Schriftliche Ausarbeitung zum Projektseminar Optimierung von OptiMates

Nicolas Knorr (NK)
Jakob Roth (JR)
Athanassios Bakas (AB)

8. August 2021

Inhaltsverzeichnis

1 Einleitung (NK)

Seit Jahrtausenden schon beschäftigen sich Menschen damit, den ihnen ureigenen Standpunkt im Universum zu erforschen. Darunter fallen nicht nur beispielsweise biologische oder gar religiöse Überlegungen, auch auf einer physikalischen Ebene möchte man in Erfahrung bringen, wo man sich, aus astronomischer Sicht, befindet und insbesondere, wovon man umgeben ist. Dies setzt auch die umfangreiche Betrachtung der erdnahen Himmelskugel und der Gestirne voraus. In dieser Hinsicht eröffnet die moderne Technik immer präzisere Möglichkeiten, um genau jenes Vorhaben voranzutreiben. Konnten die alten Ägypter schon mithilfe trigonometrischer Ansätze die Distanz zwischen Erde und Mond berechnen, gibt es heute modernste Teleskope, mithilfe derer es möglich ist, hochauflösende Aufnahmen des erdfernen Raumes zu machen.
So befindet sich derzeit im Vera C. Rubin Observatory ein Spiegelteleskop im Aufbau, das es erlaubt, Bilder des Himmels mit einem Bildwinkel von 3,5° aufzunehmen. Es ist ein vielerseits nachgefragtes Anliegen, mit diesem Teleskop die Himmelskugel, die die Erde umgibt, gänzlich zu erfassen, das heißt, den von der Erde aus sichtbaren Nachthimmel zu kartographieren. Auf diese Weise soll es möglich werden, sich ständig aktualisierende Sternenkarten zu konstruieren und etwaige Daten für Asteroidenüberwachungssysteme zu erhalten.
In diesem Artikel sollen Wege diskutiert werden, wie es mithilfe geschickter Modellierung und Programmierung erreicht werden kann, die erdumgebende Himmelskugel mit dem Teleskop des Vera C. Rubin Observatory vollständig und auf eine möglichst optimale Art und Weise abzubilden. Das dazugehörige Modell umfasst die Einheitssphäre S^2, die als Himmelskugel verstanden werden kann, und sphärische Kugelkappen mit einem Durchmesser von 3,5°, die so auf der Kugeloberfläche platziert werden sollen, dass eine optimale Pflasterung ohne Lücken entsteht. Dies bedeutet, dass die S^2 mit einer möglichst geringen Anzahl an Kappen vollständig überdeckt wird. Rolfes und Vallentin [1] konnten diesbezüglich Schranken für die Überdeckungszahl und auch für die Überdeckungsdichte finden.
Der hier gewählte Ansatz basiert im Allgemeinen darauf, dass ein Graph geeigneter Gestalt gewählt wird und dessen Knoten auf die Kugeloberfläche gelegt werden. Prinzipiell kommen dafür mehrere Graphen in Frage, die in der Folge ausführlicher beschrieben werden. Das Resultat ist eine Menge von Knoten auf der Kugeloberfläche, die potentielle Mittelpunkte für die Kugelkappen darstellen. Die Auswahl der entsprechenden Knoten unterliegt einem Greedy-Verfahren, das im Abschnitt zur Implementierung erläutert wird. Die Lösung des gewählten Ansatzes ist also, eine minimal bedeckende Knotenmenge für die Einheitssphäre zu finden, wobei jene Knotenmenge die Mittelpunkte der ausgewählten sphärischen Kappen beinhaltet.

2 Mathematische Modelle (NK)

Bezüglich der Modellierung kann man leicht einsehen, dass ein graphentheoretischer Ansatz nicht nur anschaulich ein leicht nachvollziehbares Modell liefert, sondern auch, dass damit vielversprechende Ergebnisse erwartet werden können. Das Ziel ist also, einen Graphen zu finden, dessen Knoten allesamt auf der Einheitssphäre liegen. Dieser Graph bildet die Basis des Modells, auf dem im Anschluss ein Algorithmus operieren wird. In der hier beschriebenen Methode wird von den Knoten des Graphen eine Teilmenge durch einem Greedy- Algorithmus nach dem „take-the-best“-Prinzip ausgewählt, deren Elemente die Mittelpunkte der Kugelkappen repräsentieren. Auf diese Weise soll schließlich eine vollständige Überdeckung der Kugeloberfläche mit möglichst wenig Überschneidungsfläche resultieren. Es werden in diesem Abschnitt drei grundsätzlich verschiedene Graphen beziehungsweise Knotenverteilungen vorgestellt. Darüber hinaus wird erörtert, was für und was gegen die einzelnen Verteilungen spricht.

2.1 Zufallsverteilung (NK)

Der intuitivste Ansatz ist vermutlich, die Knoten des Graphen nach dem Zufallsprinzip auf der S^2 zu verteilen. Schließlich benötigt man dazu nur die Parametrisierung der Einheitskugel in Polarkoordinaten

$$\begin{aligned} x &= \sin(\theta) \cdot \cos(\varphi) \\ y &= \sin(\theta) \cdot \sin(\varphi) \\ z &= \cos(\theta) \end{aligned}$$

und dann wählt man die Winkel $\theta \in [0, \pi]$ und $\varphi \in [0, 2\pi)$ zufällig aus.
Die nachfolgende Graphik, die dieses Vorgehen exemplarisch mit 100 000 Knoten veranschaulicht, zeigt, dass diese Knotenverteilung Ballungen an den Polen und nur einen lichten Knotenbestand im Bereich des Äquators mit sich zieht.

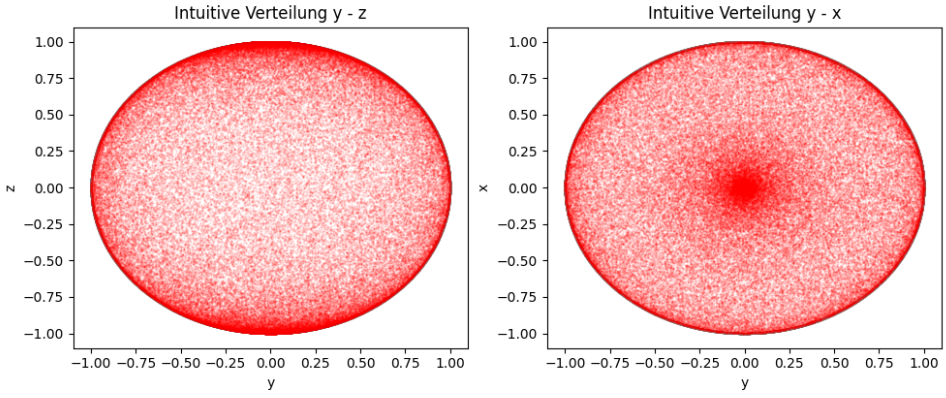


Abbildung 1: Zufallsverteilung, Seitenansicht (links), Draufsicht (rechts)

