Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Artem Chernyshov

Einsperren von Polygonen mit Barrieren

20. Oktober 2007

Seminararbeit im WS 2007/2008

Zusammenfassung

In dieser Arbeit wird eine Übersicht über das Problem des Einsperrens von Polygonen mit Barrieren gegeben und wir lernen Algorithmen kennen, die alle möglichen einsperrenden Positionierungen der Barrieren berechnen und ebenfalls schnell entscheiden können, ob eine Positionierung einsperrend ist.

Als Kernliteratur für diese Seminararbeit diente [8].

Inhaltsverzeichnis

1 Einführung

Das Problem des Einsperrens von Polygonen kann wie folgt definiert werden: Wir suchen einen Algorithmus, der eine beschränkte Anzahl an Barrieren um ein Polygon positioniert, so dass es unmöglich wird, das Polygon aus der Umgebung zu bewegen, ohne dass die Barrieren im Weg sind. Außerdem verlangen wir, dass alle möglichen Platzierungen von Barrierentupeln berechnet werden, so dass später nachgeprüft werden kann, ob eine angefragte Platzierung das Polygon einsperrt.

Im gegebenen Problem werden wir zuerst Algorithmen für zwei scheibenförmige Barrieren untersuchen. Wir werden zeigen, dass das Problem für punktförmige Barrieren in $O(n^2\ log\ n)$ gelöst werden kann und mit einer passenden Datenstruktur der Größe $O(n^2)$ in $O(log\ n)$ nachgeprüft werden kann, ob eine angefragte Platzierung einsperrend ist. Zu diese Ergebnis kamen Pipattanasomporn und Sudsang [6], jedoch werden wir diese Ergebnisse auf scheibenförmige Barrieren erweitern. Für die Lösung des Problems werden wir Pseudotriangulation der Umgebung, Zellendekomposition, Konnektivitätsgraphen und eine Technik zum Verarbeiten von Ereignissen verwenden.

Für das Einsperren mit drei Barrieren werden wir das Zwei-Barrieren-Problem um eine dritte Barriere erweitern und die Platzierung für die ersten zwei Barrieren vorgeben. Es werden alle möglichen einsperrenden Platzierungen für die dritte Barriere berechnet. Wir werden zeigen, dass die dritte Barriere einen sogenannten Gleichgewichtsgriff erzeugt und dass wir diese Tatsache ausnutzen können, um einen Algorithmus zu erstellen, der alle einsperrenden Platzierungen für nicht-konvexe Polygone berechnet. Die Laufzeit dieses Algorithmus wird auf $O(n^6\ log^2\ n)$ deterministischer Zeit und $O(n^6\ log\ n(log\ log\ n)^3)$ erwarteter Zeit abgeschätzt.

2 Definitionen und Annahmen

Das gegebene einfache Polygon P hat keine Löcher und ist begrenzt durch n Kanten. Sei P_d die Minkowski-Summe von P mit einer am Ursprung zentrierten Scheibe mit Radius d. Dabei sei erwähnt, dass die Minkowski-Summe als $P_d = P \oplus \Delta_d$ mit $A \oplus B = \{a + b | a \in A, b \in B\}$ definiert ist. Abbildung 1(a) zeigt, wie das Polygon P mit der Scheibe Δ_d am Ursprung des Koordinatensystems ausgerichtet wird und jeder Punkt einer Figur wird mit jeweils einem Punkt der anderen Figur über Addition verknüpft wird. In der Abbildung 1(b) ist das Polygon P bereits von einer d-dicken Hülle umgeben. Die Platzierung von scheibenförmigen Barrieren mit Radius d um P ist also äquivalent mit der Platzierung von punktförmigen Barrieren um das generalisierte Polygon P_d, das von Geraden und Bögen umgeben ist. Für eventuell entstandene Löcher im Polygon P_d reicht schon eine Barriere für den ein-

sperrenden Zustand, deshalb werden wir uns nur auf die Platzierungen von zwei bzw. drei Barrieren konzentrieren.

