus $l_{q''} \subseteq U^*$ folgt $l_{q'} = l_q \backslash U^*$ und somit auch $B_r = l_q \backslash U^*$.

Somit wurde gezeigt:

$$s_r \in \{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l_q \in L : \{m_j \mid u_{ji} \in l_q\} = D \wedge B_p = l_q \backslash U^*\}$$

Es wurde gezeigt, dass für alle $s_r \in \{s \mid s \in S \wedge \beta(s) = D\}$ gilt:

$$s_r \in \{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l_q \in L : \{m_j \mid u_{ji} \in l_q\} = D \wedge B_p = l_q \backslash U^*\},$$

woraus folgt:

$$\begin{aligned}
&\#\{s \mid s \in S \wedge \beta(s) = D\} \\
&\leq \#\{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge \exists l_q \in L : \{m_j \mid u_{ji} \in l_q\} = D \wedge B_p = l_q \backslash U^*\} \\
&\leq \sum_{l \in \{g \mid g \in L \wedge \{m_j \mid u_{ji} \in g\} = D\}} \#\{s_p \mid s_p \in S \wedge \beta(s_p) = D \wedge B_p = l \backslash U^*\} \\
&\leq \sum_{l \in \{g \mid g \in L \wedge \{m_j \mid u_{ji} \in g\} = D\}} x_l \qquad \text{(laut Zuordnung von } x_l\text{)}
\end{aligned}$$

Somit wird Bedingung (2) des *ILP* erfüllt.

c) Es ist zu zeigen, dass die Bedingung (3) des *ILP* erfüllt wird, dass heißt es muss gelten: $\sum_{l \in L} x_l \cdot \#(l \cap U^*) \leq k$.

Direkter Beweis:

$$\begin{aligned}
&\sum_{l \in L} x_l \cdot \#(l \cap U^*) \\
&= \sum_{l \in L} \#\{s_p \mid s_p \in S \wedge \beta(s_p) = \{m_j \mid u_{jr} \in l\} \wedge B_p = l \backslash U^*\} \cdot \#(l \cap U^*) \\
&= \sum_{l \in L} \#\{(s_p, u_{qw}) \mid s_p \in S \wedge \beta(s_p) = \{m_j \mid u_{jr} \in l\} \wedge B_p = l \backslash U^* \wedge \\
&\qquad\qquad u_{qw} \in l \cap U^*\} \\
&= \#\{(s_p, u_{qw}) \mid s_p \in S \wedge \exists l \in L : \beta(s_p) = \{m_j \mid u_{jr} \in l\} \wedge B_p = l \backslash U^* \wedge \\
&\qquad\qquad u_{qw} \in l \cap U^*\} \\
&= \#\{(s_p, m_q) \mid (s_p, m_q) \in A \wedge \exists l \in L : \beta(s_p) = \{m_j \mid u_{jr} \in l\} \wedge B_p = l \backslash U^* \wedge \\
&\qquad\qquad \exists u_{qw} \in l \cap U^*\} \\
&= \#\{(s_p, m_q) \mid (s_p, m_q) \in A \wedge \forall u_{qr} \in U : ((s_r, m_q), u_{qr}) \notin R\} \\
&= \#A - \#R \\
&\leq k
\end{aligned}$$

Somit wird die Bedingung (3) des *ILP* erfüllt.

Da diese Belegung alle Bedingungen des *ILP* erfüllt, gilt $sol_{ILP}(x) \neq \emptyset$.

2. Es ist zu zeigen, dass gilt: $sol_{ILP}(x) \neq \emptyset \Rightarrow x \in SSP$

   Beweis:

   Sei $sol_{ILP}(x) \neq \emptyset$.

   $\Rightarrow$ es existiert eine Belegung $\varphi\colon \{x_l | l \in L\} \rightarrow \mathbb{N}$ für die Variablen des *ILP*, so dass alle Bedingungen des *ILP* erfüllt werden.

   Erstellung einer Relation R für *SSP*:

   $R = \emptyset$

   $\forall D \subseteq M : Q_D = \{s_i \mid s_i \in S \wedge \beta(s_i) = D\}$

   foreach $l \in L$ mit $\{m_j \mid u_{jb} \in l\} = D$

   $Q_{D'}$ enthält $min\{\#Q_D, x_l\}$ viele Elemente aus Q_D

   foreach gewählte $s_i \in Q_{D'}$

   $R := R \cup \{((s_i, m_r), u_{rq}) \mid m_r \in D \wedge u_{rq} \in l \cap U\}$

   $Q_D := Q_D \backslash \{s_i\}$

   endfor

   endfor

   a) Es ist zu prüfen, ob gilt $R \subseteq A \times U$:

      Laut obiger Zuordnung gilt:

$$\begin{aligned}
&\forall((s_i, m_r), u_{jq}) \in R : r = j \wedge m_r \in \beta(s_i) \wedge u_{jq} \in l \cap U \\
&\Rightarrow \forall((s_i, m_r), u_{rq}) \in R : m_r \in \beta(s_i) \wedge u_{rq} \in U \\
&\Rightarrow \forall((s_i, m_r), u_{rq}) \in R : (s_i, m_r) \in A \wedge u_{rq} \in U \\
&\Rightarrow \forall((s_i, m_r), u_{rq}) \in R : ((s_i, m_r), u_{rq}) \in A \times U
\end{aligned}$$

      Somit gilt $R \subseteq A \times U$.

   b) Es wird gezeigt, dass gilt $\#R \geq \#A - k$:

      Direkter Beweis:

      Sei $z\colon S \rightarrow L$ eine Abbildung mit $z(s_i) = \{l | l \in L \wedge s_i$ gehört zu den für l ausgewählten Studierenden$\}$.

