Mathematische Aspekte des Zauberwürfels

Bachelorarbeit

Fabian Müller
Matrikel-Nr. 2672340

20. September 2013

Erstprüfer: Prof. Dr. Heribert Vollmer
Zweitprüfer: Dr. Arne Meier

Institut für Theoretische Informatik
Leibniz Universität Hannover

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Hannover, den 13. Oktober 2013

Fabian Müller

Übersicht

Diese Arbeit wird sich mit der Lösbarkeit von Magic-Cubes beschäftigen. Der Magic-Cube soll mathematisch modelliert werden. Dabei sollen drei verschiedene Entscheidungsprobleme angegeben werden. Abschließend werden wir uns mit der Lösung dieser Probleme auseinandersetzen und mit Hilfe eines Algorithmus zeigen, dass die Entscheidungsprobleme in der Komplexitätsklasse L liegen.

Inhaltsverzeichnis

1 Einleitung

Der Magic-Cube wurde Mitte der 1970er Jahre von dem ungarischen Professor für Architektur Ernő Rubik erfunden. Ursprünglich sollte der Würfel das räumliche Denkvermögen von Rubiks Studenten spielerisch fördern. Seit 1977 ist der Magic-Cube auf dem freien Markt erhältlich.

Anfang der 1980er Jahre erreichte der Magic-Cube Kultstatus und wurde schnell zum Thema wissenschaftlicher Arbeiten. Im Jahr 2010 wurde gezeigt, dass „Gods Number“ für den Magic-Cube mit Kantenlänge drei – nach einer gängigen Metrik – gleich 20 ist [1]. Das bedeutet, dass jede lösbare Konfiguration des Magic-Cube mit Kantenlänge drei in maximal 20 Bewegungen gelöst werden kann. In der Arbeit „Algorithms for Solving Rubik's Cubes“ [4] wird unter anderem gezeigt, dass ein effizienter „Gods Algorithm“ für Magic-Cubes mit Kantenlänge n nur dann existiert, wenn $\mathsf{P} = \mathsf{NP}$ gilt.

Heutzutage gibt es Weltmeisterschaften in den verschiedensten Disziplinen des Magic-Cube. In diesen Wettbewerben geht es darum, Magic-Cubes möglichst schnell oder mit wenig Bewegungen zu lösen. Oft kommen auch noch erschwerende Maßnahmen hinzu. So ist es den Teilnehmern einiger Wettbewerbe beispielsweise nur erlaubt eine Hand zu benutzen oder sie müssen dabei eine Augenbinde tragen.

Ziel dieser Arbeit ist es, die Lösbarkeit von Magic-Cubes zu untersuchen. Wir gehen dabei von folgendem Szenario aus: Ein Teilnehmer eines Wettbewerbs ist gerade mit dem Lösen eines Würfels beschäftigt, als er ihm auf den Boden fällt. Der Würfel zerspringt in seine Einzelteile. Den Wettbewerb kann er nicht mehr gewinnen, allerdings interessiert uns das an dieser Stelle auch nicht. Wir wollen uns mit der Frage beschäftigen, ob der Magic-Cube noch lösbar ist, wenn man ihn wieder zufällig zusammensetzt.

Zur Beantwortung dieser Frage werden wir in Kapitel 2 dieser Arbeit die mathematischen Grundlagen des Magic-Cube klären. Zunächst wird eine intuitive Darstellung des Magic-Cube betrachtet, die auf den einzelnen Bestandteilen des Würfels aufbaut. Diese eignet sich gut, um die Eigenschaften des Magic-Cube genau abzubilden. Wir werden bereits in diesem Modell zwischen zwei Magic-Cube Varianten unterscheiden. Es gibt zum einen den mechanischen Magic-Cube – die „normale“ Variante, wie sie im Handel zu erwerben ist – und zum anderen den magnetischen Magic-Cube – eine allgemeinere Variante – die wir später noch genauer betrachten. Danach werden wir ein zweites Modell sehen, das auf der Graphentheorie aufbaut. Die Algorithmen – die wir in Kapitel 3 angeben – werden dann mit der Graphendarstellung arbeiten. Es wird des Weiteren ein Algorithmus angeben, der das intuitive Modell auf das Graphenmodell abbildet. Im letzten Abschnitt des Kapitels werden wir drei verschiedene Probleme der Lösbarkeit definieren.

In Kapitel 3 werden wir uns mit der Lösung der zuvor definierten Entscheidungsprobleme beschäftigen. Es werden die Unterschiede der Probleme verdeutlicht und mit Hilfe von bereits bekannten Lösungsmethoden die Komplexität der Entscheidungsprobleme auf die Klasse P eingegrenzt. Anschließend beschäftigen wir uns mit dem Entwickeln eigener Algorithmen, die die Probleme in logarithmischen Platz lösen können.

2 Grundlagen

In diesem Kapitel werden wir uns mit den Grundlagen des Magic-Cube vertraut machen. Wir werden zunächst ein intuitives Modell angeben, das den Magic-Cube mathematisch ausdrückt. Dieses Modell verwenden wir, um die Problemstellung genau abzubilden. Weiterhin werden wir ein graphentheoretisches Modell angeben, mit dem dann unsere Algorithmen aus Kapitel 3 arbeiten werden. Für die intuitive (geometrische) Darstellung werden wir einen Algorithmus angeben, der diese in die Graphendarstellung umwandelt. Die Ein- und Ausgabe des Algorithmus sollen dabei in der gleichen Raumkomplexität sein. Für die Graphendarstellung werden wir Bewegungen definieren und uns damit näher beschäftigen. Am Ende des Kapitels werden wir schließlich die Entscheidungsprobleme definieren, die wir uns dann in Kapitel 3 genauer anschauen werden.

2.1 Geometrische Darstellung des Magic-Cube

Wie oben bereits erwähnt werden wir zunächst den Magic-Cube mathematisch definieren. Es handelt sich dabei um eine geometrische Definition. Dafür müssen wir uns zunächst mit den Bestandteilen eines Magic-Cube vertraut machen. Wenn man einen Magic-Cube auseinander baut, erkennt man schnell, dass er aus verschiedenen Steinen besteht. Es gibt drei verschiedene Arten von Steinen. Diese sind: Ecksteine, Kantensteine und Mittelsteine. Sie besitzen unterschiedlich viele Oberflächen. Die Steine seien wie folgt definiert:

Definition 2.1. Ein *Stein* ist ein 2-Tupel $k = (i, d)$. Wobei i eine natürliche Zahl ist, die den Stein eindeutig bestimmt. d ist ein Tupel der Größe 1 bis 3 bestehend aus Zahlen (Farben) der Menge $\{1, 2, 3, 4, 5, 6\}$. Steine mit Tupel d der Größe $1, 2, 3$ werden im Folgenden als *Mittelsteine*, *Kantenteine*, *Ecksteine* bezeichnet.

Die Steine werden eindeutig über eine natürliche Zahl definiert. Die Färbung ist dabei bis zu einem gewissen Grad variabel. Zum Beispiel handelt es sich bei den Steinen $k_1 = (1, (1, 2, 3))$ und $k_1 = (1, (3, 1, 2))$ und dieselben. Ein Stein mit $d = (1, 3, 2)$ wiederum kann nicht den Wert $i = 1$ haben, wenn er zu dem gleichen Magic-Cube gehören soll zu dem k_1 gehört.

Definition 2.2. Sei $k = (i, d)$ ein Stein. Eine Permutation der gleichen Kombination von d heißt dann *Drehung* des Steins.

Um einen fertigen Magic-Cube zu erhalten, benötigen wir eine Menge von Steinen und eine Anordnung dieser Steine im Raum. Dazu definieren wir Positionen, denen die Steine zugeordnet werden können.

Definition 2.3. Eine *Position* ist ein 1 bis 3-Tupel aus natürlichen Zahlen.

Mit einer Menge von Positionen, einer Menge von Steinen und einer Zuordnung der Steine auf die Positionen können wir nun den fertigen Magic-Cube definieren. Allerdings machen wir noch einen kleinen Umweg über die Seiten.

Definition 2.4. Eine *Seite* S mit Kantenlänge $n \geq 2$ ist ein 3-Tupel (K, P, z). K ist die Menge der Steine, P die Menge der Positionen und $z\colon K \to P$ eine Funktion, die den Steinen eine Position zuordnet. Dabei gilt:

$$\forall k = (i, d) \in K, \forall p \in P : z(k_x) = p_y \implies |d| = |p|$$

Die Menge der Steine K (Menge der Positionen P) besteht aus n^2 verschiedenen Steinen (Positionen). Es gibt immer genau 4 Positionen für Ecksteine, $4 \cdot (n-2)$ für Kantensteine und $(n-2)^2$ für Mittelsteine. Für die Positionen gelten folgende Einschränkungen in Hinsicht auf die Nachbarschaft:

(i) Ein Stein $k = (i, d)$ hat genau $5 - |d|$ Adjazenten
(ii) Zwischen zwei Ecksteinen befinden sich mindestens $n - 2$ Kantensteine
(iii) Ein Eckstein ist niemals ein Adjazent eines Mittelsteins
(iv) Ein Kantenstein hat genau einen Mittelstein als Adjazenten

Als nächstes können wir den Magic-Cube als Vereinigung von sechs Seiten definieren. Die Mengen K und P dieser sechs Seiten sind dabei nicht disjunkt, sondern überschneiden sich teilweise.

Definition 2.5. Ein 3-dimensionaler *Magic-Cube* mit Kantenlänge $n \geq 2$ ist ein 3-Tupel (K, P, z). $K = \bigcup_{i=1}^{6} K_i$ ist die Menge der Steine, $P = \bigcup_{i=1}^{6} P_i$ die Menge der Positionen und $z\colon K \to P$ eine Funktion, die den Steinen eine Position zuordnet. Wobei K_i, P_i gemeinsam mit einer Funktion z_i eine Seite $S_i = (K_i, P_i, z_i)$ bilden. Für die Funktion z gilt:

$$\forall k \in K, \forall p \in P\colon z(k) = p \Leftrightarrow \exists i\colon z_i(k) = p$$

Es gelten folgende Eigenschaften für die Nachbarschaft der Seiten:

(i) Eine Seite hat immer genau 4 benachbarte Seiten
(ii) Für zwei benachbarte Seiten S_x und S_y gilt:

$$|K_x \cap K_y| = |P_x \cap P_y| = n$$

$$k = (i, d) \in K_x \cap K_y \implies |d| \geq 2$$

$$p \in P_x \cap P_y \implies |p| \geq 2$$

(iii) Für drei Seiten S_x, S_y und S_z, mit S_y und S_z sind benachbart mit S_x gilt:

$$|K_x \cap K_y \cap K_z| = |P_x \cap P_y \cap P_z| \leq 1$$

Beispiel 2.6. Magic-Cube mit Kantenlänge drei:

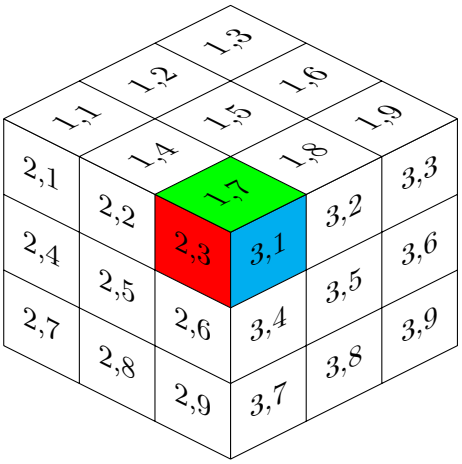

Abbildung 1: Der Stein $(i, (\text{grün,blau,rot}))$ ist an der Position $((1,7),(2,3),(3,1))$

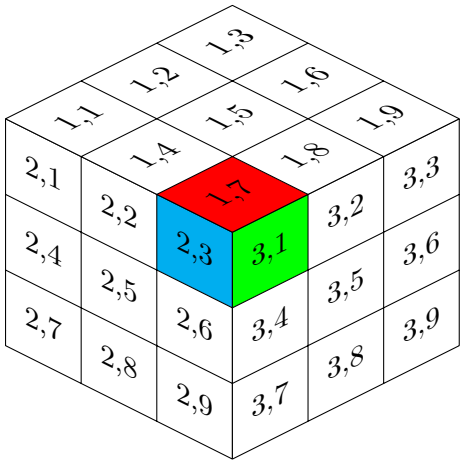

Abbildung 2: Der gleiche Stein mit einer anderen Drehung an der gleichen Position

Als Nächstes schauen wir uns die Menge der Positionen etwas genauer an. Diese lässt sich sinnvoll in Teilmengen zerlegen. Wir unterteilen die Menge P zunächst in die Menge der Positionen für Mittelsteine (P_m), für Kantensteine (P_k) und für Ecksteine (P_e). Die Positionenmengen für Mittel- und Kantensteine werden wir weiter in mehrere Mengen P_{m_i} und P_{k_i} aufteilen, sodass durch Bewegungen – die später in Kapitel 2.4 definiert werden – immer nur die Steine innerhalb dieser Teilmengen vertauscht werden können.