Um einen einwandfreien Ablauf des Greedy-Verfahrens zu gewährleisten, ist es ratsam, auf einer im

Mittel äquidistanten Zufallsverteilung zu operieren. Dazu ist es nötig, die Verteilung zu dämpfen. Dies erfolgt beispielsweise über die Einbindung des Arcuscosinus in das Intervall von θ:

$$\theta \in [\arccos(-1), \arccos(1)]$$
$$\varphi \in [0, 2\pi)$$

Das Ergebnis ist eine Zufallsverteilung, die einer Gleichverteilung ähnelt.
Diese Verteilung ist jedoch noch mit zwei weiteren Problemen behaftet, die sich nicht so einfach beheben lassen. Zum einen verstehen wir die im Modell verwendeten Kugelkappen als offene Mengen. Das bedeutet, dass sie ausschließlich aus inneren Punkten bestehen, was letztlich bei jedem (theoretischen) Schnittpunkt einer oder mehrerer (abgeschlossener) Kappen dazu führt, dass dieser unüberdeckt bleibt. In der Folge ist es nötig, eine erheblich größere Anzahl an Kappen zur Überdeckung der Schnittpunkte zu verwenden, als, wenn dieses Problem nicht bestünde. Zum anderen kann mit der Zufallsverteilung keine gültige, also vollständige, Überdeckung garantiert werden. Die Wahrscheinlichkeit dafür geht nur für eine sehr große Knotenanzahl gegen 1. Bei einem Versuch mit 40 000 Knoten konnte keine vollständige Überdeckung erzielt werden.
So simpel und intuitiv dieses Modell ist, tut man also gut daran, dem Graphen etwas mehr Struktur zu geben. Dies führt zum nächsten Modell.

2.2 Ikosaeder (NK)

Die Idee dieses Verfahrens besteht darin, die Einheitskugel von unten durch ein Ikosaeder anzunähern und zunächst auf diesem zu modellieren. Anschließend projiziert man die auf dem Ikosaeder bestimmten Knoten geeignet auf die Kugeloberfläche.
Das Ikosaeder ist einer der platonischen Körper mit 20 Seitenflächen und 12 Ecken.

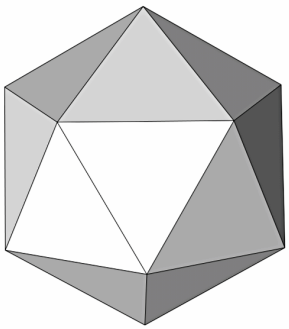


Abbildung 2: Bildquelle: http://www.3d-meier.de/tut12/Ik/Ikosaeder1.gif

Durch eine ausreichend hochfrequente Interpolation der Dreiecksflächen erhält man ein Gitter, bestehend aus Dreiecken, auf der Oberfläche des Ikosaeders, mit dem eine gültige Überdeckung garantiert werden kann. Man geht dazu rekursiv vor, das heißt, man findet die Mittelpunkte der Seiten der bestehenden Dreiecke und verwendet diese als Eckpunkte für die nächste Generation an Dreiecken. Alle so entstandenen Schnittpunkte des Gitters interpretiert man als Knoten eines Graphen, die abschließend noch auf die Kugeloberfläche projiziert werden müssen. Nachfolgende Abbildung verdeutlicht das Vorgehen:

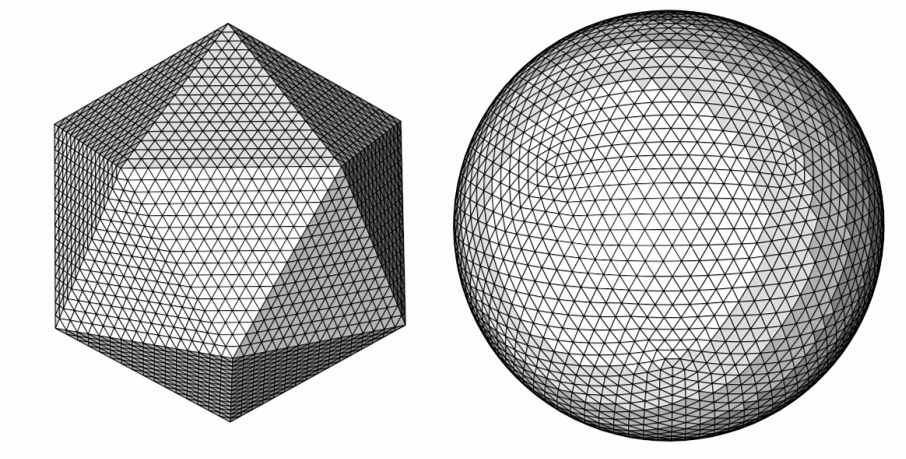


Abbildung 3: bearbeitet nach:

http://www.3d-meier.de/tut12/Ik/Ikosaeder16.gif,
http://www.3d-meier.de/tut12/Ik/Ikosaeder16Dome.gif

Auf diese Weise erhält man eine strukturierte Knotenverteilung auf der S^2. Eine mögliche Art der Projektion ist dabei die Normierung der Vektoren der einzelnen Knoten. Diese erhalten dadurch eine Länge von 1 und liegen somit auf der Einheitskugel. Das Problem, das sich daraus ergibt, ist, dass durch diese Art der Zentralprojektion, ebenso wie bei der unmodifizierten Zufallsverteilung, Ballungen von Knoten, und damit keine Gleichverteilung, entstehen. An den Stellen, an denen das Ikosaeder näher an der Kugeloberfläche liegt, als an anderen, wie etwa an den ursprünglichen 12 Ecken und auch an den Kanten, liegt eine verdichtete Knotenverteilung vor, an sphärenfernen Bereichen entsprechend eine eher lichte. Möchte man die Lösung an dieser Stelle weiter verbessern, so liefert Bauer [2] einen Algorithmus für ein homogenisiertes Ikosaeder. Dieser ist allerdings sehr komplex zu implementieren.

