Institut für Formale Methoden der Informatik

Abteilung Algorithmik

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3706

Abdeckung von Verschnittresten unter Konnektivitätsbedingungen

Markus Hildinger

Studiengang:	Dipl. Informatik
Prüfer:	Prof. Dr.-Ing. Stefan Funke
Betreuer:	Prof. Dr.-Ing. Stefan Funke
begonnen am:	17.12.2014
beendet am:	17.06.2015
CR-Klassifikation:	D2.2, D2.3, D2.6

Abstract

In vielen industriellen Anwendungsgebieten sind geometrische Packungsprobleme zu lösen, so möchte man zum Beispiel bei der Verarbeitung von Blech den Verschnitt minimieren:
Gegeben ist eine Menge von Blechteilen, von denen möglichst viele auf einem größeren Blech verteilt werden. Die verbleibenden Blechreste sollen dann noch von der Arbeitsfläche entfernt werden. Typischerweise geschieht dies durch „Wegstanzen“ der Blechreste. Es stehen hierfür verschieden Stanzköpfe zur Verfügung. Ziel ist es, mit möglichst wenig Stanzvorgängen und möglichst wenig Wechseln des Stanzkopfes alle Blechreste zu entfernen.
Hierbei gilt es zwei Dinge zu beachten:

- die verbleibende Restfläche sollte aus Stabilitätsgründen zusammenhängend bleiben

- beim Stanzvorgang muss mindestens die Hälfte der Stanzkopffläche auch wirklich mit Material unterlegt sein (d.h. Ausschließliche Benutzung des größten Stanzkopfes ist nicht immer möglich)

Im Rahmen dieser Diplomarbeit wurde von Grund auf ein Verfahren entwickelt, welches dieses Problem zu lösen versucht. Um die Komplexität des Problems zu reduzieren wurden einige Einschränkungen bezügliche der Probleminstanzen vorgenommen. So wurde festgelegt, dass nur polygonale Wertstücke vom Blech entfernt werden dürfen und die Form der zur Auswahl stehenden Stanzköpfe muss rund sein.
Der Hauptaugenmerk der Arbeit liegt auf der Beachtung der Nebenbedingungen. Vor allem für die Sicherstellung des Zusammenhangs der Fläche wurden Ideen entwickelt um umgesetzt. Hierbei spielt die mediale Achse eine besondere Rolle, indem sie als Grundlage für einen Großteil der vorgestellten Verfahren eingesetzt wird. Neben dem Einsatz für die Konnektivitätstest dient sie zusätzlich als Struktur für eine vereinfachte Darstellung einer Fläche. Besondere Merkmale des zu überdeckenden Gebiets können so besser erkannt und genutzt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Wie in den meisten Bereichen der Industrie wird auch bei der Verarbeitung von Blechen danach gestrebt, das menschliche Eingreifen in den Ablaufprozess zu minimieren. So wird beispielsweise beim Einsatz von Stanzmaschinen versucht, die Berechnung der Stanzpositionen und Stanzschritte komplett in die Hand der Maschine zu legen. Dies beginnt bei der Berechnung einer möglichst optimalen Packung der Bauteile, sodass die Verschnittreste minimiert werden und endet bei der vollständig maschinellen Entsorgung dieser Reste, durch zerstanzen der übrigen Fläche.
Beide Probleme können als Optimierungsprobleme formuliert werden und sind eng miteinander verwandt. Beim Packungsproblem wird versucht, möglichst viele Formen auf einer Fläche unterzubringen, die sich nicht gegenseitig überschneiden dürfen (Maximierungsproblem). Beim Überdeckungsproblem hingegen muss die komplette Fläche überdeckt werden, wobei versucht wird die Anzahl der Schritte so gering wie möglich zu halten (Minimierungsproblem). Der Zusammenhang der beiden Verfahren wird klar, wenn man eine jeweils optimale Lösung der beiden Probleme betrachtet. Beim Packungsproblem wäre eine Lösung optimal, wenn die komplette Fläche durch die Werkstücke abgedeckt werden könnte. Wenn nun alle verwendeten Formen die gleiche Größe besitzen, ist diese Lösung auch optimal für das Überdeckungsproblem.
Als Vorbereitung auf diese Arbeit wurde eine Studienarbeit angefertigt, die sich mit dem ersten Teil des Prozessablaufs befasst, dem Packungsproblem [Hildi14]. In dieser Arbeit wird der zweite Teil, also das Überdeckungsproblem behandelt.

Im folgenden Beispiel wurden zunächst die weißen Teile aus der Fläche ausgestanzt. Ziel ist es, den schwarzen Bereich mit möglichst wenig Stanzvorgängen, z.B. eines kreisförmigen Stanzkopfes, in verschiedenen Größen, zu entfernen.

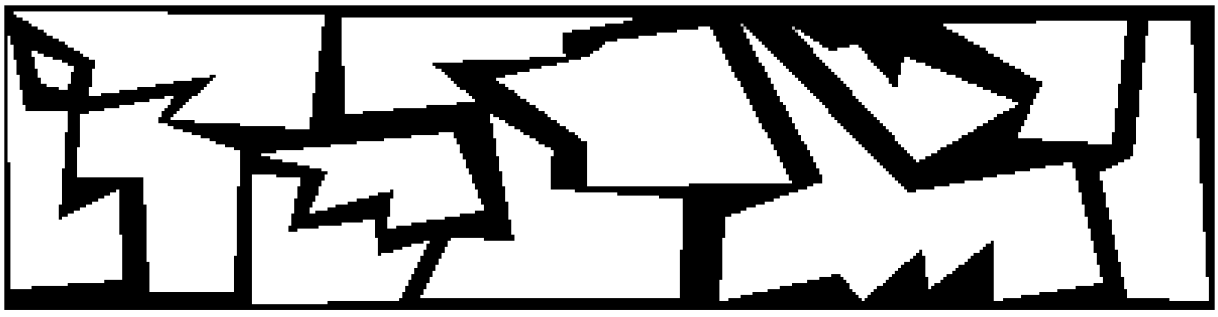

Abb. 1: Bsp. einer Probleminstanz für einen Überdeckungsalgorithmus

Überdeckungsprobleme:

Allgemein formuliert wird bei einem Überdeckungsproblem die Frage gestellt, ob eine, aus kleinere Teilen zusammengesetzte Menge oder Struktur eine andere, größere Menge oder Struktur komplett abdeckt. Ziel der Optimierungsfunktion ist es, eine möglichst kleine Anzahl von Teilmengen oder Teilstrukturen zu finden, für die dies zutrifft.

Überdeckungsprobleme (engl. covering problems) gehören zur Klasse der NP-vollständigen Probleme. Dies bedeutet, es gibt keinen effizienten Algorithmus (Polynomialzeit-Algorithmus), der in jedem Fall eine optimale Lösung für jede Probleminstanz liefern kann, es sei denn, es kann bewiesen werden, dass die Komplexitätsklasse NP gleich der Komplexitätsklasse P ist. Bisher gibt es weder einen Beweis für die Gleichheit noch einen dagegen. Da für keines der vielen bekannten NP-schweren Probleme bislang ein effizienter Algorithmus gefunden worden ist, liegt die Vermutung jedoch nahe, dass die Komplexitätsklassen nicht gleich sind [Vazi01].
Ein oft eingesetztes Verfahren bei Optimierungsproblemen ist die lineare Programmierung. Hier besteht eine interessante Beziehung zu Überdeckungsproblemen. So kann jedes Integer Lineare Programm mit ausschließlich positiven Constraints und einer positiven Optimierungsfunktion als ein Überdeckungsproblem betrachtet werde [Vazi01].

Literatur:

Das wohl bekannteste Überdeckungsproblem ist das Set Cover Problem, bei dem eine Menge U und eine bestimme Anzahl von Teilmengen T_i von U gegeben sind. Hierbei gilt es, eine möglichst kleine Anzahl von Teilmengen T_i auszuwählen, sodass alle Elemente aus U in mindestens einer der ausgewählten Teilmengen vorhanden sind.
Das hier vorgestellte Problem kann, sofern die Randbedingungen keine Berücksichtigung finden, als ein solches Set Cover Problem dargestellt werden. Jeder Punkt der zu überdeckenden Fläche gilt hierbei als Element der Menge U. Die Teilmengen T_i stellen hierbei die bei einem gültigen Stanzschritt überdeckten Punkte dar. Durch die Einführung der Randbedingungen lassen sich die zahlreichen für das Set Cover Problem entwickelten Algorithmen [Vazi01] [Card14] jedoch nicht übertragen.
Ein ebenfalls sehr verwandtes und viel beachtetes Problem ist das „Polygon Covering“ Problem. Hierbei muss eine polygonale Grundfläche durch kleinere Polygone überdeckt werden. In der Literatur beschränkt man sich hierbei meist auf einfache Formen wie Überdeckung eines rechtwinkligen Polygons mit Quadraten [Bar98], eines rechtwinkligen Polygons mit Rechtecken [Hein07] oder eines Polygons ohne Spitze Winkel (alle größer 90°) mit Rechtecken oder Quadraten [Levco97]. Für die Lösung von besonderen Klassen dieses Problems wurden bereits Linearzeit-Verfahren entwickelt [Bar98].

Besondere Aspekte dieser Arbeit:

Wie oben bereits erwähnt, besteht die zentrale Aufgabe in der Überdeckung der Restfläche, die zurückbleibt, nachdem die Werkstücke aus dem Blech herausgestanzt wurden. Es handelt sich demnach um eine Flächenüberdeckung vergleichbar mit dem „Polygon Covering“. Da die Aufgabe aber in Anlehnung an das praktische Problem im Bezug auf Stanzmaschinen betrachtet wird, sind einige zusätzliche Nebenbedingungen zu beachten, welche eine zentrale Rolle in dieser Arbeit spielen werden:

- Minimieren der Stanzkopfwechsel
- Mindestens die Hälfte des Stanzkopfes muss tatsächlich mit Material unterlegt sein
- Die zu überdeckende Fläche muss aus Stabilitätsgründen zu jedem Zeitpunkt zusammenhängend bleiben.

Die erste Bedingung dient dazu, einen möglichst schnellen Prozessablauf zu ermöglichen. Jeder Wechsel eines Stanzkopfes wirkt sich hier negativ auf die Arbeitszeit aus. Aus Gründen der Sicherheit wurde die zweite Bedingung eingeführt, um ein mögliches Abknicken des Blechs zu verhindern. Der dritte Punkt dient ebenfalls der Prozesssicherheit, wobei hier das eventuelle Verrutschen von kleinen, losen Teilen der Restfläche verhindert werden soll.

Um die Komplexität des Problems im Rahmen zu halten, wurden einige Vereinfachungen vorgenommen. So wird die zu überdeckende Fläche als Rechteck definiert, aus welchem Polygone entfernt werden. Dadurch wird gewährleistet, dass zu Beginn des Algorithmus keine gekrümmten Kanten existieren. Außerdem wird die Auswahl an Stanzköpfen auf ausschließlich runde Stanzköpfe beschränkt.
Der vorgestellte Algorithmus soll also eine gegebene Fläche, bestehend aus einem Rechteck mit polygonalen Löchern, mithilfe von runden Stanzköpfen abdecken. Dabei sind die oben genannten Nebenbedingungen zu beachten.

2 Grundlagen

In diesem Kapitel wird die Durchführung des eigentlichen Überdeckungsalgorithmus vorbereitet. Zunächst werden die Parameter des Programms definiert, also diejenigen Informationen, die der Nutzer eingeben muss, um ein Problem zu beschreiben, welches vom Algorithmus gelöst werden soll. Des weiteren wird auf einen zentralen Aspekte eingegangen, der durch die diskretisierte Darstellung der Probleminstanz entsteht, und es werden Vorverarbeitungsschritte beschrieben, die essenziell für die Durchführung des Algorithmus sind. Schließlich wird auch noch auf den Umgang mit der zentralen Randbedingung, der Konnektivität, eingegangen.

2.1 Erstellung der Probleminstanz

Bevor das eigentliche Überdeckungsproblem gelöst werden kann, muss ein Blick auf die Probleminstanz geworfen werden. Unter einer Probleminstanz wird hier eine zu überdeckende Fläche sowie eine Menge an Stanzköpfen, die der Algorithmus nutzen kann, verstanden.
Die zu überdeckende Fläche wird, wie bereits beschrieben, auf Basis einer rechteckigen Grundfläche definiert. Zunächst muss deshalb die Größe dieser Grundfläche bestimmt werden. Da diese eine rechteckige Form haben soll, genügt es, hier die Ausdehnung in x- und in y-Richtung anzugeben. Die Fläche wird durch ein Rastergitter diskretisiert dargestellt, wobei die angegebenen Werte als Anzahl der Gitterzellen in die jeweilige Richtung verstanden wird. Die Distanzwerte werden anhand der Manhattan-Distanz [Royer01] berechnet.
Weiter wird nun die Form und die Position aller Werkstücke benötigt, die im vorangegangenen Arbeitsschritt aus dieser rechteckigen Grundfläche herausgestanzt wurden. Die Darstellung dieser Werkstücke als Polygone erlaubt hier die Übergabe mehrerer Punktlisten, von welchen jede die Eckpunkte eines Polygons in der richtigen Reihenfolge beinhaltet.
Mit Hilfe der Grundfläche und den Polygone kann nun die initiale Restfläche bestimmt werden, welche es vom Algorithmus zu überdecken gilt.

Definition Restfläche

> Die Restfläche beschreibt denjenigen Teil der Grundfläche, welcher noch von Material bedeckt ist, das es vom Algorithmus zu entfernen gilt.

Für das Entfernen der Polygone von der Grundfläche wurde ein kleiner Algorithmus erstellt, der speziell auf die hier gegebenen Bedingungen zugeschnitten wurde. Da dieser nicht direkt mit der Problemstellung in Verbindung steht, wird hier zunächst nicht näher darauf eingegangen. Eine detaillierte Beschreibung des Algorithmus findet sich in Kapitel 6.
Wie bereits erwähnt, werden zunächst nur runde Stanzköpfe zugelassen. Der Nutzer kann hier, ab-

hängig von der Größer der Grundfläche, verschieden Radien auswählen. Der größtmögliche Radius wird dabei immer so festgelegt, dass ca. 100 Quadrate mit einer Seitenlänge, die diesem Radius entspricht, komplett auf der Grundfläche untergebracht werden können. Der kleinste Radius wird immer Null sein, wobei ein Stanzkopf mit diesem Radius genau einen Punkt der diskretisierten Fläche überdeckt. Damit die Laufzeit im Rahmen bleibt, wird die maximale Anzahl der Stanzköpfe auf 20 reduziert.

2.2 Nachbarschaftsbeziehung

Durch die Diskretisierung der Fläche durch ein Rastergitter kann die Nachbarschaft zweier Punkte nicht eindeutig definiert werden. Es stehen hier zwei alternative Definitionen zur Verfügung, die beide in den vorgestellten Algorithmen verwendet werden.

Definition Vierer-Nachbarschaft

> Wird die Vierer-Nachbarschaft zu Grunde gelegt, gelten zwei Punkte als benachbart, wenn sie sich eine gemeinsame Kante in der Gitterstruktur teilen. Abgesehen von den Randpunkten besitzt somit jeder Punkt genau vier Nachbarpunkte.

Definition Achter-Nachbarschaft

> Wird die Achter-Nachbarschaft zu Grunde gelegt, gelten zwei Punkte als benachbart, wenn sich ihre Gitterzellen an einem oder an mehreren Punkten berühren. Abgesehen von den Randpunkten hat somit jeder Punkt genau acht Nachbarpunkte.

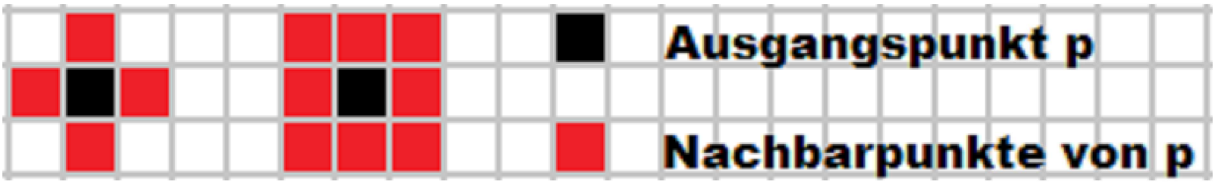

Abb. 2: Bsp. Nachbarschaftsdefinition; links Vierer-, rechts Achter-Nachbarschaft

2.3 Mediale Achse

Eines der ersten Probleme, das sich bei näherer Betrachtung der Aufgabenstellung zeigt, stellt die zu überdeckende Restfläche dar. Diese Fläche kann mehr oder weniger beliebige Formen annehmen, was ein systematisches Vorgehen nahezu unmöglich macht. Der menschliche Betrachter kann in einer Fläche womöglich Regelmäßigkeiten oder Strukturen feststellen, welche für den Algorithmus wichtig oder zumindest hilfreich sein könnten. Diese Strukturen existieren aber in den allermeisten Fällen auf einer höheren Ebene, sodass es sehr schwer, wenn nicht gar unmöglich ist, diese auf algorithmischem Weg zu erfassen.

Der erste Schritt ist deshalb, die Fläche auf eine möglichst einfache und intuitive Form abzubilden, welche auch maschinell verarbeitet werden kann. Die Medial Achse, die im Jahr 1967 von Harry Blum [Blum67] zuerst vorgestellt wurde, bietet sich hier an.

Definition mediale Achse nach Blum

> In einer zweidimensionalen Fläche F, besteht die mediale Achse MA_F aus den Mittelpunkten aller maximaler Kreise in F. Ein Kreis K ist genau dann ein maximaler Kreis in F, wenn er vollständig in F liegt und an mindestens zwei Stellen tangential den Rand der Fläche berührt.