Definition 2.7. Die Menge der Positionen P lässt sich wie folgt unterteilen:

$$P = P_e \cup P_k \cup P_m,$$

wobei P_e die Menge der Positionen für Ecksteine ($|p| = 3$), P_k die Menge der Positionen für Kantensteine ($|p| = 2$) und P_m die Menge der Positionen für Mittelsteine ($|p| = 1$) ist.

Die Mengen P_k, P_m lassen sich weiter unterteilen. Für die Menge P_k gilt:

$$P_k = \bigcup_{i=1}^{\lfloor \frac{1}{2} \cdot (n-1) \rfloor} P_{k_i}$$

Dabei gilt für beliebige $p_x, p_y \in P_{k_i}$:

$$\min(d(p_x, p_z) \mid p_z \in P_e) = \min(d(p_y, p_z) \mid p_z \in P_e),$$

Für die Menge P_m gilt:

$$P_m = \bigcup_{i=1}^{x} P_{m_i}$$

Wobei $x = \sum_{i=0}^{\lfloor \frac{1}{2} \cdot (n-1) \rfloor} i$
Dabei gilt für beliebige $p_x, p_y \in P_{m_i}$:

$$\forall P_{k_j} \subseteq P_k : \min(d(p_x, p_z) \mid p_z \in P_{k_j}) = \min(d(p_y, p_z) \mid p_z \in P_{k_j})$$

Magic-Cubes mit ungerader Kantenlänge enthalten jeweils eine Positionenmenge für Kanten- und Mittelsteine, die sich von den anderen unterscheiden, da sich die Positionen dieser Menge genau in der Mitte jeder Seite befinden. Diese beiden Mengen werden später noch eine größere Rolle spielen, deshalb werden wir ihnen einen eigenen Namen geben.

Definition 2.8. Die Positionenmenge P_{k_i} (P_{m_i}) mit $|P_{k_i}| = 12$ ($|P_{m_i}| = 6$) wird im Folgenden mit der Formel P'_k (P'_m) abgekürzt.

Diese Mengen treten bei geraden Kantenlängen nicht auf, da es keine eindeutige Mitte gibt.

Beobachtung.

(i) Die Menge der Positionen P (Steine K) enthält $|P| = |K| = 6 \cdot n^2 - 12 \cdot n + 8$ verschiedene Elemente. Die Teilmengen P_e, P_k, P_m enthalten $|P_e| = 8, |P_k| = (n-2) \cdot 12, |P_m| = (n-2)^2 \cdot 6$ verschiedene Elemente.

(ii) Die Menge der Steine K enthält ebenfalls $|P| = |K| = 6 \cdot n^2 - 12 \cdot n + 8$ verschiedene Elemente

(iii) Die Menge der Positionen für Kantensteine P_k eines Magic-Cube mit Kantenlänge $n \geq 2$ besteht immer aus genau (n mod 2) 12-elementigen und $\lfloor \frac{n-2}{2} \rfloor$ 24-elementigen Teilmengen P_{k_i}.

(iv) Die Menge der Positionen für Mittelsteine P_m eines Magic-Cube mit Kantenlänge $n \geq 2$ besteht immer aus genau $a = (n \bmod 2)$ 6-elementigen, $b = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor - 2} i$ 48-elementigen und $c = \sum_{i=0}^{\lfloor \frac{1}{2} \cdot (n-1) \rfloor} i - a - b$ 24-elementigen Teilmengen P_{m_i}.

Um einen handelsüblichen Magic-Cube (im Folgenden *Magic-Cube$_{mech}$*) korrekt abzubilden, bedarf es noch einer Erweiterung. Es ist bei einem Magic-Cube$_{\mathsf{mech}}$ – ausgehend von einem bereits zusammengebauten Cube – aufgrund der Mechanik nicht möglich alle Kanten- und Mittelsteine beliebig miteinander zu vertauschen.

Definition 2.9. Ein Magic-Cube$_{\mathsf{mech}}$ mit Kantenlänge $n \geq 2$ ist ein Magic-Cube mit folgenden Einschränkungen in Hinsicht auf die Positionen der Kanten- und Mittelsteine:

(i) Sei K_k die Menge der Kantensteine ($|d| = 2$), dann gilt für den Magic-Cube$_{\mathsf{mech}}$:

$$K_k = \bigcup_{i=1}^{\lfloor \frac{1}{2} \cdot (n-1) \rfloor} K_{k_i}$$

Wobei Steine aus K_{k_i} immer nur Positionen aus P_{k_i} mit jeweils einer Drehung zugeordnet werden können. Die K_{k_i} enthalten dabei 24 Elemente.
Sonderfall: Bei ungeraden Kantenlängen gibt es immer jeweils eine Menge K_{k_i} mit zwölf Elementen. Für Steine dieser Menge gibt es keine Einschränkung bezüglich der Drehung.

(ii) Sei K_m die Menge der Mittelsteine ($|d| = 1$) und $x = \sum_{i=0}^{\lfloor \frac{1}{2} \cdot (n-1) \rfloor} i$, dann gilt für den Magic-Cube$_{\mathsf{mech}}$:

$$K_m = \bigcup_{i=1}^{x} K_{m_i}$$

Wobei Steine aus K_{m_i} immer nur Positionen aus P_{m_i} zugeordnet werden können.

(iii) Für Magic-Cube$_{\mathsf{mech}}$ mit ungerader Kantenlänge gibt es genau eine Menge $K_{m_i} \subseteq K_m$ mit sechs Elementen. Die Steine aus dieser Menge haben immer eine feste Position aus der Menge P'_m.

Beispiel 2.10.

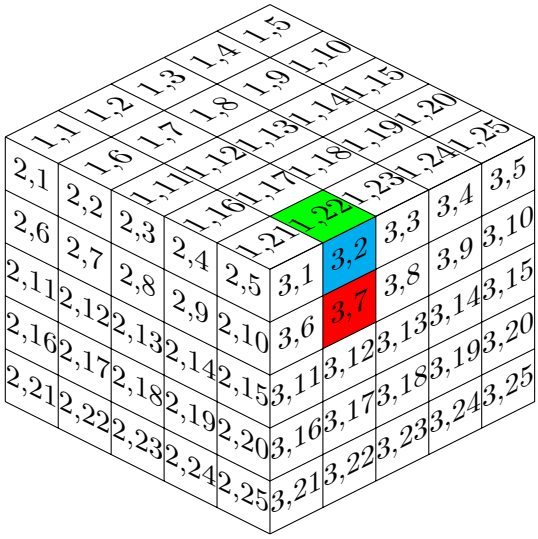

Abbildung 3: Vor dem Tausch: Die Steine $(i, (\text{grün,blau})), (j, (\text{rot}))$ sind an den Positionen $((1, 22), (3, 2)), ((3, 7))$

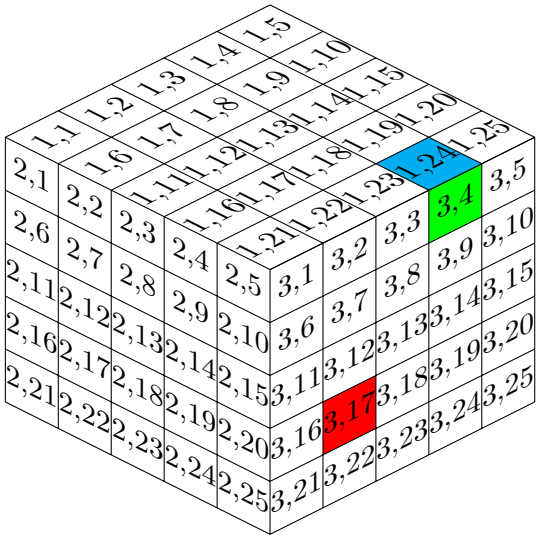

Abbildung 4: Nach dem Tausch: Die Steine $(i, (\text{blau,grün})), (j, (\text{rot}))$ sind an den Positionen $((1, 24), (3, 4)), ((3, 17))$

Wir haben also zwei verschiedene Arten von Magic-Cubes. Die nicht mechanische Variante findet man im Handel bislang noch nicht, man kann sie allerdings – z.B. mit Hilfe von Magneten – selbst herstellen, oder mit dem Computer simulieren. Sprechen wir im Folgenden von einem Magic-Cube, dann meinen wir die allgemeinere magnetische Variante. Die mechanische wird wie oben mit Magic-Cube$_{\mathsf{mech}}$ abgekürzt.

2.2 Graphendarstellungen des Magic-Cube

In diesem Kapitel werden wir ein weiteres Modell des Magic-Cube definieren. Der Magic-Cube wird in dieser Darstellung durch einen Graphen und eine Färbung ausgedrückt. Die Algorithmen, die wir später verwenden, werden ausschließlich mit dieser Darstellung arbeiten.

Definition 2.11. Ein zu Abbildung 5 isomorpher Graph $M_n = (V, E)$ heißt Graphendarstellung eines Magic-Cube.

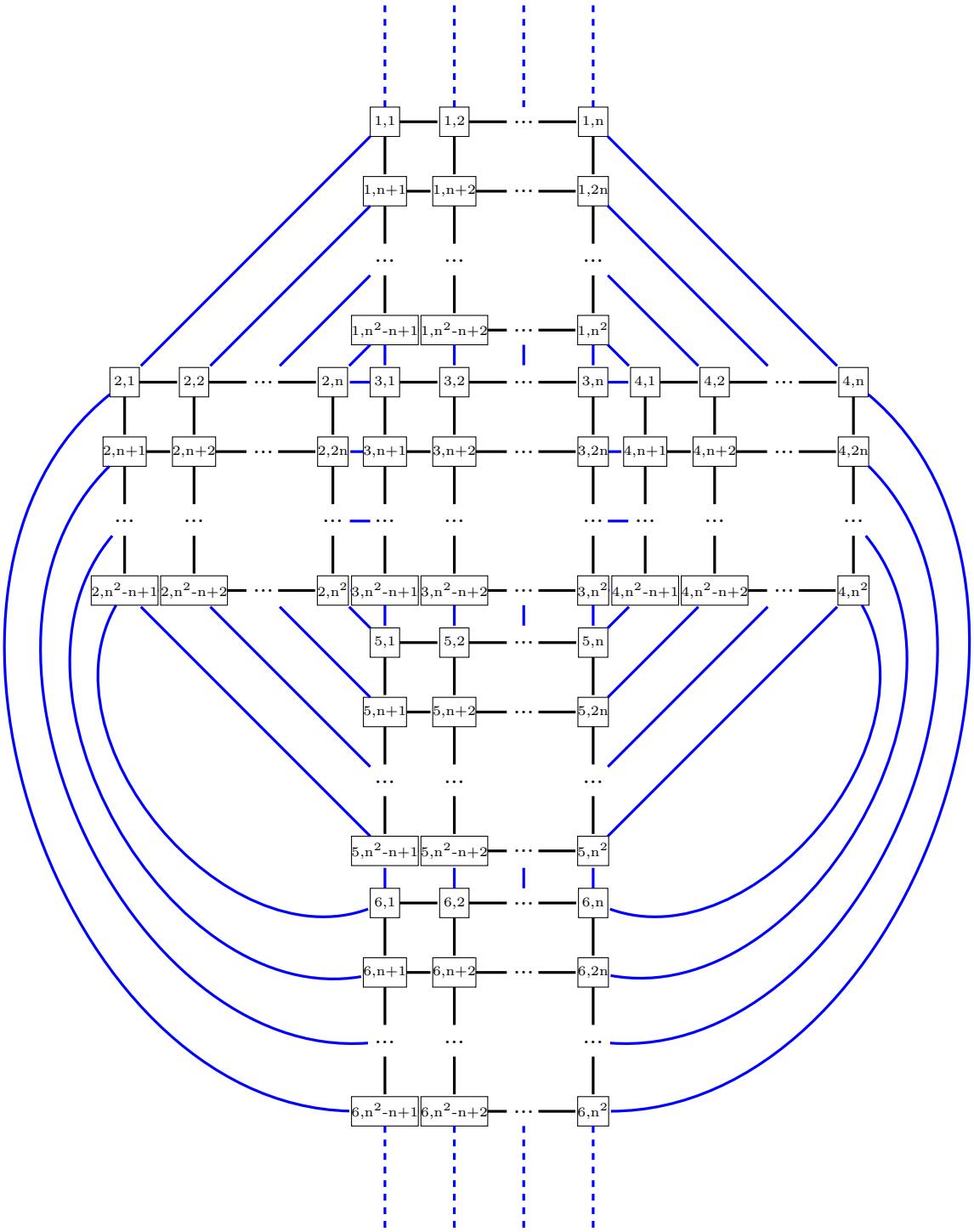

Abbildung 5: Der Knoten $v_{i,j}$ ist mit i, j beschriftet.

Beobachtung. Die Graphendarstellung eines Magic-Cube hat folgende Eigenschaften:
(i) $M_n = (V, E)$ ist 4-regulär
(ii) $M_n = (V, E)$ ist planar

Die Graphendarstellung eines Magic-Cube besteht aus verschiedenen Teilgraphen, die wir *Oberflächen* nennen.

Definition 2.12. Sei $M_n = (V, E)$ die Graphendarstellung eines Magic-Cube. Ein Teilgraph $O_i = (V_i, E_i)$ von M_n, mit $V_i = \{v_{i,k} \mid k \in \{1, \ldots, n^2\}\}$ und $E_i = E \cap V_i \times V_i$ heißt *Oberfläche*.