2.3 Kugelspirale (NK)

Zuletzt sei in diesem Abschnitt das Modell der Kugelspirale kurz angesprochen. Da jenes unserem Lösungsmodell entspricht, finden sich weitere Ausführungen dazu in den folgenden Kapiteln.
Eine Kugelspirale lässt sich wie folgt parametrisieren:

$$\begin{aligned} x &= \sin(\theta) \cdot \cos(C \cdot \theta) \\ y &= \sin(\theta) \cdot \sin(C \cdot \theta) \\ z &= \cos(\theta) \\ \theta &\in [0, \pi] \end{aligned}$$

Man erkennt hierbei die Ähnlichkeit zur Parametrisierung der Einheitskugel. Einzig der ursprüngliche Winkel φ ist nun linear abhängig vom Winkel θ. Der Faktor C ist dabei ein Maß für die Zahl der Windungen, die die Spirale um die Kugel herum zurücklegt, wenn man ihn halbiert. Eine graphische Veranschaulichung einer Kugelspirale um die Einheitskugel sieht man hier:

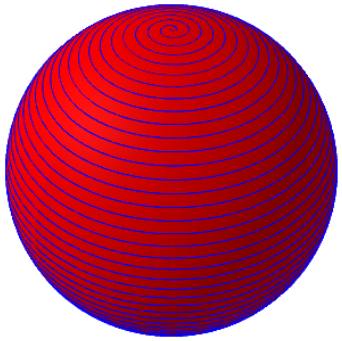


Abbildung 4: Kugelspirale, $C = 100$, 50 Windungen

Nun sind auf der Spirale die Knoten des Graphen zu platzieren. Auch hierbei sind viele Möglichkeiten denkbar. Beispielsweise können die Knoten mit gleichem Abstand, aber auch adaptiv oder völlig zufällig verteilt werden. Wir halten uns hier wieder an Bauer, der in seinem Artikel die sog. „Bauerspirale“ vorschlägt. Dabei setzt man C auf $\sqrt{n \cdot \pi}$ mit n als Anzahl der zu verteilenden Knoten. Entsprechende Koordinaten der Punkte kann man in [2] nachlesen.
Eine umfangreiche Analyse des Modells folgt im nächsten Abschnitt.

3 Modellanalyse und Lösungsverfahren (JR)

Zunächst möchten wir die Konstruktion der Bauer-Spirale nach [2] näher erläutern. Wie schon im letzten Abschnitt kurz angerissen, wird hierbei der Parameter C einer Kugelspirale auf $C = \sqrt{n \cdot \pi}$ gesetzt, wobei n die Anzahl der Knoten auf der Spirale ist. Die sphärischen Koordinaten der Knoten auf der Spirale setzen wir auf

$$\begin{aligned} z_k &= 1 - \frac{2k-1}{n} \\ \theta_k &= \arccos(z_k) \\ \varphi_k &= C \cdot \theta_k \end{aligned}$$

In diesem Fall sind die euklidischen Koordinaten der Knoten auf der Spirale gleich

$$\begin{aligned} x_k &= \sin(\theta_k) \cdot \cos(\varphi_k) \\ y_k &= \sin(\theta_k) \cdot \sin(\varphi_k) \\ z_k &= 1 - \frac{2k-1}{n} \text{ (wie oben)} \end{aligned}$$

Ein Vergleich mit der allgemeinen Parametrisierung in Kapitel 2 auf Seite 5 zeigt uns, dass wir hier tatsächlich n Knoten auf einer Kugelspirale verteilt haben.
Alle in Kapitel 2 genannten Vorschläge zur Verteilung von Knoten auf der Sphäre bieten den Vorteil einer einfachen Modellierung und Implementierung.
So fällt es etwa nicht schwer, zufällig Punkte auf der Sphäre zu verteilen. Jedoch stellt uns der Ansatz einer Zufallsverteilung (2.1) vor einige Probleme. Zum einen haben wir keine Kontrolle über die Gleichmäßigkeit der Knotenverteilung, zum anderen können wir i. A. auch für große n keine gültige Überdeckung der Sphäre mit n Kappen gewährleisten. In Kapitel 2 haben wir hierzu schon Beispiele angeführt. Es ist wenig überraschend, dass eine Zufallsverteilung von Knoten wenig zielführend bei der Lösung des Optimierungsproblems ist, die Kugel mit möglichst wenigen sphärischen Kappen zu überdecken.
Vielversprechender ist der Ansatz des Ikosaeders (2.2) zur Verteilung von Knoten auf der Sphäre. Die Konstruktion des Gitters wurde bereits in Kapitel 2 näher erläutert. Hierbei macht man sich zum Vorteil, dass für das zugrundeliegende planare Optimierungsproblem, also der Überdeckung einer Ebene mit gleich großen Kreisen, bereits Lösungskonstruktionen existieren. Außerdem lässt sich das sphärische Optimierungsproblem in guter Näherung durch das planare Optimierungsproblem modellieren, wie wir in unserem Vortrag des Systemarchitekten zu Modellierung und Methodik erfahren konnten. Ein großer Vorteil dieses Lösungsansatzes ist, dass wir leicht eine gültige Überdeckung der Sphäre konstruieren können. Bei der Wahl eines besonders feinen Gitters auf dem Ikosaeder ist auch nach Projektion auf die Sphäre weiterhin eine gültige Überdeckung (d.h. jeder beliebige Punkt auf der Sphäre wird überdeckt). Allerdings ist die resultierende Überdeckung nicht gleichmäßig, sondern es entstehen Pole mit besonders dichter Lage der Kappen, aber auch Abschnitte mit sehr geringer Anzahl an Kappen auf der Sphäre, wie wir bereits in Kapitel 2 gesehen haben.

Aus verschiedenen Gründen haben wir uns für die Verteilung von Knoten auf der Sphäre mithilfe der Bauerspirale nach [2] und der Parametrisierung wie zu Beginn dieses Abschnittes entschieden, denn dieser Ansatz bringt besonders viele Vorteile mit sich. So sind etwa die Knoten auf der Spirale

bereits durch Angabe der Knotenzahl n und des „Steigungs“-Parameters C eindeutig bestimmt und daher das Gitter auf der Kugel leicht zu konstruieren. Wir erhalten nach [2] eine besonders gleichmäßige Verteilung von Knoten auf der Sphäre. „Gleichmäßig“ bedeutet in diesem Zusammenhang, dass die statistische Varianz der Dichte der Knoten auf der Kugel nahezu 0 ist. Mit anderen Worten, die Größe der Voronoi-Zellen der verteilten Knoten weicht nur gering vom Mittelwert ab. Dabei sei noch kurz auf die Konstruktion von Voronoi-Zellen hingewiesen. Diese bestehen aus der Umgebung des jeweiligen Knotens, die alle Punkte auf der Sphäre enthält, die näher am jeweiligen Knoten, als an allen anderen, liegen.

