Bundeswettbewerb Informatik

Lösungshinweise

Allgemeines

Es ist immer wieder bewundernswert, wie viel an Ideen, Wissen, Fleiß und Durchhaltevermögen in den Einsendungen zur 2. Runde eines Bundeswettbewerbs Informatik stecken. Um aber die Allerbesten für die Endrunde zu bestimmen, müssen wir die Arbeiten kritisch begutachten und hohe Anforderungen stellen. Deswegen sind Punktabzüge die Regel und Bewertungen mit Pluspunkten über die Erwartungen hinaus die Ausnahme.

Spannende bzw. schwierige Erweiterungen der Aufgabenstellung sind Extrapunkte wert, wenn sie auch praktisch realisiert wurden. Weitere Ideen ohne Implementierung und geringe Verbesserungen der bereits implementierten Lösung einer Aufgabe gelten allerdings nicht als geeignete Erweiterungen. Intensive theoretische Überlegungen wie z. B. ein korrekter Beweis zur Komplexität des Problems werden ebenfalls mit zusätzlichen Punkten belohnt.

Falls Ihre Einsendung nicht herausragend bewertet wurde, lassen Sie sich auf keinen Fall entmutigen! Allein durch die Arbeit an den Aufgaben und ihren Lösungen hat jede Teilnehmerin und jeder Teilnehmer viel gelernt; diesen Effekt sollten Sie nicht unterschätzen. Selbst wenn Sie die Lösung zu nur einer Aufgabe einreichen konnten, so kann die Bewertung Ihrer Einsendung bei der Anfertigung künftiger Lösungen hilfreich für Sie sein.

Bevor Sie sich in die Lösungshinweise vertiefen, lesen Sie bitte kurz die folgenden Anmerkungen zu den Einsendungen und beiliegenden Unterlagen durch.

Bewertungsbogen

Aus der 1. Runde oder auch aus früheren Wettbewerbsteilnahmen kennen Sie den Bewertungsbogen, der angibt, wie Ihre Einsendung die einzelnen Bewertungskriterien erfüllt hat. Auch in dieser Runde können Sie den Bewertungsbogen im Anmeldesystem AMS einsehen. In der 1. Runde ging die Bewertung noch von 5 Punkten aus, von denen bei Mängeln abgezogen werden konnte. In der 2. Runde geht die Bewertung von zwanzig Punkten aus, bei denen Punkte abgezogen und manchmal auch hinzuaddiert werden konnten. In dieser Runde gibt es auch deutlich mehr Bewertungskriterien als in der 1. Runde.

Terminlage

Für Abiturientinnen und Abiturienten ist der Terminkonflikt zwischen Abiturvorbereitung und 2. Runde sicher nicht ideal. Doch leider bleibt dem Bundeswettbewerb Informatik nur die erste Jahreshälfte für diese Runde: In der zweiten Jahreshälfte läuft nämlich die zweite Runde des

Mathematikwettbewerbs, dem wir keine Konkurrenz machen wollen. Aber: die Bearbeitungszeit für die Runde beträgt etwa vier Monate. Frühzeitig mit der Bearbeitung der Aufgaben zu beginnen ist der beste Weg, zeitliche Engpässe am Ende der Bearbeitungszeit gerade mit der wichtigen, ausführlichen Dokumentation der Aufgabenlösungen zu vermeiden. Aufgaben der 2. Runde sind oft deutlich schwerer zu lösen, als sie auf den ersten Blick erscheinen. Erst bei der konkreten Umsetzung einer Lösungsidee stößt man manchmal auf Besonderheiten bzw. noch zu lösende Schwierigkeiten, was dann zusätzlicher Zeit bedarf. Daher ist es sinnvoll, die einzureichenden Aufgaben nicht nacheinander, sondern relativ gleichzeitig zu bearbeiten, um nicht vom zeitlichen Aufwand der jeweiligen Aufgabe kurz vor Ablauf der Bearbeitungszeit unangenehm überrascht zu werden.

Dokumentation

Es ist sehr gut nachvollziehbar, dass Sie Ihre Energie bevorzugt in die Lösung der Aufgaben, die Entwicklung Ihrer Ideen und deren Umsetzung in Software fließen lassen. Doch ohne eine verständliche Beschreibung der Lösungsideen und ihrer jeweiligen Umsetzung, eine übersichtliche Dokumentation der wichtigsten Komponenten Ihrer Programme, eine geeignete Kommentierung der Quellcodes und eine ausreichende Zahl sinnvoller Beispiele (welche die verschiedenen bei der Lösung des Problems zu berücksichtigenden Fälle abdecken) ist eine Einsendung nur wenig wert.

Bewerterinnen und Bewerter können die Qualität Ihrer Aufgabenlösungen nur anhand dieser Informationen vernünftig einschätzen. Mängel in der Dokumentation der Einsendung können nur selten durch Ausprobieren und Testen der Programme ausgeglichen werden – wenn die Programme denn überhaupt ausgeführt werden können: Hier gibt es gelegentlich Probleme, die meist vermieden werden könnten, wenn Lösungsprogramme vor der Einsendung nicht nur auf dem eigenen, sondern auch einmal auf einem fremden Rechner auf Lauffähigkeit getestet würden. Insgesamt sollte die Erstellung der Dokumentation die Programmierarbeit begleiten oder ihr teilweise sogar vorangehen: Wer nicht in der Lage ist, Idee und Modell präzise zu formulieren, bekommt auch keine saubere Umsetzung hin, in welcher Programmiersprache auch immer.

Einige unterhaltsame Formulierungsperlen sind im Anhang wiedergegeben.

Bewertung

Bei den im Folgenden beschriebenen Lösungsideen handelt es sich nicht um perfekte Musterlösungen, sondern um sinnvolle Lösungsvorschläge. Dies sind also nicht die einzigen Lösungswege, die wir gelten ließen. Wir akzeptieren vielmehr in der Regel alle Ansätze, auch ungewöhnliche, kreative Bearbeitungen, die die gestellte Aufgabe vernünftig lösen und entsprechend dokumentiert sind. Einige der Fußnoten in den folgenden Lösungsvorschlägen verweisen auf weiterführende Fachliteratur für besonders Interessierte; Lektüre und Verständnis solcher Literatur wurden von den Teilnehmenden natürlich nicht erwartet.

Unabhängig vom gewählten Lösungsweg gibt es aber Dinge, die auf jeden Fall von einer guten Lösung erwartet wurden. Zu jeder Aufgabe wird deshalb in einem eigenen Abschnitt, jeweils am Ende des Lösungsvorschlags erläutert, auf welche Kriterien bei der Bewertung dieser Aufgabe besonders geachtet wurde. Dabei können durchaus Anforderungen formuliert werden, die aus der Aufgabenstellung nicht hervorgingen. Letztlich dienen die Bewertungskriterien dazu,

die allerbesten unter den sehr vielen guten Einsendungen herauszufinden. Außerdem gibt es aufgabenunabhängig einige Anforderungen an die Dokumentation (klare Beschreibung der Lösungsidee, genügend aussagekräftige Beispiele und wesentliche Auszüge aus dem Quellcode) einschließlich einer theoretischen Analyse (geeignete Laufzeitüberlegungen bzw. eine Diskussion der Komplexität des Problems) sowie an den Quellcode der implementierten Software (mit übersichtlicher Programmstruktur und verständlicher Kommentierung) und an das lauffähige Programm (ohne Implementierungsfehler). Wünschenswert sind auch Hinweise auf die Grenzen des angewandten Verfahrens sowie sinnvolle Begründungen z. B. für Heuristiken, vorgenommene Vereinfachungen und Näherungen. Geeignete Abbildungen und eigene zusätzliche Eingaben können die Erläuterungen in der Dokumentation gut unterstützen. Die erhaltenen Ergebnisse für die Beispieleingaben (ggf. mit Angaben zur Rechenzeit) sollten leicht nachvollziehbar dargestellt sein, z. B. durch die Ausgabe von Zwischenschritten oder geeignete Visualisierungen. Eine Untersuchung der Skalierbarkeit des eingesetzten Algorithmus hinsichtlich des Umfangs der Eingabedaten ist oft ebenfalls nützlich.

Danksagung

Alle Aufgaben wurden vom Aufgabenausschuss des Bundeswettbewerbs Informatik entwickelt: Peter Rossmanith (Vorsitz), Hanno Baehr, Jens Gallenbacher, Rainer Gemulla, Torben Hagerup, Christof Hanke, Thomas Kesselheim, Arno Pasternak, Holger Schlingloff, und Melanie Schmidt sowie (als Gäste) Wolfgang Pohl und Hannah Rauterberg.

An der Erstellung der im Folgenden skizzierten Lösungsideen wirkten neben dem Aufgabenausschuss vor allem folgende Personen mit: (Aufgabe 1), (Aufgabe 2) sowie (Aufgabe 3). Allen Beteiligten sei für ihre Mitarbeit ganz herzlich gedankt.

Aufgabe 1: Laubmaschen

1.1 Lösungsidee

In dieser Aufgabe soll dem Hausmeister geholfen werden, einen quadratischen Schulhof mit Hilfe eines Laubbläsers von Laub zu befreien. Dazu soll eine Simulation für das Laubblasen implementiert und eine Strategie entwickelt werden, mit der möglichst viele Blätter auf einem Nicht-Rand-Quadrat gesammelt werden können. Die Anzahl der Blasvorgänge spielt dabei eine untergeordnete Rolle. Im Rahmen der Simulation muss sich Gedanken dazu gemacht werden, an welchem Punkt der Hausmeister das Laubblasen aufgibt und die Arbeit mit dem Besen beendet. Sollte die Simulation auch für nicht rechteckige Schulhofformen funktionieren, handelt es sich um eine Erweiterung der Aufgabenstellung.

Die Wirkung eines Blasvorgangs ist in Abbildung 1.1 dargestellt. Für Quadrate am Rand und in den Ecken des Schulhofs ist die Wirkung nicht vorgegeben und muss eigenständig festgelegt werden. Da der Hausmeister den Schulhof nicht verlassen darf, stellt dies eine besondere Herausforderung dar. Die Wirkung eines Blasvorgangs sollte außerdem für jedes Blatt einzeln simuliert werden.

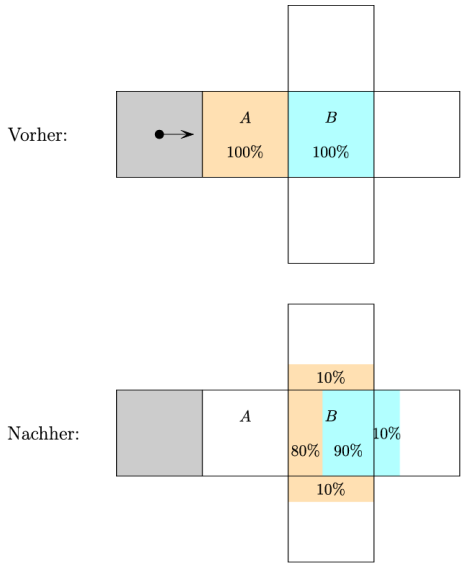


Abbildung 1.1: Grafik aus der Aufgabenstellung mit prozentualen Anteilen der Blätter auf den Quadraten A und B vor und nach einem Blasvorgang. Der Hausmeister steht auf dem grauen Quadrat und bläst in Pfeilrichtung.

Um unnötige Blasvorgänge und damit eine ineffiziente Strategie zu verhindern, sollten nur produktive Blasvorgänge durchgeführt werden. Als produktiv werten wir einen Blasvorgang dann, wenn sich nach seiner Durchführung eine möglichst große Anzahl an Blättern näher am Zielquadrat befindet oder sich global durch Kombination des Vorgangs mit anderen Blasvorgängen eine bessere Blattverteilung ergibt als zuvor.

Dafür ist es sinnvoll, möglichst weit entfernt vom Zielquadrat zu beginnen und Quadrate mit einer möglichst großen Blattanzahl priorisiert anzublasen. Auf diese Weise werden auch Blätter auf Nachbarquadraten in Richtung Zielquadrat befördert. Außerdem wird eine Abbruchbedingung für die Simulation benötigt: Es muss entschieden werden, ab wann im Verhältnis zur

Simulationsdauer genug Blätter auf dem Zielquadrat gesammelt wurden oder kein positiver Effekt mehr erwartet wird. Um möglichst produktive Züge zu ermöglichen, ist es außerdem möglich, dass der Hausmeister ein Gedächtnis besitzt. So kann er sich merken, welche Züge er bereits ausgeführt hat oder Züge in die Zukunft planen, ohne dass der Schulhof nach jedem Blasvorgang evaluiert werden muss. Welches Nicht-Rand-Quadrat als Zielquadrat festgelegt wird, macht in der Regel keinen nennenswerten Unterschied.

Nachdem im folgenden Abschnitt die Blas-Wirkung auf Rand- und Eckquadraten behandelt wurden, werden zwei verschiedene Ansätze für eine Blas-Strategie diskutiert.

Bei der ersten Strategie (Abschnitt 1.3) wird flexibel, rein aufgrund von Entfernung und Anzahl der Blätter auf einem Quadrat entschieden, wie geblasen wird. Die zweite Strategie basiert auf vordefinierten Phasen (Abschnitt 1.5), die nacheinander ausgeführt werden. Der Vorteil im Gegensatz zur ersten Strategie ist, dass hier auch Blasvorgänge ausgeführt werden, die nach der Metrik der ersten Variante kontraproduktiv wären, aber in Kombination mit anderen Zügen einen positiven Effekt haben.

1.2 Rand und Ecken

Die Auswirkungen eines Blasvorgangs sind für Quadrate am Rand und in den Ecken nicht vorgegeben. Da der Hausmeister den Schulhof nicht verlassen darf, muss man für diese Fälle also selbst Festlegungen treffen. In der Aufgabenstellung ist gefordert, möglichst viele Blätter auf einem Quadrat zu sammeln. Daher gehen wir der Einfachheit halber davon aus, dass der Schulhof von einer Wand umschlossen ist und die Blätter den Schulhof nicht verlassen können. Eine Implementierung, bei der die Blätter den Schulhof verlassen können, wird zwar durch die Aufgabenstellung nicht ausgeschlossen, hier aber aufgrund der zu erwartenden schlechteren Ergebnisse nicht diskutiert.

Es gibt drei Fälle, die betrachtet werden müssen, wie in Abbildung 1.2 dargestellt. Die grau schraffierten Quadrate befinden sich außerhalb des Schulhofes und sind durch eine Wand vom Schulhof getrennt. Dort können also keine Blätter landen. Die drei Fälle treten auch gedreht und gespiegelt auf.

1. Blätter werden direkt gegen eine *Wand* geblasen. In diesem Fall liegt das Quadrat B außerhalb des Schulhofes. Deswegen müssen die Blätter, die eigentlich auf dem Quadrat B, sowie links und rechts davon landen würden, auf andere Quadrate umgeleitet werden.
2. Blätter werden gegen eine *Ecke* geblasen. Wie in Fall 1 liegt das Quadrat B außerhalb des Schulhofes und die Blätter müssen abgelenkt werden. Die alternativen Quadrate sind hier noch stärker eingeschränkt.
3. Die Blätter werden an der *Seite* entlang einer Wand geblasen. In diesem Fall liegt das Quadrat B innerhalb des Schulhofes und es müssen nur die Blätter, die andernfalls links von B landen würden, umgeleitet werden. Im Gegensatz zu den anderen beiden Fällen liegen hier Blätter auf B, die dann auch weiter geblasen werden.

Um die Blas-Wirkung an diesen Stellen möglichst sinnvoll festzulegen, muss definiert werden, wie sich die Blätter realistischerweise verhalten, wenn sie gegen eine Wand stoßen. Siehe auch Abbildung 1.3.

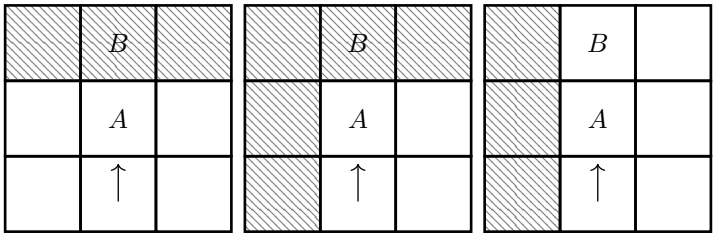


Abbildung 1.2: Die drei Randfälle, für die jeweils die Blas-Wirkung festgelegt werden muss. Von links nach rechts die Fälle Wand, Ecke und Seite. Schraffierte Quadrate liegen außerhalb des Schulhofes. Der Hausmeister steht immer auf dem Quadrat mit dem Pfeil.

1. Fall: Wand
Bei dem Fall, dass direkt gegen eine Wand geblasen wird, wäre es eine einfache Lösung festzulegen, dass die Blätter, die gegen die Wand stoßen, auf das Quadrat *A* zurückgeworfen werden. Da es sich aber nicht um einen abprallenden Ball handelt, sondern um Blätter, die vom Luftstrom getragen werden, wäre dies wenig realistisch. Viel mehr wird die Luft vom Laubbläser, die Blätter an der Wand in alle Richtungen verteilen. Um dies zu vereinfachen, wird festgelegt, dass die Blätter entweder nach links oder nach rechts mit jeweils 50% Wahrscheinlichkeit abgelenkt werden. In der unteren Reihe von Abbildung 1.3 bleiben die Blätter wie bisher mit 90% auf Quadrat *B* liegen. Die restlichen 10% verteilen sich gemäß der Regelerweiterung auf die beiden Quadrate rechts und links von *B*.

2. Fall: Ecke
In diesem Fall werden die meisten Blätter aufgrund der Ecke nach rechts abgelenkt. Es wird festgelegt, dass die Blätter mit 90% nach rechts weiterfliegen und mit 10% in der Ecke "hängen“ bleiben. In der zweiten Reihe werden die Blätter entsprechend von *B* mit 9% nach rechts abgelenkt.

3. Fall: Seite
Nur die ca. 10% der Blätter, die links von *A* landen würden, sind in diesem Fall betroffen. Weil diese links von der Wand blockiert werden, gehen wir davon aus, dass sie auf das Quadrat *B* zurückfliegen.

Diese Festlegungen werden in Abbildung 1.3 dargestellt. Die Prozentwerte der ersten Reihe beziehen sich auf den Anteil der Blätter, der von *A* aus weggeblasen wird. In der zweiten Reihe beziehen sich die prozentualen Anteile auf die Blätter von Quadrat *B*, da von einem Quadrat weiter unten geblasen wird.

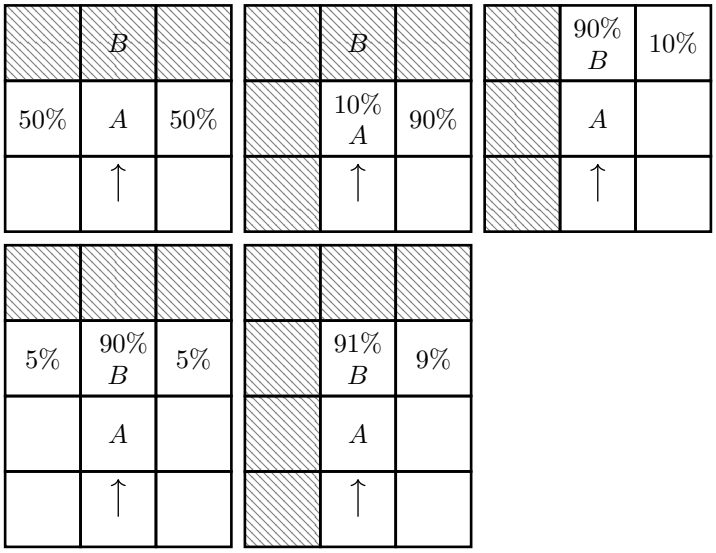


Abbildung 1.3: Blas-Wirkung in den drei Randfällen. Die Prozentwerte der oberen Reihe beziehen sich immer auf die 100% der Blätter von Quadrat *A*. Die Werte in der zweiten Reihe beziehen sich auf die 100% der Blätter von Quadrat *B*. Im dritten Fall gibt es nur eine Darstellung, da die Verteilung von Quadrat *B* der von Quadrat *A* entspricht. Der Hausmeister steht immer auf dem Quadrat mit dem Pfeil.

1.3 Strategie 1: Priorisierung der Felder

Bei dieser Strategie werden alle Quadrate einzeln mit einer Priorität p bewertet und davon abhängig entschieden, welches Quadrat als nächstes angeblasen wird. Hierfür wird für jedes Quadrat eine Priorität p berechnet. Diese ist das Produkt aus der Anzahl der Blätter b und der Distanz zum Zielquadrat d. Es gilt also $p = b \cdot d$. Ist dieser Wert hoch, wird aufgrund der Annahmen ein produktiver Blasvorgang erwartet und diese Quadrate sollen zuerst angeblasen werden. Haben zwei Quadrate den gleichen Wert, spielt die Reihenfolge keine Rolle. Zur Berechnung der Distanz d wird die Manhattan-Distanz verwendet, da Blätter nicht diagonal geblasen werden können.

