

DIPLOMARBEIT

---

# Das Population-Haplotyping-Problem

Graphentheoretische Ansätze

---

MARIA KOCH

TU Bergakademie Freiberg, 2008

## Danksagung

Für seine Unterstützung und die vielen wertvollen Ratschläge und Hinweise möchte ich besonders Herrn Prof. Dr. Ingo Schiermeyer sehr danken.

Außerdem danke ich meinen Eltern für ihre Hilfe und den Rückhalt, den sie mir während der letzten Jahre geboten haben.

Ich danke meinem HERRN, der alles wirkt.

Freiberg, März 2008

Maria Koch

# Inhaltsverzeichnis

|   |           |
|---|-----------|
| <b>Abbildungsverzeichnis</b>  | <b>4</b>  |
| <b>Tabellenverzeichnis</b>  | <b>6</b>  |
| <b>1 Einführung</b>   | <b>7</b>  |
| 1.1 Biologische Hintergründe und Anwendung . . . . .                      | 7         |
| 1.2 Aufbau der Arbeit . . . . .   | 9         |
| <b>2 Ein mathematisches Modell für das Population-Haplotyping-Problem</b> | <b>10</b> |
| 2.1 Definitionen und Bezeichnungen . . . . .                              | 10        |
| 2.2 Das Problem . . . . .   | 14        |
| 2.3 Beschreibung des mathematischen Modells . . . . .                     | 15        |
| 2.3.1 Zwei Beispiele . . . . .  | 16        |
| 2.3.2 Erste Schranken . . . . .   | 17        |
| 2.4 Komplexität . . . . .   | 18        |
| 2.4.1 Komplexität im allgemeinen Fall . . . . .                           | 18        |
| 2.4.2 Komplexität des PHP für $(k,l)$ -beschränkte Eingaben . . . . .     | 21        |
| 2.4.3 Beschränkung der Anzahl der mehrdeutigen Stellen . . . . .          | 22        |
| <b>3 Exkurs: Clarks Rule</b>  | <b>26</b> |
| 3.1 Der Algorithmus . . . . .   | 26        |
| 3.2 Mögliche Probleme . . . . .   | 28        |

---

|          |  |           |
|----------|--|-----------|
| <b>4</b> | <b>Lösungsansätze aus der Graphentheorie</b>             | <b>30</b> |
| 4.1      | Clark-Konsistenzgraphen . . . . .                        | 30        |
| 4.1.1    | Die Idee . . . . .                                       | 30        |
| 4.1.2    | Spezialfall Cliquen . . . . .                            | 32        |
| 4.1.3    | Spezialfall Graphen mit beschränkter Baumweite . . . . . | 34        |
| 4.1.4    | Spezialfall Bipartite Graphen . . . . .                  | 36        |
| 4.2      | Pedigree-Graphen . . . . .                               | 37        |
| 4.3      | Fragment-Konflikt-Graphen . . . . .                      | 39        |
| <b>5</b> | <b>Minimal Rainbow-Subgraph Problem MRS</b>              | <b>43</b> |
| 5.1      | Das Modell . . . . .                                     | 43        |
| 5.2      | MRS auf speziellen Graphenklassen . . . . .              | 44        |
| 5.2.1    | Beschränkung der auftretenden Farben . . . . .           | 45        |
| 5.2.2    | Sterne . . . . .   | 48        |
| 5.2.3    | Reguläre Graphen . . . . .                               | 48        |
| 5.2.4    | Wege . . . . .   | 50        |
| 5.3      | Ein Greedy-Algorithmus . . . . .                         | 51        |
| 5.3.1    | Der Algorithmus . . . . .                                | 51        |
| 5.3.2    | Ein Beispiel . . . . .                                   | 53        |
| 5.3.3    | Analyse des Greedy-Algorithmus . . . . .                 | 54        |
| 5.4      | Ein weiterer approximativer Algorithmus . . . . .        | 57        |
| 5.4.1    | Der Algorithmus . . . . .                                | 57        |
| 5.4.2    | Analyse . . . . .  | 58        |
| 5.4.3    | Verbesserung der Lösung . . . . .                        | 58        |
| 5.5      | MRS mit Schlingen . . . . .                              | 59        |
| <b>6</b> | <b>Ausblick</b>  | <b>63</b> |
|          | <b>Literaturverzeichnis</b>                              | <b>64</b> |

# Abbildungsverzeichnis

|      |   |    |
|------|---|----|
| 2.1  | Ein 6-Stern mit Mittelpunktsknoten $v$ . . . . .                                | 11 |
| 2.2  | Beispiele für Regenbogenfärbungen . . . . .                                     | 13 |
| 2.3  | Ein Beispiel . . . . .  | 20 |
| 2.4  | Der Graph $G_g$ . . . . .   | 23 |
| 3.1  | Beispiel für Clarks Rule . . . . .  | 28 |
| 4.1  | Der Clark-Konsistenzgraph von Beispiel 2 . . . . .                              | 32 |
| 4.2  | Ein Beispiel: links nach Definition 4.6 und rechts nach Definition 4.7. . . . . | 38 |
| 4.3  | Ein zweites Beispiel mit einem Kreis . . . . .                                  | 39 |
| 4.4  | Der Fragment-Konflikt-Graph . . . . .   | 42 |
| 5.1  | Zulässig 1-gefärbter Graph . . . . .  | 45 |
| 5.2  | zulässig 2-gefärbter Graph . . . . .  | 46 |
| 5.3  | Der Beispielgraph $G$ . . . . .   | 53 |
| 5.4  | Der erste Durchlauf des Algorithmus . . . . .                                   | 53 |
| 5.5  | Der Graph $G^*$ nach dem ersten Durchlauf . . . . .                             | 54 |
| 5.6  | Der zweite Durchlauf des Algorithmus . . . . .                                  | 54 |
| 5.7  | Der Graph $G$ und die Menge $V^*$ (rot) . . . . .                               | 55 |
| 5.8  | Beispiel . . . . .  | 56 |
| 5.9  | Ein worst-case-Beispiel . . . . .   | 58 |
| 5.10 | Instanz für das MRS mit Schlingen . . . . .                                     | 60 |

---

|   |    |
|---|----|
| 5.11 Instanz des MRS mit obiger Zusatzannahme . . . . . | 61 |
|---|----|

# Tabellenverzeichnis

|     |  |    |
|-----|--|----|
| 3.1 | Darstellung der Genome . . . . .                                 | 28 |
| 4.1 | konsistente Haplotypen . . . . .                                 | 31 |
| 4.2 | Die vier Fragmente . . . . .                                     | 40 |
| 4.3 | Die SNPs (rot) auf den Fragmenten . . . . .                      | 40 |
| 4.4 | Aufteilung der Fragmente . . . . .                               | 40 |
| 4.5 | Instanz mit neun Fragmenten . . . . .                            | 41 |
| 5.1 | Analyse der Laufzeit der einzelnen Algorithmenschritte . . . . . | 55 |

# 1 Einführung

Die Entschlüsselung des menschlichen Genoms im Rahmen des Humangenomprojektes hat sehr viel Aufmerksamkeit erregt. Schon seit Beginn der Arbeit wurden auch mathematische Methoden verwendet. Diese Diplomarbeit beschäftigt sich mit der Anwendung von verschiedenen graphentheoretischen Modellen auf ein weiterführendes Problem. Die zu Grunde liegenden Sachverhalte werden im folgenden Abschnitt dargestellt.

## 1.1 Biologische Hintergründe und Anwendung

Das menschliche Genom kann als Folge über dem Alphabet  $\{G, C, A, T\}$  betrachtet werden. Nach [Lan04] sind 99% der Stellen in einem Genom bei allen Menschen identisch. An den anderen Stellen kann es mehrere Möglichkeiten geben. Ein *single nucleotide polymorphism* (SNP) oder eine Punktmutation ist eine solche Stelle in diesem Genom, an der mit einer Wahrscheinlichkeit von mindestens 5% verschiedene Belegungen existieren (andere Quellen legen eine andere Grenze fest, z.B. [Bon03] mit 10%). Das menschliche Genom ist *diploid*, d.h. die menschliche DNA besteht aus Paaren von Chromosomen, jeweils einem väterlichen und einem mütterlichen. In Bezug auf ein bestimmtes SNP ist ein Individuum entweder *homozygot*, d.h. das Genom hat an dieser Stelle in beiden Chromosomen den gleichen Wert, oder *heterozygot*, d.h. das Genom weist an dieser Stelle zwei unterschiedliche Werte auf. Ein *Haplotyp* bezeichnet die Werte der SNPs eines Chromosomensatzes, d.h. die komprimierte Darstellung des Chromosoms, bei der nur die Stellen betrachtet werden,

an denen möglicherweise eine Punktmutation auftritt. Die Genome sind durch die Forschungsbemühungen im Rahmen des Humangenomprojektes bekannt; das Ziel des *Haplotyping* ist die Bestimmung der Haplotypen, die eine gegebene Menge von Genomen erklären. Um einen Genotyp  $g$  zu erklären, müssen zwei Haplotypen  $h_1$  und  $h_2$  gefunden werden, die, wieder auf die zu Grunde liegenden Chromosomen zurückgeführt, genau das gesuchte Genom ergeben. Ich verwende die folgende Notation:  $g = h_1 \otimes h_2$ .

Beim allgemeinen *population haplotyping problem* ist für eine gegebene Menge  $\mathcal{G}$  von Genomen eine Menge  $\mathcal{H}$  von Haplotypen gesucht, so dass für alle Genome  $g$  aus  $\mathcal{G}$  zwei Haplotypen  $h_1$  und  $h_2$  aus  $\mathcal{H}$  existieren mit  $g = h_1 \otimes h_2$ . Das PHP kann unter verschiedenen Zielsetzungen oder Zielfunktionen betrachtet werden, von denen jetzt die in [Lan04] aufgeführten kurz angedeutet werden sollen:

- **Perfekte Phylogenese:** In diesem Modell muss die Lösung gewisse Anforderungen hinsichtlich des Stammbaumes erfüllen. Das heißt, es muss ein Baum existieren, in dem die Blätter die Haplotypen sind und jedes SNP eine Kante markiert. Das Entfernen der Kante, die mit dem SNP  $s$  gekennzeichnet ist, teilt den Baum in zwei Teile. Dann haben alle Haplotypen in derselben Baumhälfte den gleichen Wert auf  $s$ .
- **Clarks Rule:** In dieser Version des Problems wird sukzessiv eine Menge  $\mathcal{H}$  mit Hilfe der Clarks Rule konstruiert. Dieser Algorithmus wird in einem Exkurs in 3 beschrieben.
- **Pure Parsimony:** Hier ist das Ziel eine Menge  $\mathcal{H}$  minimaler Kardinalität zu finden. Diese Herangehensweise beruht auf der Annahme, dass unter allen Möglichkeiten, ein Phänomen zu erklären, die zu wählen ist, die die wenigsten Annahmen benötigt. Gesucht ist eine minimale Menge von Haplotypen, die alle beobachteten Genotypen erklären. Die vorliegende Arbeit beschränkt sich auf diese Variante des Problems.

Diese SNPs sind die vorherrschende Form der menschlichen Variabilität und deshalb von enormer Bedeutung für die Forschung, z.B. bei der Untersuchung von Medikamentenunverträglichkeiten.

## 1.2 Aufbau der Arbeit

Das zweite Kapitel dieser Arbeit beschreibt ein mathematisches Modell für das PHP und beschäftigt sich mit Aussagen zur Komplexität des Problems. Danach folgt ein Exkurs zu Clarks Rule. Im vierten Kapitel werden einige graphentheoretische Modelle und die daraus folgenden Resultate für das PHP vorgestellt. Kapitel fünf hat ein spezielles, in der Literatur noch nicht beschriebenes Modell, das Minimal-Rainbow-Subgraph-Problem, zum Thema. Das letzte Kapitel zeigt einige offene Probleme und mögliche weitere Forschungsrichtungen auf.

# 2 Ein mathematisches Modell für das Population-Haplotyping- Problem

In diesem Kapitel werden ein mathematisches Modell für das Population-Haplotyping-Problem (PHP) und die Komplexität des Problems im allgemeinen Fall und in einigen Spezialfällen betrachtet. Zu Beginn sollen einige Begriffe und Notationen eingeführt werden, die für die folgenden Betrachtungen relevant sind.

## 2.1 Definitionen und Bezeichnungen

### Graphen und Untergraphen

**Definition 2.1.** Ein Graph  $G$  ist ein geordnetes Paar  $(V(G), E(G))$  von disjunkten Mengen. Dabei ist  $V(G) \neq \emptyset$  die Menge von Knoten in  $G$  und  $E(G) \subseteq V(G)^2$  die Menge von Kanten  $G$ .  $|V(G)|$  wird Ordnung des Graphen  $G$  genannt.

**Definition 2.2.** Eine Kante  $e \in E$  heißt Schlinge, wenn sie zwei identische Endknoten hat, z.B.  $e = (x, x)$ ,  $x \in V(G)$ .

**Definition 2.3.** Ein einfacher Graph ist ein Graph ohne Schlingen und Mehrfachkanten.

**Definition 2.4.** Zwei Knoten heißen *adjazent*, wenn sie durch eine Kante verbunden sind. Eine Kante  $e$  und ein Knoten  $x$  heißen *inzident*, wenn  $x$  ein Endknoten der Kante  $e$  ist.

**Definition 2.5.** Die Nachbarschaft  $N(v)$  eines Knotens  $v$  ist die Menge der Kanten, die zu  $v$  adjazent sind.  $|N(v)|$  heißt Grad des Knotens  $v$ .

*Bezeichnung:*

Das Maximum aller Knotengrade wird mit  $\Delta(G)$  bezeichnet.

**Definition 2.6.** Ein Weg ist eine Folge von Knoten  $\{v_1, \dots, v_p\}$  in einem Graphen  $G$ , wobei immer  $v_i$  und  $v_{i+1}$  adjazent sein müssen für alle  $i = 1, \dots, p - 1$ . Dann ist  $p$  die Länge des Weges. Sind alle Knoten paarweise verschieden, so spricht man von einem Pfad.

Nun sollen einige spezielle Graphen definiert werden.

**Definition 2.7.** Ein Graph  $G$  heißt  $r$ -regulär, wenn alle seine Knoten den Grad  $r$  haben.

**Definition 2.8.** Ein zusammenhängender Graph mit  $n$  Knoten, bei dem  $n - 1$  Knoten den Grad eins haben, wird  $(n-1)$ -Stern genannt. Dabei sind die Knoten vom Grad eins die Außenknoten und der Knoten mit Grad  $\geq 1$  ist der Mittelpunkt.

