Technische Universität Berlin
Fakultät IV: Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik
Algorithmik und Komplexitätstheorie (AKT)

Algorithmen zum Finden Spezieller Stabiler Mengen in Intervallgraphen

mit Anwendungen beim Corporate Car Sharing

Bachelorarbeit

von Phillip Fiala

zur Erlangung des Grades „Bachelor of Science“ (B. Sc.)
im Studiengang Informatik

Erstgutachter:	Prof. Dr. Rolf Niedermeier
Zweitgutachter:	Prof. Dr. Thorsten Koch (Fakultät II)
Betreuer:	M. Bentert, Dr. R. Bredereck, Prof. Dr. Rolf Niedermeier

06.2019

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den 14.06.2019 Phillip Fiala.

Zusammenfassung

Zahlreiche Planungsprobleme, wie die Zuweisung von Fahrzeugen im Corporate Car Sharing, lassen sich als NP-schweres Listenfärbungs-Problem modellieren. Dabei ist ein ungerichteter Graph, eine Farbmenge und für jeden Knoten eine Liste von zulässigen Farben der Farbmenge gegeben. Gesucht ist eine Färbung aller Knoten, sodass benachbarte Knoten unterschiedlich gefärbt sind und jeder Knoten mit einer Farbe aus seiner zulässigen Farbliste gefärbt wird. Dieses Problem lässt sich auf das Problem der Inkrementellen Listenfärbung erweitern, indem zusätzlich eine zulässige Färbung für alle Knoten bis auf einen gegeben ist. Damit können inkrementelle Planungsprobleme, wie die Zuweisung eines Fahrzeugs zu einer neuen Buchung im Corporate Car Sharing, modelliert werden.

In dieser Arbeit wird die NP-Vollständigkeit von Inkrementelle Listenfärbung mit Hilfe des Spezialfalls auf vollständige Splitgraphen gezeigt. Des Weiteren wird gezeigt, dass es eine Reduktion, welche die strukturellen Eigenschaften der Eingabe in gewisser Weise beibehält, auf das Problem Inkrementelle Farbenreiche Stabile Mengen existiert, sodass Algorithmen zum Lösen von Inkrementelle Farbenreiche Stabile Mengen genutzt werden können, um das Problem Inkrementelle Listenfärbung zu lösen.

Zusätzlich werden Datenreduktionsalgorithemen mit polynomieller Laufzeit vorgestellt, welche gültige Eingaben für Inkrementelle Listenfärbung auf äquivalente Eingaben mit bestimmten strukturellen Eigenschaften einschränken.

Zum Lösen des Problems werden drei Suchbaumalgorithmen und zwei dynamische Programme vorgestellt. Die Laufzeiten der Algorithmen werden anschließend bezüglich der Anzahl der Intervalle und der Anzahl der Farben mit Hilfe von künstlich erzeugten Testinstanzen experimentell analysiert.

Inhaltsverzeichnis

1 Einleitung

Viele Unternehmen stellen ihren Mitarbeitern zur beruflichen und/oder privaten Nutzung Fahrzeuge zur Verfügung. Dies kann durch persönliche Dienstwagen für Mitarbeiter oder durch Corporate Car Sharing mit Poolfahrzeugen umgesetzt werden. Dabei bietet die Umsetzung mit persönlichen Dienstwagen für Mitarbeiter die höchste Mobilität, da der Dienstwagen zu jedem Zeitpunkt genutzt werden kann. Beim Corporate Car Sharing werden Unternehmensfahrzeuge in einem Fahrzeugpool mehreren Mitarbeitern zur Verfügung gestellt. Dabei ist die Anzahl der Fahrzeuge im Fahrzeugpool meist deutlich kleiner als die Anzahl der Mitarbeiter, welche Fahrzeuge aus dem Fahrzeugpool nutzen dürfen. Die Mitarbeiter können die Fahrzeuge je nach Bedarf und Verfügung für dienstliche oder private Zwecke nutzen. Da so mehrere Personen ein Fahrzeug nutzen, ist Corporate Car Sharing wesentlich kostengünstiger und nachhaltiger in der Nutzung der Fahrzeuge. Durch die damit einhergehenden Kostenersparnisse ist dieses Konzept auch für kleine Unternehmen attraktiv.

Damit ein Mitarbeiter beim Corporate Car Sharing ein Fahrzeug nutzen kann, muss dieser angeben, über welchen Zeitraum das Fahrzeug benutzt werden soll und welche zusätzlichen Kriterien, wie zum Beispiel einen großen Kofferraum, das Fahrzeug mindestens erfüllen soll. Im Folgenden wird dies als Buchungswunsch bezeichnet. Anhand des Buchungswunsches wird dem Nutzer durch einen Fahrzeugzuweisungsalgorithmus, wenn möglich, ein passendes Fahrzeug aus dem Fahrzeugpool zugewiesen, wodurch aus dem Buchungswunsch eine akzeptierte Buchung wird. Um möglichst viele Buchungswünsche zu erfüllen, wird dem Nutzer erst kurz vor Buchungsbeginn mitgeteilt, welches Fahrzeug ihm zugewiesen wurde. Dadurch hat der Algorithmus die Möglichkeit bereits akzeptierten Buchungen andere, den Kriterien der jeweiligen Buchung entsprechende Fahrzeuge zuzuweisen. Das hier beschriebene Problem ist ein typisches Planungsproblem auf Intervallgraphen, welches in verschiedenen Anwendungsbereichen gelöst werden muss [Kol+07]. Dabei ist ein *Intervallgraph* $G = (V, E)$ ein Graph, dessen Knoten auf der Zeitachse als rechtsoffene Intervalle dargestellt werden können, welche genau dann benachbart sind, wenn sich die Intervalle überschneiden.

Wie viele andere Planungsprobleme lässt sich das Problem der Zuweisung von Fahrzeugen im Corporate Car Sharing als NP-schweres Problem der Listenfärbung auf Intervallgraphen modellieren. Dieses Problem wurde von Erdős, Rubin und Taylor [ERT79] eingeführt und ist wie folgt definiert:

Listenfärbung [LF]
Eingabe: Ein ungerichteter Graph $G = (V, E)$, eine Farbmenge C und für jeden Knoten $v \in V$ eine Liste $L : V \to 2^C$ von zulässigen Farben.
Frage: Gibt es für G eine L-zulässige Färbung col_G?

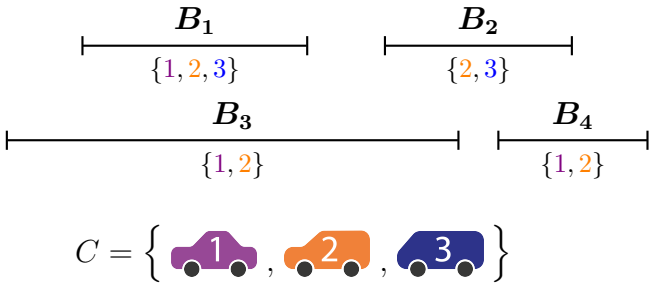


Abbildung 1.1: Hier wird das Planungsproblem im Corporate Car Sharing als Problem der Listenfärbung auf Intervallgraphen dargestellt.

Dabei ist eine Funktion $\text{col}_G : V \to C$ eine *L-zulässige Färbung*, wenn jedem Knoten eine Farbe aus der entsprechenden Liste zugewiesen wird und benachbarte Knoten unterschiedliche Farben zugewiesen bekommen.

Das Problem der Zuweisung von Fahrzeugen lässt sich wie in Abbildung 1.1 dargestellt modellieren. Dabei entspricht jedes Intervall des Intervallgraphen G einer Buchung B_i. Die Farbmenge C entspricht der Menge aller Fahrzeuge und die Listen von zulässigen Farben entsprechen den Fahrzeugen, welche die Buchungskriterien erfüllen. Da die Intervalle den Buchungen und die Farben den Fahrzeugen entsprechen, weist eine L-zulässige Färbung col_G jeder Buchung ein Fahrzeug und sich überschneidenden Buchungen unterschiedliche Fahrzeuge zu.

Weitere Anwendungen, welche durch das Problem der Listenfärbung auf Intervallgraphen modelliert werden können, sind zum Beispiel die Kanalzuweisung von Mesh-Netzwerken und die Registerzuordnung für Rechenoperationen. Bei dem Kanalzuweisungsproblem für Mesh-Netzwerke sind die verfügbaren Frequenzen durch die Hardware beschränkt und zwei Geräte können aufgrund von Interferenzen nicht gleichzeitig auf einer Frequenz senden [Ram+06]. Bei der Registerzuordnung für Rechenoperationen besitzen die Recheneinheiten unterschiedliche Funktionalitäten [ZW03].

In dieser Arbeit wird das Problem der Listenfärbung um die Eingabe einer zulässigen Färbung für alle Knoten bis auf einen erweitert. Diese Erweiterung des Problems wird Inkrementelle Listenfärbung genannt. Des Weiteren wird gezeigt, dass eine gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe mit ähnlichen strukturellen Eigenschaften für Inkrementelle Vollständig Farbenreiche Stabile Menge reduziert werden kann, sodass die von van Bevern, Mnich, Niedermeier und Weller [Bev+15] eingeführten parametrisierte Algorithmen zum Finden von stabilen Mengen genutzt werden können um das Problem Inkrementelle Listenfärbung zu lösen. Des Weiteren werden Datenreduktionsalgorithmen mit einer polynomiellen Laufzeit von $O(|C|\cdot n^4 \log n)$ vorgestellt. Mit Hilfe dieser Algorithmen wird jede gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Inkrementelle Listenfärbung oder Listenfärbung mit bestimmten strukturellen Eigenschaften reduziert. Zum Lösen von Inkrementelle Listenfärbung werden fünf Algorithmen vorgestellt, zwei dynamische Programme mit Laufzeit $O(2^{|C|}\cdot n)$ und $O(2^Q \cdot n)$ sowie ein Suchabaumalgorithmus mit Laufzeit $O(\Gamma^{|C|} \cdot n)$ zum Lösen

von Inkrementelle Vollständig Farbenreiche Stabile Menge und zwei Suchbaumalgorithmen zum Lösen von Listenfärbung beziehungsweise zum Lösen von Inkrementelle Listenfärbung mit einer Laufzeit von $O(|C|^n \cdot n)$.

Die Arbeit ist wie folgt aufgebaut. In Kapitel 2 werden die grundlegende Notation und Definitionen zusammengefasst. Des Weiteren wird das Problem Listenfärbung erweitert, sodass die erweiterten Probleme direkt das Corparate Car Sharing Problem lösen, in welchem inkrementell neue Buchungswünsche dazukommen. In Kapitel 3 wird bewiesen, dass die in dieser Arbeit zu lösenden Probleme NP-vollständig sind. In Kapitel 4 wird eine Datenstruktur für Intervallgraphen und eine Datenstruktur für Farbmengen vorgestellt. In Kapitel 5 werden Datenreduktionsalgorithmen vorgestellt, um die Eingabe auf eine äquivalente Eingabe mit bestimmten strukturellen Eigenschaften zu reduzieren. Anschließend werden in Kapitel 6 Algorithmen zum Lösen der Probleme Listenfärbung, Inkrementelle Listenfärbung und Vollständig Farbenreiche Stabile Menge erläutert. Darauffolgend findet in Kapitel 7 eine experimentelle Auswertung der implementierten Algorithmen statt. Abschließend wird in Kapitel 8 ein Ausblick über die Inhalte der Arbeit und weitere sich daraus entwickelnde Themen gegeben.

2 Grundlegende Definitionen und speziellere Probleme

In diesem Kapitel werden in Abschnitt 2.1 die für diese Arbeit wichtigen grundlegenden Notationen und Definitionen zusammengefasst. In Abschnitt 2.2 wird das Problem Vollständig Farbenreiche Stabile Menge definiert und der Zusammenhang zum Corporate Car Sharing erläutert. Abschließend werden in Abschnitt 2.3 die Probleme Inkrementelle Listenfärbung und Inkrementelle Vollständig Farbenreiche Stabile Menge definiert, welche es für das Planungsproblem im Corporate Car Sharing zu lösen gilt.

2.1 Grundlegende Definitionen

Alle in diesem Abschnitt angegeben grundlegende Notationen und Begriffe sind ähnlich zu denen von Diestel [Die16]. In dieser Arbeit werden nur ungerichtete endliche Graphen $G = (V, E)$ mit Knotenmenge $V(G)$ und Kantenmenge $E(G)$ betrachtet. Falls bekannt ist, von welchem Graphen gesprochen wird, wird die Knotenmenge und Kantenmenge mit V und E bezeichnet. Die Anzahl der Knoten wird mit $n := |V|$ und die Anzahl der Kanten mit $m := |E|$ bezeichnet. In einem Graphen $G = (V, E)$ sind zwei Knoten $v, w \in V$ *benachbart*, wenn $\{v, w\} \in E$ gilt. Die *Nachbarschaft* $N_G(v)$ eines Knotens $v \in V$ aus G ist die Menge aller benachbarten Knoten von v mit $N_G(v) \cap \{v\} = \emptyset$. Ein Graph $G' = (V', E')$ ist ein *Subgraph* von $G = (V, E)$, wenn V' eine Teilmenge von V ist und E' eine Teilmenge von E ist. Der Graph $G[V'] = (V', E')$ ist ein *induzierter Subgraph* von $G = (V, E)$, wenn V' eine Teilmenge von V ist und E' alle Kanten zwischen Knoten aus V' enthält, die auch in E enthalten sind, das heißt es gilt $V' \subseteq V$ und $E' = \big\{\{u, v\} \in E \mid u, v \in V'\big\}$. Sei ein Graph $G = (V, E)$ und ein Knoten $v \in V$ gegeben, dann ist $G - v := G\big[V \setminus \{v\}\big]$ definiert als induzierter Subgraph von G, welcher durch die Knotenmenge $V \setminus \{v\}$ entsteht. Ein *Pfad* in G von Knoten v_1 zu Knoten v_l ist ein Tupel $(v_1, v_2, \ldots, v_l) \in V^l$ von Knoten mit $\{v_i, v_{i+1}\} \in E$ für $i \in \{1, \ldots, l-1\}$. Dabei wird v_1 als Startknoten und v_l als Endknoten des Pfades $(v_1, v_2, \ldots, v_l)$ bezeichnet.

Eine Knotenmenge $C \subseteq V$ ist eine *Clique* in G, wenn sie nur paarweise benachbarte Knoten enthält, das heißt es gilt $\{v, w\} \in E$ für alle $v, w \in C$ mit $v \neq w$. Eine Clique $C \subseteq V$ heißt *maximale Clique*, wenn es keine größere Clique $C' \subseteq V$ mit $|C| < |C'|$ gibt. Sei G ein Graph, dann ist die *Cliquenzahl* $\omega(G)$ die Anzahl der Knoten der maximalen Clique in G. Eine Knotenmenge $S \subseteq V$ ist eine *stabile Menge* in G, wenn sie nur paarweise nicht benachbarte Knoten enthält, das heißt es gilt $\{v, w\} \notin E$ für alle $v, w \in S$ mit $v \neq w$. Eine Menge $M \subseteq V$ ist eine *Knotenüberdeckungsmenge* von G, wenn für alle

Kanten $\{v, w\} \in E$ mindestens $v \in M$ oder $w \in M$ gilt, das heißt M enthält mindestens einen Knoten jeder Kante aus E. Ein Graph G heißt *zusammenhängend*, falls es für alle Knoten $v, w \in V$ einen Pfad in G mit v als Startknoten und w als Endknoten gibt. Eine *Zusammenhangskomponente* von G ist ein maximal zusammenhängender induzierter Subgraph von G.

Eine *Farbmenge* $C := \{1, 2, ..., k\}$ ist eine Menge von natürlichen Zahlen. Für eine Farbmenge C und einen Graph $G = (V, E)$ ist die Zuweisung $\text{col}_G : V \to C$ eine *Färbung* von G. Für den Graph G mit der Färbung col_G, ist die Menge $S \subseteq V$ eine *farbenreiche stabile Menge*, wenn S eine stabile Menge ist und wenn für alle Knoten $v, w \in S$ mit $v \neq w$ $\text{col}_G(v) \neq \text{col}_G(w)$ gilt. Eine *zulässige Färbung* für den Graphen G ist eine Zuweisung von Farben auf Knoten $v \in V$, sodass $col(v) \neq col(w)$ für jede Kante $\{v, w\} \in E$ gilt. Sei G ein Graph, dann ist die *chromatische Zahl* $\chi(G)$ die kleinste Zahl k, sodass eine zulässige Färbung col_G mit der Farbmenge $\{1, \ldots, k\}$ existiert. Sei zu der Farbmenge C und dem Graphen G für jeden Knoten $v \in V$ eine Liste von zulässigen Farben durch $L : V \to 2^C$ gegeben, dann ist die Zuweisung $\text{col}_G : V \to C$ eine *L-zulässige Färbung* von G, wenn $\text{col}_G(v) \in L(v)$ für alle $v \in V$ und $\text{col}_G(v) \neq \text{col}_G(w)$ für alle $\{v, w\} \in E$ gilt. Sie L eine Zuweisung, welche jedem Knoten aus V eine Liste von zulässigen Farben aus der Farbmenge C zuweist, dann heißt $L[V']$ für die Knotenmenge $V' \subseteq V$ die *L-induzierte Zuweisung*, welche jedem Knoten aus V' die gleiche Liste von zulässigen Farben aus der der Farbmenge C zuweist. Das heißt für $V' \subseteq V$ gilt $L[V'] : V' \to 2^C$ mit $L[V'](v) = L(v)$ für alle Knoten $v \in V'$. Für einen Graphen G, eine Farbmenge C mit Teilmenge $C' \subseteq C$ und einer Färbung $\text{col}_G : V \to C$ ist $V[C'] := \{v \in V \mid \text{col}_G(v) \in C'\}$ als die *induzierte Knotenmenge bezüglich der Farbmenge* C' definiert.

Ein *Intervallgraph* $G = (V, E)$ ist ein Graph, dessen Knoten auf der Zeitachse als rechtsoffene Intervalle dargestellt werden können und genau dann benachbart sind, wenn die Intervalle sich überschneiden. Sei i ein Intervall der Intervalldarstellung des Intervallgraphen G, dann ist $v_i \in V$ als der zugehörige Knoten in G definiert. Ein Graph G heißt *perfekt*, wenn für jeden induzierten Subgraphen von G die chromatische Zahl gleich der Cliquenzahl ist, das heißt es gilt $\chi(G') = \omega(G')$ für alle induzierten Subgraphen G' von G. Jeder Intervallgraph ist ein perfekter Graph.

Ein Algorithmus besitzt eine *polynomielle Laufzeit*, wenn die Laufzeit höchstens polynomiell mit der Größe der Eingabe n wächst, das heißt es gilt für den Algorithmus $T(n) = O(n^k)$ für eine natürliche Konstante k.

Parametrisierte Algorithmen sind Algorithmen dessen Laufzeit durch ein Parameter k des zu lösenden Problems bestimmt wird. Ein Problem ist *fixed-parameter tractable* (FPT) bezüglich Parameter k, wenn ein Algorithmus existiert, welcher jede Instanz des Problems mit Größe n in einer Laufzeit von $f(k) \cdot p(n)$ löst, wobei f eine berechenbare Funktion und p ein beliebiges Polynom ist. Das heißt parametrisierte Algorithmen akzeptieren exponentielle Laufzeiten, was für das Lösen von NP-schweren Problemen derzeit sowieso unumgänglich ist, und können zugleich für kleine Parameter k effizient sein [Bev+15].

2.2 Vollständig Farbenreiche Stabile Menge

Da in dieser Arbeit das Problem Listenfärbung auf Intervallgraphen mit Hilfe von angepassten parametrisierten Algorithmen zum Finden von stabilen Mengen von van Bevern, Mnich, Niedermeier und Weller gelöst wird [Bev+15], wird in diesem Abschnitt das Problem VFSM eingeführt und der Zusammenhang zum Corporate Car Sharing hergestellt. Das Problem ist wie folgt definiert:

Vollständig Farbenreiche Stabile Menge [VFSM]
Eingabe: Ein Graph $G = (V, E)$, eine Farbmenge C und eine Färbung $\text{col}_G : V \to C$.
Frage: Gibt es für G eine farbenreiche stabile Menge der Größe $|C|$?

Im Folgenden wird das Problem Vollständig Farbenreiche Stabile Menge mit VFSM abgekürzt. Da VFSM ein allgemeineres Problem ist, als für den Anwendungsfall benötig wird, wird die Eingabe (G, C, col_G) so eingeschränkt, dass gültige Eingaben nur Instanzen sind, welche dem Anwendungsfall im Corporate Car Sharing entsprechen.

Sei $G = (V, E)$ ein Graph und K_i mit $i \in \{1, \ldots, k\}$ die Knotenmengen der Zusammenhangskomponenten von G. Sei C eine Farbmenge und $\text{col}_G : V \to C$ eine Färbung für G. Dann ist G *farbenreich bezüglich der Zusammenhangskomponenten*, wenn jeder Knoten aus einer Zusammenhangskomponente mit einer unterschiedlichen Farbe gefärbt wird. Das heißt für jede Knotenmenge K_i gilt $\text{col}_G(v) \neq \text{col}_G(w)$ für alle Knoten $v, w \in K_i$ mit $v \neq w$. Der Graph G ist *farbkonsistent*, wenn eine Menge $E_C \subseteq \binom{C}{2}$ existiert, sodass für jede Zusammenhangskomponente gilt, dass für alle Knoten $v, w \in V$ $\{\text{col}_G(v), \text{col}_G(w)\} \in E_C$ gilt, genau dann wenn $\{v, w\} \in E$ gilt.

Sei der Graph G mit Farbmenge C und Färbung col_G farbenreich bezüglich der Zusammenhangskomponenten und farbkonsistent. Dann entspricht G, C und col_G einer Instanz, welche mit dem Planungsproblem beim Corporate Car Sharing übereinstimmt. Eine solche Instanz ist in Abbildung 2.1 dargestellt. Die Farbmenge C entspricht der Menge aller Buchungen B_i und die Intervalle $B_{i,j}$ in G entsprechen den zuweisbaren Fahrzeugen aller Buchungen B_i. Das heißt, wenn für Buchung B_i das Fahrzeug j ein zuweisbares Fahrzeug ist, so wird dies als Intervall $B_{i,j}$ in G abgebildet. Die Färbung col_G färbt die Intervalle so, dass Intervalle die Farbe der Buchung erhalten, aus der sie erzeugt wurden. Da die Farbmenge den Buchungen entsprechen, weißt eine farbenreiche stabile Menge S für G mit Größe $|S| = |C|$ jeder Buchung genau ein Fahrzeug zu. Da S eine stabile Menge in G ist, wird keinen sich überschneidenden Buchungen das gleiche Fahrzeug zugewiesen.

2.3 Inkrementelle Listenfärbung und Inkrementelle Vollständig Farbenreiche Stabile Menge

Mit der Annahme, dass Buchungswünsche inkrementell vom Algorithmus abgearbeitet werden, folgt, dass der Algorithmus zum Zeitpunk der Abarbeitung eines neuen Buchungswunsches eine Lösung für alle bis zu diesem Zeitpunk akzeptierten Buchungen

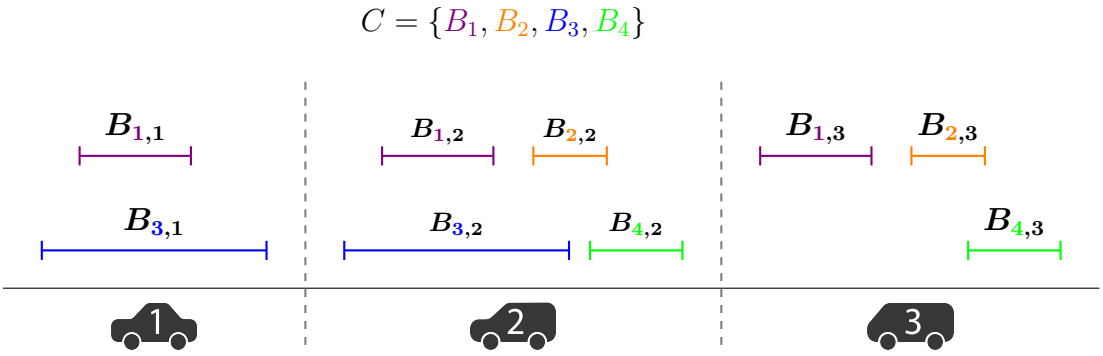


Abbildung 2.1: Hier wird das Planungsproblem im Corporate Car Sharing als Problem von Vollständig Farbenreiche Stabile Menge dargestellt.

kennt. Um dies mit den Problemen LF und VFSM abbilden zu können, werden diese um die bekannte Teillösung erweitert. Dabei wird die Eingabe von LF um einen Knoten $v^* \in V$ und eine L-zulässige Färbung col_{G-v^*} für den Graphen $G - v^*$ erweitert, sodass folgendes Problem entsteht:

Inkrementelle Listenfärbung [ILF]

Eingabe: Ein Graph $G = (V, E)$, ein Knoten $v^* \in V$, eine Farbmenge C, für jeden Knoten $v \in V$ eine Liste $L : V \to 2^C$ von zulässigen Farben und eine L-zulässige Färbung col_{G-v^*}.

Frage: Gibt es für G eine L-zulässige Färbung col_G?

Im Folgenden wird das Problem Inkrementelle Listenfärbung mit ILF abgekürzt. Die Eingabe von VFSM wird um eine Farbe $c^* \in C$ und eine farbenreiche stabile Menge S der Größe $|C| - 1$ für $G\big[V\,[C \setminus \{c^*\}]\,\big]$ erweitert, sodass folgendes Problem entsteht:

Inkrementelle Vollständig Farbenreiche Stabile Menge [IVFSM]

Eingabe: Ein Graph $G = (V, E)$, eine Farbmenge C, eine Färbung $\text{col}_G : V \to C$, eine Farbe $c^* \in C$ und eine farbenreiche stabile Menge S der Größe $|C| - 1$ für $G\big[V\,[C \setminus \{c^*\}]\,\big]$.

Frage: Gibt es für G eine farbenreiche stabile Menge der Größe $|C|$?

Im Folgenden wird das Problem Inkrementelle Vollständig Farbenreiche Stabile Menge mit IVFSM abgekürzt.

3 Komplexität

In diesem Kapitel wird die NP-Vollständigkeit von Inkrementelle Listenfärbung gezeigt. Mit Hilfe einer Reduktion für jede gültige Eingabe von Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Inkrementelle Vollständig Farbenreiche Stabile Menge wird die NP-Vollständigkeit von Inkrementelle Vollständig Farbenreiche Stabile Menge bewiesen. Unter Annahme, dass $P \neq NP$ gilt, wird somit gezeigt, dass es keine Algorithmen mit polynomieller Laufzeit gibt, welche die Problemstellungen lösen. Da die in diesem Kapitel gezeigte Reduktion die strukturellen Eigenschaften der Eingabe von Inkrementelle Listenfärbung weitestgehend beibehält, können zum Lösen von Instanzen für Inkrementelle Listenfärbung sowohl Algorithmen zum Lösen von Inkrementelle Listenfärbung als auch zum Lösen von Inkrementelle Vollständig Farbenreiche Stabile Menge genutzt werden.

In Abschnitt 3.1 wird auf die Komplexität der Problemstellungen Listenfärbung und Inkrementelle Listenfärbung eingegangen. Dabei wird jeweils mit Hilfe einer Reduktion auf einen Spezialfall des jeweiligen Problems die NP-Vollständigkeit gezeigt. Mit dem Wissen, dass Listenfärbung und Inkrementelle Listenfärbung NP-vollständig sind, wird in Abschnitt 3.2 gezeigt, dass die Problemstellungen Vollständig Farbenreiche Stabile Menge und Inkrementelle Vollständig Farbenreiche Stabile Menge NP-vollständig sind.

3.1 NP-Vollständigkeit von Listenfärbung und Inkrementelle Listenfärbung

In diesem Abschnitt wird gezeigt, dass folgende Probleme NP-vollständig sind:

Listenfärbung [LF]
Eingabe: Ein Graph $G = (V, E)$, eine Farbmenge C und für jeden Knoten $v \in V$ eine Liste $L : V \to 2^C$ von zulässigen Farben.
Frage: Gibt es für G eine L-zulässige Färbung col_G?

Inkrementelle Listenfärbung [ILF]
Eingabe: Ein Graph $G = (V, E)$, ein Knoten $v^* \in V$, eine Farbmenge C, für jeden Knoten $v \in V$ eine Liste $L : V \to 2^C$ von zulässigen Farben und eine L-zulässige Färbung col_{G-v^*}.
Frage: Gibt es für G eine L-zulässige Färbung col_G?

Dabei wird die NP-Vollständigkeit mit Hilfe von Reduktionen auf Spezialfälle von LF und ILF gezeigt.