In Abbildung 1.4 ist mit dem roten Pfeil die Luftlinie zwischen der Ecke und dem Mittelquadrat eingezeichnet, die blauen Pfeile stellen die Manhattan-Distanz dar. Die Manhattan-Distanz wird berechnet, indem der Betrag der Differenz von zwei Koordinatenpaaren a und b genommen und die so entstandenen Differenz-Werte für x und y summiert werden $\left(\sum |a-b|\right)$.

Abbildung 1.5 zeigt die so berechneten Prioritätswerte für einen quadratischen Schulhof, der initial auf jedem Quadrat 100 Blätter hat.

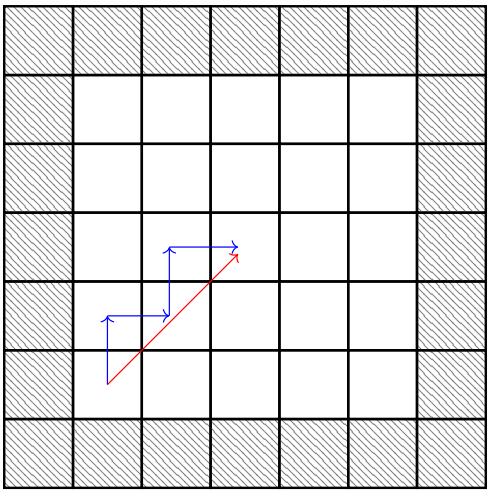


Abbildung 1.4: Verdeutlichung der Manhattan-Distanz. In rot ist die Luftlinie zu sehen, in blau die Manhattan-Distanz.

400	300	200	300	400
300	200	100	200	300
200	100	0	100	200
300	200	100	200	300
400	300	200	300	400

Abbildung 1.5: Prioritätswerte eines Schulhofs, der auf allen Quadraten initial 100 Blätter hat.

Für Blasvorgänge werden folgende Kriterien festgelegt:

- Es werden nur produktive Blasvorgänge durchgeführt. Hierzu wird vor jedem Blasvorgang die erwartete resultierende Blattverteilung berechnet und der Gesamtscore des Spielfeldes vor und nach dem Blasvorgang verglichen. Nur wenn der Gesamtscore steigt, handelt es sich um einen produktiven Blasvorgang und er wird durchgeführt.
- Blasvorgänge finden immer in Richtung des Zielquadrates statt.
- Bei Randquadraten die keine Ecken sind, wird in Richtung des Zielquadrates am Rand entlang geblasen.
- In Ecken wird hineingeblasen.
- Leere Quadrate (Priorität 0) werden nicht angeblasen.
- Das Zielquadrat hat immer die Priorität 0.

Ein Blasvorgang soll die Blätter immer einen Schritt näher an das Zielquadrat bringen. Es muss also in Richtung des Zielquadrates geblasen werden.

Um die Richtung für einen Blasvorgang auf ein Quadrat zu bestimmen, wird erst überprüft, ob die x-Koordinate der des Zielquadrates entspricht. Wenn nicht, muss in positive oder negative x-Richtung geblasen werden, um die Distanz zu verringern. Ist die x-Koordinate gleich der des Zielquadrates, wird nach der gleichen Logik in positive oder negative y-Richtung geblasen. Dies gilt auch für Randquadrate mit der Einschränkung, dass nur entweder x- oder y-Richtung möglich ist.

Bei Blasvorgängen in Ecken gibt es wegen der dortigen Randbehandlung einen positiven Effekt, wenn in die Ecke hineingeblasen wird. Deswegen wird hier als Sonderfall festgelegt, dass aus einem angrenzenden Quadrat in die Ecke hinein geblasen wird. Die Wirkung eines solchen Blasvorgangs ist in Abschnitt 1.2 beschrieben.

Bei allen möglichen Blasvorgängen für ein Quadrat wird die erwartete Blattverteilung des Blasvorgangs und die daraus resultierenden Prioritätswerte auf den beteiligten Quadraten berechnet und anhand des entstehenden Scores der veränderten Quadrate entschieden, ob der Blasvorgang produktiv wäre und somit geblasen wird oder nicht. Wurde für ein Quadrat ein Blasvorgang durchgeführt, werden die weiteren möglichen Blasvorgänge nicht geprüft und das Quadrat gilt als bearbeitet.

Es wäre nun theoretisch möglich, immer nur das Quadrat mit der höchsten Priorität auszuwählen und einen Blasvorgang in die Richtung des Zielquadrates durchzuführen, falls dieser produktiv ist. Hier kann es allerdings vorkommen, dass die Simulation vorzeitig abgebrochen wird, da es möglich sein kann, dass es für kein Quadrat mit der höchsten Priorität einen produktiven Zug gibt.

Außerdem benötigt diese Strategie sehr viel Rechenzeit, da nach jedem Blasvorgang die Priorität der Quadrate neu berechnet und das Quadrat mit der höchsten Priorität ermittelt werden muss. Um dies zu umgehen, müssten Ausnahmen für Quadrate mit geringerer Priorität geschaffen werden. Da diverse Ausnahmeregelungen vom Aufwand her vergleichbar mit dem Einsatz einer Prioritätswarteschlange sind, wird die Strategie entsprechend angepasst:

Nachdem die Priorität für alle Quadrate ermittelt wurde, werden diese in einer Prioritätswarteschlange absteigend geordnet. Die Quadrate werden dann entsprechend der festgelegten Reihenfolge bearbeitet. Nachdem alle Quadrate einmal angeblasen wurden, ist eine *Runde* der Simulation abgeschlossen. Die Prioritätswerte für alle Quadrate werden neu berechnet, und die nächste *Runde* startet.

Wir legen fest, dass die Summe aller Prioritätswerte im Laufe der Simulation sinken soll, um die globale Situation zu verbessern. Dafür wird die bisherige minimale Prioritätssumme gespeichert. Wird diese für eine festgelegte Zahl an Runden nicht überschritten, bricht die Simulation ab. Experimentell hat sich der Wert von 100 Runden als sinnvoll im Bezug auf Verbesserung der Blattverteilung in Abhängigkeit von der Laufzeit der Simulation herausgestellt.

Abbildung 1.6 zeigt das Ergebnis einer Runde des naiven Ansatzes. Die Blätter werden in der Mitte des Randes der jeweiligen Seite gesammelt und nicht in die Mitte befördert. Wenn sehr viele Runden durchgeführt werden, wird jeder produktive Blasvorgang entsprechend oft ausgeführt, wodurch die Blätter am Rand irgendwann in der Mitte landen. Für einen 5×5-Schulhof dauert dies etwa 300 Runden. Abbildung 1.7 zeigt das Ergebnis nach 335 Runden.

9	21	409	0	14
17	0	55	0	0
286	51	756	77	289
0	20	0	21	15
11	24	416	0	9

36	63	818	0	56
51	0	55	0	0
572	51	0	77	578
0	40	0	42	45
44	72	832	0	36

Abbildung 1.6: Links die Blattanzahl und rechts die Prioritätswerte auf einem 5×5-Schulhof, der mit 100 Blättern auf jedem Quadrat gestartet ist und auf dem dann eine *Runde* des naiven Ansatzes durchgeführt wurde.

0	0	42	0	0
0	0	215	0	0
41	270	1850	27	24
0	0	0	0	0
0	0	31	0	0

Abbildung 1.7: Ergebnis des naiven Ansatz auf einem 5×5-Schulhof mit initial 100 Blättern auf jedem Quadrat nach Runde 335.

1.4 Allgemeine Überlegungen für einen effektiveren Ansatz

Bei einem Blasvorgang werden die Blätter einzeln nach der Verteilung in Abbildung 1.1 bewegt. Wenn aber auf dem angeblasenen Quadrat schon eine große Anzahl an Blättern liegt, ist es möglich, dass durch einen Blasvorgang die Blätter nicht gesammelt, sondern weiter verteilt werden. Dargestellt ist dies in der Abbildung 1.8.

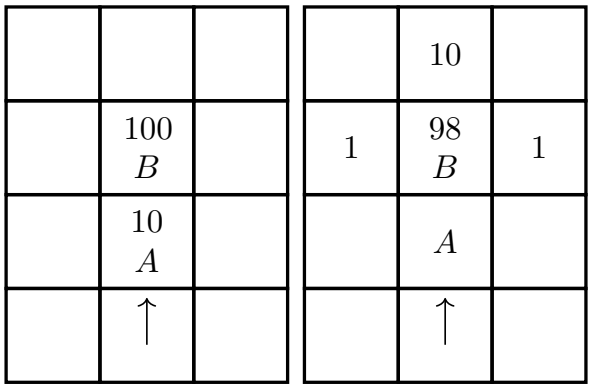


Abbildung 1.8: Vorher-Nachher-Beispiel eines kontraproduktiven Blasvorgangs. Auf dem Schulhof befinden sich insgesamt nur 110 Blätter. Auf Quadrat B liegen bereits 100 Blätter, und die 10 Blätter von Quadrat A sollen ebenfalls auf Quadrat B geblasen werden. Durch einen direkten Blasvorgang nimmt die Anzahl an Blättern auf Quadrat B ab.

Basierend auf dieser Beobachtung wird die Abbruchbedingung für direktes Blasen auf das Zielquadrat wie folgt formuliert: Wenn auf dem Zielquadrat mehr als $8/9$ aller Blätter liegen, erwarten wir keine produktiven direkten Blasvorgänge mehr. Außerdem ist jeder Blasvorgang kontraproduktiv, wenn 80% der Blattanzahl auf Quadrat A weniger ist als 10% der Blattanzahl von Quadrat B. Die Berechnung der Schwelle ist in Folgendem dargestellt. Für die Berechnung wird angenommen, dass alle Blätter des Schulhofes auf den Quadraten A und B liegen, um die Schranke nach oben abzuschätzen.

$$\frac{8}{10}A - \frac{1}{10}B = 0 \tag{1.1}$$
$$\Leftrightarrow \quad 8A - B = 0 \tag{1.2}$$
$$\Leftrightarrow \quad 8A = B \tag{1.3}$$
$$\tag{1.4}$$
$$A + B = 1 \tag{1.5}$$
$$\text{Einsetzen von oben} \Leftrightarrow \quad A + 8A = 1 \tag{1.6}$$
$$\Leftrightarrow \quad A = \frac{1}{9} \tag{1.7}$$
$$\Leftrightarrow \quad B = \frac{8}{9} \tag{1.8}$$

Aufgrund der Vorgaben werden einzelne Blätter bei einem Blasvorgang nicht geradeaus geblasen, sondern nach links und rechts abgelenkt. Durch das gezielte Freiräumen von Spalten, Zeilen und dem Rand soll dieser Effekt minimiert werden.

In Abbildung 1.9 ist eine Möglichkeit zum Freiräumen von Spalten gezeigt. Hierfür wird zunächst die mittlere Spalte durch dreimalige Blasvorgänge freigeräumt und anschließend die beiden benachbarten Spalten. Der Nachteil hierbei ist, dass die mittlere Spalte am Ende des beschriebenen Vorgangs weiterhin Blätter enthält. Dieser Vorgang muss also häufiger wiederholt werden, bis die mittlere Spalte tatsächlich leer ist.

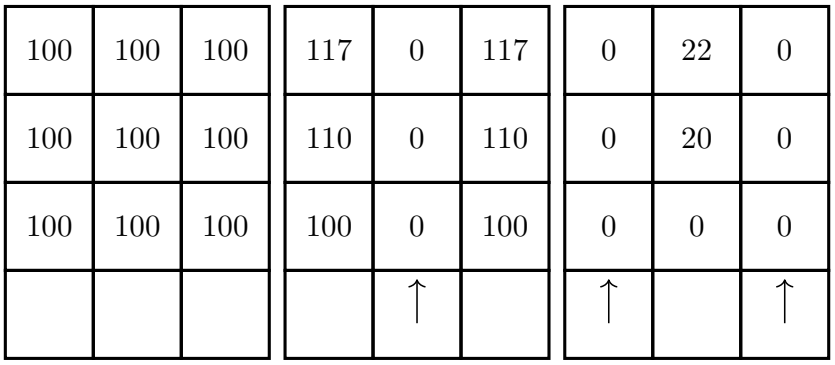


Abbildung 1.9: Beispiel für das Freiräumen von Spalten. Zunächst wird die mittlere Spalte freigeräumt, danach die beiden äußeren. An den Rändern der Abbildung geht der Schulhof weiter, die Quadrate sind nicht eingezeichnet. Die Abbildungen stellen den Zustand vor dem Blasvorgang und nach den jeweils markierten abgeschlossenen Blasvorgängen einer Spalte dar.

Ein Freiräumen Reihe für Reihe wird in Abbildung 1.10 dargestellt. Es wird mit der unteren Reihe begonnen und dann aufsteigend gearbeitet. Hier fällt auf, dass freigeräumte Quadrate tatsächlich frei bleiben.

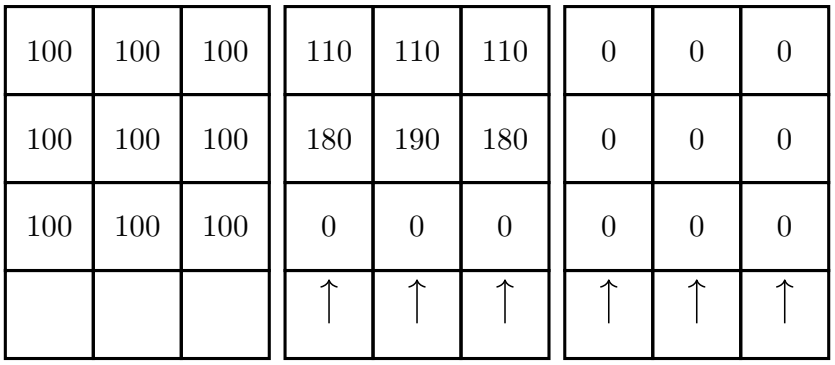


Abbildung 1.10: Beispiel für das Freiräumen von Reihen. Zunächst wird die unterste Reihe freigeräumt, danach aufsteigend die beiden oberen. An den Rändern der Abbildung geht der Schulhof weiter, die Quadrate sind nicht eingezeichnet. Die Abbildungen stellen den Zustand vor den Blasvorgängen, nach der ersten Reihe und nach allen drei Reihen dar.

Bei Randquadraten besteht die Schwierigkeit darin, dass Blätter dort nur durch Blasvorgänge am Rand entlang mit 10% Wahrscheinlichkeit in die Mitte geblasen werden (siehe Abbildung 1.3, 3. Fall). Ansonsten bewegen sie sich lediglich auf Randquadraten. Die Strategie zum Freiräumen der Randquadrate besteht also darin, möglichst viele Blätter auf einem Quadrat an der Seite des Schulhofes zu sammeln und anschließend mit Blasen entlang der Seite in die Mitte zu befördern. Zunächst werden die Blasvorgänge einmal im Uhrzeigersinn am Rand entlang durchgeführt, in Abbildung 1.11 ist das Ergebnis einer Umrundung des Schulhofes dargestellt.

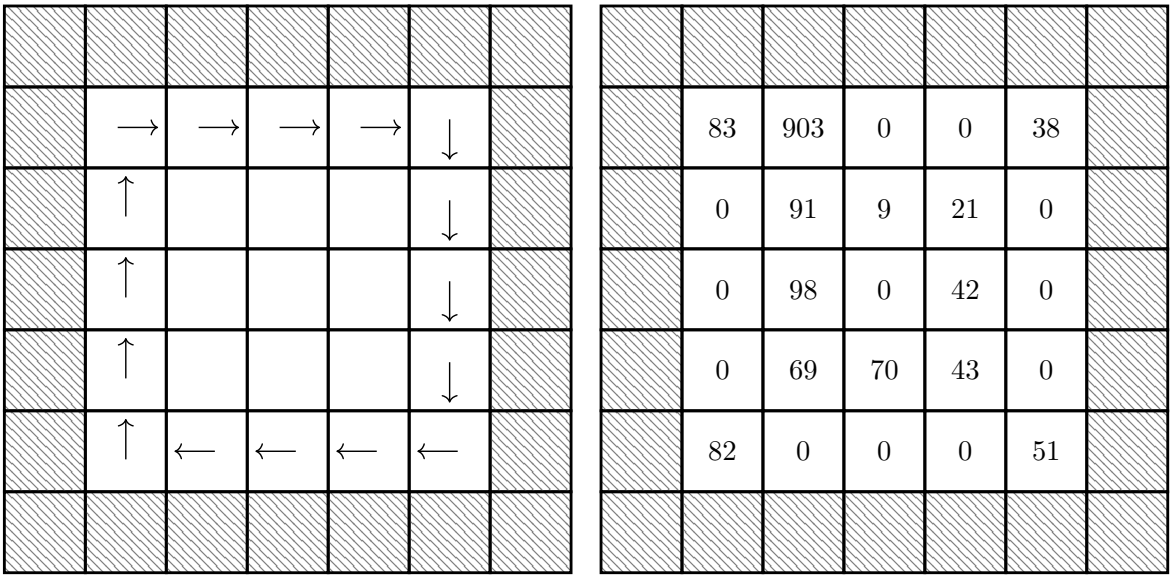


Abbildung 1.11: Muster, in dem der Hausmeister bläst, um Blätter in einem Randquadrat zu konzentrieren. Rechts ist der Zustand des Schulhofs, nachdem der Hausmeister einmal im Uhrzeigersinn den Rand abgegangen ist und auf jedem Quadrat einmal geblasen hat. Am Anfang waren auf jedem Randquadrat 100 Blätter.

Nach der ersten Umrundung startet der Hausmeister ein Quadrat weiter rechts als zuvor und umrundet erneut den Schulhof. So werden die meisten Blätter auf dem ersten Quadrat rechts neben der oberen linken Ecke gesammelt. Einige Blätter sammeln sich in den Ecken an und einige werden bereits Richtung Mitte befördert. Abschließend wird der angesammelte Blätterhaufen durch Blasen nach links und rechts entlang der oberen Reihe in die Mitte befördert. Damit ist das einmalige *Freiräumen des Randes* abgeschlossen.

Erwähnenswert ist, dass dieses Freiräumen des Randes bei Schulhöfen mit der Größe 3×3 nicht funktioniert, da mindestens ein Blasvorgang entlang eines Randes möglich sein muss, um Blätter in die Mitte des Schulhofes zu befördern (siehe Abbildung 1.12).

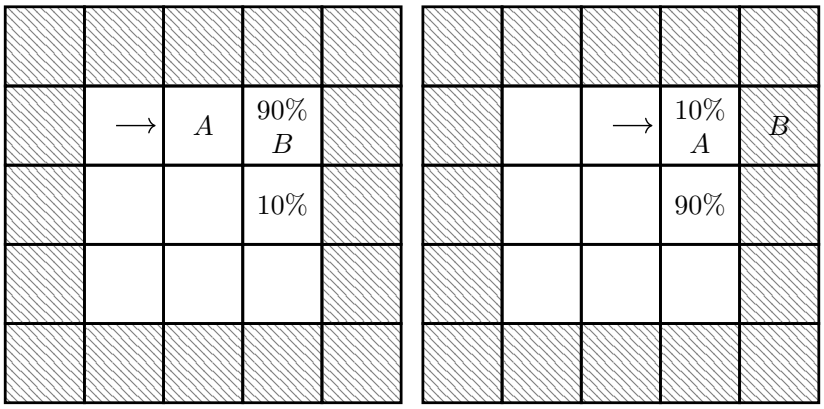


Abbildung 1.12: Blasvorgänge am Rand eines 3×3 Schulhofs. In diesem Falle ist es nicht möglich, Blätter vom Rand in die Mitte zu befördern.

Die letzte allgemeine Überlegung ist, dass die angegebenen Wahrscheinlichkeiten für die Blattverteilung nach einem Blasvorgang zwar vorliegen, die tatsächliche Blattverteilung in der Simu-

lation aber von dieser theoretischen Verteilung abweicht. Dies geschieht, da jedes Blatt einzeln und nicht der Haufen als Ganzes simuliert wird. Dieser Effekt verstärkt sich, je mehr Blasvorgänge durchgeführt werden. Daraus lässt sich schließen, dass sich nicht immer die gleiche Reihenfolge von aufeinander folgenden Abläufen definieren lässt, sondern einzelne Phasen mit festen Schritten so lange durchgeführt werden sollten, bis ein festgelegter Schwellwert erreicht ist.

1.5 Strategie 2: Phasenbasierte Strategie

Die Phasen-Strategie nutzt die eben gemachten Überlegungen und besteht aus drei *Phasen*. Zunächst wird der Rand entsprechend dem Muster in Abbildung 1.11 freigeräumt (*Phase 1*). Anschließend werden die Blätter in der Mitte zusammen geblasen (*Phase 2*), dazu wird eine Kombination aus Freiräumen von Spalten und Reihen verwendet. In der abschließenden *Phase 3* werden angrenzende Blätter mit Abstand auf das Zielquadrat geblasen.

Für Schulhöfe größer als 5×5 kann das mittlere Quadrat des Schulhofs als Zielquadrat festgelegt werden. Bei Schulhöfen der Größe 5×5 sind die Optionen aber durch den nur 2 Quadrate entfernten Rand begrenzt. Deswegen wählen wir als Zielquadrat das Quadrat oberhalb der Mitte. Die Mitte auf einem Schulhof der Kantenlänge n ist definiert als $\lfloor n/2 \rfloor + 1$.