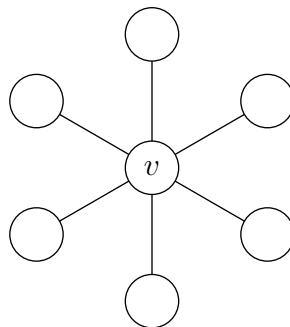


Abbildung 2.1: Ein 6-Stern mit Mittelpunktsknoten  $v$

**Definition 2.9.** Ein Graph  $G = (V, E)$  heißt *bipartit*, wenn eine Partition der Knotenmenge  $V$  in zwei Mengen  $U$  und  $V$  existiert, so dass alle Kanten  $e \in E$  einen Endpunkt in  $U$  und den anderen in  $W$  haben.

**Definition 2.10.** Ein Graph  $G = (V_u, E_u)$  ist ein Untergraph des Graphen  $G = (V, E)$ , wenn gilt:  $V_u \subseteq V$  und  $E_u \subseteq E$ .  $H$  ist ein induzierter Untergraph von  $G$ , wenn  $E_u$  aus all den Kanten besteht, die in  $G$  beide Endpunkte in  $V_u$  haben.

**Definition 2.11.** Ein zusammenhängender Graph, der keine Kreise enthält, heißt Baum. Ein Baum mit einem ausgezeichneten Knoten, der Wurzel, werden Wurzelbäume genannt. Dabei heißen Knoten vom Grad eins Blätter und Knoten von höherem Grad innere Knoten. Sei  $d(v)$  der Abstand eines Knotens zur Wurzel. Dann heißt  $u$  Kind von  $v$ , wenn  $d(v) = d(u) - 1$ .

## Kantenfärbungen

**Definition 2.12.** Eine Kantenfärbung  $\mathfrak{C}$  eines Graphen  $G$  ist eine Zuordnung  $\mathfrak{C} = E(G) \rightarrow C$ , wobei  $C$  die Menge von möglichen Farben ist.

**Definition 2.13.** Eine Kantenfärbung eines Graphen  $G=(V,E)$  heißt zulässig, wenn adjazente Kanten verschiedene Farbe haben.

**Definition 2.14.** Eine Kantenfärbung eines Graphen  $G = (V, E)$  heißt  $p$ -Kantenfärbung, wenn  $G$  mit genau  $p$  Farben gefärbt ist.

*Bezeichnung:*

Mit  $G = (V, E, p)$  werden Graphen mit Knotenmenge  $V$ , Kantenmenge  $E$  und einer zulässigen  $p$ -Kantenfärbung bezeichnet.

**Definition 2.15.** Eine Kantenfärbung eines (Unter-)Graphen  $G = (V, E, p)$  heißt  $p$ -Regenbogenfärbung, wenn jede der  $p$  Farben genau einmal auftritt.

**Definition 2.16.** Eine Kantenfärbung eines (Unter-)Graphen  $G = (V, E, p)$  heißt verallgemeinerte Regenbogenfärbung (Siehe Abbildung 2.2), wenn jede der  $p$  Farben mindestens einmal auftritt, d.h. es existiert eine Kantenmenge  $V' \subseteq V$  mit einer Regenbogenfärbung.

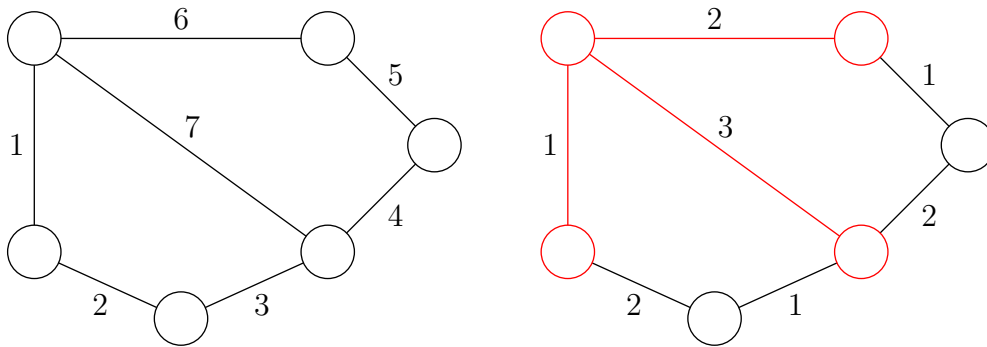


Abbildung 2.2: Beispiele für Regenbogenfärbungen

Der linke Graph besitzt eine Regenbogenfärbung mit sieben Farben und der rechte Graph eine verallgemeinerte Regenbogenfärbung mit drei Farben. Der rotmarkierte Teilgraph mit der Kantenmenge  $V^*$  hat eine (echte) Regenbogenfärbung.

*Bezeichnung:*

Die Menge der Graphen mit  $p$ -Regenbogenfärbung wird mit  $R_p$  und die Menge der Graphen mit verallgemeinerter  $p$ -Regenbogenfärbung mit  $R_p^*$  bezeichnet.

## Komplexität

**Definition 2.17.** Sei  $OPT(X)$  die optimale Lösung eines Problems und  $A(X)$  das Ergebnis eines Algorithmus  $A$  für eine Instanz  $X$ . Dann heißt

$$r_A = \frac{A(X)}{OPT(X)} \text{ für Minimierungsprobleme bzw. } r_A = \frac{OPT(X)}{A(X)} \text{ für Maximierungsprobleme.}$$

die Approximationsgüte des Algorithmus.

**Definition 2.18.** Die Laufzeit eines Algorithmus bezeichnet die maximale Rechenzeit, die der Algorithmus zur Lösung eines Problems benötigt.

**Definition 2.19.** NP bezeichnet die Klasse aller Entscheidungsprobleme, die von einer nichtdeterministischen Turingmaschine bezüglich der Eingabelänge in Polynomialzeit entschieden werden können.

**Definition 2.20.** *Ist ein Optimierungsproblem APX-schwer, dann existiert eine Konstante  $e > 0$ , so dass dieses Problem nicht mit Faktor  $1 + e$  approximiert werden kann, es sei denn, es wäre  $P = NP$ .*

**Bemerkung.** *In diesem Fall ist das Berechnen approximativer Lösungen einer bestimmten Güte genauso schwer wie das Finden optimaler Lösungen.*

**Definition 2.21.** *Ein Problem heißt fixed-parameter-tractable, wenn es sich durch Fixieren einer Eingabegröße effizient lösen lässt.*

**Definition 2.22.** *Sei  $G = (V, E)$  ein Graph und  $W \subseteq V$ . Dann ist  $W$  eine Knotenüberdeckung von  $G$ , wenn jede Kante  $e \in E$  mindestens einen Endpunkt in  $W$  hat. Das Ziel des Problems NODE-COVER ist es, eine minimale Menge  $W$  zu finden, die  $G$  überdeckt. Das Problem NODE-COVER ist für bipartite Graphen nach dem Satz von König (1931) in polynomieller Zeit lösbar.*

*Bezeichnung:*

$\sigma_G$  bezeichnet die Knotenüberdeckungszahl, d.h. die Größe einer minimalen Knotenüberdeckung, eines Graphen  $G$

**Definition 2.23.** *NC2 ist eine Beschränkung des Problems NODE-COVER auf Graphen  $G$  mit  $\sigma_G \geq \frac{n}{2}$*

## 2.2 Das Problem

Das Population-Haplotyping-Problem (PHP) kann auf die folgende Weise formalisiert werden:

**Eingabe:**

Eine Population, d.h. eine Menge  $\mathfrak{G}$  von Genomen.

**Ausgabe:**

Eine Menge  $\mathfrak{H}$  kleinster Kardinalität von Haplotypen, so dass gilt:

$$\forall g \in \mathfrak{G} \exists h_1, h_2 \in \mathfrak{H} : g = h_1 \otimes h_2. \quad (2.1)$$

**Bemerkung.** *In der Natur treten nicht alle theoretisch möglichen Haplotypen auch tatsächlich auf. Dieser Aspekt wird jedoch in den folgenden Betrachtungen keine Rolle spielen.*

## 2.3 Beschreibung des mathematischen Modells

Ein Haplotyp kann als binärer Vektor verstanden werden, wobei eine 1 bzw. eine 0 an der  $i$ -ten Stelle die beiden möglichen Belegungen des  $i$ -ten SNP kodiert.

Dementsprechend wird ein Genom durch einen Vektor derselben Länge kodiert. Dabei gilt: Wenn  $g = h_1 \otimes h_2$ , dann gilt für  $g$ , wenn  $g[i]$  die  $i$ -te Komponente des Vektors  $g$  bezeichnet,

$$g[i] = \begin{cases} h_1[i] & , \text{ falls } h_1[i] = h_2[i] \\ 2 & \text{sonst} \end{cases} \quad \text{für } i = 1, \dots, n, \quad (2.2)$$

wobei  $n$  die Anzahl der SNPs bezeichnet.

Mit dieser Notation kann das PHP auf die folgende Weise formuliert werden:

### **Eingabe:**

Eine  $(m \times n)$ -Matrix  $M$ , wobei  $m$  die Anzahl der Genome in der Population und  $n$  die Anzahl der SNPs darstellt. Jede Zeile der Matrix stellt ein Genom dar. Diese Notation ist vor allem sinnvoll für die Formulierung des Problems als Lineares Programm (Siehe [Lan04]).

### **Ausgabe:**

Eine  $(2m \times n)$ -Binärmatrix  $M'$ . Hierbei repräsentiert jede Zeile einen Haplotyp. Es besteht eine Korrespondenz zwischen den Zeilen der Genommatrix und jeweils zwei Zeilen der Haplotypenmatrix, d.h. jedes Genom korrespondiert zu den zwei Haplotypen, die das Genom erklären.

**Bemerkung.** *Wenn der Vektor zum Genom  $g$  nur 0- und 1-Einträge enthält, dann ist er identisch zu den Vektoren der beiden Haplotypen  $h_1$  und  $h_2$ , wenn gilt  $g = h_1 \otimes h_2$ .*

### 2.3.1 Zwei Beispiele

Durch die folgenden Beispiele soll die obige Beschreibung veranschaulicht werden:

#### Beispiel 1

$$\mathfrak{G} = \{(1201), (1010), (2000), (1002), (1000)\}$$

$$M = \begin{pmatrix} 1 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad M' = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Es gehören immer die zwei Zeilen zum selben Genom. Diese sind durch horizontale Linien von den anderen getrennt. Die Lösung ist in diesem Fall die Menge  $\mathfrak{H} = \{(1101), (1001), (1010), (0000), (1000)\}$ . Allgemein gilt  $|\mathfrak{H}| \leq 2|\mathfrak{G}|$ .

#### Beispiel 2 (aus [Lan04])

$$\mathfrak{G} = \{(0221), (0011), (2102), (2220)\}$$

$$M = \begin{pmatrix} 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix} \quad M'_1 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad M'_2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

In diesem zweiten Beispiel fällt eine Besonderheit im Vergleich zum ersten Beispiel auf. Die Darstellung der Genome durch Haplotypen und damit auch die Lösung ist nicht eindeutig. Es gibt mehrere Möglichkeiten, die Genome mit mehr als einer 2 darzustellen.

Zwei davon sind durch die Matrizen  $M'_1$  und  $M'_2$  angegeben. Die zugehörigen Lösungsmengen sind  $\mathfrak{H}_1 = \{(0101), (0011), (1100), (0010)\}$  und  $\mathfrak{H}_2 = \{(0101), (0011), (1100), (1110), (0000)\}$ .

**Definition 2.24.** *Eine mehrdeutige Stelle in einem Genom ist eine Position, die in der Vektordarstellung durch eine 2 dargestellt wird.*

**Definition 2.25.** *Ein Haplotyp  $h$  und ein Genom  $g$  sind konsistent, wenn sie an allen Stellen identisch sind, die keine mehrdeutigen Stellen von  $g$  sind.*

### 2.3.2 Erste Schranken

Sei  $OPT$  die minimale Kardinalität einer Menge  $\mathfrak{H}$ , die Eigenschaft 2.1 erfüllt. Dann können mit Hilfe einiger einfacher Überlegungen, die in [Lan04] beschrieben werden, erste Schranken für den Wert von  $OPT$  aufgestellt werden.

- **obere Schranke:**  $OPT \leq 2|\mathfrak{G}|$

Für eine gegebene Menge  $\mathfrak{G}$  kann auf die folgende Weise eine Lösung  $\mathfrak{H}$  mit

$OPT = |\mathfrak{H}| \leq 2|\mathfrak{G}|$  konstruiert werden. Für ein gegebenes Genom  $g \in \mathfrak{G}$ , sei  $h_1$  der Haplotyp, der überall null ist, wo  $g$  null ist und sonst eins und  $h_2$  der Haplotyp, der überall eins ist, wo  $g$  eins ist und sonst null. Dann gilt  $g = h_1 \otimes h_2$ . Das bedeutet, dass im ungünstigsten Fall für jedes Genom ein neues Paar Haplotypen benötigt wird.

- **untere Schranke:**  $OPT > \sqrt{2|\mathfrak{G}|}$

Für eine Menge  $\mathfrak{H}$ , die eine gegebene Menge  $\mathfrak{G}$ , mit  $|\mathfrak{G}| > 1$ , erklärt, gilt:  $|\mathfrak{H}| > \sqrt{2|\mathfrak{G}|}$ , da zu jedem  $g \in \mathfrak{G}$  ein ungeordnetes Paar von Haplotypen zugeordnet werden kann, wenn  $\mathfrak{G}$  durch  $\mathfrak{H}$  erklärt wird. Damit gilt:  $|\mathfrak{G}| \leq \binom{|\mathfrak{H}|}{2} = \frac{|\mathfrak{H}|(|\mathfrak{H}|-1)}{2}$ .

**Bemerkung.** Die untere Schranke gilt nur, wenn  $\mathfrak{G}$  keine homozygoten Genome enthält, da nur dann für jedes Genom zwei verschiedene Haplotypen benötigt werden. Im allgemeinen Fall gilt  $|\mathfrak{G}| \leq \binom{|\mathfrak{H}|+1}{2} = \frac{(|\mathfrak{H}|+1)|\mathfrak{H}|}{2}$ . Daraus folgt

$$OPT \geq -\frac{1}{2} + \sqrt{\frac{1}{4} + 2|\mathfrak{G}|}$$

## 2.4 Komplexität

### 2.4.1 Komplexität im allgemeinen Fall

**Satz 2.1.** *PHP ist NP-schwer.*

ohne Beweis (Siehe [Sha05])

**Satz 2.2.** *PHP ist APX-schwer.*

*Beweisidee:*

Der exakte Beweis wird in [Lan04] ausgeführt.

Es wird eine zweistufige Reduktion von NODE-COVER durchgeführt.