Die mediale Achse kann demnach als eine Art Mittelpunktlinie eines zweidimensionalen Gebietes bezeichnet werden. Um sie ein wenig anschaulicher zu machen, wird nun die dritte Dimension zur Hilfe genommen. Man stelle sich eine Tischplatte in Form der Fläche F vor. Auf dieser Tischplatte soll nun sehr feiner, trockener Sand aufgeschüttet werden, und zwar so viel wie möglich. Die daraus resultierende Bergstruktur besitzt in einer perfekten Umgebung die Eigenschaft, das die Höhe des Sandes proportional zur Entfernung vom Rand der Fläche ist. Genau an den Stellen, an denen die Entfernung zum Rand nicht eindeutig definiert ist, verzeichnet diese Bergstruktur eine Gratlinie. Wird der Verlauf all dieser Gratlinien nun senkrecht auf die Tischplatte projiziert, bilden sie exakt den Verlauf der medialen Achse nach. Aus diesem Grund werden die Teilsegmente der medialen Achse im folgenden oft auch als Gratlinien bezeichnet.
Mit der medialen Achse besitzt man nun ein Hilfsmittel, welches die Struktur der Fläche sehr schön wiedergibt und algorithmisch wesentlich einfacher zu handhaben ist. Hat die Fläche beispielsweise eine spitze Ecke, so wird diese durch den Endpunkt einer Gratlinie definiert. Werden zusätzlich die Distanzwerte zum Rand gespeichert, so lässt sich anhand der medialen Achse ebenfalls ablesen, wie breit die Fläche an der jeweiligen Stelle ist.
Sucht man nun für die Fläche geeignete Punkte, die sich als Positionen für einen Stanzschritt anbieten, so kommt man ebenfalls nicht umhin, die mediale Achse zu beachten. Somit wird mit dieser Struktur eine sehr einleuchtende Vorauswahl an möglichen Stanzpositionen mitgeliefert. Zusätzlich bietet sie, unter bestimmten Voraussetzungen, ein Hilfsmittel für den Konnektivitätstest, welcher im Endeffekt einen Großteil der Laufzeit des vorgestellten Algorithmus ausmachen wird.

2.3.1 Diskretisierte mediale Achse

Durch die Diskretisierung der Fläche kann die mediale Achse nicht exakt dargestellt werden. Deshalb wird im folgenden eine Definition für die mediale Achse gegeben, die mit der Diskretisierung des Feldes vereinbar ist. Gleichzeitig wird uns eine intuitive Möglichkeit für die Berechnung der medialen Achse mitgeliefert.

Definition mediale Achse (alternativ)

> Ein Punkt, der Teil der medialen Achse MA_F der Fläche F ist, besitzt keinen eindeutig definierten kürzesten Pfad P zum Rand der Fläche. Es existieren immer mindestens zwei solcher Pfade. Ein Pfad ist hier zunächst als ein gerades Liniensegment definiert.

Der Zusammenhang der beiden Definitionen ist einfach zu erkennen, indem der Radius eines maximalen Kreises um den Punkt P mit der Länge des Kürzesten Pfades von P zum Rand gleichgesetzt wird. Da der Kreis den Rand der Restfläche an mindestens zwei Stellen berührt,

gibt es auch mindestens zwei kürzeste Pfade.
Nun gilt es, die mediale Achse für eine diskretisierte Fläche zu bestimmen. Hierzu muss zunächst eine Definition für einen kürzester Pfad in einer diskretisierten Fläche gegeben werden.

Definition kürzester Pfad in einer diskretisierter Fläche

> In einer diskretisierten Fläche wird ein Pfad P von Punkt A nach Punkt B durch eine Liste von Punkten dargestellt, für die gilt: $P_{(First)} = A$, $P_{(Last)} = B$, $P_{(i+1)}$ und $P_{(i)}$ stehen in Vierer-Nachbarschaft zueinander. Ein Pfad P heißt kürzester Pfad, wenn für alle Pfade Q gilt, dass die Länge von Q ≥ der Länge von P ist.

Die Kombination der beiden letzten Definitionen reicht jedoch noch nicht aus. Zunächst kann festgestellt werden, dass sehr viele Punkte der Fläche fälschlicherweise als Teil der medialen Achse erkannt werden können. In Abb.3 stellen die schwarzen Punkte ein diskretisiertes Liniensegment dar. Die grünen Punkte werden nach der bisherigen Definition alle als Punkte der medialen Achse erkannt, da für jeden der Punkte zwei kürzeste Pfade berechnet werden können. Diese Pfade verlaufen genau gleich, bis am roten Punkt der eine Pfad nach unten weiter läuft, während der andere Pfad nach rechts geht. Diese Problem lässt sich umgehen, indem gefordert wird, dass die Endpunkte der Pfade zu unterschiedlichen Randsegmenten gehören müssen.

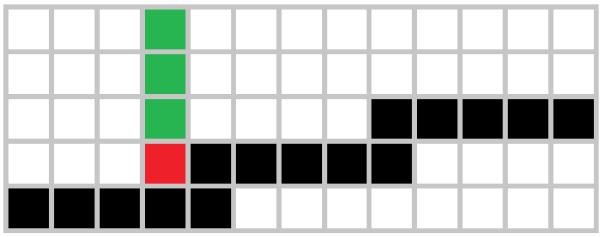

Abb. 3: Bsp. für eine falsch erkannte mediale Achse

Ein weiteres Problem ergibt sich, wenn die mediale Achse genau zwischen zwei diskretisierten Punkten verläuft, da keiner der beiden Punkte als Teil der medialen Achse erkannt werden kann. Um die mediale Achse sinnvoll nutzen zu können, darf sie jedoch keine Lücken aufweisen. In solchen Fällen muss also einer der beiden Punkte in die mediale Achse aufgenommen werden. Da nicht einheitlich definiert werden kann, welcher der beiden Punkte aufgenommen wird, kann dies zu Schlangenlinien führen, wie in Abb.4 illustriert wird.

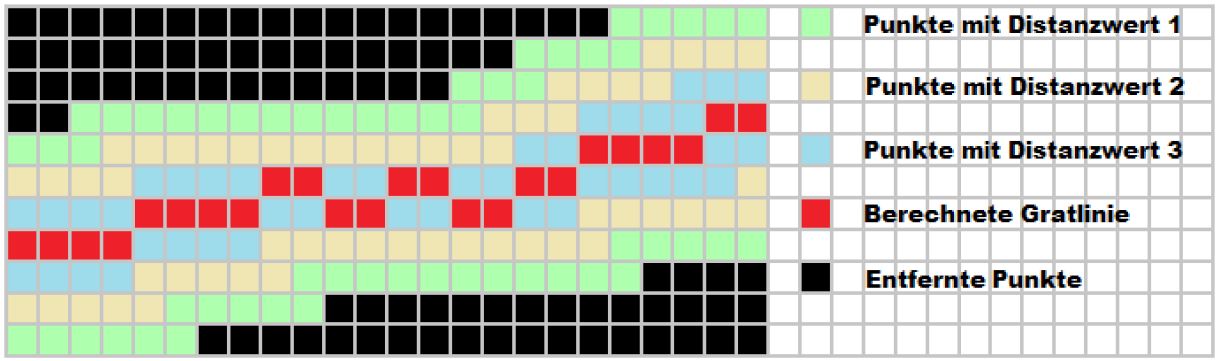

Abb. 4: Berechnete Gratlinie verläuft in Schlangenlinien aufgrund der Diskretisierung

2.3.2 Berechnung der medialen Achse

Für die Berechnung der medialen Achse wird hier nun auf die alternative Definition zurückgegriffen. Ein Punkt ist demnach also genau dann Teil dieser Achse, wenn er zwei kürzeste Pfade zu unterschiedlichen Randsegmenten besitzt. Für jeden Punkt der Restfläche muss deshalb die Länge des kürzesten Pfades zum Rand gespeichert werden. Hierfür definieren wir den Distanzwert eines Punktes.

Definition Distanzwert eines Punktes

> Der Distanzwert eines Punktes ist gleich der Länge des kürzesten Pfades zum Rand der Fläche.

Es bieten sich zwei alternative Methoden für die Berechnung der mediale Achse an: Eine globale und eine lokale. Bei der globalen Methode, wird die mediale Achse für die gesamte Fläche im Ganzen berechnet. Da sich bei einer lokalen Änderung der Fläche die mediale Achse auch nur lokal ändert, kann eine komplette Neuberechnung vermieden werden, indem eine Methode zur lokalen Berechnung verwendet wird. In dieser Arbeit wird die globale Berechnung nur einmal bei der Initialisierung der rechteckigen Grundfläche durchgeführt. Das macht diese Berechnung sehr einfach, weshalb hier nicht weiter darauf eingegangen wird. Alle sonstigen Änderungen an der medialen Achse werden nur lokal berechnet.

Wird nun also ausgehend von einem neu initialisierten Feld oder einer sonstigen beliebigen Restfläche eine Figur herausgestanzt, ändern sich die Distanzwerte von genau denjenigen Punkten, die näher an dieser Figur liegen als am bisherigen Rand der Restfläche. Da die Distanzwerte dieser Punkte ohnehin geändert werden müssen, bietet sich folgendes Vorgehen an: Vom Rand der Figur ausgehend wird Schritt für Schritt nach außen gearbeitet und die Distanzwerte der Punkte werden so lange aktualisiert, bis der Algorithmus an einem Punkt angelangt ist, an dem der nächste Nachbarpunkt einen niedrigeren Distanzwert aufweist als der aktuelle Punkt. In diesem Fall wird der aktuelle Punkt als Gratpunkt gekennzeichnet und der entsprechende Nachbarpunkt nicht weiter bearbeitet. Im Falle eines Stanzvorgangs mit einem konvexen Stanzkopf ist hiermit alles getan.

Bei nicht konvexen Polygonen hingegen existieren Eckpunkte am Polygon, deren Innenwinkel größer als 180° ist. Dementsprechend besitzt die Restfläche an dieser Stelle eine spitze Ecke, an der ein Endpunkt einer Gratlinie liegen muss. Die dazugehörige Gratlinie wird allerdings beim bisherigen Vorgehen nicht berücksichtigt. Das Problem hierbei ist, dass das Polygon als Einheit betrachtet wird und nicht zwischen den einzelnen Segmenten des Polygons unterschieden werden kann. Aus diesem Grund wird eine Gratlinie, die von zwei unterschiedlichen Segmenten des selben Polygons definiert wird, vom Algorithmus nicht als solche erkannt. Ein nicht konvexes Polygon muss demnach gesondert betrachtet werden.

2.3.3 Vollendung der Gratlinien bei nicht konvexen Polygonen

Die Vollendung der Gratlinien kann in zwei Schritte unterteilt werden. Im ersten Schritt müssen zunächst alle Eckpunkte des Polygons mit einem Innenwinkel größer als 180° gefunden werden, da in jedem dieser Eckpunkte der Endpunkt einer Gratlinie liegen wird. Von diesen Eckpunkten ausgehend können die Gratlinien nun rekursiv berechnet werden, indem der

Nachbarpunkt mit dem höchsten Distanzwert als nächster Punkt der Gratlinie gewählt wird. Dies wird fortgeführt, bis ein bereits berechneter Punkt der medialen Achse erreicht wird oder kein Nachbarpunkt gefunden werden kann, dessen Distanzwert größer oder gleich dem des aktuellen Punktes ist.
Durch die Diskretisierung des Feldes kann jedoch nicht immer ein eindeutiger Nachbarpunkt gefunden werden. Deshalb wird zusätzlich die Richtung, in der die Gratlinie verläuft, eine entscheidende Rolle bei der Bestimmung des nächsten Punktes spielen. Mit Hilfe der Richtung und der Distanzwerte kann nun ein eindeutiger nächster Gratpunkt bestimmt werden, womit jedoch noch nicht alle Fälle abgedeckt sind.

Hat das Polygon beispielsweise die Form eines Ringes, der an einer Stelle durchtrennt und leicht auseinander gebogen wurde, sodass ein breiter Bereich im Innern entsteht, der nur durch einen engen Flaschenhals mit dem Rest der Restfläche verbunden ist, so wird dieser Ausgang meist nicht gefunden.

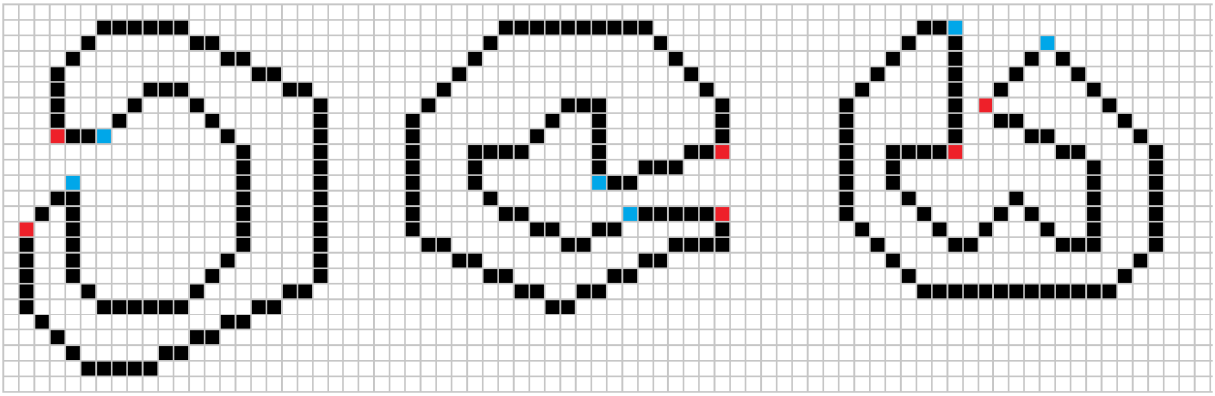

Abb. 5: Polygone mit engen Öffnungen, farbige Eckpunkt-Paare definieren Engstelle

Die Idee zur Lösung dieses Problems ist, dass eine solche Engstelle immer von einem oder mehreren Eckpunkt-Paaren des Polygons definiert wird. Wird also festgestellt, dass die Gratlinien mit der obigen Methode nicht vollendet werden können, müssen diese Engstellen gefunden werden. Hierzu werden nun Verbindungslinien zwischen allen Punkten des Polygons gezogen, sofern diese Linien nicht durch das Innere des Polygons verlaufen. Es genügt hierbei nur Punkte mit einem Innenwinkel von weniger als 180° zu betrachten. Jede dieser Verbindungslinien muss per Definition eine Gratlinie schneiden. Ist dies nicht der Fall, kann entlang der Linie derjenige Punkt mit dem höchsten Distanzwert als ein Punkt der Gratlinie definiert werden. Sobald dann die Richtung der Gratlinie in diesem Punkt bestimmt ist, kann von hier ausgehend, nach dem oben beschriebenen, rekursiven Prinzip, in beide Richtungen die Gratlinie gezeichnet werden.

2.4 Randbedingung Konnektivität

Eine der zentralen Randbedingung, welche in dieser Arbeit betrachtet wurde, ist die Konnektivität. Es gibt hierbei zwei mögliche Betrachtungsweisen, welche beide ihre Berechtigung haben. Entweder soll die Restfläche zu jedem Zeitpunkt in einem Stück zusammenhängend sein oder die Umgebung des Feldes wird als Teil der Restfläche betrachtet, womit die Bedingung der Konnektivität darauf reduziert werden kann, dass von jedem Punkt der Restfläche eine

Verbindung zum Rand des Feldes existieren muss. Im Folgenden wurde Letztere als Konnektivitätsbedingung verwendet. Die Algorithmen können bei Bedarf allerdings relativ einfach so abgeändert werden, dass sie für die alternative Bedingung anwendbar sind. Da aufgrund der Wahl der Konnektivitätsbedingung die zu bearbeitende Restfläche nicht mehr zwangsläufig an einem Stück sein muss wird, nun ein Teilgebiet definiert.

Definition Teilgebiet

> Eine Teilgebiet TG ist ein zusammenhängender Teil der vom Algorithmus zu überdeckenden Restfläche. Teilgebiete müssen vollständig sein, dass heißt, es darf kein Pfad, entlang noch existierender Punkte der Restfläche, zwischen zwei unterschiedlichen Teilgebieten existieren. Alle Teilgebiete zusammengenommen ergeben die Menge der Punkte, die noch überdeckt werden muss.

Als Ausgangsbedingung wird angenommen, dass die Restfläche zusammenhängend ist. Dies bedeutet, dass für alle Teilgebiete eine Verbindung zum Rand des Feldes existiert. Sollte dies nicht der Fall sein, können die einzelnen, nicht zusammenhängenden Restflächen getrennt von einander betrachtet werden.
Der in dieser Arbeit vorgestellte Algorithmus wird rundenbasiert ablaufen. Da die Konnektivität zu Beginn gegeben ist, gilt es für jeden Einzelschritt zu zeigen, dass dessen Durchführung die Konnektivität der Restfläche nicht zerstört. Da ein solcher Einzelschritt nur lokale Veränderungen bewirkt, genügt es in manchen Fällen, nur das lokale Umfeld zu betrachten, um die Konnektivität zu garantieren. Somit muss nur in den restlichen Fällen die komplette Restfläche untersucht werden.
Es wurden hierfür zwei sehr ähnliche Algorithmen implementiert. Der Algorithmus basierend auf der medialen Achse bietet, vor allem bei denjenigen Fällen, bei denen das lokale Umfeld keine Schlüsse über die Konnektivität zulässt, einen erheblichen Performance-Vorteil. Gleichzeitig werden hier jedoch einige Forderungen an die Probleminstanz gestellt, die vor allem bei eventuellen Erweiterungen nicht mehr unbedingt gewährleistet werden können. In Kapitel 7 wird hierauf näher eingegangen.

2.4.1 Algorithmus ohne mediale Achse

Zunächst wird eine lokale Bedingung gesucht, die die Konnektivität nach einem Stanzschritt gewährleistet. Hierfür definieren wir eine Menge von Punkten, die noch existierenden Nachbarpunkte.

Definition noch existierender Nachbarpunkt

> Ein noch existierender Nachbarpunkt NEN ist ein Punkt, der bezügliche der Achter-Nachbarschaft am Stanzkopf anliegt und nach dem Stanzschritt noch Teil der Restfläche sein wird.

Da nur runde Stanzköpfe benutzt werden, bilden die NEN Kreissegmenten entlang des Stanzkopfrandes. Nun lässt sich sagen, dass die Restfläche auch nach dem Stanzschritt zusammenhängend bleiben wird, wenn sich alle NEN zu einem einzigen Kreissegment zusammenfügen lassen. Sofern die Konnektivität vor dem Stanzschritt nicht verletzt war, kann sie auch nach danach nicht verletzt sein, da alle NEN zum gleichen Teilgebiet gehören.

Aufgrund der Festlegung der Konnektivitätsbedingung gibt es hierbei einen kleinen Sonderfall zu beachten. Obwohl die lokale Bedingung nicht verletzt wurde und alle NEN zum gleichen Teilgebiet gehören, kann es geschehen, dass durch den Stanzschritt die einzige Verbindung dieses Teilgebiets zum Rand gekappt wird. Dieser Fall kann dann eintreten, wenn ein kompletter Randabschnitt vom Stanzkopf überdeckt wird. Sollte der Stanzkopf also über den Rand hinausragen, muss die lokale Suche erweitert werden.
Ist einer der Punkte, die bei diesem Stanzschritt entfernt werden, ein Randpunkt der Grundfläche, so muss zusätzlich sichergestellt werden, dass mindestens einer der NEN ebenfalls ein Randpunkt der Grundfläche ist. Sollte dies nicht der Fall sein ist klar, dass ein Randabschnitt komplett entfernt wurde. Dies bedeutet dann, dass für das entsprechende Teilgebiet eine andere Verbindung zum Rand existieren muss, damit die Konnektivität erhalten bleibt.