Phase 1: Rand freiräumen

Der Rand wird mit Hilfe der in Abschnitt 1.4 beschriebenen Strategie freigeräumt. Dieses Freiräumen wird so häufig wiederholt, bis sich weniger als 10% der ursprünglichen Blätter auf den Randquadraten befinden. Dieser Schwellwert von 10% lässt sich auch verringern, jedoch werden abhängig von den Zufallsergebnissen der Blattverteilung sehr viele Blasvorgänge benötigt, um zum Beispiel auf 0% zu kommen. Es ist aber zu erwarten, dass auch das irgendwann passieren könnte.

Phase 2.1: Blätter zusammenblasen

Diese Version von Phase 2 wird nur beim ersten Simulationsschritt ausgeführt. Die Reihen werden von oben bis zur Zielreihe und von unten bis zur Zielreihe freigeräumt, sodass die meisten Blätter in der mittleren Reihe gesammelt werden. Anschließend werden die Blätter auf dem Zielquadrat gesammelt, indem die Zielreihe von links und rechts in Richtung Mitte geblasen wird (siehe Abbildung 1.13).

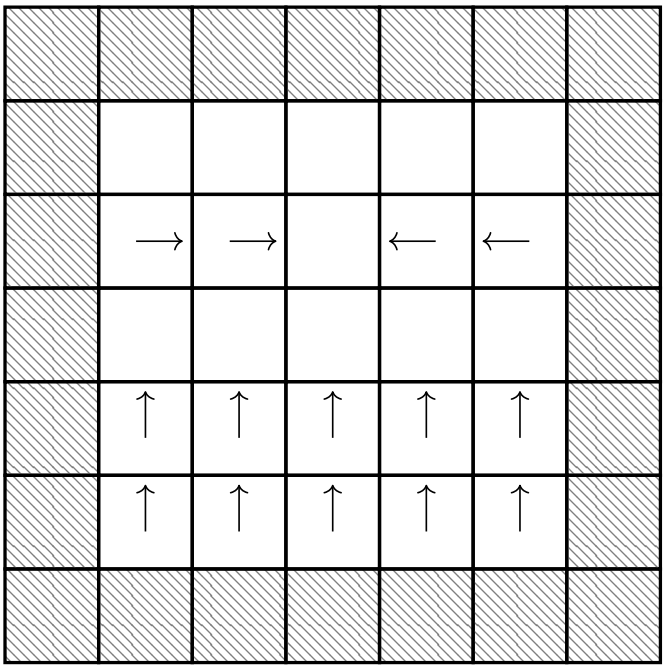
Phase 2.2: Blätter zusammenblasen

Diese Version der Phase 2 ähnelt der Phase 2.1, nur werden die Blätter auf dem Quadrat direkt unter dem Zielquadrat konzentriert. Der linke Teil von Abbildung 1.14 zeigt diesen Vorgang. Dafür wird beim Blasen von oben die mittlere Reihe ausgelassen, um nicht auf das Zielquadrat zu blasen. Bei Schulhöfen größer als 5×5 werden auch noch die Blätter der Quadrate in der mittleren Spalte oberhalb des Ziels, aber nicht die des Randquadrats nach rechts geblasen. Das ist nötig, weil sich dort Blätter sammeln, die ansonsten liegen bleiben würden. Bei Abbildung 1.13 sammeln sich diese oben am Rand und werden deswegen dort eingesammelt.

Phase 3: Blasen mit Abstand

Es werden Blasvorgänge mit einem Quadrat Abstand durchgeführt, um jeweils 10% der Blätter vom Quadrat vor dem Zielquadrat (V) ohne Seitenverluste auf das Zielquadrat zu befördern. Dafür wird mit einem Quadrat Abstand auf V in Richtung Zielquadrat geblasen. Diese Phase tritt erst ein, wenn keine direkten produktiven Blasvorgänge mehr möglich sind, da bereits $8/9$ aller Blätter auf dem Zielquadrat liegen. Wie beim Freiräumen des Randes wird das Blasen mit Abstand wiederholt, bis 90% der Blätter von V auf das Zielquadrat geblasen wurden.

Die phasenbasierte Strategie beginnt mit Phase 1 und führt nach Erreichen der gewünschten Genauigkeit Phase 2.1 aus. Abbildung 1.13 zeigt rechts ein Beispiel für das Ergebnis der beiden Phasen für den ersten Simulationsschritt. Links sind die Quadrate markiert, die vom Hausmeister angeblasen werden, um nach dem Freiräumen des Randes die Blätter auf dem Zielquadrat zu sammeln. Alle Quadrate haben am Anfang 100 Blätter.

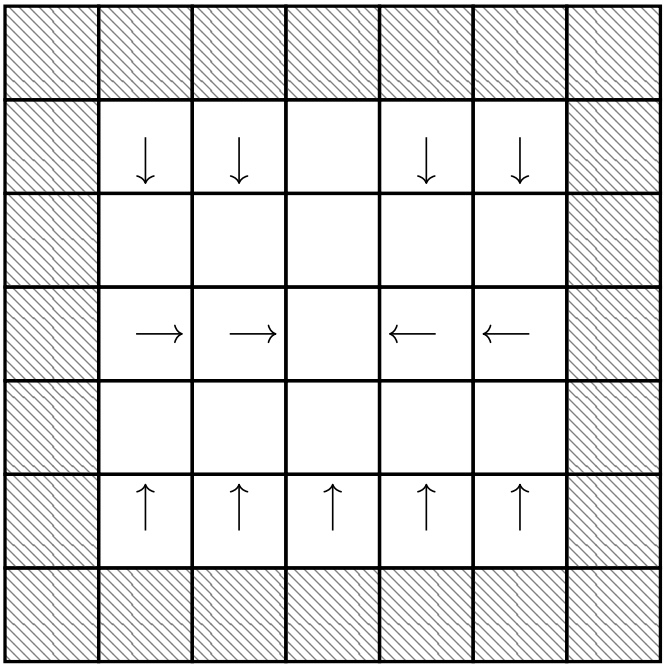


7	197	202	17	7
41	0	1724	102	51
0	0	139	0	0
0	0	0	0	0
11	0	0	0	2

Abbildung 1.13: Links ist zu sehen, wo der Hausmeister sich jeweils hinstellt und in welche Richtung er bläst, um die Blätter in Reihen in Richtung Zielquadrat zu bewegen. Rechts ist das Ergebnis für einen Schulhof, der mit 100 Blättern auf jedem Quadrat startet.

Nun wird mithilfe der Abbruchbedingung geprüft, ob noch produktive Züge möglich sind: Wenn bereits $8/9$ aller Blätter auf dem Zielquadrat liegen, wird zu Phase 3 übergegangen. Wenn nicht (wie bei dem beispielhaft gewählten 5×5 Schulhof, wo gerade mal 70% der Blätter auf dem Zielquadrat liegen), wird die Simulation mit Phase 1 und Phase 2.2 wiederholt.

Damit überhaupt noch produktive Züge auf das Zielquadrat möglich sind, müssen möglichst viele Blätter auf einmal direkt auf das Zielquadrat geblasen werden. Um dies zu erreichen, wird der Rand wie gewohnt bis zu einer bestimmten Genauigkeit freigeräumt und die Blätter anschließend in Phase 2.2 auf dem Quadrat direkt unter dem Zielquadrat konzentriert.



1	16	0	0	0
2	9	2114	12	1
11	0	202	14	9
0	0	106	0	0
2	0	0	0	1

Abbildung 1.14: Links die angepasste Strategie zum Sammeln der Blätter auf dem Quadrat unterhalb des Zielquadrats, ohne dabei die Blätter auf dem Zielquadrat wegzublasen. Rechts das Ergebnis der gesamten Phase 2.2 aufbauend auf dem Beispiel in Abbildung 1.13.

Nun sollte eine große Menge Blätter auf dem Quadrat direkt unterhalb des Zielquadrates gesammelt worden sein. Es wird geprüft, ob ein Blasvorgang, der diese gesammelten Blätter auf das Zielquadrat befördert, produktiv ist. Ist dies der Fall, wird der Blasvorgang ausgeführt und anschließend die Abbruchbedingung für den Algorithmus überprüft. Die Aneinanderreihung von Phase 1 und Phase 2.2 wird wiederholt,bis die Abbruchbedingung von $8/9$ aller Blätter auf dem Zielquadrat erreicht ist. In Abbildung 1.14 liegen auf dem Zielquadrat mehr als 8 mal so viele Blätter wie auf dem Quadrat darunter. Die Blätter in der Mitte können also nicht direkt auf das Ziel geblasen werden und Phase 3 wird angewendet: Es werden wiederholt aus der Entfernung 10% der Blätter vom Nachbarquadrat auf das Zielquadrat geblasen, bis 90% der Blätter des Nachbarquadrats auf dem Zielquadrat liegen. Abbildung 1.15 zeigt das Ergebnis dieser letzten Phase.

1	16	0	0	0
2	9	2379	12	1
11	10	20	27	9
0	0	0	0	0
2	0	0	0	1

Abbildung 1.15: Zeigt das Ergebnis nach Abschluss der 3. Phase. Die Blätter wurden mit Abstand auf das Zielquadrat geblasen. Die Darstellung baut auf dem Beispiel in Abbildung 1.14 auf.

1.6 Simulation

Für die Aufgabe soll ein Programm geschrieben werden, welches Blasvorgänge simulieren kann. Das bedeutet, es muss der Schulhof mit der Anzahl der Blätter auf jedem Quadrat gespeichert werden und bei Durchführung eines Blasvorgangs entsprechend aktualisiert werden. Wahrscheinlichkeiten müssen auf jedes Blatt angewendet werden, und wenn ein Blasvorgang Richtung Wand oder entlang einer Wand geschieht, müssen die Festlegungen für Rand- und Eckquadrate angewandt werden. Um mehrere Strategien direkt miteinander vergleichen zu können, ist es hilfreich einen seed für den Zufallsgenerator einzubauen. So sind die Ergebnisse reproduzierbar. Um eine generelle Aussage über die Anzahl der Blätter zu treffen, sollten Ergebnisse aus Beispielen der gleichen Größe mit zufälligen seeds zusammengefasst werden.

Datenstruktur 2D-Array Bei einem quadratischen Schulhof bietet es sich an, ein 2D-Array mit Integer-Werten zu verwenden. Dieser wird dann bei einem Blasvorgang aktualisiert. Um zu verhindern, dass '`index out of range`'-Fehler geworfen werden, kann das Array einen Rand von einem Quadrat bekommen, sodass es um zwei Quadrate breiter und höher ist als der eigentliche Schulhof. Das ist auch sinnvoll, wenn Blasvorgänge an Rändern und Ecken Blätter aus dem Schulhof entweichen lassen.

Datenstruktur Graph Vorstellbar wäre auch ein Ansatz, der jedes Quadrat als Knoten eines Graphen betrachtet. Jedes Quadrat hätte dann eine Verbindung zu den vier Nachbarn. Die Knoten am Rand haben entsprechend nur zwei oder drei Nachbarn. Dies könnte mit einer Adjazenzliste realisiert werden, oder aber es wird eine Klasse Knoten erstellt, deren Objekte die Knoten darstellen. Eine Klasse Randknoten könnte von der Klasse Knoten erben. Diese Datenstruktur hat den Vorteil, dass alle Nachbarn eines Quadrats bekannt sind und objektorientiert Funktionen für das Blasen verwendet werden können, die die entsprechenden Nachbarn ansprechen. Weil die Simulation in der Lage sein sollte, einen Blasvorgang für ein bestimmtes Quadrat zu simulieren, muss dieses allerdings zum Beispiel durch Koordinaten identifizierbar sein. Wenn mit Koordinaten gearbeitet wird, macht das Verwenden einer solchen Struktur anstatt eines 2D-Arrays wenig Sinn, weil trotzdem die Objekte nach Koordinaten identifizierbar

sein müssen. Bezogen auf die Strategie kann es allerdings Vorteile bringen, wenn Prioritäten und Blatt-Anzahlen in diesem Netz geteilt werden können.

Ob ein objektorientierter Ansatz sinnvoll ist, hängt von der Programmiersprache ab. In Python beispielsweise ist es deutlich effizienter, eine `numpy array`- Datenstruktur zu wählen.

Rand und Ecken Blasvorgänge, die am Rand liegen, müssen entweder mit den entsprechenden Festlegungen durchgeführt werden oder aber sind ungültig. Das gilt offensichtlich, wenn der Hausmeister außerhalb des Schulhofes stehen würde. Aber auch, wenn bezogen auf Abbildung 1.1 das Quadrat *A* außerhalb des Schulhofes liegt, ist der Zug ungültig.

Um festzustellen, wann ein Quadrat außerhalb ist, müssen die Indizes für beide Koordinaten größer oder gleich 0, sowie kleiner als die Länge und Breite des Schulhofes sein. Wenn das 2D-Array größer ist als der Schulhof, müssen die Koordinaten entsprechend korrigiert werden. Eine Alternative dazu wäre, ein zusätzliches 2D-Array anzulegen, welches für jedes Quadrat speichert, ob es innerhalb des Schulhofs liegt oder nicht. Damit könnten auch kompliziertere Schulhofformen behandelt werden.

Setup Die Simulation des Blasvorgangs wird in dieser Simulation in zwei Abschnitte geteilt. Zuerst gibt es das Setup, welches die Blätter-Anzahl auf den Quadraten *A* und *B* (Abbildung 1.1) auf 0 setzt und sich die Anzahlen zwischenspeichert. Auch hier muss die Randbehandlung Fehler verhindern. Danach folgt die Verteilung der Blätter.

Verteilung der Blätter Die Blätter von den Quadraten *A* und *B* müssen bei jedem Blasvorgang verteilt werden. Dafür wird für jedes Blatt ausgewürfelt, welche der drei bzw. zwei Optionen zutreffen werden. Die Randbehandlung wird dann für die Quadrate um *B* herum und für *B* selbst durchgeführt. Welches Quadrat wie viele Blätter erhält, wird bereits vor der Randbehandlung festgelegt. Die Randbehandlung verhindert dann, dass Blätter auf einem Quadrat außerhalb des Schulhofes landen und verteilt diese Blätter neu.

1.7 Vergleich der Laufzeit

Die Aufgabe gibt keine Kriterien vor, was die Anzahl der Blasvorgänge und damit verbunden die Gesamtsimulationsdauer angeht. Eine Lösung, die unverhältnismäßig lange benötigt, um die der Simulation zu Grunde liegende Strategie zu berechnen und durchzuführen, sollte aber selbstverständlich vermieden werden.

Die Laufzeit für jede *Runde* des naiven Ansatzes beträgt mit Prioritätswarteschlange $\mathcal{O}(n^2)$, wobei n durch die Anzahl der Seitenquadrate eines Schulhofes gegeben ist. Dies ergibt sich, da einerseits für jedes Quadrat die Prioritätswerte ausgerechnet und andererseits die meisten Quadrate angeblasen werden. Für das Auswählen der Blasrichtung wird jeweils nur eine konstante Menge von Operationen durchgeführt, weil nur eine feste Anzahl an Quadraten in der Nähe dafür relevant sind (maximal 4). Das Gleiche gilt für die Berechnung der erwarteten Blattverteilung und dem daraus resultierenden Score, um zu entscheiden, ob der Zug tatsächlich ausgeführt werden sollte.

Bei dem phasenbasierten Ansatz muss die Laufzeit der verschiedenen Phasen unterschieden werden.

Die Phase 1 besteht aus dem zweifachen Sammeln der Blätter vom gesamten Rand in der oberen Reihe. Es werden also alle vier Seiten der Länge n zweimal angeblasen, was in $\mathcal{O}(8n)$ bzw. $\mathcal{O}(n)$ entspricht. Danach werden die Blätter in der oberen Reihe abwechselnd hin und her geblasen, um Blätter vom Rand wegzubefördern. Dabei bleiben immer 90% der Blätter am Rand. Der Prozess wird abgebrochen, wenn sich nur noch 10% der ursprünglichen Blätter am Rand befinden. Wir erwarten, dass das unabhängig von der Größe des Schulhofes nach 20˜30 Blasvorgängen der Fall ist, weil $0.9^{20} = 0.098$. Durch die Blätter, die in den Ecken hängen bleiben, werden aber meistens mehr als 20 Blasvorgänge benötigt. Damit hat das Randfreiräumen bei einem Schulhof mit Seitenlänge n insgesamt eine Laufzeit von $\mathcal{O}(4n+30) = \mathcal{O}(n)$.

Für Phase 2.1 und Phase 2.2, das Sammeln der Blätter in der Mitte, ergibt sich eine Laufzeit von $\mathcal{O}(n^2)$, weil es höchstens zwei Reihen und eine Spalte gibt, von denen aus nicht geblasen werden darf. Auf alle anderen Quadraten muss sich der Hausmeister stellen und in eine vorgegebene Richtung blasen, um die Blätter zu sammeln. Ob die Blätter in der Mitte oder in einem Quadrat oberhalb der Mitte gesammelt werden, macht nur einen konstanten Unterschied. Nach $\mathcal{O}$-Notation zählt das also als gleiche Laufzeit und Phasen 2.1 und 2.2 liegen beide insgesamt bei $\mathcal{O}(n^2)$.

Phase 3 besteht lediglich aus einen Blasvorgang mit Abstand in eine vorgegebene Richtung und hat daher eine Laufzeit von $\mathcal{O}(1)$. Diese Phase wird unabhängig von der Größe des Schulhofes abgebrochen, wenn sich nur noch 10% der ursprünglichen Blätter auf dem Quadrat vor dem Zielquadrat befinden. Dies kann, analog zum Randfreiräumen aufgrund von $0.9^{20} = 0.098$ mit einer konstanten Laufzeit von maximal $\mathcal{O}(30)$ abgeschätzt werden, was insgesamt einer konstanten Laufzeit, also $\mathcal{O}(1)$ entspricht.

1.8 Vergleich der Qualität verschiedener Ansätze

Der phasenbasierte Ansatz mit den genannten Schwellwerten für Phase 1 und 3 führt bei 100 Blättern pro Quadrat und einem Schulhof der Größe 5×5 dazu, dass 95.6% aller Blätter auf einem Quadrat gesammelt werden. Bei einem Schulhof der Größe 6×6 sind es 93.8% und bei einem Schulhof der Größe 10×10 sind es 95.1%. Diese Zahlen wurden über 100 Beispiele der gleichen Größe ohne festen seed gemittelt. Wenn der Schwellwert für verbleibende Blätter am Rand auf 0% gesetzt werden würde, würde die Simulationszeit entsprechend steigen. Gleiches gilt auch für die Schwelle in Phase 3. Dafür könnten auf diese Weise bis zu 100% der Blätter auf dem Zielquadrat gesammelt werden.

Beim naiven Ansatz verhindern vor allem die Blätter am Rand, dass ein vergleichbar gutes Ergebnis erzielt wird, da diese quasi nie entgegen der Priorität geblasen werden. Darüber hinaus kann dieser Ansatz die Blätter nur in der Mitte und nicht auf dem Quadrat oberhalb der Mitte sammeln. Liegen dort bereits viele Blätter, führt das zu kontraproduktiven Erwartungen der Blasvorgänge, und diese werden deswegen nicht durchgeführt. So liegen zwar viele Blätter im mittleren Bereich des Schulhofes, aber werden nicht auf einem Quadrat zusammengeführt. Im Gegensatz dazu sammelt der phasenbasierte Ansatz erst möglichst viele Blätter auf einem Quadrat oberhalb des Zielquadrates, um diese dann gesammelt auf das Zielquadrat zu blasen und anschließend noch in Phase 3 mit Abstand zu sammeln. Durch diesen Unterschied erwarten wir beim naiven Ansatz selbst ohne Einbeziehung des Randes, dass der Schwellwert von $8/9$ nicht erreicht wird. Dieser Ansatz könnte durch eine zusätzliche Verwendung von Blasvorgängen mit einem Quadrat Abstand wie in Phase 3 verbessert werden, in diesem Falle würde aber auch hier die Simulationsdauer deutlich ansteigen.

1.9 Beispiele

Phasen-Ansatz: Größe 5×5

Erster und zweiter Schritt

7	197	202	17	7
41	0	1724	102	51
0	0	139	0	0
0	0	0	0	0
11	0	0	0	2

1	16	0	0	0
2	9	2114	12	1
11	0	202	14	9
0	0	106	0	0
2	0	0	0	1

End-Schritt: 95.2% der Blätter sind auf dem Zielquadrat.

1	16	0	0	0
2	9	2379	12	1
11	10	20	27	9
0	0	0	0	0
2	0	0	0	1

Phasen-Ansatz: Größe 6×6

Erster und zweiter Schritt

9	0	121	0	0	4
3	52	139	258	33	7
66	0	0	2415	128	63
0	0	67	204	0	0
0	0	0	0	0	0
13	0	0	0	0	18

2	0	2	0	33	1
0	0	0	0	0	0
0	2	17	2998	31	5
1	0	0	276	23	48
0	0	4	156	0	0
1	0	0	0	0	0

End-Schritt: 93.7% der Blätter sind auf dem Zielquadrat.