Für die Spezialfälle von LF und ILF werden zuerst Splitgraphen eingeführt. Ein Graph $G = (V, E)$ ist ein *Splitgraph*, wenn dessen Knoten in eine Clique $C \subsetneq V$ und eine stabile Menge $S \subsetneq V$ partitioniert werden können, sodass $C \cap S = \emptyset$ und $C \cup S = V$ gilt. Ein Graph G ist ein *vollständiger Splitgraph*, wenn G ein Splitgraph mit Clique C und stabiler Menge S ist und alle Knoten aus S benachbart zu jedem Knoten aus C sind.

Der Spezialfall Listenfärbung auf vollständigen Splitgraphen wird im Folgenden mit LFVS abgekürzt und der Spezialfall Inkrementelle Listenfärbung auf vollständigen Splitgraphen wird im Folgenden mit ILFVS abgekürzt. Da für das Zuordnungsproblem im Bereich Corporate Car Sharing die NP-Vollständigkeit der Probleme auf Intervallgraphen interessant ist, wird zuerst gezeigt, dass jeder vollständige Splitgraph ein Intervallgraph ist.

Satz 3.1.1. *Jeder vollständige Splitgraph $G = (V, E)$ ist ein Intervallgraph.*

Beweis. Sei $G = (V, E)$ ein vollständiger Splitgraph. Dann gibt es eine Clique $C \subseteq V$ und eine stabile Menge $S \subseteq V$, sodass $C \cap S = \emptyset$ und $C \cup S = V$ gilt. Es bleibt zu zeigen, dass alle Knoten von G auf der Zeitachse als rechtsoffene Intervalle dargestellt werden können, welche sich genau dann überschneiden, wenn die zugehörigen Knoten in G benachbart sind. Die Knoten aus C lassen sich als rechtsoffene Intervalle mit Startzeitpunkt 0 und Endzeitpunkt $|S|$ darstellen. Alle so dargestellten Intervalle überschneiden sich in der Intervalldarstellung und bilden weiterhin eine Clique im Intervallgraphen. Sei $F : S \to \{0, \ldots, |S| - 1\}$ eine bijektive Funktion, welche jedem Knoten aus S eine natürliche Zahl zwischen 0 und $|S| - 1$ als Startzeitpunkt zuweist. Mit Hilfe von F lässt sich jeder Knoten $v \in S$ als rechtsoffenes Intervall mit Startzeitpunkt $F(v)$ und Endzeitpunkt $F(v) + 1$ darstellen. Da alle Intervalle rechtsoffen sind und F eine bijektive Funktion ist, überschneiden sich keine Intervalle der Knotenenge S. Daher bilden diese weiterhin eine stabile Menge im Intervallgraphen. Weil jedem Knoten aus S ein Startzeitpunkt zwischen 0 und $|S| - 1$ zugewiesen wurde, überschneiden sich alle Intervalle der Knotenmenge S mit allen Intervallen der Knotenmenge C. Das heißt jeder Knoten aus C ist mit jedem Knoten aus S im Intervallgraphen benachbart. Da alle Knoten von G auf der Zeitachse als rechtsoffene Intervalle dargestellt wurden und sich alle Intervalle überschneiden, wenn sie in G benachbart sind, so folgt dass jeder vollständige Splitgraph ein Intervallgraph ist. □

Des Weiteren kann gezeigt werden, dass jeder vollständige Splitgraph mit einer nicht leeren Kantenmenge ein zusammenhängender Intervallgraph ist.

Satz 3.1.2. *Jeder vollständige Splitgraph $G = (V, E)$ mit $E \neq \emptyset$ ist ein zusammenhängender Intervallgraph.*

Beweis. Sei $G = (V, E)$ ein vollständiger Splitgraph mit $E \neq \emptyset$. Aus Satz 3.1.1 folgt, dass G ein Intervallgraph ist. Es bleibt zu zeigen, dass G zusammenhängend ist. Da G ein vollständiger Splitgraph ist und $E \neq \emptyset$ gilt, gibt es eine nichtleere Clique $C \subseteq V$ und eine stabile Menge $S \subsetneq V$, sodass $C \cap S = \emptyset$ und $C \cup S = V$ gilt. Die Menge C ist nichtleer,

da G mindestens eine Kante enthält und in jedem vollständigen Splitgraphen für jede Kante $\{v, w\} \in E$ entweder $v, w \in C$ oder $v \in C$ und $w \in S$ gilt. Da C eine nichtleere Clique ist und jeder Knoten aus C mit jedem Knoten aus S benachbart ist, so folgt dass es mindestens einen Knoten $v^* \in C$ gibt, welcher mit allen Knoten in G benachbart ist. Somit ist G ein zusammenhängender Graph. Daher folgt, dass $G = (V, E)$ mit $E \neq \emptyset$ ein zusammenhängender Intervallgraph ist. □

Als nächstes wird gezeigt, dass es einen Algorithmus mit polynomieller Laufzeit gibt, welcher aus jeder Eingabe für das Knotenüberdeckungsproblem eine äquivalente Eingabe für Listenfärbung auf Vollständigen Splitgraphen konstruiert. Das NP-vollständige Knotenüberdeckungsproblem ist wie folgt definiert [Kar72]:

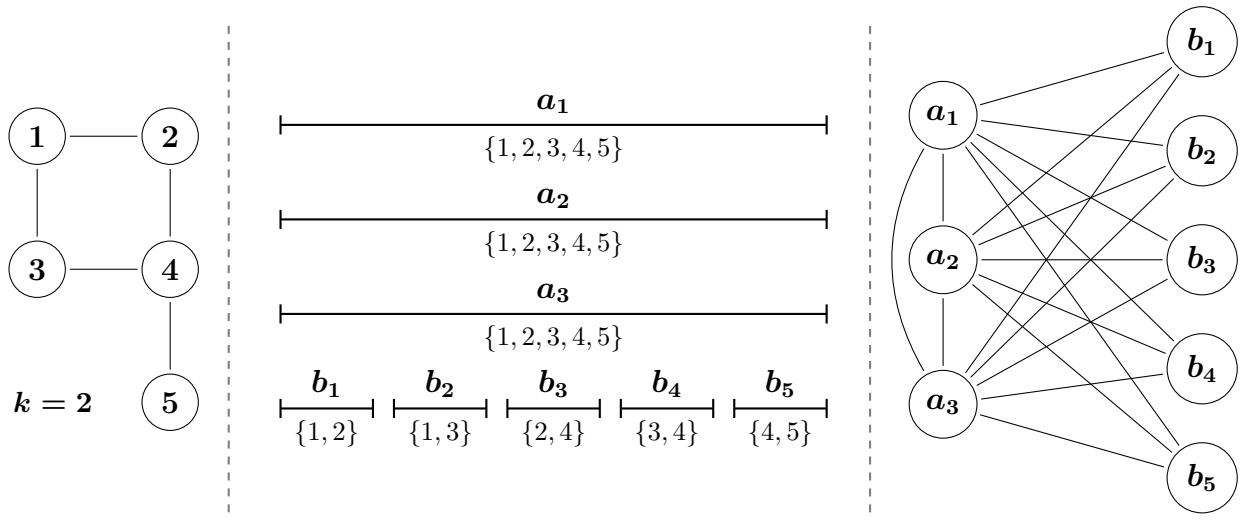
Knotenüberdeckungsproblem [KUE]
Eingabe: Ein Graph $G = (V, E)$ und eine natürliche Zahl k.
Frage: Gibt es für G eine Knotenüberdeckungsmenge $M \subseteq V$ mit $|M| \leq k$?

Im Folgenden wird das Knotenüberdeckungsproblem mit KUE abgekürzt.

Lemma 3.1.3. *Es gibt einen Algorithmus $R_{\text{KUE}\to\text{LFVS}}$ mit polynomieller Laufzeit, welcher aus jeder Eingabe $(G = (V, E), k)$ für* KUE *eine äquivalente Eingabe $(G^* = (V^*, E^*), C, L)$ für* LFVS *konstruiert.*

Beweis. Sei $(G = (V, E), k)$ eine Eingabe für KUE wie in Abbildung 3.1 (a) dargestellt. Um aus dieser Eingabe eine äquivalente Eingabe für LFVS zu konstruieren, wird zuerst $C = \{1, \ldots, |V|\}$ gesetzt. Als nächstes wird eine Menge $A = \{a_1, \ldots, a_{n-k}\}$ von Knoten mit $L(a_i) = C$ für $i = 1, \ldots, n - k$ erzeugt. Sei $F : V \to \{1, \ldots, |V|\}$ eine bijektive Funktion, welche jedem Knoten aus V eine natürliche Zahl zwischen 1 und $|V|$ als Farbe zuweist. Mit Hilfe von F wird für jede Kante $\{v, w\} \in E$ ein Knoten b_j mit $L(b_j) = \{F(v), F(w)\}$ erzeugt. Sei $B = \{b_1, \ldots, b_m\}$ die Menge der so erzeugten Knoten. Dann ist der Graph $G^* = (A \cup B, \{\{a_i, a_j\} \mid a_i, a_j \in A, a_i \neq a_j\} \cup \{\{a_i, b_j\} \mid a_i \in A, b_j \in B\})$ ein vollständiger Splitgraph, wobei A die Clique und B die Stabile Menge bildet. In Abbildung 3.1 ist ein so konstruierter vollständiger Splitgraph G^* in (b) als Intervalldarstellung und in (c) als Graph abgebildet. Der Splitgraph G^*, die Farbmenge C und die jeweiligen zulässigen Farblisten L der Knoten aus G^* bilden eine gültige Eingabe für LFVS. Das Erzeugen der Knoten benötigt $O((n-k)+m)$ Schritte, da $n-k$ Knoten für A und m Knoten für B erzeugt werden. Das Erzeugen der Kanten in G^* benötigt $O(n^2 + (n \cdot m))$ Schritte, da $O(n^2)$ Kanten der Clique A und $n \cdot m$ Kanten zwischen allen Knoten aus A und B erzeugt werden. Somit ist $R_{\text{KUE}\to\text{LFVS}}$ ein polynomieller Algorithmusm, welcher aus jeder Eingabe von KUE eine gültige Eingabe von LFVS konstruiert.

Als nächstes wird die Korrektheit der Reduktion gezeigt. Daher wird im Folgenden gezeigt, dass eine Ja-Instanz für KUE nach der Reduktion durch $R_{\text{KUE}\to\text{LFVS}}$ ebenfalls eine Ja-Instanz für LFVS ist. Sei $(G = (V, E), k)$ eine Ja-Instanz von KUE, dann gibt es für G eine Knotenüberdeckungsmenge M der Größe k. Sei $M^* = \{F(v) \mid v \in M\}$ die Farbmenge der Knoten aus M. Nach Konstruktion existiert für jeden Knoten $b \in B$



(a) (b) (c)

Abbildung 3.1: In (a) ist ein Graph G mit einer gesuchten Knotenüberdeckung der Größe $k = 2$ abgebildet. Die Intervalldarstellung in (b) entspricht dem zu G gehörigen Graphen nach Konstruktion durch $R_{\text{KUE}\to\text{LFVS}}$. Die erzeugten Intervalle a_i mit $L(a_i) = C$ überschneiden alle Intervalle im Intervallgraphen. Die b_i Intervalle entsprechen den Kanten von G, dabei bilden die jeweiligen Endknoten der Kante die Liste $L(b_i)$ der zulässigen Farben für Knoten b_i. In Abbildung (c) ist der durch die Reduktion entstandene vollständige Splitgraph abgebildet.

mindestens eine zulässige Farbe $c \in L(b)$, für die $c \in M^*$ gilt, da die zulässigen Farblisten der Knoten in B aus den Kanten von G konstruiert wurden. Da B in G^* eine stabile Menge ist, kann für jeden Knoten $b \in B$ eine beliebige Farbe $c \in L(b)$ mit $c \in M^*$ gewählt und $\text{col}_{G^*}(b) = c$ gesetzt werden, ohne die Eigenschaften einer L-zulässigen Färbung von G^* zu verletzen. Des Weiteren gilt $|A| = n - k$, $|V| = |C|$, $|M| = k$ und $|M| = |M^*|$, woraus $|A| = n - k = |V| - |M| = |C| - |M^*|$ folgt. Für alle Knoten $a \in A$ gilt nach Konstruktion $L(a) = C$. Somit können die Knoten aus der Clique A mit jeweils unterschiedlichen Farben aus $C \setminus M^*$ gefärbt werden, sodass $\text{col}_{G^*}(a) \neq \text{col}_{G^*}(a')$ für alle $a, a' \in A$ und $a \neq a'$ gilt. Damit ist die Färbung col_{G^*} eine L-zulässige Färbung für G^*, woraus folgt, dass $R_{\text{KUE}\to\text{LFVS}}(G, k)$ ebenfalls eine Ja-Instanz von LFVS ist.

Abschließend wird gezeigt, dass eine Ja-Instanz für LFVS nach Konstruktion von $R_{\text{KUE}\to\text{LFVS}}$ ebenfalls eine Ja-Instanz für KUE ist. Sei $(G^* = (V^*, E^*), C, L)$, mit einem vollständigem Splitgraphen G^*, einer Farbmenge C und für jeden Knoten $v \in V^*$ eine Liste von zulässigen Farben $L : V^* \to 2^C$, eine Ja-Instanz für LFVS, welche durch $R_{\text{KUE}\to\text{LFVS}}(G = (V, E), k)$ konstruiert wurde. Sei col_{G^*} die L-zulässige Färbung der Ja-Instanz für LFVS. Nach Konstruktion, lässt sich V^* in eine Clique A und eine stabile Menge B partitionieren. Sei $C_B = \{\text{col}_{G^*}(b) \mid b \in B\}$ die Menge der durch die L-zulässigen Färbung zugewiesenen Farben von B. Nach Konstruktion gilt $L(v) \subseteq C$ für alle Knoten $v \in V^*$, woraus $C_B \subseteq C$ folgt. Angenommen es existiert in G eine

Kante $\{v, w\} \in E$, sodass $F(v) \notin C_B$ und $F(w) \notin C_B$ gilt. Nach Konstruktion wurde für die Kante $\{v, w\}$ ein Knoten $b^* \in B$ mit $L(b^*) = \{F(v), F(w)\}$ erzeugt. Da $F(v) \notin C_B$ und $F(w) \notin C_B$ gilt, folgt dass $\text{col}_{G^*}(b^*) \notin L(b^*)$ gilt. Dies ist ein Widerspruch dazu, dass col_{G^*} eine L-zulässige Färbung ist. Also kann in G keine Kante $\{v, w\} \in E$ mit $F(v) \notin C_B$ und $F(w) \notin C_B$ existieren. Somit ist $M = \{F^{-1}(c) \mid c \in C_B\}$ eine Knotenüberdeckungsmenge von G. Sei $C_A = \{\text{col}_{G^*}(a) \mid a \in A\}$ die Menge der durch die L-zulässige Färbung zugewiesenen Farben von A. Da A eine Clique ist und somit alle Knoten von A untereinander benachbart sind, gilt $|C_A| = |A|$. Des Weiteren sind alle Knoten aus A mit allen Knoten aus B benachbart, wodurch $\text{col}_{G^*}(a) \neq \text{col}_{G^*}(b)$ für alle Knoten $a \in A$ und alle Knoten $b \in B$ gilt. Da $|A| = n - k$, $|V| = n$ und $|C| = |V|$ gilt, können den Knoten aus B nur maximal k unterschiedliche Farben zugewiesen werden. Somit folgt, dass $|C_B| \leq k$ und insbesondere $|M| \leq k$ gilt, da F eine bijektive Funktion ist. Damit wurde gezeigt, dass M eine Knotenüberdeckungsmenge mit $|M| \leq k$ für G ist, woraus folgt, dass (G, k) ebenfalls eine Ja-Instanz für KUE ist. □

Mit Hilfe von $R_{\text{KUE}\to\text{LFVS}}$ kann nun gezeigt werden, dass Listenfärbung auf vollständigen Splitgraphen NP-vollständig ist.

Satz 3.1.4. Listenfärbung *auf vollständigen Splitgraphen ist NP-Vollständig.*

Beweis. Gemäß der Definition von NP-Vollständigkeit ist zu zeigen, dass Listenfärbung auf vollständigen Splitgraphen in NP liegt und dass es NP-schwer ist [Kar72]. Zuerst wird gezeigt, dass es in NP liegt. Dazu färbt ein nichtdeterministischer polynomieller Algorithmus zunächst alle Knoten von $G = (V, E)$, indem für jeden Knoten $v \in V$ eine Farbe $c \in L(v)$ nichtdeterministisch geraten und $\text{col}_G(v) = c$ gesetzt wird. Die Korrektheit der L-zulässigen Färbung col_G ist in $O(m)$ Schritten verifizierbar, indem für jede Kante $\{v, w\} \in E$ geprüft wird, ob $\text{col}_G(v) \neq \text{col}_G(w)$ gilt. Somit wurde gezeigt, dass ein nichtdeterministischer polynomieller Algorithmus existiert, welcher die Eingabe akzeptiert, wenn mindestens eine zulässige Listenfärbung für G existiert. Also liegt Listenfärbung auf vollständigen Splitgraphen in NP.

Um zu zeigen, dass LFVS NP-schwer ist, wird der polynomielle Algorithmus $R_{\text{KUE}\to\text{LFVS}}$ aus Lemma 3.1.3 benutzt. Dieser konstruiert für jede Eingabe $(G = (V, E), k)$ für KUE eine äquivalente Eingabe $(G^* = (V^*, E^*), C, L)$ für LFVS. Durch Karp [Kar72] ist bekannt, dass KUE NP-vollständig ist. Da KUE dadurch NP-schwer ist, folgt mit Hilfe des polynomiellen Algorithmus $R_{\text{KUE}\to\text{LFVS}}$, dass LFVS NP-schwer ist, da jede Eingabe für KUE auf eine Eingabe für LFVS reduziert werden kann. □

Nach Satz 3.1.1 ist jeder vollständige Splitgraph ein Intervallgraph. Daraus folgt, dass Listenfärbung auf vollständigen Splitgraphen auch ein Spezialfall von Listenfärbung auf Intervallgraphen ist, womit sich folgendes Korollar ergibt.

Korollar 3.1.5. Listenfärbung *auf Intervallgraphen ist NP-vollständig.*

Des Weiteren ist nach Satz 3.1.2 jeder vollständige Splitgraph $G = (V, E)$ mit $E \neq \emptyset$ ein zusammenhängender Intervallgraph. Da der Algorithmus $R_{\text{KUE}\to\text{LFVS}}$ nur für die

triviale Eingabe $(G = (V, E), k)$ mit $k = |V|$ für KUE einen vollständigen Splitgraphen G^* mit $E^* = \emptyset$ konstruiert, kann daraus gefolgert werden, dass Listenfärbung auf zusammenhängenden Intervallgraphen ebenfalls NP-vollständig ist.

Um die NP-Vollständigkeit von ILFVS zu zeigen, wird vorher gezeigt, dass Kanteninkrementelle Knotenüberdeckung NP-vollständig ist. Das Problem ist wie folgt definiert:

Kanteninkrementelle Knotenüberdeckung [KIKUE]
Eingabe: Ein Graph $G = (V, E)$, eine natürliche Zahl k, eine Kante $e^* \in E$ und eine Knotenüberdeckungsmenge M^* der Größe k für $G^* = (V, E \setminus \{e^*\})$.
Frage: Gibt es eine Knotenüberdeckung $M \subseteq V$ für G mit $|M| \leq k$?

Im Folgenden wird das Problem Kanteninkrementelle Knotenüberdeckung mit KIKUE abgekürzt.

Satz 3.1.6. Kanteninkrementelle Knotenüberdeckung *ist NP-Vollständig.*

Beweis. Gemäß der Definition von NP-Vollständigkeit ist zu zeigen, dass KIKUE in NP liegt und dass es NP-schwer ist [Kar72]. Zuerst wird gezeigt, dass KIKUE in NP liegt. Sei $G = (V, E)$ ein Graph, k eine natürliche Zahl, $e^* \in E$ eine Kante aus G und M^* eine Knotenüberdeckungsmenge der Größe k für $G^* = (V, E \setminus \{e^*\})$. Dann gibt es einen nichtdeterministischen polynomiellen Algorithmus, welcher nichtdeterministisch k Knoten aus G rät. Sei M die Menge der so geratenen Knoten, dann ist in $O(m)$ Schritten verifizierbar, ob M eine Knotenüberdeckungsmenge für G ist, indem für jede Kante $\{v, w\} \in E$ geprüft wird, ob $v \in M$ oder $w \in M$ gilt. Somit wurde gezeigt, dass für KIKUE ein nichtdeterministischer polynomieller Algorithmus existiert, welcher die Eingabe akzeptiert, wenn es für den Graphen G mindestens eine Knotenüberdeckungsmenge M der Größe k gibt. Also liegt KIKUE in NP.

Mit Hilfe einer Reduktion von KUE auf KIKUE wird im Folgenden gezeigt, dass KIKUE NP-schwer ist. Dafür wird zuerst gezeigt, dass eine Eingabe von KUE in polynomieller Zeit in eine Eingabe von KIKUE umgewandelt werden kann. Sei $(G = (V, E), k)$ eine Eingabe für KUE. Um daraus eine äquivalente Eingabe für KIKUE der Form $(G' = (V', E'), k', e^*, M^*)$ zu konstruieren, wird der Graph G kopiert und um die Knotenmenge $S = \{s_1, \ldots, s_{n-k}\}$, für die $S \cap V = \emptyset$ gilt, erweitert, sodass $V' = V \cup S$ gilt. Des Weiteren wird die Kantenmenge um die Kante $e^* = \{s_1, s_2\}$ und um die Kanten zwischen jedem kopierten Knoten aus V und jedem Knoten aus S erweitert, sodass $E' = E \cup \{\{v, s\} \mid v \in V, s \in S\} \cup \{e^*\}$ gilt. Sei $G' = (V', E')$ der so konstruierte Graph. In Abbildung 3.2 ist der so konstruierte Graph G' abgebildet. Nach Konstruktion ist G ein induzierter Subgraph von G' und die Menge S ist im Graphen $G^* = (V', E' \setminus \{e^*\})$ eine stabile Menge der Größe $n - k$. Sei $k' = |V|$, dann ist die Menge $M^* = V$ eine Knotenüberdeckungsmenge für $G^* = (V', E' \setminus \{e^*\})$, da jede Kante $\{v, w\} \in E' \setminus \{e^*\}$ entweder eine Kante in G oder eine Kante zwischen G und S ist, woraus $v \in V$ oder $w \in V$ folgt. Somit bildet $(G' = (V', E'), k', e^*, M^*)$ eine Eingabe für KIKUE. Das Erzeugen der Knoten benötigt $O(n + (n - k))$ Schritte, da alle Knoten aus G kopiert und $n - k$ Knoten der Menge S erzeugt werden. Das Erzeugen der Kanten

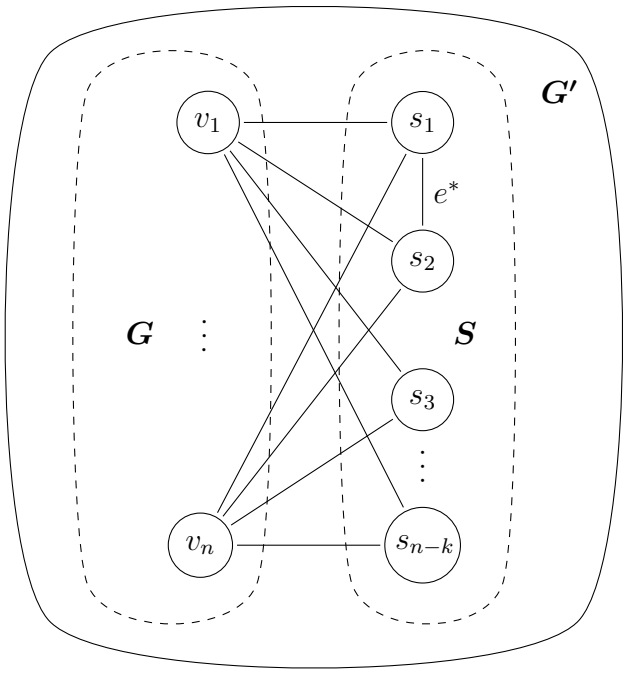


Abbildung 3.2: Hier dargestellt ist der konstruierte Graph G' nach Reduktion von KUE auf KIKUE, dabei ist G eine Kopie des Ausgangsgraphen, e^* die Kante welche inkrementell hinzugefügt wird und S eine stabile Menge in G' ohne Kante e^*.

in G' benötigt $O(m+n\cdot(n-k)+1)$ Schritte, da alle Kanten aus G kopiert, die Kante e^* und $n\cdot(n-k)$ Kanten zwischen allen Knoten aus V und S erzeugt werden. Somit kann eine Eingabe von KUE in polynomieller Zeit in eine Eingabe von KIKUE umgewandelt werden.

Als nächstes wird die Korrektheit der Reduktion gezeigt. Das heißt es wird gezeigt, dass eine Ja-Instanz von KUE nach der Reduktion ebenfalls eine Ja-Instanz von KIKUE ist und dass eine Ja-Instanz von KIKUE nach Konstruktion ebenfalls ein Ja-Instanz von KUE ist. Sei $(G=(V,E),k)$ eine Ja-Instanz von KUE und sei $(G'=(V',E'),k',e^*,M^*)$ die durch die Reduktion entstandene Eingabe für KIKUE. Da G nach Konstruktion von G' als ein induzierter Subgraph in G' enthalten ist und für G eine Knotenüberdeckungsmenge M der Größe k existiert, ist M eine Knotenüberdeckungsmenge des induzierten Subgraphen $G'[V]$. Das heißt für jede Kante $\{v,w\}\in E'$ mit $\{v,w\}\in E$ gilt $v\in M$ oder $w\in M$. Nach Konstruktion gilt $E'=E\cup\{\{v,s\}\mid v\in V,s\in S\}\cup\{e^*\}$ und zusätzlich ist e^* eine Kante zwischen zwei Knoten aus S, woraus für alle Kanten $\{v,w\}\in E'\setminus E$ folgt, dass $v\in S$ oder $w\in S$ gilt. Damit ist die Menge $M'=M\cup S$ eine Knotenüberdeckungsmenge für G', weil durch M alle Kanten aus E abgedeckt und durch S alle Kanten aus $\{\{v,s\}\mid v\in V,s\in S\}\cup\{e^*\}$ abgedeckt werden. Da $|M|=k$, $|S|=n-k$ und $k'=|V|$ gilt, so folgt dass M' eine Knotenüberdeckungsmenge der Größe $|M|+|S|=k+n-k=n=|V|=k'$ ist. Somit ist die Eingabe für KUE nach

der Reduktion ebenfalls eine Ja-Instanz von KIKUE.

Abschließend wird gezeigt, dass eine Ja-Instanz von KIKUE nach Konstruktion ebenfalls eine Ja-Instanz von KUE ist. Sei $(G' = (V', E'), k', e^*, M^*)$ eine Ja-Instanz von KIKUE, welche aus der Reduktion der Instanz $(G = (V, E), k)$ für KUE entstanden ist. Da eine Ja-Instanz für KIKUE gegeben ist, existiert eine Knotenüberdeckungsmenge M' der Größe k' für G'. Da nach Konstruktion $k' = |V|$ gilt und e^* eine Kante zwischen zwei Knoten aus S ist, gibt es einen Knoten $v^* \in V'$ aus der Knotenmenge V des induzierten Subgraphen $G'[V]$, für den $v^* \notin M'$ gilt. Da v^* mit jedem Knoten aus S benachbart ist, folgt für alle $s \in S$, dass $s \in M'$ gilt, weil sonst eine Kante $\{v^*, s^*\} \in E'$ existiert, für die $v^* \notin M'$ und $s^* \notin M'$ gilt, was ein Widerspruch dazu wäre, dass M' eine Knotenüberdeckungsmenge für G' ist. Nach Konstruktion gilt $E' = E \cup \{\{v, s\} \mid v \in V, s \in S\} \cup \{e^*\}$, sodass $v \in S$ oder $w \in S$ für alle Kanten $\{v, w\} \in E' \setminus E$ gilt. Daraus folgt, dass $M = M' \setminus S$ eine Knotenüberdeckungsmenge des induzierten Subgraphen $G'[V]$ ist. Da $|M'| = k'$, $|S| = n - k$ und $k' = |V|$ gilt, folgt, dass $|M| = |M'| - |S| = k' - n + k = |V| - n + k = k$ gilt. Da der induzierte Subgraph $G'[V]$ dem Graphen G entspricht, ist M eine Knotenüberdeckungsmenge der Größe k für G. Somit ist eine Ja-Instanz von KIKUE nach Konstruktion ebenfalls eine Ja-Instanz für KUE. Damit wurde gezeigt, dass KIKUE NP-schwer ist, woraus folgt, dass KIKUE NP-vollständig ist. □

Mit Hilfe der NP-vollständigen Problemstellung Kanteninkrementelle Knotenüberdeckung kann nun die NP-Vollständigkeit von Inkrementelle Listenfärbung auf vollständigen Splitgraphen gezeigt werden, woraus dann folgt, dass Inkrementelle Listenfärbung NP-vollständig ist.