Kommt es nun zu einer Situation, in der die lokale Bedingung verletzt wird, können drei Fälle unterschieden werden:

1) Die nicht zusammenhängenden Kreissegmente, die durch die NEN definiert werden, gehören immer noch zum gleichen Teilgebiet

2) Durch den Stanzschritt wurde eine Teilgebiet in zwei oder mehrere Teilgebiete aufgespalten, sodass die Kreissegmente zu verschiedenen Teilgebieten gehören.

3) Der Sonderfall ist eingetreten, sodass ein Teilgebiet möglicherweise keine Verbindung mehr zum Rand besitzt

Nun gilt es eine globale Bedingung festzulegen, die für alle drei Fälle anwendbar ist. Klar ist, dass von jedem NEN eine Verbindung zum Rand der Grundfläche existieren muss. Da jedoch die NEN bereits zu Kreissegmenten zusammengefügt wurden, genügt es, für jedes der Kreissegmente eine solche Verbindung zu finden. Sollte bei dieser Suche, zusätzlich zum Rand der Grundfläche, eines der anderen Kreissegmente gefunden werden, so muss für dieses natürlich keine eigene Suche mehr gestartet werden.

2.4.2 Algorithmus basierend auf der medialen Achse

Wie bereits erwähnt, kann die mediale Achse als Hilfsmittel verwendet werden, um die Konnektivität der Fläche zu gewährleisten. Um die Korrektheit der Idee zu verstehen, betrachten wir zunächst, welche Formen die Teilgebiete der Restfläche annehmen können.

Definition Extremität der Restfläche

> Als eine Extremität der Restfläche wird ein zusammenhängender Teil dieser Fläche bezeichnet, der durch einen Schnitt entlang einer Geraden, die vollständig innerhalb der Restfläche verläuft, vom Rest der Restläche abgetrennt werden kann.

Es gilt nun zu zeigen, dass es in keinem Fall eine Extremität der Fläche geben kann, welche nicht von einer Gratlinie durchlaufen wird. Zu Beginn haben wir gefordert, dass das Feld nur polygonale Löcher enthalten darf. Ergo sind alle Begrenzungen der Restfläche zunächst gerade Segmente. Die einzigen Extremitäten der Restfläche werden also definiert durch die Eckpunkte der Grundfläche, sowie durch diejenigen Eckpunkte der Polygone mit einem Innenwinkel von mehr als 180°. Für beide Fälle wird vom Algorithmus eine Gratlinie erstellt.

Solange nun alle verwendeten Stanzköpfe eine konvexe Form besitzen, was im Falle von ausschließlich runden Stanzköpfen der Fall ist, können neue „Extremitäten“ der Restfläche nur an Stellen entstehen, an denen ein Stanzkopf über den aktuellen Rand der Restfläche hinausragt. Da auch an diesen Stellen eine Gratlinie erzeugt wird, ist die geforderte Bedingung zu jeder Zeit erfüllt. Aufgrund dieser Eigenschaft der Restfläche und der Tatsache, dass nur konvexe Stanzköpfe genutzt werden, lässt sich nun eine lokale Kennektivitätsbedingung wie folgt formulieren:

1) Liegt kein Punkt, der Teil der medialen Achse ist, unter dem Stanzkopf, bleibt die Konnektivität erhalten.

2) Liegt der Stanzkopf über der medialen Achse, genügt es, entlang der Nachbarpunkte des Stanzkopfs eine Verbindung zwischen denjenigen NEN zu finden, die bisher Teil der medialen Achse waren. Alle anderen Nachbarpunkte müssen in diesem Fall nicht mehr beachtet werden.

Zu 1) : Der Zusammenhang der Restfläche wird genau dann zerstört, wenn ein Teil der Restfläche vom Rest abgetrennt wird. Um dies mit einem konvexen Stanzkopf zu erreichen muss es sich bei dem abgetrennten Bereich um eine Extremität der Fläche handeln. Da alle Extremitäten von einer Gratlinie durchlaufen werden, kann bei einem Stanzschritt, bei dem keine Gratlinie durchtrennt wurde, auch nichts abgetrennt werden.

Zu 2) : Tritt dieser Fall ein, wurden eine oder mehrere Gratlinien durchtrennt. Es könnte somit sein, dass der Stanzschritt die Verbindung zu einem Teil der Restfläche, welche durch eine dieser Gratlinien definiert wurde, komplett durchtrennt hat. Existiert allerdings entlang der Nachbarpunkte eine Verbindung zwischen den NEN, die Teil der medialen Achse sind, bleibt auch in diesem Fall die Konnektivität gewahrt, da alle Gratlinien zum selben Teilgebiet gehören.
Der selbe Sonderfall, der beim einfachen Algorithmus bereits erwähnt wurde, muss auch hier beachtet werden. Die Lösung hierfür lässt sich jedoch eins zu eins übertragen, sodass nicht weiter darauf eingegangen werden muss.

Wird bei 2) keine Verbindung gefunden oder tritt der Sonderfall ein, muss, genau wie beim anderen Algorithmus, eine globale Suche gestartet werden. Hier kommt nun der eigentliche Vorteil der Gratlinien zum tragen. Während beim einfachen Algorithmus die komplette Restfläche durchsucht werden muss, hat man nun mit den Gratlinien einen vorgezeichneten Weg und ist in der Lage, wesentlich effizienter eine Verbindung zum Rand zu finden, falls eine solche existiert.

2.4.3 Änderung der Konnektivitätsbedingung

Sollte die alternative Konnektivitätsbedingung bevorzugt werden, müssen die Algorithmen leicht abgeändert werden. Bei der lokalen Bedingung ändert sich nichts, abgesehen vom beschriebenen Sonderfall, welcher hier keine Rolle mehr spielt.
Bei der globalen Suche sieht dies jedoch anders aus. Der Rand gilt nun nicht mehr als Teil der Restfläche und spielt daher keine Rolle mehr. Ebenso wird die Definition der Teilgebiete überflüssig, da die Konnektivitätsbedingung nun nur noch ein solches Teilgebiet zulässt, was per Definition genau der Restfläche entspricht.

Beim Algorithmus ohne Gratlinien kann die lokale Bedingung übertragen werden auf die globale Suche. Ziel ist es, eine Verbindung zwischen allen NEN zu finden. Kann dies lokal nicht erreicht werden, wird global nach einer solchen gesucht.
Beim Algorithmus basierend auf der medialen Achse müssen auch hier nur die Punkte der Gratlinien betrachtet werden. Die globale Bedingung lautet nun, dass entlang der Gratlinien, von jedem NEN, der Teil der medialen Achse ist, ein Weg zu jedem anderen von diesen NEN gefunden werden muss.

3 Überdeckungsalgorithmus

In diesem Kapitel wird der eigentliche Überdeckungsalgorithmus vorgestellt. Zunächst wird die Grundidee für das Vorgehen näher beleuchtet. Danach wird auf den Ablauf des Algorithmus eingegangen und die einzelnen Schritte werden detailliert beschrieben.

3.1 Greedy Strategie

Als Ausgangspunkt für den Lösungsalgorithmus wurde eine Greedy-Strategie gewählt. Der zentrale Aspekt einer solchen Strategie ist, dass schrittweise immer derjenige Folgezustand ausgewählt wird, der zum aktuellen Zeitpunkt den größten Gewinn verspricht. Um dies umsetzten zu können, müssen also Zustände und Zustandsübergänge definiert werden. Zusätzlich wird für die Entscheidung, welcher Folgezustand den größten Gewinn verspricht, noch eine Bewertungsfunktion benötigt.

Definition Zustand

> Ein zustand wird hier durch die Form der Restfläche und den zuletzt benutzten Stanzkopf bestimmt. Zu Beginn des Algorithmus wird angenommen, dass der größtmögliche Stanzkopf zuletzt benutzt wurde. Ein Zustand ist gültig, wenn die Restfläche, entsprechend der gewählten Konnektivitätsbedingung, zusammenhängend ist.

Definition Zustandsübergang

> Ein Zustandsübergang ist ein Stanzschritt, der durch einen Stanzkopf und dessen Position auf der Grundfläche bestimmt wird. Ein solcher Zustandsübergang ist gültig, wenn er einen gültigen Zustand in einen andern gültigen Zustand überführt und wenn mindestens 50% der vom Stanzkopf überdeckten Fläche noch Teil der Restfläche ist.

Ausgehend von einem gültigen Zustand sollten nun eigentlich alle möglichen Nachfolgezustände, die durch einen beliebigen gültigen Stanzschritt erreichbar sind, miteinander verglichen werden. Es ist abzusehen, dass die Zahl dieser Nachfolgezustände, abhängig von der Größe der Grundfläche und der Anzahl der möglichen Stanzschritte, sehr groß werden kann. Damit die Laufzeit des Algorithmus im annehmbaren Rahmen bleibt, muss deshalb für jeden Zustand eine Vorauswahl unter den gültigen Zustandsübergängen getroffen werden.

Der nächste Schritt besteht nun in der Bewertung der möglichen Folgezustände. Hierbei müssen also alle Restflächen, die diesen Zuständen entsprechen, untersucht werden.
Betrachtet man jedoch die Stanzschritte, so fällt auf, dass diese nur lokale Änderungen an der

Restfläche vornehmen. Daher scheint es auch sinnvoll, nur die lokalen Veränderungen zu bewerten, anstatt jedes mal aufs neue die komplette Restfläche zu betrachtet. Aus diesem Grund wird hier vom eigentlichen Vorgehen eines Greedy-Algorithmus abgewichen, indem NICHT die Zustände, sondern die Zustandsübergänge bewertet werden. Dies birgt zwar das Problem, dass Änderungen an verschiedenen Ecken der Restfläche schwierig miteinander zu vergleichen sind, der benötigte Zeitaufwand für die Bewertung lässt sich auf diese Weise jedoch stark reduzieren, da nur noch diejenigen Punkte beachtet werden müssen, die unter dem Stanzkopf oder in dessen direkten Nachbarschaft liegen.
Da nur eine kleine Region, nämlich die Stanzposition selbst, betrachtet werden muss, müssen für die Bewertung auch keine Änderungen an der Restfläche vorgenommen werden. Es genügt, die aktuelle Beschaffenheit der Fläche an einer möglichen Position zu betrachten.

3.2 Ablauf des Überdeckungsalgorithmus

Mit Hilfe der nun getätigten Vorüberlegungen kann der Ablauf des Algorithmus angegeben werden. Es wird sich hierbei um ein rundenbasiertes Vorgehen handeln. Jede Runde stellt hierbei einen einzelnen Stanzschritt dar und kann in drei Abschnitte unterteilt werden :

1) Suchen des Stanzschritts, der den größten Erfolg verspricht.

2) Durchführung des Stanzschritts.

3) Überprüfung der Restfläche auf mögliche Problemstellen und Beseitigung dieser Stellen.

Der dritte Punkt ist eine Notwendigkeit, die sich aus der Änderung der Bewertungsfunktion, hin zur Bewertung der Zustandsübergänge ergibt. Bei der Beschreibung der Bewertungsfunktion wird hierauf näher eingegangen.
Der Algorithmus wird nun diese drei Schritte solange durchlaufen, bis die komplette Restfläche entfernt wurde.

3.2.1 Finden des erfolgversprechendsten Stanzschrittes

Die Vorauswahl der möglichen Stanzschritte wird von zwei Faktoren bestimmt. Wie bereits erwähnt, bieten sich die Gratlinien als mögliche Stanzpositionen an, da sie eine Art „Mitte“ der Fläche darstellen. Deshalb wird sich der Algorithmus auf die Punkte der medialen Achse als mögliche Stanzpositionen beschränken.
Wenn die mediale Achse die Mitte der Fläche darstellt, so stellen die Endpunkte dieser Linien ebenfalls eine besondere Klasse von Punkten dar. Durch die diversen Forderungen an die Probleminstanz sind diese Punkte nämlich genau diejenigen Eckpunkte der Restfläche, an denen die Randsegmente einen Innenwinkel der Fläche von weniger als 180° definieren. Ein sehr nahe liegender Gedanke ist nun, die Restfläche von den Ecken ausgehend Stück für Stück zu verkleinern. Aus diesem Grund wird eine weitere Forderung aufgestellt, dass nämlich immer ein solcher Eckpunkt der Restfläche, also der Endpunkt einer Gratlinie, unter dem Stanzkopf zu liegen hat.

Für jeden möglichen Stanzschritt, der nicht von der Vorauswahl eliminiert wurde, müssen nun folgende Dinge überprüft werden:

- Handelt es sich hierbei um einen gültigen Stanzschritt (Zustandsübergang)
- Welche Bewertung erhält dieser Stanzschritt

Mit Blick auf die Laufzeit wird hierbei der Gültigkeitstest für einen Stanzschritt unterteilt. Zunächst wird geprüft, ob genügen Material unter dem Stanzkopf liegt. Bevor jedoch getestet wird, ob es sich beim entstehenden Nachfolgezustand um einen gültigen Zustand handelt, wird die Bewertung der Stanzposition berechnet. Nur wenn die Bewertung besser ist als die von allen anderen bis dato getesteten gültigen Stanzschritte, wird der Nachfolgezustand näher betrachtet.
Der Test, ob genügend Material unter einem Stanzkopf liegt, ist trivial und wird deshalb nicht näher ausgeführt. Für die Bestimmung der Gültigkeit eines Zustandes wird der Konnektivitätstest benötigt, der bereits in Kapitel 2.4 näher betrachtet wurde. Es fehlt somit nur noch eine Bewertungsfunktion für die Vervollständigung des Algorithmus, welche im Folgenden Schritt für Schritt aufgebaut wird.

3.2.1.1 Bewertungsfunktion

Angefangen bei der einfachsten Idee, werden mögliche Probleme nacheinander betrachtet und es wird versucht, die Funktion so abzuändern, dass diese umgangen werden können.
Der Ausgangspunkt der Funktion ergibt sich durch Betrachten der Problemstellung. Je mehr Fläche pro Arbeitsschritt überdeckt wird, desto weniger Arbeitsschritte werden benötigt, um die komplette Fläche zu entfernen. Daher bietet es sich an, zunächst die Summe aller Punkte, die bei einem Stanzschritt entfernt werden, als Bewertung zu wählen.
Für die Betrachtung des Problems ohne Nebenbedingungen scheint dieses Vorgehen sehr erfolgversprechend. Werden die Nebenbedingungen jedoch mit einbezogen, zeigen sich schnell einige offensichtliche Schwächen. Es wird nach Möglichkeit mit dem größten zur Verfügung stehenden Stanzkopf irgendwo auf der Restfläche gestanzt werden, sodass die Fläche unter dem Stanzkopf möglichst vollständig mit Material bedeckt ist. Aufgrund der Konnektivitätsbedingung müssen allerdings am Rand dieser Stanzposition immer kleine Stege zurückbleiben, damit auch die Eckpunkte der Restfläche mit dem Rest zusammenhängend bleiben. Aufgrund der zweiten Randbedingung werden diese Stege dann am Ende des Algorithmus mit wesentlich kleineren Stanzköpfen entfernt werden müssen, was sehr viele Stanzschritte erfordern wird.

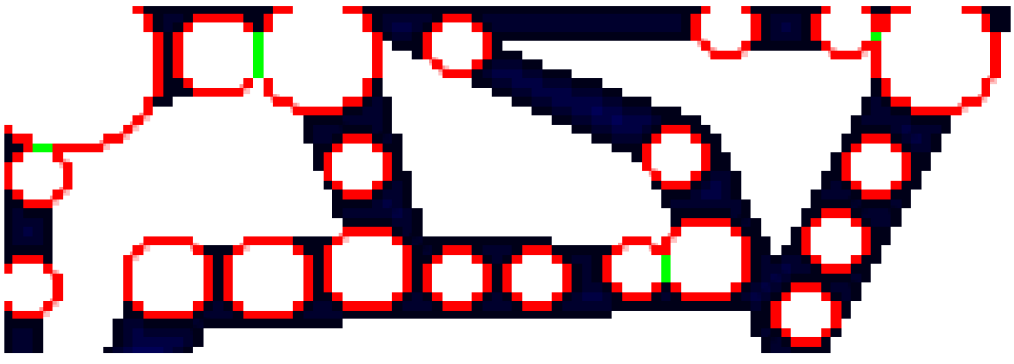

Abb. 6: Einfache Bewertungsfunktion (ohne Beachtung von Gratlinien-Endpunkten)

Durch die Vorbedingung, dass immer ein Gratlinien-Endpunkt unter dem Stanzkopf liegen muss, wird diesem Problem bereits entgegengewirkt. Als ein weiteres Mittel zur Vermeidung dieser Artefakte bietet es sich an, die Entfernung von Randpunkten zu belohnen. So soll eine Stanzposition, die am Rand der Restfläche liegt und diesen sogar etwas überschneidet, besser bewertet werden als eine Position in der Mitte, auch wenn in der Mitte mehr Fläche weggestanzt wird. Hierbei hilft die Information über die Distanz eines Punktes zum Rand der Restfläche, welche beim Erstellen der Gratlinien bereits berechnet wurde. Anstatt die entfernten Punkte einfach zu zählen, können diese je nach ihrer Lage gewichtet werden. Je näher ein Punkt am Rand liegt, desto „wertvoller“ soll er sein.
Die eben bereits genutzte Vorbedingung kann zusätzlich noch verstärkt werden, indem Endpunkte von Gratlinien zusätzlich in der Bewertungsfunktion belohnt werden. Es ist dann für eine Stanzposition von besonderem Vorteil, wenn gleich mehrere dieser Punkte weggestanzt und somit ganze Randsegmente auf einmal entfernt werden.

Eine weitere Degeneration des Feldes, die es wenn möglich zu verhindern gilt, sind lange enge „Schläuche“, wie in Abb.7 illustriert.