Beobachtung.
Die Kantenmenge E_i einer Oberfläche O_i besteht aus folgenden Kanten:

$$E_i = \bigcup_{\substack{k=1,\\ k \bmod n \neq 0}}^{n^2} (v_{i,k}, v_{i,k+1}) \cup \bigcup_{\substack{k=1,\\ 0<k \bmod n^2 < n^2-n \neq 0}}^{n^2} (v_{i,k}, v_{i,k+n})$$

Für die geometrische Darstellung wurde der Begriff Oberfläche nicht definiert, aber er wäre leicht zu übertragen. In den Abbildungen 6 und 7 ist der Unterschied zwischen den beiden Begriffen graphisch veranschaulicht. Es ist dabei festzuhalten, dass zu der Seite in Abbildung 6 noch sechs Flächen gehören, die aus perspektivischen Gründen nicht zu sehen sind.

Beispiel 2.13.

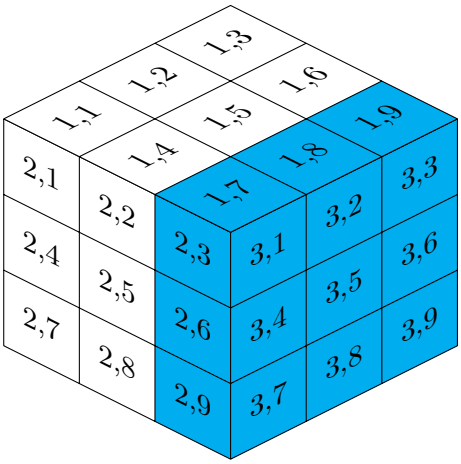

Abbildung 6: Seite eines Magic-Cube

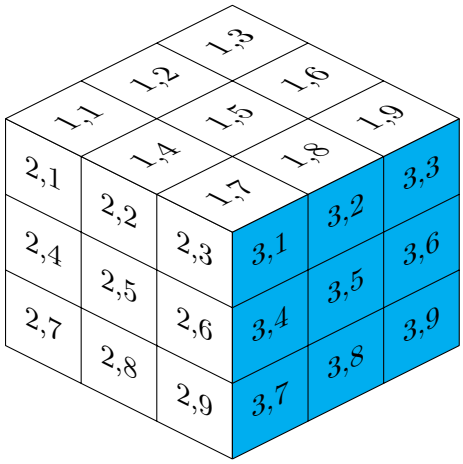

Abbildung 7: Oberfläche eines Magic-Cube

Zur Vervollständigung der graphentheoretischen Abbildung des Magic-Cube fehlt noch die Färbung der Knoten. Die Färbung ist eine Funktion, die die Knoten auf die Zahlen 1 bis 6 abbildet.

Definition 2.14. Eine Funktion $f\colon V \to \{1, 2, 3, 4, 5, 6\}$ heißt *Färbung* von M_n.

2.3 Zusammenhang zwischen den beiden Darstellungen

In diesem Unterkapitel werden wir den Zusammenhang zwischen geometrischer Darstellung und Graphendarstellung untersuchen. Dazu werden wir zunächst einen Algorithmus angeben, der aus einem Magic-Cube in geometrischer Darstellung einen Magic-Cube in Graphendarstellung erzeugt. Anschließend werden wir die verschiedenen Bestandteile der geometrischen Darstellung auch in der Graphendarstellung angeben.

Folgender Algorithmus wandelt einen Magic-Cube in geometrischer Darstellung in die Graphendarstellung um:

Eingabe : $\langle K, P, z\rangle$, die geometrische Darstellung eines Magic-Cube

1 **for** *Position* $p \in P$ **do**
2 **for** $c \leftarrow 0$ **to** $|p| - 1$ **do**
3 $V \leftarrow v_{(p_{c_1}, p_{c_2})}$
4 $f(v_{(p_{c_1}, p_{c_2})}) \leftarrow d_c$, mit $z(k) = p$, für einen Stein $k = (i, d)$
5 **for** $g \leftarrow 0$ **to** c **do**
6 $E \leftarrow (v_{(p_{g_1}, p_{g_2})}, v_{(p_{c_1}, p_{c_2})})$
7 **end**
8 **end**
9 **end**
10 **for** $v_{(a,b)} \in V$ **do**
11 **for** $u_{(c,d)} \in V$ **do**
12 **if** $a = c$ **and** $(c + 1 = d$ **or** $c + n = d)$ **then**
13 $E \leftarrow (v_{(a,b)}, v_{(c,d)})$
14 **end**
15 **end**
16 **end**

Ausgabe : $\langle M_n^z(K) = (V, E), f\rangle$

Algorithmus 1 : Geometrische Darstellung in Graphendarstellung

Platzbedarf der Ausgabe: In Zeile 1 bis 4 werden für jede Position p Knoten entsprechend der Elemente in p erzeugt und jedem dieser Knoten wird ein Farbwert zugewiesen. Da $|p|$ für alle Positionen nicht größer als drei werden kann, benötigen die Knotenmenge V und die Funktion f maximal das Dreifache an Speicher der Positionen. In Zeile 5 bis 7 werden für jede Position maximal drei Kanten erzeugt (diese werden teilweise überschrieben). Für jeden Knoten $v_{(a,b)}$ werden in Zeile 13 maximal zwei neue Kanten erzeugt. Die Kantenmenge E benötigt also $O(3 \cdot |P| + 2 \cdot |V|)$ Platz. Insgesamt hat die Ausgabe einen Platzbedarf von $O(3 \cdot |P| + 3 \cdot |V|) = O(12 \cdot |P|)$ und befindet sich somit in der gleichen Raumkomplexität wie die Eingabe.

Beobachtung.

(i) Alle Knoten die durch blaue Kanten verbunden sind, entsprechen einer Position der geometrischen Darstellung.
(ii) Eine Kante $(v_{x,y}, v_{x,z})$ innerhalb einer Oberfläche (schwarz) steht für zwei benachbarte Positionen p_a, p_b, mit $v_{x,y} \in p_a$, $v_{x,z} \in p_b$ in der geometrischen Darstellung.
(iii) Ein Stein $k = (i, d)$ auf der Position p der geometrischen Darstellung entspricht der Färbung $f(p_i) = d_i$, mit $p_i \in p$, $d_i \in d$.

Im Folgenden können wir also auch die Begriffe Position und Stein in der Graphendarstellung verwenden.

2.4 Bewegungen

Zur vollständigen Beschreibung des Magic-Cube fehlt noch ein wichtiger Bestandteil, der bislang weder in der geometrischen noch in der Graphendarstellung definiert wurde. Es handelt sich dabei um die Bewegungen der einzelnen Steine (bzw. Farben der Knoten). Wir werden diese Bewegungen nur für die Graphendarstellung definieren, da unsere Algorithmen mit dieser arbeiten werden.

Dazu werden wir zunächst Kreise definieren, auf denen sich die Farben der Knoten jeweils verschieben können. Diese Kreise teilen wir auf in Kreise der Oberflächen und Kreise des Magic-Cube.

Definition 2.15. Sei $O_i = (V_{O_i}, E_{O_i})$ eine Oberfläche eines $M_n = (V, E)$. Ein *Bewegungskreis der Oberfläche* B_{O_i} von O_i ist ein Kreis in dem Graphen O_i. Ein Knoten $u \in V_{O_i}$ befindet sich auf dem k-ten Bewegungskreis $B_{O_i,k}$ von O_i, wenn gilt:

$$k = \min\{d(u, v) \mid v \in V \setminus V_{O_i}\}$$

Ein Bewegungskreis einer Oberfläche besteht also aus den Knoten der Oberfläche, die alle die gleiche minimale Entfernung zu einer anderen Oberfläche haben. Bei ungeraden Kantenlängen gibt es pro Oberfläche einen Bewegungskreis mit einem Knoten, der zu der Menge P'_m gehört. Da diese Bewegungskreise einelementig sind und somit keine Auswirkungen auf unsere Betrachtung haben, werden wir diese im Folgenden ignorieren.

Beispiel 2.16. Bewegungskreise einer Oberfläche $O_i \subset M_5$:

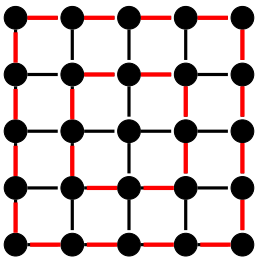

Abbildung 8

Beobachtung.

(i) Eine Oberfläche $O_i = (V_{O_i}, E_{O_i}) \subset M_n$ hat immer genau $\lfloor \frac{n}{2} \rfloor$ verschiedene Bewegungskreise.

(ii) Der k-te *Bewegungskreis* von O_i enthält $4(n - 2 \cdot k + 1)$ Knoten.

Nun fehlt noch die Definition der Bewegungskreise des Magic-Cube. Diese Kreise enthalten Knoten von vier verschiedenen Oberflächen.

Definition 2.17. Ein *Bewegungskreis des Magic-Cube* B von M_n ist ein Kreis bestehend aus $4 \cdot n$ Knoten von insgesamt vier verschiedenen Oberflächen. Für zwei Knoten $u, v \in B$ gilt:

$$\forall \text{Oberflächen } O\colon B \cap O = \emptyset\colon \min\{d(u,x) \mid x \in O\} = \min\{d(v,x) \mid x \in O\}$$

Die Bewegungskreise des Magic-Cube bestehen also aus Knoten von vier Oberflächen und sind über die Abstände zu den nicht beteiligten Oberflächen definiert. Es gibt für jeden Bewegungskreis eines Magic-Cube genau zwei Oberflächen, die nicht an dem Kreis beteiligt sind. Die Mindestabstände der Knoten des Bewegungskreises müssen zu jeder dieser Oberflächen gleich sein.

Beispiel 2.18. Alle Bewegungskreise von M_3 und dessen Seiten:

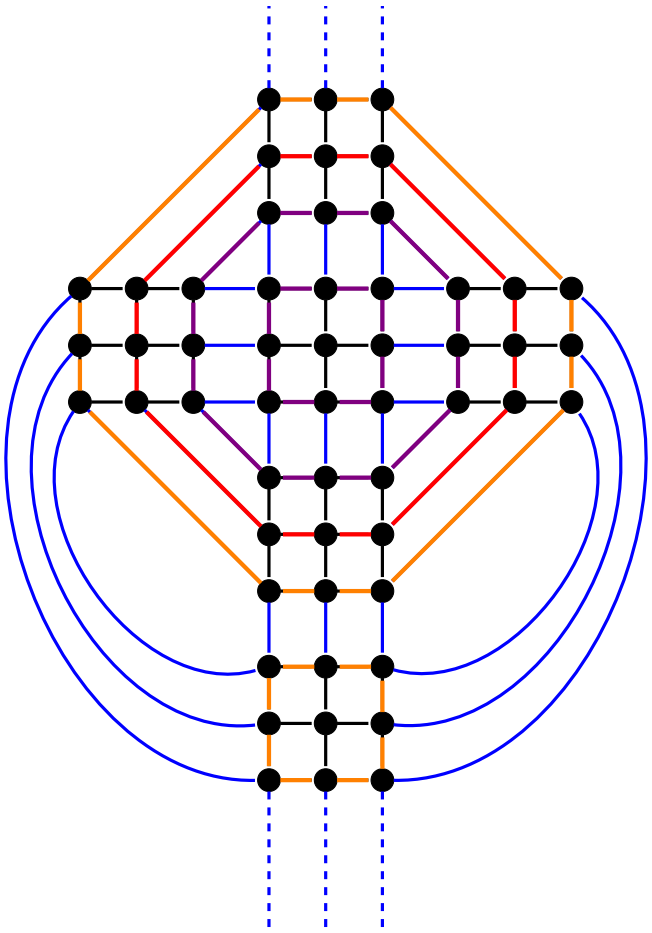

Abbildung 9

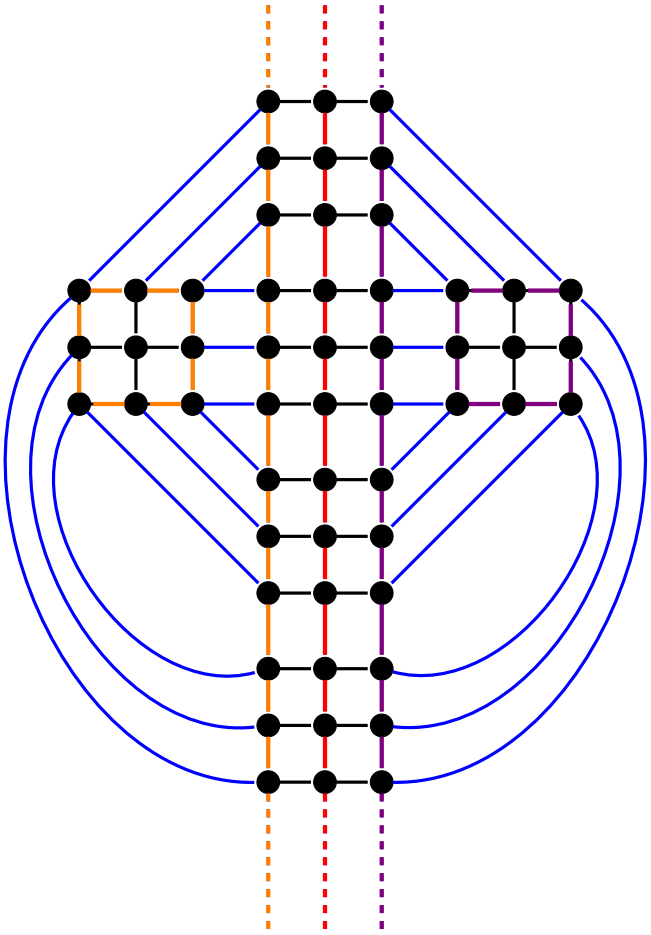

Abbildung 10

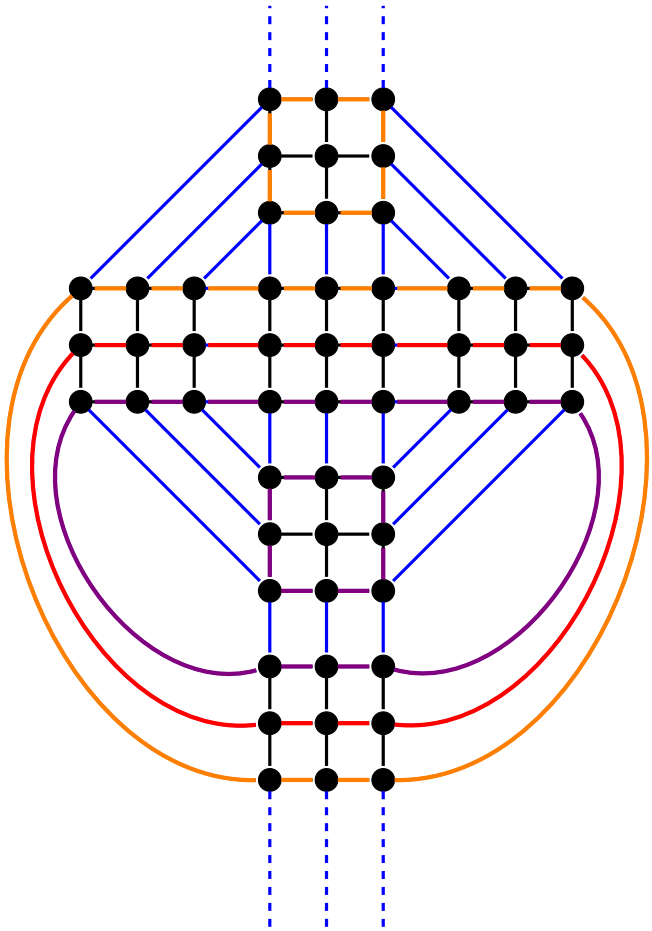

Abbildung 11

Mit Hilfe der verschiedenen Bewegungskreise können wir nun Bewegungen definieren. Es handelt sich dabei um Verschiebungen der Farben auf den entsprechenden Kreisen.

Definition 2.19. Eine *Bewegung* von M_n ist eine Färbung, die jedem Knoten aus V die Farbe seines n-ten Vorgängers auf einem Bewegungskreis in dem Graphen M_n zuordnet. Die Farben der Knoten, die sich nicht auf entsprechendem Bewegungskreis befinden, werden dabei nicht verändert.

Sonderfall: Enthält der Bewegungskreis zwei Knoten, die zu der gleichen Eckposition gehören, dann gehören die Bewegungskreise der eingeschlossenen Oberfläche (die den dritten Knoten der Position enthält) mit zur Bewegung. Dabei wandern die Farben innerhalb der Oberfläche immer um $n - 2k + 1$ Knoten auf dem k-ten Bewegungskreis der Oberfläche.

Es gibt also zwei Arten von Bewegungen, wobei immer genau ein Bewegungskreis eines Magic-Cube B zugrunde liegt. Des Weiteren werden die Farben der Knoten auf

den Bewegungskreisen einer Oberfläche genau dann verschoben, wenn diese Oberfläche von B eingeschlossen wird. Das bedeutet, dass die minimale Entfernung der Oberfläche zu den Knoten des Bewegungskreises gleich eins ist.

Beispiel 2.20. Bewegungen 2 des M_3:

v	$v_{1,4}$	$v_{1,5}$	$v_{1,6}$	$v_{2,2}$	$v_{2,5}$	$v_{2,8}$	$v_{4,2}$	$v_{4,5}$	$v_{4,8}$	$v_{5,4}$	$v_{5,5}$	$v_{5,6}$
$f(v)$	rot	rot	rot	gelb	gelb	gelb	blau	blau	blau	grün	grün	grün
$b_2(f(v))$	gelb	gelb	gelb	grün	grün	grün	rot	rot	rot	blau	blau	blau

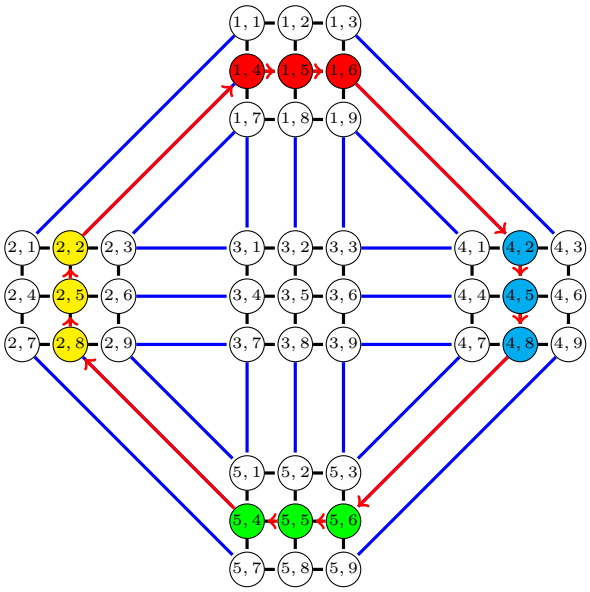

Abbildung 12: Färbung vor Bewegung 2

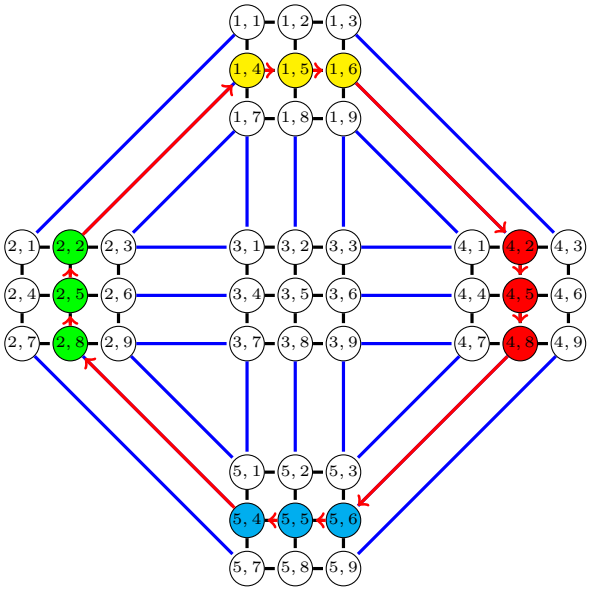

Abbildung 13: Färbung nach Bewegung 2

Beobachtung.

(i) Für einen M_n gibt es $6n$ verschiedene Bewegungen entlang der $3n$ verschiedenen Bewegungskreise in beide Richtungen.

(ii) Bewegungen sind nicht kommutativ.

Notation:

(i) Seien f, f' zwei Färbungen und f' die Färbung nach den Bewegungen $i, \ldots, j$ (ausgehend von f). Die folgende Notation ist äquivalent zu $f' = b_j(\ldots(b_i(f(v))))$:

$$f' = ((f(v), (i, \ldots, j)).$$

(ii) Seien f, f' zwei Färbungen und f' ist die Färbung nach einer k-fachen Wiederholung der Bewegungsabfolge i (d.h. i kann für mehrere Bewegungen stehen). Die folgende Notation ist äquivalent zu $f' = (f(v), (i, \ldots, i))$:

$$f' = ((f(v), ([i]^k)).$$

Mit den verschiedenen Bewegungen können wir nun bestimmte Muster bzw. Färbungen erzeugen. Unser Ziel ist es eine Färbung zu erreichen, bei der alle Oberflächen gleich gefärbt sind.

Definition 2.21. Ein M_n gilt als *gelöst*, wenn für alle Oberflächen O gilt:

$$\forall u, v \in V_O \colon f(v) = f(u)$$

Eine *Lösung* ist eine Abfolge von *Bewegungen*, die einen M_n in den gelösten Zustand bringt. Ein M_n heißt *lösbar*, wenn eine Lösung für ihn existiert. Die geometrische Darstellung eines Magic-Cube ist genau dann lösbar, wenn die Graphendarstellung des Magic-Cube lösbar ist.

2.5 Entscheidungsprobleme

Mit der Definition von Lösbarkeit stellt sich die Frage, wann ein Magic-Cube lösbar ist. Man könnte annehmen, dass dies generell der Fall ist, da die Würfel im Handel in der Regel in einer gelösten Konfiguration verkauft werden. Baut man einen Magic-Cube allerdings auseinander und setzt ihn zufällig neu zusammen (unter den Regeln, die in Kapitel 1 aufgestellt wurden), dann ist nicht garantiert, dass der Würfel danach noch lösbar ist. Um die Lösbarkeit zu untersuchen, definieren wir zunächst die verschiedenen Entscheidungsprobleme.

Definition 2.22. Sei $\langle K, P, z\rangle$ die geometrische Darstellung eines beliebigen genau 6-farbigen Magic-Cube (Magic-Cube$_{\mathsf{mech}}$) mit Kantenlänge n. Die Mengen K und P haben dabei die Eigenschaft, dass eine Funktion $z' \colon K \to P$ existiert, sodass $\langle K, P, z'\rangle$ lösbar ist. Des Weiteren sei $M_n^z(K), f$ die durch Algorithmus 1 berechnete Graphendarstellung des Magic-Cube. Das Entscheidungsproblem ist dann wie folgt definiert:

$$\mathsf{MagicCubeSOL} = \left\{ \langle M_n^z(K), f\rangle \;\middle|\; \begin{array}{l}\langle M_n^z(K), f\rangle \text{ ist die Graphendarstellung eines} \\ \text{Magic-Cube. } M_n^z(K) = (V, E) \text{ ist unter Färbung} \\ f \text{ lösbar.}\end{array} \right\}.$$

$$\mathsf{MagicCubeSOL}_{\mathsf{mech}} = \left\{ \langle M_n^z(K), f\rangle \;\middle|\; \begin{array}{l}\langle M_n^z(K), f\rangle \text{ ist die Graphendarstellung eines} \\ \text{Magic-Cube}_{\mathsf{mech}}. \; M_n^z(K) = (V, E) \text{ ist unter} \\ \text{Färbung } f \text{ lösbar.}\end{array} \right\}.$$

Das Entscheidungsproblem ist also auf Grundlage der Graphendarstellung definiert. Durch die Anforderung, dass die Graphendarstellung mit Hilfe von Algorithmus 1 aus einer geometrischen Darstellung zu berechnen ist, werden die Eigenschaften der geometrischen auf die Graphendarstellung übertragen. Zudem soll es für die geometrische Darstellung mindestens eine lösbare Konfiguration geben. Diese Eigenschaft resultiert aus der Annahme, dass ein lösbarer Magic-Cube heruntergefallen und in seine Bestandteile zersprungen ist. Wir gehen auch davon aus, dass es sich um einen Magic-Cube mit genau sechs Farben handelt.

Das Problem lässt sich noch weiter verallgemeinern. Wir können ein Problem ohne die oben geforderten Eigenschaften definieren, das nur noch auf der Graphendarstellung aufbaut.

Definition 2.23. Sie M_n die Graphendarstellung eines beliebigen Magic-Cube und f eine Färbung von M_n.

$$\mathsf{MagicCubeColorability} = \left\{ \langle M_n, f \rangle \;\middle|\; \begin{array}{l} M_n \text{ ist ein Graph } G = (V, E) \text{ und } f \text{ eine} \\ \text{Färbung von } M_n. \; M_n \text{ ist lösbar.} \end{array} \right\}.$$

Wir verlangen bei MagicCubeColorability also weder, dass es eine lösbare Konfiguration gibt, noch, dass der Würfel mit genau sechs Farben gefärbt ist. MagicCubeColorability entspricht der Frage nach der Lösbarkeit eines Magic-Cube, dessen Knoten (Flächen) zufällig gefärbt (beklebt) werden. Mit der Definiton von MagicCubeColorability beenden wir dieses Kapitel. Wir werden im Weiteren die Komplexität der drei Entscheidungsprobleme analysieren.

3 Ergebnisse

In diesem Kapitel werden wir die Unterschiede der drei verschiedenen Entscheidungsprobleme verdeutlichen und untersuchen, inwiefern wir bereits bekannte Lösungsmethoden und Erkenntnisse verwenden können, um die Komplexität der Entscheidungsprobleme auf die Klasse P einzuschränken. Abschließend geben wir eigene Algorithmen zur Lösung der Probleme an, deren Platzbedarf sich logarithmisch zu Eingabe verhalten wird.