      Es gilt $\forall s_i \in S : \exists l \in L : z(s_i) = \{l\}$ (Beweis für $\forall s_i \in S : \#z(s_i) = 1$ s.u.).

$\Rightarrow \forall s_i \in S : \forall u_{rq} \in l \backslash U^* : ((s_i, m_r), u_{rq}) \in R$ mit $l \in z(s_i)$

(laut Zuordnung von R)

$\Rightarrow \forall s_i \in S : \forall u_{rq} \in l \cap U^* : \forall u \in U : ((s_i, m_r), u) \notin R)$

$\Rightarrow \forall s_i \in S : \#\{(s_i, m_r) \mid m_r \in \beta(s_r) \wedge \forall u \in U : ((s_i, m_r), u) \notin R\}$

$= \#(l \cap U^*)\}$ mit $l \in z(s_i)$

$\Rightarrow \#\{(s_i, m_r) \mid (s_i, m_r) \in A \wedge \forall u \in U : ((s_i, m_r), u) \notin R\} = \sum_{s_i \in S} \#(l \cap U^*)$

mit $z(s_i) = \{l\}$

$\Rightarrow \#A - \#R = \sum_{s_i \in S} \#(l \cap U^*) \text{mit} z(s_i) = \{l\}$

$\Rightarrow \#A - \#R \leq \sum_{l \in L} x_l \cdot \#(l \cap U^*)$

$\Rightarrow \#A - \#R \leq k$ (laut (3) des ILP)

$\Rightarrow \#R \geq \#A - k$

Beweis für $\forall s_i \in S : \#z(s_i) = 1$:

Widerspruchsbeweis:

1. Fall : Sei $\#z(s_i) > 1$ für $s_i \in S$.

   Dann gilt $\exists l_j, l_h \in z(s_i)$ mit $l_j \neq l_h$.

   Somit wird für $D = \beta(s_i)$ die Foreach-Schleife einmal mit l_j und einmal mit l_h durchlaufen.

   oBdA.: Bei der Erstellung von R wird die Foreach-Schleife zuerst mit l_j und später mit l_h durchlaufen.

   Da $l_j \in z(s_i)$ gehört s_i zu den für l_j ausgewählten Studierenden und ist somit in $Q_{D'}$ enthalten.

   Im Foreach-Schleifendurchlauf von l_j wird folglich die innere Foreach-Schleife mit s_i durchlaufen.

   In dem Schleifendurchlauf mit l_j wird somit s_i aus Q_D entfernt.

   Demnach gilt anschließend für den Schleifendurchlauf mit l_h, dass $s_i \notin Q_D$.

   Somit gehört s_i nicht zu den für l_h ausgewählten Studierenden.

   Dies führt zu einem Widerspruch.

   Es wurde hiermit gezeigt, dass gilt $\forall s_i \in S : \#z(s_i) \leq 1$.

2. Fall : Angenommen es existiert ein $s_i \in S$ mit $\#z(s_i) = 0$.

$\Rightarrow s_i$ wird nie die innere Schleife bei der Erstellung von R durchlaufen.

$$\Rightarrow \#Q_D > \sum_{l \in \{g | g \in L \wedge \{m_j | u_{jb} \in g\} = D\}} x_l \text{ mit } D = \beta(s_i)$$

$$\Rightarrow \#\{s_i \mid \beta(s_i) = D\} > \sum_{l \in \{g | g \in L \wedge \{m_j | u_{jb} \in g\} = D\}} x_l$$

Dies widerspricht der 2. Bedingung des *ILP* und somit gilt

$$\forall s_i \in S : \#z(s_i) > 0$$

Es wurde gezeigt, dass für alle $s_i \in S$ gilt $\#z(s_i) = 1$.

3. Es ist zu zeigen, dass R alle Bedingungen des *SSP* erfüllt.

   Beweis:

   a) 1. Bedingung des *SSP*:

   $\forall((s_i, m_j), u_{rq}) \in R : r = j$ (u_{rq} ist Übungsgruppe von m_j)

   Laut der Zuordnung von R wird diese Bedingung erfüllt.

   b) 2. Bedingung des *SSP*: $\#\{s_i \mid s_i \in S \wedge ((s_i, m_j), u_{jd}) \in R\} \leq \tau(u_{jd})$

   Direkter Beweis:

$$\forall u_{jd} \in U : \#\{s_i \mid s_i \in S \wedge ((s_i, m_j), u_{jd}) \in R\} \tag{4.17}$$

$$= \sum_{l \in \{f | f \in L \wedge u_{jd} \in f\}} \#\{s_i | s_i \in S \wedge z(s_i) = \{l\}\} \tag{4.18}$$

$$\leq \sum_{l \in \{f | f \in L \wedge u_{jd} \in f\}} x_l \tag{4.19}$$

$$\leq c_{jd} \text{ (laut (1) des } ILP) \tag{4.20}$$

$$= \tau(u_{jd}) \tag{4.21}$$

   Erläuterung:
   Einem Studierenden s_i werden genau die Übungen zugeordnet, die in $l = z(s_i)$ enthalten sind. Somit gilt für alle Studierenden s_i, die an der Übung u_{jd} teilnehmen, dass u_{jd} ein Element aus $z(s_i)$ ist. Demnach entspricht die Studierendenmenge aus (4.17) der Menge aus Zeile (4.18). Laut der erstellten Relation R werden maximal x_l viele Studierende l zugeordnet. Das heißt, zum einem Element $l \in L$ existieren maximal x_l viele Studierende s mit $z(s) = l$. Somit ist die Kardinalität der Menge aus Zeile (4.18.) maximal so groß wie der Wert aus Zeile (4.19). In Folge der Bedingung (1) des *ILP* ergibt sich die Zeile (4.20). Basierend auf der Definition des *ILP* gilt für alle $u_{ij} \in U$, dass $c_{ij} = \tau(u_{ij})$ und daher folgt aus Zeile (4.19) die

Zeile (4.20).
Infolgedessen gilt, dass R die 2. Bedingunge des *SSP* erfüllt.

c) 3. Bedingung des *SSP*:

$\forall s_i \in S : \forall((s_i, m_j), u_{jb}), ((s_i, m_{j'}), u_{j'b'}) \in R : m_j \neq m_{j'} \Rightarrow \delta(u_{jb}, u_{j'b'}) = 0$

Widerspruchsbeweis: Angenommen Bedingung 3 wird nicht erfüllt.