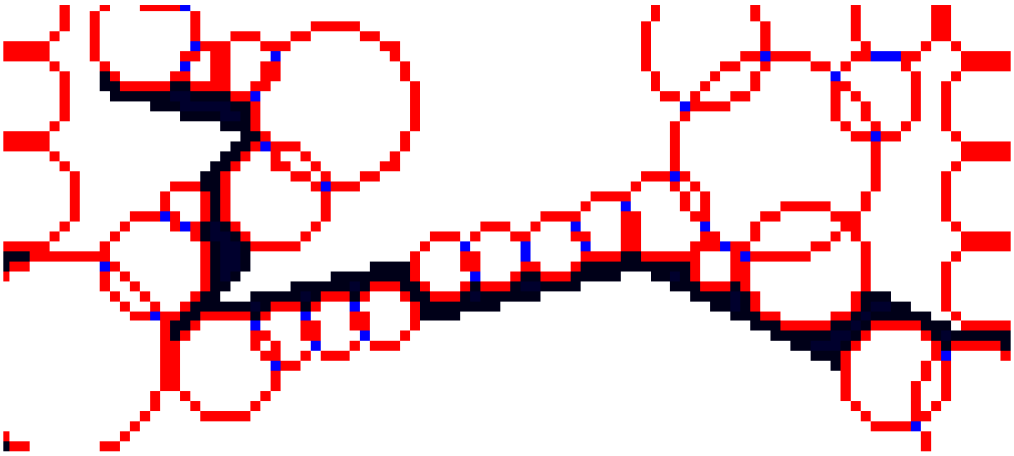

Abb. 7: Langer, enger Bereich der Restfläche, mit Ausgang rechts

Um solche zu erkennen, werden Gratpunkte mit sehr geringer Distanz zum Rand der Restfläche besonders gekennzeichnet und auch hier belohnt der Algorithmus das Entfernen solcher Punkte. Leider genügt dies nicht, um all diese Fälle zu vermeiden. Deshalb gilt es zusätzlich herauszufinden, wie ein solcher „Schlauch“ entstehen kann. Zwei unterschiedliche Fälle können hierbei unterschieden werden.

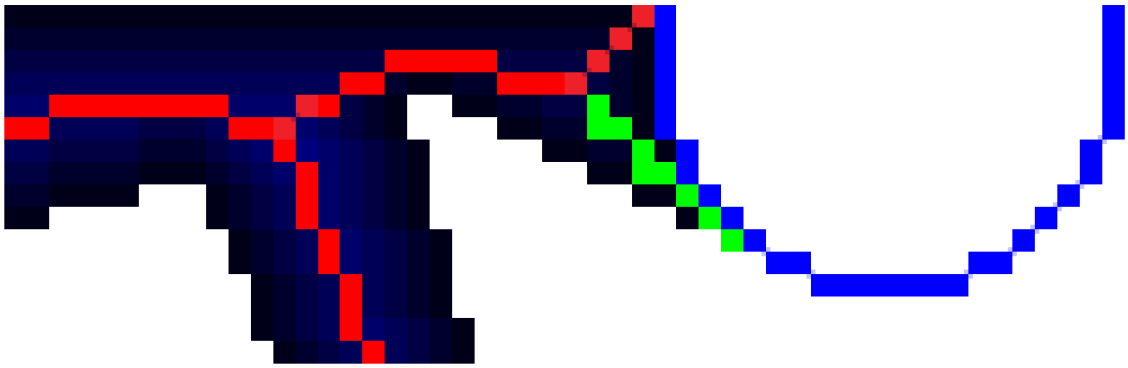

Abb. 8: Spitze Zunge der Restfläche

1. Fall:

Nach einem Stanzschritt bleibt eine sehr spitze, längliche Zunge stehen (Abb.8). Es ist klar, dass ein Stanzschritt, der diese Zunge entfernt, bei der bisherigen Bewertungsmethode sehr schlecht abschneiden wird. Zwar sind die Punkte alle nahe am Rand, bekommen demnach eine hohe Bewertung und die Gratpunkte haben ebenfalls eine geringe Distanz, dies kann aber kaum ausgleichen, dass hier nur mit einem sehr kleinen Stanzkopf gestanzt werden kann und dementsprechend nur wenige Punkte entfernt werden können. Es bleibt die Möglichkeit, an den bisherigen Schrauben zu drehen, um das Entfernen solcher Zungen für den Algorithmus attraktiv zu machen. Ab einem bestimmten Punkt wird sich dies allerdings auf das Gesamtergebnis sehr negativ auswirken, da die Belohnung für das Entfernen von möglichst vielen Punkten im Vergleich zu den Belohnungen für die Ausnahmeregelungen untergehen wird. Der Algorithmus würde in diesem Fall nur mit kleinen Stanzen am Rand der Restfläche arbeiten und nicht versuchen, möglichst große Flächen auf einmal zu entfernen. Deshalb wird angestrebt, eine solche Situation frühzeitig zu erkennen und sie unabhängig von der Bewertungsfunktion zu entfernen. Sie muss deshalb hier nicht weiter berücksichtigt werden.

2. Fall:

Die zweite Möglichkeit, wie ein solcher „Schlauch“ entstehen kann, ist nicht so einfach zu erkennen. Er tritt ein, wenn eine breite Stelle der Restfläche durch einen Stanzschritt verengt wird.

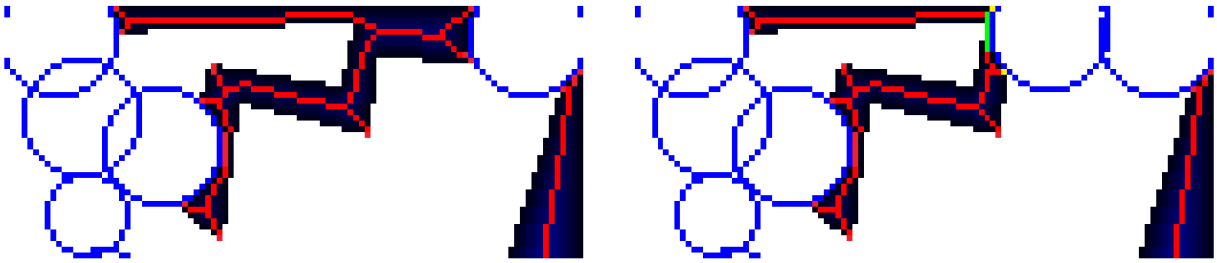

Abb. 9: Entstehung einer Engstelle (grün), neue Gratlinien-Endpunkte (gelb)

Es ist klar, dass eine solcher Stanzschritt grundsätzlich nicht schlecht bewertet werden wird. Es werden viele Punkte entfernt und dabei auch noch solche, die nahe am Rand liegen, wenn auch nicht ganz am Rand. Auf beiden Seiten der Engstelle finden sich nun größere Teile der Restfläche und natürlich erzeugt der Stanzschritt auf jeder Seite der Engstelle einen neuen Gratlinien-Endpunkt. In ungünstigen Fällen kann es nun geschehen, dass diese Engstelle verlängert wird, indem an einem der neu erzeugten Endpunkte weitergearbeitet wird.

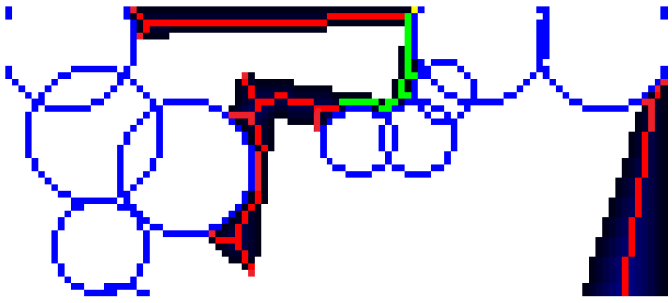

Abb. 10: Verlängerung der Engstelle (grün)

Um einen solchen Fall frühzeitig zu erkennen und zu verhindern, wird eine neue Bedingung eingeführt: Für jeden Stanzschritt müssen die Anzahl der Punkte mit der Distanz i größer sein als die Anzahl der Punkte mit der Distanz i+1. Wird bei einer Verletzung dieser Bedingung, für kleine i, die Bewertung des Stanzschrittes stark herunter gesetzt, wird die Wahrscheinlichkeit der Entstehung einer solchen Engstelle reduziert. Des weiteren sorgt diese Bedingung dafür, dass sehr spitze Ecken, die entstehen könnten, falls eine Stanzpostition zu weit innerhalb der Restfläche liegt, verhindert werden.
Eine alternative oder zusätzliche Bedingung, die eine solche Entwicklung verhindern könnte, wäre folgende: Klar ist, dass bei einem Stanzschritt, der die Restfläche verengen wird, relativ viel „neuer“ Rand entsteht. Der Algorithmus sollte deshalb wenn möglich den Rand verkürzen, anstatt ihn zu verlängern. Somit kann ein Stanzschritt zusätzlich nach der Größe der neu entstehenden Randsegmente bewertet werden.

Zusammenfassend sollten folgende Punkte in die Bewertung eines Stanzschrittes einfließen:

- Mit jedem Stanzschritt sollte so viel Material wie möglich von der Restfläche entfernt werden.
- Der Stanzkopf sollte möglichst selten gewechselt werden.
- Die Restfläche sollte so breit wie möglich gehalten werden.
- Sehr spitz zulaufende Ecken sollten wenn möglich vermieden werden.
- Wird trotzdem eine spitz zulaufende Ecke erkannt, muss sie sofort beseitigt werden.

Die ersten beiden Punkte sind durch die Aufgabenstellung gegeben, die anderen lassen sich, wie oben beschrieben, aus der ersten Forderung ableiten.
Klar ist, dass sich die meisten dieser Regeln gegenseitig beeinflussen. Da jede Probleminstanz anders ist, wird sich keine optimale Gewichtung der Regeln finden lassen, für die immer ein sehr gutes Ergebnis berechnet wird. Für jede Ausgangsposition wird eine andere Gewichtung die optimale sein. Eine Lösung hierfür wäre, das Programm mehrfach mit unterschiedlichen Gewichtungen zu starten und das beste Ergebnis zu wählen. In Kapitel 7 wird ein möglicher Lösungsvorschlag kurz angesprochen.

3.2.2 Durchführung des Stanzschrittes

Ist der bestmögliche nächste Stanzschritt berechnet, wird dieser durchgeführt. Die Punkte der Restfläche, die unterhalb es Stanzkopfes liegen, werden entfernt und die in Kapitel 2.3 vorgestellte Funktion zur Aktualisierung der Distanzwerte und der medialen Achse wird durchlaufen.

3.2.3 Suche nach potenziellen Problemstellen

Wie in Kapitel 3.2.1 bereits angedeutet soll hier erkannt werden, wenn die Gefahr besteht, dass eine spitz zulaufende Ecke der Restfläche im normalen Bewertungssystem keine Beachtung mehr findet. Wird ein solcher Fall erkannt, sollte diese Ecke umgehend beseitigt werden. Dies muss geschehen, bevor die nächste Stanzposition nach den regulären Regeln des Algorithmus ermittelt wird, damit die Situation bereinigt ist, bevor sie sich im schlechtesten Fall noch verschlimmern kann.
Die hier umgesetzte Idee basiert auf den Endpunkten der Gratlinien. Logischerweise muss am äußersten Ende einer spitz zulaufenden Ecke ein solcher Endpunkt liegen. Kann ein solcher Punkt nicht mehr durch einen Mittelgroßen Stanzkopf entfernt werden, gilt er und seine Umgebung als gefährdet. Für den nun folgenden Stanzschritt gilt, dass dieser Endpunkt entfernt werden muss. Es ist somit nicht nötig, die komplette Restfläche nach dem bestmöglichen Stanzschritt zu durchsuchen, es genügt die nähere Umgebung des Punktes zu betrachten. Die Bewertungsfunktion hingegen bleibt die selbe.

4 Implementierung

Nachdem die grundlegenden Ideen erklärt wurden, wird im folgenden etwas mehr ins Detail gegangen und es werden einige zentrale Aspekte der Implementierung vorgestellt.
Als Programmiersprache wurde C++ verwendet und als Entwicklungsumgebung kam der QtCreator zum Einsatz, welcher sich aufgrund seines GUI-Designers und der Qt-Klassenbibliotheken für das Arbeiten mit geometrischen Strukturen angeboten hat.
Im folgenden wird die Darstellung der Probleminstanz kurz näher besprochen, bevor einige Funktionen und Algorithmen im Detail betrachtet werden.

4.1 Probleminstanz

Zur Darstellung der Arbeitsfläche wurde ein zweidimensionales Feld verwendet. Die daraus resultierende Diskretisierung der Fläche auf ein Rastergitter birgt einige Probleme, auf die bereits in Kapitel 2 eingegangen wurde, ist aber einfach zu handhaben und ein intuitives Modell. Auf eine alternative Lösung zur Diskretisierung wird in Kapitel 7 näher eingegangen.

Für jeden Punkt der Fläche werden nun folgende Informationen gespeichert:

```
bool exist;
bool akt;
int dis;

bool isGrat;
bool lowGratPoint;
bool gratEndPoint;
```

Die Variable „exist“ gibt darüber Auskunft, ob der Punkt noch zur Restfläche gehört oder ob er bereits entfernt wurde. „akt“ wird benötigt um zu verhindern, dass ein Punkt beim Durchsuchen der Fläche mehrfach besucht wird. „dis“ gibt den Distanzwert eines Punktes an.
Die anderen Werte werden für die Darstellung der medialen Achse verwendet. Ist ein Punkt Teil der medialen Achse („isGrat“ = true), wird zusätzlich noch vermerkt, ob es sich bei dem Punkt um einen Gratpunkt mit sehr niedrigem Distanzwert („lowGratPoint“) oder gar um einen Endpunkt einer Gratlinie („gratEndPoint“) handelt.

Bei der Initialisierung des Feldes wird ein zweidimensionaler Array erstellt, in dem alle Punkte zunächst noch existieren. Als Distanzwert der jeweiligen Punkte wird die minimale Distanz zum Rand des Feldes gewählt. Hierbei ist klar, dass die Distanzwerte zweier Nachbarpunkte, basierend auf der Vierer-Nachbarschaft, sich höchstens um eins unterscheiden.
Die Gratlinien werden hier zunächst im 45° Winkel von den Eckpunkten weglaufen. Zwischen den Stellen, an denen sich jeweils zwei dieser Gratlinien treffen, wird zusätzlich noch ein senkrechtes oder ein waagerechtes Gratlinien-Segment liegen, sofern das Feld keine quadratische Form besitzt. In einem Quadrat werden sich alle vier von den Eckpunkten ausgehenden Linien in einem Punkt in der Mitte treffen.
Nachdem dann alle Werkstücke nacheinander entfernt wurden, existieren auf dem Feld nur noch diejenigen Punkte, die Teil der initialen Restfläche sind.

Da alle verwendeten Stanzköpfe rund sein werden, genügen für ihre Darstellung im Programm der Mittelpunkt, welcher auch gleichzeitig die Position eines Stanzkopfes angibt, sowie der Radius. Mit diesen beiden Werten kann berechnet werden, ob ein diskretisierter Punkt P vom Stanzkopf überdeckt wird oder nicht. Dabei wird die euklidische Distanz zwischen P und dem Mittelpunkt mit dem Radius verglichen. Ist die Distanz kleiner als der Radius, liegt der Punkt innerhalb des Kreises, sonst nicht. Zu beachten ist hierbei die Rundung. Wird ohne Rundung verglichen, besitzen die Kreise (vor allem kleine Kreise) eine etwas merkwürdige Form. Aus diesem Grund wird die euklidische Distanz gerundet und erst dann mit dem Radius verglichen, welcher per Definition bereits ganzzahlig ist.

Abb. 11: Kreise mit Radien 2, 6 und 15; links mit Rundung, rechts ohne

4.2 Aktualisierung der medialen Achse

Hier wird nun ein detaillierter Blick auf die Aktualisierung der Distanzwerte geworfen. Diese Funktion liefert gleichzeitig die neu entstehenden Gratlinien, die zwischen den verschiedenen herausgestanzten Figuren verlaufen.
Beim Entfernen einer geometrischen Figur von der Grundfläche werden die Randpunkte derselben in einer Liste gespeichert, welche die Basis für diese Funktion darstellt. In einem FiFo-Verfahren (First in - First out) wir diese Liste nun abgearbeitet. Dabei wird bei jedem Durchlauf zunächst das erste Element aus der Liste entfernt und sein Distanzwert ermittelt (disA). Daraufhin werden alle seine Nachbarpunkte bestimmt. Wichtig ist hierbei, dass nur diejenigen Nachbarpunkte bearbeitet werden, die im Laufe des aktuellen Stanzschrittes noch nicht betrachtet wurden. Auf diese Weise wird sichergestellt, dass nur vom Stanzkopf ausgehend nach außen gearbeitet wird. Je nach Distanzwert eines noch nicht betrachteten Nachbarpunks (disN) werden nun 3 Fälle Unterschieden:

1. Fall : disA < disN

Tritt dieser Fall ein, bedeutet dies, dass die Distanz des Nachbarpunktes zum neuen Rand kleiner oder gleich seiner Distanz zum alten Rand ist. Der Distanzwert des Punktes wird in diesem Fall aktualisiert und auf disA + 1 gesetzt. Da der Distanzwert des Punktes vom neuen Rand bestimmt wird, muss mit ihm weitergearbeitet werden. Deshalb wird er hinten an die Liste angehängt.

2. Fall : disA = disN

Da bereits sichergestellt ist, dass der Nachbarpunkt noch nicht bearbeitet wurde, ist klar, dass sein Distanzwert von einem alten Randpunkt aus bestimmt worden ist. Die Distanz des aktuellen Punktes hingegen wurde vom neuen Rand aus ermittelt. Sind beide Distanzwerte gleich bedeutet dies, dass die eigentliche Gratlinie zwischen den beiden Punkten verlaufen muss. Für einen solchen Fall wurde festgelegt, dass einer der beiden Punkte zu einem Punkt der diskretisierten medialen Achse bestimmt werden muss. Es wird hier immer der aktuelle Punkt gewählt.
Zusätzlich muss eine kleine Besonderheit beachtet werden. Da alle Randpunkte der Figur in der Liste gespeichert wurden, kann es vorkommen, dass ein Nachbarpunkt N eines Randpunktes A bereits in einem früheren Schritt entfernt wurde. Da der Punkt N noch nicht als bearbeitet markiert ist und, genau wie der Punkt A, einen Distanzwert von Null besitzt, wird der Punkt A vom Algorithmus als Gratpunkt erkannt, was natürlich nicht der Fall sein sollte. Aus diesem Grund wird eine Ausnahme eingeführt, sodass der Fall disA = disP = 0 einfach übergangen wird.