Natürlich sind wir an dieser Stelle noch nicht fertig, sondern erst am Beginn der Arbeit. Wir möchten unser Vorgehen genauer beschreiben. Zunächst legen wir mittels der Bauerspirale Knoten, die ein Gitter, entsprechend der Knotenmenge eines Graphen, auf der Sphäre formen. Bei unserem Optimierungsproblem handelt es sich nun um ein (Minimal-)Dominating-Set-Problem auf dieser Knotenmenge. Wir werden einen Greedy-Algorithmus verwenden, um eine Teilmenge dieses Gitters als Kappenmittelpunkte einer gültigen Überdeckung auszuwählen. Zudem werden weitere Kappenmittelpunkte in die Überdeckung aufgenommen oder, falls notwendig, bereits bestehende Kappenmittelpunkte umgesetzt bzw. entfernt. Außerdem werden wir zum Schluss den von den Projektbetreuern zur Verfügung gestellten Lösungsprüfer verwenden, um die Gültigkeit der Überdeckung zu verifizieren. Falls nötig, werden in diesem Schritt auch noch einige fehlende Kappenmittelpunkte in den Graphen aufgenommen.

Bei der Implementierung und Programmierung werden wir uns einige Gedanken zu der Optimierung dieses Lösungskonzeptes machen. Eine besonders wichtige Rolle wird hierbei eine passende Wahl der definierenden Parameter der Kugelspirale spielen, um z. B. den Aufwand beim anschließenden Greedy-Algorithmus zu minimieren oder vom Lösungsprüfer hinzugefügte Kappenmittelpunkte zu reduzieren. Wir werden also verschiedene Wahlen der Parameter austesten und dabei jeweils die berechnete Anzahl der Kappen für eine gültige Überdeckung der Sphäre vergleichen. In den kommenden Kapiteln werden unsere gewonnenen Erkenntnisse zum Lösungsansatz der Bauerspirale mit anschließender Optimierung vorgestellt.

4 Implementierung (AB)

4.1 Vorüberlegungen zur Implementierung (AB)

Wie in den vorangehenden Kapiteln angesprochen, unterteilt sich das gewählte Lösungsverfahren in drei Phasen: In der ersten Phase wird die Kugeloberfläche diskretisiert. Hierfür wurden in Kapitel 2 drei Möglichkeiten vorgestellt. Daran anschließend kann aus der resultierenden Punktmenge $P \subset S^2$ ein Graph $G = (V, E)$ generiert werden, indem

$$V = P, \qquad E = \{(x, y) \in P^2 | d(x, y) < r\}$$

gewählt werden. Hierbei beschreibt d die Abstandsfunktion und $r \in \mathbb{R}^+$ den Öffnungsradius. Für die in Kapitel 1 angesprochene Sternenkarte ergibt sich beispielsweise:

$$d : P^2 \to [0, \infty) \text{ mit } d(x, y) = \arccos(x \cdot y) \text{ und } r = 1.75°$$

In der zweiten Phase wird eine dominierende Menge (eng. Dominating-Set) S des Graphen G bestimmt [3]. Dies ist eine Menge S, sodass für alle $v \in V$ gilt, dass $v \in S$ oder ein Nachbar von v in S. Also:

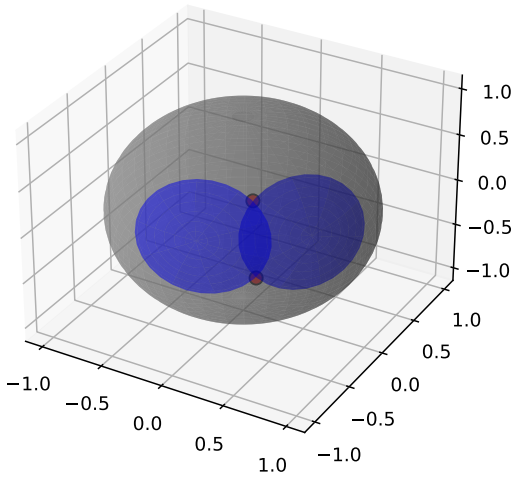
$$S \subset G \text{ ist dominierende Menge } \iff \forall v \in V : v \in S \text{ oder } \exists x \in S : (x, v) \in E$$

(a) Dominierende Menge: $\{A, C\}$ (b) Dominierende Menge: $\{B, D, F\}$

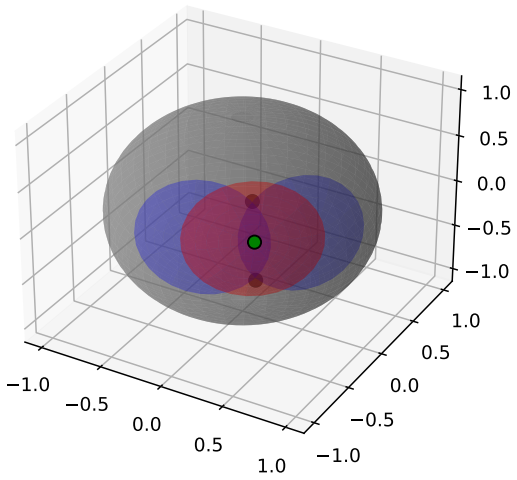
Abbildung 5: Beispiele für dominierende Mengen für einen Beispielgraphen

Hierzu wird ein Greedy-Ansatz verfolgt, sodass auf Basis einer Abbildung h, die Heuristik genannt wird, iterativ die beste Option ausgewählt wird [4]. Die Heuristik h beurteilt in diesem Fall die Güte eines Knotens des Graphen auf der Basis von mehreren Merkmalen. Ein Merkmal, welches trivialerweise für die Bestimmung einer dominierenden Mengen eingesetzt werden kann, ist die Anzahl der neu überdeckten Knoten.
Weitere Überlegungen haben ergeben, dass die Schnittpunkte zweier Lösungskugelkappen (vgl. Abbildung 6a) interessant sein können. Da die Kugelkappen offen sind, werden die Schnittpunkte zweier Kugelkappen nicht direkt von den dazugehörigen Kappen überdeckt. Daher kann eine dichte Ansammlung von nicht überdeckten Schnittpunkten auf ein Loch in der Überdeckung hinweisen, welches von der bisherige Lösung nicht überdeckt wird. Dieses Merkmal verbessert weder die Suche nach einer dominierenden Menge, noch beschleunigt sie diese. Es beeinflusst lediglich die resultierende Menge derart, dass die Lösung der Diskretisierung möglichst alle Punkte auf der Sphäre S^2 überdeckt.
Betrachtet man die Abbildung 6b, so fällt schnell auf, dass sich die Ergebnisse durch das Hinzufügen dieses Merkmals nicht zwangsläufig verbessern müssen. Eine Kappe, welche ihren Mittelpunkt in der Schnittfläche zweier Lösungskappen hat, überdeckt einerseits wenig neue Fläche. Andererseits werden zwei Schnittpunkte überdeckt, sodass diese Kappe bei großer Gewichtung der Schnittpunkte als sehr

gute Option betrachtet werden kann. Um dem entgegenzuwirken haben wir ein weiteres Merkmal hinzugefügt, welches die Anzahl der überdeckten Mittelpunkte solcher Überschneidungen angibt. Dieses kann sich in der Abbildung h sich negativ auf die Güte der betrachteten Option auswirken und diesen Effekt eindämmen.