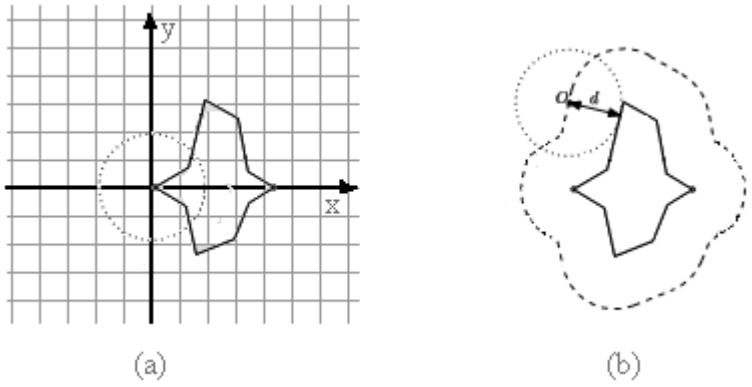

Abbildung 1: Minkowski-Summe von P

O.B.d.A können wir davon ausgehen, dass das Polygon P_d in einem ausreichend großen Rechteck B eingeschlossen ist. Sei $F \subset \mathbb{R}^2$ als $F = B \setminus int(P_d)$ definiert. Somit ist F die gesamte Umgebung für die Platzierung von Barrieren. Im Verlauf dieser Seminararbeit werden wir von punktförmigen Barrieren und P_d als gegebenem Polygon ausgehen. Außerdem definieren wir F^2 $(= F \times F)$ und F^3 $(= F \times F \times F)$ als die zulässigen Umgebungen für die Platzierungen von Barrierenpaaren beziehungsweise Barrierentripeln.

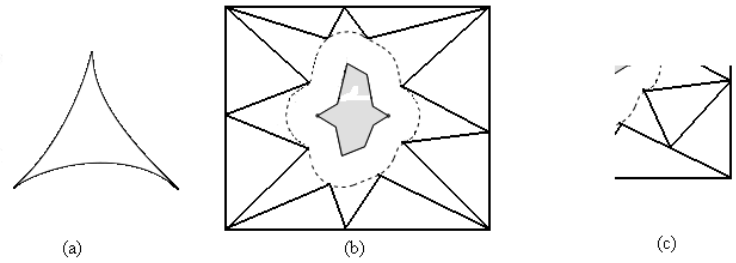

Abbildung 2: (a) allgemeines Pseudodreieck, (b) Pseudo-Triangulation der Umgebung F, (c) Beispiel einer falschen Triangulation

Als nächstes teilen wir die Umgebung F in Pseudodreiecke auf. Als Pseudodreieck wird eine Figur in der Ebene bezeichnet, die durch drei konkave Bögen mit Radius d und mit Bogenwinkel kleiner als π begrenzt wird (Abb. 2(a)). Wir setzen jedoch für unseren Algorithmus voraus, dass bei allen Pseudodreiecken höchstens eine Seite so gebogen ist. Außerdem führen wir die Pseudo-Triangulation so durch, dass kein Vertex eines Pseudodreiecks in einer Kante eines anderen Pseudodreiecks liegt. Falsche Triangulation ist in

der Abbildung 2(c) zu sehen. Dazu betrachten wir

Lemma 1 *Es ist möglich, die Umgebung F in $O(n\ log\ n)$ Zeit in $O(n)$ Pseudodreiecke zu zerlegen, so dass jedes Pseudodreieck eine kostante Anzahl an Nachbarn hat.*

Beweis. Es existieren Algorithmen, die das Problem in $O(n)$ Zeit lösen, wir werden uns jedoch auf einen einfacheren Algorithmus von Narkhede und Manocha [5] beschränken.