Satz 3.1.7. Inkrementelle Listenfärbung *auf vollständigen Splitgraphen ist NP-vollständig.*

Beweis. Nach der Definition von NP-Vollständigkeit ist zu zeigen, dass ILFVS in NP liegt und dass es NP-schwer ist [Kar72]. Dass ILFVS in NP liegt, kann genauso gezeigt werden wie in Satz 3.1.4 gezeigt wurde, dass LFVS in NP liegt. Dazu färbt ein nichtdeterministischer polynomieller Algorithmus zunächst alle Knoten von $G = (V, E)$, indem für jeden Knoten $v \in V$ eine Farbe $c \in L(v)$ nichtdeterminitsich geraten und $\text{col}_G(v) = c$ gesetzt wird. Die Korrektheit dieser L-zulässigen Färbung col_G ist in $O(m)$ Schritten verifizierbar, indem für jede Kante $\{v, w\} \in E$ geprüft wird, ob $\text{col}_G(v) \neq \text{col}_G(w)$ gilt. Somit wurde gezeigt, dass ein nichtdeterministischer polynomieller Algorithmus existiert, welcher die Eingabe akzeptiert, wenn mindestens eine zulässige Listenfärbung für G existiert. Also liegt Inkrementelle Listenfärbung auf vollständigen Splitgraphen in NP.

Mit Hilfe einer Reduktion von KIKUE auf ILFVS wird im Folgenden gezeigt, dass ILFVS NP-schwer ist. Dafür wird zuerst gezeigt, dass jede Eingabe der Form $(G = (V, E), k, e^*, M^*)$ für KIKUE in polynomieller Zeit auf eine Eingabe der Form $(G^* = (V^*, E^*), v^*, C, L, \text{col}_{G-v^*})$ für ILFVS reduziert werden kann. Mit Hilfe des polynomiellen Algorithmus aus Lemma 3.1.3 wird durch $R_{\text{KUE}\to\text{LFVS}}(G, k)$ der Graph $G^* = (V^*, E^*)$, die Farbmenge $C = \{1, \ldots, n\}$ und die jeweiligen zulässigen Farblisten L

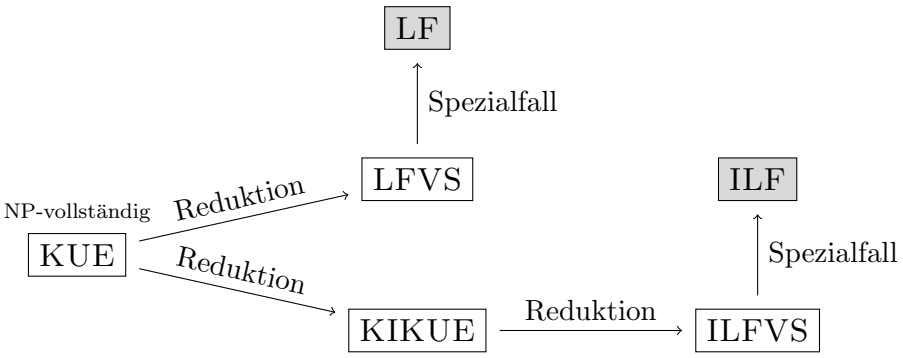


Abbildung 3.3: In dieser Abbildung wird dargestellt von welchen Problemen reduziert wird, um die NP-Vollständigkeit für LF und ILF zu zeigen.

der Knoten aus V^* konstruiert. Nach Konstruktion gibt es für G^* eine Clique A der Größe $n-k$ mit $L(a) = C$ für alle Knoten $a \in A$ und eine stabile Menge B der Größe m, sodass $A \cap B = \emptyset$ und $A \cup B = V^*$ gilt. Des Weiteren entspricht nach Konstruktion jede Kante $\{v, w\} \in V$ einem Knoten $b \in B$ mit $L(b) = \{F(v), F(w)\}$, wobei $F : V \to C$ eine bijektive Funktion von Knoten aus V auf Farben in C ist. Da $e^* \in E$ gilt, existiert nach Konstruktion ein Knoten $v^* \in B$ mit $L(v^*) = \{F(v'), F(w')\}$ und $e^* = \{v', w'\}$. Es bleibt zu zeigen, wie die Knotenüberdeckungsmenge M^* für $G' = (V, E \setminus \{e^*\})$ in eine L-zulässige Färbung $\text{col}_{G^*-v^*}$ umgewandelt wird. Sei $C^* = \{F(v) \mid v \in M^*\}$ die Farbmenge der Knoten aus M^*. Durch die Reduktion existiert für jeden Knoten $b \in B \setminus \{v^*\}$ mindestens eine zulässige Farbe $c \in L(b)$, für die $c \in C^*$ gilt, da die zulässigen Farblisten der Knoten in B aus den Kanten von G konstruiert wurden. Da B in G^* eine stabile Menge ist, kann für jeden Knoten $b \in B \setminus \{v^*\}$ eine Farbe $c \in L(b)$ mit $c \in C^*$ gewählt und $\text{col}_{G^*-v^*}(b) = c$ gesetzt werden, ohne die Eigenschaften einer L-zulässigen Färbung von $G^* - v^*$ zu verletzen. Des Weiteren gilt $|A| = n - k$, $|V| = |C|$, $|M^*| = k$ und $|M^*| = |C^*|$, woraus $|A| = n - k = |V| - |M^*| = |C| - |C^*|$ folgt. Für alle Knoten $a \in A$ gilt nach Konstruktion $L(a) = C$. Somit können die Knoten aus der Clique A mit jeweils unterschiedlichen Farben aus $C \setminus C^*$ gefärbt werden, sodass $\text{col}_{G^*-v^*}(a) \neq \text{col}_{G^*-v^*}(a')$ für alle $a, a' \in A$ und $a \neq a'$ gilt. Damit ist die Färbung $\text{col}_{G^*-v^*}$ eine L-zulässige Färbung für $G^* - v^*$, woraus folgt, dass $(G^* = (V^*, E^*), v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für ILFVS ist. Zum Erstellen von $\text{col}_{G^*-v^*}$ werden $O(n)$ Schritte benötigt, da jedem Knoten $b \in B$ eine Farbe aus $L(b)$ und jedem Knoten $a \in A$ ein Farbe aus $C \setminus C^*$ zugewiesen wird. Da $R_{\text{KUE}\to\text{LFVS}}$ ein polynomieller Algorithmus ist, kann eine Eingabe von KIKUE in polynomieller Zeit in eine gültige Eingabe von ILFVS umgewandelt werden.

Für die Reduktion von KIKUE auf ILFVS wurde, bis auf die Umwandlung der Kante e^* und der Knotenüberdeckungsmenge M^* für $G' = (V, E \setminus \{e^*\})$ auf den Knoten v^* und die L-zulässige Färbung $\text{col}_{G^*-v^*}$ für $G^* - v^*$, nur $R_{\text{KUE}\to\text{LFVS}}$ aus Lemma 3.1.3 benutzt. Entsprechend gibt es für G^* eine L-zulässige Färbung col_{G^*}, genau dann, wenn es für G eine Knotenüberdeckung der Größe k gibt. Damit wurde gezeigt, dass ILFVS NP-schwer ist, woraus folgt, dass ILFVS NP-vollständig ist. □

In Abbildung 3.3 ist dargestellt wie die Probleme, welche verwendet wurden um die

NP-Vollständigkeit von LF und ILF zu zeigen, in Bezug zueinander stehen. Zuerst wurde das NP-vollständige Problem KUE auf LFVS reduziert. Da LFVS ein Spezialfall von LF ist, konnte gefolgert werden, dass LF NP-vollständig ist. Als nächstes wurde mit Hilfe einer Reduktion von KUE auf KIKUE gezeigt, dass das Hilfsproblem KIKUE NP-vollständig ist. Damit konnte die NP-Vollständigkeit von ILFVS durch eine Reduktion von KIKUE auf ILFVS gezeigt werden. Da ILFVS ein Spezialfall von ILF ist, konnte gefolgert werden, dass ILF NP-vollständig ist.

3.2 NP-Vollständigkeit von Vollständig Farbenreiche Stabile Menge und Inkrementelle Vollständig Farbenreiche Stabile Menge

In diesem Abschnitt wird gezeigt, dass folgende Probleme NP-vollständig sind:

Vollständig Farbenreiche Stabile Menge [VFSM]
Eingabe: Ein Graph $G = (V, E)$, eine Farbmenge C und eine Färbung $\text{col}_G : V \to C$.
Frage: Gibt es für G eine farbenreiche stabile Menge der Größe $|C|$?

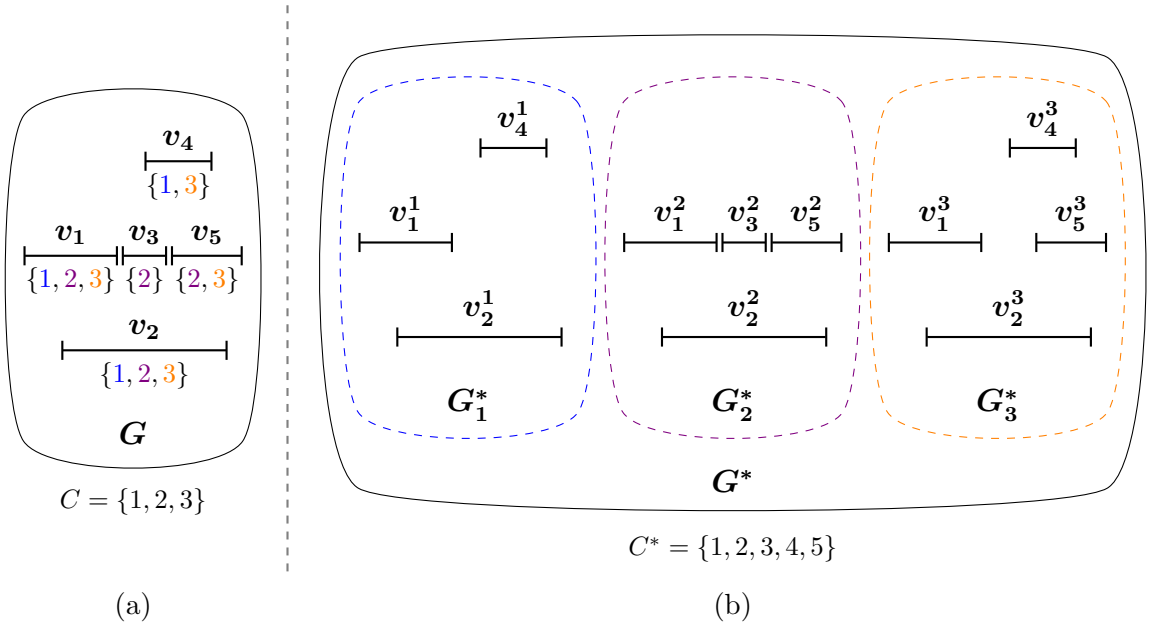
Inkrementelle Vollständig Farbenreiche Stabile Menge [IVFSM]
Eingabe: Ein Graph $G = (V, E)$, eine Farbmenge C, eine Färbung $\text{col}_G : V \to C$, eine Farbe $c^* \in C$ und eine farbenreiche stabile Menge S der Größe $|C| - 1$ für $G\big[V\,[C \setminus \{c^*\}]\,\big]$
Frage: Gibt es für G eine farbenreiche stabile Menge der Größe $|C|$?

Dafür wird zuerst gezeigt, dass es einen Algorithmus mit polynomieller Laufzeit gibt, welcher aus jeder Eingabe für Listenfärbung eine äquivalente Eingabe für Vollständig Farbenreiche Stabile Menge konstruiert.

Lemma 3.2.1. *Es gibt einen Algorithmus $R_{\text{LF}\to\text{VFSM}}$ mit polynomieller Laufzeit, welcher aus jeder Eingabe $(G = (V, E), C, L)$ für* LF *eine äquivalente Eingabe $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ für* Vollständig Farbenreiche Stabile Menge *konstruiert.*

Beweis. Die Konstruktion um aus einer Instanz von LF, wie in Abbildung 3.4 (a) dargestellt, eine äquivalente Instanz von VFSM, wie in Abbildung 3.4 (b) illustiert, zu konstruieren, besteht aus drei Schritten. Diese lassen sich wie folgt verallgemeinern. In Schritt eins werden induzierte Subgraphen anhand der Farben erzeugt. Anschließend werden in Schritt zwei die Knoten der so erzeugten Graphen umbenannt. Im letzten Schritt werden die Knotenmengen und Kantenmengen der Graphen vereinigt, sodass ein äquivalenter Graph für die Eingabe von VFSM erzeugt wird. Im Folgenden werden diese Schritte detaillierter erläutert.

Sei $(G = (V, E), C, L)$ eine Eingabe für LF mit $V = \{v_1, \ldots, v_n\}$ wie in Abbildung 3.4 (a) gegeben. Um für VFSM eine Eingabe der Form $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$



(a) (b)

Abbildung 3.4: In (a) ist eine Eingabe (G, C, L) für LF in Intervalldarstellung illustriert. In (b) ist die nach der Reduktion konstruierte Eingabe $(G^*, C^*, \text{col}_{G^*})$ dargestellt, dabei entsprechen die Subindizes der Knoten der Färbung col_{G^*}. Die induzierten Subgraphen G_i^* von G^* sind bis auf Umbenennung der Knoten immer noch induzierte Subgraphen von G.

zu konstruieren, wird für jede Farbe $c \in C$ der induzierte Subgraph $G'_c = G[V_c]$ der Knotenmenge $V_c = \{v_i \mid v_i \in V, c \in L(v_i)\}$ erzeugt. Durch Umbenennung der Knotenmengen der so erzeugten Graphen $G'_c = (V'_c, E'_c)$ mit $c \in C$ entstehen Graphen $G^*_c = (V^*_c, E^*_c)$ mit $V^*_c = \{v^c_i \mid v_i \in V'_c\}$ und $E^*_c = \{\{v^c_i, v^c_j\} \mid \{v_i, v_j\} \in E'_c\}$, sodass $V^*_{c_1} \cap V^*_{c_2} = \emptyset$ für alle $c_1, c_2 \in C$ gilt. Mit Hilfe dieser Graphen wird aus der Knotenmenge $V^* = \bigcup_{c \in C} V^*_c$ und der Kantenmenge $E^* = \bigcup_{c \in C} E^*_c$ der Graph $G^* = (V^*, E^*)$, wie in Abbildung 3.4 (b) dargestellt, konstruiert. Als Farbmenge für VFSM wird $C^* = \{1, \ldots, n\}$ definiert. Nach Konstruktion von G^* wird die Färbung $\text{col}_{G^*} : V^* \to C^*$ so konstruiert, dass $\text{col}_{G^*}(v^c_i) = i$ für alle $v^c_i \in V^*$ gilt. Der Graph G^*, die Farbmenge C^* und die Färbung col_{G^*} bilden die Eingabe für VFSM. Der Algorithmus $R_{\text{LF}\to\text{VFSM}}$ mit Eingabe $(G = (V, E), C, L)$ für LF benötigt maximal $O(|C| \cdot (n + m) + |C| \cdot (n + 1))$ Schritte, um die zugehörige Eingabe $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ für VFSM zu konstruieren, da höchstens $|C|$ mal alle n Knoten und m Kanten von G kopiert und umbenannt werden, was $O(|C| \cdot (n+m))$ Schritten entspricht. Des Weiteren wird die Farbmenge C^* in $O(n)$ Schritten konstruiert und für die Konstruktion der Färbung werden maximal $O(|C| \cdot n)$ Schritte benötigt, da der Algorithmus maximal $|C| \cdot n$ Knoten erzeugt. Damit ist $R_{\text{LF}\to\text{VFSM}}$ ein polynomieller Algorithmus.

Als nächstes wird die Korrektheit der Reduktion gezeigt. Somit wird im Folgenden gezeigt, dass eine Ja-Instanz von LF nach Reduktion durch $R_{\text{LF}\to\text{VFSM}}$ ebenfalls eine Ja-Instanz von VFSM ist. Sei $(G = (V, E), C, L)$ eine Ja-Instanz von LF und

sei $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ die durch $R_{\text{LF}\to\text{VFSM}}(G, C, L)$ konstruierte Eingabe für VFSM. Da (G, C, L) eine Ja-Instanz von LF ist, existiert eine L-zulässige Färbung col_G, sodass nach Definition einer L-zulässigen Färbung $\text{col}_G(v_i) \in L(v_i)$ für alle $v_i \in V$ und $\text{col}_G(v_i) \neq \text{col}_G(v_j)$ für alle $\{v_i, v_j\} \in E$ gilt. Nach Konstruktion existiert für jeden Knoten $v_i \in V$ und jeder Farbe $c \in L(v_i)$ genau ein Knoten $v_i^c \in V^*$, woraus folgt, dass $v_i^{\text{col}_G(v_i)} \in V^*$ gilt. Sei $S = \{v_i^{\text{col}_G(v_i)} \mid v_i \in V\}$ die Menge aller Knoten $v_i^{\text{col}_G(v_i)} \in V^*$, welche durch Knoten $v_i \in V$ mit Farbe $\text{col}_G(v_i) \in L(v_i)$ konstruiert wurde. Da $V = \{v_1, \ldots, v_n\}$ und nach Konstruktion $C^* = \{1, \ldots, n\}$ sowie $\text{col}_{G^*}(v_i^c) = i$ für alle $i \in \{1, \ldots, n\}$ und alle $c \in C$ gilt, so folgt dass $\{\text{col}_{G^*}(v_i^{\text{col}_G(v_i)}) \mid v_i \in V\} = \{1, \ldots, n\} = C^*$ gilt. Daraus folgt, dass S eine farbenreiche Menge ist. Da $|C^*| = n$ und $|V| = n$ gilt, folgt, dass $|S| = |\{v_i^{\text{col}_G(v_i)} \mid v_i \in V\}| = n = |C^*|$ gilt. Also ist S eine farbenreiche Menge der Größe $|C^*|$. Es bleibt zu zeigen, dass S eine stabile Menge in G^* ist. Angenommen S ist keine stabile Menge in G^*, dann existiert eine Kante $e^* = \{v_i^{\text{col}_G(v_i)}, v_j^{\text{col}_G(v_j)}\}$ zwischen zwei Knoten aus S. Da nach Konstruktion von G^* Knoten nur benachbart sind, wenn sie in einem der induzierten Subgraphen G'_c für $c \in C$ benachbart sind, folgt, dass $\text{col}_G(v_i) = \text{col}_G(v_j)$ gilt, was ein Widerspruch dazu ist, dass col_G eine L-zulässige Färbung ist. Also ist S eine farbenreiche stabile Menge für G^* der Größe $|C^*|$, woraus folgt, dass $R_{\text{LF}\to\text{VFSM}}(G, C, L)$ ebenfalls eine Ja-Instanz für VFSM ist.

Abschließend wird gezeigt, dass eine Ja-Instanz von VFSM nach Konstruktion von $R_{\text{LF}\to\text{VFSM}}$ ebenfalls eine Ja-Instanz von LF ist. Sei $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ eine Ja-Instanz, welche durch $R_{\text{LF}\to\text{VFSM}}(G = (V, E), C, L)$ konstruiert wurde. Zuerst wird gezeigt, dass nach Konstruktion für den Graphen G, die Farbmenge C und den Farblisten L aller Knoten aus V eine Färbung $\text{col}_G : V \to C$ existiert. Da die Instanz $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ eine Ja-Instanz von VFSM ist, existiert eine farbenreiche stabile Menge $S \subseteq V^*$ für G^* der Größe $|C^*|$. Da S eine farbenreiche Menge für G^* der Größe $|C^*|$ ist, gilt $\{\text{col}_{G^*}(v_i^c) \mid v_i^c \in S\} = C^*$. Da $C^* = \{1, \ldots, n\}$, $|S| = |C^*|$ und nach Konstruktion $\text{col}_{G^*}(v_i^c) = i$ für alle $v_i^c \in V^*$ gilt, existiert nach Konstruktion für alle $i \in \{1, \ldots, n\}$ genau ein Knoten $v_i^c \in S$ mit $c \in C$. Daher ist $\text{col}_G(v_i) = c$ für alle Knoten $v_i^c \in S$ eine Färbung für G. Es bleibt zu zeigen, dass col_G eine L-zulässige Färbung für G ist. Dafür ist zu zeigen, dass $\text{col}_G(v_i) \in L(v_i)$ für alle $v_i \in V$ und $\text{col}_G(v_i) \neq \text{col}_G(v_j)$ für alle $\{v_i, v_j\} \in E$ gilt. Zuerst wird gezeigt, dass $\text{col}_G(v_i) \in L(v_i)$ für alle $v_i \in V$ gilt. Angenommen es existiert für die Färbung col_G ein Knoten $v_i^* \in V$ mit $\text{col}_G(v_i^*) \notin L(v_i^*)$. Nach Konstruktion von col_G folgt aus $\text{col}_G(v_i^*) = c^*$, dass $v_i^{c^*} \in S$ gilt. Da $\text{col}_G(v_i^*) \notin L(v_i^*)$ gilt, folgt, dass $v_i^* \notin V_{c^*}$ mit $V_{c^*} = \{v_i \mid v_i \in V, c^* \in L(v_i)\}$ gilt, woraus wiederum nach Konstruktion folgt, dass $v_i^{c^*} \notin V^*$ und damit auch $v_i^{c^*} \notin S$ gilt. Dies ist ein Widerspruch dazu, dass $v_i^{c^*} \in S$ gilt, woraus folgt, dass $\text{col}_G(v_i) \in L(v_i)$ für alle Knoten $v_i \in V$ gilt. Nun wird gezeigt, dass $\text{col}_G(v_i) \neq \text{col}_G(v_j)$ für alle $\{v_i, v_j\} \in E$ gilt. Angenommen es existiert eine Kante $\{v_k^*, v_l^*\} \in E$, sodass $\text{col}_G(v_k^*) = c^* = \text{col}_G(v_l^*)$ mit $c^* \in C$ gilt. Da $\text{col}_G(v_i) = c$ für alle $v_i^c \in S$ gilt, folgt, dass $v_k^{c^*}, v_l^{c^*} \in S$ gilt. Nach Konstruktion von G^* gilt für beide Knoten $v_k^{c^*}, v_l^{c^*} \in V_{c^*}^*$, da für beide Knoten $c^* \in L(v_k)$ und $c^* \in L(v_l)$ gilt. Da $G_{c^*}^* = (V_{c^*}^*, E_{c^*}^*)$ nach Konstruktion durch einen induzierten Subgraphen von G entstanden ist, gilt $\{v_k^{c^*}, v_i^{c^*}\} \in E_{c^*}^*$. Da $E_{c^*}^* \subseteq E^*$ gilt, ist dies ein Widerspruch dazu, dass S eine stabile Menge in G^* ist. Somit gilt $\text{col}_G(v_i) \neq \text{col}_G(v_j)$

für alle $\{v_i, v_j\} \in E$. Also ist die Färbung col_G eine L-zulässige Färbung für G, woraus folgt, dass (G, C, L) ebenfalls eine Ja-Instanz für LF ist. □

Des Weiteren kann gezeigt werden, dass $R_{\text{LF}\to\text{VFSM}}$ für jede gültige Eingabe einen Graphen G^*, eine Farbmenge C^* und eine Färbung col_{G^*} konstruiert, sodass G^* farbkonsistent und farbenreich bezüglich der Zusammenhangskomponenten ist. Damit entspricht die konstruierte Eingabe für VFSM genau einer Eingabe des Zuordnungsproblems im Corporate Car Sharing.

Lemma 3.2.2. *$R_{\text{LF}\to\text{VFSM}}(G, V, L)$ konstruiert einen Graphen $G^* = (V^*, E^*)$, eine Farbmenge C^* und eine Färbung col_{G^*}, sodass G^* farbenreich bezüglich der Zusammenhangskomponenten und farbkonsistent ist.*

Beweis. Der Graph G^* wurde nach Konstruktion von $R_{\text{LF}\to\text{VFSM}}$ durch die Vereinigung von Knotenmengen und Kantenmengen der Graphen $G_c^* = (V_c^*, E_c^*)$ erzeugt, welche wiederum durch induzierte Subgraphen von G erzeugt wurden. Des Weiteren wurden die Knoten der Graphen G_c^* so umbenannt, dass $V_{c_1}^* \cap V_{c_2}^* = \emptyset$ für alle $c_1, c_2 \in C$ gilt, woraus folgt, dass $\{v_i^{c_1}, v_j^{c_2}\} \in E^*$ nur dann gilt, wenn $\{v_i, v_j\} \in E$ und $c_1 = c_2$ gilt. Daraus folgt, dass jede Zusammenhangskomponente ein induzierter Subgraph eines Graphen G_c^* für $c \in C$ ist. Damit können in jeder Zusammenhangskomponente keine zwei Knoten $v_i^c, v_j^c \in V^*$ mit $i = j$ existieren. Da nach Konstruktion die Indizes der Knotenmenge $V = \{v_1, \ldots, v_n\}$ den zugewiesenen Farben von col_{G^*} entsprechen, das heißt es gilt $\text{col}_{G^*}(v_i^c) = i$ für alle $v_i^c \in V^*$, so folgt, dass G^* farbenreich bezüglich der Zusammenhangskomponenten ist. Da alle Graphen G_c^* für $c \in C$ aus induzierten Subgraphen von G erzeugt wurden, folgt, dass wenn $\{v_i^c, v_j^c\} \in E^*$ eine Kante einer Zusammenhangskomponente ist, dass für jedes Knotenpaar $v_i^{c'}, v_j^{c'} \in V^*$ einer anderen Zusammenhangskomponente ebenfalls $\{v_i^{c'}, v_j^{c'}\} \in E^*$ gilt. Somit ist G^* kantenvollständig bezüglich der Farben in den Zusammenhangskomponenten. □

Mit Hilfe von $R_{\text{LF}\to\text{VFSM}}$ wird anschließend die NP-Vollständigkeit von VFSM auf Intervallgraphen gezeigt.

Satz 3.2.3. Vollständig Farbenreiche Stabile Menge *auf Intervallgraphen ist NP-vollständig.*

Beweis. Gemäß der Definition der NP-Vollständigkeit ist zu zeigen, dass VFSM in NP liegt und dass es NP-schwer ist [Kar72]. Zuerst wird gezeigt, dass VFSM in NP liegt. Sei ein Graph $G = (V, E)$, eine Farbmenge C und eine Färbung $\text{col}_G : V \to C$ gegeben. Dann gibt es einen nichtdeterministischen polynomiellen Algorithmus, welcher nichtdeterministisch $|C|$ Knoten aus V rät. Sei S die Menge der so geratenen Knoten, dann ist in $O((|S|^2 + |S|)/2)$ Schritten verifizierbar, dass S eine farbenreiche stabile Menge der Größe $|C|$ ist, indem für alle Knoten $v, w \in S$ mit $v \neq w$ überprüft wird, ob $\{v, w\} \notin E$ und $\text{col}_G(v) \neq \text{col}_G(w)$ gilt. Das heißt es wird überprüft, ob keine Knoten aus S in G benachbart sind und dass alle Knoten aus S unterschiedliche Farben besitzen. Somit wurde gezeigt, dass für VFSM ein nichtdeterministischer polynomieller Algorithmus

existiert, welcher die Eingabe akzeptiert, wenn es für den Graphen G mindestens eine farbenreiche stabile Menge der Größe $|C|$ gibt. Also liegt VFSM in NP.

Um zu zeigen, dass VFSM NP-schwer ist, wird der polynomielle Algorithmus $R_{\text{LF}\to\text{VFSM}}$ aus Lemma 3.2.1 benutzt. Dieser konstruiert für jede Eingabe $(G = (V, E), C, L)$ für LF eine äquivalente Eingabe $(G^* = (V^*, E^*), C^*, \text{col}_{G^*})$ für VFSM. Durch Korollar 3.1.5 ist bekannt, dass Listenfärbung auf Intervallgraphen NP-vollständig ist. Da LF dadurch NP-schwer ist, folgt mit Hilfe des polynomiellen Algorithmus $R_{\text{LF}\to\text{VFSM}}$, dass VFSM NP-schwer ist, da jede Eingabe für LF auf eine Eingabe für VFSM reduziert werden kann. □

Abschließend wird die NP-Vollständigkeit von IVFSM auf Intervallgraphen gezeigt. Durch Lemma 3.2.2 wird zusätzlich gezeigt, dass IVFSM auf Graphen, welche farbenreich bezüglich der Zusammenhangskomponenten und farbkonsisten sind, NP-vollständig ist.