(a) Visualisierung der Schnittpunkte (rot) zweier Lösungskappen (blau)



(b) Visualisierung des Mittelpunktes (grün) und der Schnittpunkte (schwarz) zweier Lösungskappen (blau) und der Lösungskappe um den Mittelpunkt (rot)

Abbildung 6: Visualisierung der Schnitt- und Mittelpunkte der Schnittflächen zweier Lösungskappen

Somit ergibt sich die folgenden Heuristik, welche die einzelnen Knoten bewertet:

$$h : V \to \mathbb{R}_0^+$$
$$v \mapsto w_1 \cdot h_{new\_cover}(v) + w_2 \cdot h_{new\_inter}(v) + w_3 \cdot h_{mids}(v)$$

Die Abbildungen $h_{new\_cover}, h_{new\_inter}, h_{mids} \to \mathbb{N}_0$ beschreiben

- die Anzahl der überdeckten Knoten, die von der aktuellen Lösung noch nicht überdeckt werden,
- die Anzahl der noch nicht überdeckten Schnittpunkte von Lösungskugelkappen und
- die Anzahl der überdeckten Mittelpunkte von Schnittflächen.

$w_1, w_2, w_3 \in \mathbb{R}$ beschreiben Parameter, die noch passend bestimmt werden müssen. Dazu sei auf das Kapitel 5 verwiesen.
In der letzten Phase wird ein Ungleichungssystem, aufbauend auf der Menge S, gelöst, um Punkte auf der Kugeloberfläche zu finden, die nicht durch eine der Kugelkappen um die Mittelpunkte $p \in S$ abgedeckt werden. Dieses wurde von Jan Rolfes und Andreas Bärmann entwickelt und bereitgestellt.

4.2 Technische Implementierung und Aufwandsanalyse (AB)

Aus den gerade geschilderten Phasen ergibt sich der in Algorithmus 1 auf Seite 12 dargestellte Pseudo-Code, welcher kurz erklärt werden soll. Grundsätzlich benötigt das Verfahren die Parameter r als Radius der Kugelsegmente, N als Anzahl der für die Diskretisierung genutzten Knoten, w_{explor} als Faktor für die iterative Exploration während der Suche und die gewichtete Funktion h_{w_1,w_2,w_3} als Heuristik.
Die Zeilen 1 bis 5 beschreiben die Initialisierung der benötigten Variablen, die Diskretisierung der Kugel durch ein Gitter und die Generierung des Graphen G. In den anschließenden Zeilen 6 bis einschließlich 17 wird die Suche nach einer dominierenden Menge beschrieben. Hierbei werden die aktuell betrachteten Knoten in einer Warteschlange (PrioQueue) gespeichert, die als Maxheap implementiert wird [5]. Daher besteht jeder Iterationsschritt aus der Entnahme der besten Option aus der Warteschlange, der Aktualisierung der Mengen der Schnittpunkte und Mittelpunkte und der Aktualisierung der Fitness anliegender Knotenpunkte.
Bei der Neubewertung einzelner Knoten muss unterschieden werden, da dieser unter Umständen bereits mit einer besseren Bewertung in der Warteschlange eingetragen ist (vgl. Zeilen 11-12). Eine Möglichkeit ist es, das entsprechende Tupel im Baum zu suchen. Dies resultiert in einer hohen Laufzeit. Schneller ist es, Tupel beispielsweise in einer Hashmap als ungültig zu markieren und ungültige Elemente bei Aufrufen der Funktion pop() zu überspringen. Dadurch wird bei der Aktualisierung lediglich das alte Tupel als ungültig markiert und das neue Gewicht eingefügt [6].
Ist ein Knoten noch nicht in der Warteschlange enthalten, so kann dieser direkt eingefügt werden (vgl. Zeile 13-16). Da nach der Terminierung des Algorithmus alle Punkte der Sphäre S^2 überdeckt sein müssen, wird der Graph der Lösungsmittelpunkte zusammenhängend sein. Daher reicht es, nur anliegende Knoten im Umkreis zu betrachten. Hierzu wird der Parameter $w_{explor} > 0$ zur Skalierung in Abhängigkeit des Radius r genutzt (vgl. Zeile 13).
Abgeschlossen wird der Algorithmus mit der dritten Phase, in der Knoten, die von S nicht überdeckte werden, ermittelt und hinzugefügt werden (vgl. Zeile 17).

Nun soll das Laufzeit- und Speicherverhalten analysiert werden, wobei sich jeweils auf die Analyse bezüglich des Parameters N beschränkt wird.
Innerhalb der Zeilen 1 bis 5 werden lediglich die Initialisierungen von konstanter Laufzeit, die Gittererstellung von linearer Laufzeit und die Generierung des Graphen in quadratischer Laufzeit durchgeführt. Damit ergibt sich für diesen Teil eine Komplexitätsklasse $\mathcal{O}(N^2)$.
Anschließend wird bis zu N-mal die beste Option aus der Warteschlange entnommen und für maximal alle N Knoten die Heuristik berechnet. Der resultierende Wert wird jeweils in der Warteschlange aktualisiert oder in diese eingefügt (vgl. Zeile 7, 11-16). Die Zeilen 8 und 9 besitzen jeweils die Komplexität $\mathcal{O}(|S|) = \mathcal{O}(N)$, da in beiden Fällen die Distanz zu N Knotenpunkten berechnet werden muss. Somit ergibt sich für die Zeilen 6 bis 16 die Zeitkomplexitätsklasse:

$$\mathcal{O}(N \cdot (\log(N) + 2 \cdot N + N \cdot \log(N) \cdot \mathcal{O}(h))) = \mathcal{O}(N^3 \cdot \log(N))$$

An dieser Stelle wurde verwendet, dass eine effiziente Berechnung der Heuristik mit Matrizen für überdeckte Knoten, lineare Zeit benötigt[1].
Eine Laufzeitanalyse des letzten Schrittes gestaltet sich hier schwer, da dieser sehr von der Güte der Lösung abhängt, die von allen Parametern beeinflusst wird. Somit ist aussagekräftige Laufzeitkomple-

[1] Da der gesamt Code in Python geschrieben ist und die Berechnung der Heuristik und Distanzen zu großen Teilen von C-basierten Modulen durchgeführt werden, besitzt diese Schranke in der Praxis kaum Aussagekraft.