2	0	2	0	33	1
0	0	0	0	0	0
0	2	17	3373	31	5
1	0	17	27	36	48
0	0	4	0	0	0
1	0	0	0	0	0

Phasen-Ansatz: Größe 10×10

Erster Schritt

8	0	0	0	120	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
10	60	179	273	337	702	226	164	67	11
261	0	0	0	0	5580	305	0	0	204
0	0	122	194	250	601	193	96	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	12

Zweiter Schritt

0	0	0	0	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	4	0	0	0	0
1	2	1	7	18	8287	96	9	1	0
4	0	0	0	0	914	97	0	0	2
0	0	0	1	1	501	19	3	0	0
0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	22	0	0	0	0
1	0	0	0	0	0	0	0	0	1

End-Schritt: 95.2% der Blätter sind auf dem Zielquadrat.

0	0	0	0	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	4	0	0	0	0
1	2	1	7	18	9523	96	9	1	0
4	0	0	0	42	91	143	0	0	2
0	0	0	1	1	0	19	3	0	0
0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	22	0	0	0	0
1	0	0	0	0	0	0	0	0	1

Prioritäten-Strategie: Größe 5×5

Runde 1 und Runde 335: 74% der Blätter sind auf dem Zielquadrat.

9	21	409	0	14
17	0	55	0	0
286	51	756	77	289
0	20	0	21	15
11	24	416	0	9

0	0	42	0	0
0	0	215	0	0
41	270	1850	27	24
0	0	0	0	0
0	0	31	0	0

Nach Runde 335 bricht das Programm ab.

Prioritäten-Strategie: Größe 6×6

Runde 1 und Runde 225: 76.3% der Blätter sind auf dem Zielquadrat.

11	0	0	487	27	10
0	0	0	0	0	0
33	34	78	145	37	0
367	16	0	1244	75	362
0	0	0	119	0	22
5	0	37	475	0	16

0	0	0	47	0	0
0	0	0	0	0	0
0	0	0	312	0	0
30	0	322	2747	39	55
0	0	0	0	0	0
0	0	0	48	0	0

Nach Runde 325 bricht das Programm ab.

Prioritäten-Strategie: Größe 10×10

Runde 1

11	0	0	0	59	723	0	0	0	10
0	0	0	0	0	0	0	0	0	0
0	0	0	0	23	0	21	0	15	6
23	18	18	0	17	0	0	26	15	19
39	70	177	0	0	418	134	81	34	27
692	32	54	98	0	3931	240	0	0	713
54	20	16	259	176	475	173	134	0	50
6	13	13	13	14	0	17	11	9	16
0	0	0	0	0	0	0	0	0	0
8	0	0	0	39	725	0	0	0	15

Runde 359: 79.1% der Blätter sind auf dem Zielquadrat.

0	0	0	0	0	39	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
31	1	0	1	975	7906	90	0	0	49
6	0	0	0	0	871	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	25	2	0	0	0

Nach Runde 359 bricht das Programm ab.

1.10 Bewertungskriterien

Die aufgabenspezifischen Bewertungskriterien werden hier erläutert.

1. Lösungsweg

(1) *Problem adäquat modelliert*: Jede sinnvolle Modellierung des Schulhofes ist erlaubt. Es muss möglich sein zu prüfen, ob ein Feld ein Randfeld ist oder sogar außerhalb des Schulhofes liegt. Wie die Aufgabenstellung andeutet, kann man die Wahrscheinlichkeit für jedes Blatt einzeln betrachten. Es ist aber auch in Ordnung, die Blattverteilung über eine Zufallsverteilung zu modellieren, wenn diese nachvollziehbar hergeleitet wird. Nur ganzzahlige positive Zahlenwerte sind als Blattanzahl auf den Quadraten erlaubt.

(2) *Behandlung von Randfeldern*: Die Blas-Wirkung für jede Art von Randfeldern muss sinnvoll und konsistent festgelegt werden. Die gleichen Regeln wie für normale Felder zu verwenden oder die Aufgabe durch triviale Rand-Regeln zu vereinfachen führt zu 4 Punkten Abzug, mit guter Begründung zu 2 Punkten Abzug.

(3) *Sinnvolles Abbruchkriterium*: Es muss eine gut begründete Möglichkeit geben zu entscheiden, ob und wann die Simulation endet und der Hausmeister mit dem Besen weiter arbeitet.

(4) *Verfahren nicht unnötig ineffizient*: Gleich welche Strategie der Hausmeister insgesamt verfolgt, sollten die Blasvorgänge die Situation global verbessern. Insbesondere sollten keine unproduktiven Züge ausgeführt werden, es sei denn, dies wird gut begründet.

(5) *Laufzeit des Verfahrens in Ordnung*: Die Laufzeit der verwendeten Strategie sollte angemessen sein. Je nach Ansatz kann hier betrachtet werden

- ... das Finden der nächsten Platzierung des Hausmeisters und der besten Blasrichtung, falls die Strategie jeden Zug einzeln berechnet. Da dies insgesamt aufwändig ist, sollte $\mathcal{O}(n)$ pro Zug eher nicht überschritten werden.
- ... die Berechnung einer Runde über alle Felder (wie beim naiven Ansatz). Hier ist $\mathcal{O}(n^2)$ eine obere Grenze.
- ... der Durchlauf einer einzelnen Phase der Gesamtstrategie.

(6) *Verfahren mit korrekten Ergebnissen*: Der Großteil der Blätter muss am Ende auf dem vorgegebenen Zielfeld liegen, welches nicht am Rand ist.

(7) *Strategie liefert gute Ergebnisse*: Die Qualität der Ergebnisse hängt zuvorderst von der Qualität der Strategie ab. Wird die Strategie so gewählt oder führen andere Elemente des gewählten Verfahrens dazu, dass auch bei längerer Dauer der Simulation der Anteil der nicht auf dem Zielfeld versammelten Blätter eher hoch bleibt, ist das negativ zu bewerten. Besonders ungeeignet sind Ansätze, bei denen es sogar grundsätzlich nicht möglich ist, manche Nicht-Zielfelder leer zu blasen, und die dem Hausmeister deswegen besonders viel Kehr-Arbeit überlassen.
Am anderen Ende des Spektrums stehen Strategien, die besonders schnell zu einer starken Befüllung des Zielfeldes und zur vollständigen Entleerung anderer Felder führen. Besonders gute Ansätze können mit Pluspunkten belohnt werden. Keine Pluspunkte gibt es, wenn diese guten Ergebnisse nur auf Grund von trivialen Randbedingungen erzeugt werden.

(8) *Insgesamt gute Ergebnisse erzielt*: Die Qualität der letztlich in der Einsendung präsentierten Ergebnisse hängen aber auch davon ab, wie die Simulation durchgeführt wurde. Auch bei guter Strategie können ein allzu früher Abbruch der Simulation oder vielleicht

auch die Wahl von Simulationsparametern dazu führen, dass die effektiv erzielten Ergebnisse nicht so gut sind wie möglich.

2. Theoretische Analyse

(1) *Verfahren/Qualität insgesamt gut begründet*: Es muss sinnvoll begründet werden, welche Strategie für die Randbehandlung gewählt wurde, um reale Bedingungen möglichst gut abzubilden. Auch muss erläutert werden, warum die dokumentierte Hausmeisterstrategie gewählt wurde, um möglichst viele Blätter auf einem nicht-Randquadrat zu sammeln. Die Arbeit mit Wahrscheinlichkeiten für die Verteilung der einzelnen Blätter und deren Auswirkungen im Bezug auf die Simulation müssen berücksichtigt werden. Schwächen des Verfahren müssen erkannt und diskutiert werden.

(2) *Behandlung von Spezialschulhöfen*: Wenn aufgrund der gewählten Randbehandlung oder der Strategie Ergebnisse auf kleinen (4×4 oder kleiner) oder deutlich größeren Schulhöfen nicht möglich sind, muss dies erkannt und entsprechend begründet werden; ansonsten gibt es Punktabzug. Werden Ausnahmefälle oder alternative Strategien für diese Schulhofgrößen diskutiert, kann ein Pluspunkt gegeben werden.

(3) *Gute Überlegungen zur Laufzeit des Verfahrens*: Die Laufzeit einzelner Elemente der Strategie (vgl. Kriterium 5) sollten nicht zwingend formal, aber korrekt und nachvollziehbar charakterisiert werden. Außerdem sollten Überlegungen zur Simulationsdauer aufgestellt werden. Hierbei muss insbesondere das Verhältnis der Simulationsdauer zur Genauigkeit der Lösung abgewogen und begründet werden.

3. Dokumentation

(3) *Vorgegebene Beispiele dokumentiert*: Es müssen Schulhöfe in mindestens drei verschiedenen Größen aufgeführt werden.

(5) *Ergebnisse nachvollziehbar dargestellt*: Je nach Verfahren sind dafür Zwischenstände nach einer Zugabfolge oder nach einzelnen Zügen nötig. Mindestens sollte eine 2D-Darstellung des Schulhofes im Endzustand, die Blattanzahl auf den einzelnen Quadraten und die Anzahl der benötigten Züge dokumentiert werden.

Aufgabe 2: Stilvolle Päckchen

Jedes einzelne Kleidungsstück hat zwei Attribute: die Kleidungssorte (Schuhe, Hose, Jacke ...), abgekürzt mit **S**orte oder *S*, und die Stilrichtung (leger, sportlich, elegant ...), abgekürzt mit **R**ichtung oder *R*. In jeder Box müssen alle Kleidungsstücke miteinander kombinierbar sein. In der folgenden Lösung wird angenommen, dass dies auch für Kleidungsstücke der gleichen Sorte gelten muss. So müssen in einer Box auch zwei Paar Schuhe eine zueinander passende Stilrichtung haben, auch wenn in der Regel nur ein Paar Schuhe gleichzeitig getragen wird. Dies lässt eine Vereinfachung des Algorithmus zu, da für jede Box nur eine Liste an passenden Stilrichtungen gespeichert werden muss, statt einer Liste für jede enthaltene Sorte.

Es gibt mehrere Ansätze, die für diese Aufgabe gute Ergebnisse erzielen können. Ein Algorithmus, welcher immer eine optimale Lösung findet, wird nur bei sehr einfachen Beispieldateien umsetzbar sein. Um zu garantieren, dass eine Lösung optimal ist, müssen alle möglichen Verteilungen der Kleidungstücke auf Boxen untersucht werden. Mit S Sorten und R Richtungen können maximal R^S Boxen mit genau einem Kleidungsstück pro Sorte erzeugt werden. Eine vollständig gefüllte Box hätte maximal $\binom{R+3-1}{3}^S$ mögliche Kombinationen aus Kleidungsstücken, wenn alle Stilrichtungen miteinander kombinierbar sind. Der Term $\binom{R+3-1}{3}$ entspricht hierbei dem dreimaligen Ziehen aus R Elementen mit Zurücklegen und ohne Beachtung der Reihenfolge. Bei fünf Richtungen und vier Sorten sind das bereits $\binom{5+3-1}{3}^4 \approx 10^6$ Möglichkeiten für eine einzelne Box. Im schlimmsten Fall müssen nun alle Kombinationen aus diesen Boxen getestet werden, um eine optimale Lösung zu finden. Der Suchraum wird somit schnell zu groß, um ihn vollständig abzulaufen und garantiert eine optimale Lösung zu finden.

2.1 Lösungsansatz

Aufgrund der hohen Komplexität eines garantiert optimalen Lösungsverfahrens beschränkt sich der hier vorgestellte Lösungsansatz auf Heuristiken, die gute Lösungen finden, welche nicht zwangsläufig optimal sind. Es werden zunächst Boxen erstellt, die genau ein Kleidungsstück von jeder Sorte enthalten, wobei die passenden Stile beachtet werden. Diese Boxen werden als minimal bezeichnet. Sobald aus den noch übrigen Kleidungsstücken keine neuen minimalen Boxen erzeugt werden können, wird versucht die bereits erzeugten minimalen Boxen aufzufüllen. Bei beiden Prozessen werden die Entscheidungen zufällig getroffen, und der Algorithmus wird häufig wiederholt, um Lösungen mit möglichst wenig übrigen Kleidungsstücken zu finden.

2.2 Maximierung der Boxanzahl

Es ist stets besser eine neue Box zusammenzustellen, anstatt eine bereits vorhandene aufzufüllen. Durch das Hinzufügen von Kleidungsstücken zu einer vorhandenen Box kann die Anzahl der Möglichkeiten, ein Kleidungsstück in eine Box zu packen, nur gleich bleiben oder sinken. Wenn jedoch eine neue Box erstellt wird, kann die Anzahl der Möglichkeiten für die übrigen Kleidungsstücke gleich bleiben oder steigen.

Zunächst wird mithilfe von Backtracking eine Liste M aller möglichen minimalen Boxen erstellt. Hierzu wird in jedem Backtracking-Schritt eine Stilrichtung zu einer Sorte Kleidung

Algorithmus 1 MinimalBoxListeErstellen

```
procedure BoxFüllen(b, s)                    ▷ b ist die Box und s die Sorte
    if b enthält alle Sorten then
        füge b zu M hinzu
        return
    end if
    for r ∈ Richtungen do
        if r passt zu b und (s, r) hat mindestens ein Kleidungsstück then
            füge (s, r) zu b hinzu
            BoxFüllen(b, s + 1)
            entferne (s, r) aus b
        end if
    end for
end procedure
```

Algorithmus 2 BoxenAuswählen

```
while M nicht leer do
    wähle b zufällig aus M
    füge b zur Lösung L hinzu
    streiche die in b verwendeten Kleidungsstücke
    for x ∈ M do
        if nicht genügend Kleidungsstücke für x then
            entferne x aus M
        end if
    end for
end while
```

gewählt und getestet, ob sie in die derzeitige Box passt. Passt sie, so wird die nächste Sorte ausgewählt. Passt sie nicht, wird für diese Sorte eine neue Stilrichtung ausprobiert. Dies geschieht für alle Sorten und alle Stilrichtungen. Der folgende Pseudocode 1 erläutert das Erstellen der Liste aller möglichen Boxen.

Nun wird eine Box aus dieser Liste zufällig ausgewählt, der Lösungsliste L hinzugefügt und die Anzahl der noch übrigen Kleidungsstücke mit den jeweiligen Sorten und Richtungen dieser Box um eins verringert. Darauf hin wird für jede Box in der Liste M getestet, ob für die Zusammenstellung einer solchen Box noch genügend Kleidungsstücke übrig sind. Wenn nicht, wird diese Box aus der Liste entfernt. Dies wird solange wiederholt bis die Liste M leer ist. Der Pseudocode 2 erläutert diesen Teilprozess.

Der Algorithmus terminiert, wenn keine weiteren Boxen mehr erzeugt werden können und somit die Anzahl der Boxen lokal maximiert ist. Es ist zu beachten, dass aufgrund der zufälligen Auswahl die Anzahl der Boxen in der Lösung zwar lokal maximal ist, aber über mehrere Versuche nicht die gleiche Anzahl an Boxen entstehen muss.

2.3 Auffüllen der Boxen

Im nächsten Schritt werden möglichst viele der übrig gebliebenen Kleidungsstücke in die erstellten Boxen gepackt. Hierzu wird eine Liste mit allen Möglichkeiten für das Hinzufügen

Stilrichtung	Integer
0	111
1	011
2	101

Tabelle 1: Werte der Integer

Variable	Wert
Box	111
Zu Richtung 1 passend	011
Bitweises Und	011

Tabelle 2: Bitweises Und

eines neuen Kleidungsstücks zu einer bestehenden Box erstellt. Für jede Sorte und Richtung, bei der noch Kleidungstücke übrig sind, wird getestet, in welche Boxen dieses Kleidungsstück passt. Es entsteht eine Liste mit Tripeln, wobei ein Tripel (s, r, i) dafür steht, dass ein Kleidungsstück mit Sorte s und Richtung r in Box i gepackt werden kann. Dann wird zufällig ein Element aus dieser Liste der Möglichkeiten ausgewählt und das Kleidungsstück in die entsprechende Box gepackt. Aus der Liste werden nun alle Tripel gelöscht, welche aufgrund der veränderten Box oder aufgrund der Anzahl der noch übrigen Kleidungsstücke nicht mehr möglich sind. Dies wird wiederholt, bis die Liste leer ist.

2.4 Überprüfen der Stilrichtungen

Um die Prüfung zu beschleunigen, ob eine Stilrichtung in eine Box passt, wird für jede Stilrichtung ein Bitvektor als Integer abgespeichert. Das i-te Bit dieses Integers für eine Stilrichtung r ist 1, wenn Stilrichtung i zu r passt, sonst 0. Eine Box enthält nun neben ihren Kleidungsstücken noch einen Integer, der bitweise alle in der Box enthaltenden Stile speichert. Das Hinzufügen von Kleidungsstücken kann schnell überprüft und ausgeführt werden.

Im Folgenden Beispiel gibt es 3 Stilrichtungen. Stilrichtung 0 passt zu allen anderen, Stilrichtung 1 passt zu 2, und Stilrichtung 2 passt zu 0. Jede der Richtungen passt auch zu sich selbst. Die Integer zu diesen Richtungen haben dann die Werte von Tabelle 1.

In eine leere Box passt zunächst jede Richtung, somit ist ihr Integer in diesem Beispiel 111. Wird ein Kleidungsstück mit Richtung 0 hinzugefügt, so ändert sich dieser Integer nicht, da weiterhin alle Richtungen möglich sind. Wenn ein Kleidungsstück mit Richtung 1 hinzugefügt wird, verändert sich der Integer der Box. Das *bitweise Und* des Integers für Richtung 1 und des Integers für die Box muss berechnet werden. Tabelle 2 zeigt eine solche Operation.

Nun können der Box also lediglich Kleidungsstücke der Stilrichtung 1 und 2 hinzugefügt werden.

2.5 Laufzeitanalyse

Es gibt maximal R^S verschiedene Boxen mit einem Kleidungsstück pro Sorte, mit R als Anzahl der Stilrichtungen und S als Anzahl der Kleidungssorten. Bei N Kleidungsstücken ist eine obere

Grenze für die Anzahl der Boxen in der Lösung $B = \lfloor \frac{N}{S} \rfloor$. Die Laufzeit für das Erstellen der Boxen beträgt somit:

$$\mathcal{O}\left(\left\lfloor \frac{N}{S} \right\rfloor R^S\right) = \mathcal{O}\left(\frac{N \cdot R^S}{S}\right)$$

Für das Auffüllen der Boxen wird eine Liste aller Möglichkeiten für das Hinzufügen eines Kleidungsstücks zu einer Box erzeugt. Diese Liste hat maximal $B \cdot R \cdot S$ Elemente. Somit beträgt die Laufzeit für das Auffüllen:

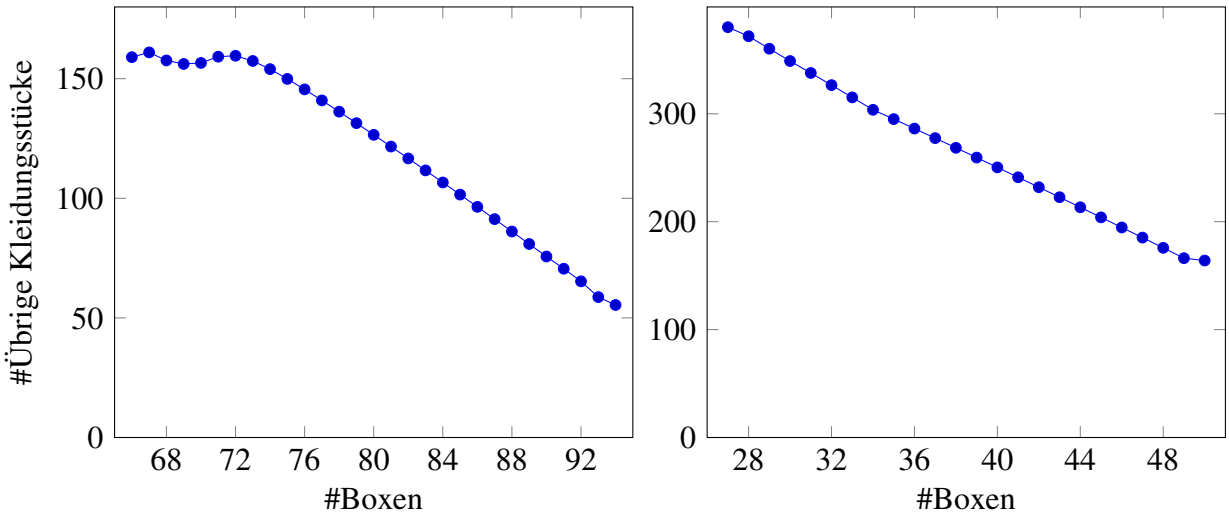
$$\mathcal{O}\left(\left\lfloor \frac{N}{S} \right\rfloor R \cdot S\right) = \mathcal{O}(N \cdot R)$$

Die Gesamtlaufzeit ist somit:

$$\mathcal{O}\left(\frac{N \cdot R^S}{S} + N \cdot R\right)$$

2.6 Quantitative Optimierungen

Der hier beschriebene Algorithmus wird nun oftmals wiederholt, um Ergebnisse mit möglichst wenig übrigen Kleidungsstücken zu erzeugen. Mehr Iterationen des Algorithmus können zu besseren Ergebnissen führen, falls das Minimum nicht bereits erreicht wurde. Es gibt jedoch einige Heuristiken, die das Finden von besseren Lösungen beschleunigen können. Bei den Beispieldateien haben die besten Lösungen oft auch die höchste Anzahl an Boxen. Bei `paeckchen6.txt` und `paeckchen7.txt` sinken die im Durchschnitt übrigen Kleidungsstücke, wie in Abbildung 2.1 zu sehen.