### 1. Reduktion NODE-COVER auf NC2

Sei  $G = (V, E)$  ein einfacher ungerichteter Graph mit  $|V| = n$  und  $|E| = m$  und sei  $\sigma_G$  die Knotenüberdeckungsanzahl, also die minimale Kardinalität eines NODE-COVERS.

Mithilfe eines Satzes von Nemhauser und Trotter(1975) und einem polynomiellen Algorithmus NC2Reduktion (siehe [Lan04]) kann man zeigen, dass das Problem NC2 APX-schwer ist. Für einen gegebenen Graphen  $G = (V, E)$ , einer Instanz für NODE-COVER, wird ein Graph  $G' = (V', E')$  und eine Menge  $Q \subseteq V$  konstruiert, so dass gilt: Wenn  $X$  eine minimale Knotenüberdeckung von  $G'$  ist, dann ist  $X \cup Q$  eine minimale Knotenüberdeckung von  $G$ .

### 2. Reduktion NC2 auf PHP

Sei  $G = (V, E)$  eine Instanz für NC2. Es wird nun eine Matrix  $M$  mit  $m + n$  Zeilen (Genome) und  $n + 1$  Spalten konstruiert.  $\mathfrak{G}_M$  sei die Menge der Zeilen in  $M$ . Die Zeilen werden mit den Elementen aus  $V \cup E$  und die Spalten mit den Elementen aus  $V \cup \{s\}$  indiziert, wobei  $s \notin V$  ein zusätzliches Symbol ist. Dann werden die Einträge der Matrix wie folgt gebildet:

$$M[u, u] = 0, \text{ für alle } u \in V$$

$$M[u, v] = 1, \text{ für alle } u, v \in V \text{ mit } u \neq v$$

$$M[u, s] = 0, \text{ für alle } u \in V$$

$$M[e, v] = 2, \text{ für alle } u \in V \text{ mit } e \in E \text{ inzident zu } v$$

$$M[e, v] = 1, \text{ für alle } u \in V \text{ mit } e \in E \text{ nicht inzident zu } v$$

$$M[e, s] = 2, \text{ für alle } e \in E$$

An einem Beispiel soll die obige Konstruktion illustriert werden.

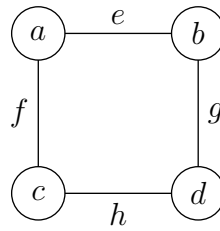


Abbildung 2.3: Ein Beispiel

Der Graph aus Abbildung 2.3 wird mit folgender Matrix assoziiert.

|     | $a$ | $b$ | $c$ | $d$ | $s$ |
|-----|-----|-----|-----|-----|-----|
| $a$ | 0   | 1   | 1   | 1   | 0   |
| $b$ | 1   | 0   | 1   | 1   | 0   |
| $c$ | 1   | 1   | 0   | 1   | 0   |
| $d$ | 1   | 1   | 1   | 0   | 0   |
| $e$ | 2   | 2   | 1   | 1   | 2   |
| $f$ | 2   | 1   | 2   | 1   | 2   |
| $g$ | 1   | 2   | 1   | 2   | 2   |
| $h$ | 1   | 1   | 2   | 2   | 2   |

Die folgenden beiden Aussagen rechtfertigen die Reduktion.

- Sei  $X$  ein Vertex-Cover von  $G$ . Dann gibt es ein  $\mathfrak{H}_X$ , dass  $\mathfrak{G}_M$  erklärt mit  $|\mathfrak{H}_X| = n + |X|$ .
- Sei  $\mathfrak{H}$  eine minimale Menge von Haplotypen, die  $\mathfrak{G}_M$  erklärt, so dass die Anzahl der 0en in den Haplotypen in  $\mathfrak{H}$  so klein wie möglich ist, dann besitzt  $G$  eine Knotenüberdeckung  $X_{\mathfrak{H}}$  mit  $|X_{\mathfrak{H}}| = |\mathfrak{H}| - n$ .

Zum Abschluss des Beweises ist noch zu zeigen, dass eine  $1 + \epsilon$ -Approximation des PHP eine  $1 + 3\epsilon$ -Approximation des NC2 impliziert, was ein Widerspruch ist zu NC2 APX-schwer.

□

## 2.4.2 Komplexität des PHP für $(k, l)$ -beschränkte Eingaben

Das allgemeine PHP ist NP-vollständig. In diesem Abschnitt wird die Komplexität des PHP auf einer speziellen Klasse von Funktionen betrachtet, die in [Sha05] wie folgt definiert wird. Dabei sind hier nur die Resultate und Beweisideen dargestellt.

**Definition 2.26.** *Eine Instanz für das PHP, heißt  $(k, l)$ -beschränkt, wenn die zugehörige Genommatrix  $M$  höchstens  $k$  2-Einträge pro Zeile und höchstens  $l$  2-Einträge pro Spalte aufweist. Eine Instanz heißt  $(\star, l)$ -beschränkt, wenn nur die Anzahl der 2-Einträge in den Spalten beschränkt wird und die Anzahl der 2-Einträge in den Zeilen beliebig hoch sein kann. Analog sind  $(k, \star)$ -beschränkte Instanzen definiert.*

### Beispiel 2 (Siehe Seite 16)

Die Instanz mit  $\mathfrak{G} = \{(0221), (0011), (2102), (2220)\}$  ist nach der obigen Definition  $(3, 2)$ -beschränkt.

**Satz 2.3.** *PHP ist NP-schwer für  $(4, 3)$ -beschränkte Eingabeinstanzen.*

**Definition 2.27.** *Das Problem des 3-dimensionalen Matching (3DM) ist wie folgt definiert.*

*Seien  $X$ ,  $Y$  und  $Z$  disjunkte Mengen mit jeweils  $\nu$  Elementen und sei  $C = \{c_1, \dots, c_\mu\}$  eine Menge von  $\mu$  Tripeln auf der Menge  $X \times Y \times Z$ . Gesucht ist dann eine Menge  $C' \subseteq C$  maximaler Kardinalität die paarweise disjunkte Tripel enthält.*

*Beweisidee:*

Es wird eine Reduktion auf 3DM durchgeführt, bei denen jedes Element in höchstens drei Tripeln auftaucht (3DM3). Dabei wird eine Matrix aus einer gegebenen Instanz für (3DM3) konstruiert, die die Genommatrix für eine  $(4, 3)$ -beschränkte Eingabe des PHP bildet.

**Satz 2.4.** *PHP ist APX-schwer für  $(4,3)$ -beschränkte Eingabeinstanzen.*

*(ohne Beweis)*

In [Ies06] wird noch ein weiterer Fall vorgestellt.

**Satz 2.5.** *PHP ist NP-schwer für  $(3,3)$ -beschränkte Eingabeinstanzen.*

*Beweisidee:*

Der Beweis erfolgt mit Hilfe einer ähnlichen Reduktion wie der von Satz 2.3.

**Satz 2.6.** *PHP ist fixed parameter tractable in der Zahl der Haplotypen.*

*Beweisidee:*

Das Problem kann bei fester Anzahl von Haplotypen auf das 2-SAT-Problem zurückgeführt werden. Der Beweis befindet sich in [Sha05].

### 2.4.3 Beschränkung der Anzahl der mehrdeutigen Stellen

Die Ideen in diesem Kapitel stammen aus [Lan06].

Es sei  $\mathcal{G}$  eine Menge von Genomen und  $\mathfrak{H}_{\mathcal{G}}$  die Menge von Haplotypen, die mit Elementen aus  $\mathcal{G}$  konsistent sind.

**Bemerkung.** *Sei  $\mathfrak{H}$  die minimale Menge von Haplotypen, die  $\mathcal{G}$  erklären. Dann gilt  $\mathfrak{H} \subseteq \mathfrak{H}_{\mathcal{G}}$ .*

**Bemerkung.** *Wenn jedes Genom in  $\mathcal{G}$  maximal  $k$  mehrdeutige Stellen enthält, dann gilt  $|\mathfrak{H}_{\mathcal{G}}| \leq 2^k |\mathcal{G}|$ .*

*Bezeichnung:*

Sei  $g$  ein Genom mit genau  $t$  ( $t \leq k$ ) mehrdeutigen Stellen und  $s$  ein binärer Vektor der Länge  $t$ . Mit  $g(s)$  wird dann der Haplotyp bezeichnet, der aus  $g$  entsteht, wenn für alle  $i \in \{1, \dots, t\}$  das  $i$ -te 2-Symbol durch die  $i$ -te Komponente von  $s$  ersetzt wird.

### Ein polynomieller Fall: maximal 2 mehrdeutige Stellen

Das Problem wird reduziert auf die Suche nach einer Knotenüberdeckung in einem bipartiten Graphen, was effektiv möglich ist.

Zu Beginn wird der Fall behandelt, in dem jedes Genom genau zwei mehrdeutige Stellen aufweist. Die entstehenden Resultate können leicht auf den allgemeinen Fall übertragen werden. Für jedes Genom  $g \in \mathfrak{G}$  wird ein Graph  $G_g = (V_g, E_g)$  (Siehe Abbildung 2.4) betrachtet, wobei gilt

$$V_g = \{g(00), g(01), g(10), g(11)\} \text{ und}$$

$$E_g = \{g(00)g(01), g(01)g(11), g(11)g(10), g(10)g(00)\}.$$

Jeder Knoten in Abbildung 2.4 repräsentiert einen Haplotypen.

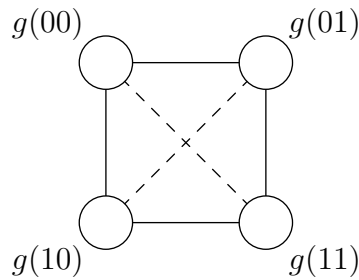


Abbildung 2.4: Der Graph  $G_g$

**Bemerkung.** Jede Knotenüberdeckung benutzt mindestens zwei gegenüberliegende Knoten des Quadrats (In Abbildung 2.4 durch gepunktete Linien verbunden).

Sei nun  $G = (V, E)$  der Graph mit

$$V = \bigcup_{g \in \mathfrak{G}} V_g \text{ und } E = \bigcup_{g \in \mathfrak{G}} E_g.$$

**Satz 2.7.** Sei  $\mathfrak{H}$  eine Teilmenge von  $\mathfrak{H}_{\mathfrak{G}}$ . Dann wird  $\mathfrak{G}$  durch  $\mathfrak{H}$  genau dann erklärt, wenn  $\mathfrak{H}$  eine Knotenüberdeckung von  $G$  darstellt.

*Beweis:*

Der Beweis wird in zwei Schritten geführt.

- Angenommen,  $\mathfrak{H}$  ist eine Knotenüberdeckung von  $G$ . Dann ist für alle Genome  $g$ ,  $\mathfrak{H}$  ein Vertex-Cover für  $G_g$ . Daraus folgt, dass  $\mathfrak{H}$  jedes Genom  $g \in G$  erklärt.
- Angenommen,  $\mathfrak{H}$  erklärt  $g$ . Das heißt, dass entweder  $g(00), g(11) \in H$  oder  $g(10), g(01) \in H$  sind. Dann überdeckt  $\mathfrak{H}$   $E_g$ , und da  $\mathfrak{H}$  alle Genome  $g \in G$  erklärt, folgt, dass  $\mathfrak{H}$   $E = \bigcup_{g \in \mathfrak{G}} E_g$  überdeckt.

□

**Satz 2.8.** *Der Graph  $G$  ist bipartit.*

*Beweis:*

Der Graph  $G$  ist ein Untergraph des Hyperwürfels, da zwei Knoten genau dann adjazent sind, wenn sie sich an genau einer Stelle unterscheiden, und ist damit bipartit. □

Da NODE-COVER für bipartite Graphen ein polynomielles Problem ist, ist auch das PHP in diesem Fall polynomiell lösbar.

Diese Erkenntnisse können leicht übertragen werden auf Instanzen mit Genomen mit höchstens zwei mehrdeutigen Stellen. Falls  $g$  keine mehrdeutigen Stellen hat, dann muss der Haplotyp  $h = g$  auf jeden Fall in der Lösung sein. Das entspricht der Suche nach einer Knotenüberdeckung  $X$  in  $G - \{g\}$ . Die Lösung ist dann  $X \cup \{h\}$ . Falls  $g$  genau eine mehrdeutige Stelle hat, dann müssen die beiden Haplotypen  $h_0 = g(0)$  und  $h_1 = g(1)$  beide in die Lösung aufgenommen werden. Das entspricht der Suche nach einer Knotenüberdeckung  $X$  in  $G - \{h_0, h_1\}$ . Zur Lösung  $X$  muss dann noch  $\{h_0, h_1\}$  hinzugefügt werden.

**Bemerkung.** *Das Problem kann in  $O(n|G| + |G|^{\frac{3}{2}})$  gelöst werden.*

**k mehrdeutige Stellen**

Für den Fall  $k > 2$  ist in [Lan06] ein Algorithmus angegeben, der eine  $2^{k+1}$ -Approximation an die Optimallösung liefert. Dieser basiert auf einer Verallgemeinerung der Überlegungen für den Fall  $k = 2$ . Dabei wird aus den Beziehungen der zu  $\mathcal{G}$  konsistenten Haplotypen ein Hypergraph konstruiert und das Problem auf Knotenüberdeckungen in Hypergraphen zurückgeführt. Die Benutzung effizienter Algorithmen der dualen linearen Programmierung liefert die angegebene Approximationsgüte.

## 3 Exkurs: Clarks Rule

Wie in der Einführung schon erwähnt, wird sich das folgende Kapitel mit einem Algorithmus von Andrew Clark, *Clarks Rule*, beschäftigen. Er wird in [Cla90] beschrieben. Die grobe Vorgehensweise und möglicherweise auftretende Probleme sollen hier dargelegt werden.

### 3.1 Der Algorithmus

Gegeben sei eine Population in Form einer Menge  $\mathcal{G}$  von Genomen. Dann gibt es zwei Gruppen von Haplotypen, die auf jeden Fall in der Lösungsmenge  $\mathcal{H}$  enthalten sein müssen.

1. Homozygote Genome sind mit ihrem erzeugenden Haplotyp identisch und dieser muss in  $\mathcal{H}$  enthalten sein.
2. Genome mit nur einer mehrdeutigen Stelle können nur durch ein eindeutig bestimmtes Paar von Haplotypen dargestellt werden und damit müssen diese beiden Haplotypen zwingend Elemente von  $\mathcal{H}$  sein.

Das Ziel besteht darin, mit möglichst wenigen Haplotypen die Genome der Population zu erklären; daher ist es sinnvoll zu versuchen, die schon identifizierten Haplotypen zur Erzeugung anderer Genome erneut zu verwenden und auf diese Weise mit Hilfe einer Kette, einem *Clark-Pfad*, die Genome der Population zu erzeugen.