Algorithm 1: Lösungsverfahren mittels Greedy-Suche nach dominierender Menge. Aufbauend auf [4]

```
   input: r, N, w_explor, h_{w1,w2,w3}
 1 h = h_{w1,w2,w3};
 2 V = generateLattice(N);
 3 G = (V,E) = generateGraph(V) ;
 4 S, Inter, Mids = {},{},{};
 5 PrioQueue = {(v0,0)} ;                          // v0 ∈ V als initialer Knoten
 6 while S is not a Dominating-Set do
 7 |   (v, _ ) = PrioQueue.pop();
 8 |   Inter.update(v, S);
 9 |   Mids.update(v, S);
10 |   S = S ∪ {v};
11 |   foreach w ∈ PrioQueue.elements and d(v,w) < 2 · r do
12 |   |   w.updateFittness(S, Inter, Mids, h);
13 |   foreach w ∈ V \ PrioQueue.elements and d(v,w) < w_explor · r do
14 |   |   ele = (w, _);
15 |   |   ele[1] = w.set_and_getFittness(S, Inter, Mids,h);
16 |   |   PrioQueue.put(ele);
17 S = S ∪ collectMissing(S) ;
18 return S
```

xität in Abhängigkeit von N nur für die Zeilen 1 mit 16 sinnvoll. Diese ist:

$$\mathcal{O}(N^2 + N^3 \cdot \log(N)) = \mathcal{O}(N^3 \cdot \log(N))$$

Um den benötigten Speicher zu analysieren, reicht es aus, die in den Zeilen 2 bis 5 initialisierten Mengen zu betrachten und die maximale Größe abzuschätzen. Zunächst ist $|V| = N$ und $|E| \leq N^2$, denn G ist ein ungerichteter Graph ohne Mehrfachkanten. Da V trivialerweise eine dominierende Menge ist, muss $|S| \leq N$ sein. Für jede Wiederholung der Zeilen 6 bis 16 werden maximal $|S|$ Schnittflächen generiert und $|V|$ Punkte in die Warteschlange eingefügt. Diese Zeilen werden $|S|$-mal wiederholt, sodass

$$|\text{Inter}|, |\text{Mids}|, |\text{PriooQueue}| \leq N^2$$

ist. Es ergibt sich also eine Speicherkomplexität von

$$\mathcal{O}(N + N^2 + N + 3 \cdot N^2) = \mathcal{O}(4 \cdot N^2) = \mathcal{O}(N^2)$$

Wie in Kapitel 5 gezeigt wird, ist eine Größenordnung von 10^5 bis 10^6 für N bei kleinen Radien realistisch. Bei einer bitweisen Speicherung dieser Daten im Hauptspeicher würden demnach bis zu $4 \cdot 10^{12}$ Bit = 500 GB benötigt. Somit wäre dieses Verfahren bei kleinen Radien aufgrund eines zu großen Speicherverbrauchs nicht mehr allein im Hauptspeicher zu speichern. Dadurch würde sich die Laufzeit wegen der erhöhten Zugriffszeiten drastisch erhöhen.
Um dem entgegenzuwirken, kann der oben dargestellte Algorithmus abschnittsweise auf die Sphäre beziehungsweise das Gitter angewandt werden (vgl. Abbildung 7). Durch dieses iterative Erweitern einer Lösung muss zu jedem Zeitpunkt nur ein Bruchteil des Gitters betrachtet werden, sodass auch die Variablen quadratischer Größe einen deutlich kleineren Speicherverbrauch erfordern.

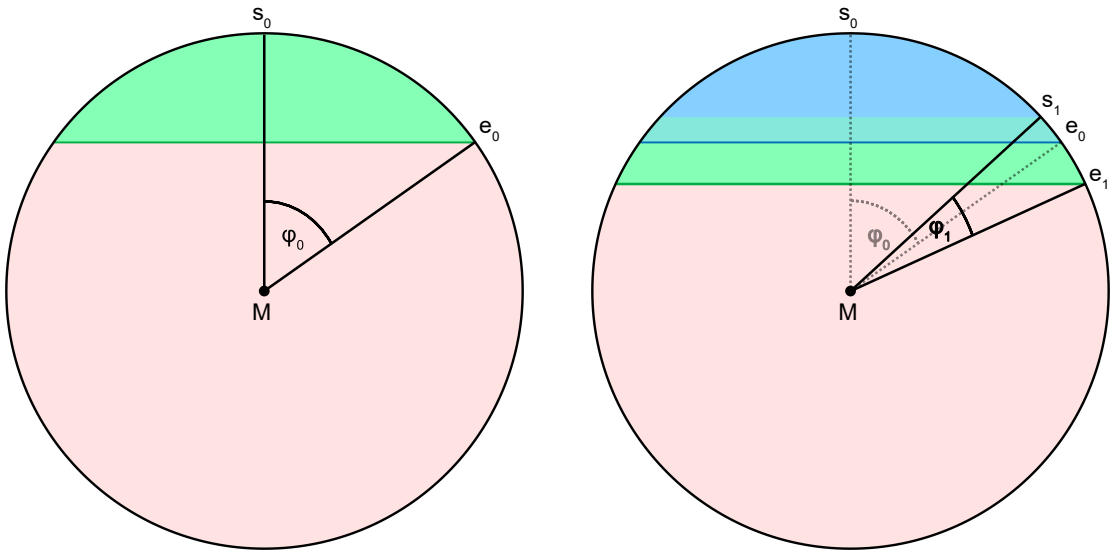


Abbildung 7: Die ersten beiden Unterteilung des iterativen Vorgehens. Hierbei beschreibt der grüne Abschnitt jeweils das aktuell betrachtete Segment. In blau wird der Bereich markiert, für den eine Überdeckung bereits berechnet wurde.