(a) `paeckchen6.txt` (b) `paeckchen7.txt`

Abbildung 2.1: Verhältnis von Anzahl der Boxen und übriger Kleidungsstücke

Eine einfache Heuristik vermeidet das Auffüllen, wenn die Anzahl der Boxen nicht mindestens so groß ist wie die maximale bisher gefundene Kombination von minimalen Boxen. Da

sehr große Boxlisten jedoch sehr selten sind, werden oft nur wenige Iterationen des Auffüllens ausgeführt. Um dem entgegenzuwirken, werden 10 Auffüllversuche pro gefundene maximale Boxliste durchgeführt.

Eine weitere Verbesserung ist die Verwendung von Multithreading, um mehr Durchläufe des Algorithmus in der gleichen Zeit zu schaffen. Dies ist wohlgemerkt komplexitätstheoretisch keine Verbesserung. Aufgrund der Funktionsweise des Algorithmus, können mit der Aufbringung von mehr Laufzeit aber bessere Ergebnisse bezogen auf die gleiche Laufzeit erzielt werden. Es ist also sinnvoll die Ergebnisse der Optimierungen bei gleichem Zeitaufwand zu vergleichen. Dies wird in Abschnitt 2.8 dargestellt.

2.7 Qualitative Optimierung

Eine Heuristik, welche die zufällige Auswahl in dem Algorithmus in eine vorteilhafte Richtung lenkt, könnte im Schnitt zu besseren Ergebnissen führen. Wenn minimale Boxen, die besonders häufig möglich wären, zuerst gewählt werden, kann das öfter zu großen Boxlisten führen. Wenn zu einem Zeitpunkt des Algorithmus von minimaler Box A und B maximal fünf weitere aus den noch übrigen Kleidungstücken erzeugt werden können und von Box C nur drei, dann wird zufällig entweder Box A oder B gewählt. Diese *Häufigkeitsheuristik* führt trotz der erhöhten Rechenzeit wesentlich häufiger zu größeren Boxlisten, wie in Abbildung 2.2 zu erkennen.

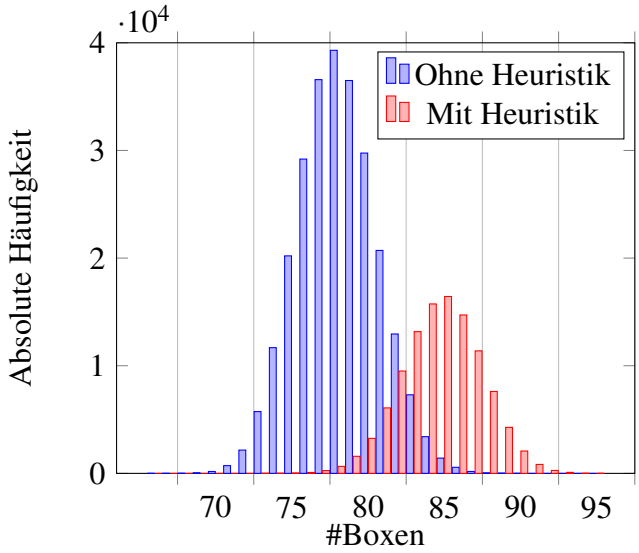


Abbildung 2.2: Vergleich der Häufigkeitsheuristik

Für die Daten in Abbildung 2.2 wurden für 10 Sekunden Boxlisten erzeugt. Es ist erkennbar, dass zwar weniger Listen erzeugt wurden und trotzdem mehr Boxlisten mit sehr vielen Boxen. Ein Backtracking-Algorithmus, welcher alle maximal großen Boxlisten erzeugt und daraufhin diese auffüllt, ist ebenfalls denkbar. Große Boxlisten sind selten, wie in Abbildung 2.3 zu erkennen.

Von den $6 \cdot 10^7$ Boxkombinationen aus der Verteilung in Abbildung 2.3 bestand eine aus 50 Boxen. Da dieser Algorithmus einer vollständigen Suche ähnelt, ist er nur auf bestimmte Beispieldateien anwendbar und führt bei gleicher Rechenzeit nur bei `paeckchen7.txt` und `paeckchen5.txt` zu besseren Ergebnissen.

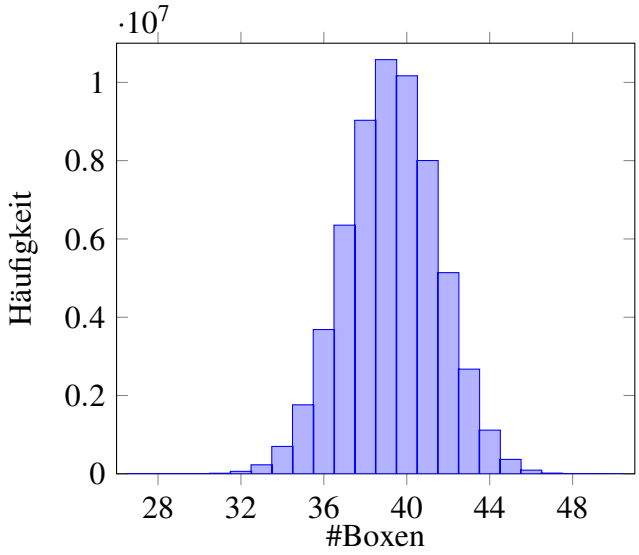


Abbildung 2.3: Verteilung der Größe von Boxkombinationen bei `paeckchen07.txt`

2.8 Vergleich der Optimierungen

Es gibt 3 Optimierungen, die hier getestet wurden. Die erste Optimierung (O_1) ist der Verzicht auf das Auffüllen der Boxen, wenn die Anzahl der Boxen nicht mindestens so groß ist wie beste bisher gefundene Boxliste. Die zweite Optimierung (O_2) ist das Erstellen der Boxen mit der Häufigkeitsheuristik, und die dritte Optimierung (O_3) ist die Verwendung von Multithreading, hier mit 5 parallel laufenden Threads. Die folgende Tabelle zeigt die im Schnitt übrigen Kleidungsstücke, zusammen mit ihrer Streuung. Für die Daten wurden 100 Versuche mit je 90 Sekunden Rechenzeit ausgeführt. Die Ergebnisse für die Beispiele `paeckchen0.txt` bis `packchen3.txt` sind bei allen Varianten gleich und daher für diesen Vergleich uninteressant.

Methode	`paeckchen4.txt`	`paeckchen5.txt`	`paeckchen6.txt`	`paeckchen7.txt`
-	11.70 ± 1.07	19 ± 0.00	46.79 ± 3.31	164.29 ± 3.24
O_1	11.85 ± 1.31	19 ± 0.00	51.03 ± 5.77	159.03 ± 2.82
O_2	11.00 ± 0.85	19 ± 0.00	43.02 ± 3.16	160.84 ± 2.67
O_3	11.16 ± 0.83	19 ± 0.00	25.07 ± 1.04	155.79 ± 1.62
$O_1 + O_2$	10.32 ± 1.22	19 ± 0.00	26.31 ± 2.70	151.81 ± 1.65
$O_1 + O_2 + O_3$	9.23 ± 0.89	19 ± 0.00	23.83 ± 1.01	150.04 ± 1.17

Tabelle 3: Vergleich der Optimierungen

Es ist auffällig, dass die erste Optimierung im Durchschnitt bei `paeckchen4.txt` und `paeckchen6.txt` zu schlechteren Ergebnissen führt. Die Streuung ist jedoch höher, was dazu führt, dass ähnlich gute Bestwerte gefunden werden können. Die höhere Streuung lässt sich dadurch erklären, dass nur selten das Auffüllen ausgeführt wird, wenn sehr früh eine Lösung mit hoher Boxanzahl gefunden wurde. Abgesehen von dieser Anomalie ist erkennbar, dass alle verwendeten Optimierungen zu Verbesserungen führen.

Algorithmus 3 GeneriereMaximaleBoxkombinationen

```
procedure KombinationErstellen(i, k)        ▷ i Index in L, k die derzeitige Kombination
    if i = |L| then
        if |k| > |K| then
            L = ∅
            K = k
        end if
        if |k| = |K| then
            füge k zu L hinzu
        end if
        return
    end if
    max = ∅
    for b ∈ {i..|L| − 1} do
        füge die maximale Anzahl der Box L[b] zu max hinzu
    end for
    if |k| + summe(max) < |K| then
        return
    end if
    for a ≤ maximale Anzahl der Box L[i] do
        füge a mal die Box L[i] zu k hinzu
        reduziere Anzahl übriger Kleidungsstücke a mal

        KombinationErstellen(i + 1, k)

        erhöhe Anzahl übriger Kleidungsstücke a mal
        entferne alle Boxen L[i] aus k
    end for
end procedure
```

2.9 Backtracking zur Maximierung der Boxanzahl

Ein Backtrackingalgorithmus, welcher alle maximal großen Kombination aus minimalen Boxen erzeugt, könnte es ermöglichen, bessere Ergebnisse zu erzielen. Hierfür wird zunächst die Liste L aller möglichen minimalen Boxen erzeugt. Dann wird bei jedem Backtrackingschritt die Häufigkeit einer Box in der Boxkombination gezählt. Die größten gefundenen Kombinationen K werden in einer Liste gespeichert. Wenn eine Kombination gefunden wurde, die Größer als alle vorherigen war, wird die Liste geleert, bevor das neue Maximum eingefügt wird, sodass nur maximal große Kombinationen in der Liste sind. Ist an einer Stelle erkennbar, dass mit den verbleibenden minimalen Boxen keine maximale Kombination erzeugt werden kann, so wird die Suche in diesem Teil des Suchbaumes frühzeitig abgebrochen. Hierfür wird abgeschätzt, wie viele Boxen noch erzeugt werden können. Die maximale Anzahl einer minimalen Boxen ist limitiert durch das Kleidungsstück aus der Box von dem die geringste Stückzahl übrig ist. Der folgende Pseudocode 3 beschreibt das Verhalten dieses Algorithmus.

2.10 Linear Program

Eine Implementierung als Integer Linear Program (ILP) `https://de.wikipedia.org/wiki/Ganzzahlige_lineare_Optimierung` ist ebenfalls denkbar. Hierfür werden zunächst alle möglichen Boxen erstellt. Dafür muss der Algorithmus für die Erstellung der minimalen Boxen nur leicht angepasst werden. Es wird nun eine ganzzahlige Variable für jede Art von Box gewählt, welche beschreibt, wie oft diese Box verwendet wird. Für jede Kombination aus Richtung und Sorte wird nun eine Ungleichung eingeführt, welche beschreibt, dass nicht mehr Kleidungsstücke verwendet werden können als für diese Kombination vorhanden sind. Jede Box, die ein Kleidungsstück dieser Sorte und Richtung enthält, kommt dann in dieser Ungleichung vor.

Im folgenden Beispiel gibt es zwei Boxen, wobei eine Box p_0 zwei und die andre p_1 nur ein Kleidungsstück (s,r) enthält. Wenn es insgesamt fünf Kleidungsstücke mit Sorte s und Richtung r gibt, dann sieht die Ungleichung für (s,r) folgendermaßen aus:

$$2p_0 + p_1 \leq 5$$

Die zu maximierende lineare Funktion ist die Gesamtanzahl aller verwendeten Kleidungsstücke.

Eine solche Formulierung kann mit Hilfe eines ILP Solvers gelöst werden. Die Laufzeit ist hier abhängig von der Anzahl der Variablen und der Anzahl der Gleichungen. Die Anzahl der verschiedenen Boxen und somit die Anzahl der Variablen für die einzelnen Beispieldateien ist in folgender Tabelle 4 zusammengetragen.

Beispiel	Sorten	Richtungen	Sorten · Richtungen	Boxarten
`paeckchen0.txt`	3	2	6	72
`paeckchen1.txt`	3	4	12	2133
`paeckchen2.txt`	3	9	27	1141
`paeckchen3.txt`	5	10	50	2698725
`paeckchen4.txt`	7	24	168	2366334
`paeckchen5.txt`	5	21	105	6678
`paeckchen6.txt`	5	10	50	82296
`paeckchen7.txt`	5	10	50	27216

Tabelle 4: Gleichungs-und Variablenanzahl

Die Anzahl der Gleichungen ist das Produkt aus der Anzahl der Richtungen und der Anzahl der Sorten. Es ist zu erkennen, dass bei `paeckchen3.txt` und `paeckchen4.txt` eine sehr große Anzahl von Variablen entsteht. Das führt dazu, dass der Solver diese Probleme in annehmbarer Zeit nicht optimal lösen kann. Für die Ergebnisse wurden die beiden Solver `lp_solve` und `Cbc` verwendet.

2.11 Beispiele

Die besten Ergebnisse, die beim Testen gefunden wurden, werden hier zusammengetragen. Die Ergebnisse enthalten jedoch oft sehr viele Boxen, weshalb nur für *paeckchen0.txt* die Zusam-

menstellung angegeben wird. Tabelle 5 zeigt alle gefundenen Ergebnisse, zusammen mit der Anzahl der Boxen in der besten Lösung.

Beispiel	Sorten	Richtungen	Kleidungsstücke	Boxen	**Übrig Algo**	**Übrig ILP**
paeckchen0.txt	3	2	11	3	0	0
paeckchen1.txt	3	4	499	142	0	0
paeckchen2.txt	3	9	32	6	0	0
paeckchen3.txt	5	10	96	13	2	-
paeckchen4.txt	7	24	437	35	5	-
paeckchen5.txt	5	21	174	19	19	19
paeckchen6.txt	5	10	770	100	22	22
paeckchen7.txt	5	10	700	50	145	131

Tabelle 5: Ergebnisse des Algorithmus' und des ILP

Der Backtracking-Algorithmus erzeugt für *paeckchen7.txt* alle 21354312 Kombinationen aus 50 Boxen. Nach etwa einer Stunde Rechenzeit wurde eine Lösung mit 138 übrigen Kleidungsstücken gefunden. Die Lösung mit 19 übrigen Kleidungsstücke bei *paeckchen5.txt* wurde etwas schneller gefunden als beim normalen Algorithmus. Bei *paeckchen0.txt* und *paeckchen2.txt* sind die Ergebnisse vergleichbar mit denen des normalen Algorithmus. Auf den anderen Beispieleingaben konnten aufgrund der sehr hohen Rechenzeit kein Ergebnis gefunden werden. Das Ergebnis des ILP Solvers bei *paeckchen4.txt* war nach etwa 6 Stunden Rechenzeit optimal und eine Lösung mit 0 übrigen Kleidungsstücken wurde gefunden. In den hier präsentierten Lösungen wurde sich weniger darauf konzentriert optimale Lösungen mit einem ILP-Solver zu finden, als Ideen zu präsentieren, die ebenfalls zu möglichst guten Ergebnissen führen. Deswegen fehlen die Angaben für *paeckchen3.txt* und *paeckchen4.txt*.

paeckchen0.txt

Sorte	# Richtung 1	# Richtung 2
1	3	0
2	0	1
3	0	1
1	0	1
2	0	1
3	0	1
1	0	1
2	0	1
3	0	1

Tabelle 6: Lösung für *paeckchen0.txt*

Hier wäre auch eine Lösung mit nur 2 Boxen möglich. Da die Anzahl der Boxen zunächst maximiert wird, können Boxen entstehen, welche zusammengeführt werden könnten.

2.12 Bewertungskriterien

Die aufgabenspezifischen Bewertungskriterien werden hier erläutert.

1. Lösungsweg

(1) *Problem adäquat modelliert*: Die Modellierung muss es ermöglichen, für jede Box die enthaltenen Kleidungssorte und Stile zu speichern. Sie sollte eine schnelle Prüfung möglicher Kombinationen für eine Box nicht erschweren. Wenn das Problem als ILP modelliert wird, dann müssen die Gleichungen das Problem korrekt beschreiben.

(2) *Verfahren nicht unnötig ineffizient*: Das Überprüfen ob ein Kleidungsstück mit der Eigenschaft S, R mit der Box kompatibel ist und das Hinzufügen sollten effizient implementiert sein. Wenn z. B. die Kompatibilität mit jedem anderen Kleidungsstück der Box einzeln überprüft wird, so werden zwei Punkte abgezogen.

(3) *Laufzeit des Verfahrens in Ordnung*: Das Verfahren sollte nicht exponentielle Laufzeit in N, der Anzahl der Kleidungsstücke, haben. Verfahren, die alle möglichen Boxen generieren, also R^{3S} Boxen, wie ILP oder genetische Algorithmen, führen ebenfalls zu Punktabzug. Wenn der Algorithmus mit anderen Einstellungen (zum Beispiel maximale Iterationen) länger benötigt, soll dies nicht zu Punktabzug führen.

(4) *Speicherbedarf in Ordnung*: Da die Beispieleingaben eher kleine Eingabegrößen haben, sollte sich der benötigte Speicher in Grenzen halten. Es müssen einerseits die Kleidungsstücke mit S und R sowie die Zuordnung Kleidungsstücke und Box gespeichert werden. Weiterhin die möglichen Kombinationen an Stilrichtungen für jede Box. Der Speicherbedarf der verwendeten Datenstrukturen sollte in Ordnung sein. Für Speicherbedarf im Bereich R^{3S} gibt es Punktabzug.

(5) *Verfahren mit korrekten Ergebnissen*: Das Verfahren muss die in der Aufgabenstellung vorgegebenen Regeln korrekt beachten, also insbesondere diese beiden: (a) Jede Box enthält von jeder Sorte mindestens ein Kleidungsstück, jedoch nicht mehr als drei. (b) Alle eingepackten Stile müssen zusammen passen. Jede Verletzung führt zu zwei Punkten Abzug. Wenn weniger Kleidungsstücke übrig bleiben, oder mehr verpackt werden, als möglich, siehe Optimum in der Tabelle bei Kriterium (6), so kann von mindestens einer Regelverletzung ausgegangen werden.

(6) *Verfahren mit guten Ergebnissen*: Die Güte der einzelnen Ergebnisse kann variieren, sollte aber nicht zu weit von den hier vorgestellten Ergebnissen abweichen. Eine Tabelle gibt Richtwerte an, nach denen die Güte der erzielten Ergebnisse eingestuft werden. Sie enthält für vier Güte-Stufen jeweils die Werte *Maximum übrige Kleidungsstücke / Minimum verpackte Kleidungsstücke*, welche erreicht oder sogar unterschritten (bzw. überschritten) werden müssen, um eine Lösung dieser Qualität zu haben. Liegen die Ergebnisse im Bereich von *Okay*, werden zwei Punkte abgezogen, für schlechtere Ergebnisse vier. Für Ergebnisse im Bereich der Beispiellösung gibt es einen Pluspunkt, für noch bessere zwei. Wenn die guten Ergebnisse nur durch einen ILP-Solver erreicht wurden, werden keine Pluspunkte vergeben.

2. Theoretische Analyse

(1) *Verfahren / Qualität insgesamt gut begründet*: Es muss insbesondere erkannt werden, wenn das verwendete Verfahren keine optimalen Ergebnisse liefert. Es muss begründet

Güte	P0	P1	P2	P3	P4	P5	P6	P7
Okay	0 / 11	50 / 449	20 / 12	30 / 66	150 / 287	60 / 114	150 / 620	250 / 450
Gut	0 / 11	0 / 499	0 / 32	2 / 94	20 / 417	30 / 144	50 / 720	160 / 540
Lösung	0 / 11	0 / 499	0 / 32	2 / 94	5 / 432	19 / 155	22 / 748	145 / 555
Optimum	0 / 11	0 / 499	0 / 32	2 / 94	0 / 437	19 / 155	22 / 748	131 / 569

Tabelle 7: Richtwerte

werden, dass das Verfahren nur korrekte Ergebnisse erzeugt und warum Optimierungsideen zu besseren Ergebnissen führen. Falls verwendete Algorithmen auf einigen Beispielen zu keinen oder besonders schlechten Ergebnissen führen, muss dies erkannt und diskutiert werden.

(2) *Gute Überlegungen zur Laufzeit des Verfahrens*: Die Laufzeit des Verfahrens muss nicht zwingend formal, aber nachvollziehbar und korrekt charakterisiert werden.

(3) *NP-Schwere des Problems erkannt*: Falls die NP-Schwere erkannt und korrekt begründet wird, etwa durch eine (informelle) Reduktion von einem bekannten NP-schweren Problem, können Pluspunkte vergeben werden.

3. Dokumentation

(3) *Vorgegebene Beispiele dokumentiert*: Es müssen alle Beispiele `paeckchen0.txt` bis `paeckchen7.txt` dokumentiert werden.

(5) *Ergebnisse nachvollziehbar dargestellt*: Es muss die Anzahl übriger oder verpackter Kleidungsstücke angegeben werden. Außerdem muss für mindestens ein Beispiel der Inhalt der gepackten Boxen angegeben werden. Besonders gute oder ausführliche Darstellungen der Ergebnisse können mit einem Pluspunkt versehen werden.