Diese Überlegungen führen zu folgender Vorgehensweise, einem Algorithmus, der als Clarks Rule bezeichnet wird.

1. Identifiziere die homozygoten und eindeutigen heterozygoten Genome. Diese können als gelöst betrachtet werden, da deren Darstellung mit Haplotypen eindeutig bestimmt ist und damit die beteiligten Haplotypen auf jeden Fall in die Lösung gehören.
2. Bestimme, ob einer der bestimmten Haplotypen, zur Erklärung eines weiteren (noch nicht erklärten) Genoms benutzt werden kann. Wenn nicht, endet der Algorithmus, sonst geht es mit Schritt drei weiter.
3. Bestimme das entsprechende Gegenstück, das das identifizierte Genom ergänzt.

Diese Schritte werden wiederholt, bis entweder alle Genome erklärt sind oder alle Clark-Pfade zu Ende sind.

### Beispiel

Betrachtet wird eine Population mit einer Menge von Genomen  $\mathfrak{G} = \{A, B, C, D, E, F, G\}$ .

Kleine Buchstaben bezeichnen in diesem Beispiel Haplotypen. Dabei sei  $A$  ein homozygoten Genom, was als  $A = a \otimes a$  darstellbar ist.  $B$  sei ein heterozygoten Genom mit möglicherweise sehr vielen mehrdeutigen Stellen und damit sehr vielen Darstellungsmöglichkeiten. Eine dieser Möglichkeiten der Darstellung ist  $B = a \otimes b$ . Für die restlichen Genome gibt Tabelle 3.1 die angenommenen Möglichkeiten der Darstellung an. Für Genome mit mehreren mehrdeutigen Stellen gibt es mehrere Möglichkeiten der Darstellung. Die in der Tabelle angegebenen Kombinationen sollen jeweils in der Menge der Möglichen enthalten sein. Mit Hilfe dieser Angaben entsteht die Abbildung 3.1. Diese gibt den aus diesem Beispiel entstehenden Clark-Pfad wieder. Das Genom  $G$  ist nicht mit dem Clark-Pfad verbunden und wird vom Algorithmus übersehen (Siehe Abschnitt 3.2).

Tabelle 3.1: Darstellung der Genome

| Genom | mögliche Darstellung |
|-------|----------------------|
| A     | $a \otimes a$        |
| B     | $a \otimes b$        |
| C     | $b \otimes c$        |
| D     | $b \otimes d$        |
| E     | $d \otimes e$        |
| F     | $d \otimes f$        |
| G     | $g \otimes h$        |

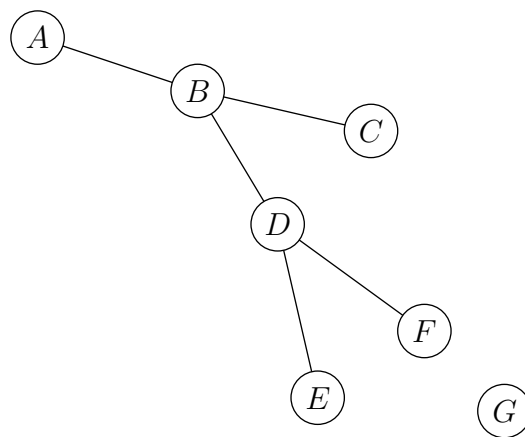


Abbildung 3.1: Beispiel für Clarks Rule

## 3.2 Mögliche Probleme

Beim Durchlauf des Algorithmus können verschiedene Probleme auftreten, die die Ergebnisse verfälschen oder den Algorithmus unwirksam machen können. Diese werden in [Cla90] beschrieben, außerdem werden Untersuchungen über die Wahrscheinlichkeit des Auftretens angestellt.

- Wenn keine homozygoten Genome oder Genome mit nur einer mehrdeutigen Stelle in der Population vorhanden sind, dann findet der Algorithmus keinen Startpunkt und scheitert deshalb.
- Es ist möglich, dass am Ende des Algorithmus Genome übrigbleiben, die von keinem Clark-Pfad erreicht werden können.
- Ein Haplotyp kann fälschlicherweise in einen Clark-Pfad aufgenommen werden.

Die Wahrscheinlichkeiten für das Auftreten dieser Ereignisse können berechnet werden und hängen von der Länge der DNA-Sequenz, der durchschnittlichen Anzahl heterozygoter Stellen pro Nukleotid, der Populationsgröße und der Neukombinationsrate ab.

# 4 Lösungsansätze aus der Graphentheorie

In der Literatur existieren sehr viele verschiedene Lösungsansätze für das PHP, die Methoden der Graphentheorie verwenden. Einige davon werden in diesem Kapitel vorgestellt.

## 4.1 Clark-Konsistenzgraphen

Der erste Ansatz zur Lösung des PHP, der in [Sha05] beschrieben ist, benutzt Clark-Konsistenzgraphen. Dazu werden einige zusätzliche Definitionen benötigt.

**Definition 4.1.** *Ein Haplotyp, der mit zwei verschiedenen Genomen konsistent ist, heißt gemeinsamer Haplotyp der beiden Genome.*

### 4.1.1 Die Idee

Mit Hilfe dieser Notation kann der folgende Graph definiert werden.

**Definition 4.2.** *Ein Graph heißt Clark-Konsistenzgraph, wenn er auf folgende Weise aus dem PHP konstruiert wird: Die Knoten sind Genome und zwei Knoten werden genau dann verbunden, wenn sie einen gemeinsamen Haplotyp besitzen.*

**Bemerkung.** Die Bezeichnung Clark-Konsistenzgraph kommt von der inhaltlichen Verwandtschaft zu Clarks Rule, die in Kapitel 3 beschrieben wurde. Pfade im Clark-Konsistenzgraphen entsprechen den in Kapitel 3 erklärten Clark-Pfaden.

### Beispiel 2 (Siehe Seite 16)

$$\mathfrak{G} = \{(0221), (0011), (2102), (2220)\}$$

In der folgenden Tabelle sind alle konsistenten Haplotypen zu den Genomen aus  $\mathfrak{G}$  aufgelistet. Die Zahlen hinter einigen Haplotypbezeichnungen kennzeichnen die mehrfach vorkommenden Haplotypen.

Tabelle 4.1: konsistente Haplotypen

| <b>Genom</b> | <b>konsistente Haplotypen</b>  |
|--------------|--|
| (0221)       | (0111)<br>(0001)<br>(0101) - 1<br>(0011) - 2                                     |
| (0011)       | (0011) - 2   |
| (2102)       | (1101)<br>(0100)<br>(1100) - 3<br>(0101) - 1                                     |
| (2220)       | (0000)<br>(0010)<br>(0100)<br>(0110)<br>(1000)<br>(1010)<br>(1100) - 3<br>(1110) |

Anhand dieser Tabelle kann der Konsistenzgraph konstruiert werden (Abbildung 4.1).

Mit Hilfe der Clark-Konsistenzgraphen ist es möglich, für spezielle Instanzen bessere Lösungen als die in Abschnitt 2.4 gezeigten zu erzielen. Im Folgenden werden die Resultate aus [Sha05] und [Ies06] inklusive der Beweisideen vorgestellt.

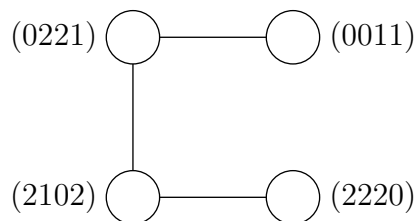


Abbildung 4.1: Der Clark-Konsistenzgraph von Beispiel 2

### 4.1.2 Spezialfall Cliques

Die erste Klasse von Instanzen, die unter bestimmten Voraussetzungen eine Verbesserung der Komplexität zulässt, sind Instanzen, deren Clark-Konsistenzgraphen Cliques bilden. Solche Graphen entstehen, wenn jedes beliebige Paar von Genomen einen gemeinsamen Haplotyp hat. Das bedeutet, dass die zugehörige Genommatrix o.B.d.A. nur 0- und 2-Einträge enthält. Speziell wird der Haplotyp, der nur Nullen enthält, *trivial* genannt, denn er ist allen Genomen gemeinsam. Es wird angenommen, dass der Genotyp, der nur Nullen enthält, nicht auftritt.

**Satz 4.1.** *PHP bleibt NP-schwer für Cliqueninstanzen.*

*Beweisidee:*

Der Beweis basiert auf einer ähnlichen Konstruktion, wie Satz 2.3. □

**Satz 4.2.** *PHP ist polynomiell lösbar für  $(\star, 2)$ -beschränkte Cliqueninstanzen.*

**Bemerkung.** *Der Clark-Pfad in einer  $(\star, 2)$ -beschränkten Cliqueninstanz ist eindeutig bestimmt, wenn seine Konstruktion beendet wird, sobald der triviale Haplotyp hinzugefügt wird.*

**Definition 4.3.** *In einer  $(\star, 2)$ -beschränkten Cliqueninstanz heißt folgende Konstruktion Cliquen-Clark-Pfad zum Haplotypen  $h$ : Wenn der Haplotyp  $h$  nur mit einem einzigen Genom  $g$  konsistent ist, dann ist der Cliquen-Clark-Pfad der Clark-Pfad, der den trivialen Haplotyp meidet und bei  $g$  startet. Wenn  $h$  zu zwei Genomen  $g_1$  und  $g_2$  konsistent ist, dann wird jeweils ein Clark-Pfad von  $g_1$  und  $g_2$  aus gebildet, der den trivialen Haplotyp vermeidet, und durch eine zusätzliche Kante zwischen  $g_1$  und  $g_2$  werden die beiden Pfade verbunden.*

**Bemerkung.** *Der Cliquen-Clark-Pfad kann auch einen Kreis bilden.*

*Beweisidee:*

Folgende Beobachtungen können festgehalten werden:

1. In einer  $(\star, k)$ -beschränkten Cliqueninstanz ist jeder nichttriviale Haplotyp gemeinsamer Haplotyp von höchstens  $k$  Genotypen.
2. Jede Lösung einer  $(\star, k)$ -beschränkten Cliqueninstanz enthält mindestens  $\frac{2n}{k+1} + 1$  Elemente. Wenn der triviale Haplotyp nicht in der optimalen Lösung enthalten ist, dann hat diese eine Kardinalität von wenigstens  $\frac{2n}{k}$ .
3. Für  $(\star, k)$ -beschränkte Cliqueninstanzen ist jede Lösung eine  $(k+1)$ -Approximation an die optimale Lösung.

Mit Hilfe dieser Beobachtungen ist es nun möglich einen polynomiellen Algorithmus für  $(\star, 2)$ -beschränkte Cliqueninstanzen zu konstruieren. Dabei gelten folgende Schranken:

$$\frac{2n}{3} \stackrel{\text{Beobachtung 2}}{\leq} OPT \leq n + 1$$

Das Problem wird reduziert auf die Suche nach Clark-Kreisen, die mit dem folgenden Algorithmus in Polynomzeit möglich ist. Die Genotypen, die einen Cliquen-Clark-Kreis der Länge  $k$  enthalten, können mit  $k$  Haplotypen dargestellt werden. Der untenstehende Algorithmus findet alle Cliquen-Clark-Kreise im Clark-Konsistenzgraphen. Alle übrigen Genome (die in keinem Kreis enthalten sind) werden durch den trivialen Haplotypen und einen zusätzlichen Haplotypen für jeden verbleibenden Genotypen dargestellt.

Die folgenden vier Schritte werden so lange wiederholt, bis alle Genome entfernt sind, die gemeinsame Haplotypen besitzen. Dadurch werden alle Cliquen-Clark-Kreise in  $\mathfrak{G}$  gefunden.

1. Wähle zwei Genome  $g_1, g_2$ , die gemeinsame Haplotypen besitzen.
2. Sei  $h$  der Haplotyp mit  $h[i] = 1$ , wenn  $g_1[i] = g_2[i] = 2$ .
3. Konstruiere den Clark-Pfad von  $h$ .

4. Wenn dies ein Kreis ist, füge alle gefundenen Haplotypen zur optimalen Lösung hinzu und lösche alle gefundenen Genotypen.

Die Korrektheit des Algorithmus geht aus den folgenden drei Aussagen hervor.

- Sei  $\mathcal{G}$  eine  $(\star, 2)$ -beschränkte Cliquesinstanz und  $g, g'$  und  $g''$  drei Genome von  $\mathcal{G}$ . Dabei gilt, dass  $g$  und  $g'$  einen Haplotypen  $h$  gemeinsam haben und  $g$  und  $g''$  einen Haplotypen  $h'$  und es gelte  $g = h \otimes h'$ . Dann hat  $h$  in jeder Position eine 1, in der  $g$  und  $g'$  eine 2 haben.
- Für eine  $(\star, 2)$ - beschränkte Cliquesinstanz gehört jedes nichttriviale Genom zu höchstens einem Cliques-Clarkpfad.
- Die bestmögliche Lösung für eine  $(\star, 2)$ -beschränkte Cliquesinstanz ohne Cliques-Clark-Pfade hat eine Kardinalität  $n + 1$ .

□

### 4.1.3 Spezialfall Graphen mit beschränkter Baumweite

**Definition 4.4.** Ein Graph  $G$  hat die Baumweite  $k$ , wenn  $G$  eine Knotenüberdeckung  $\{X_i\}_{i \in I}$  ermöglicht, für die gilt:

1.  $|X_i| \leq k + 1$  für alle  $i$ .
2. Von jeder Kante  $gg'$  gibt es ein  $X_i$ , das sowohl  $g$  als auch  $g'$  enthält.
3. Die Mengen  $X_i$  können Knoten  $i$  eines binären Wurzelbaumes  $T$  zugeordnet werden, so dass wenn  $j$  auf einem Pfad zwischen  $i$  und  $k$  in  $T$ , dann ist  $X_i \cap X_k \subseteq X_j$ .

**Definition 4.5.** Eine Instanz heißt abzählbar, wenn die Anzahl der Haplotypen, die zu den gegebenen Genomen konsistent sind, polynomiell ist.

**Bemerkung.** Definition 4.5 ist äquivalent zu der Aussage: Eine Instanz heißt abzählbar, wenn sie  $(O(\log n), \star)$ -beschränkt ist.

**Satz 4.3.** *Es gibt einen polynomiellen Algorithmus für abzählbare Instanzen, wenn der Clark-Konsistenzgraph ein Graph mit beschränkter Baumweite ist.*

*Beweis:*

Der in [Sha05] beschriebene Algorithmus arbeitet auf dem in Definition 4.4 beschriebenen Wurzelbaum  $T$  der Überdeckungsmengen  $X_i$  mit dynamischer Programmierung und konstruiert eine optimale Lösung auf einem Teilbaum, beginnend bei der Wurzel.