$\Rightarrow \exists s_i \in S : \exists((s_i, m_j), u_{jb}), ((s_i, m_{j'}), u_{j'b'}) \in R : m_j \neq m_{j'} \wedge \delta(u_{jb}, u_{j'b'}) = 1$

Sei $D_i \subseteq M$ mit $\beta(s_i) = D_i$, dann gilt $s_i \in Q_{D_i}$.

Sei $z(s_i) = \{l \mid l \in L \wedge s_i$ gehört zu für l ausgewählten Studierenden$\}$.

Da $((s_i, m_j), u_{jb}), ((s_i, m_{j'}), u_{j'b'}) \in R$ gilt

$$\exists l, l' \in z(s_i) : u_{jb} \in l \wedge u_{j'b'} \in l'$$

1. Fall : $l = l'$

$$\Rightarrow \exists l \in L : \{l \in z(s_i) \wedge u_{jb} \in l \wedge u_{j'b'} \in l\}$$

Auf Grund von $m_j \neq m_{j'}$ gilt $u_{jb} \neq u_{j'b'}$.

Laut Definition von L gilt:

$$\forall l \in L : \forall u_{ir}, u_{js} \in l : u_{ir} \neq u_{jd} \Rightarrow \delta(u_{ir}, u_{js}) = 0$$

Dies führt zu einem Widerspruch.

2. Fall : $l \neq l'$

$$\Rightarrow \exists l, l' \in z(s_i) : l \neq l' \wedge u_{jb} \in l \wedge u_{j'b'} \in l'$$
$$\Rightarrow \#z(s_i) > 1$$

Dies führt zu einem Widerspruch, da bereits bewiesen wurde, dass für alle $s_i \in S$ gilt $\#z(s_i) = 1$.

Folglich erfüllt R die dritte Bedingung des *SSP*.

d) 4. Bedingung des *SSP*: $\forall((s_i, m_j), u_{jb}), ((s_i, m_j), u_{pq}) \in R : u_{jb} = u_{pq}$

Widerspruchsbeweis: Angenommen Bedingung 4 wird verletzt.

$$\Rightarrow \exists((s_i, m_j), u_{jb}), ((s_i, m_j), u_{pq}) \in R : u_{jb} \neq u_{pq}$$
$$\Rightarrow \exists((s_i, m_j), u_{jb}), ((s_i, m_j), u_{jq}) \in R : u_{jb} \neq u_{jq}$$
(laut Bedingung (1) des *SSP*)

Da $((s_i, m_j), u_{jb}), ((s_i, m_j), u_{jq}) \in R$ existiert ein $l \in L$, so dass gilt:

$$z(s_i) = \{l\} \wedge u_{jb} \in l \wedge u_{jq} \in l$$

Folglich gilt ebenfalls:

$$\exists l \in L : \exists u_{jb}, u_{pq} \in L : u_{jb} \neq u_{pq} \wedge j = p$$

Dies führt zu einem Widerspruch bezüglich der Definition von L und demnach gilt, dass R die Bedingung 4 des *SSP* erfüllt.

Aus dem Beweis, dass $R \subseteq A \times U$ mit $\#R \geq \#A - k$ alle Bedingungen des *SSP* erfüllt, folgt: $x \in SSP$ vermöge R.

Es wurde gezeigt, dass das *SSP* mit Hilfe des angegebenen *ILP* gelöst werden kann. Die Menge der Variablen ist eine Teilmenge der Potenzmenge von $U \cup U^*$ mit $\#(U \cup U^*) = p + \#M \leq 2 \cdot p$. Die Anzahl der Variablen ist somit maximal so groß wie die Kardinalität dieser Potenzmenge und daher ist die Anzahl der Variablen höchsten $2^{2 \cdot p}$. Dies bedeutet, dass für ein festes p die Anzahl der Variablen des *ILP* konstant ist. Auf Grund dieser Eigenschaft kann der Satz 4.5 angewendet werden und daher gilt, dass das *ILP* für einen festen Parameter p in linearer Zeit gelöst werden kann. Dies bedeutet, dass laut Lenstra ein Algorithmus existiert, der für ein festes p eine polynomielle Laufzeit hat und daher ein *FPT-Algorithmus* für das parametrisierte *SSP* mit der Parametrisierung $\kappa(x) = p$ ist. Nachdem *Lenstra* im Jahr 1983 einen Algorithmus veröffentlichte, der das *ILPF* für eine konstante Anzahl von Parametern in linearer Zeit löst [Len83], entwarf *Ravi Kannan* aufbauend auf diesem Resultat einen effizienteren Algorithmus für *ILPF*, den er im Jahr 1987 publizierte [Kan87]. Der folgende Satz basiert auf dem Ergebnis von *Kannan* [Nie02].