Satz 3.2.4. Inkrementelle Vollständig Farbenreiche Stabile Menge *auf Intervallgraphen ist NP-vollständig, selbst wenn der Eingabegraph farbenreich bezüglich der Zusammenhangskomponenten und farbkonsistent ist.*

Beweis. Nach der Definition von NP-Vollständigkeit ist zu zeigen, dass ILFVS in NP liegt und dass es NP-schwer ist [Kar72]. Dass IVFSM in NP liegt, kann genauso gezeigt werden wie in Satz 3.2.3 gezeigt wurde, dass VFSM in NP liegt. Sei ein Graph $G = (V, E)$, eine Farbmenge C und eine Färbung $\text{col}_G : V \to C$ gegeben. Dann existiert ein nichtdeterministischer polynomieller Algorithmus, welcher nichtdeterministisch $|C|$ Knoten aus V rät. Sei S die Menge der so geratenen Knoten, dann ist in $O((|S|^2+|S|)/2)$ Schritten verifizierbar, dass S eine farbenreiche stabile Menge der Größe $|C|$ ist, indem für alle Knoten $v, w \in S$ mit $v \neq w$ überprüft wird, ob $\{v, w\} \notin E$ und $\text{col}_G(v) \neq \text{col}_G(w)$ gilt. Das heißt es wird überprüft, ob keine Knoten aus S in G benachbart sind und dass alle Knoten aus S unterschiedliche Farben besitzen. Somit wurde gezeigt, dass für IVFSM ein nichtdeterministischer polynomieller Algorithmus existiert, welcher die Eingabe akzeptiert, wenn es für den Graphen G mindestens eine farbenreiche stabile Menge der Größe $|C|$ gibt. Also liegt IVFSM in NP.

Mit Hilfe einer Reduktion von ILF auf IVFSM wird im Folgenden gezeigt, dass IVFSM NP-schwer ist. Dafür wird zuerst gezeigt, dass jede Eingabe der Form $(G = (V, E), v^*, C, L, \text{col}_{G-v^*})$ für ILF in polynomieller Zeit auf eine Eingabe der Form $(G^* = (V^*, E^*), C^*, \text{col}_{G^*}, c^*, S^*)$ für IVFSM reduziert werden kann. Mit Hilfe des polynomiellen Algorithmus aus Lemma 3.2.1 wird durch $R_{\text{LF}\to\text{VFSM}}(G, C, L)$ der Graph $G^* = (V^*, E^*)$, die Farbmenge $C^* = \{1, \ldots, n\}$ und die Färbung col_{G^*} konstruiert. Durch Lemma 3.2.2 folgt, dass G^* farbenreich bezüglich der Zusammenhangskomponenten und farbkonsistent ist. Nach Konstruktion von $R_{\text{LF}\to\text{VFSM}}$ entsprechen die Indizes der Knotenmenge $V = \{v_1, \ldots, v_n\}$ den zugewiesenen Farben von col_{G^*}, das heißt es gilt $\text{col}_{G^*}(v_i^c) = i$ für alle Knoten $v_i^c \in V^*$. Setze $c^* = \text{col}^*_G(v^*)$, damit c^* nach Konstruktion dem Index von v^* entspricht. Es bleibt zu zeigen, wie die L-zulässige Färbung col_{G-v*} für $G - v^*$ in eine farbenreiche stabile Menge S^* der Größe $|C^*| - 1$ für $G^*\left[V^*[C^* \setminus \{c^*\}]\right]$ umgewandelt wird. Sei $S^* = \{v_i^c \in V^* \mid v_i \in V \setminus \{v^*\}, \text{col}_{G-v^*}(v_i) = c\}$ eine Teilmenge

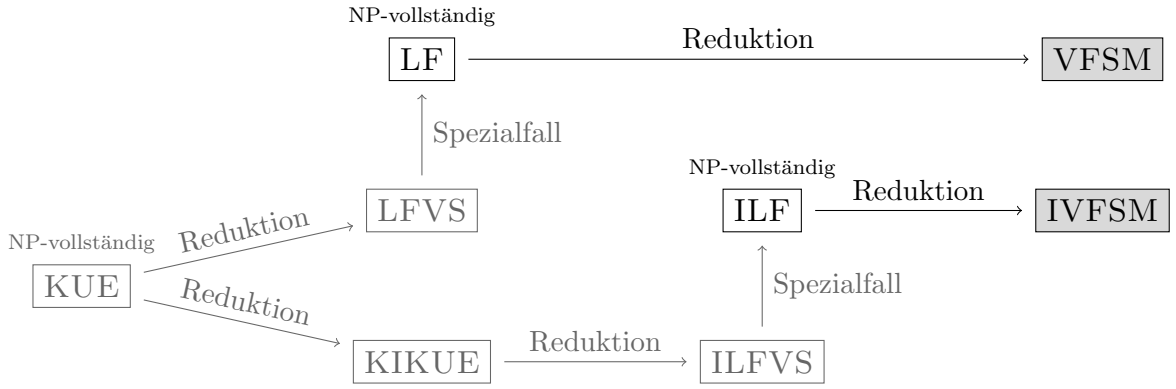


Abbildung 3.5: In dieser Abbildung wird dargestellt von welchen Problemen reduziert wird, um die NP-Vollständigkeit für VFSM und IVFSM zu zeigen.

von V^*, welche aus der Knotenmenge $V \setminus \{v^*\}$ und der Färbung col_{G-v^*} für $G - v^*$ konstruiert wird. Angenommen S^* ist keine stabile Menge für $G^* - v^*$, dann existiert eine Kante $\{v_i^{c_1}, v_j^{c_2}\} \in E^*$ mit $v_i^{c_1}, v_j^{c_2} \in S^*$. Da $\{v_i^{c_1}, v_j^{c_2}\} \in E^*$ gilt, folgt nach Konstruktion von G^*, dass $\{v_i, v_j\} \in E$ gilt. Des Weiteren gibt es nach Konstruktion von G^* nur Kanten zwischen Knoten $v_i^{c_k}, v_j^{c_l} \in V^*$ mit $c_k = c_l$, da G^* durch die Vereinigung der Kantenmengen und Knotenmengen aller nach Farben induzierten Subgraphen G_c^* von G konstruiert wurde. Somit gilt $c_1 = c_2$, woraus nach Konstruktion von S^* folgt, dass $col_G(v_i) = col_G(v_j)$ gilt. Dies ist ein Widerspruch dazu, dass col_G eine L-zulässige Färbung für G ist. Also ist S^* eine stabile Menge für $G^* - v^*$. Da nach Konstruktion $\text{col}_{G^*}(v_i^c) = i$ für alle Knoten $v_i^c \in V^*$ gilt und S^* für jeden Knoten $v_i \in V \setminus \{v^*\}$ mit $\text{col}_{G-v^*}(v_i) = c$ genau einen Knoten $v_i^c \in V^*$ enthält, folgt, dass $\text{col}_{G^*}(v_i^{c_1}) \neq \text{col}_{G^*}(v_i^{c_1})$ für alle Knoten $v_i^{c_1}, v_i^{c_1} \in S^*$ gilt. Damit ist S^* eine farbenreiche stabile Menge. Da $|V| = n$ und $C^* = \{1, \ldots, n\}$ gilt, folgt, dass $|S^*| = |V| - 1 = n - 1 = |C^*| - 1$ gilt. Damit ist S^* eine farbenreiche stabile Menge für $G^* - v^*$ der Größe $|C^*| - 1$, woraus folgt, dass $(G^* = (V^*, E^*), C^*, \text{col}_{G^*}, c^*, S^*)$ eine gültige Eingabe für IVFSM ist. Das Erstellen von S^* benötigt $O(n)$ Schritte, da für jeden Knoten aus $V \setminus \{v^*\}$ ein Knoten für S erzeugt wird. Da $R_{\text{LF}\to\text{VFSM}}$ ein polynomieller Algorithmus ist, kann eine Eingabe von ILF in polynomieller Zeit in eine gültige Eingabe von IVFSM umgewandelt werden.

Für die Reduktion von ILF auf IVFSM wurde, bis auf die Umwandlung des Knotens v^* und der Färbung col_{G-v^*} auf die Farbe $c^* \in C^*$ und die farbenreiche stabile Menge $S^* \subseteq V^*$ der Größe $|C^*| - 1$, nur $R_{\text{LF}\to\text{VFSM}}$ aus Lemma 3.2.1 benutzt. Entsprechend gibt es für G^* eine farbenreiche stabile Menge der Größe $|C^*|$, genau dann, wenn es für G eine L-zulässige Färbung col_G gibt. Damit wurde gezeigt, dass IVFSM NP-schwer ist, woraus folgt, dass IVFSM NP-vollständig ist. □

In Abbildung 3.5 ist dargestellt wie die Probleme, welche verwendet wurden um die NP-Vollständigkeit von VFSM und IVFSM zu zeigen, in Bezug zueinander stehen. Zuerst wurde das NP-vollständige Problem LF auf VFSM reduziert. Mit Hilfe dieser Reduktion wurde das NP-vollständige Problem ILF auf IVFSM reduziert.

4 Datenstrukturen für Intervallgraphen und Farbmengen

In diesem Kapitel werden Datenstrukturen für Intervallgraphen und Farbmengen vorgestellt, welche in den weiteren Kapiteln genutzt werden. Zusätzlich werden die Laufzeiten für Operationen auf diesen diskutiert.

Als Datenstruktur für Intervallgraphen wird ein *Intervallbaum* verwendet [CLR09]. Dieser benutzt einen ausbalancierten Rot-Schwarz-Baum als grundlegende Datenstruktur, welche dafür sorgt, dass Operationen wie das Einfügen oder Löschen eines Intervalls im Intervallbaum in $O(\log n)$ Schritten möglich sind. Jeder Knoten des Intervallbaums enthält ein Intervall, dessen Startzeitpunkt der Schlüssel für den Rot-Schwarz-Baum ist. Dies ermöglicht eine nach Startzeit sortierte Ausgabe aller Intervalle in $O(n)$ Schritten, indem der Baum traversiert wird. Dabei wird zuerst der linke Teilbaum, dann die Wurzel und schließlich der rechte Teilbaum durchlaufen. Als zusätzliche Information enthält jeder Knoten den letztmöglichen Intervallendpunkt im eigenen Teilbaum. Dies ermöglicht das Finden eines Intervalls in $O(\log n)$ Schritten und das Finden aller Intervalle, welche ein gegebenes Intervall schneiden, in $O(\min(n, k \log n))$ Schritten, wobei k die Anzahl der Intervalle ist, welche sich mit dem gegebenen Intervall schneiden. Da k durch die Anzahl der Farben beschränkt ist, werden für das Finden aller Intervalle, welche ein gegebenes Intervall schneiden, maximal $O(\min(n, |C| \log n))$ Schritte benötigt. Zur Vereinfachung der späteren Laufzeitanalysen wird für das Finden aller Intervalle, welche ein gegebenes Intervall schneiden, angenommen, dass dies in $O(n)$ Schritten passiert.

Als Datenstruktur für Farbmengen wird eine Bitdarstellung verwendet. Dabei wird jede Farbe durch ein Bit dargestellt. Die Farbe c aus der Farbmenge $C = \{0, 1, \ldots, k\}$ entspricht dem Bit an Bitposition c. Ist der Bit an Bitposition c null, so ist c kein Element der Farbmenge, ist es eins, so ist c ein Element der Farbmenge. Als Beispiel ist hier die Teilfarbmenge $\{2, 4, 5, 7\}$ der Farbmenge $\{0, 1, \ldots, 7\}$ angegeben:

$$00101101 = 180$$

Operationen wie das Einfügen einer Farbe oder Farbmenge in die Farbmenge und das Löschen einer Farbe oder Farbmenge aus der Farbmenge benötigen konstante Zeit. Des Weiteren benötigt die Vereinigung und der Schnitt zweier Farbmengen auch nur konstante Zeit. Ein weiterer Vorteil dieser Datenstruktur ist, dass so Farbmengen direkt als Index eines Arrays benutzt werden können. So entspricht jede Bitdarstellung sowohl einer Teilmenge der Farbmenge $\{0, 1, \ldots, 7\}$ als auch einem der Indizes von 0 bis 255.

5 Vorverarbeitung der Eingabe

In diesem Kapitel wird zuerst die kompakte Intervallgraphdarstellung eingeführt, welche grundlegend für die FPT-Algorithmen aus Kapitel 6 sind. Im Weiteren wird gezeigt, wie mit Hilfe von polynomiellen Datenreduktionsalgorithmen eine Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe mit bestimmten strukturellen Eigenschaften für Inkrementelle Listenfärbung oder Listenfärbung reduziert werden kann. Dabei basieren bestimmte Datenreduktionen auf Beobachtungen aus der Praxis.

5.1 Kompakte Intervallgraphen

Die „Kompaktheit“ von Intervallgraphen bezieht sich auf die Anzahl verschiedener Start- und Endpunkte der Intervalle, welche in der Intervalldarstellung eines Intervallgraphen benötigt werden. Kompakte Intervallgraphen wurden von van Bevern u. a. [Bev+15] eingeführt und sind wie folgt definiert:

Definition 5.1.1. Eine Intervalldarstellung ist *k-kompakt*, wenn der Start- und Endpunkt jedes Intervalls in $\{1, 2, \ldots, k\}$ liegt und die Intervalle aufsteigend nach ihren Startpunkten sortiert sind. Ein Intervallgraph G ist *k-kompakt*, wenn für G eine *k-kompakte* Intervalldarstellung existiert.

Des Weiteren wurde von van Bevern u. a. [Bev+15] gezeigt, dass jede Intervalldarstellung eines Intervallgraphen G in $O(n \log n)$ Schritten in eine k-kompakte Intervalldarstellung für G umgewandelt werden kann, sodass jeder Punkt aus $\{1, 2, \ldots, k\}$ mindestens einen Intervallstart- und einen Intervallendpunkt enthält und dass k die kleinste Zahl ist, sodass G k-kompakt ist.

Da die später erläuterten FPT-Algorithmen als Eingabe einen Graphen mit einer k-kompakten Intervalldarstellung für ein minimales k erwarten, ist mit Algorithmus 5.1 der Algorithmus angegeben, welcher für einen Intervallgraphen eine k-kompakte Intervalldarstellung für ein minimales k erzeugt. Der Algorithmus sortiert zuerst alle Intervallpunkte (Start- und Endpunkte) aufsteigend. Anschließend wird über alle Intervallpunkte iteriert und jedem Intervallpunkt wird die kleinste ganze Zahl zugewiesen, sodass sich alle Intervalle der Intervallpunkte, welche diese Zahl zugewiesen bekommen haben, im Intervallgraphen überschneiden. Da die Intervallpunkte aufsteigend sortiert sind, muss hierfür nur überprüft werden, ob auf einen Endpunkt ein Startpunkt folgt.

Algorithmus 5.1 k-kompakte Intervalldarstellung

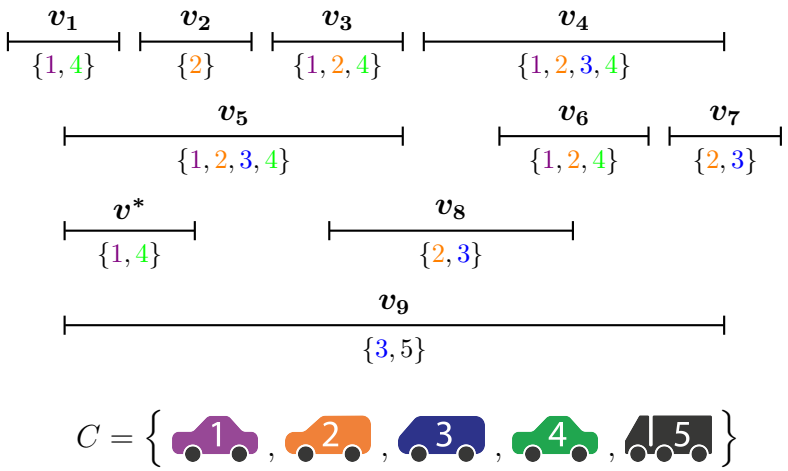
```
1: L ← sortiere alle Intervallpunkte (Start- und Endpunkte der Intervalle) aufsteigend,
       sodass Endpunkte vor gleichen Startpunkten sortiert sind
2: k ← 1
3: for i ← 1 to 2n do
4:    if i ≠ 1 and L[i − 1] ist Endpunkt and L[i] ist Startpunkt then
5:       k ← k + 1
6:    end if
7:    L[i] ← k
8: end for
```

5.2 Datenreduktion

In diesem Abschnitt werden Datenreduktionen für Listenfärbung und Inkrementelle Listenfärbung vorgestellt und die zugehörigen Algorithmen erläutert. Obwohl Datenreduktionen für Listenfärbung und Inkrementelle Listenfärbung gezeigt werden, wird davon ausgegangen, dass für die Datenreduktion eine gültige Eingabe für Inkrementelle Listenfärbung gegeben ist. Das Ergebnis dieser Reduktion kann dann entweder eine äquivalente Eingabe für das Problem Inkrementelle Listenfärbung oder für das Problem Listenfärbung sein. Da sowohl in LF als auch in ILF ein L-zulässige Färbung für den Eingabegraphen gesucht wird, löst ein Algorithmus für das Problem LF, auch das Problem ILF, wobei die Färbung des induzierten Subgraphen der Knotenmenge $V \setminus \{v^*\}$ ignoriert wird.

Anhand der Beispieleingabe aus Abbildung 5.1 werden in den folgenden Abschnitten die Datenreduktionen vorgestellt. Die Eingabe besteht aus einem Graphen $G = (V, E)$ mit zehn Knoten, einer Farbmenge C aus fünf Farben, einer zuweisbaren Farbliste $L(v)$ für jeden Knoten $v \in V$ und einer L-zulässigen Färbung col_{G-v^*}. Mit Hilfe der Datenreduktionen kann diese Eingabe auf eine äquivalente Eingabe mit bestimmten strukturellen Eigenschaften und nur fünf Knoten reduziert werden. Des Weiteren gibt die Spalte col_G in Abbildung 5.1 die Färbung der Knoten an, welche durch die Datenreduktion entfernt wurden, sodass eine Färbung für das reduzierte Problem mit dieser Teilfärbung eine Färbung für G bildet. Einige Datenreduktionen beruhen auf Beobachtungen aus dem Bereich des Corporate Car Sharings, daher sei hier nochmal erwähnt, dass jedes Intervall im Intervallgraphen einer Buchung entspricht und die Farben den Fahrzeugen entsprechen. Eine Farbliste eines Knotens ist somit eine Liste von zuweisbaren Fahrzeugen.

In Abschnitt 5.2.1 wird eine Datenreduktion vorgestellt, welche Knoten mit nur einer zuweisbaren Farbe entfernt. In der Praxis tritt dies bei Buchungen mit zusätzlichen Kriterien auf, welche nur von einem Fahrzeug erfüllt werden können. Die zweite Datenreduktion aus Abschnitt 5.2.2 entfernt Knoten, welchen eine lokal einzigartige Farbe zugewiesen werden kann. In der Praxis entspricht dies einer Buchung, welche ein zuweisbares Fahrzeug besitzt, das keiner sich überschneidenden Buchung zugewiesen werden kann. In der Datenreduktion, welche in Abschnitt 5.2.3 vorgestellt wird, werden



	col_{G-v^*}	col_G
v_1	1	-
v_2	2	-
v_3	1	-
v_4	4	-
v_5	4	-
v_6	1	-
v_7	2	-
v_8	2	-
v_9	3	-

Abbildung 5.1: Hier wird eine Beispieleingabe $(G, v^*, C, L, \text{col}_{G-v^*})$ für Inkrementelle Listenfärbung dargestellt. Dabei entspricht v^* dem neu hinzugefügten Knoten, C der Farbmenge und col_{G-v^*} der L-zulässigen Färbung für den Graphen $G - v^*$.Die Spalte col_G gibt die Färbung der Knoten an, welche durch die Datenreduktionen entfernt wurden.

zusammenhängende Knotenmengen mit sich überschneidenden Farbmengen entfernt. In der Praxis sind dies sich überschneidende Buchungen, welchen Fahrzeuge gleicher Fahrzeugklasse zugewiesen werden können. In Abschnitt 5.2.4 wird eine Datenreduktion vorgestellt, welche die Eingabe auf eine äquivalente Eingabe mit nur einer Zusammenhangskomponente beschränkt. Diese Datenreduktion ist besonders wichtig, da in der Praxis einzelne Buchungen existieren, welche über Nacht oder über das Wochenende gehen und somit die einzige Verbindung zwischen den Subgraphen sind. Sollte eines dieser Intervalle durch eine Datenreduktion entfernt werden können, so können direkt ganze Zusammenhangskomponenten aus der Eingabe entfernt werden. Abschließend wird in Abschnitt 5.2.5 beschrieben, welche strukturelle Eigenschaft die reduzierte Eingabe nach Anwendung von allen Datenreduktionen besitzt.

5.2.1 Eindeutige Färbbarkeit

In diesem Abschnitt wird anhand einer Reihe von Beobachtungen eine Datenreduktion vorgestellt, welche Knoten mit bestimmten Eigenschaften entfernt und so eine äquivalente Eingabe für Inkrementelle Listenfärbung oder Listenfärbung erzeugt. Zunächst wird definiert, was die eindeutige Färbbarkeit eines Knotens ist.

Sei ein Graph $G = (V, E)$ und für jeden Knoten $v \in V$ eine Liste $L(v)$ von zulässigen Farben gegeben. Ein Knoten $v \in V$ heißt *eindeutig färbbar*, wenn $|L(v)| = 1$ gilt, das heißt, es gibt für v nur eine zulässige Farbe. Die Farbe eines eindeutig färbbaren Knotens $v \in V$ wird *eindeutige Farbe* genannt. Sei der Knoten $v \in V$ eindeutig färbbar und sei c die eindeutige Farbe von v. Dann sei $L[v \to c] : V \setminus \{v\} \to C$ mit $L[v \to c](w) := L(w)$ für alle $w \in V \setminus (N_G(v) \cup \{v\})$ und $L[v \to c](w) := L(w) \setminus \{c\}$ für alle $w \in N_G(v)$ die

Zuweisung von zulässigen Farblisten bezüglich des eindeutig färbbaren Knotens v.

Mit Hilfe dieser Definitionen können die Beobachtungen für die Datenreduktion beschrieben werden. Zuerst wird gezeigt, dass es für eine Eingabe von Inkrementelle Listenfärbung mit einem eindeutig färbbaren Knoten $v \in V \setminus \{v^*\}$ eine $L[v \to c]$-zulässige Färbung für den induzierten Subgraphen $G\big[V \setminus \{v^*, v\}\big]$ gibt.

Beobachtung 5.2.1. *Sei $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung*, $v \in V \setminus \{v^*\}$ ein eindeutig färbbarer Knoten mit eindeutiger Farbe c und $L[v \to c]$ die Zuweisung von zulässigen Farblisten bezüglich des eindeutig färbbaren Knotens v. Dann gibt es eine $L[v \to c]$-zulässige Färbung für den induzierten Subgraphen $G\big[V \setminus \{v^*, v\}\big]$.*

Beweis. Sei $G' = G\big[V \setminus \{v^*, v\}\big]$ der induzierte Subgraph und c die eindeutige Farbe von v. Nach Definition eines eindeutig färbbaren Knotens gilt $|L(v)| = 1$, womit $\mathrm{col}_{G-v^*}(v) = c$ folgt. Da col_{G-v^*} eine L-zulässige Färbung ist, gibt es keinen Knoten $w \in N_{G-v^*}(v)$ mit $\mathrm{col}_{G-v^*}(w) = c$. Somit gilt nach Definition von $L[v \to c]$ für alle Knoten $w \in V(G-v^*)\setminus\{v\}$ $\mathrm{col}_{G-v^*}(w) \in L[v \to c](w)$, da nur die Farbe c aus allen Farblisten der Nachbarknoten von v entfernt wurde. Daher ist $\mathrm{col}_{G'}$ mit $\mathrm{col}_{G'}(w) := \mathrm{col}_{G-v^*}(w)$ für alle Knoten $w \in V \setminus \{v^*, v\}$ eine $L[v \to c]$-zulässige Färbung. □

Nun wird gezeigt, dass eine gültige Eingabe für Listenfärbung mit einem eindeutig färbbaren Knoten $v \in V$ auf eine äquivalente Eingabe für Listenfärbung reduziert werden kann.

Beobachtung 5.2.2. *Sei (G, C, L) eine gültige Eingabe für* Listenfärbung*. Falls ein eindeutig färbbarer Knoten $v \in V$ mit eindeutiger Farbe c existiert, so ist $\big(G\big[V \setminus \{v\}\big], C, L[v \to c]\big)$ mit $L[v \to c]$ als Zuweisung von zulässigen Farblisten bezüglich des eindeutig färbbaren Knotens v eine äquivalente Eingabe für* Listenfärbung*.*

Beweis. Sei (G, C, L) eine Ja-Instanz für Listenfärbung. Dann existiert eine L-zulässige Färbung col_G. Da v ein eindeutig färbbarer Knoten mit eindeutiger Farbe c ist, folgt $\mathrm{col}_G(v) = c$. Das heißt für alle Nachbarknoten $w \in N_G(v)$ gilt $\mathrm{col}_G(w) \neq c$. Nach Definition von $L[v \to c]$ gilt $L[v \to c](w) = L(w) \setminus \{c\}$ für alle Knoten $w \in N_G(v)$. Somit ist $\mathrm{col}_{G[V\setminus\{v\}]}$ mit $\mathrm{col}_{G[V\setminus\{v\}]}(w) := \mathrm{col}_G(w)$ für alle Knoten $w \in V \setminus \{v\}$ eine $L[v \to c]$-zulässige Färbung.

Sei $\big(G\big[V \setminus \{v\}\big], C, L[v \to c]\big)$ eine Ja-Instanz für Listenfärbung. Dann gibt es eine $L[v \to c]$-zulässige Färbung $\mathrm{col}_{G[V\setminus\{v\}]}$. Nach der Konstruktion von $L[v \to c]$ gilt für den eindeutig färbbaren Knoten v mit der eindeutigen Farbe $c \in C$ $\mathrm{col}_{G[V\setminus\{v\}]}(w) \neq c$ für alle $w \in N_G(v)$. Somit ist col_G mit $\mathrm{col}_G(w) := \mathrm{col}_{G[V\setminus\{v\}]}(w)$ für alle Knoten $w \in V \setminus \{v\}$ und $\mathrm{col}_G(v) = c$ eine L-zulässige Färbung von G.

Somit wurde gezeigt, dass $\big(G\big[V \setminus \{v\}\big], C, L[v \to c]\big)$ eine äquivalente Eingabe für Listenfärbung ist. □

Mit Hilfe von Beobachtung 5.2.1 und Beobachtung 5.2.2 kann gefolgert werden, dass die Datenreduktion auch für eine Eingabe von Inkrementelle Listenfärbung mit einem eindeutig färbbaren Knoten $v \in V \setminus \{v^*\}$ genutzt werden kann, was zum folgenden Korollar führt.

Korollar 5.2.3. *Sei* $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ *eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls im Graphen* G *ein eindeutig färbbarer Knoten* $v \in V \setminus \{v^*\}$ *mit eindeutiger Farbe* c *existiert, so ist* $(G', v^*, C, L[v \to c], \mathrm{col}_{G'-v^*})$ *mit* $G' = G\big[V \setminus \{v\}\big]$ *eine äquivalente Eingabe für* Inkrementelle Listenfärbung.

Der Fall, dass der Knoten v^* einer Eingabe von Inkrementelle Listenfärbung eindeutig färbbar ist, muss gesondert behandelt werden, da das Entfernen dieses Knotens aus der Eingabe für Inkrementelle Listenfärbung eine Eingabe für Listenfärbung erzeugt.

Beobachtung 5.2.4. *Sei* $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ *eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls der Knoten* v^* *eindeutig färbbar ist, so ist* $\big(G\big[V \setminus \{v^*\}\big], C, L[v^* \to c]\big)$ *mit* c *der eindeutigen Farbe von* v^* *eine äquivalente Eingabe für* Listenfärbung.

Beweis. Sei $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ eine Ja-Instanz für Inkrementelle Listenfärbung. Dann existiert eine L-zulässige Färbung col_G. Da v^* ein eindeutig färbbarer Knoten mit eindeutiger Farbe c ist, folgt $\mathrm{col}_G(v^*) = c$. Das heißt für alle Nachbarknoten $w \in N_G(v^*)$ gilt $\mathrm{col}_G(w) \neq c$. Nach Definition von $L[v^* \to c]$ gilt $L[v^* \to c](w) = L(w) \setminus \{c\}$ für alle Knoten $w \in N_G(v^*)$. Somit ist $\mathrm{col}_{G[V\setminus\{v^*\}]}$ mit $\mathrm{col}_{G[V\setminus\{v^*\}]}(w) := \mathrm{col}_G(w)$ für alle Knoten $w \in V \setminus \{v^*\}$ eine $L[v^* \to c]$-zulässige Färbung für $G\big[V \setminus \{v^*\}\big]$.