Sei  $r$  die Wurzel von  $T$ ,  $v_1$  und  $v_2$  die beiden Kinder eines Knotens  $v$  und  $X_v$  die Menge an Genomen, die laut Konstruktion zu  $v$  gehört. Eine Menge  $\mathfrak{H}$  von Haplotypen löst einen Knoten  $v$ , wenn  $\mathfrak{H} = \{h_1, \dots, h_{|X_v|}\}$  und ein Genom  $i$  in  $X_v$  ist konsistent zu  $h_i$ .

Sei  $D(v)$  die optimale Lösung für eine Unterinstanz, also einen Unterbaum mit der Wurzel  $v$ .  $D(v, H)$  bezeichnet die optimale Lösung in dieser Unterinstanz für eine Multimenge  $\mathfrak{H}$ , die  $r$  löst. Dann ist  $D(r) = \min_{\mathfrak{H}} D(r, \mathfrak{H})$ . Dann gilt die Rekursionsformel

$$D(r, \mathfrak{H}) = \min_{\mathfrak{H}_1, \mathfrak{H}_2} \{D(r_1, \mathfrak{H}_1) + D(r_2, \mathfrak{H}_2) + \Delta(r, r_1, r_2, \mathfrak{H}, \mathfrak{H}_1, \mathfrak{H}_2)\}.$$

Dabei lösen die  $\mathfrak{H}_i$ ,  $i = 1, 2$   $r$  und stimmen auf den Haplotypen, die die gemeinsamen Genome von  $X_r$  und  $X_{r_i}$  erzeugen, überein.  $\Delta(r, r_1, r_2, \mathfrak{H}, \mathfrak{H}_1, \mathfrak{H}_2)$  ist ein korrigierender Term für den Fall, dass der Durchschnitt  $X$  von  $X_{r_1}$  und  $X_{r_2}$  nicht leer ist. Sei  $x$  die Anzahl von Haplotypen, die benötigt wird um die Genome von  $X$  zu erzeugen und  $y$  die Anzahl der Haplotypen, die benötigt werden um  $X_r - (X_{r_1} \cup X_{r_2})$  zu erzeugen. Dann ist  $\Delta(r, r_1, r_2, \mathfrak{H}, \mathfrak{H}_1, \mathfrak{H}_2) = y - x$ .

Am Ende der Rekursion, für ein Blatt  $v$ , gilt dann  $D(v, \mathfrak{H})$  ist die Anzahl an unterschiedlichen Haplotypen in der Menge, die aus den Haplotypen in  $\mathfrak{H}$  und den jeweiligen Gegenstücken in Bezug auf  $X_v$  besteht.

$D(r)$  kann mit einem Durchlauf durch den Baum  $T$  in polynomieller Zeit bestimmt werden. □

#### 4.1.4 Spezialfall Bipartite Graphen

In diesem Abschnitt sollen Instanzen des PHP untersucht werden, deren Clark-Konsistenzgraph bipartit ist.

**Bemerkung.** *Wenn der Clark-Konsistenzgraph einer PHP-Instanz bipartit ist, kann jeder Haplotyp nur gemeinsamer Haplotyp von höchstens zwei Genomen sein. Daher ist die Anzahl  $n$  von Genotypen eine untere Schranke für jede Lösung.*

**Satz 4.4.** *PHP ist NP-schwer, wenn der Clark-Konsistenzgraph der Eingabeinstanz bipartit ist und der längste Clark-Pfad die Länge 2 hat.*

*Beweis:*

Der Beweis verwendet eine Reduktion auf (3DM3), wie im Beweis von Satz 2.3 beschrieben.

*Folgerung:*

PHP ist APX-schwer, wenn der Clark-Konsistenzgraph der Eingabeinstanz bipartit ist und der längste Clark-Pfad die Länge 2 hat.

Da jeder Haplotyp nur von maximal zwei Genomen ein gemeinsamer Haplotyp sein kann, ist jede Lösung automatisch eine 2-Approximation der optimalen Lösung. Im Folgenden soll diese einfache Aussage verbessert werden, indem das Problem auf das Matching-Problem reduziert wird.

*Lemma:*

Wenn der längste Clark-Pfad die Länge 1 hat, dann ist PHP in Polynomzeit lösbar.

Nun ist es möglich, einen 1,5-Approximationsalgorithmus anzugeben, der aus den folgenden drei Schritten besteht:

1. Bestimme ein maximales Matching im Clark-Konsistenzgraphen.
2. Erzeuge jedes Paar von Genomen in dem Matching durch einen gemeinsamen Haplotyp.

3. Erzeuge die verbleibenden Genome durch beliebige konsistente Haplotypen.

*Lemma:*

Der obige Algorithmus hat eine Approximationsgüte von 1,5 für Instanzen, deren Clark-Konsistenzgraphen bipartit ist.

*Beweis:*

Betrachtet wird eine Instanz des PHP mit bipartitem Clark-Konsistenzgraphen. Sei  $m$  die Größe eines maximalen Matchings in  $G$  und  $n$  die Zahl der Genotypen. Die Lösung des Algorithmus hat eine Größe von  $2n - m$ . Sei  $\mathfrak{H}$  eine optimale Lösung des PHP auf der gegebenen Instanz und  $e$  die Anzahl der Genompaare, die einen gemeinsamen Haplotypen in dieser Lösung haben. Dann ist  $|\mathfrak{H}| = 2n - e$  und die Gütegarantie beträgt

$$\frac{2n - m}{2n - e} \leq \frac{2n - m}{2n - 2m} \leq \frac{3}{2}.$$

Die erste Ungleichung gilt, weil der Grad jedes Knotens  $\leq 2$  ist. Die zweite Ungleichung gilt, da  $2m \leq n$  und da für  $m = \frac{n}{2}$  die schlechteste Schranke angenommen wird. □

## 4.2 Pedigree-Graphen

Anders als in den bisher betrachteten Ansätzen werden bei Pedigree-Graphen die Individuen der Population nicht unabhängig voneinander betrachtet, sondern in Form eines Stammbaums. Dabei gelten zwei verschiedene Voraussetzungen, die hilfreich erscheinen.

1. Der Pedigree-Graph gibt die Struktur des Stammbaumes wieder und
2. die Mendelschen Gesetze werden als gültig vorausgesetzt, was bedeutet, dass im Stammbaum keine Mutationen auftreten, sondern nur die Vererbung und Neukombination der Gene betrachtet wird. Die Mendelschen Gesetze helfen daher bei der Bestimmung einiger Haplotypen, können jedoch nach [Bon03] das Problem noch nicht wesentlich vereinfachen.

**Definition 4.6.** Ein Pedigree-Graph ist ein gerichteter, schwach zusammenhängender Graph  $G = (V, E)$ , wobei  $V = M \cup F \cup N$ . Dabei bezeichnet  $M$  die Knoten, die die männlichen Mitglieder des Stammbaumes darstellen,  $F$  die weiblichen und  $N$  die Verbindungsknoten.  $M$  und  $F$  werden Individuen genannt. Kanten existieren nur zwischen Individuen-Knoten und Verbindungsknoten. Der Eingangsgrad eines Individuen-Knotens darf höchstens 1 sein. Der Eingangsgrad eines Verbindungsknotens muss genau 2 betragen; davon beginnt eine Kante bei einem männlichen und eine bei einem weiblichen Knoten. Der Ausgangsgrad muss echt positiv sein.

Es existiert noch eine andere äquivalente Definition.

**Definition 4.7.** Ein Pedigree-Graph  $G$  ist ein schwach zusammenhängender gerichteter Graph, in dem jeder Knoten den Eingangsgrad 0 oder 2 hat.

### Beispiel 1

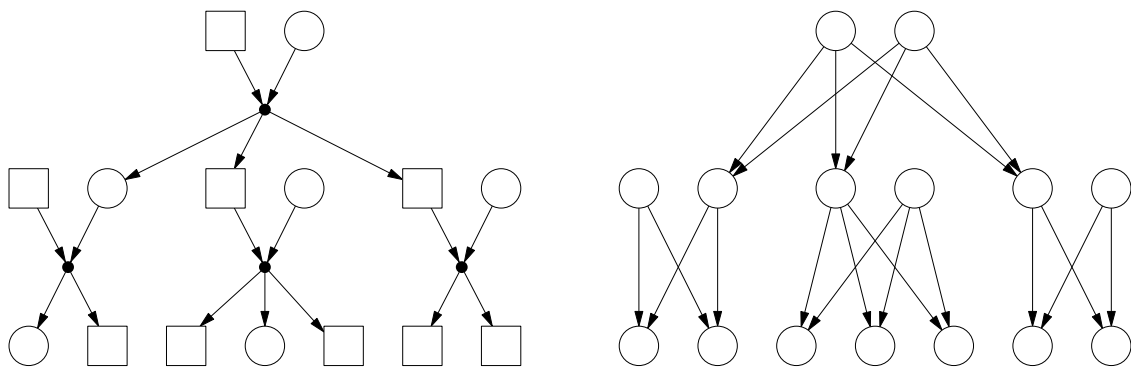


Abbildung 4.2: Ein Beispiel: links nach Definition 4.6 und rechts nach Definition 4.7.

Den in Abbildung 4.2 gezeigten Figuren liegen dieselben Daten zu Grunde. In der linken Abbildung sind durch die eckigen bzw. runden großen Knoten die männlichen bzw. weiblichen Individuen-Knoten dargestellt. Die kleineren schwarzen Knoten sind Verbindungsknoten nach Definition 4.6.

## Beispiel 2

Die Abbildung 4.3 zeigt ein Beispiel eines Pedigree-Graphen nach Definition 4.7 mit einem (ungerichteten) Kreis.

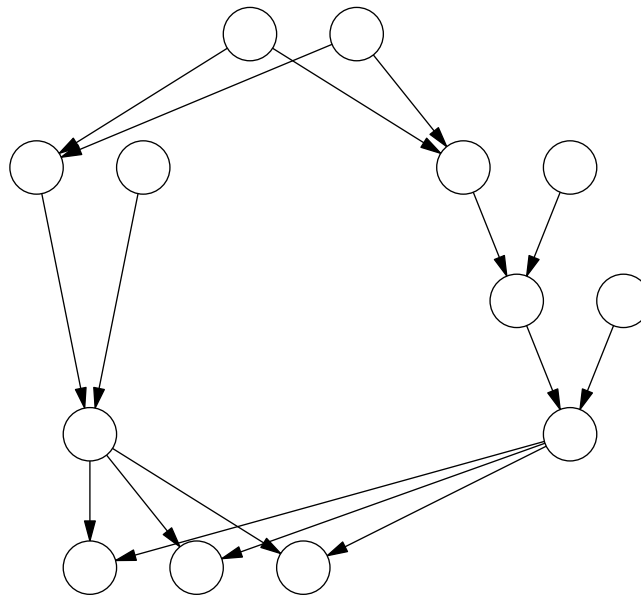


Abbildung 4.3: Ein zweites Beispiel mit einem Kreis

In [Bon03] werden verschiedene zum PHP ähnliche Probleme betrachtet, doch es werden an dieser Stelle keine für diese Arbeit relevanten Resultate erzielt.

## 4.3 Fragment-Konflikt-Graphen

Der folgende Absatz ist den Fragment-Konflikt-Graphen, einem Ansatz für das Individual Haplotyping Problem (IHP), gewidmet. Dieses Problem unterscheidet sich in der Herangehensweise sehr stark vom PHP und ist hier nur wegen der interessanten graphentheoretischen Ansätze mit aufgeführt. Im Folgenden sind die Ergebnisse aus [Bon03], [Lip02] und [Lan01] zusammengefasst. Die Idee basiert auf der Feststellung, dass in der Realität immer nur kleine, sich überlappende Fragmente eines DNA-Stranges bekannt sind, die Fehler enthalten können und von denen man nicht weiß, von welchem der beiden Chromosomenstränge sie

stammen. Es ist von manchen Paaren dieser Fragmente bekannt, dass sie von demselben Chromosomenpaar stammen müssen und welchen Abstand sie haben. Das Problem besteht dann darin, unter Verwendung dieser Fragmente die zwei Haplotypen zu rekonstruieren. Schwierig oder sogar unlösbar ist dies, wenn es Lücken in den Daten gibt, es also SNPs gibt, die auf keinem der Fragmente auftreten.

### Ein Beispiel mit vier Fragmenten

Im folgenden Beispiel sind vier Fragmente (dargestellt in Tabelle 4.2) eines DNA-Stranges gegeben. In Tabelle 4.3 sind die SNPs rot markiert und in Tabelle 4.4 ist dann die Aufteilung der Fragmente auf die beiden Haplotyp- bzw. Chromosomenstränge sichtbar.

Tabelle 4.2: Die vier Fragmente

```

A T C G A T G C A T G
  T C G A T G G A T G A A T
    A T G C A T G C A T G C A A
      A T G A A T G C A T G C A

```

Tabelle 4.3: Die SNPs (rot) auf den Fragmenten

```

A T C G A T G C A T G
  T C G A T G G A T G A A T
    A T G C A T G C A T G C A A
      A T G A A T G C A T G C A

```

Tabelle 4.4: Aufteilung der Fragmente

```

A T C G A T G C A T G
    A T G C A T G C A A
      T C G A T G G A T G A A T
        A T G A A T G C A T G C A

```

Es werden also Tupel  $(S, F, r)$  betrachtet. Dabei stellt  $S$  eine Menge von SNPs dar und  $F$  eine Menge von Fragmenten.  $r$  ist eine Relation  $r : S \times F \rightarrow \{O, A, B\}$ , die

(durch O) angibt ob, ein SNP auf einem bestimmten Fragment liegt, und wenn dies der Fall ist, zu welchem Haplotypen bzw. Chromosom (A oder B) das Fragment gehört. Wenn man eine Ordnung der Menge  $S$  und der Menge  $F$  vorgibt, dann kann die Eingabe auch als Matrix verstanden werden.

Eine wichtige Eigenschaft von Instanzen dieser Art ist die *consecutive 1s property* (C1P), das bedeutet, es existiert eine Reihenfolge der SNPs, so dass in jeder Spalte der Matrix alle nicht-0-Werte direkt aufeinander folgen.

Das Ziel des IHP ist nun, eine Partition von  $F$  in zwei Blocks  $H_1$  und  $H_2$  zu finden, so dass folgende Eigenschaften gelten.

$$\forall s \in S \forall f, f' \in H_i : (r(s, f) = r(s, f')) \vee (r(s, f) = O) \vee (r(s, f') = O), i = 1, 2$$

Wenn diese existiert, ist die IHP für diese Instanz lösbar.