3.1 Beziehungen zwischen den Entscheidungsproblemen

Zunächst werden wir uns mit den Unterschieden der Entscheidungsprobleme beschäftigen. Dafür werden wir uns verschiedene Instanzen anschauen und überprüfen für welche Probleme sie gültig sind.

Beobachtung.

$$\mathsf{MagicCubeSOL}_{\mathsf{mech}} \subsetneq \mathsf{MagicCubeSOL} \subsetneq \mathsf{MagicCubeColorability}.$$

Beispiel 3.1. $\mathsf{MagicCubeSOL}_{\mathsf{mech}} \subsetneq \mathsf{MagicCubeSOL}$:
Sei folgender Graph eine Oberfläche eines bereits gelösten M_5, der jeweils eine gültige Instanz für beide Probleme darstellt.

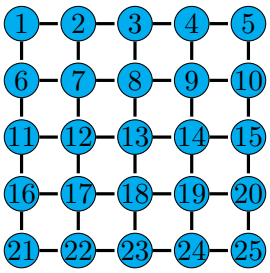

Abbildung 14

Vertauscht man nun die Steine auf Position 7 und Position 8, dann erhält man einen gelösten M_5, der eine gültige Instanz des Problems MagicCubeSOL darstellt, jedoch keine für $\mathsf{MagicCubeSOL}_{\mathsf{mech}}$.

Beispiel 3.2. $\mathsf{MagicCubeSOL} \subsetneq \mathsf{MagicCubeColorability}$:
Sei der Graph in Abbildung 13 ein gelöster M_3, der eine gültige Instanz für beide Probleme ist. Dann ist der Graph in Abbildung 14 (bestehen aus den gleichen Steinen) keine gültige Instanz für MagicCubeSOL, da es unter anderem keinen Eckstein mit den Farben weiß, gelb und rot im Ursprungsgraphen gibt.

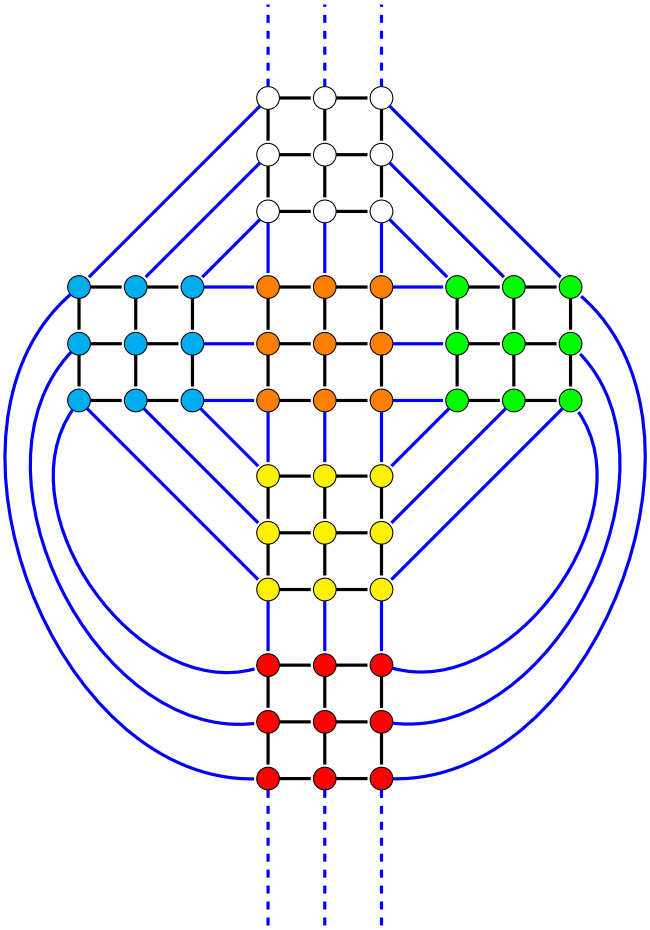

Abbildung 15

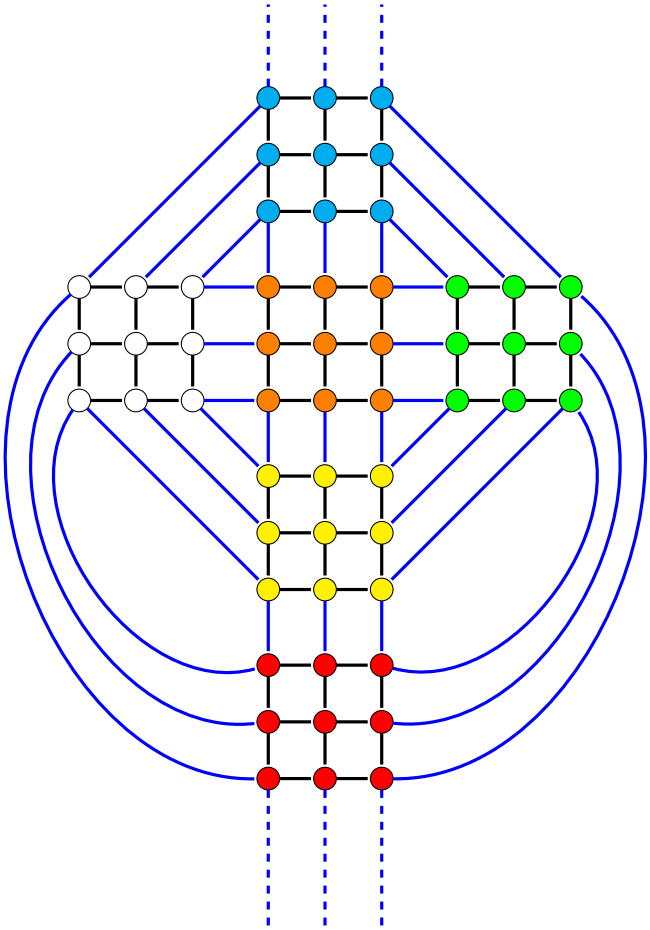

Abbildung 16

Da nun die Mengenbeziehungen zwischen den Problemen klar sind, werden wir uns mit den bereits bekannten Erkenntnissen beschäftigen und untersuchen, inwiefern wir diese verwenden können.

3.2 Gods Number

Der Begriff *Gods Algorithm* bezeichnet für kombinatorische Puzzle und mathematische Spiele den Algorithmus, den ein höheres Wesen verwenden würde, um das jeweilige Spiel perfekt zu lösen. *Gods Number* ist für den Magic-Cube die maximale Anzahl an Bewegungen, die man brauchen würde, wenn man Gods Algorithm verwendet.

Definition 3.3. Sei M_n ein beliebiger Magic-Cube und F die Menge aller Färbungen, für die es eine Lösung gibt. x_f^* ist die Menge aller Bewegungsabfolgen, die $\langle M_n, f\rangle$, mit $f \in F$ in den gelösten Zustand bringen. Für *Gods Number* G_{qtm} gilt dann:

$$G_{qtm} = \max\{\min\{|x| \mid x \in x_f^*\} \mid f \in F\}$$

Satz 3.4. *Für den M_3 gilt: Gods Number $G_{qtm} \leq 29$*

In der Einleitung wurde erwähnt, dass Gods Number für den M_3 gleich 20 sei, es ist dabei festzuhalten, dass diese Zahl aufgrund der „half-turn metric“ (auch „face-turn metric“ genannt) zustande kommt. Das bedeutet, dass eine doppelte Bewegung einer Seite genauso gewertet wird, wie eine einfache. Dies kann in [1] nachgelesen werden. Wir werden allerdings später mit der genaueren „quarter-turn metric“ arbeiten. Für unsere weitere Betrachtung ist es allerdings unwichtig, welchen Wert die obere Schranke hat, solange sie konstant ist. Wir könnten also auch von dem Wert 40 ausgehen, das würde heißen, wir nehmen an, dass jede Bewegung doppelt ausgeführt wird. Es wurde jedoch bereits gezeigt, dass Gods Number G nach der „quarter-turn metric“ maximal 29 ist [2].

Satz 3.5. *Für den M_2 gilt: Gods Number $G_{qtm} = 14$*

Es handelt sich erneut um die maximale Zahl der benötigten Bewegungen nach der „quarter-turn metric“. Legt man die „half-turn metric“ zugrunde, dann erhält man für Gods Number den Wert 11. Beides kann in [3] nachgelesen werden.

3.3 Bekannte Lösungsmethoden

Naheliegend wäre es, einfach die vorhandenen Methoden zum Lösen eines Magic-Cube zu verwenden, um eine Lösung zu finden. Wenn man mit einer dieser Methoden eine Lösung findet, dann ist der Magic-Cube lösbar, wenn nicht, dann ist er unlösbar. Wir werden uns in diesem Unterkapitel mit einer weit verbreiteten Methode beschäftigen.

Bei dieser Methode löst man zuerst die Mittelsteine, um anschließend die Kantensteine so zu platzieren, dass gleich gefärbte Kantensteine nebeneinander liegen. Anschließend löst man den Würfel mit Hilfe der Fridrich-Methode [5] wie einen M_3 und interpretiert dabei die gelösten Mitten jeweils als einen einzigen Mittelstein und die gleichfarbigen Kanten als einen Kantenstein. Beim Lösen der Mittelsteine benötigt man für jeden Stein eine konstante Anzahl an Bewegungen, die Anzahl der Steine hängt jedoch von der Größe des Würfels ab, es gibt $6 \cdot (n-2)^2$ verschiedene. Gleiches gilt für die Platzierung der Kantensteine, auch hier benötigt man für jeden einzelnen Stein eine konstante Anzahl an Bewegungen und es gibt $12 \cdot (n-2)$ verschiedene Steine. Für verschiedene Kantenlängen gibt es außerdem Sonderfälle, die eintreten können. Man muss sie beseitigen, bevor man den M_3 löst. Jeden dieser Sonderfälle kann man mit Hilfe einer Bewegungsabfolge konstanter Länge beseitigen. Das Lösen des M_3 lässt sich ebenfalls durch eine Konstante nach oben beschränken. Insgesamt hat man also einen Algorithmus, der in Polynomialzeit läuft und eine Lösung zu einer Instanz berechnet.

Wir haben gesehen, dass es einen Polynomialzeitalgorithmus gibt, der einen beliebigen M_n löst. Diesen können wir verwenden, um auch unsere Entscheidungsprobleme zu lösen. Dazu lassen wir – wie oben bereits erwähnt – einfach den Algorithmus laufen. Findet dieser eine Lösung, dann akzeptieren wir, findet er keine, so lehnen wir ab. Daraus folgt, dass die Entscheidungsprobleme maximal in der Komplexitätsklasse P liegen können. Das bedeutet, es besteht die Möglichkeit, dass die Entscheidungsprobleme in NL oder einer niedrigeren Klasse liegen. Dies werden wir im Rest dieses Kapitels untersuchen.

3.4 MagicCubeSOL$_{\text{mech}}$

Wir beschäftigen uns zuerst mit MagicCubeSOL$_{\text{mech}}$. Dafür benötigen wir eine eindeutige Benennung der verschiedenen Bewegungen. Diese werden wir auch in den weiteren Kapiteln verwenden.

In Abbildung 17 sind alle Bewegungskreise eines M_n markiert. Im Folgenden seien die Bewegungen o.B.d.A wie folgt gewählt:

	in Pfeilrichtung			gegen Pfeilrichtung		
	rot	grün	violett	rot	grün	violett
Hell	1	$n+1$	$2 \cdot n+1$	$1'$	$[n+1]'$	$[2 \cdot n+1]'$
...	...	...	...	...	...	...
Dunkel	n	$2 \cdot n$	$3 \cdot n$	n'	$[2 \cdot n]'$	$[3 \cdot n]'$