Sei $\big(G\big[V\setminus\{v^*\}\big], C, L[v^* \to c]\big)$ eine Ja-Instanz für Listenfärbung. Dann gibt es eine $L[v^* \to c]$-zulässige Färbung $\mathrm{col}_{G[V\setminus\{v^*\}]}$. Nach der Konstruktion von $L[v^* \to c]$ gilt für den eindeutig färbbaren Knoten v^* mit der eindeutigen Farbe $c \in C$ $\mathrm{col}_{G[V\setminus\{v^*\}]}(w) \neq c$ für alle $w \in N_G(v^*)$. Somit ist col_G mit $\mathrm{col}_G(v) := \mathrm{col}_{G[V\setminus\{v^*\}]}(v)$ für alle Knoten $v \in V \setminus \{v^*\}$ und $\mathrm{col}_G(v^*) = c$ eine L-zulässige Färbung von G.

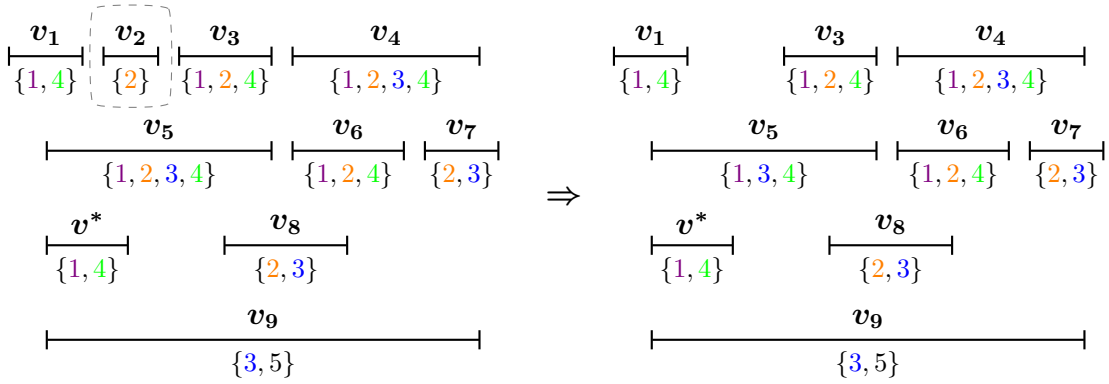
Somit wurde gezeigt, dass $\big(G\big[V \setminus \{v^*\}\big], C, L[v^* \to c]\big)$ eine äquivalente Eingabe für Listenfärbung ist. □

Für die in diesem Abschnitt beschriebene Datenreduktion gilt, dass jede Lösung $\mathrm{col}_{G[V\setminus\{v\}]}$ einer reduzierten Eingabe durch das Erweitern der Färbung $\mathrm{col}_{G[V\setminus\{v\}]}$ mit dem eindeutig färbbaren Knoten v und dessen eindeutiger Farbe c zu einer Lösung für die Originaleingabe wird. Falls $\mathrm{col}_{G[V\setminus\{v\}]}$ eine $L[v \to c]$-zulässige Färbung für $G[V \setminus \{v\}]$ ist, so ist die Färbung col_G mit $\mathrm{col}_G(w) := \mathrm{col}_{G[V\setminus\{v\}]}(w)$ für alle Knoten $w \in V \setminus \{v\}$ und $\mathrm{col}_G(v) := c$, nach Definition von $L[v \to c]$, eine L-zulässige Färbung für G.

Falls der Knoten $v^* \in V$ eindeutig färbbar ist und zusätzlich alle Nachbarknoten durch col_{G-v^*} mit einer anderen Farbe als der eindeutigen Farbe von v^* gefärbt werden, so gibt es eine L-zulässige Färbung für den Graphen G.

Beobachtung 5.2.5. *Sei* $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ *eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls* v^* *eindeutig färbbar mit eindeutiger Farbe* c *ist und* $\mathrm{col}_{G-v^*}(w) \neq c$ *für alle Nachbarknoten* $w \in N_G(v^*)$ *gilt, so gibt es eine* L*-zulässige Färbung* col_G.

Beweis. Da $\mathrm{col}_{G-v^*}(w) \neq c$ für alle Nachbarknoten $w \in N_G(v^*)$ gilt, ist col_G mit $\mathrm{col}_G(v) = \mathrm{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus \{v^*\}$ und $\mathrm{col}_G(v^*) = c$ eine L-zulässige Färbung. □



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
col_{G-v^*}	1	2	1	4	4	1	2	2	3
col_G	-	2	-	-	-	-	-	-	-

Abbildung 5.2: Hier wird die Datenreduktion durch eindeutig färbbare Knoten dargestellt. Der Knoten v_2 ist eindeutig färbbar mit der eindeutigen Farbe 2. Nach der Reduktion wurde v_2 aus dem Graphen entfernt, die Farbe aus den zuweisbaren Farblisten der Nachbarknoten entfernt und $\text{col}_G(v_2) = 2$ gesetzt.

In Abbildung 5.2 ist die Datenreduktion anhand des Beispiels dargestellt. In der Praxis entspricht der eindeutig färbbare Knoten v_2 mit der eindeutigen Farbe 2 einer Buchung, welcher aufgrund bestimmter Buchungskriterien nur das Fahrzeug 2 zugewiesen werden kann. Nach der Reduktion wurde der Knoten v_2 aus dem Graphen entfernt, die Farbe 2 aus den zuweisbaren Farblisten der Nachbarknoten von v_2 entfernt und $\text{col}_G(v_2) = 2$ gesetzt.

Abschließend wird der Algorithmus 5.2 für die Datenreduktion vorgestellt. Dieser beschränkt sich auf eine Eingabe für Inkrementelle Listenfärbung und auf eindeutig färbbare Knoten $v \in V \setminus \{v^*\}$. Die Algorithmen, welche v^* berücksichtigen oder als Eingabe eine Eingabe für Listenfärbung erwarten, laufen analog zu dem hier präsentierten Algorithmus ab. In dem Algorithmus entspricht der Graph G^* dem Originalgraphen vor der Datenreduktion. Der Graph G ist ein induzierter Subgraph des Graphen G^*. Der Algorithmus entfernt iterativ alle Knoten $v \in V \setminus \{v^*\}$ mit $|L(v)| = 1$. In jedem Schritt wird zuerst der Färbung col_{G^*} die eindeutige Farbe c des eindeutig färbbaren Knotens v zugewiesen, sodass $\text{col}_{G^*}(v) = c$ gilt. Anschließend wird die Farbe c aus allen zulässigen Farblisten $L(w)$ der Nachbarknoten $w \in N_G(v)$ entfernt und überprüft, ob dadurch neue eindeutig färbbare Knoten entstehen. Abschließend wird G auf den induzierten Subgraphen $G[V \setminus \{v\}]$ reduziert, was die Grundlage für die nächste Iteration bildet.

Um die Laufzeit der Datenreduktion zu berechnen werden hier alle Laufzeiten der einzelnen Schritte erläutert. Das initiale Finden der Knotenmenge $\{v \in V \setminus \{v^*\} \mid |L(v)| = 1\}$ benötigt $O(n)$ Schritte, da jeder Knoten einmal überprüft werden muss. Die

Algorithmus 5.2 *eindeutigeFärbbarkeit*$(G, L, \mathrm{col}_{G^*}, v^*, \mathrm{col}_{G^*-v^*})$

```
1: W ← erstelle eine FIFO-Liste („First In - First Out“) mit den Methoden W.poll(),
       und W.add(c), dabei entfernt W.poll() das erste Element aus der Liste und
       gibt es zurück und W.add(v) fügt v an das Ende der Liste
2: W ← füge zu W alle Knoten der Menge {v ∈ V \ {v*} | |L(v)| = 1} hinzu
3: while W nicht leer do
4:    v ← W.poll()
5:    c ← col_{G−v*}(v)
6:    col_{G*}(v) ← c
7:    for all w ∈ N_G(v) do
8:       if c ∈ L(w) then
9:          L(w) ← L(w) \ {c}
10:         if w ≠ v* und |L(w)| = 1 then
11:            W.add(w)
12:         end if
13:      end if
14:   end for
15:   G ← G[V \ {v}]
16: end while
17: return (G, L, col_{G*})
```

Anzahl der Iterationen der While-Schleife ist durch die Anzahl der Knoten beschränkt, da in jeder Iteration ein Knoten aus G entfernt wird. Eine Iteration der While-Schleife benötigt $O(n + \log n)$ Schritte. Dies setzt sich zusammen aus $O(n)$ Schritten für das Finden aller Nachbarknoten von v und aus $O(\log n)$ Schritten für das Entfernen von v aus G. Das Entfernen der Farbe c aus allen zulässigen Farblisten der benachbarten Knoten von v spielt keine Rolle, da aufgrund der Datenstruktur für Farbmengen, welche in Kapitel 4 beschrieben wurde, diese Operation in konstanter Zeit ausführbar ist. Die Gesamtlaufzeit setzt sich somit wie folgt zusammen:

$$O\Big(\underbrace{n}_{\substack{\text{Finden von eindeutig} \\ \text{färbbaren Knoten}}} + \underbrace{n}_{|V|} \cdot \big(\underbrace{n + \log n}_{\substack{\text{Iteration} \\ \text{While-Schleife}}} \big) \Big) = O(n^2)$$

5.2.2 Lokal einzigartige Farben

In diesem Abschnitt wird anhand einer Reihe von Beobachtungen eine Datenreduktion vorgestellt, welche Knoten mit einer bestimmten lokalen Eigenschaft entfernt und so eine äquivalente Eingabe für Inkrementelle Listenfärbung oder Listenfärbung erzeugt. Zunächst wird definiert, was ein lokal einzigartig färbbarer Knoten ist.

Sei ein Graph $G = (V, E)$ und für jeden Knoten $v \in V$ eine Liste $L(v)$ von zulässigen Farben gegeben. Ein Knoten $v \in V$ ist *lokal einzigartig färbbar*, wenn eine Farbe $c \in L(v)$ existiert, sodass $c \notin L(w)$ für alle benachbarten Knoten $w \in N_G(v)$ gilt. Die Farben eines

lokal einzigartig färbbaren Knotens $v \in V$, welche in keiner Farbliste eines Nachbarknotens von v vorkommen, werden *lokal einzigartige Farben* genannt.

Mit Hilfe dieser Definitionen können die Beobachtungen für die Datenreduktion beschrieben werden. Zuerst wird gezeigt, dass es für eine Eingabe von Inkrementelle Listenfärbung mit einem lokal einzigartig färbbaren Knoten $v \in V \setminus \{v^*\}$ eine L-zulässige Färbung $\text{col}^*_{G-v^*}$ mit $\text{col}^*_{G-v^*}(v) = c$ gibt.

Beobachtung 5.2.6. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls es in G ein lokal einzigartig färbbaren Knoten $v \in V \setminus \{v^*\}$ mit lokal einzigartiger Farbe $c \in L(v)$ gibt, so gibt es eine L-zulässige Färbung $\text{col}^*_{G-v^*}$ mit $\text{col}^*_{G-v^*}(v) = c$.*

Beweis. Sei v der lokal einzigartig färbbarer Knoten mit lokal einzigartiger Farbe $c \in L(v)$. Nach der Definition eines lokal einzigartig färbbaren Knotens mit lokal einzigartiger Farbe $c \in L(v)$ gilt $c \notin L(w)$ für alle benachbarten Knoten $w \in N_G(v)$. Somit gilt $c \neq \text{col}_{G-v^*}(w)$ für alle benachbarten Knoten $w \in N_G(v)$. Damit ist $\text{col}^*_{G-v^*}$ mit $\text{col}^*_{G-v^*}(w) := \text{col}_{G-v^*}(w)$ für alle $w \in V \setminus \{v, v^*\}$ und $\text{col}^*_{G-v^*}(v) := c$ eine L-zulässige Färbung für $G[V \setminus \{v^*\}]$. □

Für den Fall, dass der Knoten v^* einer Eingabe von Inkrementelle Listenfärbung lokal einzigartig färbbar ist, kann gezeigt werden, dass eine L-zulässige Färbung col_G existiert.

Beobachtung 5.2.7. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Wenn der Knoten v^* ein lokal einzigartig färbbarer Knoten ist, dann gibt es eine L-zulässige Färbung col_G.*

Beweis. Da v^* ein lokal einzigartig färbbarer Knoten ist, folgt nach der Definition eines lokal einzigartig färbbaren Knotens, dass es eine Farbe $c \in L(v^*)$ gibt, sodass $c \notin L(w)$ für alle benachbarten Knoten $w \in N_G(v^*)$ gilt. Das heißt für die L-zulässige Färbung col_{G-v^*} gilt $c \neq \text{col}_{G-v^*}(w)$ für alle Knoten $w \in N_G(v^*)$. Somit ist die Färbung col_G mit $\text{col}_G(v) := \text{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus \{v^*\}$ und $\text{col}_G(v^*) := c$ eine L-zulässige Färbung. □

Nun wird gezeigt, dass eine gültige Eingabe für Listenfärbung mit einem lokal einzigartig färbbaren Knoten $v \in V$ auf eine äquivalente Eingabe für Listenfärbung reduziert werden kann.

Beobachtung 5.2.8. *Sei (G, C, L) eine gültige Eingabe für* Listenfärbung. *Falls ein lokal einzigartig färbbarer Knoten $v \in V$ existiert, so ist $\big(G[V \setminus \{v\}], C, L[V \setminus \{v\}]\big)$ eine äquivalente Eingabe für* Listenfärbung.

Beweis. Sei (G, C, L) eine Ja-Instanz für Listenfärbung mit einem lokal einzigartig färbbaren Knoten $v \in V$. Dann gibt es eine L-zulässige Färbung col_G. Insbesondere gibt es für jeden induzierten Graphen $G[V']$ mit $V' \subseteq V$ eine $L[V']$-zulässige Färbung. Daraus folgt, dass $\text{col}_{G[V \setminus \{v\}]}$ mit $\text{col}_{G[V \setminus \{v\}]}(w) := \text{col}_G(w)$ für alle Knoten $w \in V \setminus \{v\}$ eine $L[V \setminus \{v\}]$-zulässige Färbung ist.

Sei (G, C, L) eine Nein-Instanz. Dann existiert keine L-zulässige Färbung col_G. Angenommen es gibt eine $L[V \setminus \{v\}]$-zulässige Färbung $\text{col}_{G[V\setminus\{v\}]}$ für den induzierten Subgraphen $G\big[V \setminus \{v\}\big]$. Da v ein lokal einzigartig färbbarer Knoten ist, gibt es nach Definition von lokal einzigartig färbbaren Knoten eine Farbe $c \in L(v)$, sodass $c \notin L(w)$ für alle benachbarten Knoten $w \in N_G(v)$ gilt. Somit gilt $c \neq \text{col}_{G[V\setminus\{v\}]}(w)$ für alle Knoten $w \in N_G(v)$. Daraus folgt, dass col_G mit $\text{col}_G(w) := \text{col}_{G[V\setminus\{v\}]}(w)$ für alle Knoten $w \in V \setminus \{v\}$ und $\text{col}_G(v) := c$ eine L-zulässige Färbung für G ist. Dies ist ein Widerspruch dazu, dass (G, C, L) eine Nein-Instanz ist, also existiert keine $L[V \setminus \{v\}]$-zulässige Färbung $\text{col}_{G[V\setminus\{v\}]}$, woraus folgt, dass $\big(G\big[V \setminus \{v\}\big], C, L[V \setminus \{v\}]\big)$ ebenfalls eine Nein-Instanz ist.

Somit wurde gezeigt, dass $\big(G\big[V \setminus \{v\}\big], C, L\big[V \setminus \{v\}\big]\big)$ eine äquivalente Eingabe für Listenfärbung ist. □

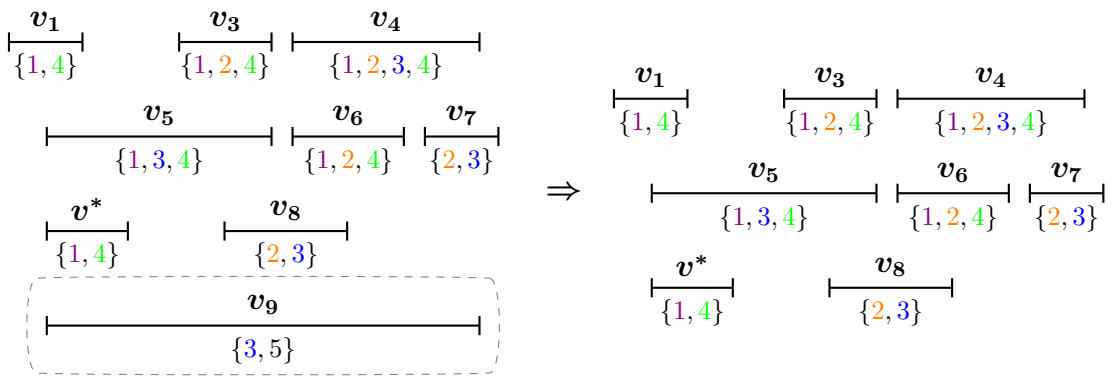
Mit Hilfe von Beobachtung 5.2.6 und Beobachtung 5.2.8 kann gefolgert werden, dass die Datenreduktion auch für eine Eingabe von Inkrementelle Listenfärbung mit einem lokal einzigartig färbbaren Knoten $v \in V \setminus \{v^*\}$ genutzt werden kann, was zum folgenden Korollar führt.

Korollar 5.2.9. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls ein lokal einzigartig färbbarer Knoten $v \in V \setminus \{v^*\}$ existiert, so ist die Eingabe $\big(G', v^*, C, L', \text{col}_{G'-v^*}\big)$ mit $G' = G\big[V \setminus \{v\}\big]$ und $L' = L[V \setminus \{v\}]$ eine äquivalente Eingabe für* Inkrementelle Listenfärbung.

Für die in diesem Abschnitt beschriebene Datenreduktion gilt, dass jede Lösung $\text{col}_{G[V\setminus\{v\}]}$ einer reduzierten Eingabe durch das Erweitern der Färbung $\text{col}_{G[V\setminus\{v\}]}$ mit dem lokal einzigartig färbbaren Knoten v und dessen lokal einzigartiger Farbe c zu einer Lösung für die eigentliche Eingabe wird. Falls $\text{col}_{G[V\setminus\{v\}]}$ eine $L[V \setminus \{v\}]$-zulässige Färbung für $G[V \setminus \{v\}]$ ist, so ist die Färbung col_G mit $\text{col}_G(w) := \text{col}_{G[V\setminus\{v\}]}(w)$ für alle Knoten $w \in V \setminus \{v\}$ und $\text{col}_G(v) := c$ nach Definition eines lokal einzigartig färbbaren Knotens eine L-zulässige Färbung für G.

In Abbildung 5.3 ist die Datenreduktion anhand des Beispiels dargestellt. In der Praxis entspricht der lokal einzigartig färbbare Knoten v_9 mit der lokal einzigartigen Farbe 5 einer Buchung, welche das zuweisbare Fahrzeug 5 besitzt, welches keiner sich überschneidenden Buchung zugewiesen werden kann. Nach der Reduktion wurde der Knoten v_9 aus dem Graphen entfernt und $\text{col}_G(v_9) = 5$ gesetzt.

Abschließend wird der Algorithmus 5.3 für die Datenreduktion vorgestellt. Im Algorithmus entspricht der Graph G^* dem Originalgraphen vor der Datenreduktion und der Graph G einem induzierten Subgraphen des Graphen G^*. Der Algorithmus startet, indem er alle Knoten des Eingabegraphen in einer Knotenmenge V' abspeichert. Anschließend wird so lange über die Knotenmenge V' iteriert, bis diese kein Knoten mehr enthält. In jeder Iteration wird ein Knoten v aus V' entfernt und anschließend geschaut, ob dieser lokal einzigartig färbbar ist. Dies geschieht, indem das relative Komplement F der Farbliste $L(v)$ und der Vereinigung aller Farblisten der Nachbarknoten von v berechnet wird. Ist F die leere Menge, so wird mit der nächsten Iteration weiter gemacht. Ist F



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
col_{G-v^*}	1	2	1	4	4	1	2	2	3
col_G	-	2	-	-	-	-	-	-	5

Abbildung 5.3: Hier wird die Datenreduktion durch lokal einzigartig färbbare Knoten dargestellt. Der Knoten v_9 ist lokal einzigartig färbbar mit der lokal einzigartigen Farbe 5. Nach der Reduktion wurde v_9 aus dem Graphen entfernt und $\text{col}_G(v_9) = 5$ gesetzt.

nicht leer, dann ist der Knoten v lokal einzigartig färbbar und besitzt $|F|$ viele lokal einzigartige Farben. Dann wird der Färbung col_{G^*} für v eine der lokal einzigartigen Farben aus F zugewiesen. Abschließend wird G auf den induzierten Subgraphen $G[V \setminus \{v\}]$ reduziert und V' wird um alle Nachbarknoten von v ergänzt, da diese für den neuen Graphen wieder lokal einzigartig färbbar sein könnten.

Um die Laufzeit der Datenreduktion zu berechnen, werden hier alle Laufzeiten der einzelnen Schritte erläutert. Die Repeat-Schleife ist durch $O(n^2)$ beschränkt, da für jeden aus V' entfernten Knoten $O(n)$ viele Knoten zu V' hinzugefügt werden können. Im Folgenden wird der Aufwand einer Iteration erläutert. Das Entfernen von v aus V' benötigt $O(\log n)$ Schritten. Durch die Datenstrukturen für den Intervallgraphen und die Farbmenge, welche in Kapitel 4 eingeführt wurden, werden für das Finden aller Nachbarknoten von v maximal $O(n)$ Schritte benötigt und aller lokal einzigartigen Farben von v maximal $O(n)$ Schritte benötigt. Das Entfernen des Knotens v aus G benötigt ebenfalls durch den Intervallbaum als Datenstruktur nur $O(\log n)$ Schritte. Die Vereinigung der Knotenmenge V' und der Nachbarknoten N benötigt $O(n \log n)$ Schritte, da überprüft werden muss, ob die Nachbarknoten schon in V' enthalten sind. Die Gesamtlaufzeit setzt sich somit wie folgt zusammen:

$$O\Big(\underbrace{n^2}_{\substack{\text{äußere}\\ \text{Schleife}}} \cdot \Big(\underbrace{\log n}_{\substack{v \text{ aus } V' \\ \text{entfernen}}} + \underbrace{n}_{\substack{\text{Finden von}\\ \text{Nachbarknoten}}} + \underbrace{n}_{\substack{\text{Finden von lokal}\\ \text{einzigartigen Farben}}} + \underbrace{\log n}_{\substack{v \text{ aus } G \\ \text{entfernen}}} + \underbrace{n \log n}_{\substack{\text{Vereinigung}\\ \text{von } V' \text{ und } N}} \Big) \Big) = O(n^3 \log n)$$

Algorithmus 5.3 *lokalEinzigartigeFarben*(G, col_{G^*})

```
1:  V' ← V(G)
2:  repeat
3:    v ← beliebiger Knoten aus V'
4:    V' ← V' \ {v}
5:    N ← N_G(v)
6:    F ← L(v) \ ⋃_{w∈N} L(w)
7:    if F ≠ ∅ then
8:      col_{G*}(v) ← c mit der ersten Farbe c ∈ F
9:      G ← G[V(G) \ {v}]
10:     V' ← V' ∪ N
11:   end if
12: until V' = ∅
13: return (G, col_{G*})
```

5.2.3 Pseudofahrzeugklassen

In diesem Abschnitt wird anhand einer Reihe von Beobachtungen eine Datenreduktion vorgestellt, welche spezielle zusammenhängende Knotenmengen mit sich überschneidenden Farbmengen entfernt und so eine äquivalente Eingabe für Inkrementelle Listenfärbung oder Listenfärbung erzeugt. Zunächst wird definiert, was eine Pseudofahrzeugklasse für eine Knotenmenge $V' \subseteq V$ ist.

Definition 5.2.10. Eine Menge von Farben $F \subseteq C$ wird *Pseudofahrzeugklasse für die Knotenmenge* $V' \subseteq V$ genannt, wenn für F und V' folgende drei Eigenschaften gelten:

1. Für alle Knoten $v \in V'$ gilt $F \subseteq L(v)$.

2. Der induzierte Subgraph $G[V']$ ist zusammenhängend.

3. Keine Farbe aus F ist in einer Liste der zulässigen Farben der benachbarten Knoten der Knotenmenge V' enthalten, das heißt für jeden Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus V'$ und jede Farbe $c \in F$ gilt $c \notin L(w)$.

Des Weiteren beschränkt sich diese Datenreduktion auf Intervallgraphen. Da jeder Intervallgraph ein perfekter Graph ist und somit die chromatische Zahl gleich der Cliquenzahl ist, kann für die Reduktion folgendes Korollar verwendet werden:

Korollar 5.2.11. *Sei G ein Intervallgraph, dann existiert für G eine zulässige Färbung* col_G *mit der Farbmenge* $\{1, \ldots, \omega(G)\}$.

Mit Hilfe der Definition für Pseudofahrzeugklassen und dem Korollar 5.2.11 können die Beobachtungen für die Datenreduktion beschrieben und bewiesen werden. Zuerst wird folgende Beobachtung gezeigt:

Beobachtung 5.2.12. *Sei $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ mit Intervallgraph G eine gültige Eingabe für* Inkrementelle Listenfärbung *und F eine Pseudofahrzeugklasse für die Knotenmenge $V' \subseteq V$. Falls $\omega\big(G[V']\big) \leq |F|$ gilt, so gibt es entweder für:*

1. *$v^* \notin V'$ eine L-zulässige Färbung $\mathrm{col}^*_{G-v^*}$ mit $\mathrm{col}^*_{G-v^*}(v) := \mathrm{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus (V' \cup \{v^*\})$ und $\mathrm{col}^*_{G-v^*}(v) \in F$ für alle Knoten $v \in V'$ oder*
2. *$v^* \in V'$ eine L-zulässige Färbung col_G.*

Beweis. Sei F eine Pseudofahrzeugklasse für die Knotenmenge $V' \subseteq V$, sodass $\omega\big(G[V']\big) \leq |F|$ gilt. Aus Korollar 5.2.11 folgt, dass es für den induzierten Subgraphen $G[V']$ eine L-zulässige Färbung $\mathrm{col}_{G[V']}$ gibt, für die $\mathrm{col}_{G[V']}(v) \in F$ für alle Knoten $v \in V'$ gilt. Nach der Definition einer Pseudofahrzeugklasse gilt für jeden Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus V'$ und jede Farbe $c \in F$ $c \notin L(w)$.

Falls $v^* \notin V'$ gilt, folgt, dass $\mathrm{col}_{G-v^*}(w) \notin F$ für alle Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus \big\{V' \cup \{v^*\}\big\}$ gilt. Somit ist die Färbung $\mathrm{col}^*_{G-v^*}$ mit $\mathrm{col}^*_{G-v^*}(v) := \mathrm{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus (V' \cup \{v^*\})$ und $\mathrm{col}^*_{G-v^*}(v) := \mathrm{col}_{G[V']}(v)$ für alle Knoten $v \in V'$ eine L-zulässige Färbung für den Graphen $G - v^*$.

Falls $v^* \in V'$ gilt, folgt, dass $\mathrm{col}_{G-v^*}(w) \notin F$ für alle Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus V'$ gilt. Somit ist die Färbung col_G mit $\mathrm{col}_G(v) := \mathrm{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus V'$ und $\mathrm{col}_G(v) := \mathrm{col}_{G[V']}(v)$ für alle Knoten $v \in V'$ eine L-zulässige Färbung für den Graphen G. □

Nun wird gezeigt, dass eine gültige Eingabe für Listenfärbung mit einer Pseudofahrzeugklasse F für eine Knotenmenge $V' \subseteq V$ auf eine äquivalente Eingabe für Listenfärbung reduziert werden kann, wenn $\omega\big(G[V']\big) \leq |F|$ gilt.

Beobachtung 5.2.13. *Sei (G, C, L) mit Intervallgraph G eine gültige Eingabe für* Listenfärbung *und F eine Pseudofahrzeugklasse für die Knotenmenge $V' \subseteq V$. Falls $\omega\big(G[V']\big) \leq |F|$ gilt, so ist $\big(G\big[V \setminus V'\big], C, L\big[V \setminus V'\big]\big)$ eine äquivalente Eingabe für* Listenfärbung.

Beweis. Sei (G, C, L) mit Intervallgraphen G eine Ja-Instanz für Listenfärbung mit einer Pseudofahrzeugklasse F für die Knotenmenge $V' \subseteq V$, sodass $\omega\big(G[V']\big) \leq |F|$ gilt. Dann gibt es eine L-zulässige Färbung col_G. Insbesondere gibt es für jeden induzierten Graph $G[W]$ mit $W \subseteq V$ eine $L[W]$-zulässige Färbung, woraus folgt, dass $\mathrm{col}_{G[V \setminus V']}$ mit $\mathrm{col}_{G[V \setminus V']}(v) = \mathrm{col}_G(v)$ für alle Knoten $v \in V \setminus V'$ eine $L[V \setminus V']$-zulässige Färbung ist.