Aufgabe 3: Die Siedler

In dieser Aufgabe sollen möglichst viele Ortschaften in einem durch ein Polygon begrenzten Gebiet platziert werden, wobei Vorgaben der Regierung sowohl die Flächennutzung optimieren als auch das Verbreiten von Krankheiten eindämmen sollen. So muss ein Mindestabstand von $r_{\text{min}} = 10\,\text{km}$ zwischen den Ortschaften eingehalten werden. Außerdem soll das Verbreiten von Krankheiten zwischen den Ortschaften verhindert werden, wofür zwei Optionen zur Verfügung stehen: (a) Es soll ein Gesundheitszentrum im Gebiet errichtet werden, das in einem Radius von 85 km alle Ortschaften schützt. (b) Sind zwei Ortschaften mindestens $r_{\text{sicher}} = 20\,\text{km}$ voneinander entfernt, können sich ebenfalls keine Krankheiten zwischen diesen ausbreiten.

3.1 Lösungsidee

Um die Positionen von N Ortschaften und die des Gesundheitszentrums zu beschreiben, werden $2N+2$ Variablen benötigt, die jeweils die x- und y-Koordinaten der Positionen beschreiben. Um nun das Entscheidungsproblem „mindestens N Ortschaften können im Gebiet platziert werden“, zu lösen, muss der entsprechend $(2N+2)$-dimensionale Suchraum nach einer gültigen Platzierung durchsucht werden. Dabei treten grundsätzlich zwei Probleme auf: Einerseits ist der Suchraum selbst extrem hochdimensional und die zu optimierende Größe stetig. Dadurch müssten im Falle einer Brute-Force-Optimierung extrem viele Punkte abgesucht werden. Andererseits sind die Variablen selber nicht unabhängig voneinander, können also auch nicht getrennt optimiert werden; verschiebt man beispielsweise das Gesundheitszentrum, können bestimmte Platzierungen der Ortschaften im Sinne der Regierungsvorgaben ungültig werden. Wenn n_x und n_y die Anzahl diskretisierter Punkte der x- bzw. y-Richtung sind, wären $\mathcal{O}(n_x^{N+1} n_y^{N+1})$ Parametersätze zu untersuchen.

Um die Komplexität der Aufgabe zu reduzieren, werden wir in zwei Schritten vorgehen:

1. Zunächst konstruieren wir einen Greedy-Algorithmus, der basierend auf einigen wenigen Startparametern das Gebiet mit möglichst vielen Ortschaften füllt. Die Idee dahinter ist, dass eine sehr gute Lösung durch eine Folge annähernd optimaler Platzierungen von Ortschaften erreicht werden kann. Dadurch genügt es, Kriterien zu konstruieren, die für eine bereits platzierte Menge von Ortschaften eine sehr gute Platzierung für eine nächste Ortschaft (insofern noch Platz im Gebiet ist) bestimmen. So kann man das Gebiet nach und nach mit nahezu optimalen Ortschaftsplatzierungen füllen und erhält am Ende einen sehr guten Besiedlungsplan.
2. Danach verwenden wir einen Optimierungsalgorithmus, um die vom Greedy-Algorithmus vorgenommene Platzierung hinsichtlich der Startparameter zu verbessern. Hier ist das Ziel, die Startparameter zu finden, bei denen möglichst viele Ortschaften in das Gebiet gesetzt werden.

Interpretation der Aufgabenstellung

Die in der Aufgabenstellung festgehaltenen Regeln der Regierung bieten Interpretationsspielraum. Für die Implementierung sind allerdings genaue Festlegungen wichtig:

- Das Gesundheitszentrum kann einerseits zwingend innerhalb einer Ortschaft gebaut sein, andererseits aber auch einfach freistehend im Gebiet platziert werden. Um eine Ver-

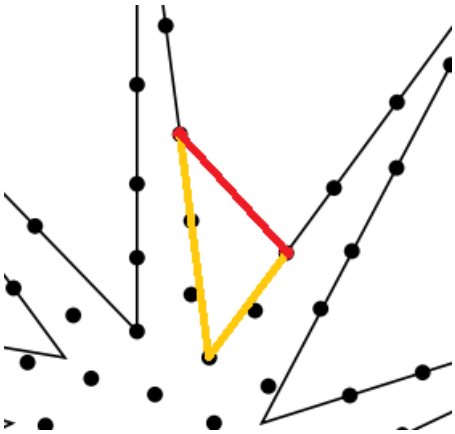


Abbildung 3.1: In rot ist die euklidische Distanz bzw. die Luftlinie zwischen den beiden roten Orten dargestellt. In Orange ist die tatsächliche Distanz zwischen den beiden roten Punkten im Polygon dargestellt.

gleichbarkeit mit möglichst vielen Lösungen zu ermöglichen, werden wir in den Beispielen beides untersuchen; primär konzentriert sich diese Lösung aber auf den Fall eines freistehenden Gesundheitszentrums.

- Ortschaften dürfen sich exakt auf dem Rand des Gebiets befinden.
- Sowohl die für den Mindestabstand als auch die Ansteckungssicherheit relevanten Abstände werden bei uns immer über die Luftlinie (den euklidischen Abstand) berechnet. Hier kann bspw. auch nur der tatsächliche minimale Abstand zwischen zwei Punkten im Polygon berücksichtigt werden, wie in Abbildung 3.1 dargestellt.
- Angenommen Ortschaft 1 ist innerhalb des schützenden Umkreises des Gesundheitszentrums, Ortschaft 2 ist hingegen außerhalb dessen Reichweite. Nun kann es zwei Interpretationen geben, welcher Abstand zwischen diesen Ortschaften für den Schutz von Ortschaft 2 nötig ist.

  (I_1) Von Ortschaft 1 geht keine Gefahr aus, weil diese im Schutzradius ist. Ortschaft 2 kann deshalb bis auf den Minimalabstand von 10 km an Ortschaft 1 heranrücken.

  (I_2) Weil Ortschaft 2 nicht selbst innerhalb das Schutzradius liegt, muss sie zu Ortschaft 1 den vollen Sicherheitsabstand von 20 km wahren.

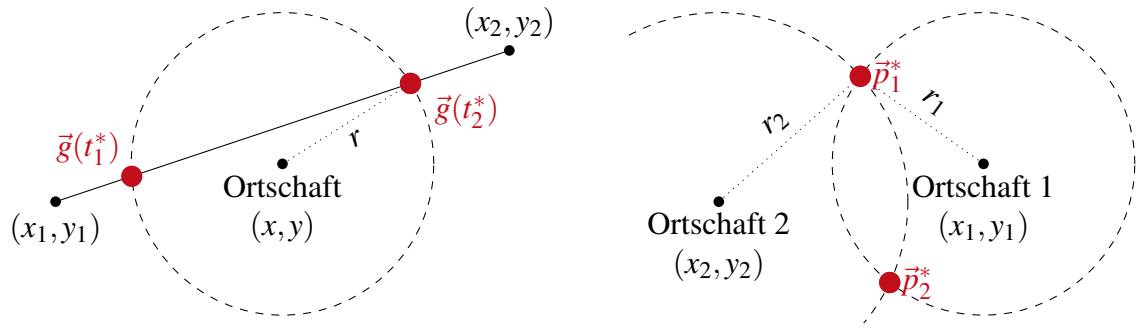
  Beide Interpretationen werden hier untersucht.

Greedy-Heuristiken und Geometrie

Der Kern unserer Lösung sind die Greedy-Heuristiken, also die Leitsätze, nach denen die nächste zu platzierende Ortschaft festgelegt wird. Diese basieren auf der folgenden Annahme:

Jede neu hinzugefügte Ortschaft sollte so nah an den bereits platzierten sein, wie nach den Vorgaben der Regierung möglich.

Um die Ortschaften möglichst nah beieinander zu platzieren, setzen wir diese so, dass sie genau den benötigten Sicherheitsabstand voneinander haben. Dabei beschränken wir uns auf die in Abbildung 3.2 dargestellten Fälle. Für diese ergeben sich die möglichen Positionen neuer Ortschaften wie folgt:



(a) Kandidaten $\vec{g}(t_1^*)$ und $\vec{g}(t_2^*)$ für neue Ortschaften, wenn der Umkreis einer Ortschaft bei (x,y) mit Abstand-Radius r sich mit einer Seite $(x_1,y_1)-(x_2,y_2)$ des Polygons schneidet.

(b) Kandidaten $\vec{p}_1^*$ und $\vec{p}_2^*$ für neue Ortschaften, wenn zwei Ortschaften bei (x_1,y_1) und (x_2,y_2) mit durch die Vorgaben bestimmten Sicherheitsabständen r_1 und r_2 nahe genug beieinander sind.

Abbildung 3.2: Skizzen für die beiden für den Greedy-Ansatz relevanten geometrischen Konstruktionen. Die beiden roten Punkte markieren jeweils Positionen, die die Vorgaben der Regierung erfüllen und damit Kandidaten für die Platzierung einer Ortschaft sind.

(a) Liegt eine Ortschaft nah einer Grenze des Gebiets, so sind die Schnittpunkte ihres Sicherheitsradius mit der begrenzenden Polygonseite zwei Kandidaten für neue Ortschaftspositionen. Die Konstruktion ist in Abbildung 3.2a dargestellt. Um die Schnittpunkte auszurechnen, beschreiben wir eine Polygonseite g, also die Strecke von einer Polygonecke $\vec{s}_1 = (x_1,y_1)$ zur nächsten Ecke $\vec{s}_2 = (x_2,y_2)$, als

$$\vec{g}(t) = \vec{s}_1 + t(\vec{s}_2 - \vec{s}_1), \quad t \in [0,1].$$

Die Schnittpunkte bei t^* liegen nun auf $\vec{g}(t)$ und haben von $\vec{o} = (x,y)$ den Abstand r. Es gilt also

$$\|\vec{g}(t^*) - \vec{o}\|_2 = \sqrt{(x_1 + t^*(x_2 - x_1) - x)^2 + (y_1 + t^*(y_2 - y_1) - y)^2} = r.$$

Nun geben die Lösungen dieser Gleichung mit $t^* \in [0,1]$ die Koordinaten der Schnittpunkte des Schutzkreises mit der Geraden an.

(b) Schneiden sich die beiden Schutzkreise zweier Ortschaften 1 und 2, so können wie in Abbildung 3.2b dargestellt neue Ortschaften auf diesen Schnittpunkte platziert werden. Seien $\vec{o}_1 = (x_1,y_1)$ und $\vec{o}_2 = (x_2,y_2)$ die Koordinaten der beiden Ortschaften mit Sicherheitsabständen r_1 und r_2. Dann erfüllen die möglichen vorgeschlagenen Positionen neuer Ortschaften $\vec{p}^*$ die Gleichungen

$$\|\vec{p}^* - \vec{o}_1\|_2 = r_1, \qquad \|\vec{p}^* - \vec{o}_2\|_2 = r_2,$$

deren Lösung eben diese Positionen ergibt.

Mit Hilfe dieser beiden Konstruktionen lässt sich immer eine als nächste zu platzierende Ortschaft finden, solange das Polygon zusammenhängend ist, also nicht aus zwei getrennten Gebieten besteht. Darauf basierend verwenden wir die wie folgt priorisierten Greedy-Heuristiken:

(a) Keine Priorisierung (b) Unterschiedliche Kanten priorisiert

Abbildung 3.3: Skizzen von Positionen, die durch sukzessives Schneiden mit dem Rand des Gebiets entstanden sind. Wenn die Positionen auf unterschiedlichen Kanten bevorzugt werden, können enge Spitzen des Gebiets effizienter ausgefüllt werden.

(G_1) Wenn möglich[1], platziere eine Ortschaft, die nicht am Rand des Gebiets liegt. Diese ist also durch das Überschneiden zweier Sicherheitskreise entstanden.

(G_2) Wenn möglich, platziere eine Ortschaft, die am Rand des Gebiets liegt. Ist diese durch den Schnittpunkt einer Polygonseite mit dem Umkreis einer ebenfalls auf dem Rand des Gebiets liegenden Ortschaft entstanden, bevorzugen wir Ortschaften, die auf einer anderen Seite des Polygons liegen. Dadurch kann, wie in Abbildung 3.3 dargestellt, der Platz in eng zulaufenden Spitzen effizienter genutzt werden.

Die Reihenfolge G_1 vor G_2 ist dabei so gewählt, dass zunächst zusammenhängende Innenflächen des Polygons gefüllt werden, bevor Ortschaften auf dem Rand des Gebiets gesetzt werden. Diese Reihenfolge führt zu besseren Ergebnissen als die umgekehrte Reihenfolge. Außerdem platzieren wir bevorzugt Ortschaften, die innerhalb des Schutzradius des Gesundheitszentrums liegen. Dadurch wird zuerst der dichter bebaubare Bereich befüllt, und insgesamt können mehr Ortschaften platziert werden.

Durch Tests hat sich für manche Gebiete ergeben, dass es sich ebenfalls anbietet, die Ecken des Gebiets als Koordinaten für die Platzierung neuer Ortschaften in Betracht zu ziehen. Diese fügen wir dann allen außerhalb des Schutzradius liegenden Ortschaften priorisiert gegenüber ein. Insofern diese Heuristik für die Lösung der Beispieldateien genutzt wurde, ist es entsprechend vermerkt.

Abhängig davon, ob Interpretation I_1 oder I_2 für die Abstandsvorgabe herangezogen wird, muss bei der Wahl der Radien zur Berechnung der jeweiligen Schnittpunkte eine unterschiedliche Wahl getroffen werden. Für I_1, wenn also nur von außerhalb des Schutzradius befindlichen Ortschaften der volle Sicherheitsabstand eingehalten werden muss, kann der Sicherheitsradius von den entsprechenden Ortschaften wie folgt gewählt werden:

$$r_i = \begin{cases} r_{\text{min}} & \text{wenn Ort } i \text{ innerhalb des Schutzradius,} \\ r_{\text{sicher}} & \text{wenn Ort } i \text{ außerhalb des Schutzradius.} \end{cases}$$

Wählt man die Interpretation I_2, kann zunächst auch diese Vorschrift angewandt werden. Nach dem Berechnen der Kandidatenpositionen muss allerdings geprüft werden, ob die neuen Position außerhalb des Schutzkreises liegen. Ist dem so, muss $r_i = r_{\text{sicher}}$ gelten, der Punkt muss möglicherweise verworfen werden und mit dem neuen Radius erneut berechnet werden.

[1] Eine Platzierung ist möglich, wenn sie sowohl den Vorgaben der Regierung entspricht als auch im durch das Polygon beschränkten Gebiet liegt.

Alternative Lösungsansätze

Es wurden noch zwei alternative Ansätze genauer betrachtet:

1. Anstatt die Ortschaften auf den Schnittpunkten der Sicherheitskreise zu platzieren, können mögliche Kandidatenpositionen beispielsweise ein Gitter aus Punkten innerhalb des Polygons sein. Wird eine Ortschaft platziert, werden entsprechend alle noch vorhandenen Punkte des Gitters, die eine ungültige Position ergeben würden, entfernt. Diese Lösung ist allerdings sehr ähnlich zur Schnittpunktberechnung, nur dass sie die vorhandene Fläche durch den Abstand der Gitterpunkte ineffizienter nutzt als die Schnittpunktmethode.
2. Anstatt die Ortschaften konstruktiv zu platzieren, wurde ebenfalls eine physikalisch inspirierte Simulationsmethode untersucht. Hierbei wurde zwischen den einzelnen Ortschaften eine abstoßende Kraft eingeführt, die eine gleichmäßige Verteilung der Ortschaften ermöglichen soll. Außerdem wurden die Ortschaften entsprechend ihres Sicherheitsabstands mit harten Rändern versehen, zwischen denen es zu Kollisionen kommen kann. Genauso kollidiert eine Ortschaft mit dem Rand des Polygons, vergleichbar mit Hockeypucks, die sowohl miteinander als auch mit der Stadionwand kollidieren. Ein Gleichgewicht wird dann erreicht, in dem N Ortschaften im Gebiet platziert werden, diese durch die abstoßenden Kräfte verschoben werden und kollidieren und nach einiger Zeit durch Reibung wieder Energie aus dem System entnommen wird. Allerdings ist diese Methode sehr fehleranfällig durch das Hinzufügen des Gesundheitszentrums, da nicht im Vornherein klar ist, da sich der Sicherheitsabstand zu einer Ortschaft während der Simulation ändern kann (bspw. wenn diese den Schutzkreis verlässt). Dementsprechend wurde sie verworfen.
3. Das in dieser Aufgabenstellung beschriebene Problem ähnelt dem Finden einer dichtesten Kreispackung[2]. Dabei geht es darum, eine Reihe von Kreisen in einem begrenzten Raum so anzuordnen, dass sie sich nicht überlappen. Im Fall der Siedler richtet sich der Durchmesser eines Kreises, also einer Ortschaft, danach, wie weit entfernt vom Gesundheitszentrum dieser platziert wird. Es gibt also Kreise mit 5 km Radius und welche mit 10 km Radius. Auch muss überlegt werden, wie mit Kreisen umgegangen wird, die am Rand des Sicherheitsradius des Gesundheitszentrums liegen.

3.2 Umsetzung

Im Folgenden wird auf konkrete Details der Umsetzung eingegangen.

Datenstrukturen und Umgang mit Rundungsfehlern

Um ein natürliches Wachstum zu ermöglichen, verwalten wir die Kandidatenpositionen in einer Prioritätswarteschlange (priority queue). Immer, wenn ein neuer Kandidat akzeptiert wird, werden dieser Liste die nach den oben genannten Kriterien generierten neuen Kandidatenpositionen hinzugefügt. Diese erhalten nach den oben genannten Kriterien (zunächst alle die im Schutzradius sind; sortiert nach G_1 über G_2 zu priorisierende zuerst) Prioritäten und werden entsprechend dieser solange abgearbeitet, bis die Schlange leer ist.

[2] `https://de.wikipedia.org/wiki/Kreispackung`

Zum Vermerken der bereits platzierten Ortschaften bieten sich an (a) ein Quadtree, um die Laufzeit bei der Kollisionserkennung zu minimieren, oder (b) eine Liste, um die Laufzeit zum Hinzufügen eines akzeptierten Elements zu minimieren.

Die Positionen der Orte und des Gesundheitszentrums werden von uns durch Gleitkommazahlen (Floats) dargestellt. Bei der Implementierung kann es dementsprechend zu Problemen durch Rundungsfehler kommen. Diese umgehen wir, indem wird die zum Platzieren verwendeten Abstände etwas größer wählen, als die später beim Überprüfung der Regeln benutzten Pendants. Durch dieses Vorgehen wird ein Teil der eigentlich nutzbaren Fläche verschwendet, weswegen es sinnvoll ist, den zusätzlichen Abstand möglichst klein zu wählen. Sei r der von der Regierung geforderte Abstand. Dann verwenden wir zum Platzieren $r' = (1+\kappa)r$, wobei $\kappa \approx \sqrt{\varepsilon}$ der von uns gewählte Sicherheitsfaktor und ε die Präzision der verwendeten Gleitkommadarstellung ist. Unsere Wahl von κ ist so getroffen, dass der bei der Berechnung der Schnittpunkte entstandene Fehler durch die verwendete Quadratwurzel ignoriert werden kann. Die dadurch pro Ortschaft verschwendete Fläche ist in Tabelle 8 dargestellt. Bereits für 64-bit Gleitkommazahlen ist der Fehler effektiv vernachlässigbar, entsprechend verwenden wir diese.

Innerhalb oder außerhalb des Polygons?

Damit eine Ortschaft platziert werden kann, muss ihre Position nicht nur den Vorgaben der Regierung genügen, sondern ebenfalls innerhalb des durch ein Polygon beschränken Gebiets liegen. Um dies zu überprüfen, verwenden wir den Punkt-in-Polygon-Test nach Jordan[3]. Dieser bestimmt für ein Polygon mit k Seiten in linearer Laufzeit $\mathcal{O}(k)$, ob ein beliebiger Punkt innerhalb oder außerhalb eines Polygons liegt. Wie in Abbildung 3.4 skizziert, wird dabei für einen Punkt gezählt, wie oft ein von ihm ausgehender Strahl das Polygon schneidet. Wenn s die Anzahl von Schnittpunkten ist, so liegt der Punkt innerhalb, wenn s ungerade ist. Ist s null oder gerade, liegt der Punkt außerhalb.

Für Punkte, die exakt auf dem Rand des Gebiets platziert wurden, wird dieser Test nicht verwendet, um Rundungsfehler zu vermeiden. Stattdessen werden diese Punkte entsprechend unserer Auslegung der Regeln ohne Weiteres als gültig angesehen.

Laufzeit

Um in Algorithmus 4 alle Schnittpunkte eines Kandidaten mit den bereits platzierten n Ortschaften zu berechnen und die Sicherheitsabstände zu überprüfen, ist beim Verwenden eines

[3] `https://de.wikipedia.org/wiki/Punkt-in-Polygon-Test_nach_Jordan`

Datentyp	Rel. Präzision ε	Verschwendete Rel. Fläche $(r'/r)^2 - 1$
32-bit Float	$\approx 1 \cdot 10^{-7}$	$\approx 7 \cdot 10^{-4}$
64-bit Float	$\approx 2 \cdot 10^{-16}$	$\approx 3 \cdot 10^{-8}$
128-bit Float	$\approx 2 \cdot 10^{-34}$	$\approx 3 \cdot 10^{-17}$

Tabelle 8: Zusammenfassung der durch die Erweiterung des Platzierungsradius von r auf $r' = (1+\kappa)r$ verschwendeten Fläche für verschiedene Datentypen. Hier ist wie im gesamten Text $\kappa = \sqrt{\varepsilon}$.