3. Fall : disA > disN

Für diesen Fall muss gezeigt werden, dass die Gleichung

$$disA = disN + 1$$

immer wahr ist:

Das Feld wird so initialisiert, dass sich die Distanzwerte eines auf Basis der Vierer-Nachbarschaft benachbarten Punktpaares um höchstens eins unterscheidet. Diese Bedingung soll nun nach jedem Stanzschritt erfüllt sein. Im Verlauf des Algorithmus werden die Distanzwerte nur durch das Entfernen eines Punktes (Distanzwert wird auf Null gesetzt) und durch die aktuell besprochene Funktion geändert. Indem alle Randpunkte einer Figur, nach dem Entfernen derselben, an diese Funktion übergeben werden, obliegt es allein ihr dafür zu garantieren, dass die obige Bedingung auch nach dem Stanzschritt wieder gültig ist.
Zunächst kann festgestellt werden, dass es eine Verletzung der Bedingung nur entlang derjenigen Punkte geben kann, die in der hier verwendeten Liste gespeichert sind. Gleichzeitig gilt, dass bei einer solchen Verletzung der Punkt aus der Liste stets den kleineren der beiden Distanzwerte besitzt. Wird eine Verletzung vom Algorithmus erkannt, wird sie aufgehoben, indem der Distanzwert des Nachbarpunktes verringert wird (siehe 1.Fall). Somit wird sich der Distanzwerte eines Punktes niemals erhöhen. Für jeden Punkt, der neu in die Liste aufgenommen wird, gilt demnach, dass sein Distanzwert vor der Aufnahme größer oder gleich seinem neuen Distanzwert ist.

Nehmen wir nun an wir befinden uns im 3.Fall und es gilt:

$$disA > disN + 1.$$

Wir wissen, dass der Distanzwert des aktuellen Punktes vor seiner Aufnahme in die Liste größer oder gleich dem jetzigen Distanzwert war.

$$disA \leq disA\_alt$$

Weiter wissen wir, dass sich die alte Distanz um höchstens eins von der Distanz seines Nachbarpunktes unterschieden hat.

$$disA\_alt \leq disN + 1$$

Aus diesen beiden Gleichungen ergibt sich:

$$disA \leq disN + 1,$$

was ein Widerspruch zur getätigten Annahme ist. Somit ist gezeigt, dass die Gleichung

$$disA = disN + 1$$

für den dritten Fall immer zutreffend ist und die Distanz des aktuelle Punktes zum alten Rand genau der Distanz zum neuen Rand entspricht. Der aktuelle Punkt ist somit ein eindeutiger Punkt der medialen Achse.

4.2.1 Sonderfall nicht konvexe Polygone

Nun müssen noch diejenigen Gratlinien berechnet werden, die von verschiedenen Segmenten eines einzelnen Polygons definiert werden. Wie bereits festgestellt wurde, werden diese Gratlinien immer in einem Eckpunkt des Polygons enden, dessen Innenwinkel größer als 180° ist. Es bietet sich deshalb an, alle diese Eckpunkte des Polygons zu suchen und von diesen ausgehend die Gratlinien zu berechnen.
Die Eckpunkte lassen sich einfach bestimmen, indem entlang der Segmente des Polygons gelaufen und für jeden Punkt berechnet wird, ob der Rand des Polygons hier einen Links- oder einen Rechtsknick beschreibt. Wird im Uhrzeigersinn gelaufen, werden diejenigen Eckpunkte gespeichert, die einen Linksknick beschreiben, andernfalls die Eckpunkte mit einem Rechtsknick.
Für jeden der so ermittelten Punkte muss als nächstes die Richtung bestimmt werden, in die die hier endende Gratlinie verläuft. Hierzu wird der Außenwinkel des Polygons am Eckpunkt P betrachtet, welcher natürlich kleiner als 180° sein muss. Dieser Winkel wird von den beiden Segmenten des Polygons definiert, die sich in P treffen. Die zu diesem Winkel gehörende Winkelhalbierende bestimmt genau die Mitte zwischen den Segmenten und definiert somit auch die Richtung der Gratlinie. Der erste Punkt, der auf dieser Winkelhalbierenden liegt und noch Teil der Restfläche ist, ist demnach der Endpunkt der Gratlinie. Zusätzlich zum Endpunkt wird noch dessen Vorgänger in Richtung des Eckpunktes des Polygons gespeichert, womit die Richtung der Gratlinie implizit festgehalten wird. Ist der Winkel relativ groß, wird dieser zweite Punkt der Eckpunkt selbst sein. Bei kleineren Winkeln ist dies jedoch nicht zwangsläufig der Fall.

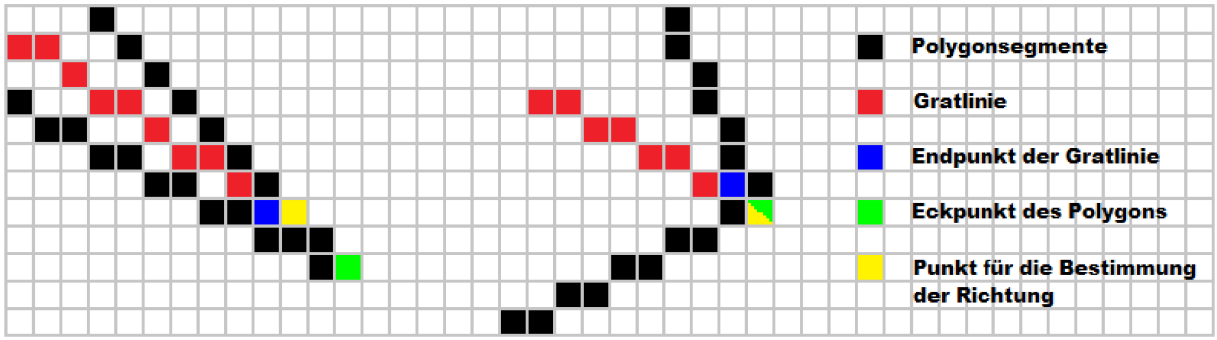

Abb. 12: Bestimmung des ersten Punktpaares an den Eckpunkten eines Polygons

Diese beiden Punkte bilden nun ein Paar, welches eine unvollendete Gratlinie definiert. Der Algorithmus berechnet zunächst alle diese Paare, die von einem der Eckpunkte des Polygons definiert werden und speichert diese in einer Liste, bevor damit begonnen wird die Gratlinien zu zeichnen.

Sind alle Ecken des Polygons bearbeitet worden, wird die Liste der Punktpaare Schritt für Schritt abgearbeitet. Mit Hilfe eines solchen Paares lässt sich der weitere Verlauf der Gratlinie abschätzen. Da eine solche Gratlinie grundsätzlich keine plötzlichen Richtungsänderungen aufweist, kann davon ausgegangen werden, dass einer der drei, dem richtungsbestimmenden Punkt gegenüberliegenden Punkte, die Fortsetzung der Gratlinie sein wird.

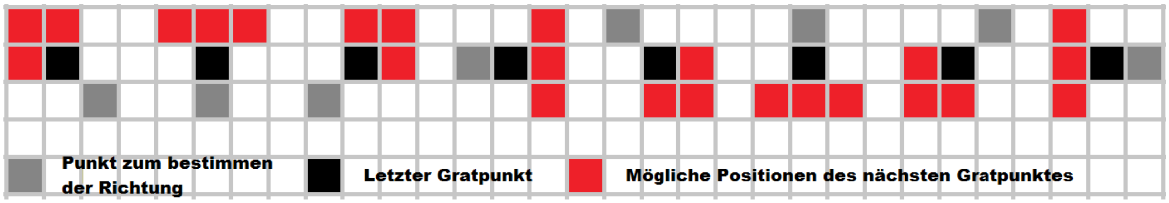

Abb. 13: Bestimmung des nächsten Gratpunktes

Es gilt demnach herauszufinden, welcher der Punkte am wahrscheinlichsten ein Gratpunkt ist. Hierzu werden die Distanzwerte dieser Punkte ermittelt: disL, disM und disR. Es werden nun folgende Fälle unterschieden:

1. Fall : Einer der Distanzwerte ist echt größer als die beiden anderen

    In diesem Fall wird der Punkt mit dem höchsten Distanzwert als neuer Gratpunkt gewählt.

2. Fall : Alle Distanzwerte sind gleich groß

    Hier wird der mittlere Punkt als nächster Gratpunkt definiert werden.

3. Fall : (disM < disL) und (disM < disR)

    Hier wird die Gratlinie gespalten und es werden zwei neue Punktpaare erzeugt.

4. Fall : disR = disM > disL

Hier wird der rechte Punkt als neuer Gratpunkt bestimmt. Da disL echt kleiner als disM ist, kann die Gratlinie nicht direkt durch den mittleren Punkt laufen. Sie verläuft also entweder zwischen dem mittleren und dem rechten Punkt oder sie verläuft direkt über den rechten Punkt. In beiden Fällen ist die Benennung des rechten Punktes korrekt.

5. Fall : disL = disM > disR

Hier wird der nächste Gratpunkt der linke Punkt sein, aufgrund der selben Überlegung wie im vierten Fall.

Nun kann mit dem neuen Punktpaar, bestehend aus letztem Gratpunkt und neu berechnetem Gratpunkt, das beschriebene Verfahren rekursiv erneut aufgerufen werden. Dies wird so lange wiederholt, bis einer der in Frage kommenden neuen Punkte bereits ein Gratpunkt ist. In diesem Fall ist die Gratlinie fertig gezeichnet.

Wichtig ist allerdings zu erwähnen, dass eine Gratlinie nur dann fortgeführt werden darf, wenn der Distanzwert des neu berechneten Punktes größer oder gleich dem des aktuellen Punktes ist. Wird dies nicht beachtet können völlig falsche Gratlinien entstehen, wie in Abb.14 gezeigt wird.

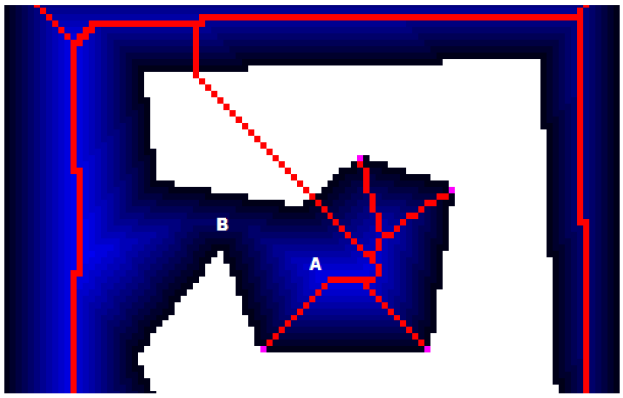

Abb. 14: Falsch berechnete Gratlinie wegen Nadelöhr

Aufgrund dieser Einschränkung ist klar, dass die mediale Achse in Abb.14 nicht vollendet werden kann, da der Ausgang bei B einem Nadelöhr gleicht und die Gratlinie, die dort hindurchführt, kleinere Distanzwerte aufweist als die Gratlinie bei A. Man wird folglich mit dem bisherigen Vorgehen nie von A nach B gelangen. Für dieses Problem gilt es also noch eine Lösung zu finden. Zuvor wird jedoch ein weiterer Spezialfall untersucht.

In Abb.15 kann man erkennen, dass an der Stelle A die Gratlinie, die von unten nach oben verläuft, senkrecht auf die horizontale Gratlinie trifft. Wird die senkrechte Gratlinie nun zuerst berechnet, wird der Algorithmus an dieser Stelle abbrechen, da alle Nachbarpunkte, die für die Fortführung der Gratlinie in Betracht gezogen werden, einen zu kleinen Distanzwert besitzen. Der Algorithmus wird deshalb mit den Gratlinien der beiden anderen Eckpunkte des Polygons fortfahren. Hat er sich nun bis zum Punkt A vorgearbeitet, erkennt er dort einen bereits existierenden Gratpunkt und bricht seine Suche ab, obwohl die Gratlinie eigentlich nach links fortgeführt werden sollte.

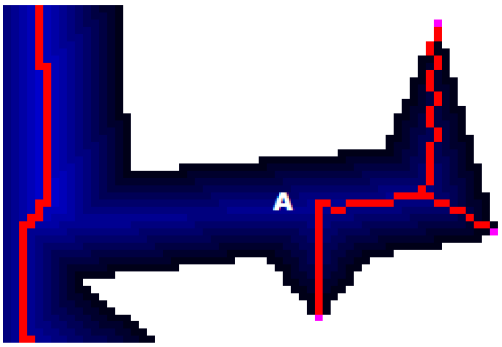

Abb. 15: Senkrecht zueinander stehende Gratlinien

Aus diesem Grund wird eine zusätzliche Abfrage für den Fall einer Gratlinie eingeführt, die senkrecht auf einer anderen Gratlinie steht.
Findet der Algorithmus in der vorgegebenen Richtung keinen neuen Gratpunkt wird überprüft, ob an dieser Stelle eine Gratlinie im Winkel von 90° kreuzt. Hierzu wird für die beiden Nachbarpunkte links und rechts vom aktuellen Punkt getestet ob sie als Gratpunkt in der entsprechenden Richtung in Frage kommen. Ist dies der Fall, werden zwei neue Punktpaare erstellt und in die Liste der nicht vollendeten Gratlinien geschrieben. Sie werden somit erst am Ende bearbeitet, nachdem alle Eckpunkte des Polygons bereits abgearbeitet sind. Auf diese Weise wird stets vom Rand in Richtung Mitte der Restfläche gearbeitet, womit Komplikationen vermieden werden können.

Kommen wir nun zur Überbrückung des Nadelöhrs. Wie bereits beschrieben, wird entlang der Verbindungslinien zwischen allen Eckpunkten des Polygons nach möglichen Gratpunkten gesucht. Bei einer sehr großen Steigung wird das Verbindungssegment hierbei in y-Richtung abgelaufen, um eine möglichst feine Abtastung zu garantieren. Für jeden Punkt auf diesen Verbindungslinien wird getestet, ob es sich um einen möglichen Gratpunkt handelt. Falls ja, wird er gespeichert.

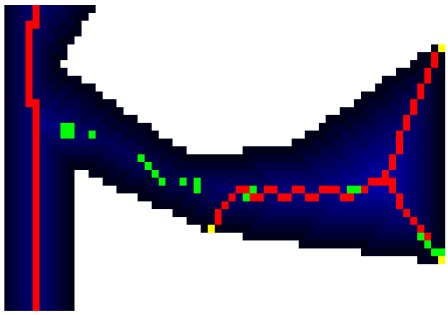

Abb. 16: Mögliche Punkte der noch nicht gezeichneten medialen Achse (grün)

Nun werden die eben gefundenen Punkte einer nach dem anderen heraus gepickt und untersucht. Ist der Punkt selbst oder einer seiner Nachbarn bereits ein Gratpunkt, wird er ignoriert und ein weiterer Punkt wird gewählt. Ist dies nicht der Fall, wird die Richtung der Gratlinie bestimmt, die durch diesen Punkt verläuft. Diese Gratlinie wird dann vom Punkt ausgehend in beide Richtungen gezeichnet, bis eine bereits existierende Linie erreicht wird. Auf diese Weise werden alle Punkte abgearbeitet. Die meisten Punkte der Liste werden hierbei jedoch ignoriert werden, da in den meisten Fällen nur ein Punkt benötigt wird um die Linie zu zeichnen.

4.2.2 Erkennen eines Gratpunktes / Bestimmung der Richtung einer Gratlinie

Bisher wurde nicht darauf eingegangen, wie ein Gratpunkt als solcher erkannt werden kann und wie sich die Richtung der Gratlinie durch einen Punkt bestimmen lässt. Hierfür muss die Umgebung des Punktes abgetastet werden. Angenommen der Punkt P ist Teil einer Gratlinie. Wird nun eine Gerade durch P gezogen, die im Winkel von 90° zur Gratlinie liegt, so müssen alle Punkte auf dieser Geraden Distanzwerte haben, die kleiner oder gleich dem Distanzwert von P sind. Um die Bedingung möglichst lokal festzumachen betrachten wir nur fünf auf dieser Geraden nebeneinander liegende Punkte, wobei P der mittlere der fünf sein soll. Um nun zu verhindern, dass auch falsche Punkte als mögliche Gratpunkte erkannt werden, genügt die Forderung nach „kleiner-gleich“ für die vier Punkte nicht mehr. Für die beiden Punkte im Abstand eins zu P bleibt die Forderung bestehen, für die Punkte im Abstand zwei wird jedoch das „kleiner-gleich“ in ein „echt kleiner“ umgewandelt (Abb.17). Es werden hierbei acht mögliche Richtungen unterschieden.

Abb. 17: Alle acht möglichen Richtungen für den Gratlinien-Test

Mit diesem Hilfsmittel können nun beide noch offenen Fragen gelöst werden. Gilt es herauszufinden, ob ein Punkt Teil einer Gratlinie ist, wird für alle acht Richtungen geprüft ob die oben genannte Bedingung zutrifft. Ist dies für mindestens eine Richtung der Fall, ist der Punkt ein Gratpunkt.

Indem wir uns die Richtung merken, haben wir gleichzeitig auch eine Antwort auf die Frage nach der Richtung der Gratlinie. Hierbei muss allerdings beachtet werden, dass in den allermeisten Fällen mehrere Richtungen ein positives Ergebnis zurückgeben. Um die Richtung mit Sicherheit bestimmen zu können, werden die beiden Nachbarpunkte, die in Richtung der vermeintlichen Gratlinie liegen bestimmt und es wird getestet, ob es sich bei ihnen ebenfalls um Gratpunkte handelt. Falls nicht werden die anderen möglichen Richtungen überprüft, bis die Richtige gefunden wurde.

4.2.3 Endpunkte von Gratlinien / niedrige Gratlinien

Für die Bewertungsfunktion und für die Einschränkung des Suchraums sind die Endpunkte der Gratlinien, sowie Stellen, an denen die Punkte der Gratlinien sehr niedrige Distanzwerte aufweisen, von zentraler Bedeutung. Wie diese erkennt und verwaltet werden, wurde bisher allerdings noch nicht angesprochen.
Niedrige Gratpunkte können während der Berechnung der Gratlinien sehr einfach ermittelt werden. Wird ein Gratpunkt als solcher erkannt, wird sein Distanzwert überprüft. Ist dieser klein genug, wird der Punkt als niedriger Gratpunkt markiert.
Etwas schwieriger gestaltet sich die Sache allerdings beim ermitteln der Endpunkte von Gratlinien. Klar ist, dass ein Endpunkt gleichzeitig ein niedriger Gratpunkt sein muss, da er am

Rand der Restfläche liegt und somit einen Distanzwert von eins besitzt. Da jedoch nicht alle Gratpunkte mit Distanzwert eins auch gleichzeitig Endpunkte von Gratlinien sind, kann dieses Kriterium nur zur Einschränkung der Kandidaten dienen und es muss ein weiteres Kriterium gefunden werden, um die Nicht-Endpunkte auszusortieren. Der entscheidende Unterschied besteht schlicht und einfach darin, dass der Endpunkt einer Gratlinie, im Gegensatz zu allen anderen Punkten, nur einen Liniennachbarn besitzt. Somit muss die Achter-Nachbarschaft jedes der möglichen Kandidaten untersucht werden. Wird mehr als ein Gratpunkt entdeckt handelt es sich nicht um einen Endpunkt der Gratlinie und er kann aussortiert werden.
Auch diese Methode deckt aber noch nicht alle möglichen Fälle ab, da bei der hier vorgestellten Berechnung der Gratlinien eine Situation wie in Abb.18 dargestellt eintreten kann. In diesem Spezialfall wird der Endpunkt nicht als solcher erkannt. Dies kann sogar dazu führen, dass der Algorithmus nicht korrekt terminiert, da diese Ecke der Restfläche nicht als solche erkannt wird und deshalb nie entfernt werden kann.