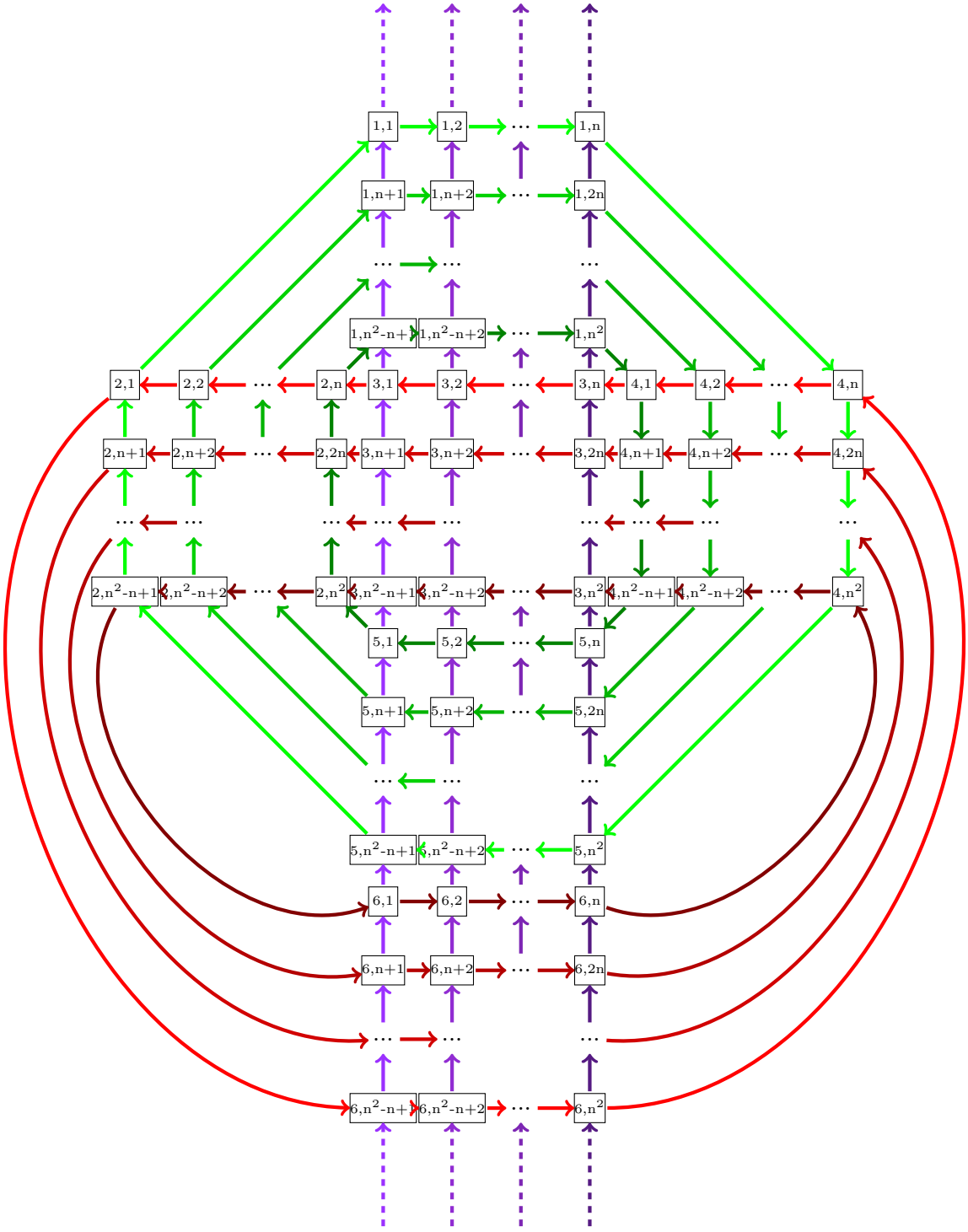

Abbildung 17: Der Knoten $v_{i,j}$ ist mit i, j beschriftet. Aus Übersichtsgründen wurden die Bewegungskreise der Oberflächen nicht markiert.

Damit wir einen Algorithmus angeben können, der unser Problem löst, müssen wir uns zunächst klar machen, was dieser überprüfen muss. Es gibt drei verschiedene Möglichkeiten eine unlösbare Konfiguration zu erzeugen.

Beobachtung.
Ausgehend von einem bereits gelösten Magic-Cube$_{\mathsf{mech}}$ führen folgende Veränderungen zu einer unlösbaren Konfiguration:

(i) Drehen eines Kantensteins.
(ii) Drehen eines Ecksteins.
(iii) Tauschen zweier Ecksteine.

Es gibt zu jedem dieser Fälle äquivalente unlösbare Konfigurationen, d.h. man kann diese Konfigurationen durch Bewegungen in einen der oben genannten Fälle überführen.

Dreht man – bei einem gelösten Magic-Cube – eine ungerade Anzahl an Kantensteinen, so erhält man die zu (i) äquivalenten Konfigurationen. Führt man für jeweils zwei „gegenüberliegende“ verdrehte Steine auf den Positionen der Menge P_{k_j} $(P_{k_j} \neq P'_k)$ die folgende Bewegungsabfolge aus, so erhält man Fall (i):

$$f' = ((f(v), \quad ([3n-j]^2, [n+1]^2, [1]^2, [2n+1+j]', [1]^2, [3n-j]', [1]^2, 3n-j, \\ [1]^2, [2n]^2, 3n-j, [2n]^2, 2n+1+j, [n+1]^2, [3n-j]^2)$$

Annahme: Die beiden Steine befinden sich auf den Positionen $((1, n^2-n+1+j), (3, 1+j))$ und $((1, n^2-j), (3, n-j))$.

Für die Positionemenge P'_k muss man folgende Bewegungen durchführen, um das gleiche Ziel zu erreichen:

$$f' = ((f(v), ([2n+1+\frac{n}{2}, 1]^3, 1, [[2n+1+\frac{n}{2}]', 1]^3, 1)$$

Annahme: Die beiden Steine befinden sich auf den Positionen $((1, n^2-\frac{n}{2}+1), (3, 1+\frac{n}{2})))$ und $((6, n^2-\frac{n}{2}+1), (1, 1+\frac{n}{2}))$.

Macht man dies bei einer geraden Anzahl an verdrehten Steinen, dann erhält man den gelösten Magic-Cube.

Das gleiche gilt für das Drehen der Ecksteine, wobei es für einen Eckstein immer zwei falsche Drehungen gibt. Mit folgender Bewegungsabfolge dreht man zwei „gegenüberliegende“ Ecksteine im Uhrzeigersinn:

$$f' = ((f(v), ([3n, 1, [3n]', 1']^2, 2n+1, [1, 3n, 1', [3n]']^2, [2n+1]'))$$

Annahme: Die beiden Steine befinden sich auf den Positionen $((1,1), (2,1), (6, n^2-n+1))$ und $((1, n^2-n+1), (2, n), (3, 1))$.

Die Mittelsteine sind paarweise innerhalb ihrer Untermengen P_{m_k} $(|P_{m_k}| > 6)$ beliebig vertauschbar mit folgender Bewegungsabfolge:

$$f' = ((f(v), ([1]^2, 2n-1, [1+j]', [2n-1]', [1]^2, 2n-1, 1+j, [2n-1]', [1]^2)$$

Annahme: Die beiden Steine befinden sich auf den Positionen $((1, n^2-2n+1+j))$ und $((3, j \cdot n+2))$.

Daher müssen sie nicht weiter überprüft werden. Eine Ausnahme stellt die Menge P'_m dar, die es nur bei Magic-Cubes mit ungeraden Kantenlängen gibt. Die Positionen dieser Menge sind nach Definition beim mechanischen Magic-Cube jedoch fest und müssen daher auch nicht gesondert geprüft werden.

Vertauscht man – ausgehend von einem gelösten Würfel – zwei komplette „Zeilen“ von Kantensteinen, so erhält man eine zu (iii) äquivalente Konfiguration. Mit „Zeile“ sind dabei alle Kantensteine gemeint die sich auf dem direkten Weg zwischen zwei bestimmten Ecksteinen befinden. Man kann mit folgender Bewegungsabfolge zwischen den äquivalenten Konfigurationen wechseln:

$$f' = ((f(v), (3n, 1, [3n]', 1', [3n]', 2n, [3n]^2, 1', [3n]', 1', 1, [3n]', [2n]'))$$

Annahme: Die erste „Zeile“ befindet sich auf den Positionen $(((1, n+1), (2, 2)), ((1, 2 \cdot n + 1), (2, 3)), ..., ((1, n^2 - 2n + 1), (2, n - 1))$, die zweite auf den Positionen $(((1, 2 \cdot n), (4, n - 1)), ((1, 3 \cdot n), (4, n - 2)), ..., ((1, n^2 - n), (4, 2))$.

Da die oben genannten Bewegungsabfolgen immer nur genau die Steine auf den jeweilig Positionen verändern und sonst alle anderen Knoten gleich bleiben, lassen sich die oben angenommen Positionen durch leichte Bewegungen erreichen. Wichtig ist bei diesen Bewegungen nur, dass die Steine, die bereits an der richtigen Position sind, nicht bewegt werden. Nach der jeweiligen Bewegungsabfolge müssen diese einfachen Bewegungen nur noch rückgängig gemacht werden.

Zur Vereinfachung der Algorithmen werden wir im Folgenden eine Fallunterscheidung zwischen geraden und ungeraden Kantenlängen machen. Zunächst untersuchen wir also das Problem $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$.

3.4.1 Gerade Kantenlängen

Für einen mechanischen Magic-Cube mit gerader Kantenlänge müssen wir lediglich Fall (ii) überprüfen. Nach Definition gilt für Magic-Cubes mit gerader Kantenlänge für alle i, dass $|P_{k_i}| = 24$. Für den mechanischen Magic-Cube gilt des Weiteren, das Steine auf den Positionen der Menge P_{k_i} nur innerhalb dieser Menge vertauscht werden können und das jeweils nur mit einer Drehung. Möchte man also einen Stein drehen, muss er an eine andere Position der selben Positionenmenge gebracht werden. Der Stein an dieser Position wiederum muss an die Position des ersten Steins – ebenfalls mit einer Drehung – eingesetzt werden. Es ist also nicht möglich einen Stein einzeln zu drehen. Fall (i) kann also nicht eintreten und muss daher auch nicht überprüft werden.

Die folgende Bewegungsabfolge tauscht die Steine an den Positionen $((1, n^2 - n + 1 + j), (3, j + 1))$ und $((1, n^2 - j), (3, n - j))$ mit den Steinen an den Positionen $((6, n^2 - n + 1 + j), (1, j + 1))$ und $((6, n^2 - j), (1, n - j)))$:

$$f' = ((f(v), ([[3n - j]^2, [1]^2]^2, [2]^2, [3n - j]^2, [2]^2))$$

Führt man diese Bewegungsabfolge für alle j aus, dann beseitigt man damit Fall (iii).

Um die Lösbarkeit eines $\text{Magic-Cube}_{\mathsf{mech}}$ zu ermitteln, müssen wir also nur überprüfen, ob die Eckstein lösbar sind. Dazu können wir alle anderen Steine „ausblenden“ und den M_n wie einen M_2, also einen Magic-Cube mit Kantenlänge 2, behandeln.

Folgender Algorithmus berechnet den zu M_n gehörenden M_2:

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 for Position p ∈ P_e do
 2 |   for v ∈ p do
 3 |   |   V' ← v
 4 |   |   f'(v) ← f(v)
 5 |   |   E' ← (v, u), mit u ∈ p
 6 |   end
 7 end
 8 for v_(a,b) ∈ V' do
 9 |   for u_(c,d) ∈ V' do
10 |   |   if a = c and (c + n − 1 = d or c + n^2 − n = d) then
11 |   |   |   E' ← (v_(a,b), v_(c,d))
12 |   |   end
13 |   end
14 end
   Ausgabe : ⟨M_2^p(K) = (V', E'), f'⟩
```

Algorithmus 2 : Umwandlung von M_n zu M_2

Der Algorithmus arbeitet mit Positionen, die in der Graphendarstellung zwar definiert aber nicht kodiert sind. Diese können jedoch vorher berechnet und zwischengespeichert werden. Die Berechnung von Positionenmengen wird in folgenden Algorithmen häufiger vorkommen. Es werden dabei immer nur eine konstante Anzahl von Positionenmengen gleichzeitig zwischengespeichert.

In Zeile 1 bis 7 erzeugt der Algorithmus für jede Eckposition drei Knoten, verbindet diese und übernimmt dessen Färbung. Von Zeile 8 bis 14 werden die Kanten innerhalb der Oberflächen erzeugt.

Platzbedarf: Es wird immer die Graphendarstellung eins Magic-Cube mit Kantenlänge 2 berechnet. Da aber der Platzbedarf der Knoten logarithmisch von der Eingabelänge abhängt, benötigt der Algorithmus logarithmisch viel Speicher.

Folgender Algorithmus entscheidet dann $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 if der zu M_n^z(K) gehörige M_2 ist unter Färbung f lösbar then
 2 |   return true
 3 else
 4 |   return false
 5 end
```

Algorithmus 3 : $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$

Es wird nur überprüft, ob der zu M_n gehörende M_2 lösbar ist. Hierzu müssen nach Satz 3.5 alle Bewegungsabfolgen der Länge 14 für den M_2 durchprobiert werden. Dazu kann die Färbung f' auf ein separates Band kopiert werden und nach den 14 Bewegungen

wieder überschrieben werden.

Platzbedarf: Es wird zunächst der zu $M_n^z(K), f$ gehörige M_2 berechnet werden, dies benötigt logarithmischen Platz (siehe Algorithmus 2). Die Überprüfung der Lösbarkeit des M_2 benötigt den Platz von f' und den eines Zählers, der immer die nächste zu überprüfende Bewegungsabfolge angibt. Der Zähler besteht aus 14 einzelnen Zählern, die alle bis maximal zwölf zählen müssen (es gibt für den M_2 zwölf verschiedene Bewegungen). Der Platzbedarf der Zähler ist also konstant. Insgesamt wird also logarithmisch viel Speicher benötigt.

3.4.2 Ungerade Kantenlängen

Für den mechanischen Magic-Cube mit ungerader Kantenlänge müssen wir alle drei oben genannten Fälle überprüfen. Allerdings tritt Fall (i) nur für die Positionenmenge P_k' auf (siehe oben).

Zur Feststellung, ob ein mechanischer Magic-Cube lösbar ist, müssen wir also die Ecksteine und die Positionenmenge P_k' überprüfen. Dazu müssen wir lediglich die Lösbarkeit des zu M_n gehörenden M_3 überprüfen.