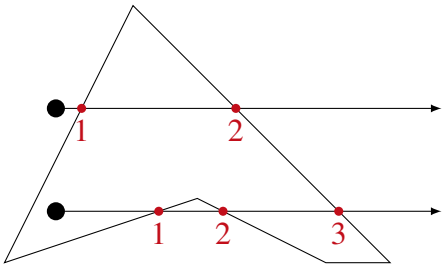


Abbildung 3.4: Illustration des Punkt-in-Polygon-Tests nach Jordan. Von den schwarzen Testpunkten aus wird ein Strahl in eine beliebige (hier horizontal nach rechts) Richtung gesendet. Schneidet dieser Strahl das Polygon gar nicht oder eine gerade Anzahl mal, liegt der Punkt außerhalb des Polygons, andernfalls innerhalb.

Algorithmus 4 Greedy-Algorithmus zum Platzieren der Ortschaften

Require: Polygon P, Startparameter $p = (x_\Gamma, y_\Gamma, x_1, y_1)$
  Initialisiere Prioritätswarteschlange $Q \leftarrow (x_1, y_1), (x_2, y_2)$
  Initialisiere Liste platzierter Ortschaften $D \leftarrow \emptyset$
  while Q nicht leer **do**
    Entferne Element Ort $\vec{o}$ mit der höchsten Priorität aus Q
    if Ort $\vec{o}$ erfüllt die Regierungsvorgaben **then**
      Füge alle Schnittpunkte mit platzierten Ortschaften im Gebiet P in Q hinzu
      Füge alle Schnittpunkte mit Polygon P in Q hinzu
      Platziere Dorf $D \leftarrow \vec{o}$
    end if
  end while
  Gebe Besiedlungsplan D zurück

Quadtree konstanter Zeitaufwand $\mathcal{O}(1)$ nötig, weil lediglich die Ortschaften in den höchstens acht benachbarten Feldern des Quadtree überprüft werden müssen. Würde für die Speicherung der Ortschaften eine Liste verwendet werden, wären $\mathcal{O}(n)$ Zugriffe nötig. Dafür wird beim Quadtree beim Hinzufügen einer akzeptierten Ortschaft die Laufzeit $\mathcal{O}(\log n)$ benötigt, wohingegen die Liste konstanten Aufwand $\mathcal{O}(1)$ erfordert. Insgesamt ergibt sich bei beiden entsprechend nach dem Hinzufügen mit vorangegangener Schnittpunktberechnung und dem Überprüfen auf die Regierungsvorgaben von N Ortschaften ein Aufwand von $\mathcal{O}(N \log N)$ für den Quadtree und ein Aufwand von $\mathcal{O}(N^2)$ für die Liste.

Praktisch kann die Liste trotzdem dem Quadtree vorzuziehen sein, weil die Problemgrößen mitunter sehr klein sind.

Genauso sind in jedem Schritt die mit dem Rand des Polygons entstehenden Schnittpunkte zu berechnen, was bei k der Anzahl von Kanten des Polygons ebenfalls linearen Aufwand $\mathcal{O}(k)$ hat. Für N platzierte Ortschaften ergibt sich die Laufzeit $\mathcal{O}(Nk)$.

Optimierungsalgorithmus

Das oben beschriebene und in Algorithmus 4 als Pseudocode gegebene Greedy-Verfahren benötigt einerseits einen Startzustand, beschrieben[4] durch die Position des Gesundheitszentrums

[4] Hier sind die Startparameter für ein frei stehendes Gesundheitszentrum beschrieben. Bei einem Gesundheitszentrum, dass innerhalb einer Ortschaft gebaut wird, geben wir die Position des Gesundheitszentrums x_Γ, y_Γ

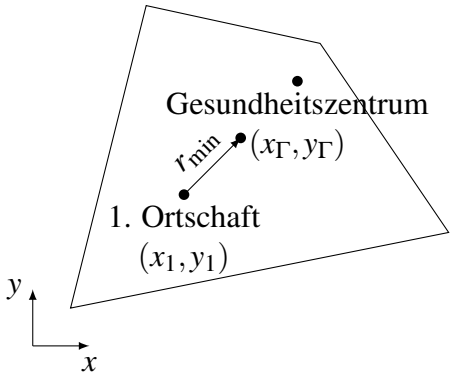


Abbildung 3.5: Skizze der Startparameter $p = (x_\Gamma, y_\Gamma, x_1, x_2)$. Sowohl das Gesundheitszentrum bei (x_Γ, y_Γ) als auch die erste Ortschaft bei (x_1, y_1) beschreiben die Startposition des Greedy-Algorithmus. Die zweite Ortschaft platzieren wir dann von der ersten Ortschaft aus in Richtung des Gesundheitszentrums so, dass diese den Regierungsvorgaben genügt. Die restlichen Platzierungen ergeben sich dann entsprechend Algorithmus 4.

x_Γ, y_Γ und der ersten Ortschaft x_1, y_1. Diese sind in Abbildung 3.5 erläutert und so gewählt, damit die einzelnen Parameter möglichst unabhängig sind und damit gut zu unserem Optimierungsverfahren passen.

Nun gilt es noch den Parametersatz $p^\star = (x_\Gamma^\star, y_\Gamma^\star, x_1^\star, y_1^\star)$ zu finden, für den Algorithmus 4 die meisten Ortschaften platzieren kann. Da die zu optimierende Größe, die Anzahl der Ortschaften, die für eine bestimmte Startposition generiert wurde, ganzzahlig ist, sind ableitungsbasierte Algorithmen nicht anwendbar. Stattdessen nutzen wir einen genetischen Algorithmus.

Dieser arbeitet mit einer Population von N Startparametern $p_1, \ldots, p_N$, von denen jeder eine Fitness, hier die damit generierte Anzahl von Ortschaften im Polygon, zugewiesen bekommt. Von diesen N Startparametern werden die εN besten direkt übernommen, ein Teil μN wird durch Kombination von Eltern bestimmt und der Rest wird zufällig aus der verbleibenden Population gewählt. Die für die Rekombination vorgesehenen Eltern werden ebenfalls zufällig gewählt, wobei bessere Eltern (solche mit einer höheren Fitness) eine höhere Selektionswahrscheinlichkeit haben. Die Rekombination findet durch Mitteln der Parameter zweier Eltern statt. Nach jedem Paaren der Elterngeneration werden außerdem in einem Teil xN der Population die Parameter zufällig angepasst, ähnlich einer zufälligen Mutation in der Natur. Bei dieser verändern wir die Parameter entsprechend der Normalverteilung $\mathcal{N}(\mu = 0, \sigma = 0.75)$. Dies wird solange wiederholt, bis in einer Population für lange Zeit keine Fortschritte mehr erreicht werden.

3.3 Beispiele

In Tabelle 9 sind die Anzahlen der von unserer Implementierung platzierten Ortschaften im Besiedlungsplan dargestellt. Die mit frei stehendem Gesundheitszentrum und der Interpretation I_1 erreichten Besiedlungspläne sind in Abbildung 3.6 dargestellt.

und den Winkel α der zweiten Ortschaft mit

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_\Gamma \\ y_\Gamma \end{pmatrix} + R(\alpha) \begin{pmatrix} r_{\min} \\ 0 \end{pmatrix}$$

und $R(\alpha)$ der 2d-Rotationsmatrix um den Winkel α an.

	I_1		I_2		Grenzwert
	Frei	Fixiert	Frei	Fixiert	
`siedler1.txt`	174_{161}	171_{159}	170_{158}	166_{154}	233
`siedler2.txt`	$\mathit{117}_{108}$	116_{107}	112_{104}	111_{103}	161
`siedler3.txt`	256_{238}	252_{234}	248_{230}	244_{226}	294
`siedler4.txt`	$\mathit{201}_{186}$	198_{184}	194_{180}	191_{177}	290
`siedler5.txt`	$\mathit{200}_{186}$	193_{179}	194_{180}	190_{176}	252

Tabelle 9: Mit genetischer Optimierung der Parameter erreichte Anzahl von Ortschaften in den entsprechenden Beispieldateien. Es sind sowohl die Interpretationen I_1 und I_2 (zum Sicherheitsabstand zwischen einer vom Gesundheitszentrum geschützten und einer nicht geschützten Ortschaft; I_1 = 10 km, I_2 = 20 km) als auch die Wahl zwischen einem freistehenden und in einem Ort fixierten Gesundheitszentrum aufgeführt. Die *kursiv* gesetzten Zahlen wurden ohne explizites Hinzufügen der Eckpunkte des Polygons erreicht. Ab dem Grenzwert markiert den Wert, ab dem eine Lösung definitiv fehlerhaft ist. Die kleingestellten Werte geben die unteren Schranken (93%) für eine qualitativ gute Lösung an.

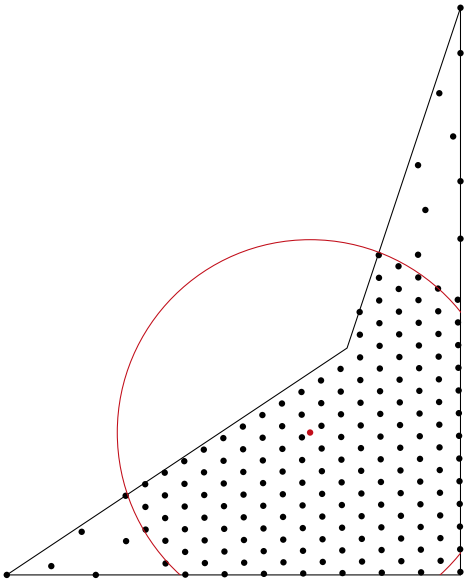
Um die Qualität einer Lösung einzuschätzen, werden zur Bewertung mindestens und höchstens erwartete Anzahlen platzierter Ortschaften herangezogen. Die unteren Schranken wurden dabei durch empirische Beobachtungen an der hier beschriebenen Methode bestimmt. Dabei wurde beobachtet, dass die besten 1% von zufällig gewählten Startparametern für Algorithmus 4 mindestens 93% bis 97%[5] der in Tabelle 9 aufgeführten Ergebnisse erreichen. Entsprechend sollten von guten Einsendungen ebenfalls $\geq 93\%$ der dort angegebenen Ergebnisse erzielt werden.

Da die Ergebnisse nicht garantiert optimal sind, wurde die obere Schranke für sicher falsche Ergebnisse durch eine konservative Abschätzung bestimmt. Sei A_{poly} die Fläche des Polygons und u_{poly} dessen Umfang. Eine Ortschaft muss bei hexagonal dichter Packung (dem Optimum der Ortschaftsverteilung) mindestens die Fläche eine Hexagons mit dem Innenkreis $r_{\min}/2$ einnehmen (genauer: für die Platzierung anderer Ortschaften ausschließen), entsprechend $A_{\min} = \sqrt{3}/2 r_{\min}^2$. Höchstens kann damit die Fläche $A_{\text{poly}} + u_{\text{poly}} * r_{\min}/2$ eingenommen werden, insgesamt passen also weniger als

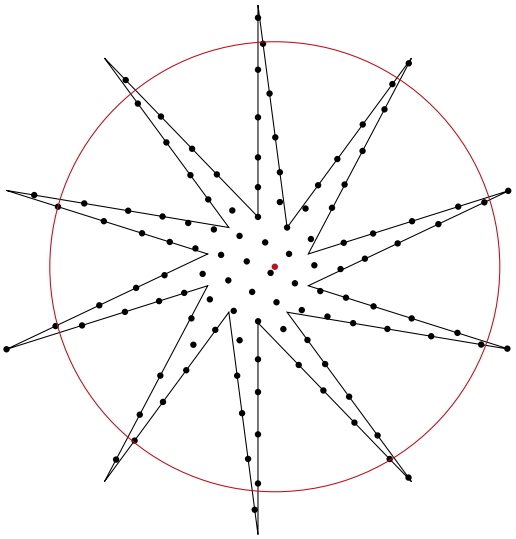
$$\frac{A_{\text{poly}} + u_{\text{poly}} * \frac{r_{\min}}{2}}{\frac{\sqrt{3}}{2} r_{\min}^2}$$

Ortschaften in das Gebiet.

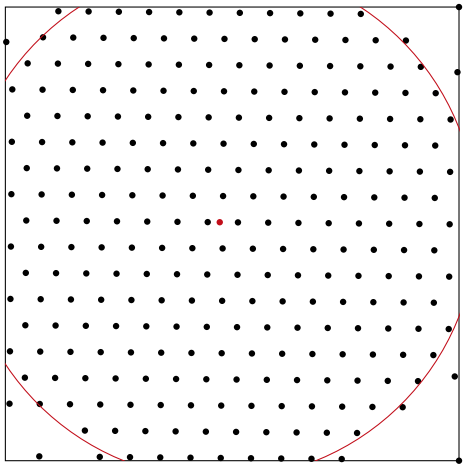
[5] Der konkrete Wert ist abhängig von der Beispieldatei.



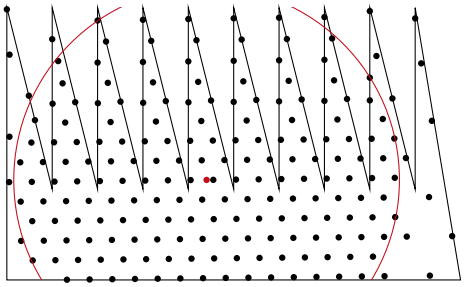
(a) `siedler1.txt` mit 174 Ortschaften



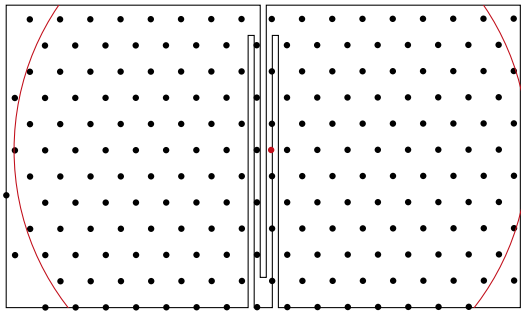
(b) `siedler2.txt` mit 117 Ortschaften



(c) `siedler3.txt` mit 256 Ortschaften



(d) `siedler4.txt` mit 201 Ortschaften



(e) `siedler5.txt` mit 200 Ortschaften

Abbildung 3.6: Illustration der in Tabelle 9 gefundenen optimalen Bebauungspläne für ein freistehendes Gesundheitszentrum und die Interpretation I_1.

3.4 Bewertungskriterien

Die aufgabenspezifischen Bewertungskriterien werden hier erläutert.

1. Lösungsweg

(1) *Problem adäquat modelliert*: Es muss möglich sein festzustellen, ob die Positionen der Ortschaften und des Gesundheitszentrums innerhalb des Polygons liegen. Die Modellierung muss ermöglichen, Abstände und ggf. Schnittpunkte korrekt zu berechnen.

(2) *Verfahren nicht unnötig ineffizient*: Bei der Platzierung der Ortschaften muss strategisch vorgegangen werden. Insbesondere sollte für das Hinzufügen weiterer Ortschaften nicht alle Punkte im Polygon geprüft werden.

(3) *Laufzeit des Verfahrens in Ordnung*: So sollte die Laufzeit für einen Durchlauf zur Platzierung von N Ortschaften $\mathcal{O}(N^2)$ nicht überschreiten.

(4) *Speicherbedarf in Ordnung*: Das Verfahren sollte höchstens die tatsächlich platzierten Ortschaften und eine Kandidatenliste speichern. Für beide ist ein Aufwand von $\mathcal{O}(N)$ (mit der Anzahl N der insgesamt platzierten Ortschaften) akzeptabel.

(5) *Verfahren mit korrekten Ergebnissen*: Ortschaften müssen mit einem Abstand von 20 km, bzw. 10 km, wenn im 85 km Radius vom Gesundheitszentrum, platziert werden. Ortschaften dürfen nur innerhalb des Polygons platziert werden. Wenn der Grenzwert in der Tabelle bei Kriterium (5) überschritten wird, kann auf jeden Fall von einer Regelverletzung ausgegangen werden. Je Regelverletzung können bis zu zwei Punkte abgezogen werden.

(6) *Verfahren mit guten Ergebnissen*: Wenn die erreichte Anzahl von Ortschaften den hier angegebenen 93 % entspricht, dann werden zwei Punkte abgezogen, für deutlich schlechtere Ergebnisse bis zu vier. Das Beispiel S2 wird stärker gewichtet, da es besondere Aussagekraft über die Güte des Verfahrens besitzt. Werden für einige Besiedlungspläne Ergebnisse über denen der Beispiellösung erreicht, können bis zu zwei Pluspunkte vergeben werden. I_1 und I_2 sind die Interpretationen zum *Abstand* zwischen einer Ortschaft 1 innerhalb des schützenden Umkreises des Gesundheitszentrums und einer Ortschaft 2 außerhalb; bei I_1 beträgt dieser Abstand 10 km, bei I_2 20 km.

	Abstand	Gesundheits-zentrum	S1	S2	S3	S4	S5
93%	I_1	Frei	161	108	238	186	186
		Fixiert	159	107	234	184	179
	I_2	Frei	158	104	230	180	180
		Fixiert	154	103	226	177	176
Lösung	I_1	Frei	174	117	256	201	200
		Fixiert	171	116	252	198	193
	I_2	Frei	170	112	248	194	194
		Fixiert	166	111	244	191	190
Grenzwert			233	161	294	290	252

Tabelle 10: Richtwerte

2. Theoretische Analyse

(1) *Platzierungsbedingungen verständlich*: Es muss eindeutig werden, ob das Gesundheitszentrum in einer Ortschaft, außerhalb oder als eigene Ortschaft selbst platziert wird. Weiterhin muss ersichtlich werden, welcher Abstand zwischen Ortschaften im Umkreis des Gesundheitszentrums und außerhalb des Gesundheitszentrum gelten müssen und ob Ortschaften direkt auf den Grenzen liegen dürfen.

(2) *Verfahren / Qualität insgesamt gut begründet*: Es muss insbesondere erkannt werden, wenn das verwendete Verfahren keine optimalen Ergebnisse liefert. Es muss sinnvoll begründet werden, warum das gewählte Verfahren eine möglichst dichte Bebauung des Gebiets ermöglichen soll. Wenn beim Verfahren Schnittpunkte berechnet werden, sollten sich Gedanken über Rundungsfehler gemacht werden und erklärt werden, wie damit umgegangen wird.

(3) *Gute Überlegungen zur Laufzeit des Verfahrens*: Die Laufzeit des Verfahrens muss nicht zwingend formal, aber nachvollziehbar und korrekt charakterisiert werden.

(4) *NP-Schwere des Problems erkannt*: Falls die NP-Schwere erkannt und korrekt begründet wird, etwa durch eine (informelle) Reduktion von einem bekannten NP-schweren Problem, können Pluspunkte vergeben werden.

3. Dokumentation

(3) *Vorgegebene Beispiele dokumentiert*: Es müssen alle Beispiele dokumentiert werden, also `siedler1.txt` bis `siedler5.txt`.

(5) *Ergebnisse nachvollziehbar dargestellt*: Es muss mindestens die Anzahl an platzierten Ortschaften angegeben werden. Weiterhin sollte der Besiedlungsplan graphisch dargestellt werden. Hierbei muss der Radius des Gesundheitszentrums erkennbar sein. Fehlt eines von beiden werden jeweils 2 Punkte abgezogen.

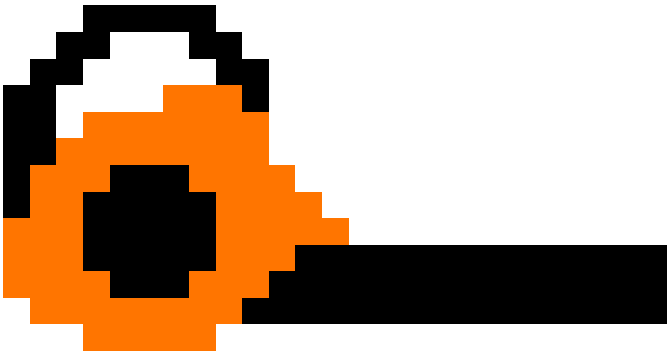
Aus den Einsendungen: Perlen der Informatik

· In diesem Abschnitt finden sich spannende Theorien, unhaltbare Behauptungen und beeindruckende Experimente. Passagen wie diese hier wurden von uns hinzugefügt, der restliche Text und die Abbildungen stammen direkt von den Teilnehmenden. Viel Spaß mit den Perlen aus den Einsendungen der zweiten Runde des 42. *Bundeswettbewerbs! Und nicht vergessen: ·*

As this competition is held in German, the program and its documentation are also written in Germany.