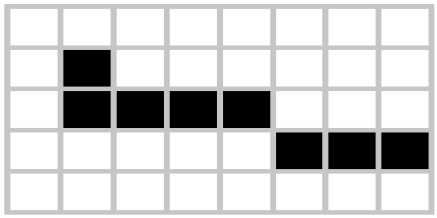

Abb. 18: Nicht erkannter Gratlinien-Endpunkt

Es gilt also eine weitere Möglichkeit zu finden, bei der auch dieser Spezialfall als ein Endpunkt erkannt wird. Zunächst kann festgestellt werden, dass eine solche Situation bei der Initialisierung des Feldes und bei der Betrachtung des Sonderfalls für nicht konvexe Polygone nicht entstehen kann. Er kann demnach nur dort eintreten, wo sich zwei oder mehrere unterschiedliche Stanzpositionen berühren. Betrachten wir nun einen Fall, in dem ein neuer Stanzschritt einen älteren überlappt.
In diesem Fall entstehen nur an denjenigen Stellen, an denen sich altes und neues Randsegment treffen neue Eckpunkte der Fläche und dementsprechend auch neue Endpunkte von Gratlinien. Diese Eckpunkte liegen offensichtlich in direkter Nachbarschaft zum Stanzkopf. Wie bereits bei den Konnektivitätstests kann auch hier der den Stanzkopf umschließende Ring betrachtet werden. Ebenfalls sind hier diejenigen Kreissegmente des Ringes interessant, die noch Teil der Restfläche sind, da sie eine neues Randsegment der Restfläche definieren. An beiden Enden eines solchen Segments wird die Restfläche einen neuen Eckpunkt erhalten und dementsprechend wird an diesen Stellen auch der Endpunkt einer Gratlinie liegen.
Zur Bestimmung dieser Punkte werden zunächst alle Punkte der neuen Randsegmente ermittelt. Daraufhin wird ein beliebiger Punkt ausgewählt und von ihm ausgehend entlang des Randsegments in beide Richtungen gelaufen, bis die Enden des Segments erreicht werden. Diese Enden werden als Gratlinien-Endpunkte markiert und es wird, sofern vorhanden, mit dem nächsten Segment fortgefahren.

Mit Hilfe der beiden vorgestellten Methoden können nun alle Endpunkte der Gratlinien erkannt werden. Die vier Eckpunkte des Feldes werden bereits bei dessen Initialisierung als solche definiert. Während nun bei der Entfernung der Bauteile die Endpunkte mit Hilfe des zuerst beschriebenen Prinzips ermittelt werden, kommt im Laufe des eigentlichen Überdeckungsalgorithmus die letztere Methode zur Anwendung.

4.3 Umsetzung der Konnektivitätstest

Wie bereits in Kapitel 2 beschrieben, wurden zwei alternative Funktionen für den Konnektivitätstest geschrieben. Da dieser für viele mögliche Stanzpositionen durchgeführt werden muss, war ein zentraler Aspekt, dass möglichst keine Veränderungen auf dem Feld vorgenommen werden, die im Nachhinein wieder rückgängig gemacht werden müssen. Deshalb wird zunächst eine Kopie der Umgebung der Stanzposition erstellt. Das Interesse gilt hierbei ausschließlich den NEN (noch existierenden Nachbarpunkten), welche die durch den Stanzschritt neu entstehenden Randsegmente darstellen. Diese Punkte werden im kopierten Feld markiert. Existiert kein solcher Punkt, kann die Konnektivität der Restfläche nicht verletzt und die Funktion beendet werden. Dieser Schritt wird für beide vorgestellte Algorithmen durchgeführt.

4.3.1 Algorithmus ohne mediale Achse

Lokale Suche:

In der erzeugten Kopie des Feldausschnitts gilt es nun zu testen, ob die markierten Punkte ein zusammenhängendes Kreissegment bilden oder nicht. Hierzu wird ein Stack erstellt, der im LiFo (Last in - First out) Verfahren abgearbeitet wird. Zunächst wird einer der markierten Punkte ausgewählt, auf den Stack geschrieben und seine Markierung entfernt. Nun wird der Stack abgearbeitet, indem immer der oberste Punkt entfernt wird und für alle Nachbarpunkte (Vierer-Nachbarschaft) überprüft wird, ob diese markiert sind oder nicht. Zu beachten ist, dass für den zuerst gewählten Punkt maximal zwei Nachbarpunkte markiert sein können, für alle weiteren jeweils nur einer. Diese gefundenen, markierten Punkte werden vom Feld entfernt und auf den Stack geschrieben. Ist der Stack abgearbeitet bedeutet dies, dass ein Kreissegment komplett von der Kopie entfernt wurde. Existieren nun keine markierte Positionen mehr, bedeutet dies, dass der komplette neu entstehende Rand Teil des selben Randsegments ist und die lokale Suche die Konnektivität garantieren kann. Falls noch markierte Punkte existieren, muss jedoch eine globale Suche durchgeführt werden.
Das Ziel der globalen Suche wird lauten, eine Verbindung der einzelnen Kreissegmente zum Rand des Feldes zu finden. Es genügt hierbei, für jedes der Kreissegmente einen einzelnen Punkt als Repräsentant zu wählen. Hierfür kann die lokale Suche erweitert werden. Das oben beschriebene Verfahren wird einfach so lange fortgesetzt, bis alle Kreissegmente bearbeitet wurden, wobei der jeweils erste gewählte Punkt als Repräsentant für das Kreissegment gespeichert wird.

Globale Suche:

Hierbei handelt es sich um eine einfache, stackbasierte Tiefensuche. Zunächst wird einer der Repräsentanten gewählt und auf den Stack geschrieben. Auch hier wird der Stack abgearbeitet, indem der oberste Punkt vom Stack herunter genommen wird und alle seine Nachbarpunkte untersucht werden. Ist ein solcher Nachbarpunkt noch Teil der Restfläche, wird er auf den Stack geschrieben.
Um keine Kopie des kompletten Feldes erstellen zu müssen, wird diese Suche auf dem originalen Feld durchgeführt. Hierbei wird es nötig sein, die Punkte des Feldes, die bereits besucht wurden, als solche zu markieren, da der Algorithmus sonst in einer Endlosschleife landen würde, indem er zwischen Punkten hin und her springt. Diese Markierung wird am Ende des

Tests für das komplette Feld wieder aufgehoben.
Für jeden neu vom Stack genommenen Punkt wird ermittelt, ob dieser ein Randpunkt des Feldes ist, oder ob es gar einer der Repräsentanten eines anderen Kreissegments ist. Wird ein solcher Repräsentant gefunden, wird dieser aus der Liste der Repräsentanten entfernt, da keine extra Suche für ihn durchgeführt werden muss. Wird hingegen ein Randpunkt erreicht, war die Suche für das Kreissegment erfolgreich.
Auf diese Weise müssen alle Repräsentanten abgearbeitet werden. Entweder es wird eine Randsuche für sie gestartet, oder sie werden während der Randsuche für einen anderen Punkt geschluckt.
Wird jede Suche erfolgreich abgeschlossen, war der Konnektivitätstest erfolgreich, falls nicht, bedeutet dies, dass der getestete Stanzschritt nicht gültig ist.

4.3.2 Algorithmus basierend auf der medialer Achse

Für den Konnektivitätstest mit medialer Achse spielen die Gratpunkte natürlich die zentrale Rolle. Wird beim Kopieren der Flächenausschnitts festgestellt, dass kein Gratpunkt vom Stanzkopf überdeckt wird, kann hier die Funktion bereits beendet werden. Andernfalls muss, zusätzlich zur Kopie, eine Liste (gratList) erstellt werden. In dieser Liste werden diejenigen Punkte gespeichert, die in der Kopie des Feldausschnittes als Teil der neuen Randsegmente markiert wurden und gleichzeitig Punkte der medialen Achse sind.

Lokale Suche:

Die lokale Suche wird nach dem selben Muster ablaufen wie oben beschrieben. Der Unterschied besteht jedoch darin, dass nicht von einem beliebigen Punkt aus gesucht wird, sondern an einem der Punkte aus der gratList begonnen werden muss. Es ist hierbei auch nicht wichtig, dass alle markierten Punkte erreicht werden, sondern es müssen nur alle Punkte aus der gratList erreicht werden können. Wurden also alle Einträge in der gratList gefunden, kann die Funktion beendet werden. Ist der Stack jedoch abgearbeitet und es sind noch Punkte in der gratList übrig, muss die globale Suche durchgeführt werden.

Globale Suche:

Bei der globalem Suche besteht der Unterschied erneut nur aus Kleinigkeiten. Was beim oben beschriebenen Algorithmus die Repräsentanten der Kreissegmente waren, sind nun die Punkte aus der gratList. Diese stellen die losen Enden der Gratlinien dar, für die Verbindungen zum Rand gefunden werden müssen. Wie bisher wird auch hier mit einem Stack gearbeitet. Anstatt jedoch die komplette Restfläche als Suchraum zu durchlaufen, wird hier nur entlang der Gratlinien gegangen, was den Laufzeit-Unterschied der beiden Verfahren bewirkt.

5 Analyse der Algorithmen

In diesem Kapitel wird die Laufzeit des Algorithmus betrachtet. Hierbei werden die einzelnen Teile miteinander verglichen um zu erkennen, an welchen Stellen noch Optimierungspotenzial liegt. Zusätzlich werden einige zufällige Probleminstanzen betrachtet und es wird die Auswirkung verschiedener Gewichtungen in der Bewertungsfunktion auf das Endergebnis getestet.

5.1 Laufzeit

Zunächst muss hier festgehalten werden, dass über die Gesamt-Laufzeit des Algorithmus keine Aussage getätigt werden kann. Diese hängt im Grunde hauptsächlich davon ab, wie viele Durchläufe benötigt werden, bis die komplette Fläche überdeckt ist.
Da in jedem Durchlauf die Anzahl der möglichen Stanzpositionen ebenfalls stark variieren kann, kann auch hier keine allgemeingültige Aussage getroffen werden. Somit müssen für die Laufzeit-Analyse die Funktionen auf unterster Ebene herangezogen werden. Da die Zeitdauer für deren Berechnung jedoch Größtenteils im Mikrosekunden-Bereich liegt, welcher sehr anfällig auf Störungen durch andere Prozesse oder die Zeitmessung selbst reagiert, kann im wesentlichen nur mit statistischen Durchschnittswerten argumentiert werden.
Im folgenden wird von oben nach unten gearbeitet. Zunächst wird ein kompletter Stanzschritt unter die Lupe genommen. Danach wird jeweils derjenige Teilschritt näher betrachtet, der den größten Anteil der Zeit in Anspruch nimmt.

5.1.1 Vergleich zwischen Suchen und Stanzen:

Der komplette Stanzschritt besteht im wesentlichen aus drei Teilen, wie in Kapitel 3.2 beschrieben:

- Zunächst die Suche nach dem erfolgversprechendsten Stanzschritt
- Durchführung des Stanzschrittes
- Suche nach möglichen Problemstellen.

Die Suche nach möglichen Problemstellen ist hierbei genauso zu vernachlässigen wie die Durchführung des Stanzschrittes. Diese nehmen ungefähr 2% der benötigten Zeit in Anspruch. Hierbei wird die Beseitigung einer Problemstelle als Suche nach einem optimalen Stanzschritt gewertet, da kaum ein Unterschied bei den beiden Vorgängen existiert. (Kapitel 3.2.3).

5.1.2 Vergleich zwischen Bewertung und Gültigkeitstest

Die Suche nach einem möglichst optimalen Stanzschritt lässt sich ebenfalls aus 3 Teile unterteilen.

- Befindet sich die Position auf einem Punkt der medialen Achse
- Bewertung eines Stanzschrittes
- Testen der Gültigkeit eines Stanzschrittes.

Hierbei ist zu beachten, dass es sich um ein Selektionsverfahren handelt. Jeder Teil der Funktion sortiert einen Teil der möglichen Stanzschritte aus, welche dann die anderen Funktionsteile nicht mehr durchläuft.
Bei insgesamt 238 Mio. Aufrufen der Funktion wurden bereits über 98% der Fälle durch den Positionstest herausgenommen. Da dieser Test nur aus einer einzigen Abfrage besteht, konnte hier keine vernünftige Zeitmessung durchgeführt werden.
Von den verbliebenen 3,678 Mio. möglichen Stanzschritten wurden weitere 55% aussortiert, da ihre Bewertung schlechter war als die eines bereits berechneten gültigen Stanzschritts. Die Zeitkosten für die Bewertung beliefen sich hierbei auf knapp 23% der Gesamtzeit.
Die restlichen 77% der Zeit wurden demnach von den 1,689 Mio. durchgeführen Gültigkeitstests benötigt.

Mit Hilfe der Zeit und der Anzahl der Schritte lässt sich auch ein Durchschnittswert für die Teilfunktionen berechnen.
Die Bewertung eines einzigen möglichen Stanzschritts dauert demnach 4,21 Mikrosekunden. Ein einzelner Gültigkeitstest hingegen 77,33 Mikrosekunden.

5.1.3 Gültigkeitstest

Diese Funktion ist ähnlich aufgebaut wie die vorherige. Es können erneut drei Teile unterschieden werden, wobei bei jedem der Teile einige mögliche Kandidaten aussortiert werden. Die Teilschritte behandeln hier die Fragen:

- Liegt ein Gratlinien-Endpunkt unter dem Stanzkopf?
- Liegt genügend Material unter dem Stanzkopf?
- Verletzt der Stanzschritt die Konnektivitätsbedingung?

Bei den Abfragen nach den Gratlinien-Endpunkten und der Menge der vom Stanzkopf überdeckten Fläche wurden hierbei von den 1,689 Mio. möglichen Stanzschritten zunächst 13% und dann noch einmal 92% der Fälle aussortiert. Wie zu erwarten war, hielt sich der hierfür benötigte Zeitaufwand, mit knapp 7% in Grenzen.
Obwohl nur für 114.000 mögliche Stanzschritte ein Konnektivitätstest durchgeführt wurde beanspruchten diese die restlichen 93% der Zeit.

In absoluten Zahlen bedeutet dies für die Konnektivitätstest, eine durchschnittliche Zeitdauer von 93 Mikrosekunden. Dies ist die bei weitem größte Zeitspanne, die eine einzelne Funktion des Algorithmus benötigt.

5.1.4 Vergleich der Konnektivitätstests

Aufgrund der obigen Zeitmessungen lässt sich sagen, dass die Konnektivitätstests die meiste Zeit in Anspruch nehmen. Die zu testenden möglichen Stanzschritte werden deshalb so weit es geht reduziert, damit diese Funktion nicht zu oft aufgerufen werden muss.
Bei den obigen Messungen wurde ausschließlich der Konnektivitätstes basierend auf der medialen Achse verwendet. Der Intuition folgende müsste dies der schnellere der beiden Algorithmen sein. Im folgenden wird nun ein detaillierter Blick auf die beiden Funktionen geworfen, um mögliches Optimierungspotenzial zu erkennen.

Beide Funktionen werden zunächst in vier Teile zerlegt:

- Kopieren der Stanzkopf-Umgebung
- Lokale Suche
- Globale Suche
- Löschen der Kopie und bereinigen des Feldes

Bereinigen des Feldes bedeutet hier die Zurücksetzung der „bearbeitet“ Markierung, welche für die globale Suche benötigt wurde.

In den beiden Tabellen werden nun die berechneten Werte für die beiden Funktionen dargestellt.

Funktion mit medialer Achse	Prozentualer Anteil an der kompletten Zeit	Absolutwert für einmaligen Durchlauf in Mikrosekunden
Kopie erstellen	11,72	33,14
Lokale Suche	2,65	7,49
Globale Suche	41,94	122,13
Löschen und Bereinigen	43,69	123,53

Funktion ohne mediale Achse	Prozentualer Anteil an der kompletten Zeit	Absolutwert für einmaligen Durchlauf in Mikrosekunden
Kopie erstellen	3,92	15,44
Lokale Suche	3,87	15,26
Globale Suche	56,55	228,8
Löschen und Bereinigen	35,66	140,46

Die Anzahl der Fälle, in denen die lokale Suche ausreicht hält sich hierbei in Grenzen. Nur in knapp 3% der Fällen kann so die Zeitaufwendige globale Suche vermieden werden.

Da das Erstellen der Kopie, sowie das Löschen derselben und das Zurücksetzten des Feldes nahezu komplett auf die globale Suche zurückgeführt werden kann, bietet sich hier die Überlegung nach einer alternativen Implementierung an. Kann ein Verfahren gefunden werden, welches Änderungen am Feld vermeiden kann könnte hier schon ein Großteil der Zeit eingespart werden.

5.2 Auswirkung von verschiedenen Gewichtungen auf Problembeispiele

Es wird nun anhand von einigen Beispielen gezeigt, wie sich Änderungen in der Gewichtung der in Kapitel 3.2.1 vorgestellten „Stellschrauben“ für die Bewertungsfunktion auswirken können.