Folgender Algorithmus berechnet den zu M_n gehörenden M_3:

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 for Position p ∈ P_e ∪ P_k' ∪ P_m' do
 2 |   for v ∈ p do
 3 |   |   V' ← v
 4 |   |   f'(v) ← f(v)
 5 |   |   E' ← (v, u), mit u ∈ p
 6 |   end
 7 end
 8 for v_(a,b) ∈ V' do
 9 |   for u_(c,d) ∈ V' do
10 |   |   if a = c and (c + (n+1)/2 = d or c + (n²+n)/2 = d) then
11 |   |   |   E' ← (v_(a,b), v_(c,d))
12 |   |   end
13 |   end
14 end
   Ausgabe : ⟨M_3^p(K) = (V', E'), f'⟩
```

Algorithmus 4 : Umwandlung von M_n zu M_3

Es werden für alle Positionen, die auch in einem M_3 enthalten sind, Knoten entsprechend der Elemente der jeweiligen Positionen erzeugt. Die so erzeugten Knoten einer Position werden durch Kanten verbunden und entsprechend f gefärbt (das alles passiert von Zeile 1 bis 7). In den restlichen Zeilen werden noch die fehlenden Kanten innerhalb der Seiten erzeugt.

Platzbedarf: Das Berechnen der drei Positionenmengen P_e, P'_k, P'_m erfordert logarithmischen Speicher. Die Ausgabe benötigt ebenfalls logarithmischen Platz, da der Speicherbedarf der Knoten von der Eingabe abhängt.

Folgender Algorithmus entscheidet dann $\mathsf{OddMagicCubeSOL}_{\mathsf{mech}}$

```
   Eingabe : ⟨M_n^z(K), f⟩
1  if der zu M_n^z(K) gehörige M_3 ist unter Färbung f Lösbar then
2  |  return true
3  else
4  |  return false
5  end
```

Algorithmus 5 : $\mathsf{OddMagicCubeSOL}_{\mathsf{mech}}$

Der Algorithmus überprüft, ob der zu M_n gehörende M_3 lösbar ist. Da Gods Number für den M_3 kleiner gleich 29 ist, müssen also alle Bewegungsabfolgen der Länge 29 ausprobiert werden. Dies ist wieder durch einen Zähler zu realisieren.

Platzbedarf: Der Platzbedarf verhält sich – wie bei Algorithmus für $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$ – logarithmisch zur Eingabe, die Argumentation kann im Grunde so übernommen werden. Es müssen nur die Zahlen angepasst werden. Nach Satz 3.4 benötigt man 29 Zähler. Diese Zähler müssen maximal bis 18 zählen.

Um das Problem $\mathsf{MagicCubeSOL}_{\mathsf{mech}}$ zu entscheiden, benötigten wir noch einen einfachen Algorithmus, der abhängig von der Kantenlänge den richtigen Algorithmus ausführt:

```
   Eingabe : ⟨M_n^z(K), f⟩
1  if n mod 2 = 0 then
2  |  return EvenMagicCubeSol_mech
3  else
4  |  return OddMagicCubeSol_mech
5  end
```

Algorithmus 6 : $\mathsf{MagicCubeSOL}_{\mathsf{mech}}$

Dass dieser letzte Algorithmus ebenfalls logarithmisch viel Platz benötigt, liegt auf der Hand. Wir haben also einen Algorithmus angegeben, der $\mathsf{MagicCubeSOL}_{\mathsf{mech}}$ in Platz $O(log(n))$ löst. Damit haben wir gezeigt, dass das Problem in der Komplexitätsklasse L liegt.

3.5 MagicCubeSOL

Um die Lösbarkeit des magnetischen Magic-Cube zu überprüfen, müssen zunächst die gleichen drei Fälle wie zuvor geprüft werden. Es kommen jedoch noch drei neue Fälle hinzu.

Beobachtung.
Ausgehend von einem bereits gelösten Magic-Cube führen folgende Veränderungen zu einer unlösbaren Konfiguration:

(iv) Vertauschen zweier verschieden gefärbter Mittelsteine mit Positionen aus zwei verschiedenen Mengen P_{m_i}, P_{m_j}.

(v) Vertauschen zweier verschieden gefärbter Kantensteine mit Positionen aus zwei verschiedenen Mengen P_{k_i}, P_{k_j}.

(vi) Vertauschen zweier Mittelsteine mit Positionen aus der Menge P'_m.

Wie auch bei den ersten drei Fällen gibt es zu den Fällen $(iv) - (vi)$ äquivalente Konfigurationen. Damit ein Magic-Cube nicht durch eine zu (iv) äquivalente Konfiguration unlösbar wird, muss gewährleistet sein, dass in jeder Positionenmenge für Mittelsteine von jeder Farbe gleich viele Elemente enthalten sind. Das bedeutet, dass durch den Tausch mehrerer Steine wieder eine lösbare Konfiguration erreicht werden kann. Gleiches gilt für (v), auch hier wird der Magic-Cube unlösbar, wenn eine Positionenmenge von einer Färbung mehr Elemente besitzt als von einer anderen. Der Fall (vi) tritt nur bei Magic-Cubes mit ungeraden Kantenlängen auf, da nur bei solchen Magic-Cubes die Menge P'_m existiert. Das passiert, wenn die relativen Positionen der Steine aus P'_m nicht zu den Färbungen der Ecksteine passen. Die tatsächlichen Positionen der Steine aus P'_m lassen sich durch Bewegungen verändern, allerdings werden sie immer die gleiche Ausrichtung zueinander haben (z.B. ist blau immer „gegenüber“ von grün). Aus den Ecken lassen sich auch solche relativen Positionen für bestimmte Farbwerte berechnen. Gibt es zum Beispiel zwei Steine aus P_e mit den Färbungen (rot, gelb, blau), (rot, gelb, grün), dann muss in der gelösten Konfiguration – wenn sie denn existiert – die grün gefärbte Oberfläche „gegenüber“ der blauen sein.

Der folgende Algorithmus berechnet die relativen Positionen der Farben für Magic-Cubes mit Kantenlänge zwei:

Eingabe : $\langle M_2^z(K), f \rangle$
1 Wähle eine Position $p_1 \in P_e$
2 $\forall v_{i,j} \in p_1 \colon color_{O_i} \leftarrow f(v_{i,j})$
3 Wähle eine Position $p_2 \in P_e$, wobei es genau zwei Knotenpaare u, v gibt mit der Eigenschaft, dass $f(u) = f(v)$, mit $u \in p_1$ und $v \in p_2$
4 $color_{O_i} \leftarrow f(v)$, mit $v \in p_2$, $f(v) \neq f(u)$, $\forall u \in p_1$ und $\{(x, y) \mid x \in O_i, y \in O_j, \} = \emptyset$, wobei $\exists w \colon w \in p_1 \colon f(w) = color_{O_j}, f(w) \neq f(z)$, $\forall z \in p_2$
5 Wähle eine Position $p_3 \in P_e$ $(p_3 \neq p_2)$, wobei es genau zwei Knotenpaare u, v gibt mit der Eigenschaft, dass $f(u) = f(v)$, mit $u \in p_1$ und $v \in p_3$
6 $color_{O_i} \leftarrow f(v)$, mit $v \in p_3$, $f(v) \neq f(u)$, $\forall u \in p_1$ und $\{(x, y) \mid x \in O_i, y \in O_j, \} = \emptyset$, wobei $\exists w \colon w \in p_1 \colon f(w) = color_{O_j}, f(w) \neq f(z)$, $\forall z \in p_3$
7 Weise der Oberfläche ohne Farbwert den letzten verbleibenden Farbwert zu.

Algorithmus 7 : Bestimmen der Farben der Oberflächen (gerade)

Der Algorithmus erwartet als Eingabe einen M_2, da nur die Elemente aus P_e für die Berechnung notwendig sind. Wir können ihn doch für jeden M_n verwenden, indem wir vorher mit Algorithmus 2 den zu M_n gehörenden M_2 berechnen.

Der Algorithmus weist jeder Oberfläche einen Farbwert zu. Dazu wird eine beliebige Eckposition gewählt und jeder Oberfläche der Position der Farbwert des beteiligten Knotens zugewiesen (Zeile $1 - 2$). Danach wird eine Position gewählt, die genau zwei gleiche Farbwerte wie die erste Position besitzt. Die beiden Werte, die sich unterscheiden, müssen zu „gegenüberliegenden“ Oberflächen gehören. Dieser Schritt wird noch einmal für eine weitere Position wiederholt, um den Farbwert der fünften Oberfläche zu ermitteln. Zuletzt wird der sechsten Oberfläche der verbleibende Farbwert zugewiesen.

Die so zugewiesenen Farbwerte müssen nicht mit den Farben der Oberflächen im gelösten Zustand übereinstimmen. Es ist jedoch an dieser Stelle nur die Ausrichtung der Farbwerte der Oberflächen interessant.

Platzbedarf: Der Algorithmus speichert für jede der sechs Oberflächen einen Farbwert. Des Weiteren muss die Positionenmenge P_e zwischengespeichert werden, was ebenfalls konstanten Platz benötigt. Der Platzbedarf ist daher konstant.

Für das Bestimmen der Oberflächenfarbwerte im Magic-Cube mit Kantenlänge drei könnte man den gleichen Algorithmus verwenden wie oben. Es geht allerdings auch wesentlich einfacher und präziser über die Menge P'_m.

Eingabe : $\langle M_3^z(K), f \rangle$
1 **for** $v_{i,j} \in P'_m$ **do**
2 $\quad color_{O_i} \leftarrow f(v_{i,j})$
3 **end**

Algorithmus 8 : Bestimmen der Farben der Oberflächen (ungerade)

Jeder Oberfläche wird der Farbwert des Knotens der Position $p \in P_{m_i}$ zugewiesen. Diese Farbwerte entsprechen dann auch denen im gelösten Zustand, wenn er denn existiert.

Platzbedarf: Wie bei Algorithmus 8 wird für alle sechs Oberflächen jeweils ein Farbwert gespeichert. Zudem muss die Positionenmenge P'_m zwischengespeichert werden. Der Speicherbedarf ist also konstant.

Bei dem mechanischen Magic-Cube konnte wegen der Einschränkungen Fall (i) nur für die Menge P'_k eintreten. Für den allgemeinen Magic-Cube müssen wir nun für alle Positionenmengen prüfen, ob es eine ungerade Anzahl an falsch gedrehten Steinen gibt. Dazu werden wir einen Algorithmus angeben, der in ähnlicher Form auch beim „Blindlösen“ eines Magic-Cube verwendet wird. Beim „Blindlösen“ geht es darum, sich den Magic-Cube einmal einzuprägen, danach die Augen zu schließen und ihn dann mit geschlossenen Augen zu lösen. Dabei merkt man sich natürlich nicht den ganzen Würfel zu jedem Zeitpunkt, sondern verwendet Bewegungsalgorithmen, die möglichst immer nur zwei Steine verändern. Man wählt also einen Stein und sucht dessen korrekte Position. Daraufhin sucht man für den Stein, der sich an der korrekten Position des ersten befindet wiederum eine korrekte Position. Diese Methode führt zum Ziel, wenn man es mit einem lösbaren Magic-Cube zu tun hat. Wir machen uns dieses Vorgehen zu Nutze und geben einen Algorithmus an, der überprüft, ob die verschiedenen Kantensteine verdreht sind oder nicht.

Der folgende Algorithmus überprüft Fall (i):

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 for P_{k_i} ∈ P_k do
 2     while ∃p ∈ P_{k_i} : done[p] = false do
 3         Wähle eine Position p ∈ P_{k_i}, mit done[p] = false
 4         Wähle einen Knoten v_{i,j} ∈ p, temp ← v_{i,j},temp' ← v_{i,j}, y ← p
 5         while done[y] = false do
 6             Wähle eine Position q ∈ P_{k_i}, mit done[q] = false und
               ∀v ∈ p: ∃l: color_{O_l} = f(v), mit |q ∩ O_l| = 1
 7             temp' ← u, mit u ∈ q, u ∈ O_x und color_{O_x} = f(temp')
 8             p ← q
 9             done[q] = true
10         end
11         if temp ≠ temp' then
12             cnt ← cnt + 1
13         end
14     end
15     if cnt mod 2 = 1 then
16         return false
17     end
18 end
19 return true
```

Algorithmus 9 : Kantenzykel

Der Algorithmus prüft für alle Kantenmengen P_{k_i}, ob es eine ungerade Anzahl an falsch gedrehten Steinen in dieser Menge gibt. Dazu wird für jede Position aus $p \in P_{k_i}$ ein boolean ($done[]$) gespeichert, der angibt, ob für die jeweilige Position ein passender Stein ermittelt wurde. Es wird nun zufällig eine Position $p \in P_{k_i}$ gewählt und in y zwischengespeichert. Weiterhin wird ein Knoten dieser Position gewählt und in $temp$ und $temp'$ gespeichert. In der Schleife, die in Zeile 6 beginnt, wird nun für die Farben unserer aktuellen Position, die korrekte Position q ermittelt über die Farbwerte der Oberflächen (siehe Algorithmus 7 bzw. 8) . Der Wert $done[q]$ wird auf wahr gesetzt, da der richtige Stein für die Position q gefunden wurde. Nun wird die in p gespeicherte Position mit q überschrieben und der in $temp'$ gespeicherte Wert wird durch den Knoten überschrieben, der die Farbe des vorher in $temp'$ gespeicherten Wertes erhalten soll. Die Schleife wird so lange ausgeführt, bis wir wieder bei der zufällig gewählten Position ankommen. Nun wird überprüft, ob die Werte $temp$ und $temp'$ übereinstimmen. Wenn das zutrifft, dann könnte man diese Steine alle auf die jeweiligen Positionen bringen und sie wären richtig gedreht. Wenn es nicht zutrifft, dann wird der Stein an Position z falsch gedreht sein. Ist letzteres der Fall, so erhöhen wir einen Zähler, der angibt, wie viele falsch gedrehte Steine wir bereits haben. Gibt es keine Position mehr, die wir zufällig wählen können, weil alle $done[]$ Werte auf wahr gesetzt wurden, so überprüfen wir den Zähler. Ist dieser gerade, so können wir alle falsch gedrehten Steine durch Bewegungen richtig drehen. Ist

das hingegen nicht der Fall, so wird immer mindestens ein Stein falsch gedreht sein und der Würfel ist somit unlösbar.