Für die einzelnen Abschnitte ist allerdings wichtig, dass sie sich in einem Winkel von $2 \cdot r$ überschneiden. Nur so kann eine akkurate Bewertung einzelner Knoten und eine passende Erweiterung der vorhandenen Lösung gewährleistet werden. Außerdem sollten die Segmente möglichst gleichgroß sein. Man berechne also für m Abschnitte aus der Oberfläche einer Kugelkappe die Start- und Endwerte für die Unterteilung der $z-$Koordinate mit

$$
\begin{aligned}
Z_i &= \begin{cases} 0 & \text{falls } i = 0 \\ \arccos(\cos(Z_{i-1} - \frac{2}{m})) - r & \text{sonst} \end{cases} \\
s_i &= \cos(Z_i) \\
e_j &= \cos(Z_{i+1} + 2r)
\end{aligned}
$$

für $i, j \in \{1, \ldots, m\}$ mit $j \neq m$ und $e_m = \cos(\pi) = -1$.
Im Algorithmus 2 sind diese Anpassungen für ein adaptives Vorgehen auf der Sphäre implementiert. Das gesamte Projekt ist auf Github unter `https://github.com/Idontker/OptiMates` zu finden.

Algorithm 2: Algorithmus 1 erweitert um adaptives Vorgehen. Änderungen in rot.

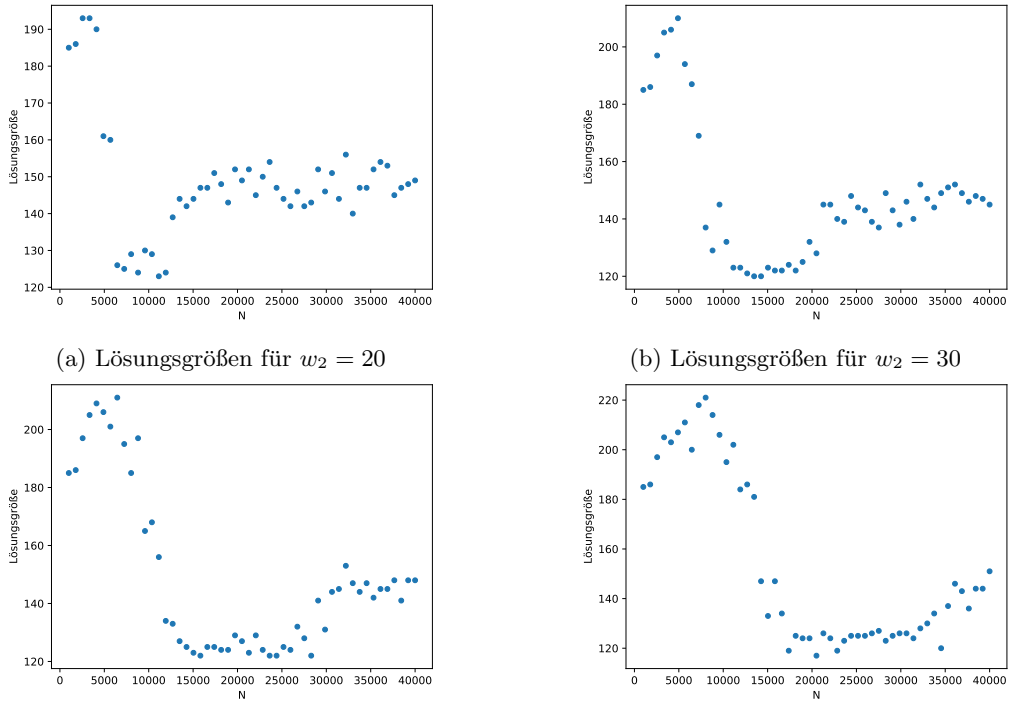
```
   input: r, N, w_explor, h_{w1,w2,w3}
 1 h = h_{w1,w2,w3};
 2 S = {};
 3 P = generateLattice(N);
 4 Parts = generateParts(P);
 5 foreach P ∈ Parts do
 6     G = (V,E) = generateGraph(P) ;
 7     Inter, Mids = {},{},{};
 8     PrioQueue = {(v0,0)} ;                          // v0 ∈ V als initialer Knoten
 9     while S is not a Dominating-Set do
10         (v, _ ) = PrioQueue.pop();
11         Inter.update(v, S);
12         Mids.update(v, S);
13         S = S ∪ {v};
14         foreach w ∈ PrioQueue.elements and d(v,w) < 2 · r do
15             w.updateFitness(S, Inter, Mids, h);
16         foreach w ∈ V \ PrioQueue.elements and d(v,w) < w_explor · r do
17             ele = (w, _);
18             ele[1] = w.set_and_getFittness(S, Inter, Mids,h);
19             PrioQueue.put(ele);
20 S = S ∪ collectMissing(S) ;
21 return S
```

5 Aufarbeitung der Daten und Ergebnisse (AB)

5.1 Parameterwahl (AB)

Betrachtet man Algorithmus 1, so fällt auf, dass für diesen eine Reihe von Parametern zu wählen sind. Für die Parameter der Heuristik kann man sich durch eine entsprechende Skalierung überlegen, dass $w_1 = 1$ gesetzt werden kann, ohne dass sich die Resultate verändern. Weiter wählen wir $w_3 = -w_2$. Dadurch fallen die überdeckten Mittelpunkte der Schnittflächen genauso ins Gewicht, wie noch nicht überdeckte Schnittpunkte, sodass Situationen wie in Abbildung 6b akkurater bewertet werden. Wir setzten auch $w_{explo} = 1.8$, da wir mir diesem die besten Erfahrungen gemacht haben. Somit sind nur noch drei Parameter zu wählen, denn der Radius r eines Kugelsegments ist durch die Problemstellung gegeben.
Grundsätzlich würde man annehmen, dass für ein größeres N die Werte besser oder zumindest nicht schlechter werden. Grund für diese Annahme ist, dass aus einem größeren N eine feinere Diskretisierung folgt. Die Abbildungen 8 zeigen allerdings, dass für N wachsend und w_2 konstant die Anzahl der Lösungsmittelpunkte für sehr große N steigt. Dies kann man dadurch erklären, dass w_2 für große N nicht mehr im Verhältnis zu der Dichte der Punkte eines Kugelsegments steht. Wählt man $w_2 = c \cdot \rho(N, r)$ für ein $c \in \mathbb{R}$ und $\rho(N, r)$ als Anzahl der Punkte auf einer Kugelkappe mit Radius r

bei einer Diskretisierung mit N Punkten[2], so zeigt sich der erwartete Effekt für wachsendes N (vgl. Abbildung 9).



(a) Lösungsgrößen für $w_2 = 20$

(b) Lösungsgrößen für $w_2 = 30$

(c) Lösungsgrößen für $w_2 = 40$

(d) Lösungsgrößen für $w_2 = 50$

Abbildung 8: Darstellung der Größe der Lösungsmengen für verschiedene Werte N und w_2, wobei der Öffnungsradius einer Kugelkappe immer $r = 13°$ ist.