Satz 4.6. *Integer Linear Programming Feasibility (ILPF) kann in $O(p^{9 \cdot p/2} \cdot L)$ arithmetischen Operationen von Integern, deren Größe sich in $O(p^{2 \cdot p} \cdot L)$ Bits befinden, gelöst werden, wobei p die Anzahl der ILP Variablen und L die Anzahl der Bits in der Eingabe sind.*

Aufbauend auf diesem Satz kann die Laufzeit des *FPT-Algorithmus* für das parametrisierte *SSP* mit dem Parameter p analysiert werden. Wie schon erwähnt, ist die maximale Anzahl der *ILP* Variablen gleich $2^{2 \cdot p}$ und somit gilt für die Laufzeit des Algorithmus, dass sie in $O(2^{p \cdot 9 \cdot 2^{2 \cdot p}} \cdot |x|)$ liegt. Folglich ist der exponentielle Teil der Laufzeit ausschließlich von p abhängig und daher gilt, dass dies ein *FPT-Algorithmus* für *SSP* ist. Die Laufzeit des *FPT-Algorithmus* aus Abschnitt 4.3.2, der mittels erschöpfender Suche das *SSP* entscheidet, ist nicht nur exponentiell bezüglich p, sondern auch k und c sind im exponentiellen Teil enthalten. Daher ist der *FPT-Algorithmus* von *Lenstra* dahingegen besser, dass weniger Variablen im exponentiellen Teil der Laufzeit enthalten sind und somit weniger Eingabedaten das exponentielle Wachstum beeinflussen. Dies bedeutet nicht, dass der *FPT-Algorithmus* von *Lenstra* generell ein besserer Algorithmus ist als die erschöpfende Suche, denn wenn man die Laufzeit dieser beiden Algorithmen betrachtet, hat der

Algorithmus von *Lenstra* eine Laufzeit in $O(2^{p \cdot 9 \cdot 2^{2 \cdot p}} \cdot |x|)$ und die Laufzeit des anderen Algorithmus befindet sich in $O(2^{p \cdot (p \cdot c + k)} \cdot (p \cdot c + k)^2 \cdot p^2)$. Somit gilt nicht, dass die Laufzeit eines Algorithmus besser ist, je weniger Variablen sie im exponentiellen Teil enthält. Dies sagt nichts über die Effizienz des Algorithmus aus, sondern nur welche Faktoren das exponentielle Wachstum beeinflussen. Zu dem parametrisierten *SSP* mit dem Parameter p existiert somit ein *FPT-Algorithmus* $\mathcal{A}$, dessen Laufzeit in $O(f(p) \cdot b(n))$ mit einem Polynom b und der Eingabelänge n ist. Da $f(k + p) \geq f(p)$ gilt, dass der Algorithmus $\mathcal{A}$ ebenfalls ein *FPT* Algorithmus für das parametrisierte *SSP* mit den Parametern k und p ist. Hieraus folgt, dass sich alle parametrisierten *SSP*, deren Parametrisierung p enthält, in *FPT* befinden. Allgemein gilt:

Satz 4.7. *Seien (Q, κ) und (Q, ν) parametrisierte Probleme mit $\kappa(x) = \sum_{p \in P} p$, $\nu(x) = \sum_{p \in P'} p$ und seien P und P' eine Menge von Eingabedaten von Q, die als Wert eine natürliche Zahl haben, mit $P' \subseteq P$. Des Weiteren gilt, dass Q NP-vollständig ist. Falls $(Q, \nu) \in FPT$, dann gilt $(Q, \kappa) \in FPT$.*

Beweis. Laut Beweisvoraussetzung gilt: $(Q, \nu) \in FPT$.

$\Rightarrow$ es existiert ein Algorithmus $\mathcal{A}$, der Q in $O(f(\nu(x)) \cdot b(n))$ entscheidet.

$\Rightarrow$ es existiert ein Algorithmus $\mathcal{A}$, der Q in $O(f(\sum_{p \in P'} p) \cdot b(n))$ entscheidet.

Da Q *NP-vollständig* ist, liefert $f(\sum_{p \in P'} p)$ eine exponentielle Zahl bezüglich $\sum_{p \in P'} p$. Aus $P' \subseteq P$ folgt somit $f(\sum_{p \in P'} p) \leq f(\sum_{p \in P} p)$.

$\Rightarrow$ es existiert ein Algorithmus $\mathcal{A}$, der Q in $O(f(\sum_{p \in P} p) \cdot b(n))$ entscheidet.

$\Rightarrow$ es existiert ein Algorithmus $\mathcal{A}$, der Q in $O(f(\kappa(x)) \cdot b(n))$ entscheidet.

$\Rightarrow (Q, \kappa) \in FPT$.

4.7 Parameter u

Im vorherigen Abschnitt wurde gezeigt, dass sich alle parametrisierten *SSP*, deren Parametrisierung p enthält, in *FPT* befinden. Des Weiteren gilt, dass die parametrisierten *SSP*, deren Parameter alle in der Menge $\{k, c, u, t\}$ enthalten sind, *para-NP-vollständig* sind. Der Parameter p beinhaltet die insgesamte Anzahl der Übungsgruppe über alle Module. Wenn gezeigt werden kann, dass sich das parametrisierte *SSP* mit dem Parameter u, der die maximale Anzahl von Übungsgruppen eines Moduls angibt, ebenfalls in *FPT* befindet, wäre dieses Resultat besser, denn u hat gewöhnlich einen deutlich kleineren Wert als p. Zur Untersuchung des parametrisierten Problems (SSP, κ) mit der Parametrisierung $\kappa(x) = u$ wird eine *Polynomialzeit-Many-one-Reduktion* vom *3-Dimensionalen Matching (3-DM)* auf *SSP* angegeben, wobei u einen festen Wert erhält.