Aufgabe 1: Laubmaschen

Diese Aufgabe schaffte es doch tatsächlich, mich in die Zeit zwischen 1. und 2. Lockdown, in den Herbst von 2020 zu versetzen, und alte Erinnerungen in mir hochkommen zu lassen, aus einer Zeit, in der ständig gelüftet wurde, sodass wir alle in Decken gehüllt im Klassenzimmer saßen und uns darüber aufregten, dass wieder einmal jemand mit dem Laubbläser versucht hat Blätter wegzublasen oder die Kehrmaschine Lärm erzeugte (und weiß der Teufel wieso, sie kam gefühlt jeden Tag). Und auch damals war uns schon bewusst, dass der Laubbläser nichts brachte, denn der Sportplatzwart blies die Blätter einfach nur vom einen Ende des Sportplatzes über die Tartanbahn zum anderen und wieder zurück. So kam es sogar zur Aussage, dass falls unser Lehrer jemals Bundeskanzler werden würde, er als erstes Laubbläser verbieten würde. Als Alternative zum Laubbläser empfehle ich Besen und Rechen.



Ideenfindung

· Um die eigenen Theorien zu untermauern, wurden sogar Experimente durchgeführt: ·

Ich hoffe sie sind genauso begeistert wie ich von diesem Experiment. Denn dann könnte ich meine Eltern davon überzeugen, dass es sich gelohnt hat für die Wissenschaft einen Laubbläser auszuleihen und das Mobiliar in den Garten zu verlagern.

Für die Modellierung habe ich mich […] von Experimenten mit Haferflocken inspirieren lassen.

Praktische Beobachtungen zeigen: Wenn Blätter gegen eine Wand geblasen werden, dann fliegen sie bei ausreichender Krafteinwirkung meist zur Seite weg.

Überlegungen zum Hausmeister und den Rändern

Würde ein Blatt eigentlich auf ein solches „Mauerfeld“ fliegen, lassen wir es stattdessen auf seinem bisherigen Feld liegen. Um es für den Hausmeister nicht unnötig kompliziert zu machen, gilt die Regelergänzung auch für Hecken und Wiesen (Könnte sich ja um Dornenhecken und Wiesen mit zahlreichen Hundehaufen handeln und da wollen natürlich weder Hausmeister noch Blätter hin).

Der Hausmeister überlegt sich noch, was passieren würde, wenn er direkt gegen den Zaun oder in eine Ecke pusten würde. Er kommt allerdings zu dem Schluss, dass das für ihn keinen Sinn macht und er darauf verzichten wird.

Das wurde nicht umgesetzt, da wie oben beschrieben meinem Hausmeister die Schritte egal sind, er will an der Sonne sein und keinen Besen aus der dunklen Kammer des Schreckens holen.

Hierbei soll uns unser Hausmeister, den wir Karl nennen, mit seinem Laubbläser helfen.

Könnten wir einfach sagen, dass die Zäune aus magischen Teleportern bestehen, die das Laub magisch auf das Zielquadrat teleportieren? Nein? Schade.

Unser Ziel ist es, den Hof zu leeren, ohne dass der Hausmeister den Eindruck hat, er müsse sich in einem Blasorchester wiederfinden!

Trotzdem schafft es der Hausmeister von den Rand- und Eckfeldern zu blasen, denn sein Laubbläser hat einen leichten Fehler und ist so stark, dass der Hausmeister damit fliegen kann.

[...] solange der Hausmeister nicht aus dem Fenster bläst, was sich als recht unwahrscheinlich klassifizieren ließe.

Es scheint, als wäre der Hausmeister die ultimative Ausprägung eines Workaholics - für nur ein einziges zusätzliches Blatt würde er sogar 200 Überstunden machen.

Überlegungen zur Strategie

Durch Versuch und Irrtum habe ich herausgefunden, dass etwa 25 Umläufe um das Feld normalerweise ausreichen

Diese Szenarien wurden sorgfältig ausgearbeitet, um eine umfassende Bandbreite von Randbedingungen und Randfällen abzudecken und Einblicke in die nuancierten Dynamiken der Blattverteilung zu bieten.

Die Komplexität bei jedem der vorgestellten Algorithmen im schlimmsten Fall jedoch $O(\infty)$ beträgt, wenn die Blätter völlig ungünstig fliegen bzw. „der Wind schlecht weht“: Es ist anzumerken, dass sich bei einer Grenzwertannäherung an unendlich die Chance, dass es zu diesem Fall kommt, gleichzeitig asymptotisch an 0 annähert.

Ausgaben

· Die Ausgaben waren teilweise überraschend! Neben schwer entzifferbaren Schulhöfen gab es auch solche mit zerfetzten Blättern: ·

Beispieleingaben:

Eingabe	Ausgabe
5, 5	[0, 0.0, 0.21080471022240005, 0, 0] [0, 0, 85.90953662801498, 0.0, 0] [0, 0, 2008.0620731318593, 177.53686766013635, 0] [0, 0, 228.0699131595452, 0.0, 0] [0, 0.0, 0.21080471022240005, 0, 0]
1, 1	[100]
3, 3	[0.0, 0, 0] [79.78161070000002, 675.7936261000001, 79.78161070000002] [0.0, 64.64315250000001, 0]

· Es existierten aber auch andere Ansätze: ·

Hier fällt mir ehrlich gesagt nichts ein, was ich aufschreiben könnte. Um zu gucken, wie die Simulation ist, sollte man sie einfach ausprobieren.

Lösung auf 3×2 Quadraten...

Schätzungen zufolge gibt es in Deutschland ungefähr 90 Milliarden Bäume [Quelle angegeben]. Angenommen jeder Baum trägt im Durchschnitt 30.000 Blätter, was zwar nicht stimmt, da nicht alle Bäume Laubbäume sind, aber für Laubbäume eine gute Approximation ist [Quelle angegeben]. Wenn nun alle Bäume Deutschlands Laubbäume wären und gleichzeitig alle ihre Blätter in diesen Schulhof kämen würden es insgesamt

$$90.000.000.000 \cdot 30.000 = 2.700.000.000.000.000$$

Blätter sein, also 108.000.000.000.000 Blätter pro Quadrat.

Aufgabe 2: Stilvolle Päckchen

Lösungsideen

Das Verpacken [sic] der Boxen ist das Hauptproblem dieser Aufgabe, da hier einige Regeln gelten.

[...] Durch einen Algorithmus die Kleidung in Päckchen packen.

Man könnte einen dynamischen Programmieransatz verwenden (hypothetisch...).

Es ist also einfacher, gleich die richtige Lösung zu produzieren.

Als Beispiel kriegt man einen grün-lila-gelb karierten Blazer fast nirgends dazu, ein weißes T-Shirt passt aber immer.

Am Ende müssen die originellen Stile dann rekonstruiert werden.

Ausgaben

Ich bin Paket: 0Produkte

Insgesamt konnten 0 Kleidungsstücke leider nicht untergebracht werden.

Möglichst viele minimale Pakete erstellen	Fehler! Textmarke nicht definiert.
Aufteilen der übrigen Artikel	Fehler! Textmarke nicht definiert.
Beispiele	***6***
paeckchen0.txt	**Fehler! Textmarke nicht definiert.**
paeckchen1.txt	**Fehler! Textmarke nicht definiert.**
paeckchen2.txt	**Fehler! Textmarke nicht definiert.**
paeckchen3.txt	**Fehler! Textmarke nicht definiert.**
paeckchen4.txt	**Fehler! Textmarke nicht definiert.**
paeckchen5.txt	**Fehler! Textmarke nicht definiert.**
paeckchen6.txt	**Fehler! Textmarke nicht definiert.**
paeckchen7.txt	**Fehler! Textmarke nicht definiert.**

Kritische Betrachtung der Ergebnisse

Komischerweise verschwinden Kleidungsstücke in manchen Beispielen (wahrscheinlich bereuen es die Hersteller so viel Neuware zu verschenken (Beweis: es passiert nur bei größeren Beispielen)). Bei vielen Beispielen ist die Anzahl von Kleidungsstücken pro Sorte viel zu hoch (Marius, Emily und Merle machen beim BWInf mit und gestern war Abgabefrist: alle drei haben kaum geschlafen und sind sehr müde und unkonzentriert).

Meiner Meinung nach ist mein Algorithmus ein schneller Algorithmus mit geringer Speicherkomplexität, um schnell an eine halbwegs gute Lösung zu kommen. Ich denke gerade im schnelllebigen Kleidungsgeschäft ist dies wichtig, da besser eine schnelle Lösung um Boxen herauszuschicken als eine langsame Lösung und dann kommt die Box viel später.

Abschließend lässt sich sagen, dass das Programm trotz der Unfähigkeit, die Anzahl an Kleidungen richtig über die Boxen zu verteilen, trotzdem Lösungen liefern kann, die optimal sind und den Regeln entsprechen.

Aufgabe 3: Die Siedler

„Die Aufgabe *Die Siedler* hat mir einige Sorgen bereitet...“ denn „In dieser Aufgabe dreht sich wortwörtlich alles um das Gesundheitszentrum […].“

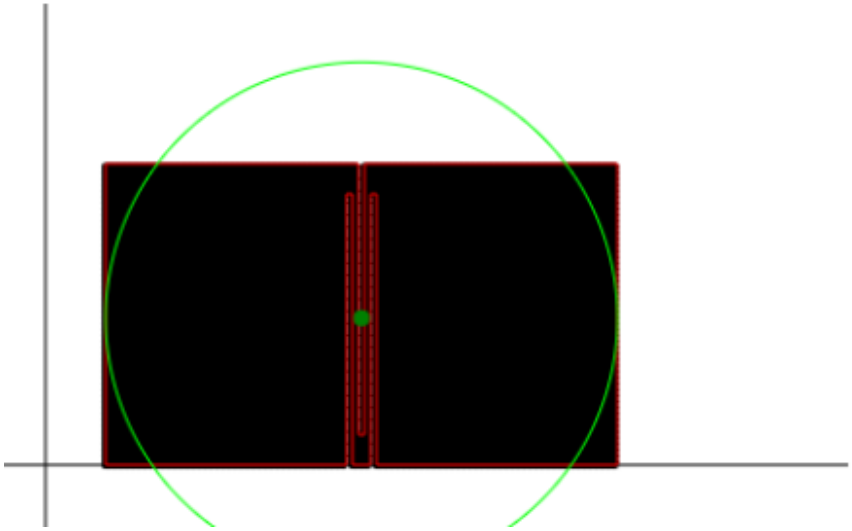
Problemerkundung

· *Es wurden Experimente durchgeführt, um eine optimale Kreispackung zu ergründen:* ·

Lösungsideen

Meine Idee ist es, auf der gesamten Fläche, die das Polygon besetzt, möglichst viele Punkte zu platzieren.

Das Programm erzeugt die erste Ortschaft am Punkt des Gesundheitszentrums - irgendwo muss man ja anfangen.

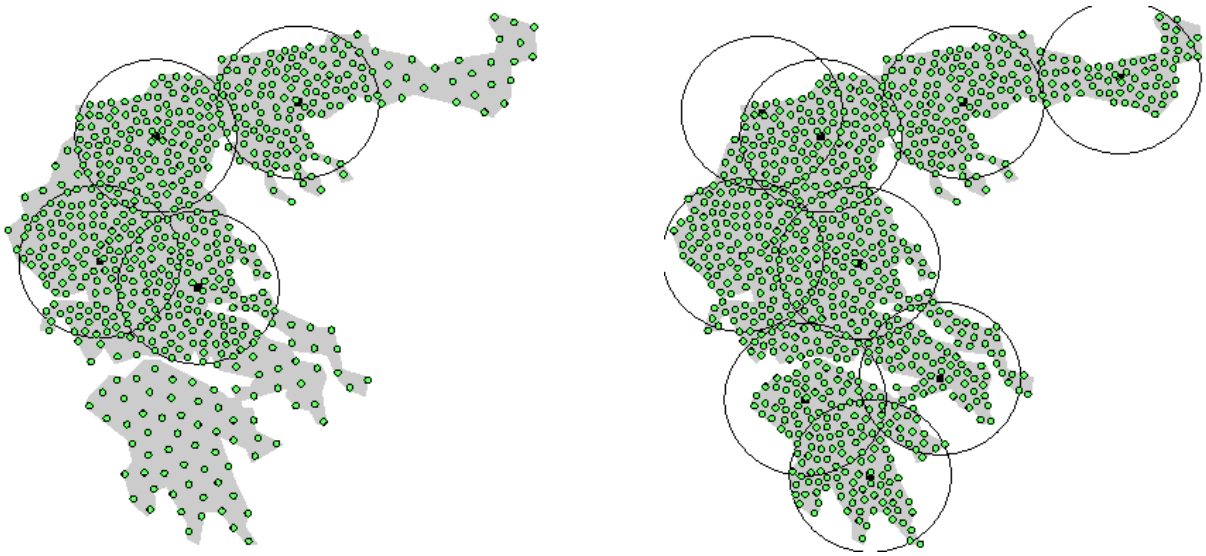


[...] wobei in schwarz alle Positionen markiert sind, an denen das Gesundheitszentrum platziert wurde.

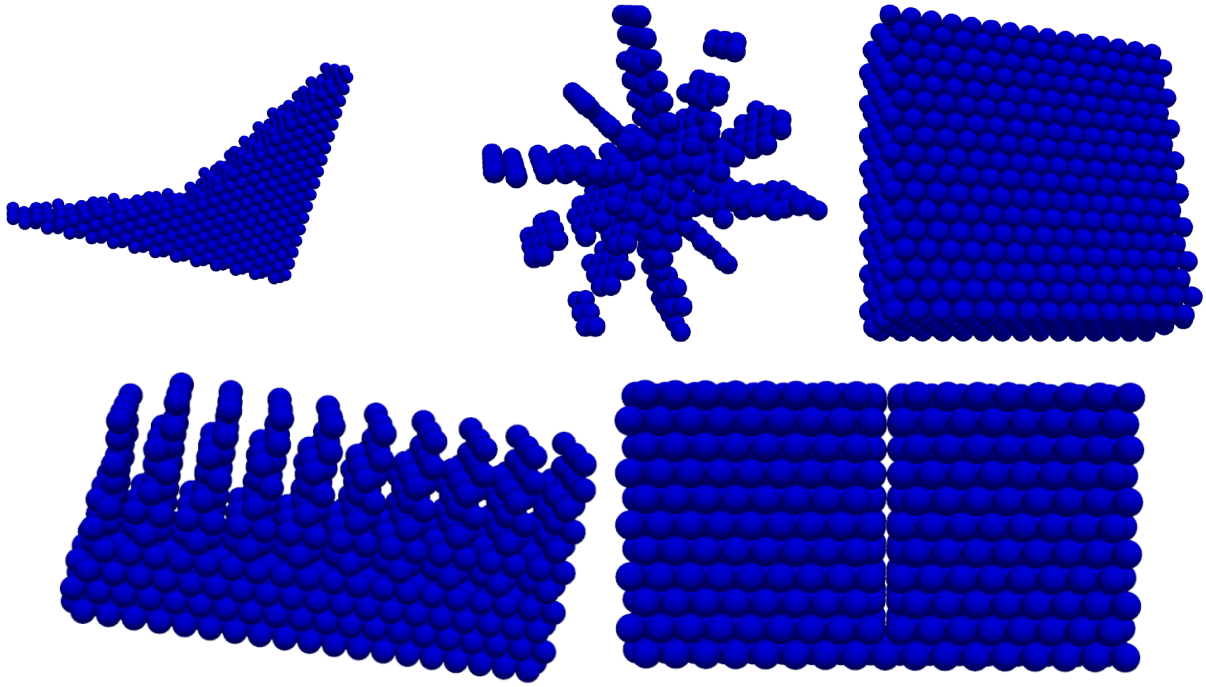
Disclaimer: Dies ist nicht die optimale Lösung [...]. Als erstes wird das Gesundheitszentrum optimal platziert.

Beispiele

· *Es wurden zahlreiche eigene Beispiele besiedelt. Hier beispielsweise Griechenland.* ·



Was allerdings fehlt, ist ein Siedlungsgebiet, das insofern realistisch ist, als dass es die Unförmigkeiten von Landesgrenzen oder Küstenverläufen abbildet. Dazu eignet sich das griechische Festland mit seinen zerklüfteten Küsten und konkaven Elementen hervorragend. [...] Seine Koordinaten sind im korrekten Maßstab angegeben [...]



· Ein anderer Teilnehmer entführt uns ins Weltall und erstellt Beispiele für eine Planetenpackung im Weltall! ·

Kritisches Hinterfragen der Ergebnisse

Aufgrund der Mathematischen Berechnungen sowie Rundungen entsteht ein Rechenfehler von max. $0.000001 = 1mm$, an diesen Stellen, wo die Dörfer zu nahe aneinander liegen, sollte dann der Verputzer gebeten werden, sparsam mit dem Putz umzugehen.

Erläuterungen zum Code

Der Dealer führt dann mit diesen Informationen mithilfe von vielen if- und for-schleifen die oben beschriebene Verteilungsstrategie durch.

Als ich jedoch die Anzahl der Tücher in der anderen Beispielausgabe reduzierte, funktionierte es.

· Algorithmennamen: · $Lime^{TM}$, $MinMax^{TM}$, $Boxer^{TM}$

[...], was aber nur bedingt gelingt und zu einem Parameterchaos führt.

Die einzige sich wahrhaftig verändernde Variable.

Es ist nicht zu einer erfolgreichen Implementierung gekommen, also entschuldige ich mich zutiefst bei dem Bewerter, der sich das alles durchlesen muss.

Danach gehe ich mit einer for-Schleife durch alle Boxen der Liste kaputt [...].

Die Funktion fillcircle teilt den großen Kreis in Zeilen auf, welche den geringstmöglichen Anstand haben.

Das Programm wurde in C++ implementiert. [Überschrift des nächsten Kapitels: main.py]

Dies ist auch gleich ein Sonderfall, da $0 = -0$

Aufgrund der Eigenarten des Programms wird es irgendwann versuchen [...].

Begründungen

Ich habe diesen Algorithmus gewählt, da er relativ einfach zu implementieren ist und ich Zeitmangel hatte (durch schlechtes Zeitmanagement).

In der Implementation sind viele Fehler, also werde nicht genauer auf diese eingehen.

Falls sich in der Idee ein Fehler eingeschlichen hat tut mir das sehr leid, es ist zuerst die Umsetzung und dann die Formulierung für die Idee entstanden)

Dies würde sich aber leicht beheben lassen und ist für diese Aufgabenstellung nicht notwendig genauer betrachtet zu werden.

Da dieser Beweis einzigartig sein soll, wurde[n unnötig komplizierte Dinge getan.]

Doch immer wenn ich gerade einen Sonderfall abgedeckt hatte, fielen mir zwei neue ein. „KEIN BOCK MEHR DAS KOMMT WEG“ - auch diese Option ist ausgeschieden.

(weil viel Arbeit und brain-aua)

· *Unterüberschrift:* · „. . . Und was, wenn nicht?“

Ich bin mir dieses Nachteils bewusst und nenne ihn deswegen hier so explizit.

Der letzte Edge Case ist der Grund warum die Hauptidee nicht für alle Eingaben funktioniert. [. . .] Bei diesem Fall kann man direkt die Hauptidee anwenden und kommt auf die Lösung.

Ich habe mich dazu entschieden, den Zufall entscheiden zu lassen, da es keine schlüssige Begründung gab, welche die Entscheidung hätte übernehmen können.

Dies liegt daran, dass das Zielfeld die vertikale und diagonale Spiegelachse außer Kraft setzt und Äquivalenzklassen nur noch aus 2 statt bestehen statt der 8 für dann, wenn kein Zielfeld vorgegeben ist.

Laufzeitanalysen

Dieser Prozess läuft solange, bis kein Speicher mehr zu Verfügung steht. Das Programm würde theoretisch dauerhaft laufen. Das, was es zeitlich begrenzt, ist die Speichermöglichkeit des eigenen Computers. In meinem Fall läuft das Programm für 15˘30 Sekunden, bevor es wegen Speichermangel abstürzt.

[. . .] lässt sich sicher sagen, dass $O(n) < O(m)$.

Die Laufzeit wurde mit einem Surface Pro 7 mit einem intel $i5-1035G4$ und 8GB LPDDR4$\times$ Ram unter dem Betriebssystem Linux(Debian) durchgefuehrt.

Ein Polynom fünften Grades fittet die Laufzeit sehr gut, also ist es $O(n^5)$!

In der Realität übersteigt sie [die Laufzeit] jedoch die Linearität, [. . .] allerdings in konstanter Natur.

[Es] ergibt sich folgende Laufzeit für das Programm für diesen Algorithmus: $O(So*St+St+2*So*St+So*St+St*(StlogSt+So+Kl+So+Kl+So+Kl+So+St+Kl*So*St+So*Kl)+Kl*So+Kl*So*Kl*Kl+Kl*So+Kl*So+100*(Kl*So+(Kl*(So+3*(So+So+Kl))+So+Kl*Kl*(So+So))+Kl*So*Kl*Kl*St)$. · *Anschließend wird P = NP vermutet.*
.

Ratgeber

Nachdem ich mich bei meinem einzigen sozialen Kontakt ChatGPT-Chan ausgeheult habe, [...].

Die svg kann das Programm selbst generieren, aber es hat keinen Dateinamenparameter, also passen Sie bei dem Ausführen auf, dass sie nicht Ihre Bitcoin in einer data.svg gespeichert haben.