Es wird ein Konflikt-Graph  $G_F$  konstruiert mit Knoten  $V = F$  und Kanten  $E = \{(f, f') | \exists s \in S, (r(s, f) \neq O) \wedge (r(s, f') \neq O) \wedge (r(s, f) \neq r(s, f'))\}$ .

**Satz 4.5.** *Eine Instanz ist genau dann lösbar, wenn  $G_F$  bipartit ist.*

### Ein Beispiel mit neun Fragmenten

Die folgende Tabelle 4.5 zeigt die neun Fragmente. Aus diesen Angaben und der

Tabelle 4.5: Instanz mit neun Fragmenten

|    |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|
| 1. | A | A |   |   |   |   |   |   |
| 2. | B | A |   |   |   |   |   |   |
| 3. |   | B | B |   |   |   | B |   |
| 4. |   |   | A | A |   |   |   |   |
| 5. |   |   |   | B | A |   |   |   |
| 6. |   |   |   |   | B | B |   |   |
| 7. |   | A |   |   |   | A | A |   |
| 8. |   |   |   |   |   |   | B | A |
| 9. |   |   |   |   |   |   | B | B |

obigen Konstruktionsvorschrift, entsteht der Graph in Abbildung 4.4.

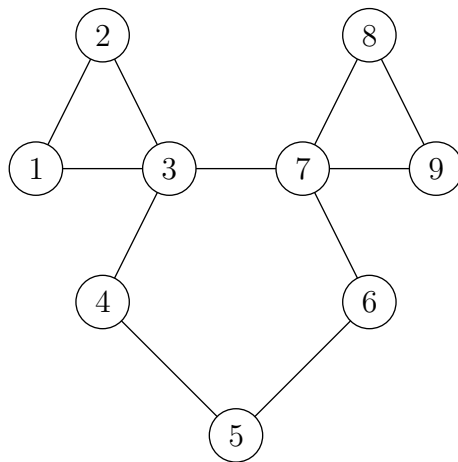


Abbildung 4.4: Der Fragment-Konflikt-Graph

Das Problem IHP ist im allgemeinen Fall NP-schwer, doch unter bestimmten Voraussetzungen, z.B. für Instanzen ohne Lücken, wird es polynomiell lösbar.

# 5 Minimal Rainbow-Subgraph Problem MRS

Kapitel 2 betrachtete ein mathematisches Modell für das PHP und Kapitel 4 einige (graphentheoretische) Lösungsansätze. In diesem Kapitel sollen

1. eine weitere Möglichkeit vorgestellt werden, das PHP mit Hilfe eines graphentheoretischen Modells zu beschreiben und
2. einige einfache Zugänge zu dem resultierenden Problem dargestellt werden.

## 5.1 Das Modell

Auf Grund der diploiden Struktur des menschlichen Genoms können die Daten des PHP als kantengefärbter Graph kodiert werden, wobei die Haplotypen als Knoten und die Genome als Kanten eines Graphen verstanden werden. Dabei werden zwei Knoten  $v_1$  und  $v_2$  genau dann durch eine Kante  $e$  verbunden, wenn gilt  $e = v_1 \otimes v_1$ . Gleiche Genome bekommen dabei die gleiche Farbe. Ein Genom ohne mehrdeutige Stellen wird durch eine Schlinge dargestellt, da die beiden erzeugenden Haplotypen identisch sind. Die Suche nach einer minimalen Menge von Haplotypen, die alle Genome erklären, entspricht dann der Suche nach einer minimalen Knotenmenge, so dass der induzierte Teilgraph Kanten aller Farben enthält und das Problem wie folgt umgeschrieben werden kann:

**Das MRS**Gegeben

Ein  $p$ -kantengefärbter Graph  $G = (V, E)$ , ohne Mehrfachkanten.

Gesucht

Eine Menge  $N \subseteq V$  minimaler Kardinalität mit  $|N| = n^*$ , so dass der von  $N$  induzierte Teilgraph eine verallgemeinerte Regenbogenfärbung enthält.

In den folgenden Abschnitten 5.1 bis 5.4 werden nur Graphen ohne Schlingen betrachtet.

**Satz 5.1.** *Der Graph  $G = (V, E)$ , der durch obige Konstruktion entsteht, ist immer zulässig gefärbt.*

*Beweis:*

Angenommen, es existiert ein Knoten (Haplotyp) an dem zwei Kanten der gleichen Farbe inzidieren. Das würde bedeuten, dass es zwei Möglichkeiten gibt, zu einem Haplotypen  $h_1$  und einem festen Genom  $g$  einen zweiten Haplotypen  $h_2$  zu finden, mit  $g = h_1 \otimes h_2$ . Doch das ist nach den Überlegungen aus Kapitel 2 unmöglich.  $\square$

**5.2 MRS auf speziellen Graphenklassen**

**Bemerkung.** *Es ist offensichtlich, dass für alle Graphen  $G$  gilt:  $n^*(G) \leq n$ .*

*Das heißt, dass für alle oberen Schranken  $n^*(G) \leq b$  gilt:*

$$n^*(G) \leq \min\{n, b\}.$$

**Satz 5.2.** *Sei  $G = (V, E, p)$  ein Graph, der einen  $p$ -gefärbten vollständig zusammenhängenden Untergraphen mit  $p$  Kanten enthält. Dann gilt*

$$n^*(G) = \frac{1}{2} + \frac{1}{2}\sqrt{1 + 8p}.$$

*Beweis:*

Sei  $G' \subseteq G$  ein Untergraph mit den oben beschriebenen Eigenschaften und  $n'$  Knoten und  $m'$  Kanten. Dann gilt  $m' = \frac{n'(n'-1)}{2}$ . Daraus folgt  $n^*(G) \leq \frac{1}{2} + \frac{1}{2}\sqrt{1+8p}$ . Die Gleichheit folgt aus der Schranke aus Absatz 5.6.  $\square$

### 5.2.1 Beschränkung der auftretenden Farben

In diesem Abschnitt betrachten wir nur einfache Graphen und beschränken die möglichen Graphen durch die Anzahl der auftretenden Farben. Sei  $p$  die Anzahl der auftretenden Farben.

**Satz 5.3.** *Ein Graph  $G = (V, E, p)$  hat einen Maximalgrad  $\Delta(G) \leq p$ .*

*Beweis:*

Angenommen, der Graph  $G = (V, E, p)$  hätte einen Maximalgrad  $\Delta(G) > p$ . Dann würden an dieser Stelle  $\Delta(G)$  Kanten aneinanderstoßen, die wegen 5.1 alle verschieden sein müssen. Das führt jedoch zu einem Widerspruch zu der Voraussetzung, dass  $p$  die Anzahl der auftretenden Farben ist.  $\square$

#### Fall $p=1$

Ein Graph  $G = (V, E, 1)$  mit einer zulässigen 1-Färbung, besteht nach Satz 5.3 nur aus isolierten Kanten. Um einen minimalen Untergraph mit verallgemeinerter Regenbogenfärbung  $G_{MRS} \in R^*$  zu finden, genügt es, eine beliebige Kante auszuwählen und beide Endknoten in die Lösung aufzunehmen.

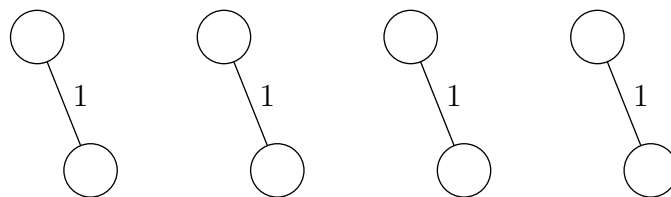


Abbildung 5.1: Zulässig 1-gefärbter Graph

**Fall  $p=2$** 

**Satz 5.4.** Für einen Graphen  $G = (V, E, 2)$  gilt:  $n^* = 5 - \Delta(G)$ .

*Beweis:*

Wenn der Graph mit nur zwei Farben gefärbt ist, besteht er aus isolierten Kanten, Wegen und geraden Kreisen, weil er einen Maximalgrad  $\leq 2$  hat und eine zulässige 2-Färbung besitzt. Wenn der Maximalgrad eins ist, dann besteht der Graph aus isolierten Kanten und für ein MRS werden in jedem Fall die vier Knoten von zwei Kanten verschiedener Farbe benötigt. Wenn der Maximalgrad zwei ist, dann enthält  $G$  einen induzierten  $P_3$ , der ein MRS darstellt. In diesem Fall gilt:  $n^* = 3$ .  $\square$

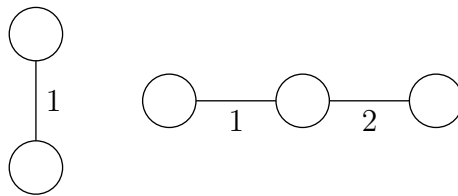


Abbildung 5.2: zulässig 2-gefärbter Graph

**Fall  $p=3$** 

**Satz 5.5.** Für einen zulässigen 3-gefärbten Graph  $G = (V, E, 3)$  ohne  $C_3$  mit Maximalgrad  $\Delta(G)$  und  $|E| \geq 1$  gilt:

$$n^*(G^*) \leq 7 - \Delta(G). \quad (5.1)$$

Wenn  $G$  einen  $C_3$  enthält, gilt:  $n^* = 3$ .

*Beweis:*

Der Beweis kann aufgeteilt werden in drei Fälle:

- $\Delta(G) = 1$ : Der Graph enthält nur isolierte Kanten, von denen drei mit jeweils verschiedenen Farben für ein MRS genügen. Damit gilt:  $n^* = 6$ .

- $\Delta(G) = 2$ : Der Graph enthält nur Wege und Kreise. Wegen  $\Delta(G) = 2$  gibt es mindestens einen Knoten mit Grad zwei, mit zwei inzidenten Kanten verschiedener Farben. Damit können 2 Farben immer mit drei Knoten abgedeckt werden. Um die dritte Farbe abzudecken, werden maximal zwei zusätzliche Knoten benötigt.

Enthält der Graph einen induzierten ungeraden Kreis der Länge  $> 3$ , dann enthält er auch einen  $P_4 \in R_3^*$  und  $n^* = 4$ .

Wenn der Graph einen  $C_3$  enthält, dann ist dieser automatisch der gesuchte Untergraph und  $n^* = 3$ .

- $\Delta(G) = 3$ : In diesem Fall gibt es einen Knoten  $v$  mit Grad drei, an den drei Kanten mit verschiedenen Farben inzidieren.  $v$  und seine drei Nachbarn bilden dann die gesuchte Knotenteilmenge und  $n^* = 4$ .

□

### allgemeiner Fall

**Satz 5.6.** *Gegeben sei ein Graph  $G = (V, E, p)$  mit Maximalgrad  $\Delta(G)$ . Dann gilt:*

$$n^*(G) \leq 2p + 1 - \Delta(G). \quad (5.2)$$

*Beweis:*

Ein Graph mit Maximalgrad  $\Delta(G)$  hat mindestens einen Knoten  $v$  vom Grad  $\Delta(G)$ . Mit  $v$  und seinen  $\Delta(G)$  Nachbarn können  $\Delta(G)$  Farben mit  $\Delta(G) + 1$  Knoten abgedeckt werden. Die übrigen  $p - \Delta(G)$  Farben benötigen im ungünstigsten Fall  $2(p - \Delta(G))$  Knoten.

$$\begin{aligned} n^*(G) &\leq \Delta(G) + 1 + 2(p - \Delta(G)) \\ &= 2p + 1 - \Delta(G). \end{aligned}$$

□

### 5.2.2 Sterne

Es ist als Folgerung aus Satz 5.6 möglich  $n^*$  für Graphen, die große Sterne enthalten, genauer zu charakterisieren.

**Satz 5.7.** *Für einen Graphen  $G = (V, E, p)$ , der einen  $p$ -Stern enthält, gilt:  $n^*(G) \leq p + 1$ .*

*Beweis:*

Für einen Graphen mit  $p$ -Stern gilt  $\Delta(G) \geq p$  und wegen Satz 5.3  $\Delta(G) \leq p$ , also:  $\Delta(G) = p$ . Mit Hilfe der Schranke aus Satz 5.6 gilt:

$$n^*(G) \leq 2p + 1 - \Delta(G) \stackrel{\Delta(G)=p}{=} p + 1.$$

□

### 5.2.3 Reguläre Graphen

**Definition 5.1.** *Ein Graph  $G=(V, E)$  heißt  $r$ -regulär, wenn alle Knoten  $v \in V$  den Grad  $r$  haben.*

Sei  $G$  ein zulässig  $p$ -gefärbter regulärer Graph. Je nach der Größe von  $r$  kann man nun verschiedene Fälle unterscheiden.

#### Fall $r=p$

In diesem Fall liegen an jedem Knoten alle  $p$  Farben an. Damit gibt es einen (nicht induzierten) Untergraphen, der einen Stern darstellt und damit gilt  $n^* \leq p + 1$ .

#### Fall $r=p-1$

**Satz 5.8.** *Sei  $G = (V, E, p)$  ein  $r$ -regulärer Graph mit  $r = p - 1$ . Dann gilt  $n^* \leq p + 2$ .*

*Beweis:*

Um  $r$  Farben zu überdecken, kann ein beliebiger Knoten  $v$  inklusive seiner Nachbarn ausgewählt werden. Es werden also  $r + 1 = p$  Knoten benötigt. Für die letzte Farbe werden dann im schlechtesten Fall noch zwei neue Knoten benötigt. Also gilt:  $n^*(G) \leq r + 1 + 2 = p + 2$ .  $\square$

### Fall $r < p - 1$

Um in diesem Fall eine bessere Aussage als Satz 5.6 zu erhalten, muss noch eine Zusatzannahme getroffen werden.

**Satz 5.9.** *Sei  $G = (V, E, p)$  ein  $r$ -regulärer Graph mit  $r < p - 1$ . Dann gilt  $n^* \leq 2p - r - s + 2$ . Dabei ist  $s = \lceil \frac{p-r}{n-1} \rceil$ .*

*Beweis:*

Sei  $G = (V, E, p)$  ein  $r$ -regulärer Graph mit  $r < p - 1$ . Jeder Knoten hat den Grad  $r$ . Daher können mit einem beliebigen Knoten und seinen Nachbarn  $r$  Farben mit  $r + 1$  Knoten abgedeckt werden. Durch die Konstruktion von  $s$  und dem Schubfachprinzip ist garantiert, dass es dann noch mindestens einen Knoten  $v$  gibt, an dem  $s$  neue Farben anliegen. Mit  $v$  und den  $s$  Nachbarn, an deren Kanten eine neue Farbe anliegt, können weitere  $s$  Farben mit  $s + 1$  Knoten abgedeckt werden. Für die restlichen  $p - r - s$  Farben werden jeweils maximal zwei neue Knoten benötigt. Die Zusammenfassung dieser Überlegungen führt zu folgendem Zusammenhang.

$$\begin{aligned} n^* &\leq r + 1 + s + 1 + 2(p - r - s) \\ &= 2p - s - r + 2. \end{aligned}$$

$\square$

**Bemerkung.** *Falls  $s \geq 2$ , dann ist diese Schranke besser als die in Satz 5.6.*

### 5.2.4 Wege

Sei  $G = (V, E, p)$  ein Weg. Unter bestimmten Voraussetzungen kann die Aussage aus Satz 5.6 noch verbessert werden.