Dabei fällt auf, dass ab einem gewissen N die Resultate nur gering schwanken. Betrachtet man statt N die Dichte für unterschiedliche Radien, so fällt auf, dass sich die Ergebnisse ab einem Schwellenwert von $\rho(N, r) \geq 150$ kaum unterscheiden. Damit kann also N in Abhängigkeit von r gewählt werden:

$$\begin{aligned} 150 \leq \rho(N, r) &= N \cdot \frac{vol(\{(x, y, z) \in S^2 : \arccos(z) < r\})}{vol(S^2)} \\ &= N \cdot \frac{2\pi(1 - \cos(r))}{4\pi} \\ \Rightarrow N &\geq 150 \cdot \frac{2}{1 - \cos(r)} \end{aligned}$$

Aus dieser Abschätzung folgt auch, dass für den Radius $r = 1.75$ die Anzahl der genutzten Knotenpunkte $N \geq 6.4 \cdot 10^6$ sein müsste. Die Betrachtung und Lösung möglicher Speicherprobleme in

[2]Hier wird angenommen, dass das gewählte Gitter einer Gleichverteilung sehr nahe kommt, sodass $\rho(N, r) = N \cdot \frac{vol(B_r)}{vol(S^2)}$ angenommen wird. Hierbei beschreibt $B_r \subset S^2$ eine Kugelkappe mit dem Radius r.

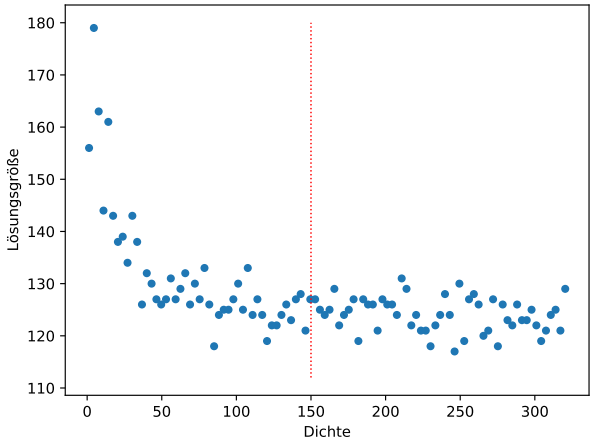


Abbildung 9: Darstellung der Größe der Lösungsmengen für verschiedene Punktedichten, wobei der Öffnungsradius einer Kugelkappe $r = 13°$ ist und w_2 in Abhängigkeit der Dichte gewählt wird.

Kapitel 4.2 war demnach sinnvoll und notwendig.
Auf der Basis weiterer Tests wurde für den Parameter w_2 ein guter Wert bei

$$w_2 = \frac{100}{513} \cdot \rho(N, r) \approx 0.195 \cdot \rho(N, r)$$

bestimmt.

5.2 Ergebnisse für diverse Radien (AB)

Abschließend sollen kurz die Ergebnisse betrachtet werden, die mit dem beschriebenen Verfahren erreicht werden. In allen getesteten Situationen generierte der Algorithmus unter Verwendung der Bauerspirale eine kleinere Lösungsmenge als bei der Verwendung eines Ikosaeders oder einer Zufallsverteilung (vgl. Kapitel 2). Die in Tabelle 1 aufgelisteten Ergebnisse sind daher alle aus einem Gitter basierend auf der Arbeit von Bauer entstanden [2].

r	untere Schranke	Bester Wert mit Algorithmus 1	Abweichung zur Schranke	Laufzeit	bekannte bessere Lösungen
70.6°	≈ 3	5	67 %	106 ms	4
37.4°	≈ 10	14	40 %	202 ms	12
22.7°	≈ 26	45	73 %	2.52 s	32
13.0°	≈ 79	117	48 %	11.6 s	99
1.75°	≈ 4288	6786	58 %	110 min	-

Tabelle 1: Auflistung der Ergebnisse unter Verwendung der oben genannten Parameter und der Bauerspirale als zugrundeliegendes Gitter. Dabei beschreibt r den Öffnungswinkel. Die untere Schranke wird mit $\frac{vol(S^2)}{vol(B_r)}$ berechnet, wobei $B_r \subset S^2$ eine Kugelkappe mit dem Radius r beschreibt. Die Laufzeit ergibt sich aus der Rechenzeit einer einfachen Ausführung. Hierzu wurde eine Recheneinheit mit folgender Hardware genutzt: Fujitsu Esprimo P758, Intel Core i7-9700, 64 GB RAM.

6 Ausblick (JR)

In diesem Artikel haben wir einige verschiedene Möglichkeiten vorgestellt, wie die Himmelskugel vollständig auf praktische Art und Weise mit sphärischen Kappen überdeckt werden kann. Besondere Aufmerksamkeit widmeten wir einem Greedy-Ansatz auf einem Gitter, das durch eine Kugelspirale konstruiert wird.
Dieser Ansatz verbindet eine anschauliche und einfache Modellierung mit einem guten Endresultat an benötigten Kappen zur Überdeckung der Himmelskugel. Außerdem birgt er noch hohes Verbesserungspotential. Sei es zum Beispiel durch Veränderung der Parameter der Kugelspirale, neuen Heuristiken für den Greedy-Algorithmus oder durch Algorithmen, bei einer gültigen Überdeckung einige Kappen zu verschieben und „überflüssige“ Kappen zu entfernen. Es bestehen also zahlreiche Möglichkeiten, in Zukunft unseren Lösungsansatz weiter zu optimieren, um bestmögliche Resultate zu erzielen.
Auf jeden Fall steht den Astronomen des Vera C. Rubin Observatory bereits mithilfe unserer bisherigen Ergebnisse eine handliche Methode zur Verfügung, mit dem im Jahr 2022 neu errichteten Spiegelteleskop eine sich aktualisierende Sternenkarte zu erstellen.

Literatur

[1] J. H. Rolfes und F. Vallentin, "Covering compact metric spaces greedily," *arXiv preprint arXiv:1710.06002*, 2017.

[2] R. Bauer, "Distribution of points on a sphere with application to star catalogs," *Journal of Guidance, Control, and Dynamics*, Jg. 23, Nr. 1, S. 130–137, 2000.

[3] M. S. Rahman u. a., *Basic graph theory*. Springer, 2017, S. 70–71.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest und C. Stein, *Introduction to algorithms*. MIT press, 2009, S. 414–440.

[5] ——, *Introduction to algorithms*. MIT press, 2009, S. 151–169.

[6] "Python 3.9.6 documentation: heapq — Heap queue algorithm." Zuletzt aufgerufen: 2021-08-06. (2021), Adresse: `https://docs.python.org/3/library/heapq.html#priority-queue-implementation-notes`.