Lemma 4.8. *3-DM* $\leq_m^p$ *SSP.*

Beweis. Das *3-DM* ist ein *NP-vollständiges* Problem, das wie folgt definiert ist:

Definition 10. *3-Dimensionales Matching (3-DM):*
Instanz: T, B, G, H, m *mit* $m \in \mathbb{N}$, $T \subseteq B \times G \times H$ *und* $\#B = \#G = \#H = m$
Frage: Existiert ein $T' \subseteq T$ *mit* $\#T' = m$*, so dass alle Elemente aus* T' *paarweise disjunkt sind?*

Für die *Polynomialzeit-Many-one-Reduktion* von *3-DM* auf *SSP* ist zu zeigen, dass eine in Polynomialzeit berechenbare Reduktionsfunktion f existiert, so dass für alle x gilt $x \in$ *3-DM* $\Leftrightarrow f(x) \in$ *SSP*.
Die Reduktionsfunktion f ist wie folgt definiert:

$f(T, B, G, H, m) =_{def} (M, S, U, \tau, \beta, \delta, k)$ mit

$M = \{m_1, ..., m_{\#T}\}$

$S = \{s_1\}$

$U = \bigcup_{m_i \in M} \{u_{i1}\}$

$\tau(u_{ij}) = 1$ für $u_{ij} \in U$

$\beta(s_1) = M$

$$\delta(u_{ij}, u_{kl}) = \begin{cases} 1, & \text{falls } t_i \cap t_k \neq \emptyset \\ 0, & \text{sonst} \end{cases}$$

$k = \#T - m$

Die aus x resultierende Instanz $f(x)$ enthält für jedes Element aus T ein Modul, welches aus genau einer Übungsgruppe besteht und Kapazität 1 hat. So existiert zu jeder Übungsgruppe ein Element in T. Des Weiteren gibt es einen Studierenden, der an allen Modulen teilnehmen möchte. Die Elemente aus B, G, H stehen jeweils für einen Termin und daher findet eine Übungsgruppe genau an den drei Terminen statt, die das zugehörige Tripel enthält. Da es nur einen Studierenden gibt und jede Übungsgruppe die Kapazität 1 hat, können die Kapazitäten der Übungsgruppen nicht überschritten werden. Falls einem Studierenden zwei Übungsgruppen zugewiesen werden, deren zugehörige Tripel nicht disjunkt sind, dann haben die beiden Übungsgruppen mindestens einen Termin gemeinsam und somit entsteht ein Terminkonflikt. Für die Menge der Übungsgruppen, die der Studierende zugewiesen bekommt, muss somit gelten, dass die zugehörigen Tripel paarweise disjunkt sind. Die Frage des *3-DM* ist, ob es eine Teilmenge von T der Kardinalität m gibt mit paarweise disjunkten Tripel. Falls diese existiert, können somit einem Studierenden m viele Elemente zugewiesen werden und das heißt für $k = \#T - m$ wäre die Frage des *SSP* ebenfalls erfüllt.

Korrektheitsbeweis:
Die Reduktionsfunktion f gibt eine *Polynomialzeit-Many-one-Reduktion* von *3-DM* auf *SSP* an, falls f in Polynomialzeit berechenbar ist und falls gilt

$$x \in \textit{3-DM} \Leftrightarrow f(x) \in \textit{SSP}$$

Die angegebene Reduktionsfunktion ist offensichtlich in polynomieller Zeit berechenbar und daher ist noch zu zeigen, dass gilt $x \in \textit{3-DM} \Leftrightarrow f(x) \in \textit{SSP}$:

1. Beweis für $x \in \textit{3-DM} \Rightarrow f(x) \in \textit{SSP}$:

   Angenommen $x \in \textit{3-DM}$.

   $\Rightarrow$ es existiert ein $T' \subseteq T$ mit $\#T' = m$, so dass alle Elemente aus T' paarweise disjunkt sind.

   Sei $R = \bigcup_{t_i \in T'} \{((s_1, m_i), u_{i1}\}$.

   Laut Definition von A und $f(x)$ gilt, $A = \bigcup_{m \in M} \{s_1, m\}$ und somit gilt $\#A = \#M = \#T$.

   Auf Grund der obigen Zuordnung von R gilt somit $\#R = \#T' = m = \#T - (\#T - m) = \#A - k$.

   Für jedes $t_i \in T'$ gilt $t_i \in T$ da $T' \subseteq T$ und somit gilt für alle $t_i \in T' : m_i \in M$.

   Demnach gilt für alle $t_i \in T'$: $(s_1, m_i) \in A$ und da $u_{i1} \in U$, folgt $R \subseteq A \times U$.

   Es gilt $R \subseteq A \times U$ und $\#R = \#A - k$ folglich ist R eine zulässige Relation für *SSP*, falls sie alle vier Bedingungen des *SSP* erfüllt. Dies wird nachfolgend untersucht.

   a) 1. Bedingung: $\forall((s_i, m_j), u_{lf}) \in R : j = l$

      Diese Bedingung wird laut obiger Zuordnung von R erfüllt.

   b) 2. Bedingung:

      $\#S' \leq \tau(u_{ld})$ mit $S' = \{s_i \mid s_i \in S \wedge ((s_i, m_l), u_{ld}) \in R\}$ für alle $u_{ld} \in U$

      Beweis: Aus $S' \subseteq S \wedge \#S = 1$ folgt $\#S' \leq 1$. Auf Grund der Reduktionfunktion gilt für alle $u_{ld} \in U$: $\tau(u_{ld}) = 1$, woraus folgt, dass für alle $u_{ld} \in U$ gilt: $\#S' \leq 1 = \tau(u_{ld})$ und somit wird diese Bedingung erfüllt.

   c) 3. Bedingung:

      $$\forall((s_i, m_j), u_{jd}), ((s_i, m_f), u_{fg}) \in R : m_j \neq m_f \Rightarrow \delta(u_{jd}, u_{fg}) = 0$$

      Widerspruchsbeweis:

      Angenommen es gilt:

$$\exists((s_i, m_j), u_{jd}), ((s_i, m_f), u_{fg}) \in R : m_j \neq m_f \wedge \delta(u_{jd}, u_{fg}) = 1$$
$$\Rightarrow \exists((s_1, m_j), u_{j1}), ((s_1, m_f), u_{f1}) \in R : m_j \neq m_f \wedge \delta(u_{j1}, u_{f1}) = 1$$
$$\Rightarrow \exists t_j, t_f \in T' : t_j \cap t_f \neq \emptyset$$

Hieraus ergibt sich ein Widerspruch, da laut Definition 10 des *3-DM* gilt, dass alle Elemente aus T' paarweise disjunkt sind. R erfüllt somit die dritte Bedingung.

d) 4. Bedingung: $\forall((s_i, m_j), u_{jd}), ((s_i, m_j), u_{jg}) \in R : u_{jd} = u_{jg}$

Direkter Beweis:

$$\forall((s_i, m_j), u_{jd}), ((s_i, m_j), u_{jg}) \in R : u_{jd} \in U \wedge u_{jg} \in U$$
$$\Rightarrow \forall((s_i, m_j), u_{jd}), ((s_i, m_j), u_{jg}) \in R : u_{jd} = u_{j1} \wedge u_{jg} = u_{j1}$$
(laut Reduktionsfunktion)
$$\Rightarrow \forall((s_i, m_j), u_{jd}), ((s_i, m_j), u_{jg}) \in R : u_{jd} = u_{jg}$$

Somit wird die vierte Bedingung des *SSP* erfüllt.

Da $R \subseteq A \times U$ mit $\#R \geq \#A - k$ alle Bedingungen des *SSP* erfüllt, gilt somit $f(x) \in SSP$ vermöge R.

2. Beweis für $f(x) \in SSP \Rightarrow x \in$ *3-DM*:

Sei $f(x) \in SSP$.

$\Rightarrow$ es existiert eine Relation $R \subseteq A \times U$ mit $\#R \geq \#A - k$, die alle Bedingungen des *SSP* erfüllt.

Sei $T' = \{t_i \mid t_i \in T \wedge ((s_1, m_i), u_{i1}) \in R\}$.

Hieraus folgt $T' \subseteq T$ und $\#T' \leq \#R$.

Falls $T' < \#R$, dann existiert ein $((s_1, m_i), u_{i1}) \in R$ mit $t_i \notin T$. Laut Reduktionfunktion existiert zu jedem $m_i \in M$ ein $t_i \in T$ und somit würde dies zu einem Widerspruch führen. Hieraus folgt:

$$\#T' = \#R \geq \#A - k = \#M - (\#T - m) = \#T - (\#T - m) = m$$

Angenommen T' ist nicht paarweise disjunkt

$$\Rightarrow \exists t_i, t_j \in T' : t_i \neq t_j \wedge t_i \cap t_j \neq \emptyset$$
$$\Rightarrow \exists((s_1, m_i), u_{i1}), ((s_1, m_j), u_{j1}) \in R : \delta(u_{i1}, u_{j1}) = 1$$

Da $t_i \neq t_j$ gilt $m_i \neq m_j$ und somit verstößt R gegen die dritte Bedingung des *SSP*. Dies führt zu einem Widerspruch, da R zulässig ist und somit gilt, dass alle Elemente aus T' paarweise disjunkt sind.

Demnach gilt $x \in$ *3-DM* vermöge T'.

Es wurde gezeigt, dass gilt $x \in$ *3-DM* $\Leftrightarrow f(x) \in$ *SSP*. Somit existiert eine *Polynomialzeit-Many-one-Reduktion* des *NP-vollständigen* Problems *3-DM* auf *SSP* vermöge f.
Das Problem *3-DM* ist für $m = \#T$ effizient lösbar, da in diesem Fall nur zu untersuchen ist, ob alle Elemente aus T paarweise disjunkt sind. Aus dieser Reduktion folgt somit, dass *SSP NP-vollständig* ist für $k > 0$.
Die *NP-Vollständigkeit* des *SSP* ist kein neues Resultat, da dies schon in Kapitel Komplexität 3 mittels der *Polynomialzeit-Many-one-Reduktion* von *3-SAT* auf *SSP* gezeigt wurde. Im Gegensatz zu der ersten Reduktion, zeigt diese Reduktion, dass *SSP NP-vollständig* ist, sogar für einen festen Wert des Parameters u. Unabhängig von der Instanz des *3-DM* hat der Parameter u immer den Wert 1. Hieraus folgt, dass wenn es einen *FPT-Algorithmus* für *SSP* mit dem Parameter u gäbe, dann wäre dessen Laufzeit in $O(f(u) \cdot b(n))$ mit einem Polynom b und somit in $O(f(1) \cdot b(n))$. Da $f(1)$ einen konstanten Wert liefert, wäre der Algorithmus in polynomieller Zeit berechenbar. Dies ist ein Widerspruch zum Resultat, dass *SSP* für $k > 0$ und $u = 1$ *NP-vollständig* ist und daher gilt, dass das parametrisierte *SSP* mit dem Parameter u für $k > 0$ nicht in der Komplexitätsklasse *FPT* liegt.

Korollar 6. *Das SSP mit dem Parameter u ist para-NP-vollständig.*

Beweis. Die Vereinigung aller *SSP-Instanzen* mit $u = 1$ ist somit für $k > 0$ *NP-vollständig* und daher gilt, laut Satz 4.3, dass (Q, κ) mit $\kappa(x) = u$ *para-NP-vollständig* ist für $k > 0$.