**Satz 5.10.** *Sei  $G = (V, E, p)$  ein (zusammenhängender) Weg,  $s = m - p$ ,  $s \geq 2$  und  $k$  die Anzahl der Farben, die in  $G$  mehrfach vorkommen. Dann gilt  $n^* \leq p + s + t$ .*

Dabei ist

$$t = \begin{cases} 0 & , \text{ falls } s + k > p - k \\ 1 & , \text{ sonst.} \end{cases}$$

**Bemerkung.** *Diese Abschätzung ist besser als die Schranke aus Satz 5.6, wenn im Verhältnis zur Länge des Weges viele Farben auftreten.*

*Beweis:*

Wenn in  $G$  viele Farben vorkommen, dann werden alle  $p + s + 1 = n$  Knoten zur Überdeckung der  $p$  Farben benötigt. Unter bestimmten Bedingungen ist es jedoch möglich, einige dieser Knoten wegzulassen. Wenn sich zwei Kanten treffen, deren Farbe mehrfach vorkommt, kann der gemeinsame Knoten der beiden Kanten weggelassen werden. Die Bedingung  $s + k > p - k + 1$  garantiert, dass mindestens zwei der nichtbenötigten Kanten adjazent sind. Wenn  $s + k \leq p - k$  gilt, ist es möglich, immer abwechselnd eine einfach vorkommende Farbe und eine mehrfach vorkommende Farbe zu vergeben, so dass keine nichtbenötigten Kanten inzidieren. Im Fall, dass  $s + k = p - k + 1$ , ist es auch möglich, dass keine zwei mehrfach vorkommenden Farben adjazent sind, doch in diesem Fall sind die beiden Kanten am Rand des Weges mit mehrfach vorkommenden Farben gefärbt und dann kann wenigstens einer der beiden Randknoten weggelassen werden.  $\square$

**Bemerkung.** *Falls  $s = 1$ , dann kommt genau eine Farbe doppelt vor, dann gilt:*

$$n^* = \begin{cases} p + 1 & , \text{ falls die doppelte Farbe einmal am Rand des Weges vorkommt} \\ p + 2 & , \text{ falls die doppelte Farbe beide Male in der Mitte des Weges auftritt.} \end{cases}$$

## 5.3 Ein Greedy-Algorithmus

### 5.3.1 Der Algorithmus

Der folgende intuitive Greedy-Algorithmus liefert eine approximative Lösung des Problems MRS ohne Schlingen:

#### **GREEDY(G, C)**

Eingabe:

Ein einfacher Graph  $G = (V, E, p)$  und eine Menge von Farben  $C = \{c_1, c_2, \dots, c_p\}$ .

Ausgabe:

Eine Teilmenge  $N$  von  $V$ , deren induzierter Teilgraph von  $G$  eine verallgemeinerte  $p$ -Regenbogenfärbung enthält.

Schritt 0:

Setze  $V_p = \emptyset$

$$C^* = C$$

$$G^* = (V^*, E^*) \cong G$$

Schritt 1:

Wähle einen Knoten  $v$  aus  $V^*$  mit maximaler Anzahl anliegender Farben aus  $C^*$ .

Schritt 2:

Füge  $v$  und alle seine Nachbarn zu  $V_p$  hinzu und bilde einen neuen Graphen  $G^* = (V^* - \{v\}, E^* - \{e \in E^* : e = (v, x), x \in N(v)\})$ . Setze  $C^* = C^* - \{colour(e) : e = vx, x \in N(v)\}$

Schritt 3:

*falls*  $C^* = \emptyset \rightarrow$  Setze  $N = V^*$  und *Stop*

*sonst* Rückkehr zu Schritt 1

**Bemerkung.** *Es ist auch eine etwas andere Vorgehensweise zur Bildung von  $G^*$  möglich, die aber zum gleichen Ergebnis führt: Dafür werden nicht nur die Kanten zwischen dem Knoten  $v$  und seinen Nachbarn gelöscht, sondern alle Kanten in den*

*schon abgedeckten Farben. Der neue Knoten  $v$  im nächsten Durchlauf ist dann ein Knoten mit Maximalgrad in  $G^*$ .*

Unter bestimmten Umständen, z.B. bei Graphen mit geringen Zusammenhangsgrad, kann es sinnvoll sein den obigen Algorithmus ein wenig zu modifizieren und den Algorithmus GREEDYN zu verwenden.

### GREEDYN( $G, C$ )

#### Eingabe:

Ein einfacher Graph  $G = (V, E, p)$  und eine Menge von Farben  $C = \{c_1, c_2, \dots, c_p\}$ .

#### Ausgabe:

Eine Teilmenge  $N$  von  $V$ , deren induzierter Teilgraph von  $G$  eine verallgemeinerte  $p$ -Regenbogenfärbung enthält.

Schritt 0:

Setze  $V_p = \emptyset$

$C^* = C$

$G^* = (V^*, E^*) \cong G$

Schritt 1:

Wähle einen Knoten  $v$  aus  $V^*$  mit maximaler Anzahl anliegender Farben aus  $C^*$ .

Schritt 2:

Füge  $v$  und alle seine Nachbarn zu  $V_p$  hinzu und bilde einen neuen Graphen  $G^* = (V^* - \{v\}, E^* - \{e \in E^* : e = (v, x), x \in N(v)\})$ , Setze  $C^* = C^* - \{colour(e) : e = vx, x \in N(v)\}$

Schritt 3:

*falls*  $C^* = \emptyset \rightarrow$  Setze  $N = V^*$  und *Stop*

*sonst* Suche in der Nachbarschaft  $N(v)$  einen Knoten  $v$  mit maximaler Anzahl anliegender Farben aus  $C^*$ . Falls  $N(v) = \emptyset$ , suche in einer anderen Komponente nach dem Knoten  $v$  mit der maximalen Anzahl anliegender Farben.

### 5.3.2 Ein Beispiel

Das folgende Beispiel soll die Funktionsweise des Algorithmus GREEDY an einem Graphen  $G$  (Abbildung 5.3) mit zehn Knoten und einem Maximalgrad von  $\Delta(G) = 5$  und einer Farbmenge  $C = \{1, \dots, 7\}$  demonstrieren. Im ersten Durchlauf des Algo-

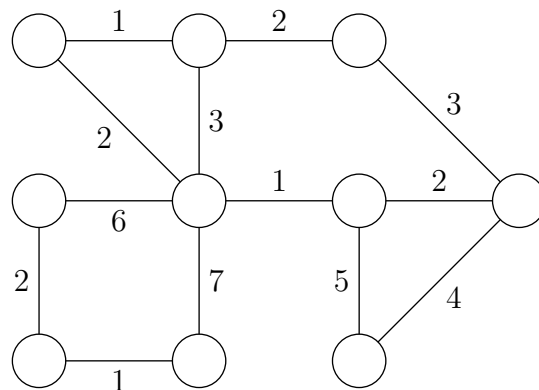


Abbildung 5.3: Der Beispielgraph  $G$

rithmus GREEDY wird einer der Knoten mit Maximalgrad ausgewählt und samt seiner Nachbarn zur Menge  $V_p$  hinzugefügt, die in Abbildung 5.4 rot dargestellt ist. Dieser Knoten und die grünen Kanten werden aus  $G$  entfernt, um den Graphen  $G^*$

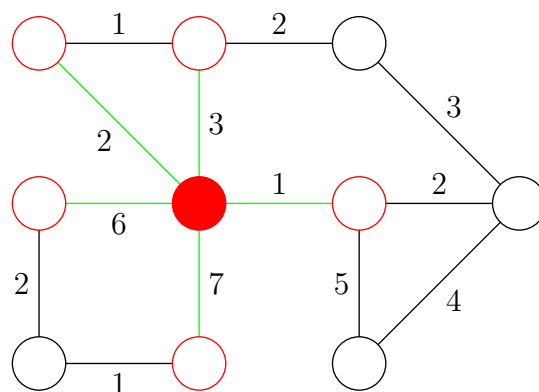


Abbildung 5.4: Der erste Durchlauf des Algorithmus

(Abbildung 5.5) mit  $C^* = \{4, 5\}$  zu bilden. Nach dem zweiten Algorithmusdurchlauf (Abbildung 5.6) ist  $C = \emptyset$  und der Algorithmus bricht mit Ausgabe der Lösung  $N$  (Abbildung 5.7) ab.

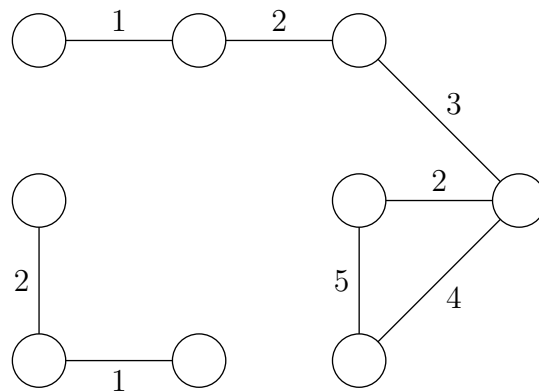
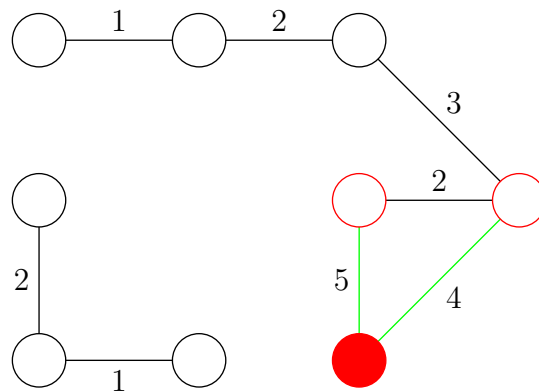
Abbildung 5.5: Der Graph  $G^*$  nach dem ersten Durchlauf

Abbildung 5.6: Der zweite Durchlauf des Algorithmus

### 5.3.3 Analyse des Greedy-Algorithmus

Der folgende Abschnitt beinhaltet die Analyse des Algorithmus GREEDY hinsichtlich der Laufzeit und der Approximationsgüte.

#### Laufzeitanalyse

In der folgenden Tabelle 5.1 sind die Laufzeit des Algorithmus nach den einzelnen Schritten und der Häufigkeit des Aufrufs aufgeschlüsselt.

**Satz 5.11.** *Der Algorithmus  $\text{GREEDY}(G, C)$  hat polynomielle Laufzeit in Knoten und Farben.*

*Beweis:*

Sei  $c$  eine Konstante,  $n$  die Knotenanzahl des Graphen  $G$  und  $p = |C|$  die Anzahl

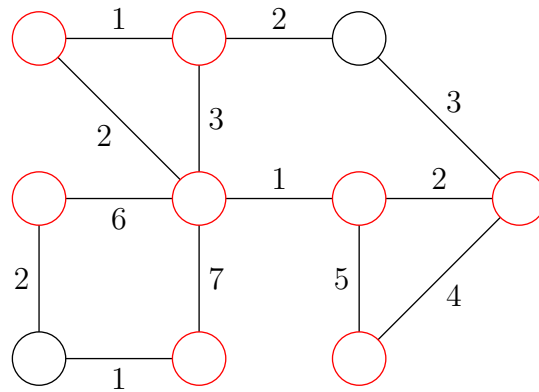
Abbildung 5.7: Der Graph  $G$  und die Menge  $V^*$  (rot)

Tabelle 5.1: Analyse der Laufzeit der einzelnen Algorithmenschritte

| Schritt | Anzahl Schritte | Anzahl Durchläufe |
|---------|-----------------|-------------------|
| 0       | $O(1)$          | 1                 |
| 1       | $O(n)$          | $p$               |
| 2       | $O(\Delta(G))$  | $p$               |
| 3       | $O(1)$          | $p$               |

der auftretenden Farben, dann kann mit Hilfe der Tabelle 5.1 folgende Abschätzung für die Laufzeit  $L$  vorgenommen werden:

$$\begin{aligned}
 L &\leq c_1 + p(n + \underbrace{\Delta(G)}_{\Delta(G) \leq p} + c_2) \\
 &\leq c_1 + pn + p^2 + c_2p \\
 &= O(pn + p^2).
 \end{aligned}$$

□

### Approximationsgüte

**Satz 5.12.** *Der Algorithmus  $\text{GREEDY}(G, C)$  hat eine Gütegarantie von  $r_A = \sqrt{2p}$ .*

*Beweis:*

Im ersten Durchlauf des Algorithmus werden  $\Delta(G)$  Farben aus der Farbmenge  $C$

entfernt und danach in jedem Schritt wenigstens eine weitere Farbe. Daher kann die Schranke aus Satz 5.6  $n^* \leq 2p + 1 - \Delta(G)$  garantiert werden. Dann kann die Approximationsgüte durch

$$\begin{aligned} r_A &= \frac{|V_p|}{n^*} \\ &\leq \frac{2p + 1 - \Delta(G)}{\sqrt{2p}} \end{aligned}$$

beschränkt werden. □

**Bemerkung.** Diese Gütegarantie ist für große  $\Delta(G)$  besser als die, welche aus den Schranken in Abschnitt 2.3.2 berechnet werden kann.

**Bemerkung.** Selbst für einfache Graphenklassen, wie Wege und Sterne können die Algorithmen GREEDY und GREEDYN keine optimalen Lösungen garantieren. Die Abbildung 5.8 zeigt einen Graphen, der nur aus Sternkomponenten besteht und für den der Algorithmus GREEDY eine nichtoptimale Lösung findet. Die optimale Lösung besteht aus acht Knoten (den beiden kleineren Sternen). Der Algorithmus benutzt neun Knoten (fünf aus dem großen Stern und jeweils noch zwei aus den kleineren).