Platzbedarf: Es werden vier temporäre Variablen verwendet und für jede Position $p \in P_{k_i}$ eine boolesche Variable. Diese Variablen können in jedem Schleifendurchlauf überschrieben werden. Gleiches gilt für die verschiedenen Mengen P_{k_i}. Die verschiedenen Mengen P_{k_i} enthalten maximal 24 Elemente. Somit ist die Anzahl der Variablen beschränkt auf insgesamt 28. Da der Speicherbedarf der Knoten logarithmisch abhängig von der Eingabe ist, erhalten wir einen Platzbedarf von $O(log(n))$.

Folgender Algorithmus überprüft, ob Fall (iv) eintritt:

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 for i ← 1 to 6 do
 2 |  cnt_i ← 0
 3 end
 4 for P_{m_i} ⊆ P_m do
 5 |  for Positionen p ∈ P_{m_i} do
 6 |  |  ∀v ∈ p: cnt_{f(v)} ← cnt_{f(v)} + 1
 7 |  end
 8 |  for i ← 2 to 6 do
 9 |  |  if cnt_i ≠ cnt_1 then
10 |  |  |  return false
11 |  |  end
12 |  |  cnt_i ← 0
13 |  end
14 |  cnt_1 ← 0
15 end
```

Algorithmus 10 : Überprüfung der Mittelpositionen

In Zeile 1 bis 3 wird für die sechs verschiedenen Farben jeweils ein Zähler initialisiert. Danach wird für alle Positionenmengen überprüft, ob diese von jeder Farbe gleich viele Elemente haben. Ist das nicht der Fall, dann ist mindestens ein Zähler ungleich des Zählers für die erste Farbe. Es wird demnach in Zeile 10 abgelehnt.

Platzbedarf: Es werden sechs verschiedene Zähler gespeichert, die maximal bis 24 zählen. Zudem müssen die verschiedenen P_{m_i} zwischengespeichert werden, diese können jedoch überschrieben werden. Der Platzbedarf verhält sich also logarithmisch zur Eingabe.

Folgender Algorithmus überprüft, ob Fall (v) eintritt:

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 for i ← 1 to 6 do
 2 |   cnt_i ← 0
 3 end
 4 for P_{k_i} ⊆ P_k do
 5 |   for Positionen p ∈ P_{k_i} do
 6 |   |   ∀v ∈ p: cnt_{f(v)} ← cnt_{f(v)} + 1
 7 |   end
 8 |   for i ← 2 to 6 do
 9 |   |   if cnt_i ≠ cnt_1 then
10 |   |   |   return false
11 |   |   end
12 |   |   cnt_i ← 0
13 |   end
14 |   cnt_1 ← 0
15 end
```

Algorithmus 11 : Überprüfung der Kantenpositionen

Der Algorithmus funktioniert genau wie der für die Mittelpositionen. Der einzige Unterschied ist, dass in Zeile 6 pro Schleifendurchlauf immer genau zwei Zähler erhöht werden (bei den Mittelpositionen ist es immer ein Zähler), da die Positionen für Kantensteine immer genau zwei Knoten enthalten.

Platzbedarf: siehe Algorithmus 10.

3.5.1 Gerade Kantenlängen

Genau wie beim mechanischen Magic-Cube mit gerader Kantenlänge kann für den allgemeinen Magic-Cube Fall (iii) nicht eintreten und muss daher nicht überprüft werden. Des Weiteren kann Fall (vi) ebenfalls nicht eintreten, da die Positionenmenge P'_{m_i} für gerade Kantenlängen nicht existiert.

Folgender Algorithmus entscheidet dann $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$

```
   Eingabe : ⟨M_n^z(K), f⟩
 1 Überprüfe die Mittelpositionen (Algorithmus 10)
 2 Überprüfe die Kantenpositionen (Algorithmus 11)
 3 if EvenMagicCubeSOL_mech(⟨M_n^z(K), f⟩) then
 4 |   Bestimme die Farben der Oberflächen von (⟨M_2, f⟩) (Algorithmus 7)
 5 |   return Kantenzykel(⟨M_n^z(K), f⟩)
 6 else
 7 |   return false
 8 end
```

Algorithmus 12 : EvenMagicCubeSOL

Der Algorithmus führt letztendlich nur noch die vorherigen Algorithmen aus. Durch Zeile 1 und 2 wird sichergestellt, dass Fall (iv) und (v) nicht eintreten. Das **if** in Zeile 3 stellt wie beim mechanischen Magic-Cube fest, ob die Ecken lösbar sind (Fall (ii)). In Zeile 4 werden dann schließlich den Oberflächen ihre Farbwerte zugewiesen, die von dem Kantenzykelalgorithmus benötigt werden, um Fall (i) zu überprüfen.

Platzbedarf: Der Algorithmus benötigt die Summe des Platzbedarfs der Algorithmen 3,7,9,10 und 11. Es wird also $O(log(n))$ Speicher benötigt.

3.5.2 Ungerade Kantenlängen

Folgender Algorithmus entscheidet dann $\mathsf{EvenMagicCubeSOL}_{\mathsf{mech}}$

```
   Eingabe : ⟨M_n^z(K), f⟩
1  Überprüfe die Mittelpositionen (Algorithmus 10)
2  Überprüfe die Kantenpositionen (Algorithmus 11)
3  if OddMagicCubeSOL_mech(⟨M_n^z(K), f⟩) then
4  |  Bestimme die Farben der Oberflächen von (⟨M_3, f⟩) (Algorithmus 8)
5  |  return Kantenzykel(⟨M_n^z(K), f⟩)
6  else
7  |  return false
8  end
```

Algorithmus 13 : OddMagicCubeSOL

Genau wie der Algorithmus für gerade Kantenlängen führt dieser Algorithmus nur die vorherigen aus. Der Unterschied zwischen den beiden Algorithmen besteht in Zeile 3 und 4. Durch Zeile 3 wird zusätzlich sichergestellt, dass Fall (iii) und Fall (vi) nicht eintreffen.

Platzbedarf: Der Algorithmus benötigt die Summe des Platzbedarfs der Algorithmen 5,8,9,10 und 11. Der Speicherbedarf verhält sich also logarithmisch zur Eingabe.

Wie auch schon bei $\mathsf{MagicCubeSOL}_{\mathsf{mech}}$ benötigen wir zusätzlich einen Algorithmus, der entscheidet, welcher der beiden Algorithmen ausgeführt werden muss:

```
   Eingabe : ⟨M_n^z(K), f⟩
1  if n mod 2 = 0 then
2  |  return EvenMagicCubeSol
3  else
4  |  return OddMagicCubeSol
5  end
```

Algorithmus 14 : MagicCubeSOL

Es ist schnell zu sehen, dass der Platzbedarf sich durch diesen Algorithmus nicht ändert. Wir haben also auch für MagicCubeSOL einen Algorithmus angegeben, der logarithmischen Platz benötigt und somit gezeigt, dass MagicCubeSOL in der Komplexitätsklasse L liegt.

3.6 MagicCubeColorability

Abschließend beschäftigen wir uns noch mit dem Problem MagicCubeColorability. Im Prinzip sind sich dieses Problem und MagicCubeSol sehr ähnlich. Würde bei der Definition von MagicCubeSol die Eigenschaft, dass es eine lösbare Konfiguration für den Magic-Cube geben muss weglassen, dann würden die Probleme praktisch zusammenfallen.

Es stellt sich die Frage, welche unlösbaren Konfigurationen hinzu kommen. Im Grunde gibt es nur zwei verschiedene. Entweder hat eine Position mehrfach die gleiche Farbe oder enthält zwei Farben, die einander „gegenüberliegen“. Beides kann sowohl bei Eck- als auch bei Kantenpositionen auftreten. Bei Mittelposition ist dies nicht der Fall, da sie genau aus einem Knoten bestehen und somit auch nur mit einer Farbe gefärbt sein können.

Die Frage ist nun, ob wir unsere Algorithmen für MagicCubeColorability wiederverwenden können und inwiefern wir sie anpassen müssen. Tatsächlich liegt die Vermutung nahe, dass die Algorithmen für MagicCubeSol – mit ein paar kleinen Änderungen – auch MagicCubeColorability entscheiden. Die neuen unlösbaren Konfigurationen die durch falsche Färbungen von Eckpositionen entstehen, werden keine Probleme für unsere Algorithmen für MagicCubeSol darstellen. Die Eckpositionen werden durch das Lösen des M_2 (bzw. M_3) überprüft. Ist also eine von den Eckpositionen nicht adäquat gefärbt, so wird abgelehnt, da der M_2 (bzw. M_3) nicht im gelösten Zustand sein kann.

Nun bleiben noch die Kantenpositionen. Durch den Algorithmus 11 wird nicht überprüft, ob eine Position an und für sich illegal gefärbt ist. Das bedeutet, es wird durch den Algorithmus nicht festgestellt, ob die Knoten einer Position zweimal mit der gleichen Farbe oder mit „gegenüberliegenden“ Farben gefärbt sind. Allerdings findet der Algorithmus 9 in diesem Fall keine passende Position für den Stein. Der Algorithmus müsste also in dem Fall ablehnen. Es ist möglich, dass es noch weitere unlösbare Konfigurationen für MagicCubeColorability gibt, dies bedarf allerdings noch weiterer Untersuchung.

Diese Betrachtung funktioniert nur dann, wenn wir weiterhin von einer Färbung mit genau sechs Farben ausgehen. Färbungen mit weniger als sechs Farben müssten noch einmal separat geprüft werden.

4 Zusammenfassung und Ausblick

In den Unterkapiteln 3.4 und 3.5 wurde gezeigt, dass sowohl das Entscheidungsproblem der Lösbarkeit für den mechanischen als auch das für den magnetischen Magic-Cube in Platz $O(log(n))$ zu entscheiden sind. Zudem haben wir die Vermutung geäußert, dass dies ebenfalls für das Problem MagicCubeColorability gilt. Dies ist allerdings noch zu zeigen.

Das Modell des Magic-Cube könnte man noch um eine Matrixdarstellung erweitern. Dabei könnte man sich an der Graphendarstellung orientieren und für jede Oberfläche eine Matrix angeben – die Einträge würden dann den Farben entsprechen. Eine Tensordarstellung wäre ebenfalls denkbar. Man könnte jeden Stein als einen Eintrag abbilden, wobei man Vektoren für die Einträge nehmen würde.

Des Weiteren wäre zu untersuchen, wie sich der Magic-Cube in höheren Dimensionen verhält. Dafür müsste man aber das in Kapitel 2 gegebene Modell anpassen. Es ist anzunehmen, dass die Entscheidungsprobleme für den m-dimensionalen Magic-Cube schwieriger zu lösen sind, als die des dreidimensionalen Würfels. Man könnte unter anderem einen Algorithmus verwenden, der dem Algorithmus 11 sehr ähnlich ist. Da aber die Dimension der meisten Steine auch mit der Dimension des Würfels wächst, würden die Zähler höhere Werte erreichen. Der Platzbedarf des Algorithmus wäre demzufolge logarithmisch abhängig von der Dimension des Würfels. Es liegt also die Vermutung nahe, dass diese Entscheidungsprobleme in der Komplexitätsklasse L oder einer höheren Komplexitätsklasse liegen.

Literaturverzeichnis

[1] Tomas Rokicki, Herbert Kociemba, Morley Davidson und John Dethridge: *God's Number is 20*. `http://www.cube20.org`, Zugriff 20.09.2013

[2] Tomas Rokicki: *Twenty-Nine QTM Moves Suffice*. `http://cubezzz.dyndns.org/drupal/?q=node/view/143`, Zugriff 20.09.2013

[3] Jaap Scherphuis: *Mini Cube, the 2×2×2 Rubik's Cube*. `http://www.jaapsch.net/puzzles/cube2.htm`, Zugriff 20.09.2013

[4] Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Anna Lubiw, Andrew Winslow: *Algorithms for Solving Rubik's Cubes*. 19th Annual European Symposium on Algorithms, arXiv:1106.5736

[5] Jessica Fridrich: *System for solving Rubik's cube*. `http://www.ws.binghamton.edu/fridrich/system.html`, Zugriff 20.09.2013