4.8 Parameter c, u, t, s

Korollar 7. *Das SSP mit den Parametern c, u, t und s ist para-NP-vollständig.*

Beweis. Im Abschnitt 4.7 wurde anhand einer *Polynomialzeit-Many-one-Reduktion* von *3-DM* auf *SSP* die *para-NP-Vollständigkeit* von (Q, κ) mit $\kappa(x) = u$ gezeigt. Die hierzu angegebene Reduktionsfunktion f weist nicht nur u einen festen Wert zu, sondern auch c, t, und s. Unabhängig von der konkreten *3-DM-Instanz* ist die Kapazität jeder Übungsgruppe immer eins und somit hat c einen festen Wert. Des Weiteren findet jede Übungsgruppe der aus der Reduktion enstandenen *SSP-Instanz* an drei Terminen statt, das heißt es gilt immer $t = 3$. Ebenso gilt, dass f immer genau einen Studierenden erzeugt ($s = 1$). Somit sind die Instanzen mit $c + u + t + s = 6$ *NP-vollständig*. Es gilt daher die *para-NP-Vollständigkeit* für (Q, κ) mit $\kappa(x) = c + u + t + s$. Aus Korollar 5 folgt, dass alle parametrisierten Probleme des *SSP*, deren Parameter in der Menge $\{c, u, t, s\}$ enthalten sind, *para-NP-vollständig* sind. Obwohl die Anzahl der Studierenden (s) normalerweise der größte Wert der Eingabe enthält, wurde gezeigt, dass sich der exponentielle Teil der Laufzeit nicht darauf beschränken lässt.

4.9 Abschließende Betrachtung des Theorems 1

Einleitend wurde das Theorem 1 angegeben, dessen Korrektheit in dieser Arbeit zu zeigen war. Dieses Theorem enthält unter anderem die Behauptung, dass sich die parametrisierten Probleme des *SSP*, deren Parameterisierungen die Gesamtzahl der Übungen (p) enthalten, in der Komplexitätsklasse *FPT* befinden. Diese Behauptung wurde sowohl anhand einer *Kernelization* 4.3.1 als auch mit Hilfe eines *FPT-Algorithmus* 4.3.2 nachgewiesen.
Des Weiteren enthält das Theorem die Aussage, dass alle parametrisierten Probleme des *SSP*, deren Parameter eine Teilmenge von $\{k, c, u, t\}$ bilden, *para-NP vollständig* sind. Im Kapitel 3 wurde anhand der Reduktion *3-SAT* $\leq_m^p$ *SSP* nachgewiesen, dass sich alle parametrisierten Probleme des *SSP* in der Klasse *para-NP* befinden. Aus dieser Reduktion folgt ebenfalls, dass das *SSP* sogar für feste Werte der Parameter k, c, u und t *para-NP vollständig* ist. In Folge des Satzes 4.3 gilt somit die *para-NP Vollständigkeit* für alle parametrisierten Probleme des *SSP*, deren Parameter in der Menge $\{k, c, u, t\}$ enthalten sind.
Das Kapitel 4.7 enthält die Reduktion *3-DM* $\leq_m^p$ *SSP*, welche für $k > 0$ die *NP-Vollständigkeit* des *SSP* für feste Werte der Parameter c, u, t und s beweist. Somit folgt aus diesem Resultat in Verbindung mit dem Satz 4.3 die Korrektheit der Behauptung der *para-NP Vollständigkeit* aller parametrisierten *SSP*, deren Parameter in $\{c, u, t, s\}$ enthalten sind, für $k > 0$. Zusammenfassend lässt sich festhalten, dass das Theorem 1 erfüllt wird.

5

Zusammenfassung und Ausblick

Die Hochschulen haben unter anderem das Problem, dass eine sehr hohe Anzahl von Studierenden auf Übungsgruppen der gewählten Module konfliktfrei zu verteilen sind. Dieses Problem wird durch zahlreiche Varianten des *Student Sectioning Problems (SSP)* repräsentiert. Für die Analyse in dieser Arbeit wurde eine möglichst praxisnahe Variante des *SSP* in Kapitel 2 definiert.
Der Hauptaspekt dieser Arbeit ist die Analyse des *SSP* bezüglich seiner parametrisierten Komplexität. Da sich aus den Resultaten der klassischen, nicht-parametrisierten, Komplexität des *SSP* Erkenntnisse bezüglich der parametrisierten Komplexität des *SSP* ziehen lassen, wurde in Kapitel 3 zunächst die klassische Komplexität des *SSP* untersucht. Diese Analyse ergab, dass es sich hierbei um ein *NP-vollständiges* Problem handelt. Ein zugehöriger Beweis mit Hilfe einer *Polynomialzeit-Many-one-Reduktion* des *NP-vollständigen 3-SAT* auf *SSP* wurde in Kapitel 3 präsentiert.
Die Resultate der Analyse der parametrisierten Komplexität des *SSP* sind in der nachfolgenden Abbildung 5.1 dargestellt.

<table>
<tr><th></th><th>s</th><th>t</th><th>u</th><th>c</th><th>k</th><th>p</th></tr>
<tr><td rowspan="3">FPT</td><td></td><td></td><td></td><td></td><td></td><td>ILP</td></tr>
<tr><td></td><td></td><td></td><td colspan="3">FPT-Algorithmus (erschöpfende Suche) + polynomielle Kernelization</td></tr>
<tr><td colspan="6">ILP</td></tr>
<tr><td rowspan="3">para-NP-vollständig</td><td>3-DM $\leq_m^p$ SSP (für k > 0)</td><td>3-SAT $\leq_m^p$ SSP</td><td>3-SAT $\leq_m^p$ SSP</td><td>3-SAT $\leq_m^p$ SSP</td><td>3-SAT $\leq_m^p$ SSP</td><td></td></tr>
<tr><td></td><td colspan="4">3-SAT $\leq_m^p$ SSP</td><td></td></tr>
<tr><td colspan="4">3-DM $\leq_m^p$ SSP (für k > 0)</td><td></td><td></td></tr>
<tr><td rowspan="2">para-NP</td><td>SSP ∈ NP</td><td>SSP ∈ NP</td><td>SSP ∈ NP</td><td>SSP ∈ NP</td><td>SSP ∈ NP</td><td>SSP ∈ NP</td></tr>
<tr><td colspan="6">SSP ∈ NP</td></tr>
</table>