Sei (G, C, L) eine Nein-Instanz. Dann existiert keine L-zulässige Färbung col_G. Angenommen es gibt eine $L[V \setminus V']$-zulässige Färbung $\mathrm{col}_{G[V \setminus V']}$ für den induzierten Subgraphen $G\big[V \setminus V'\big]$. Da F eine Pseudofahrzeugklasse für die Knotenmenge $V' \subseteq V$ ist und $\omega\big(G[V']\big) \leq |F|$ gilt, folgt aus Korollar 5.2.11, dass es für den induzierten Subgraphen $G[V']$ eine L-zulässige Färbung $\mathrm{col}_{G[V']}$ gibt, für die $\mathrm{col}_{G[V']}(v) \in F$ für alle Knoten $v \in V'$ gilt. Nach der Definition einer Pseudofahrzeugklasse gilt für jeden Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus V'$ und jede Farbe $c \in F$ $c \notin L(w)$. Daher folgt, dass $\mathrm{col}_{G[V \setminus V']}(w) \notin F$ für alle Knoten $w \in \bigcup_{v \in V'} N_G(v) \setminus V'$ gilt. Somit ist col_G

mit $\text{col}_G(v) := \text{col}_{G[V \setminus V']}(v)$ für alle Knoten $v \in V \setminus V'$ und $\text{col}_G(v) := \text{col}_{G[V']}(v)$ für alle Knoten $v \in V'$ eine L-zulässige Färbung für G. Dies ist ein Widerspruch dazu, dass (G, C, L) eine Nein-Instanz ist, also existiert keine L-zulässige Färbung $\text{col}_{G[V \setminus V']}$, woraus folgt, dass $\big(G\big[V \setminus V'\big], C, L\big[V \setminus V'\big]\big)$ ebenfalls eine Nein-Instanz ist.

Somit wurde gezeigt, dass $\big(G\big[V \setminus V'\big], C, L\big[V \setminus V'\big]\big)$ eine äquivalente Eingabe für Listenfärbung ist. □

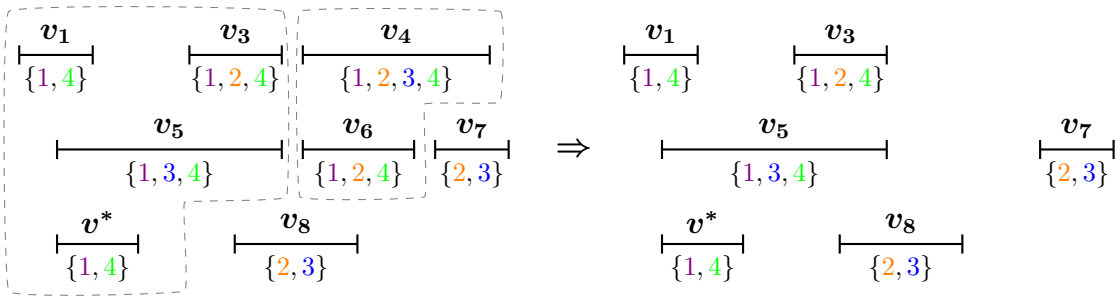
Mit Hilfe von Beobachtung 5.2.12 und Beobachtung 5.2.13 kann gefolgert werden, dass die Datenreduktion auch für eine gültige Eingabe von Inkrementelle Listenfärbung mit einer Pseudofahrzeugklasse F für eine Knotenmenge $V' \subseteq V \setminus \{v^*\}$ genutzt werden kann, was zu folgendem Korollar führt.

Korollar 5.2.14. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ mit einem Intervallgraphen G eine gültige Eingabe für* Inkrementelle Listenfärbung *und sei F eine Pseudofahrzeugklasse für die Knotenmenge $V' \subseteq V \setminus \{v^*\}$. Falls $\omega\big(G[V']\big) \leq |F|$ gilt, so ist $\big(G', v^*, C, L', \text{col}_{G'-v^*}\big)$ mit $G' = G\big[V \setminus V'\big]$ und $L' = L\big[V \setminus V'\big]$ eine äquivalente Eingabe für* Inkrementelle Listenfärbung.

Für die in diesem Abschnitt beschriebene Datenreduktion gilt, dass jede Lösung $\text{col}_{G[V \setminus V']}$ einer reduzierten Eingabe durch das Erweitern der Färbung $\text{col}_{G[V \setminus V']}$ mit der Färbung der Knotenmenge V' durch die Pseudofahrzeugklasse F zu einer Lösung für die eigentliche Eingabe wird. Falls $\text{col}_{G[V \setminus V']}$ eine $L[V \setminus V']$-zulässige Färbung für $G[V \setminus V']$ ist und $\text{col}_{G[V']}$ die $L[V']$-zulässige Färbung der Knotenmenge V' durch die Pseudofahrzeugklasse F ist, so ist die Färbung col_G mit $\text{col}_G(v) := \text{col}_{G[V \setminus V']}(v)$ für alle Knoten $v \in V \setminus V'$ und $\text{col}_G(v) := \text{col}_{G[V']}(v)$ für alle Knoten $v \in V'$ eine L-zulässige Färbung für G.

In Abbildung 5.4 ist die Datenreduktion anhand des Beispiels dargestellt. In der Praxis entsprechen Pseudofahrzeugklassen Fahrzeugen mit gleichen Eigenschaften, welche sich überschneidenden Buchungen zugewiesen werden können. Im Beispiel gibt es zwei Knotenmengen $\{v_1, v_3, v_5, v^*\}$ und $\{v_4, v_6\}$, deren induzierte Subgraphen zusammenhängend sind und dessen Knoten alle die Farben 1 und 4 in ihrer jeweiligen Farbliste enthalten. Des Weiteren kommt keine dieser Farben in einem der Nachbarknoten der jeweiligen Knotenmengen vor. Damit bilden die Fahrzeuge 1 und 4 sowohl für die Menge $\{v_1, v_3, v_5, v^*\}$ als auch für die Menge $\{v_4, v_6\}$ eine Pseudofahrzeugklasse. Da für die Reduktion die maximale Clique der induzierten Subgraphen von $\{v_1, v_3, v_5, v^*\}$ und $\{v_4, v_6\}$ kleiner gleich der Anzahl der Farben der Pseudofahrzeugklasse sein muss, kann die Reduktion nur auf die Knotenmenge $\{v_4, v_6\}$ angewendet werden, da $\{v_1, v_5, v^*\}$ eine Clique der Größe 3 bildet. Durch die Reduktion werden die Knoten v_4 und v_6 aus dem Graph entfernt und $\text{col}_G(v_4) = 1$ und $\text{col}_G(v_6) = 4$ gesetzt.

Abschließend wird der Algorithmus für die Datenreduktion vorgestellt. Dieser ist in zwei Algorithmen aufgeteilt. Der Algorithmus 5.4 überprüft, ob die Anzahl der Farben in der Pseudofahrzeugklasse F' für die Knotenmenge V' größer oder gleich der Cliquenzahl des induzierten Subgraphen $G[V']$ ist. Falls dies der Fall ist, färbt der Algorithmus alle Knoten der Knotenmenge V' mit Farben aus F'. Um die Cliquenzahl zu berechnen, werden zuerst alle Intervallpunkte, also Start- und Endpunkte, aufsteigend sortiert. Mit



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
col_{G-v^*}	1	2	1	4	4	1	2	2	3
col_G	-	2	-	1	-	4	-	-	5

Abbildung 5.4: Hier wird die Datenreduktion durch Pseudofahrzeugklassen dargestellt. Die Farbmenge $\{1,4\}$ bildet eine Pseudofahrzeugklasse für die Knotenmengen $\{v_1, v_3, v_5, v^*\}$ und $\{v_4, v_6\}$. Aufgrund der Größe der maximalen Cliquen in den induzierten Graphen der Knotenmengen, kann die Reduktion nur auf die Knotenmenge $\{v_4, v_6\}$ angewendet werden. Dabei werden die Knoten aus dem Graphen entfernt und $\text{col}_G(v_4) = 1$ und $\text{col}_G(v_6) = 4$ gesetzt.

Hilfe dieser sortierten Liste kann die Größe der maximalen Clique c_{max} berechnet werden. Dabei wird über die sortierte Liste iteriert und in jeder Iteration wird die Größe der aktuellen Clique c aktualisiert. Falls der aktuelle Intervallpunkt ein Startpunkt ist, wird c um eins erhöht und überprüft, ob c größer als c_{max} ist und c_{max} gegebenenfalls aktualisiert. Falls der Intervallpunkt ein Endpunkt ist, wird c um eins verringert. Anschließend entspricht c_{max} der Cliquenzahl von $G[V']$, sodass geprüft werden kann, ob für die Knotenmenge V' eine zulässige Färbung durch die Farbmenge F' existiert. Falls diese existiert, wird jedem Knoten aus V' mit Hilfe einer FIFO-Liste eine Farbe aus F' zugewiesen. Abschließend wird die Knotenmenge V' aus dem Graph G entfernt.

Die Laufzeit des Algorithmus 5.4 setzt sich aus $O(n \log n)$ Schritten für das Sortieren der Intervallpunkte, $O(n)$ Schritten für das Berechnen der Cliquenzahl, $O(|C|)$ Schritten für das Erstellen der FIFO-Liste, $O(n)$ Schritten für das Färben der Knotenmenge V' und $O(n \log n)$ Schritten für das Entfernen aller Knoten von V' aus G zusammen. Die Gesamtlaufzeit ist somit folgende:

$$O\Big(\underbrace{n \log n}_{\substack{\textit{Sortieren der} \\ \textit{Intervallpunkte}}} + \underbrace{n}_{\substack{\textit{For-Schleife} \\ \textit{Cliquenzahl berechnen}}} + \underbrace{|C|}_{\substack{\textit{FIFO-Liste} \\ \textit{erstellen}}} + \underbrace{n}_{\substack{\textit{For-Schleife} \\ \textit{Färben der Knoten aus } V' \\ \textit{mit Pseudofahrzeugklasse}}} + \underbrace{n \log n}_{\substack{\textit{Knotenmenge} \\ \textit{aus } G \textit{ entfernen}}} \Big)$$

$$\overset{|C| \leq n}{=} O\big(n \log n\big)$$

Der Algorithmus 5.5 sucht in G nach Pseudofahrzeugklassen $F' \subseteq C$ für Knotenmengen $V' \subseteq V$ und ruft damit den Algorithmus 5.4 auf, welcher G nach Überprüfung,

Algorithmus 5.4 *färbePseudofahrzeugklasse*$(G, \mathrm{col}_{G^*}, V', F')$

```
1:  S ← sortiere alle Intervallpunkte der Knotenmenge V' aufsteigend, dabei sind Intervallpunkte eines Knotens die Start- und Endpunkte der rechtsoffenen Intervalle in der Intervalldarstellung
2:  c ← 0        \\Größe der aktuellen Clique
3:  c_max ← 0    \\Größe der maximalen Clique in G[V']
4:  for i ← 1 to |S| do
5:    if S[i] ist ein Startpunkt then
6:      c ← c + 1
7:      if c_max < c then
8:        c_max ← c
9:      end if
10:   else
11:     c ← c − 1
12:   end if
13: end for
14: if c_max ≤ |F'| then
15:   F ← erstelle eine FIFO-Liste („First In - First Out“) mit den Methoden F.poll(), und F.add(c), dabei entfernt F.poll() das erste Element aus der Liste und gibt es zurück und F.add(c) fügt c an das Ende der Liste
16:   F ← füge zu F alle Farben der Menge F' hinzu
17:   for i ← 1 to |S| do
18:     if S[i] ist ein Startpunkt der Intervalldarstellung eines Knotens v then
19:       col_{G*}(v) ← F.poll()
20:     else
21:       F.add(col_{G*}(v))
22:     end if
23:   end for
24:   G ← G[V \ {V'}]
25: end if
26: return (G, col_{G*})
```

ob $\omega(G[V']) \leq |F'|$ gilt, auf den induzierten Subgraphen $G[V \setminus V']$ reduziert. Der Algorithmus führt die Datenreduktion so lange aus, bis es keine Pseudofahrzeugklassen $F' \subseteq C$ mit Knotenmenge $V' \subseteq V$ mehr gibt, für die $\omega(G[V']) \leq |F'|$ gilt. In jeder Iteration wird über alle Farben $c \in C$ iteriert. Für die Farbe $c \in C$ wird für jede Knotenmenge V' einer Zusammenhangskomponente des induzierten Subgraphen $G\big[\{v \in V \mid c \in L(v)\}\big]$ der Algorithmus 5.4 mit Pseudofahrzeugklasse F' aufgerufen. Dabei enthält F' alle Farben, die in allen Farblisten $L(v)$ der Knoten $v \in V'$ vorkommen und gleichzeitig in keiner Farbliste eines Knotens der „Nachbarknotenmenge“ $\bigcup_{v \in V'} N_G(v) \setminus V'$ vorkommen. Um jede Knotenmenge V' einer Zusammenhangskomponente des induzierten Subgraphen $G\big[\{v \in V \mid c \in L(v)\}\big]$ zu erhalten, wird zuerst die

Algorithmus 5.5 *entfernePseudofahrzeugklassen*$(G, C, L, \mathrm{col}_{G^*})$

```
1: repeat
2:   k ← |V(G)|
3:   for all c ∈ C do
4:     S ← sortiere Knoten aus {v ∈ V | c ∈ L(v)} nach ihren Startpunkten in der
         Intervalldarstellung
5:     V' ← {}        \\ Knoten der Zusammenhangskomponente
6:     e_max ← 0      \\ größter Endpunkt eines Intervalls in der Zusammenhangskom-
                         ponente
7:     for i ← 1 to |S| do
8:       if Startpunkt der Intervalldarstellung von S[i] ≥ e_max then
9:         if |V'| > 0 then
10:          N ← ⋃_{v∈V'} N_G(v) \ V'
11:          F' ← ⋂_{v∈V'} L(v) \ ⋃_{w∈N} L(w)
12:          (G, col_{G*}) ← färbePseudofahrzeugklasse(G, col_{G*}, V', F')
13:        end if
14:        V' ← S[i]
15:        e_max ← Endpunkt der Intervalldarstellung von S[i]
16:      else
17:        V' ← V' ∪ S[i]
18:        if Endpunkt der Intervalldarstellung von S[i] > e_max then
19:          e_max ← Endpunkt der Intervalldarstellung von S[i]
20:        end if
21:      end if
22:    end for
23:  end for
24: until k = |V(G)|
25: return (G, col_{G*})
```

Knotenmenge $\{v \in V \mid c \in L(v)\}$ nach ihren Startpunkten in der Intervalldarstellung aufsteigend sortiert. Anschließend wird über diese Knoten iteriert. Falls der Intervallstartpunkt des Knotens v kleiner als der größte Intervallendpunkt e_{max} der vorherigen Iterationen ist, so wird die Menge V' um v erweitert. Ist zusätzlich der Intervallendpunkt von v größer als e_{max} so wird e_{max} aktualisiert. Falls der Intervallstartpunkt des Knotens v größer oder gleich e_{max} ist, dann ist $G[V']$ eine Zusammenhangskomponente von $G\big[\{v \in V \mid c \in L(v)\}\big]$. Dann wird die Pseudofahrzeugklasse F' für die Knotenmenge V' berechnet und der Algorithmus 5.4 aufgerufen. Um weitere Zusammenhangskomponenten zu erhalten, wird $V' = \{v\}$ und e_{max} gleich dem Intervallendpunkt von v gesetzt.

Um die Laufzeit der Datenreduktion zu berechnen, werden hier alle Laufzeiten der einzelnen Schritte erläutert. Die Repeat-Schleife ist durch $O(n)$ beschränkt, da mindestens ein Knoten aus G entfernt werden muss, damit ein weiterer Schleifendurchlauf ausgeführt wird. Durch die äußere For-Schleife kommt der Faktor $O(|C|)$ zur Laufzeit hinzu. Das Sortieren der Knotenmenge $\{v \in V \mid c \in L(v)\}$ benötigt durch den Intervallbaum als

Datenstruktur für den Intervallgraphen nur $O(n)$ Schritte. Durch die innere For-Schleife kommt der Faktor $O(n)$ hinzu, da die Farbe c in jeder Farbliste aller Knoten vorkommen kann. Das Finden jedes Nachbarknotens aller Knoten aus V' benötigt durch den Intervallbaum als Datenstruktur $O(n)$ Schritte. Aufgrund der Bitdarstellung der Farbmengen, werden nur $O(n)$ Schritte benötigt um F' zu berechnen. Die Gesamtlaufzeit setzt sich somit wie folgt zusammen:

$$O\Big(\underbrace{n}_{\text{Repeat-Schleife}} \cdot \underbrace{|C|}_{\substack{\text{äußere} \\ \text{For-Schleife}}} \cdot \Big(\underbrace{n}_{\substack{\text{Sortieren} \\ \text{nach} \\ \text{Startpunkten}}} + \underbrace{n}_{\substack{\text{innere} \\ \text{For-Schleife}}} \cdot \Big(\underbrace{n}_{\substack{\text{Finden von} \\ \text{Nachbarknoten}}} + \underbrace{n}_{\substack{F' \\ \text{berechnen}}} + \underbrace{n \log n}_{\substack{\text{Pseudofahrzeug-} \\ \text{klasse färben}}} \Big) \Big) \Big)$$

$$= O\big(|C| \cdot n^3 \log n\big)$$

5.2.4 Reduzieren auf eine Zusammenhangskomponente

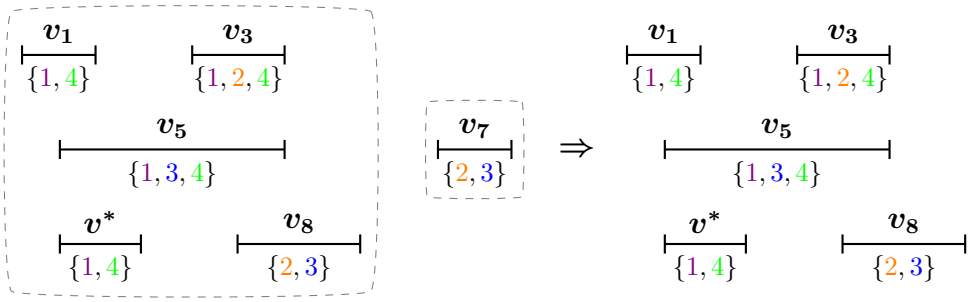
In diesem Abschnitt wird eine Datenreduktion vorgestellt, welche eine gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Inkrementelle Listenfärbung mit nur einer Zusammenhangskomponente reduziert. Diese Datenreduktion ist besonders wichtig, da in der Praxis einzelne Buchungen existieren, welche sich über Nacht oder über das Wochenende erstrecken und somit die einzige Verbindung zwischen den Subgraphen sind. Sollte eines dieser Intervalle durch eine andere Datenreduktion entfernt werden können, so können direkt ganze Zusammenhangskomponenten aus der Eingabe entfernt werden. Folgende Beobachtung zeigt die hier beschriebene Reduktion:

Beobachtung 5.2.15. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Falls der Graph G nicht zusammenhängend ist, gibt es eine Zusammenhangskomponente $G' = G[V']$ mit $V' \subseteq V$ und $v^* \in V'$, sodass $\big(G', v^*, C, L[V'], \text{col}_{G'-v^*}\big)$ eine äquivalente Eingabe für* Inkrementelle Listenfärbung *ist.*

Beweis. Um ein äquivalente Eingabe der Form $\big(G', v^*, C, L[V'], \text{col}_{G'-v^*}\big)$ mit $G' = G[V']$ aus $(G, v^*, C, L, \text{col}_{G-v^*})$ zu konstruieren, wird die Färbung $\text{col}_{G'-v^*}$ aus der Färbung col_{G-v^*} konstruiert, indem für alle Knoten $v \in V'$ $\text{col}_{G'-v^*}(v) := \text{col}_{G-v^*}(v)$ gesetzt wird. Im Folgenden wird die Korrektheit gezeigt.

Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine Ja-Instanz für Inkrementelle Listenfärbung mit nichtzusammenhängenden Graph G und einer Zusammenhangskomponente $G' = G[V']$ mit $V' \subseteq V$ und $v^* \in V'$. Dann gibt es eine L-zulässige Färbung col_G. Insbesondere gibt es für jeden induzierten Graphen von G eine L-zulässige Färbung. Daraus folgt, dass $\text{col}_{G[V']}$ mit $\text{col}_{G[V']}(v) := \text{col}_G(v)$ für alle Knoten $v \in V'$ eine $L[V']$-zulässige Färbung ist.

Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine Nein-Instanz. Dann existiert keine L-zulässige Färbung col_G. Angenommen es gibt eine $L[V']$-zulässige Färbung $\text{col}_{G[V']}$ für die Zusammenhangskomponente $G[V']$. Da G nicht zusammenhängend ist und $v^* \in V'$ gilt, ist die Färbung $\text{col}_G(v) := \text{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus V'$ und $\text{col}_G(v) := \text{col}_{G[V']}(v)$



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
col_{G-v^*}	1	2	1	4	4	1	2	2	3
col_G	-	2	-	1	-	4	2	-	5

Abbildung 5.5: Hier wird die Datenreduktion auf eine Zusammenhangskomponente dargestellt. Es gibt mit den induzierten Graphen $G\big[\{v_1, v_3, v_5, v_8, v^*\}\big]$ und $G\big[\{v_7\}\big]$ zwei Zusammenhangskomponenten. Da $v^* \in \{v_1, v_3, v_5, v_8, v^*\}$ gilt, wird v_7 aus G entfernt und $\text{col}_G(v_7) = \text{col}_{G-v*}(v_7)$ gesetzt.

für alle Knoten $v \in V'$ eine L-zulässige Färbung für G. Dies ist ein Widerspruch dazu, dass $(G, v^*, C, L, \text{col}_{G-v^*})$ eine Nein-Instanz ist, also existiert keine $L[V']$-zulässige Färbung $\text{col}_{G[V']}$, woraus folgt, dass $(G', v^*, C, L[V'], \text{col}_{G'-v^*})$ ebenfalls eine Nein-Instanz ist.

Somit wurde gezeigt, dass $\big(G', v^*, C, L[V'], \text{col}_{G'-v^*}\big)$ eine äquivalente Eingabe für Inkrementelle Listenfärbung ist. □

Für diese Datenreduktion gilt, dass jede Lösung $\text{col}_{G[V']}$ einer reduzierten Eingabe durch das Erweitern der $L[V']$-zulässigen Färbung $\text{col}_{G[V']}$ mit der L-zulässigen Färbung col_{G-v^*} zu einer Lösung für die eigentliche Eingabe wird. Falls $\text{col}_{G[V']}$ eine $L[V']$-zulässigen Färbung für $G[V']$ ist, so ist die Färbung col_G mit $\text{col}_G(v) := \text{col}_{G[V']}(v)$ für alle Knoten $v \in V'$ und $\text{col}_G(v) := \text{col}_{G-v^*}$ für alle Knoten $v \in V \setminus V'$ eine L-zulässige Färbung für G.

In Abbildung 5.5 ist die Datenreduktion anhand des Beispiels dargestellt. Im Beispiel gibt es mit den induzierten Graphen $G\big[\{v_1, v_3, v_5, v_8, v^*\}\big]$ und $G\big[\{v_7\}\big]$ zwei Zusammenhangskomponenten. Da $v^* \in \{v_1, v_3, v_5, v_8, v^*\}$ gilt, wird der einzige Knoten aus der Zusammenhangskomponente von $G\big[\{v_7\}\big]$ aus G entfernt und $\text{col}_G(v_7) = \text{col}_{G-v*}(v_7)$ gesetzt.

Abschließend wird der Algorithmus 5.6 für die Datenreduktion vorgestellt. Im Algorithmus entspricht der Graph G^* dem Originalgraphen vor der Datenreduktion und Der Graph G einem induzierten Subgraphen des Graphen G^*. Der Algorithmus erstellt zu Beginn eine nach den Intervallstartpunkten aufsteigend sortierte Knotenliste S. Des Weiteren wird der erste Knoten aus S der Menge M hinzugefügt und der größte Intervallendpunkt e_{max} auf den Intervallendpunkt des ersten Elements aus S gesetzt. Anschließend wird über alle Knoten v aus S iteriert. In jeder Iteration wird überprüft,

Algorithmus 5.6 *eineZusammenhangskomponente*$(G, v^*, \mathrm{col}_{G^*}, \mathrm{col}_{G^*-v^*})$

```
1: S ← sortiere alle Knoten nach ihren Startpunkten in der Intervalldarstellung
2: M ← {S[1]}
3: e_max ← Endpunkt der Intervalldarstellung von S[1]
4: for i ← 2 to |S| do
5:    if Startpunkt der Intervalldarstellung von S[i] < e_max then
6:       M ← M ∪ {S[i]}
7:       if Endpunkt der Intervalldarstellung von S[i] > e_max then
8:          e_max ← Endpunkt der Intervalldarstellung von S[i]
9:       end if
10:   else
11:      if v* ∈ M then
12:         break
13:      end if
14:      M ← {S[i]}
15:      e_max ← Endpunkt der Intervalldarstellung von S[i]
16:   end if
17: end for
18: for all v ∈ V \ M do
19:    col_{G*}(v) ← col_{G*−v*}(v)
20: end for
21: G ← G[M]
22: return (G, col_{G*})
```

ob der Intervallstartpunkt von v kleiner als e_{max} ist. Falls dies der Fall ist, wird die Knotenmenge M um v erweitert und geprüft, ob der Intervallendpunkt von v größer als e_{max} ist und e_{max} gegebenenfalls aktualisiert. Falls der Intervallstartpunkt von v größer als e_{max} ist, so ist $G[M]$ eine Zusammenhangskomponente. Falls $v^* \in M$ gilt, kann die For-Schleife verlassen werden. Gilt jedoch $v^* \notin M$, so existiert eine andere Zusammenhangskomponente, welche v^* enthält. Daher wird in diesem Fall $M = \{v\}$ und e_{max} auf den Intervallendpunkt von v gesetzt und mit dem nächsten Knoten aus S fortgefahren. Nachdem die Zusammenhangskomponente $G[M]$ mit $v^* \in M$ gefunden wurde, wird $\mathrm{col}_{G^*}(v) := \mathrm{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus M$ gesetzt und alle Knoten der Knotenmenge $V \setminus M$ aus G entfernt.

Um die Laufzeit der Datenreduktion zu berechnen, werden hier alle Laufzeiten der einzelnen Schritte erläutert. Das Sortieren der Knoten nach ihren Intervallstartpunkten benötigt durch den Intervallbaum als Datenstruktur $O(n)$ Schritte. Die For-Schleife ist durch $O(n)$ beschränkt, da alle Operationen innerhalb der For-Schleife in konstanter Zeit ausgeführt werden können. Das Färben der Knotenmenge $V \setminus M$ benötigt ebenfalls $O(n)$ Schritte und das Entfernen aller Knoten der Menge $V \setminus M$ aus G benötigt $O(n \log n)$ Schritte. Somit setzt sich die Gesamtlaufzeit wie folgt zusammen:

$$O(\underbrace{n}_{\substack{\text{Sortieren nach}\\ \text{Startpunkten}}} + \underbrace{n}_{\text{For-Schleife}} + \underbrace{n}_{\text{Knoten färben}} + \underbrace{n \log n}_{\text{Knoten entfernen}}) = O(n \log n)$$

5.2.5 Gewonnene strukturelle Eigenschaften durch Datenreduktion

In diesem Abschnitt wird auf die gewonnen strukturellen Eigenschaften durch die Datenreduktionen eingegangen. Nach Anwendung aller in diesem Kapitel beschriebenen Datenreduktionen auf eine gültige Eingabe für Inkrementelle Listenfärbung entsteht entweder eine äquivalente Eingabe für Inkrementelle Listenfärbung oder eine äquivalente Eingabe für Listenfärbung mit bestimmten strukturellen Eigenschaften. Um die Laufzeit der gesamten Datenreduktion und der dadurch gewonnenen strukturellen Eigenschaften zu beweisen, werden zuerst zwei Korollare bewiesen.