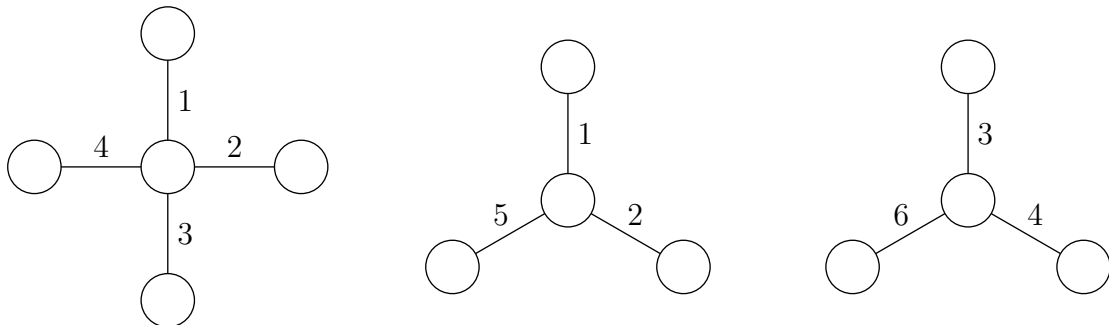


Abbildung 5.8: Beispiel

## 5.4 Ein weiterer approximativer Algorithmus

Es besteht im Falle von maximal  $t$  mehrdeutigen Stellen die Möglichkeit den folgenden intuitiven Algorithmus zu verwenden:

### 5.4.1 Der Algorithmus

#### Partition( $G, C$ )

##### Eingabe

Ein Graph  $G=(V, E, p)$  mit einer zulässigen Kantenfärbung und die Farbmengemenge  $C = \{c_1, c_2, \dots, c_p\}$ . Jede Farbe kommt maximal  $q = 2^{t-1}$  mal in  $G$  vor.

##### Ausgabe:

Eine Teilmenge  $N$  von  $V$ , deren induzierter Teilgraph von  $G$  eine verallgemeinerte  $p$ -Regenbogenfärbung enthält.

Jede Farbe taucht in dem Graphen höchstens  $q$ -mal auf.

Die Kantenmenge wird in  $p$  Farbklassen aufgeteilt. Aus jeder dieser Farbklassen wird nun nach einer zuvor zu bestimmenden Regel genau ein Element entnommen und die inzidenten Knoten werden der Menge  $V^*$  hinzugefügt. Die Auswahl der Kanten kann z.B. nach einem der folgenden Kriterien geschehen:

1. maximaler Grad an einem der beiden inzidenten Knoten oder
2. maximale Gradsumme (der beiden inzidenten Knoten) oder
3. zufällige Auswahl.

## 5.4.2 Analyse

### Laufzeit

Der Algorithmus hat eine Laufzeit von  $O(pq)$ , da in jeder der  $p$  Farbklassen eines der  $q$  Elemente ausgewählt werden muss. Wenn die Kanten zufällig ausgewählt werden, ist die Laufzeit sogar nur  $O(p)$ .

### Approximationsgüte

Die Güte des Algorithmus hängt vom gewählten Auswahlkriterium ab, allerdings kann im allgemeinen Fall nur eine Güte von  $\sqrt{2p}$  garantiert werden. Diese folgt schon aus den Schranken aus Absatz 2.3.2. Das Beispiel in Abbildung 5.9 zeigt, dass der Algorithmus unabhängig vom Auswahlkriterium im worst-case Kanten mit  $2p$  verschiedenen Knoten auswählt.

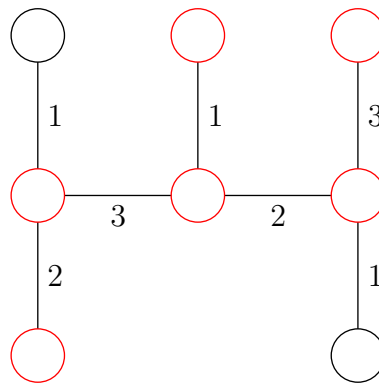


Abbildung 5.9: Ein worst-case-Beispiel

## 5.4.3 Verbesserung der Lösung

Um die Lösung des obigen Algorithmus zu verbessern, kann ein Austauschalgorithmus angewendet werden, der nach bestimmten Kriterien eine Kante  $v_1$  aus einer

Farbklasse aus der Lösung entnimmt und danach durch eine andere Kante  $v_2$  derselben Farbklasse ersetzt. Diese Vorgehensweise garantiert, dass nach jedem Durchlauf des Algorithmus die Ausgabe eine zulässige, wenn auch nicht optimale Lösung darstellt.

Die neue Kante  $v_2$  kann z.B. nach folgenden Kriterien ausgewählt werden:

- Die Gradsumme der inzidenten Knoten an  $v_2$  ist nicht kleiner als die Gradsumme von  $v_1$  oder
- der Grad an einem der inzidenten Knoten an  $v_2$  ist nicht kleiner als der Grad an  $v_1$  oder
- zufällige Auswahl.

Ein Tausch wird nur dann durchgeführt, wenn die Lösung dadurch besser wird. Dieses Vorgehen wird so lange wiederholt, bis entweder

- eine bestimmte Anzahl  $t$  von Austauschschritten durchgeführt wurde oder
- die untere Schranke aus Abschnitt 2.3.2 erreicht wurde.

Der Verbesserungsalgorithmus benötigt zur Wahl der zu vertauschenden Knoten  $p(q - 1)$  Schritte, da für jede der  $p$  Farbklassen getestet werden muss, ob eines der maximal  $q - 1$  Elemente in derselben Farbklasse eine Verbesserung bringt. Da der Algorithmus maximal  $t$  mal durchgeführt wird, bleibt die Laufzeit polynomiell in den Eingabegrößen.

## 5.5 MRS mit Schlingen

Der nun folgende Abschnitt kommt dem ursprünglichen Problem PHP wieder ein Stück näher. Ab jetzt sind in den Instanzen für das MRS auch Schlingen zugelassen. Die Schlingen entsprechen homozygoten Genomen oder Genomvektoren ohne mehrdeutige Stellen im PHP.

**Bemerkung.** Wenn Schlingen zugelassen werden hat, der Graph keine zulässige Färbung mehr.

Eine Instanz für das MRS könnte dann wie in Abbildung 5.10 aussehen. Folgende

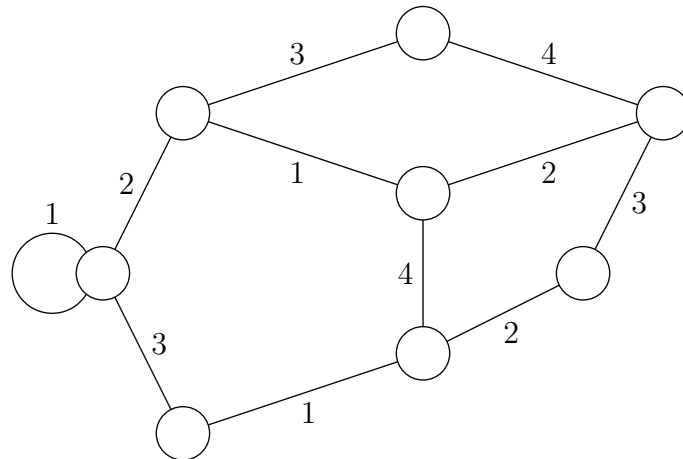


Abbildung 5.10: Instanz für das MRS mit Schlingen

Definition erscheint dann sinnvoll.

**Definition 5.2.** Eine Kantenfärbung eines Graphen heißt zulässig im weiteren Sinne, wenn der Graph nach Löschung aller Schlingen eine zulässige Kantenfärbung hat.

Die Abschätzungen aus Satz 5.6 gelten in diesem Fall auch. Sie können sogar unter bestimmten Voraussetzungen weiter verbessert werden.

**Satz 5.13.** Sei  $G = (V, E)$  eine Instanz für das MRS und an einem der Knoten, an denen der Maximalgrad  $\Delta$  angenommen wird, liegt auch eine Schlinge an. Dann gilt

$$n^* \leq 2p - \Delta(G) + 1.$$

*Beweis:*

Der Beweis funktioniert analog zum Beweis von Satz 5.6, mit dem Unterschied, dass mit dem Knoten mit Maximalgrad und der Schlinge nun  $\Delta(G) - 1$  Farben mit

$\Delta(G) - 1$  Knoten abgedeckt werden können. Für die restlichen  $p - \Delta(G) + 1$  Farben werden maximal doppelt so viel Knoten benötigt. Also gilt

$$\begin{aligned} n^*(G^*) &\leq \Delta(G) - 1 + 2(p - \Delta(G) + 1) \\ &= 2p - \Delta(G) + 1. \end{aligned}$$

□

Die Instanzen, die wir im Folgenden betrachten wollen, sind von Instanzen des PHP abgeleitet. Das bedeutet, dass die Farbe, die eine Schlinge hat, nur dort und an keiner anderen Kante (auch keiner anderen Schlinge) auftritt.

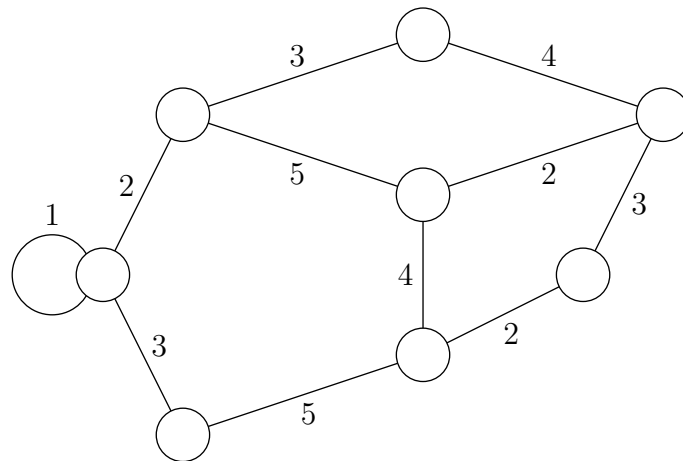


Abbildung 5.11: Instanz des MRS mit obiger Zusatzannahme

Mit dieser Zusatzannahme kann die obige Schranke weiter verbessert werden.

**Satz 5.14.** *Sei  $G$  eine Instanz des MRS, die den beschriebenen Eigenschaften genügt und  $s$  die Anzahl der Schlingen (homozygote Genome). Dann gilt:*

1. *wenn an allen Knoten mit Maximalgrad eine Schlinge anliegt, ist*

$$n^* \leq 2p - \Delta(G) - s + 2.$$

2. wenn es einen Knoten mit Maximalgrad gibt, an dem keine Schlinge anliegt, ist

$$n^* \leq 2p - \Delta(G) - s + 1.$$

*Beweis:*

Die Beweise laufen analog zu den vorherigen.

1. An einem Knoten mit Maximalgrad können  $\Delta(G) - 1$  Farben mit  $\Delta(G) - 1$  Knoten abgedeckt werden. Die restlichen  $s - 1$  Schlingen können mit ebensovielen Knoten abgedeckt werden und für die restlichen Farben werden höchstens zwei zusätzliche Knoten pro Farbe benötigt. Es gilt

$$\begin{aligned} n^*(G^*) &\leq \Delta(G) - 1 + s - 1 + 2(p - \Delta(G) + 2 - s) \\ &= 2p - \Delta(G) - s + 2. \end{aligned}$$

2. An einem der Knoten mit Maximalgrad, an dem keine Schlinge hängt, können  $\Delta(G)$  Farben mit  $\Delta(G) + 1$  Knoten abgedeckt werden. Die  $s$  Schlingen benötigen  $s$  Knoten und für die restlichen  $p - \Delta(G) - s$  sind maximal doppelt so viele Knoten erforderlich. Also gilt

$$\begin{aligned} n^*(G^*) &\leq \Delta(G) + 1 + s + 2(p - \Delta(G) - s) \\ &= 2p - \Delta(G) - s + 1. \end{aligned}$$

□

**Bemerkung.** Zur Behandlung dieser Art von Instanzen kann der in Kapitel 3 beschriebene Algorithmus Clarks Rule verwendet werden. Wenn der Graph  $G$  Schlingen enthält, bedeutet das, dass der Algorithmus auf jeden Fall einen Startpunkt findet. Wenn der Clark-Pfad irgendwann zu Ende sein sollte, kann der Algorithmus mit jeder Schlinge wieder neu gestartet werden. Falls es keine Schlingen mehr gibt, kann auf den Restgraph einer der Algorithmen GREEDY oder GREEDYN angewandt werden.

## 6 Ausblick

An dieser Stelle sind einige Fragestellungen zusammengefasst, die für zukünftige Betrachtungen interessant erscheinen, aber im Rahmen dieser Arbeit nicht beantwortet wurden.

- Es bleibt die Frage nach der Existenz eines guten approximativen Algorithmus für das PHP mit drei mehrdeutigen Stellen. Ist es möglich, die Vorgehensweise aus Abschnitt 2.4.3 zu modifizieren um diesen Fall abzudecken?
- Gibt es andere spezielle Graphenklassen, für die man bessere Resultate als die in Satz 5.6 beschriebenen erzielen kann? Ansätze dafür sind z.B. der Zusammenhangsgrad oder die Komponentenanzahl der Graphen.
- Welche anderen Modelle können eventuell helfen, einen Zugang zum PHP zu bekommen und die erreichten Resultate zu verbessern?

# Literaturverzeichnis

- [Bon03] Bonizzoni, P; Della Vedova, G.; Dondi, R. und Li, J.: The Haplotyping Problem: An Overview of Computational Models and Solutions, *Journal of Computer Science and Technology*, Vol. 18, Issue 6, pp. 675-688, 2003
- [Cla90] Clark, A.: Inference of Haplotypes from the PCR-amplified Samples of Diploid Populations, *Mol. Biol. Evol.* 7(2), pp. 111-122, 1990
- [Ies06] van Iersel, L.; Keijsper, J.; Kelk, S. und Stougie, L.: Beaches of islands of tractability: Algorithms for parsimony and minimum perfect phylogeny haplotyping problems, *Lecture Notes in Computer Science*, Vol. 4175, pp. 80-91, 2006
- [Lan01] Lancia, G. et al.: SNPs Problems, Complexity and Algorithms, *Lecture Notes in Computer Science*, Vol. 2161, pp. 182-193, 2001
- [Lan04] Lancia, G.; Pinotti, M. C. und Rizzi, R.: Haplotyping Populations by Pure Parsimony: Complexity of Exact and Approximation Algorithms, *INFORMS Journal on Computing*, Vol. 16, No 4, Fall 2004, pp. 348-359, 2004
- [Lan06] Lancia, G. und Rizzi, R.: A polynomial case of the parsimony haplotyping problem, *Operations Research Letters*, Vol. 34, issue 3, pp.289-295, 2006
- [Lip02] Lippert, R.; Schwartz, R.; lancia, G. und Istrail, S.: Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem, *Briefings in Bioinformatics*, Vol. 3, no. 1, pp. 23-31, 2002

- [Sha05] Sharan, R.; Halldórsson, B. und Istrail, S.: Islands of Tractability for Parsimony Haplotyping, *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, 2005

# Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Freiberg, den 16. März 2006.

Maria Koch