Abbildung 5.1. Übersicht der Resultate zur parametrisierten Komplexität der untersuchten parametrisierten Probleme des *SSP*

Es wurden verschiedene parametrisierte Probleme des *SSP* untersucht, die sich alle

in der Klasse *para-NP* und manche sogar in *FPT* befinden. Von einigen anderen Parametrisierungen konnte die *para-NP Vollständigkeit* der zugehörigen parametrisierten *SSP* bewiesen werden. Da sich somit alle Resultate auf *FPT*, *para-NP* oder die *para-NP Vollständigkeit* beziehen, existieren links in der Tabelle drei zugehörige Zeilen. Es wurden verschiedene Parameter in der Analyse betrachtet, daher existiert für jeden untersuchten Parameter eine Spalte. Ein Tabelleneintrag „*ILP*“ der Spalte „p“ und Zeile „*FPT*“ bedeutet, dass sich das parametrisierte Problem (SSP, p) in *FPT* befindet und dies anhand eines *ILP* bewiesen wurde. Existieren zum Resultat Bedingungen, so wird dies in Klammern angegeben, wie zum Beispiel in Spalte „s“ mit Zeile „*para-NP vollständig*“. Erstreckt sich ein Tabelleneintrag der Zeile „*para-NP vollständig*“ oder „*para-NP*“ über mehrere Spalten, so bedeutet dies, dass dieses Resultat für alle parametrisierten *SSP* gilt, deren Parametrisierung aus Parametern der jeweiligen Spalte bestehen. Im Gegensatz dazu bedeutet ein Tabelleneintrag in der Zeile „*FPT*“ einer Spalte „x“, dass dieses Resultat für alle parametrisierten Probleme gilt, deren Parametrisierung unter anderem den Parameter „x“ enthält. Somit sind in diese Tabelle genau die Resultate, die das Theorem 1 beinhaltet, abgebildet.

Neben den hier untersuchten Parametern, gibt es noch weitere mögliche Parametrisierungen für das *SSP*. Beispielsweise könnte als Parameter die Anzahl der Module (m) betrachtet werden. Ein *FPT-Algorithmus* diesbezüglich wäre ein besseres Resultat als $(SSP, p) \in FPT$, da die Anzahl der Module gewöhnlich deutlich kleiner ist als die Anzahl der insgesamten Übungen. In Folge der Ungleichung $m \cdot u \geq p$ gilt $(SSP, \kappa) \in FPT$ mit $\kappa(x) = m \cdot u$. Da hierbei der selbe Algorithmus verwendet werden kann und $m \cdot u \geq p$, ist dieses Resulatet nicht besser als $(SSP, p) \in FPT$. Daher lohnt es sich nicht das parametrisierte *SSP* mit den Parametern m und u zu untersuchen.
Des Weiteren könnte die maximale Anzahl der Module, die ein Studierender wählt, als Parameter betrachtet werden, da dieser Wert in der Regel klein ist und außerdem das *SSP* schwieriger sein sollte, je mehr Module ein Studierender wählt. Dies bedeutet, dass vermutlich die maximale Anzahl von gewählten Modulen die Schwierigkeit des *SSP* beeinflusst.
Nicht nur andere Parameter können für eine weitere Untersuchung betrachtet werden, sondern auch neue Kombinationen, wie zum Beispiel eine Parametrisierung $\kappa(x) = s + k$.
Des Weiteren wurde gezeigt, dass $(Q, s) \in$ *para-NP* und (Q, s) für $k > 0$ *para-NP-vollständig* ist. Aus diesen beiden Resultaten ergibt sich die Frage, ob (Q, s) ebenfalls für $k = 0$ *para-NP-Vollständig* ist oder ob sich dieses parametrisierte *SSP* in einer anderen Komplexitätsklasse, wie zum Beispiel in *FPT* oder in einer Klassen der W-Hierarchien befindet. Die Analyse des *SSP* bezüglich seiner parametrisierten Komplexität ist somit nicht abgeschlossen, denn es existieren noch parametrisierte *SSP*, deren Komplexitätsfrage in dieser Arbeit nicht beantwortet wurde.

Literatur

BGvS09. Berthold, Heinz, Gabriel Gösta und Thimo von Stuckrad: *CHE Zwei Jahre Hochschulpakt 2020 (1.Phase) - eine Halbzeitbilanz.* Vorlesungsfolien, 2009.

FG06. Flum, J. und M. Grohe: *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Kan87. Kannan, Ravi: *Minkowski's convex body theorem and integer programming.* Math. Oper. Res., 12(3):415–440, August 1987.

Len83. Lenstra, H.W.jun.: *Integer programming with a fixed number of variables.* Math. Oper. Res., 8:538–548, 1983.

MSTV12. Meier, Arne, Johannes Schmidt, Michael Thomas und Heribert Vollmer: *On the parameterized complexity of default logic and autoepistemic logic.* In: *Proceedings of the 6th international conference on Language and Automata Theory and Applications*, LATA'12, Seiten 389–400, Berlin, Heidelberg, 2012. Springer-Verlag.

Nie02. Niedermeier, Rolf: *Invitation to Fixed-Parameter Algorithms*, 2002.

Ros11. Rossmanith, Peter: *Parameterized Algorithms.* Vorlesungsfolien, RWTH Aachen, 2011.

Sch08. Schmitz, Heinz: *Algorithmik.* Vorlesungsfolien, Hochschule Trier, 2008.

Sch11. Schmitz, Heinz: *Theoretische Informatik.* Vorlesungsfolien, Hochschule Trier, 2011.