Beispiel 1 :

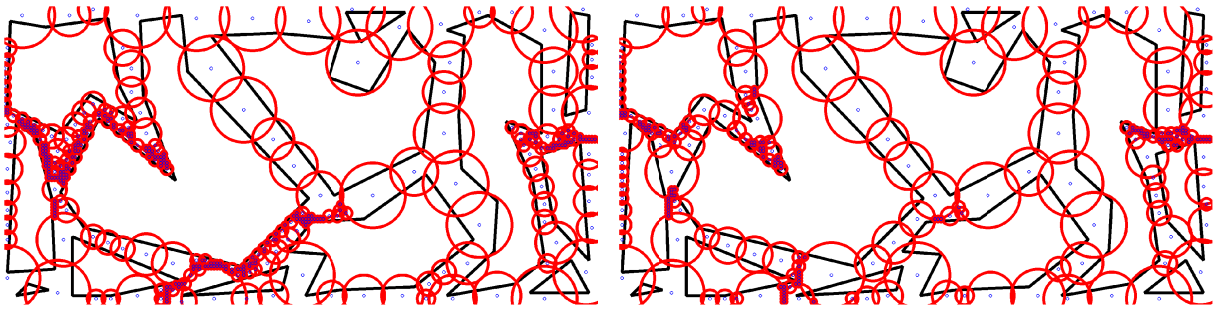

Abb. 19: Bsp. 1: Unterschied durch Ausnahmeregel

Hier handelt es sich um ein Problem, bei dem die Sonderbehandlung von Entscheidender Bedeutung ist. Bei Abb.19 wurde links zunächst darauf verzichtet und man kann sehr schön erkennen, wo die engen „Schläuche“ entstanden sind. Es wurden für die Lösung hier 362 Stanzschritte benötigt. Rechts wurde die Sonderbehandlung eingeführt und man erkennt, das zumindest der mittlere Bereich viel schöner abgedeckt werden konnte. Somit konnte die Anzahl der Stanzschritte so auf 245 reduziert werden.

Durch eine Erhöhung für die Belohnung von niedrigen Gratlinien war es möglich das Ergebnis weiter zu verbessert, sodass in der linken Lösung von Abb.20 nur noch 222 Stanzschritte erforderlich waren. Durch die Änderung der Bedingung für die Sonderbehandlung, sodass diese bereits früher Eintritt, konnte noch einmal eine Reduktion auf 140 Stanzschritte erreicht werden (Abb.20, Rechts). Hierbei ist jedoch auch zu erkennen, dass die rechte Seite der Probleminstanz vor dieser Änderung besser überdeckt wurde.

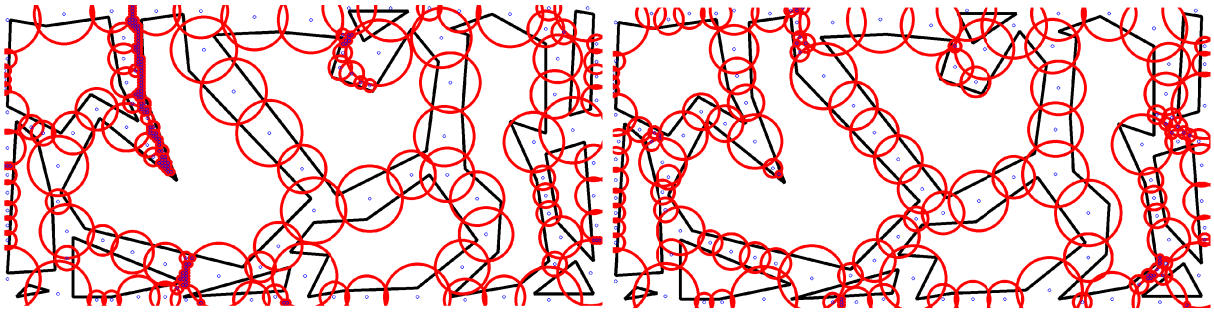

Abb. 20: Bsp. 1: Änderung durch Bewertung niedrige Gratpunkte (links), Änderung Ausnahme (rechts)

Beispiel 2 :

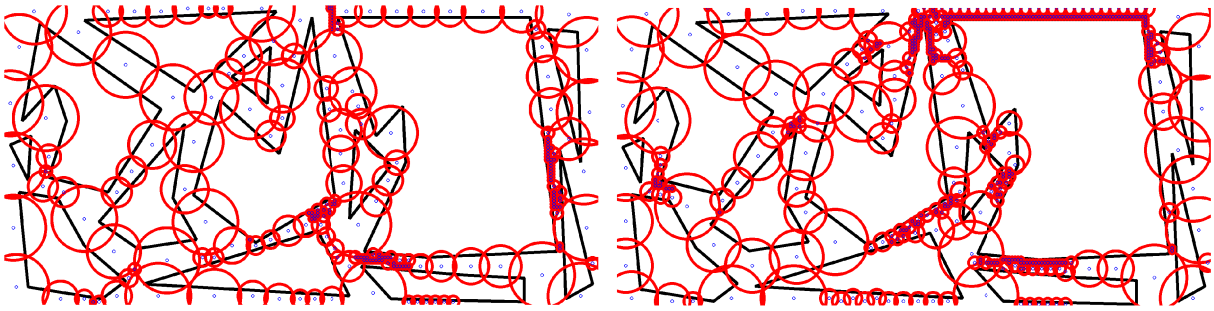

Abb. 21: Bsp. 2 : Unterschiedliche Auswirkung der Änderung bei der Ausnahme

An diesem Beispiel kann schön gezeigt werden, wie sich die oben durchgeführte Änderung an der Bedingung der Ausnahme negativ auswirken kann. In Abb.21 links sieht man die Lösung ohne Änderung (206 Stanzschritte), rechts mit Änderung (391 Stanzschritte). Es wäre hier auch möglich komplett auf die Ausnahmeregelung zu verzichten, da das Ergebnis nur unwesentlich schlechter ausfällt (218 Stanzschritte).

Abb. 22: Bsp. 2 : Ohne Bewertung niedriger Gratpunkten (links), Ohne Belohnung für Randentfernung (rechts)

Auch kleine Änderungen an den Parametern können große Auswirkungen auf die Ergebnisse haben. Wird die Bewertung der niedrigen Gratlinien nur geringfügig geändert werden aus 206 Stanzschritten plötzlich 387. Wird bestraft, dass mehr Punkte mit Distanz 3 als mit Distanz 4 unter einem Stanzkopf liegen, so verbessert sich das Ergebnis von 289 auf 196 Stanzschritte. Die besten Ergebnisse lassen sich in diesem Beispiel erzielen, wenn die Belohnung, für die Entfernung von möglichst viel Rand komplett weggelassen wird.

Beispiel 3 :

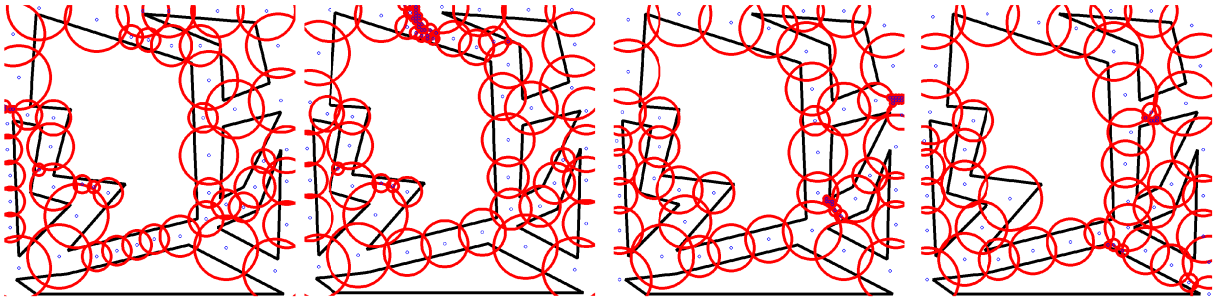

Abb. 23: Verschieden Gewichtungen für Beispiel 3

In Abb.23 werden vier mögliche Ergebnisse für Beisiel 3 gezeigt. Ausgehend vom Linken (54 Stanzschritte) wurden folgende Gewichtungen geändert:
zweites von links: Änderung der Ausnahmebedingung (93 Stanzschritte)
zweites von rechts: Bedingung wurde herausgenommen, dass mehr kleine Distanzwerte als große enthalten sein sollen (58 Stanzschritte)
rechts : Vergleich der Randlänge bevor und nach dem Stanzschritt wurde hinzugenommen (51 Stanzschritte).

Beispiel 4 :

Abb. 24: Verschiedene Gewichtungen für Beispiel 4

Hier wird gleich vorgegangen wie bei Beispiel 3. Links die Ausgangskonfiguration (110 Stanzschritte), die für die anderen Ergebnisse leicht abgeändert wurde.
Zweites von links: Verstärkung der Bedingung für die Randverkürzung (77 Stanzschritte)
Zweites von rechts: Die Belohnung für das Entfernen niedriger Gratlinien wurde erhöht. (134 Stanzschritte)
Rechts: Bestrafung wenn mehr Punkte mit hohen Distanzwerten als Puntke mit niedrigen Distanzwerten entfernt werden (209 Stanzschritte).

6 Entfernen eines Polygons von der Grundfläche

Um alle Punkte, die innerhalb der Polygone liegen, zu entfernen müssen diese Punkte zunächst berechnet werden. Ein einfacher Algorithmus zum lösen dieses Problems sieht wie folgt aus:
Zeichne durch einen Punkt P eine Gerade mit beliebiger Steigung. Beispielsweise eine waagerechte Gerade (Steigung = 0). Laufe nun entlang der Geraden, beginnend bei P in eine beliebige Richtung und zähle die Anzahl der Segmente des Polygons, die von dieser Geraden geschnitten werden. Ist diese Anzahl gerade liegt der Punkt außerhalb des Polygons, ist er ungerade liegt der Punkt innerhalb.
Bei diesem Algorithmus muss für jeden getesteten Punkt jedes Segment des Polygons überprüft werden. Die Laufzeit beträgt also O(k*n) für k = Anzahl der zu testenden Punkte und n = Anzahl der Segmente des Polygons.
Da im vorliegenden Fall, mit diesem Verfahren, alle Punkte des Polygons sowie alle umliegenden getestet werden müssen, ist dieses Vorgehen nicht zu empfehlen.
Es bietet sich hier an, das Polygon als ganzes zu betrachten, wodurch nicht mehr einzelne Punkte getestet werden müssen.

6.1 Grundlegende Idee

Die Idee ist, von links nach rechts eine senkrechte Linie über das Polygon zu ziehen, eine sogenannte Sweepline. Für jeden x-Wert können so ein oder mehrere y-Wert-Intervalle berechnet werden. Alle Punkte, die innerhalb dieser Intervalle liegen, können somit auf einmal von der Fläche entfernt werden.

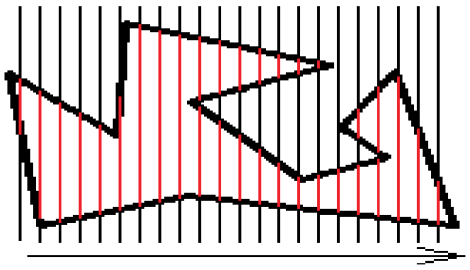

Abb. 25: Beispiel für senkrechte Sweepline

Für die Berechnung der Intervalle muss das Polygon zunächst in Dreiecke zerlegt werden. Da hier nur simple Polygone, also Polygone ohne Löcher, betrachtet werden, ist dieses Problem in O (n log n) lösbar, wenn auch hier ein Sweepline Algorithmus zum Einsatz kommt [Ber-

g00]. Im Jahr 1988 wurde von Tarjan und Van Wyk ein O(n log log n) Algorithmus entwickelt [Tarj88], woraufhin ein Jahr später schon ein O(n log* n) Verfahren vorgestellt wurde [Clark89]. 1991 zeigte schließlich Bernard Chazelle, dass jedes simple Polygon in Linearzeit trianguliert werden kann [Chaz91]. Mit abnehmender Laufzeit werden diese Algorithmen jedoch immer komplizierter, sodass sich die Umsetzung dieser Verfahren in den wenigsten Fällen lohnt.
Da eine solche Zerlegung beim vorliegenden Problem nicht allzu oft berechnet werden muss wurde hier nur ein einfacher „ear clipping“-Algorithmus mit der Laufzeit von $O(n^3)$ implementiert. Die Idee dabei ist, dass es in einem Polygon mit mindestens vier Knoten immer ein sogenanntes „ear“ gibt.

Definition „ear“

> Ein „ear“ ist ein Dreieck, dessen Eckpunkte aus drei aufeinander folgenden Punkte eines Polygons bestehen. Hierbei muss die komplette Fläche dieses Dreiecks innerhalb des Polygons liegen.

Für jedes mögliche „ear“ entlang des Polygons gilt es nun zu testen, ob dieses „ear“, bestehend aus den direkt aufeinander folgenden Polygon-Eckpunkten p1, p2 und p3, komplett im Polygon enthalten ist. Hierfür wird für alle anderen Eckpunkt des Polygons geprüft, ob einer davon innerhalb des von p1, p2 und p3 aufgespannten Dreiecks liegt.

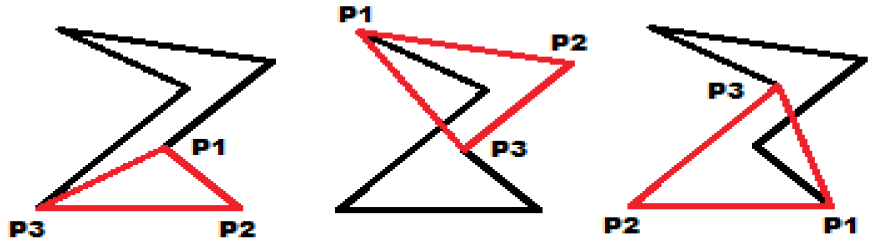

Abb. 26: Links: gültiges "ear"; Mitte und Rechts: kein gültiges "ear"

Wird kein solcher Eckpunkt gefunden, ist das überprüfte Dreieck ein gesuchtes „ear“ und kann vom Polygon entfernt werden. Dieser Schritt wird so lange wiederholt, bis vom Polygon nur noch ein Dreieck übrig bleibt.
Für jedes der entfernten Dreiecke, einschließlich für das am Ende übrig bleibende, werden nun von rechts nach links die entsprechenden y-Wert-Intervalle berechnet. Alle Punkte, die innerhalb dieser Intervalle liegen können dann von der Fläche entfernt werden.

Für die Aktualisierung der Distanzwerte werden zusätzlich jedoch noch alle Randpunkte des Polygons benötigt. Diese können beim besprochenen Verfahren nicht automatisch mitgeliefert werden. Deshalb werden zum Ende nun nocheinmal alle Randsegmente des Polygons abgelaufen und die besuchten Punkte als Randpunkte gespeichert.

6.2 Zerlegung des Polygons

Der erste Schritt ist die Zerlegung des Polygons in Dreiecke. Hierzu wird, wie oben bereits beschrieben, das Polygon Schritt für Schritt verkleinert, indem ein geeignetes Dreieck gesucht und entfernt wird (ear-clipping).
Hat ein Eckpunkt p_i des Polygons einen Innenwinkel von weniger als 180°, kann er genauer untersucht werden. Es wird eine imaginäre Verbindung zwischen seinen beiden Nachbarpunkten, p_{i-1} und p_{i+1}, gezogen, womit das zu untersuchende Dreieck definiert ist. Für jeden anderen Punkt des Polygons wird nun die Lage zu den drei Segmenten des Dreiecks bestimmt.
Hierbei werden die Segmente in derjenigen Richtung betrachtet, in der sie bei einer Umrundung des Dreiecks entlanggelaufen werden. Liegt ein Punkt für alle Segmente auf der selben Seite, also entweder links von allen oder rechts von allen, so befindet er sich innerhalb des Dreiecks, welches demnach kein gültiges „ear“ ist. In diesem Fall muss der jeweils nächste Eckpunkt des Polygons untersucht werden. Liegt jedoch keiner der Punkte innerhalb des Dreiecks, so kann der Punkt p_i aus dem Polygon entfernt werden. Für das Dreieck p_i, p_{i-1}, p_{i+1} gilt es dann noch die entsprechenden Intervalle zu berechnen.

6.3 Bestimmung der Intervalle

Für die Bestimmung der Intervalle werden die drei Punkte des Dreiecks in x-Richtung sortiert, p1, p2, p3. Es müssen nun alle x-Werte zwischen p1 und p3 durchlaufen und für jeden dieser x-Werte ein Intervall bestimmt werden. Dabei werden entweder die min-Werte oder die max-Werte vom Segment p1p3 bestimmt und die jeweils anderen Werte von den Segmenten p1p2 und p2p3. Daraus ergibt sich eine logische Unterteilung der Funktion in zwei Teile. Teil eins bearbeitet die x-Werte von p1 bis p2 und Teil zwei diejenigen von p2 bis p3. Mit Hilfe der Steigung eines Segments kann für jeden x-Wert ein zugehöriger y-Wert für das Segment berechnet werden. Für den ersten Teil setzt sich das Intervall demnach aus den Werten der Segmente p1p2 und p1p3 zusammen, für den zweiten Teil wird dann das Segment p1p2 durch p2p3 ersetzt.
Auch hier spielt die Rundung für das Aussehen und die Korrektheit der Figuren eine wichtige Rolle. Ziel hierbei sollte es sein, dass ein Punkt genau dann Teil des Segments ist, wenn die Distanz zwischen dem y-Wert des Segments zum y-Wert des Punkts an der Stelle x kleiner als 0,5 ist.

Bei genauerer Betrachtung lässt sich hier jedoch ein Problem erkennen. Da die Segmente bisher nur in x-Richtung diskretisiert werden, also für jeden x-Wert ein entsprechender y-Wert berechnet wird, werden Segmente mit einer Steigung größer eins weniger genau abgetastet als diejenigen mit einer Steigung kleiner eins. Vor allem bei sehr großen Steigungen wirkt sich dies deutlich aus.

Aus diesem Grund sollten Segmente mit einer Steigung die größer als eins ist in y-Richtung abgetastet werden. Auch hier gilt logischerweise die oben geforderte Regel für die Rundung, allerdings mit vertauschten Koordinatenachsen. Bei der Integration dieses Vorgehens in die aktuelle Funktion müssen einige Kleinigkeiten beachtet werden. Um mit bisherigen Schleifen, die über die x-Werte definiert wurden, weiterarbeiten zu können und trotzdem die Segmente in y-Richtung abzutasten, muss eine geschachtelte Schleife eingeführt werden. Für einen festen x-Wert soll nun solange entlang des Segments in y-Richtung gelaufen werden, bis sich der Funktionswert (x-Wert) ändert. Erst dann kann die äußere Schleife einen Schritt weitergehen.