Korollar 5.2.16. *Sei $(G, v^*, C, L, \text{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Dann kann diese Eingabe in $O(|C| \cdot n^4 \log n)$ Schritten auf eine äquivalente Eingabe $(G', v^*, C, L', \text{col}_{G'-v^*})$ für* Inkrementelle Listenfärbung *reduziert werden, sodass die reduzierte Eingabe folgende strukturelle Eigenschaften besitzt:*

1. *Kein Knoten aus $V(G') \setminus \{v^*\}$ ist eindeutig färbbar.*
2. *Es gibt in G' keinen lokal einzigartig färbbaren Knoten.*
3. *Es gibt in G' keine Pseudofahrzeugklasse $F \subseteq C$ für eine Knotenmenge $M \subseteq V(G')$, sodass $\omega(G[M]) \leq |F|$ gilt.*
4. *G' ist ein zusammenhängender Graph.*

Beweis. Zuerst wird die Datenreduktion zum Entfernen von eindeutig färbbaren Knoten auf alle Knoten $v \in V(G') \setminus \{v^*\}$ angewendet. Die Datenreduktion darf nicht auf den Knoten v^* angewendet werden, da sonst auf eine äquivalente Eingabe für Listenfärbung reduziert wird. Diese Reduktion benötigt $O(n^2)$ Schritte und muss nur einmal ausgeführt werden, da alle anderen Datenreduktionen die Farblisten der Knoten nicht verändern. Anschließend werden die Datenreduktionen zum Entfernen von lokal einzigartigen Knoten, zum Entfernen von Pseudofahrzeugklassen $F \subseteq C$ für eine Knotenmenge $M \subseteq V(G')$, für die $\omega(G[M]) \leq |F|$ gilt, und zum Reduzieren auf eine Zusammenhangskomponente solange in dieser Reihenfolge ausgeführt, bis keine der drei Reduktionen die Knotenmenge von G' weiter reduziert. Diese Schleife ist durch $O(n)$ beschränkt, da in jeder Iteration mindestens ein Knoten aus G' entfernt werden muss, damit eine weitere Iteration ausgeführt wird. Mit den Laufzeiten der Datenreduktion lässt sich die Laufzeit der gesamten Datenreduktion wie folgt berechnen:

$$O\big(\underbrace{n^2}_{\substack{\text{Entfernen von}\\ \text{eindeutig}\\ \text{färbbaren Knoten}}} + \underbrace{n}_{\text{Schleife}} \cdot \big(\underbrace{n^3 \log n}_{\substack{\text{Entfernen von lokal}\\ \text{einzigartigen Knoten}}} + \underbrace{|C| \cdot n^3 \log n}_{\substack{\text{Entfernen spezieller}\\ \text{Pseudofahrzeugklassen}}} + \underbrace{n \log n}_{\substack{\text{Reduzieren auf eine}\\ \text{Zusammenhangskomponente}}}\big)\big)$$

$$= O\big(|C| \cdot n^4 \log n\big)$$

□

Mit Hilfe des nächsten Korollars werden die gewonnenen strukturellen Eigenschaften der Datenreduktion auf eine Eingabe für Listenfärbung gezeigt.

Korollar 5.2.17. *Sei (G, C, L) eine gültige Eingabe für* Listenfärbung. *Dann kann diese Eingabe in $O(|C| \cdot n^4 \log n)$ Schritten auf eine äquivalente Eingabe (G', C, L') für* Listenfärbung *reduziert werden, sodass die reduzierte Eingabe die folgenden strukturellen Eigenschaften besitzt:*

1. *Es gibt in G' keinen eindeutig färbbaren Knoten.*
2. *Es gibt in G' keinen lokal einzigartig färbbaren Knoten.*
3. *Es gibt in G' keine Pseudofahrzeugklasse $F \subseteq C$ für eine Knotenmenge $M \subseteq V$, sodass $\omega(G[M]) \leq |F|$ gilt.*

Beweis. Zuerst wird die Datenreduktion zum Entfernen von eindeutig färbbaren Knoten angewendet. Diese Reduktion benötigt $O(n^2)$ Schritte und muss nur einmal ausgeführt werden, da alle anderen Datenreduktionen die Farblisten der Knoten nicht verändern. Anschließend werden die Datenreduktionen zum Entfernen von lokal einzigartigen Knoten und zum Entfernen von Pseudofahrzeugklassen $F \subseteq C$ für eine Knotenmenge $M \subseteq V(G')$, für die $\omega(G[M]) \leq |F|$ gilt, solange in dieser Reihenfolge ausgeführt, bis keine der zwei Reduktionen die Knotenmenge von G' weiter reduziert. Diese Schleife ist durch $O(n)$ beschränkt, da in jeder Iteration mindestens ein Knoten aus G' entfernt werden muss, damit eine weitere Iteration ausgeführt wird. Mit den Laufzeiten der Datenreduktion lässt sich die Laufzeit der gesamten Datenreduktion wie folgt berechnen:

$$O\Big(\underbrace{n^2}_{\substack{\text{Entfernen von} \\ \text{eindeutig} \\ \text{färbbaren Knoten}}} + \underbrace{n}_{\text{Schleife}} \cdot \Big(\underbrace{n^3 \log n}_{\substack{\text{Entfernen von lokal} \\ \text{einzigartigen Knoten}}} + \underbrace{|C| \cdot n^3 \log n}_{\substack{\text{Entfernen spezieller} \\ \text{Pseudofahrzeugklassen}}} \Big) \Big)$$

$$= O\big(|C| \cdot n^4 \log n\big)$$

□

Mit Hilfe von Korollar 5.2.16 und Korollar 5.2.17 wird im Folgenden die Laufzeit der gesamten Datenreduktion und der dadurch gewonnenen strukturellen Eigenschaften bewiesen.

Satz 5.2.18. *Sei $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ eine gültige Eingabe für* Inkrementelle Listenfärbung. *Dann kann diese Eingabe in $O(|C| \cdot n^4 \log n)$ Schritten entweder auf eine äquivalente Eingabe*

1. *$(G', v^*, C, L', \mathrm{col}_{G'-v^*})$ für* Inkrementelle Listenfärbung *reduziert werden, sodass die reduzierte Eingabe folgende strukturelle Eigenschaften besitzt:*

a) Es gibt in G' keinen eindeutig färbbaren Knoten.

b) Es gibt in G' keinen lokal einzigartig färbbaren Knoten.

c) Es gibt in G' keine Pseudofahrzeugklasse $F \subseteq C$ für eine Knotenmenge $M \subseteq V(G')$, sodass $\omega(G[M]) \leq |F|$ gilt.

d) G' ist ein zusammenhängender Graph.

oder

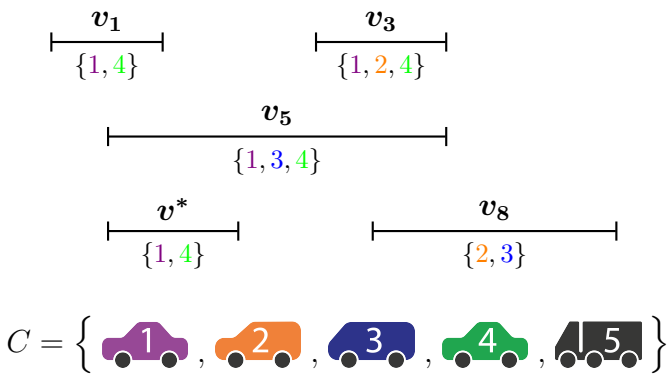
2. *(G', C, L') für* Listenfärbung *reduziert werden, sodass die reduzierte Eingabe die folgenden strukturellen Eigenschaften besitzt:*
   1. *Es gibt in G' keinen eindeutig färbbaren Knoten.*
   2. *Es gibt in G' keinen lokal einzigartig färbbaren Knoten.*
   3. *Es gibt in G' keine Pseudofahrzeugklasse $F \subseteq C$ für eine Knotenmenge $M \subseteq V$, sodass $\omega(G[M]) \leq |F|$ gilt.*

Beweis. Durch die Datenreduktion aus Korollar 5.2.16 wird $(G, v^*, C, L, \mathrm{col}_{G-v^*})$ in $O(|C| \cdot n^4 \log n)$ Schritten auf eine äquivalente Eingabe $(G', v^*, C, L', \mathrm{col}_{G'-v^*})$ reduziert. Falls nach der Reduktion $L(v^*) > 1$ gilt, so besitzt die Eingabe $(G', v^*, C, L', \mathrm{col}_{G'-v^*})$ alle angegebenen strukturellen Eigenschaften für eine reduzierte Eingabe von ILF. Falls $L(v^*) = 1$ gilt, so kann die Datenreduktion zum Entfernen eindeutig färbbarer Knoten einmalig auf alle Knoten ausgeführt werden. Diese Reduktion benötigt $O(n^2)$ Schritte und erzeugt eine äquivalente Eingabe für Listenfärbung. Mit Hilfe von Korollar 5.2.17 kann diese Eingabe in $O(|C| \cdot n^4 \log n)$ Schritten, auf eine äquivalente Eingabe (G', C, L'), reduziert werden. Diese reduzierte Eingabe besitzt dann alle angegebenen strukturellen Eigenschaften für eine reduzierte Eingabe von LF. Somit lässt sich die Gesamtlaufzeit einer Reduktion, auf eine äquivalente Eingabe (G', C, L') für Listenfärbung wie folgt berechnen:

$$O\big(\underbrace{|C| \cdot n^4 \log n}_{\substack{\text{Reduktion} \\ \text{Korollar 5.2.16}}} + \underbrace{n^2}_{\substack{\text{Entfernen von eindeutig} \\ \text{färbbaren Knoten}}} + \underbrace{|C| n^4 \log n}_{\substack{\text{Reduktion} \\ \text{Korollar 5.2.17}}}\big) = O(|C| \cdot n^4 \log n)$$

□

Abschließend wird nochmal auf das am Anfang des Kapitels vorgestellte Beispiel eingegangen. In Abbildung 5.6 ist die durch die gesamte Datenreduktion reduzierte Eingabe dargestellt. Die Eingabe ist weiterhin eine Eingabe für Inkrementelle Listenfärbung. Die Eingabe besteht nur noch aus fünf Intervallen anstelle aus zehn. Des Weiteren besitzt die Eingabe weder eindeutig färbbare Knoten oder lokal einzigartig färbbare Knoten noch Pseudofahrzeugklassen $F \subseteq C$ für eine Knotenmenge $V' \subseteq V$ mit $\omega(G[V']) \leq |F|$. Zusätzlich ist der reduzierte Eingabegraph zusammenhängend.



	col_{G-v^*}	col_G
v_1	1	-
v_2	2	2
v_3	1	-
v_4	4	1
v_5	4	-
v_6	1	4
v_7	2	2
v_8	2	-
v_9	3	5

Abbildung 5.6: Hier wird die reduzierte Eingabe dargestellt. Die Eingabe wurde von zehn Intervallen auf fünf reduziert. Des Weiteren besitzt die Eingabe alle strukturellen Eigenschaften einer reduzierten Eingabe für Inkrementelle Listenfärbung.

6 Algorithmen zum Lösen der Problemstellung

In diesem Kapitel werden Algorithmen zum Lösen der Problemstellung Inkrementelle Listenfärbung vorgestellt. Da in Kapitel 3 gezeigt wurde, dass Inkrementelle Listenfärbung und Vollständig Farbenreiche Stabile Menge NP-vollständig sind, kann jede gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Vollständig Farbenreiche Stabile Menge reduziert werden und dann durch einen Algorithmus für Vollständig Farbenreiche Stabile Menge gelöst werden. Daher werden in diesem Kapitel sowohl Algorithmen zum Lösen für Inkrementelle Listenfärbung, als auch Algorithmen zum Lösen für Vollständig Farbenreiche Stabile Menge vorgestellt. In Abschnitt 6.1 werden drei Suchbaumalgorithmen und in Abschnitt 6.2 zwei dynamische Programme vorgestellt.

6.1 Suchbaumalgorithmen

In diesem Abschnitt werden drei Suchbaumalgorithmen vorgestellt. Ein Suchbaumalgorithmus iteriert, wenn nötig, über alle Lösungsmöglichkeiten der Eingabeinstanz. In jedem Schritt findet eine Verzweigung über alle möglichen Werte einer bestimmten Eigenschaft statt, wobei die Eingabe durch das Setzen aller möglichen Werte der Eigenschaft in mehrere gültige Eingaben unterteilt wird. Falls das Setzen des Wertes die gültige Eingabe löst, bilden die Werte, welche zum Verzweigen genutzt wurden, um von der Wurzel zu dieser Eingabe zu gelangen, eine Lösung für die gesamte Eingabe. In Abschnitt 6.1.1 wird ein Suchabumalgorithmus für Listenfärbung vorgestellt, welcher über alle Farben eines Intervalls verzweigt. Anschließend wird in Abschnitt 6.1.2 ein Suchbaumalgorithmus für Inkrementelle Listenfärbung vorgestellt, welcher über alle Farben eines Intervalls verzweigt und dabei die Farbe der Teillösung priorisiert. Zuletzt wird in Abschnitt 6.1.3 ein Suchbaumalgorithmus für Vollständig Farbenreiche Stabile Menge vorgestellt, welcher über alle Intervalle einer maximalen Clique verzweigt.

6.1.1 Verzweigung über Farben

In diesem Abschnitt wird ein Suchbaumalgorithmus zum Lösen von Listenfärbung vorgestellt. Dieser kann auch für Instanzen von Inkrementelle Listenfärbung benutzt werden, indem die Teillösung vernachlässigt wird. Der Algorithmus wählt in jedem Schritt den Knoten $v \in V$ mit dem frühsten Intervallstartpunkt aus G. Anschließend verzweigt der Algorithmus über alle Farben $c \in L(v)$. Bei einer Verzweigung wird die

Farbe c aus allen Farblisten der Nachbarknoten von v entfernt und v aus G entfernt. Falls so ein Pfad von Verzweigungen entsteht, welcher für jeden Knoten $v \in V$ eine Farbe aus $L(v)$ ausgewählt hat, so ist dies eine L-zulässige Färbung für G. Mit diesem Algorithmus kann folgender Satz gezeigt werden.

Satz 6.1.1. Listenfärbung *ist in* $O(|C|^n \cdot n)$ *Schritten lösbar.*

Beweis. Der beschriebene Algorithmus löst Listenfärbung, da alle Lösungsmöglichkeiten durchprobiert werden. Für das Finden des Knotens mit dem kleinsten Intervallstartpunkt, das Löschen der Farbe c aus allen Farblisten der Nachbarknoten und für das Löschen von v aus G werden maximal $O(n)$ Schritte benötigt. Daher benötigt jeder rekursive Aufruf $O(n)$ Schritte. In jedem Rekursionsaufruf werden maximal $|C|$ viele neue Rekursionsaufrufe erzeugt, da jede Farbliste eines Knotens eine Teilmenge von C ist. Da jedes Intervall mit genau einer Farbe gefärbt werden muss, ist die Rekursionstiefe durch n beschränkt. Dies führt zu einer Gesamtlaufzeit von $O(|C|^n \cdot n)$. □

Durch Satz 6.1.1 folgt, dass Inkrementelle Listenfärbung auch in $O(|C|^n \cdot n)$ Schritten lösbar ist, indem der hier beschriebene Algorithmus verwendet wird und dabei die Teillösung vernachlässigt wird.

6.1.2 Verzweigung über Farben unter Berücksichtigung der Teillösung

In diesem Abschnitt wird ein Suchbaumalgorithmus zum Lösen von Inkrementelle Listenfärbung vorgestellt. Dieser Algorithmus berücksichtigt die Teillösung col_{G-v^*} und v^* um Lösungen mit minimalen Änderungen schneller finden zu können. Der Algorithmus wählt in jedem Schritt den Knoten $v \in V$, dessen Intervall am weitesten vom Intervall des Knotens v^* entfernt ist. Anschließend verzweigt der Algorithmus über alle Farben $c \in L(v)$, wobei als erste Farbe $\text{col}_{G-v^*}(v)$ gewählt wird, falls $v \neq v^*$ gilt. Bei einer Verzweigung wird die Farbe c aus allen Farblisten der Nachbarknoten von v entfernt und v aus G entfernt. Falls so ein Pfad von Verzweigungen entsteht, welcher für jeden Knoten $v \in V$ eine Farbe aus $L(v)$ ausgewählt hat, so ist dies eine L-zulässige Färbung für G.

Da der einzige Unterschied zum Suchbaumalgorithmus aus Abschnitt 6.1.1 darin besteht, dass zuerst die Knoten gewählt werden, dessen Intervalle am weitesten vom Intervall des Knotens v^* entfernt sind und dass bei der Verzweigung über alle Farben immer zuerst die Farbe col_{G-v^*} gewählt wird, ist die Laufzeit dieses Algorithmus auch $O(|C|^n \cdot n)$.

6.1.3 Verzweigung über Intervalle

In diesem Abschnitt wird ein Suchbaumalgorithmus zum Lösen von Vollständig Farbenreiche Stabile Menge von van Bevern u. a. [Bev+15] vorgestellt. Der Algorithmus sucht in jedem Schritt die maximale Clique K, von Intervallen dessen Intervallstartpunkt vor dem ersten Intervallendpunkt aller Knoten liegt. Zusätzlich wird

$F := \bigcup_{v \in K} col_G(v)$ berechnet. Anschließend wird für jede Farbe c in F über einen Knoten $v \in K$ mit $\text{col}_G(v) = c$ verzweigt, dessen Intervall zuerst endet. Bei der Verzweigung über das Intervall von Knoten v wird das Intervall der Lösungsmenge hinzugefügt und alle Nachbarknoten von v und alle Knoten $w \in V$ mit $col(w) = c$ aus G entfernt.

Die Laufzeit dieses Suchbaumalgorithmus ist $O(\Gamma^{|C|} \cdot n)$, wobei Γ die maximale Anzahl von Farben in einer maximalen Clique ist. Auch hier gilt, dass dieser Algorithmus auch Inkrementelle Vollständig Farbenreiche Stabile Menge löst, indem die Teillösung vernachlässigt wird.

6.2 Dynamische Programme

In diesem Abschnitt werden zwei dynamische Programme von van Bevern u. a. [Bev+15] zum Lösen von Vollständig Farbenreiche Stabile Menge vorgestellt. In Abschnitt 6.2.1 wird ein dynamisches Programm für Parameter $|C|$, die Anzahl der Farben angegeben, und in Abschnitt 6.2.2 wird ein dynamisches Programm für Parameter Q, die Anzahl der „Live-Farben“ angegeben.

6.2.1 Mit Parameter $|C|$ der Anzahl der Farben

In diesem Abschnitt wird das dynamische Programm für Parameter $|C|$, die Anzahl der Farben, zum Lösen von Vollständig Farbenreiche Stabile Menge vorgestellt. Sei (G, C, col_G) eine Eingabe für Vollständig Farbenreiche Stabile Menge, wobei G k-kompakt für ein minimales k ist. Für $i \in \{1, \ldots, k+1\}$ und $F \subseteq C$ bezeichnet $T[i, F]$ die Größe der maximalen stabilen Menge S in G, welche nur Knoten $v \in S$ enthält mit $\text{col}_G(v) \in F$ und Intervallstartpunkten größer gleich i. Für $i = k + 1$ und jede Menge $F \subseteq C$ gilt $T[i, F] = 0$. Im Weiteren wird für einen Knoten $v \in V$ der Intervallstartpunkt mit v_s und der Intervallendpunkt mit v_e bezeichnet. Für die Berechnung von $T[i-1, F]$ für $i \in \{1, \ldots, k+1\}$ und $F \subseteq C$ gibt es zwei Fälle zu beachten:

1. Es gibt eine maximale farbenreiche stabile Menge S mit einem Knoten $v \in S$ für den $v_s = i - 1$ gilt und für alle Knoten w aus S gilt $\text{col}_G(w) \in F$ und $i - 1 \leq w_s$. Dann ist $T[i-1, F] = 1 + T[v_e + 1, F \setminus \{col_G(v)\}]$.
2. Sonst ist $T[i-1, F] = T[i, F]$.

Die Rekursionsgleichung um eine maximale farbenreiche stabile Menge zu berechnen, sieht wie folgt aus:

$$T[i-1, F] = \max \begin{cases} T[i, F] \\ 1 + \max\limits_{\substack{v \in V, v_s = i-1, \\ \text{col}_G(v) \in F}} T[v_e + 1, F \setminus \{col_G(v)\}]. \end{cases}$$

Falls nach der Ausführung $T[1, C] = |C|$ gilt, so wurde eine vollständig farbenreiche stabile Menge in G gefunden. Die Laufzeit des dynamischen Programms ist $O(2^{|C|} \cdot n)$, da in jedem Schritt die Rekursionsgleichung für alle Teilmengen von C berechnet werden muss [Bev+15].

6.2.2 Mit Parameter Q der maximalen Anzahl von Live-Farben

In diesem Abschnitt wird das dynamische Programm für Parameter „maximale Anzahl von Live-Farben“ zum Lösen von Vollständig Farbenreiche Stabile Menge vorgestellt. Dieses dynamische Programm basiert auf dem in Abschnitt 6.2.1 vorgestellten dynamischen Programm und schränkt die Anzahl der berechneten Rekursionsgleichungen in jedem Schritt ein. Zuerst wird der Parameter „maximale Anzahl von Live-Farben“ definiert.

Definition 6.2.1. Sei $G = (V, E)$ ein k-kompakter Intervallgraph, C eine Farbmenge und col_G eine Färbung für G, sodass $\text{col}_G(v) \in C$ für alle Knoten $v \in V$ gilt. Für jedes $i \in \{1, \ldots, k+1\}$, sei

1. $L_i \subseteq C$ die Menge der Farben, sodass für alle Farben $c \in L_i$ ein Knoten $v \in V$ mit $\text{col}_G(v) = c$ und $v_s \leq i$ existiert, und

2. $R_i \subseteq C$ die Menge der Farben, sodass für alle Farben $c \in R_i$ ein Knoten $v \in V$ mit $\text{col}_G(v) = c$ und $v_s \geq i$ existiert.

Dann ist $Q := max_{i \in \{1,\ldots,k+1\}} |L_i \cap R_i|$ die maximale Anzahl von *Live-Farben*.

Sei (G, C, col_G) eine Eingabe für Vollständig Farbenreiche Stabile Menge, wobei G k-kompakt für ein minimales k ist. Mit Hilfe dieser Definition von Live-Farben ist offensichtlich, dass eine maximale farbenreiche stabile Menge S, welche nur Knoten mit Intervallstartpunkten mindestens i enthält, für jede Farben aus $c \in \overline{L}_i$ mit $\overline{L}_i := C \setminus L_i$ einen Knoten $v \in S$ mit $\text{col}_G(v) = c$ enthalten muss, um nach Ausführung des gesamten Programms eine vollständig farbenreiche stabile Menge für G zu erhalten. Das heißt, $T[i, F]$ muss nur für $i \in \{1, \ldots, k+1\}$ und $\overline{L}_i \subseteq F \subseteq C$ berechnet werden. Des Weiteren gilt für eine maximale farbenreiche stabile Menge S, welche nur Knote mit Intervallstartpunkten mindestens i enthält, dass $\text{col}_G(v) \in R_i$ für alle $v \in S$ gilt. Daraus folgt, dass das dynamische Programm nur $T[i, F]$ für $i \in \{1, \ldots, k+1\}$ und $\overline{L}_i \subseteq F \subseteq R_i$ lösen muss. Die Rekursionsgleichung um eine maximale farbenreiche stabile Menge zu berechnen, sieht wie folgt aus:

$$T[i-1, F] = \max \begin{cases} T[i, (F \cup \overline{L}_i) \cap R_i] \\ 1 + \max\limits_{\substack{v \in V, v_s = i-1, \\ \text{col}_G(v) \in F}} T[v_e + 1, (F \cup \overline{L}_{v_e+1}) \cap (R_{v_e+1} \setminus \{col_G(v)\})]. \end{cases}$$

Falls nach der Ausführung $T[1, C] = |C|$ gilt, so wurde eine vollständig farbenreiche stabile Menge in G gefunden. Durch die Beschränkung von F auf $\overline{L}_i \subseteq F \subseteq R_i$ folgt, dass $F \setminus \overline{L}_i \subseteq L_i \cap R_i$ gilt. Damit ist die Laufzeit des dynamischen Programms durch $O(2^Q \cdot n)$ beschränkt [Bev+15].

7 Experimentelle Auswertung

In diesem Abschnitt werden die Laufzeiten der in Kapitel 6 vorgestellten Algorithmen mit Hilfe von künstlich erzeugten Testinstanzen $(G, C, L, \text{col}_{G-v^*})$ für Inkrementelle Listenfärbung verglichen. Des Weiteren wird überprüft, welchen Einfluss die Anzahl der Intervalle in G und die Anzahl der Farben in C auf die Laufzeit haben.

Implementiert wurden die Suchbaumalgorithmen aus Abschnitt 6.1.1 und aus Abschnitt 6.1.2, welche direkt Instanzen von Inkrementelle Listenfärbung lösen. Diese besitzen beide eine Laufzeit von $O(|C|^n \cdot n)$. Zusätzlich wurde der Suchbaumalgorithmus aus Abschnitt 6.1.3 und das dynamische Programm aus Abschnitt 6.2.1 implementiert. Für diese Algorithmen wird die Instanz für Inkrementelle Listenfärbung zuerst mit der Reduktion aus Abschnitt 3.2 auf eine äquivalente Instanz für Inkrementelle Vollständig Farbenreiche Stabile Menge reduziert. Bezüglich der so reduzierten Eingabe besitzt der Suchbaumalgorithmus eine Laufzeit von $O(\Gamma^{|C|} \cdot n)$, wobei Γ die maximale Anzahl von Farben in einer Clique ist, und das dynamische Programm eine Laufzeit von $O(2^{|C|} \cdot n)$.

In Abschnitt 7.1 wird auf die Implementierungsdetails und auf die Testumgebung eingegangen. Anschließend wird in Abschnitt 7.2 die Erzeugung der künstlichen Testinstanzen erläutert und abschließend werden in Abschnitt 7.3 die experimentellen Ergebnisse ausgewertet.

7.1 Implementierungsdetails

Implementiert wurden alle Algorithmen in Kotlin 1.3.31. Der Compiler von Kotlin übersetzt alle Kotlin-Dateien zu Java-Dateien (in dieser Implementierung auf Java 8), sodass der Code auf der Java Virtual Machine (JVM) ausgeführt werden kann. Der Source-Code ist auf der beigefügten CD zu finden. Die Experimente wurden auf einem Computer mit einem Intel Core i7-5600U mit vier Kernen, die jeweils auf $2,6$GHz getaktet sind, und mit 12GB Arbeitsspeicher unter Ubuntu 18.04 ausgeführt.

7.2 Erzeugung künstlicher Testinstanzen

Um die Laufzeit der implementierten Algorithmen und die Beeinflussung von Parametern wie die Anzahl der Intervalle in G oder die Anzahl der Farben in C auszuwerten, wird in diesem Abschnitt erläutert, wie die künstlichen Testinstanzen für Inkrementelle Listenfärbung erzeugt wurden.

Bevor die Erzeugung der Testinstanzen genauer beschrieben wird, wird zuerst auf die strukturellen Eigenschaften der generierten Instanzen eingegangen. Alle erzeugten

Testinstanzen bestehen aus einem zusammenhängenden Graph $G = (V, E)$, einem Knoten v^*, einer Farbmenge C, für jeden Knoten $c \in V$ eine Liste $L : V \to 2^C$ von zulässigen Farben und einer L-zulässigen Färbung col_{G-v^*}. Der Graph G ist zusammenhängend, da ein nicht zusammenhängender Graph mit Hilfe der Datenreduktion aus Abschnitt 5.2.4 auf einen zusammenhängenden Graph, welcher v^* enthält, reduziert werden kann. Des Weiteren existiert für jede Farbe $c \in L(v^*)$ ein Knoten $v \in N_G(v^*)$ mit $\text{col}_{G-v^*}(v) = c$. Dies sorgt dafür, dass es keine triviale Lösung gibt, in der dem Knoten v^* eine Farbe aus $L(v^*)$ zugewiesen werden kann und die Färbung der Teillösung übernommen werden kann. Des Weiteren ist jedes Intervall mindestens 30 Minuten lang und jedes Intervall kann zu jeder vollen, Viertel-, halben oder Dreiviertelstunde zwischen 6 und 18 Uhr eines Werktags starten oder enden. Die Start- und Endzeit kann dabei an verschiedenen Tagen sein. Mit einer Wahrscheinlichkeitsverteilung über die Dauer eines Intervalls können so realitätsnahe Testinstanzen für das Corporate Car Sharing erzeugt werden. Mit dem Wissen, welche strukturellen Eigenschaften die Testinstanzen nach der Erzeugung besitzen, wird nun die Erzeugung der künstlichen Testinstanzen erläutert.