Bei diesem Vorgehen gilt es jedoch zusätzlich zu beachten, ob es sich um eine linke oder um ein rechte Kante des Dreiecks handelt. Um zu erkennen, warum dies wichtig ist betrachten wir ein Segment mit einer positiven Steigung von fünf.
Beginnend am Ursprung ist dieser logischerweise Teil des Segments. Bevor die äußere Schleife (x-Richtung) von 0 auf 1 springen kann, läuft die innere Schleife (y-Richtung), wie oben beschrieben, entlang des Segments, bis sich der Funktionswert (x-Wert) ändert. Dies wird bei y=3 der Fall sein, da der Funktionswert hier 0,6 annimmt. Für x = 0 gibt es also 3 Punkte, die zum Segment gehören: (0/0), (0/1), (0/2). Stellt dieses Segment nun die obere Kante des Dreiecks dar, muss logischerweise der zuletzt besuchte, also der obere Punkt (0/2) als max-Wert gewählt werden. Ist es jedoch die untere Kante des Dreiecks, ist der erste (der untere) Punkt (0/0) der Wichtige.

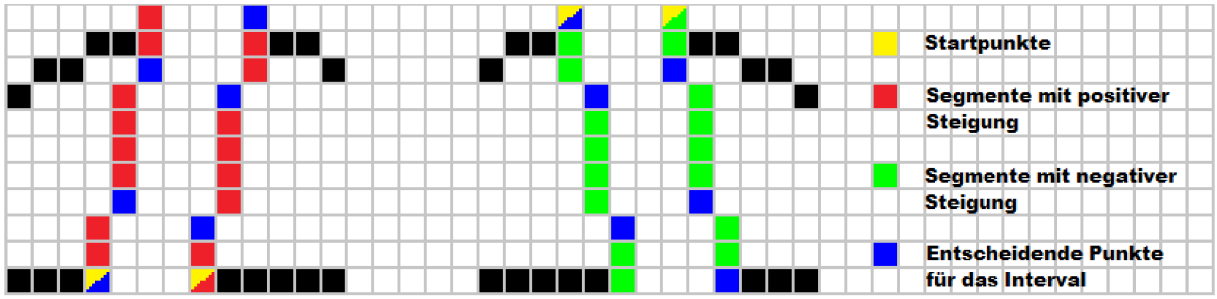

Abb. 27: Unterscheidung linkes und rechtes Segment

Der Zusammenhang, obere Kante - zuletzt besuchter Punkt, bzw. untere Kante - zuerst besuchter Punkt, stimmt allerdings nur, solange das Segment eine positive Steigung hat. Ist die Steigung negativ, dreht sich dieser Zusammenhang um. Da die Reihenfolge der besuchten Punkte vom Algorithmus vorgegeben wird, werden sowohl die Informationen über das Vorzeichen der Steigung, als auch über die Lage (obere/untere Kante) des Segments benötigt, um feststellen zu können, ob der zuerst oder der zuletzt besuchte Punkt wichtig ist. Diese Informationen lassen sich jedoch zusammenfassen, indem die Lage des Segments in x-Richtung bestimmt wird. So ist sowohl ein oberes Segment mit positiver Steigung, als auch ein unteres Segment mit negativer Steigung eine linke Kante des Dreiecks. Somit werden die beiden Fälle in denen der zuerst besuchte Punkt von Bedeutung ist abgedeckt. Die anderen beiden Fälle, in denen der zuletzt besuchte Punkt der entscheidende ist, sind demnach dann rechten Kanten.

6.4 Bestimmung der Randpunkte

Für die Bestimmung der Randpunkte muss beachtet werden, dass nur solche Punkte als Randpunkte erkannt werden, die auch als Punkte des Polygons erkannt und von der Restfläche entfernt wurden. Es wird schnell klar, dass die Segmente mit einer flachen Steigung nach dem normalen Muster bearbeitet werden können. Begonnen wird beim weiter links liegenden Endpunkt und es wird für jeden x-Wert ein y-Wert berechnet.
Da die Richtung, in die gearbeitet werden muss hier nicht mehr vorgegeben ist, könnten die Segmente, deren Steigung einen Absolutwert von größer als eins besitzen, mit einer Vertauschung der Koordinatenachsen genau gleich bearbeitet werden, vorausgesetzt, dass durch die Veränderung der Berechnung keine anderen Ergebnisse erzielt werden. Betrachtet man hier nun die geschachtelten Schleifen bei der Berechnung der Intervalle, so fällt auf, dass in der inneren Schleife alle relevanten y-Werte genau einmal durchlaufen werden. Die Position der äußeren Schleife gibt dabei den korrekt gerundeten x-Wert wieder, der bei einer Bearbeitung in y-Richtung berechnet werden würde. Somit ist klar, dass die Berechnung wie geplant durchgeführt werden kann.

7 Mögliche Erweiterungen

Im Rahmen einer Diplomarbeit kann ein solch komplexes Problem natürlich nicht bin ins letzte Detail bearbeitet werde. Im folgenden sollen einige Punkte umrissen werden, auf die in dieser Arbeit nicht mehr eingegangen werden konnte, denen jedoch Beachtung geschenkt werden sollte, da sie teilweise für eine reale Anwendung essenziell sind oder zumindest einiges an Optimierungspotenzial versprechen.

7.1 Essenzielle Erweiterungen

Weiter Stanzköpfe zulassen :

Dies ist wohl eine der Erweiterungen, die zwingend notwendig sein werden für eine mögliche Nutzung der Algorithmen in der Praxis. Um eine Erweiterung der Stanzkopfauswahl möglichst einfach zu gestalten, wurde die Verwaltung der Stanzköpfe bereits in eine extra Klasse (Tool) ausgelagert.
Sofern darauf geachtet wird, dass neue Stanzköpfe eine konvexe Form besitzen, sollte deren Einbettung in die Algorithmen kein großes Problem darstellen. Sollen allerdings auch nicht konvexe Formen zugelassen werden, wird es zwangsläufig zu Problemen kommen, vor allem was den Umgang mit der medialen Achse anbelangt. Bei der Aktualisierung derselben nach einem Stanzschritt werden dann die gleichen Sonderfälle zu beachten sein, die bisher nur beim Entfernen von nicht konvexen Polygonen von Bedeutung waren. Weitere Schwierigkeiten sind nicht auszuschließen.
Des weiteren wird der Konnektivitätstest basierend auf der medialen Achse in seiner jetzigen Form nicht mehr anwendbar sein, da durch nicht konvexe Stanzköpfe auch Bereiche abgetrennt werden können, die nicht durch eine Gratlinie definiert werden.

Umgang mit noch nicht entfernten Werkstücken :

Auch dieser Fall wird in der Praxis sicherlich von Bedeutung sein. Sollten die ausgestanzten Werkstücke noch nicht entfernt worden sein, so wird es natürlich von entscheidender Bedeutung sein, dass sich die Stanzköpfe beim Zerstanzen nicht mit den Werkstücke überlappen. Manche Randsegmente der Restfläche spielen somit eine entscheidende Rolle und dürfen nicht vom Stanzkopf überschritten werden. Bei der Verwendung von ausschließlich runden Stanzköpfen wäre eine solche Bedingung fatal, da vor allem die geraden Randsegmente mit einer sehr großen Anzahl an Stanzhüben, mit sehr kleinen Stanzköpfen, entfernt werden müssten. Somit ist die Erweiterung der Stanzkopfauswahl eine Voraussetzung für diese Randbedingung.

Abgerundete Bauteile :

Die Beschränkung auf polygonale Werkstücke ist bei den meisten praktischen Anwendungsbereichen wohl nicht möglich. Für die Zulassung anderer Werkstücke müssen jedoch einige Dinge berücksichtigt werden. Das Hauptproblem hierbei liegt erneut bei der medialen Achse. Bisher wurde angenommen, dass jede „Extremität“ der Restfläche durch eine Gratlinie definiert wird, die bis zum Rand der Fläche verläuft. Werden jedoch abgerundete Werkstücke zugelassen, kann dies nicht mehr garantiert werden, da bei einer runden Ausbuchtung keine wirkliche Ecke der Restfläche definiert ist und somit der Endpunkt der entsprechenden Gratlinie irgendwo im Innern der Fläche liegt. Dies führt zu verschiedenen Problemen.

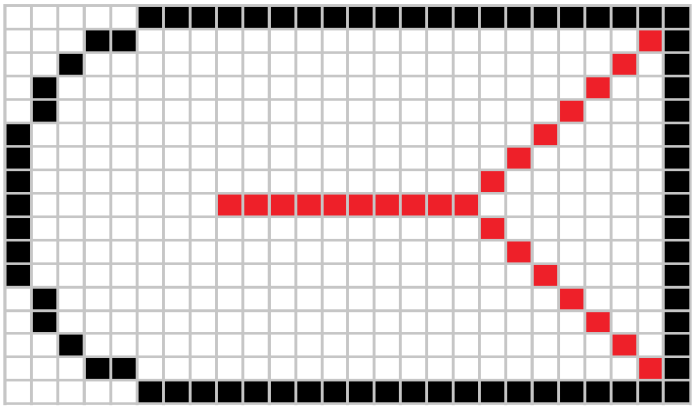

Abb. 28: Mediale Achse bei runder Flächenausbuchtung

Zunächst betrifft dies die Vorauswahl der möglichen Stanzpositionen. Die Endpunkte der Gratlinien wurden bisher als Eckpunkte der Restfläche betrachtet und boten eine gute Auswahl für eine mögliche nächste Stanzposition. Inwiefern Gratlinien-Endpunkte, die mitten im Feld liegen, hier Beachtung finden sollten müsste erst noch geklärt werden.
Des weiteren wird auch hier die Korrektheit des Konnektivitätstests basierend auf der medialen Achse ausgehebelt. Eine runde „Extremität“ wird nicht bis zum Rand von einer Gratlinie durchzogen, womit die zentrale Bedingung des Algorithmus verletzt wird.

7.2 Erweiterungen mit Optimierungspotenzial

Umsortierung der Stanzschritte :

Der Algorithmus wurde bisher in erster Linie auf die Anzahl der Stanzschritte optimiert. Die Randbedingung, dass die Stanzkopfwechsel minimiert werden sollen, wurde zwar in die Bewertungsfunktion integriert, findet sonst aber kaum Beachtung. Eine Idee, die es möglicherweise lohnen könnte weiterzuverfolgen, wäre eine Nachbearbeitung des Stanzplanes. Sofern zwei aufeinander Folgende Stanzschritte Unabhängig voneinander sind, können diese vertauscht werden. Eine Abhängigkeit zwischen den beiden besteht dann, wenn sich die Stanzköpfe überlappen oder wenn der zweite Stanzschritt ohne den ersten nicht Ausführbar ist, da sonst die Konnektivität beim zweiten verletzt werden würde. Vor allem bei kleineren Stanzköpfen liegt die Vermutung nahe, dass hier einiges an Spielraum existiert.
Durch solche Vertauschungen lässt sich möglicherweise eine Reihenfolge der Stanzschritte finden, bei der einige Wechsel des Stanzkopfes vermieden werden können.

Verschärfte Vorauswahl :

Die Testläufe wurden bisher auf einem recht überschaubaren Feld von der Größe 100*100 durchgeführt. Die Stanzkopfradien waren begrenzt auf 0-10. Selbst bei solchen, doch recht kleinen Probleminstanzen lag die Laufzeit des Algorithmus im Bereich von 20-30 Sekunden, um die 80-200 Stanzpositionen zu berechnen. Für größere Felder, und damit auch eine größere Menge an unterschiedlichen Stanzköpfen, wird die Laufzeit des vorgestellten Algorithmus sehr schnell aus dem Ruder laufen.
Ziel muss es sein, die Anzahl der Stanzköpfe und/oder die Anzahl der möglichen Stanzpositionen zu verringern. Hier scheint eine verschärfte Vorauswahl die beste Alternative zu sein.

Alternative Diskretisierung :

Eine alternative Lösung zur Diskretisierung des Feldes wäre die Verwendung eine Bienenwaben-Struktur, wodurch sich Liniensegmente, die nicht senkrechte oder waagerechte sind, besser darstellen ließen.

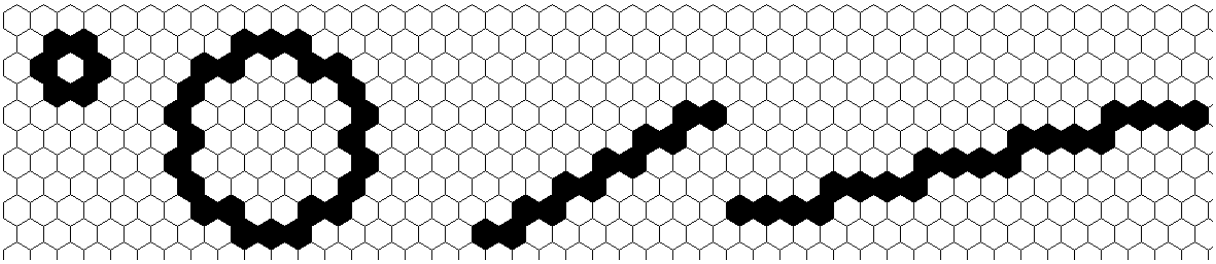

Abb. 29: Alternative Diskretisierung, Bienenwaben-Struktur

Ein weiterer großer Vorteil einer solchen Darstellung wäre, dass eine eindeutige Nachbarschaftsbeziehung zwischen zwei Punkten existiert. Beim verwendeten Modell gibt es hierfür zwei mögliche Betrachtungen. Da jeder Punkt von einem Quadrat repräsentiert wird, hat er vier Nachbarn, mit denen er Kante an Kante liegt. Es existieren allerdings nocheinmal vier Nachbarn, welche nur über die Eckpunkte benachbart sind. Somit können einerseits nur die vier Punkte als Nachbarn bezeichnet werden, die Kante an Kante mit dem Ausgangspunkt liegen, andererseits können auch die an den Eckpunkten anliegenden Punkte mit dazu gezählt werden. In diesem Fall wären es acht Nachbarpunkte. Diese Unterscheidung kann zu Problemen führen, wie im folgenden Beispiel deutlich wird.
Bei einer Bienenwaben-Struktur hingegen ist die Nachbarschaft eindeutig geregelt. Jeder Punkt hat genau sechs Nachbarpunkte, womit ein Problem wie das obere nicht entstehen kann.
Ein weiterer Vorteil ist, dass alle sechs Nachbarpunkte genau die gleiche Distanz zum Ausgangspunkt haben, was bei einer Achter-Nachbarschaft nicht der Fall ist und was in manchen Fällen beachtet werden muss.
Da das Arbeiten mit diese Struktur allerdings komplizierter und weniger intuitiv ist, wurde hier darauf verzichtet.

Mögliche Einbettung in einen evolutionären Algorithmus :

Bei der Bewertung der möglichen Stanzschritte handelt es sich um eine Funktion mit vielen Parametern. Die meisten davon stehen direkt oder indirekt in Beziehung zueinander und beeinflussen sich gegenseitig. Eine optimale Gewichtung der Parameter, sodass der Algorithmus

für jede Probleminstanz sehr gute Ergebnisse erzielt, wird es wohl nicht geben. Somit scheint es sinnvoll zu sein, den Algorithmus mehrfach mit verschiedenen Gewichtungen laufen zu lassen, um am Ende eine größere Chance auf eine sehr gute Lösung zu erhalten.
Da die Beziehungen der Parameter untereinander so vielfältig sind und die Möglichkeiten für Veränderungen so zahlreich, ist es nur bedingt möglich die Gewichtungen durch rationale Überlegungen zu optimieren. Hier eignet sich deshalb die Einbettung des Programms in einen evolutionären Algorithmus. Mit Hilfe des Zufalls und durch die Bewertung der bereits getesteten Gewichtungen wird ein solches Vorgehen zumindest ein lokales Maximum des Suchraums finden.

Erweiterung der Bewertungsfunktion :

Schlussendlich gibt es natürlich auch noch die Möglichkeit, die Bewertungsfunktion zu erweitern und zu optimieren. Auch hierfür werden noch zwei Beispiele genannt. Zunächst kann die bisher getroffene Einschränkung auf ausschließliche Betrachtung des Bereichs um den Stanzkopf aufgehoben werden. Hier könne beispielsweise eine Arte Gratlinien-Länge eingeführt werden, welche für jeden Punkt der medialen Achse die Entfernung zum Rand der Grundfläche angibt. Mit Entfernung ist hier die Länge des Pfades entlang der medialen Achse gemeint. Dies würde zwar mit der lokalen Berechnung der medialen Achse in Konflikt stehen, es könnte jedoch auch die Entstehung von sehr langgezogenen Bereichen mit dieser Information verhindert werden, indem Regionen, die sehr weit vom Rand entfernt liegen, bevorzugt bearbeitet würden.
Ein weiteres Beispiel bezieht die Betrachtung des näheren Umfeldes um den Stanzkopf mit ein. Hiermit wäre ein weitaus effektiveres Hilfsmittel zur Vermeidung von Engstellen gegeben. Zusätzlich könnte die Umgebung dahingehend geprüft werden, ob sich hier eine weitere gute Position findet, die durch den aktuellen Stanzschritt zerstört wird.

Quellenverzeichnis

Hildinger, *Algorithmen zur Optimierung von Packungsproblemen*, Fachschaft Informatik der Uni Stuttgart, 2014

Vazirani, *Approximation Algorithms*, Springer-Verlag, 2001

Heinrich-Litan, Lübbecke, *Rectangle covers revisited computationally,* Journal of Experimental Algorithmics 11: 2.6, 2007

Levcopoulos, Gudmundsson, A*pproximation algorithms for covering polygons with squares and similar problems,* Randomization and Approximation Techniques in Computer Science, 1997

Cardoso, Abreu, *An Efficient Distributed Algorithm for Computing Minimal Hitting Sets*, Graz, Austria, 2014

Bar-Yehuda, Ben-Hanoch, *A Linear-Time Algorithm for Covering Simple Polygons with Similar Rectangles,* International Journal of Computational Geometry & Applications 06: 79, 1996

Christian Royer: *Simultane Optimierung von Produktionsstandorten, Produktionsmengen und Distributionsgebieten*. Utz, Wiss., München 2001, S. 55.

Marc de Berg, Marc van Kreveld, Mark Overmars, und Otfried Schwarzkopf: *Computational Geometry* (2nd revised ed.), Springer Verlag, 2000, S. 45–61.

Tarjan, Van Wyk, *An O(n log log n)-time algorithm for triangulating a simple polygon,* SIAm Journal on Computing 17, 1988, (1): 143–178

Clarkson, Tarjan, van Wyk, *A fast Las Vegas algorithm for triangulating a simple polygon*, Discrete and Computational Geometry 4: 423–432

Chazelle, *Triangulating a Simple Polygon in Linear Time*, Discrete and Computational Geometry 6: 485–524

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Unterschrift: 

Stuttgart, 26.09.2014