Um eine Instanz mit n Intervallen und k Farben zu generieren, wird zuerst die Farbmenge $C := \{1, \ldots, k\}$ erzeugt. Als nächstes wird zufällig ein Intervall mit einer Dauer von mindestens 30 Minuten und mit Intervallstart- und Intervallendzeit zu einer vollen, Viertel-, halben oder Dreiviertelstunde zwischen 6 und 18 Uhr eines Werktags generiert. Dann werden iterativ $n-1$ weitere Intervalle generiert. Bei der Erzeugung eines Intervalls wird der Startpunkt zufällig unter allen zulässig Startpunkten ausgewählt. Ein Startpunkt ist zulässig, wenn der Startpunkt größer gleich dem frühsten Intervallstartpunkt und kleiner dem spätesten Intervallendpunkt aller bisher generierten Intervalle ist. Des Weiteren muss der Startzeitpunkt zu einer vollen, Viertel-, halben oder Dreiviertelstunde zwischen 6 und 18 Uhr eines Werktages liegen und darf nur weniger als $|C|$ von den bisher generierten Intervallen schneiden. Die Länge des Intervalls wird mit Hilfe einer Wahrscheinlichkeitsverteilung zufällig ausgewählt. Falls ein so erstelltes Intervall zu einem Zeitpunkt mehr als $|C| - 1$ von den bisher generierten Intervallen überschneidet, wird dies verworfen und ein neues generiert. So wird sichergestellt, dass später eine Färbung col_{G-v^*} existieren kann. Sei G der so generierte zusammenhängende Intervallgraph und v^* ein zufällig ausgewähltes Intervall der generierten Intervalle. Nun wird die Färbung col_{G-v^*} erzeugt, indem $\text{col}_{G-v^*}(v)$ für alle Knoten $v \in V \setminus \{v^*\}$ eine Farbe $c \in C$ zugewiesen wird, sodass $\text{col}_{G-v^*}(w) \neq c$ für alle Nachbarknoten $w \in N_G(v)$ gilt. Dies ist möglich, da G durch Konstruktion keine Cliquen enthält, deren Größe größer ist als $|C|$. Abschließend werden die Listen der zulässigen Farben für jeden Knoten erzeugt. Für jeden Knoten $v \in V \setminus \{v^*\}$ wird eine zufällige Teilmenge $F \subseteq C$ gewählt und $L(v) := F \cup \{\text{col}_{G-v^*}(v)\}$ gesetzt. Für den Knoten v^* wird eine zufällige nichtleere Teilmenge $F \subseteq \bigcup_{v \in N_G(v^*)} \text{col}_{G-v^*}(v)$ gewählt und $L(v^*) := F$ gesetzt. Damit ist col_{G-v^*} eine L-zulässige Färbung und die erzeugte Instanz $(G, C, L, \text{col}_{G-v^*})$ besitzt alle beschriebenen Eigenschaften.

Um die Algorithmen vergleichen zu können, wurden so für jede Anzahl der Intervalle $n \in \{5, \ldots, 23\}$ acht Ja-Instanzen und acht Nein-Instanzen, durch generieren einer Instanz mit direkter Überprüfung ob es eine Ja- oder Nein-Instanz ist, erzeugt. Dabei

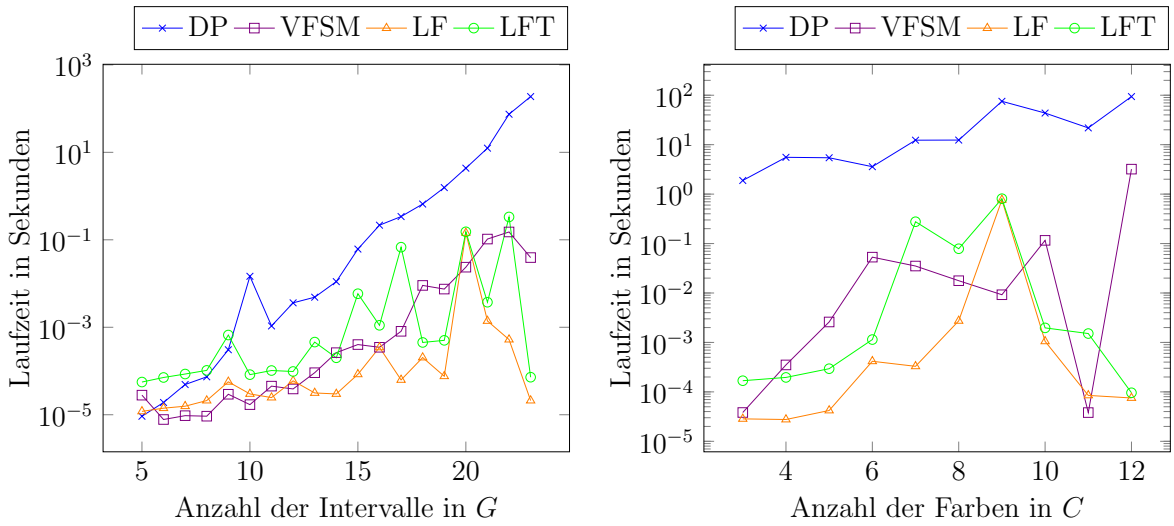


Abbildung 7.1: Hier wird die Abhängigkeit von der Anzahl der Intervalle und der Anzahl der Farben einer Instanz für Inkrementelle Listenfärbung bezüglich der Laufzeit verglichen.

wurde für jede Instanz die Anzahl der Farben zufällig zwischen 3 und $0.6n$ gewählt, da in der Praxis meistens weniger Fahrzeuge vorhanden sind als Buchungen. Instanzen mit mehr Intervallen wurden nicht generiert, da diese aufgrund von Hardwareeinschränkung nicht mehr durch alle implementierten Algorithmen lösbar waren.

7.3 Experimentelle Ergebnisse

In diesem Abschnitt werden die verschiedenen Algorithmen verglichen und die Ergebnisse ausgewertet. Im Folgenden wird das dynamische Programm zum Lösen von Vollständig Farbenreiche Stabile Menge aus Abschnitt 6.2 mit DP, der Suchbaumalgorithmus zum Lösen von Vollständig Farbenreiche Stabile Menge aus Abschnitt 6.1.3 mit VFSM, der Suchbaumalgorithmus zum Lösen von Listenfärbung aus Abschnitt 6.1.1 mit LF und der Suchbaumalgorithmus zum Lösen von Inkrementelle Listenfärbung aus Abschnitt 6.1.2 mit LFT abgekürzt.

In Abbildung 7.1 wird die Durchschnittslaufzeit aller implementierten Algorithmen bezüglich der Anzahl der Intervalle in G und der Anzahl der Farben in C verglichen. Die Durchschnittslaufzeit bezüglich der Anzahl der Intervalle und der Anzahl der Farben umfasst dabei sowohl Ja-Instanzen als auch Nein-Instanzen. Dabei fällt auf, dass ab zehn Intervallen das dynamische Programm wesentlich langsamer ist als alle Suchbaumalgorithmen. Des Weiteren fällt auf, dass der Suchbaumalgorithmus LF für jede Anzahl von Intervallen schneller ist als der Suchbaumalgorithmus LFT, welcher die Teillösung mitberücksichtigt. Schaut man sich die Durchschnittslaufzeit der Algorithmen bezüglich der Anzahl der Farben in C an, so ist auch hier wieder das dynamische Programm langsamer als alle Suchbaumalgorithmen, jedoch beeinflusst die Anzahl der Farben in C die Lauf-

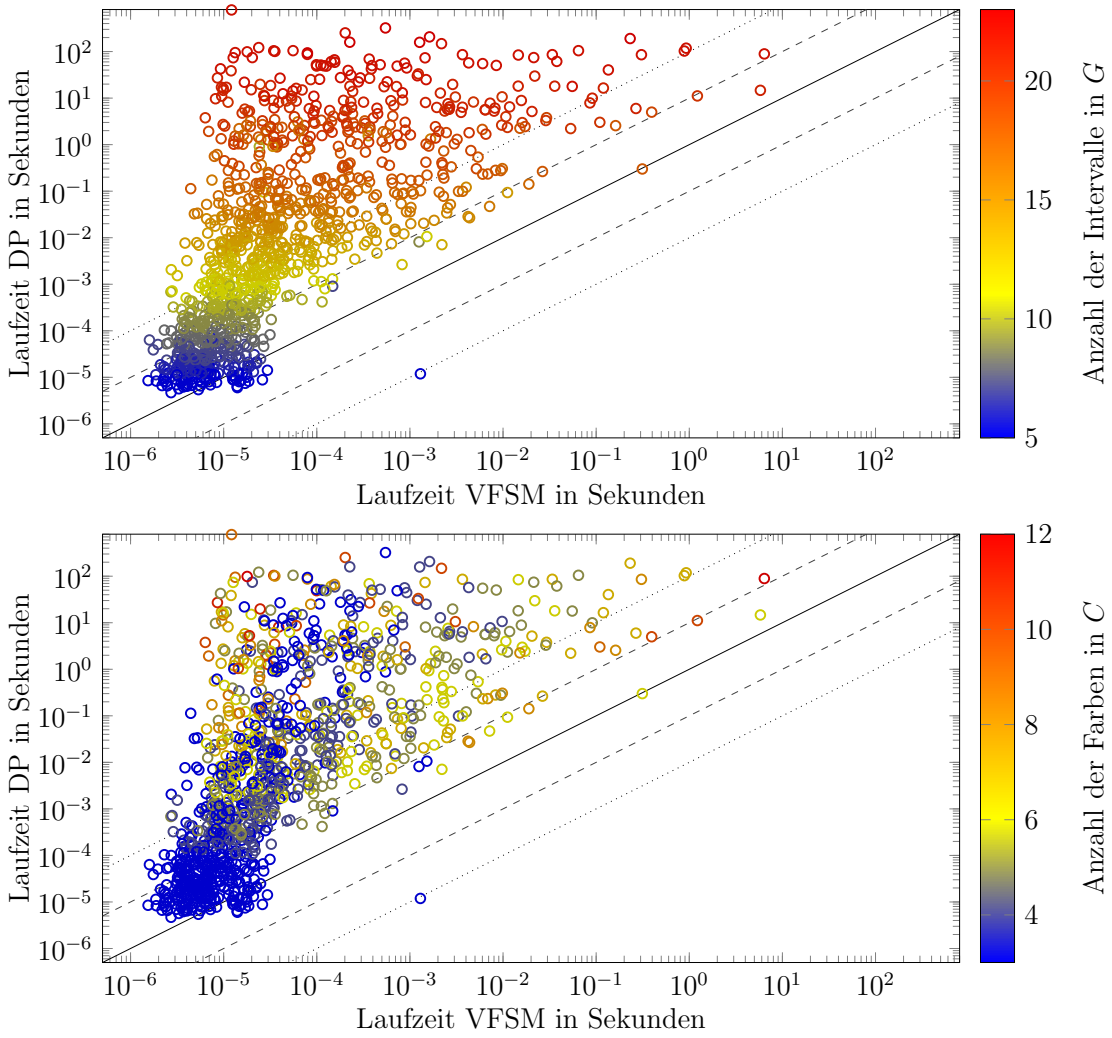


Abbildung 7.2: Hier wird die Laufzeit des Suchbaumalgorithmus für VFSM aus Abschnitt 6.1.3 mit der Laufzeit des dynamischen Programms für Vollständig Farbenreiche Stabile Menge aus Abschnitt 6.2.1 anhand der Anzahl der Intervalle und der Anzahl der Farben verglichen. Die gestrichelte Linie entspricht dem Faktor 10 und die gepunktete Linie dem Faktor 100.

zeit des dynamischen Programms nicht so stark wie die Anzahl der Intervalle. Für die Suchbaumalgorithmen LF und LFT nimmt die Durchschnittslaufzeit mit einer größeren Anzahl von Farben sogar ab.

In Abbildung 7.2 werden alle Laufzeiten des dynamischen Programms DP und des Suchbaumalgorithmus VFSM für alle Testinstanzen bezüglich der Anzahl der Intervalle und der Anzahl der Farben mit Hilfe von Heatmaps verglichen. Der Suchbaumalgorithmus VFSM ist schon für kleine Instanzen schneller als das dynamische Programm DP. Für Instanzen mit zehn Intervallen ist der Suchbaumalgorithmus um den Faktor zehn und für Instanzen mit 15 Intervallen um den Faktor 100 schneller. Der Vergleich bezüglich

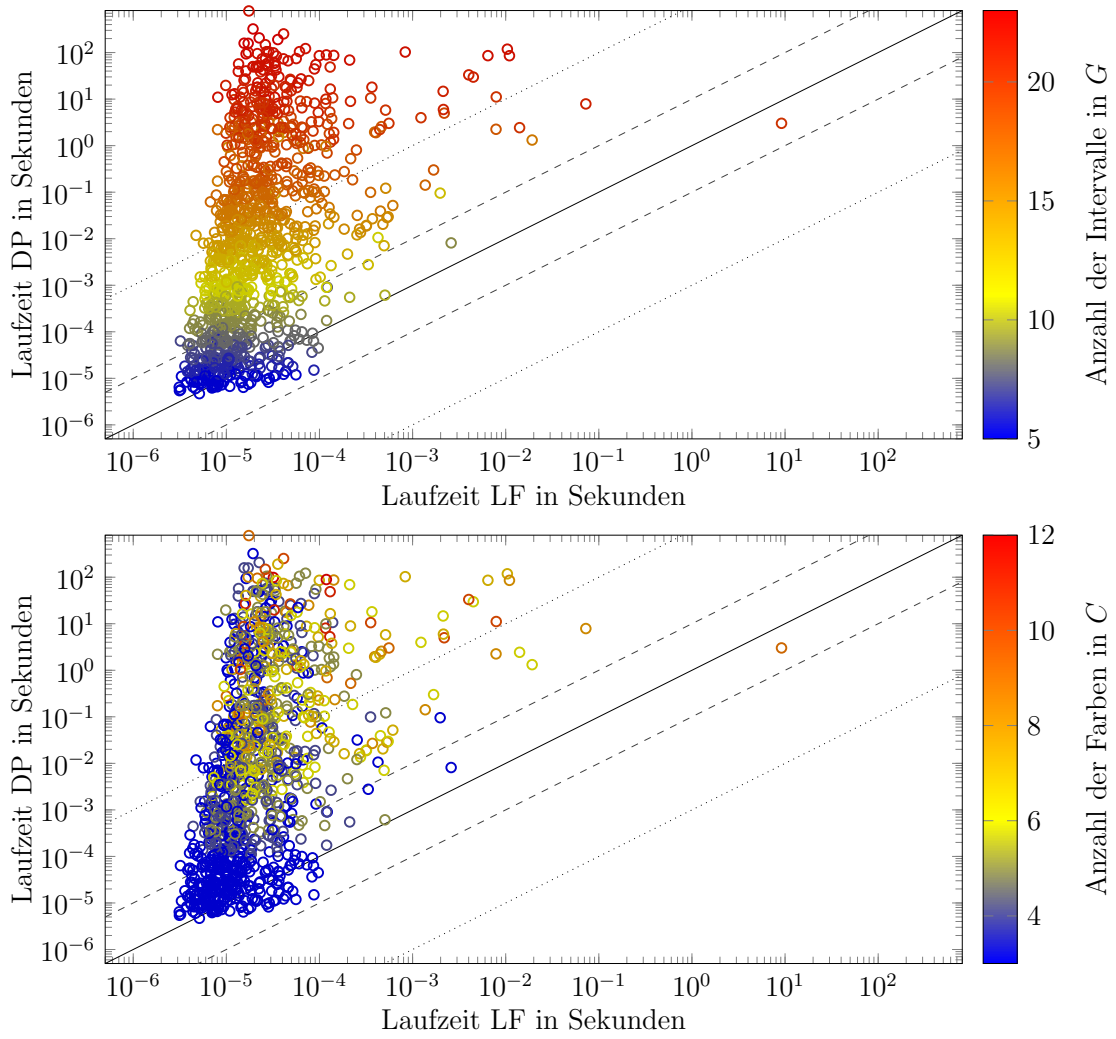


Abbildung 7.3: Hier wird die Laufzeit des Suchbaumalgorithmus für Listenfärbung aus Abschnitt 6.1.1 mit der Laufzeit des dynamischen Programms für Vollständig Farbenreiche Stabile Menge aus Abschnitt 6.2.1 anhand der Anzahl der Intervalle und der Anzahl der Farben verglichen. Die gestrichelte Linie entspricht dem Faktor 10 und die gepunktete Linie dem Faktor 1000.

der Anzahl der Farben zeigt, dass für diese Algorithmen die Laufzeit nicht stark von der Anzahl der Farben beeinflusst wird, da sowohl für Instanzen mit vier Farben als auch für Instanzen mit zehn Farben gleich lange gebraucht werden kann.

In Abbildung 7.3 werden alle Laufzeiten des dynamischen Programms DP und des Suchbaumalgorithmus LF für alle Testinstanzen bezüglich der Anzahl der Intervalle und der Anzahl der Farben mit Hilfe von Heatmaps verglichen. Der Suchbaumalgorithmus LF ist für fast alle Instanzen schneller als das dynamische Programm DP. Für Instanzen mit zehn Intervallen ist der Suchbaumalgorithmus um den Faktor zehn und für fast alle Instanzen mit 15 Intervallen um den Faktor 1000 schneller. Der Vergleich bezüglich

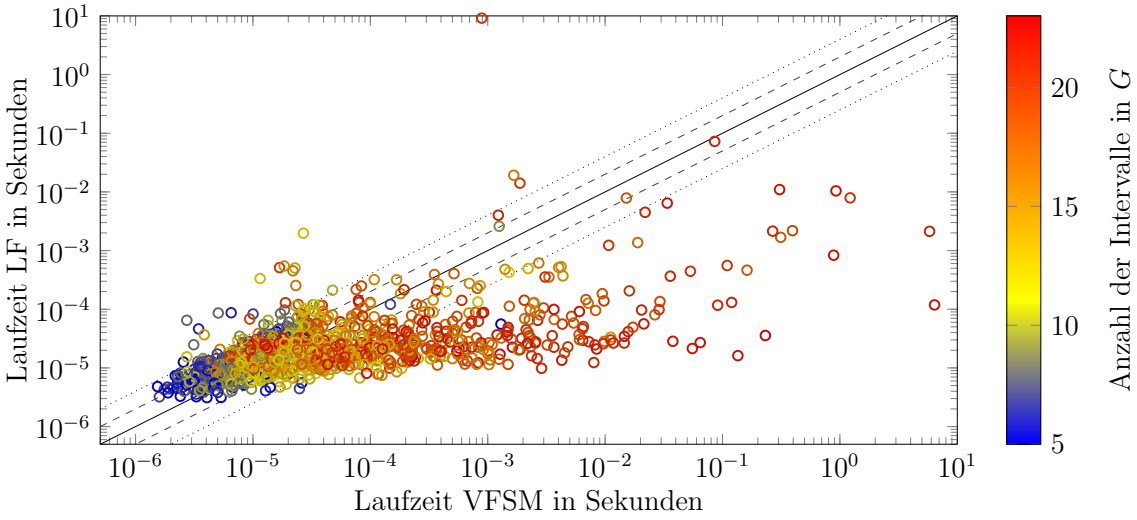


Abbildung 7.4: Hier wird die Laufzeit des Suchbaumalgorithmus für Listenfärbung aus Abschnitt 6.1.1 mit der Laufzeit des Suchbaumalgorithmus für Vollständig Farbenreiche Stabile Menge aus Abschnitt 6.1.3 anhand der Anzahl der Intervalle verglichen. Die gestrichelte Linie entspricht dem Faktor 2 und die gepunktete Linie dem Faktor 4.

der Anzahl der Farben zeigt auch hier wieder keinen stärkeren Zusammenhang mit der Laufzeit.

Nun werden die Laufzeiten der Suchbaumalgorithmen untereinander verglichen. In Abbildung 7.4 werden alle Laufzeiten des Suchbaumalgorithmus LF und des Suchbaumalgorithmus VFSM für alle Testinstanzen bezüglich der Anzahl der Intervalle mit Hilfe einer Heatmap verglichen. Für Instanzen mit wenig Intervallen ist der Suchbaumalgorithmus VFSM teilweise um Faktor zwei schneller als der Suchbaumalgorithmus LF. Für die meisten Instanzen mit mehr als 15 Intervallen ist der Suchbaumalgorithmus LF jedoch um Faktor vier schneller.

In Abbildung 7.5 werden alle Laufzeiten des Suchbaumalgorithmus LF und des Suchbaumalgorithmus LFT für alle Testinstanzen bezüglich der Anzahl der Intervalle mit Hilfe einer Heatmap verglichen. Für fast alle Instanzen und damit jeder Anzahl von Intervallen ist der Suchbaumalgorithmus LF schneller als der Suchbaumalgorithmus LFT. Für die meisten Instanzen ist er um Faktor zwei bis vier schneller.

Abschließend werden die Ergebnisse zusammengefasst. Wie aus den Auswertungen ersichtlich wird, ist das dynamische Programm schon für Instanzen mit zehn Intervallen deutlich langsamer als alle Suchbaumalgorithmen. Dies hängt damit zusammen, dass nach der Reduktion der Instanz für Inkrementelle Listenfärbung auf eine äquivalente Instanz für Vollständig Farbenreiche Stabile Menge die Anzahl der Intervalle der Anzahl der Farben entspricht. Da dies der entscheidende Parameter für die Laufzeit des dynamischen Programms DP ist, nimmt die Laufzeit des dynamischen Programms mit einer wachsenden Anzahl von Intervallen stark zu. Unerwartet ist, dass der Suchbaumalgorithmus VFSM für Instanzen mit mehr als 15 Intervallen langsa-

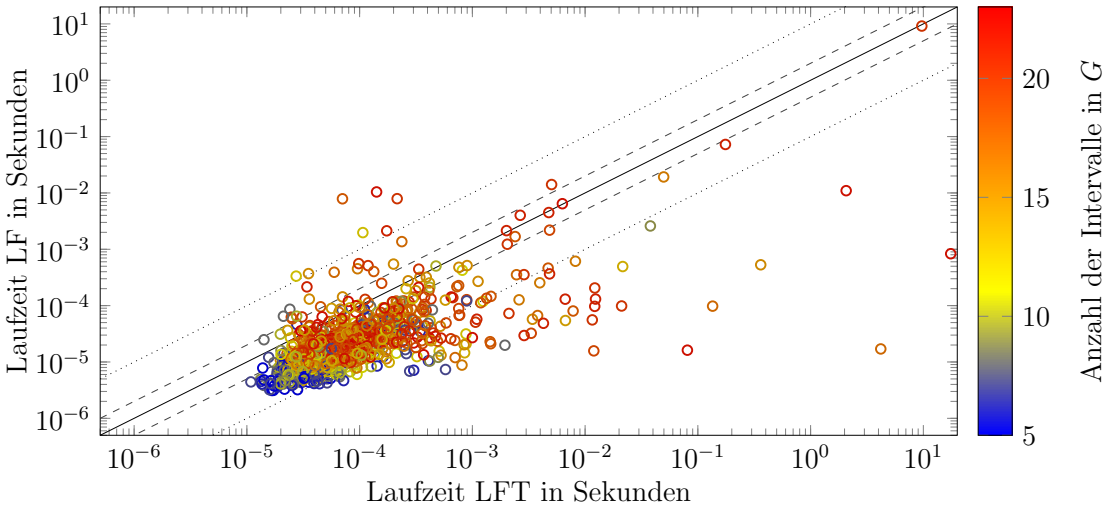


Abbildung 7.5: Hier wird die Laufzeit des Suchbaumalgorithmus für Listenfärbung aus Abschnitt 6.1.1 mit der Laufzeit des Suchbaumalgorithmus für Inkrementelle Listenfärbung aus Abschnitt 6.1.2 anhand der Anzahl der Intervalle verglichen. Die gestrichelte Linie entspricht dem Faktor 2 und die gepunktete Linie dem Faktor 10.

mer ist als der Suchbaumalgorithmus LF, da beide Algorithmen im schlimmsten Fall die gleiche Anzahl von Möglichkeiten durchprobieren müssen. Eine weitere überraschende Erkenntnis ist, dass der Suchbaumalgorithmus LFT für fast alle Instanzen langsamer ist als der Suchbaumalgorithmus LF. Hieraus lässt sich schlussfolgern, dass für fast alle Instanzen die L-zulässige Färbung col_G stark von der L-zulässigen Färbung col_{G-v^*} abweicht.

8 Ausblick

In dieser Arbeit wurde die NP-Vollständigkeit von Inkrementelle Listenfärbung gezeigt. Des Weiteren wurde gezeigt, dass eine gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Inkrementelle Vollständig Farbenreiche Stabile Menge reduziert werden kann. Die Reduktion ist besonders interessant, da die strukturellen Eigenschaften der Eingabe in gewisser Weise beibehalten werden. Mit Hilfe der Reduktion konnten Algorithmen für das Lösen von Inkrementelle Vollständig Farbenreiche Stabile Menge auch zum Lösen für das Problem Inkrementelle Listenfärbung genutzt werden.

Des Weiteren wurden Datenreduktionsalgorithmen mit einer polynomiellen Laufzeit vorgestellt. Mit Hilfe dieser Algorithmen wird jede gültige Eingabe für Inkrementelle Listenfärbung auf eine äquivalente Eingabe für Inkrementelle Listenfärbung oder Listenfärbung mit bestimmten strukturellen Eigenschaften reduziert. Da durch die Datenreduktion aus Abschnitt 5.2.3 und Abschnitt 5.2.4 ganze Knotenmengen aus G entfernt werden können und die Anzahl der Intervalle in der Eingabe signifikant für die Laufzeit aller hier vorgestellten Algorithmen ist, wäre es interessant die Laufzeiten der Algorithmen auf die durch die Datenreduktion reduzierten Eingaben zu analysieren.

Zum Lösen von Inkrementelle Listenfärbung wurden fünf Algorithmen vorgestellt, zwei dynamische Programme mit Laufzeit $O(2^{|C|} \cdot n)$ und $O(2^Q \cdot n)$ sowie ein Suchabaumalgorithmus zum Lösen von Vollständig Farbenreiche Stabile Menge mit Laufzeit $O(\Gamma^{|C|} \cdot n)$, ein Suchbaumalgorithmus zum Lösen von Listenfärbung mit Laufzeit $O(|C|^n \cdot n)$ und ein Suchbaumalgorithmus zum Lösen von Inkrementelle Listenfärbung mit Laufzeit $O(|C|^n \cdot n)$. In der experimentellen Auswertung konnte mit Hilfe von künstlich erzeugten Testinstanzen die Abhängigkeit der Laufzeiten bezüglich der Anzahl der Intervalle festgestellt werden und es konnte keine Abhängigkeit bezüglich der Laufzeit und der Anzahl der Farben festgestellt werden. Der Vergleich zwischen den Laufzeiten des dynamischen Programms und den Suchbaumalgorithmen ergab, dass die Suchbaumalgorithmen für Instanzen mit 15 Intervallen um den Faktor 100 bis 1000 schneller sind als das dynamische Programm. Ein weiteres interessantes Ergebnis war, dass der Suchbaumalgorithmus zum Lösen von Inkrementelle Listenfärbung für fast jede Testinstanz langsamer ist als der Suchbaumalgorithmus zum Lösen von Listenfärbung. Hieraus lässt sich schlussfolgern, dass für fast alle Instanzen die Lösung von der Teillösung stark abweicht und somit die Teillösung außer für die Datenreduktion kaum einen Mehrwert bietet.

Für weitere Arbeiten wäre es von Interesse das Problem Inkremetelle Listenfärbung um Prioritäten zu erweitern, da Fahrzeuge beim Coorporate Car Sharing meistens bezüglich des Kauf- oder Leasingverhältnisses zugewiesen werden, um Mehrkosten zu verhindern. Da immer häufiger Elektroautos als Betriebsfahrzeuge genutzt werden,

wäre eine Erweiterung des Problems, welche Elektroautos abbilden könnte, von großer Bedeutung. Aufschlussreich wäre auch eine Analyse von großen echten Datenmengen, um weitere Datenreduktionen bestimmen zu können oder „datengetriebene“ Parameter für FPT-Algorithmen zu finden.

Literatur

[Bev+15] R. van Bevern, M. Mnich, R. Niedermeier und M. Weller. "Interval scheduling and colorful independent sets". In: *Journal of Scheduling* 18.5 (2015), S. 449–469. URL: https://doi.org/10.1007/s10951-014-0398-5 (siehe S. 10, 14, 15, 35, 60–62).

[CLR09] T. H. Cormen, C. E. Leiserson und R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2009 (siehe S. 33).

[Die16] R. Diestel. *Graph Theory, 5th Edition*. Graduate Texts in Mathematics. Springer, 2016 (siehe S. 13).

[ERT79] P. Erdős, A. Rubin und H. Taylor. "Choosability in graphs". In: *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing*. 1979, S. 125–157 (siehe S. 9).

[Kar72] R. M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*. Hrsg. von R. E. Miller, J. W. Thatcher und J. D. Bohlinger. Boston, MA: Springer US, 1972, S. 85–103. URL: https://doi.org/10.1007/978-1-4684-2001-2_9 (siehe S. 19, 21, 22, 24, 29, 30).

[Kol+07] A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou und F. C. Spieksma. "Interval scheduling: A survey". In: *Naval Research Logistics (NRL)* 54.5 (2007), S. 530–543. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.20231 (siehe S. 9).

[Ram+06] K. N. Ramachandran, E. M. Belding, K. C. Almeroth und M. M. Buddhikot. "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks". In: *Proceedings IEEE INFOCOM 2006. 25th IEEE International Conference on Computer Communications*. 2006, S. 1–12 (siehe S. 10).

[ZW03] T. Zeitlhofer und B. Wess. "List-coloring of interval graphs with application to register assignment for heterogeneous register-set architectures". In: *Signal Processing* 83.7 (2003), S. 1411 –1425. URL: http://www.sciencedirect.com/science/article/pii/S0165168403000896 (siehe S. 10).