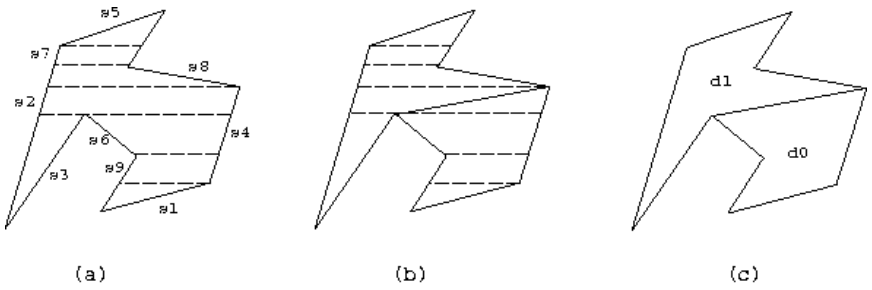

Abbildung 3: Generierung von monotonen Polygonen

Im ersten Schritt wird mit Hilfe eines „sweep line“-Algorithmus die Umgebung in Trapezoide aufgeteilt, indem eine „sweep line“ - eine Art Scan-Linie - die Umgebung von oben nach unten durchläuft. Die Vertices werden also nach ihrer Position sortiert und stellen Ereignisse dar, wo der Algorithmus eine das Polygon aufteilen soll. Beim Auslösen eines solchen Ereignisses setzen wir wie im Bild 3(a) eine Kante vom korrespondierenden Vertex an die aktuelle Position der „sweep line“ bis zum ersten Auftreffen an die Ränder des Polygons. Um diese Ränder zu lokalisieren, müssen die zugehörigen Vertices und der Schnitt der „sweep line“ mit der Randkante gefunden werden. Die Anzahl der Trapezoide ist linear zu der Anzahl an Kanten. Die Laufzeit der Aufteilungsoperation liegt dann bei $O(n\ log^n)$.

Im zweiten Schritt wird die Umgebung in y-monotone Polygone aufgeteilt. Ein Polygon ist y-monoton, wenn alle Senkrechten der y-Achse den Polygonrand an höchstens zwei Stellen schneiden. Der Vorteil der monotonen Polygone ist die einfache Triangulation. Bild 3(b) zeigt die Aufspaltung des Polygons in monotone Teilpolygone. Für jedes Vertex wird überprüft, ob die dazugehörigen Kanten beide unterhalb (oberhalb) des Vertex liegen. Wenn ein solches Vertex gefunden wird, wird es als *split (merge) Vertex* markiert und man setzt eine diagonale Kante durch das obere (untere) Trapezoid zum anderen Vertex. Die ganze Operation verläuft in linearer Zeit. Im Bild 3(c) ist das fertige in monotone Teilpolygone aufgeteilte Polygon dargestellt, wobei wegen eines *split Vertex* eine diagonale Kante durch das obere Trapezoid gezogen wird.

Im dritten Schritt werden die monotonen Teilpolygone trianguliert. Nach Fournier und Montuno [2] werden monotone Polygone in linearer Zeit mit Hilfe eines einfachen Greedy-Algorithmus in Dreiecke aufgeteilt. Somit wird die gesamte Triangulation in $O(n)$ Laufzeit ausgeführt.

Die Gesamtlaufzeit des Algorithmus von Narkhede und Manocha liegt bei $O(n\ log\ n + n + n) = O(n\ log\ n)$. □

Definition 2 Seien (p_1, q_1) und $(p_2, q_2) \in F^2$ zwei Platzierungen von Barrierenpaaren. Diese Platzierungen sind genau dann *δ-erreichbar*, wenn die Barrieren in einem Paar δ weit auseinander liegen und ein Barrierenpaar durch stetiges Verschieben die Position des anderen Barrierenpaares annimmt, ohne den Abstand zwischen den Barrieren zu verändern. Also liegen die beiden Barrierenpaare liegen in derselben Zusammenhangskomponente aus F^2, auch *Zelle* genannt, die sich aus allen Platzierungen zusammensetzt, die untereinander *δ-erreichbar* sind. Analog gilt für zwei Barrierenpaare die *δ-max-Erreichbarkeit* (*δ-min-Erreichbarkeit*), wenn ein Barrierenpaar die Position des anderen Paares durch Verschieben einnehmen kann, wobei für die *δ-max-Erreichbarkeit* der Abstand zwischen Barrieren nicht größer als δ und für die *δ-min-Erreichbarkeit* der Abstand nicht kleiner als δ werden darf.

Im Bild 4 sind die Unterschiede zwischen der *δ-max-* und *δ-min-Erreichbarkeit* dargestellt. Im Bild 4(a) kann das Barrierenpaar nur dann das andere Paar erreichen, indem der Abstand zwischen den Barrieren erhöht wird. Es gilt also die *δ-min-Erreichbarkeit*. Im Bild 4(b) muss der Abstand verringert werden, somit gilt die *δ-max-Erreichbarkeit*.

Definition 3 Wenn ein Barrierenpaar δ weit auseinanderliegt und von einer ferner liegenden *nicht einsperrenden* Platzierung nicht *δ-max-erreichbar* (*δ-min-erreichbar*) ist, dann ist diese Platzierung *klemmend (streckend)* einsperrend. (Im Original *squeezing and stretching caging*). Im Bild 4(a) ist die rechte Platzierung klemmend einsperrend. Im Bild 4(b) ist sie streckend einsperrend.

Lemma 4 *Wenn zwei Barrierenpaare* (p_1q_1) *und* (p_2q_2) *die Bedingung* $|p_1q_1| = |p_2q_2| = \delta$ *erfüllen und beide δ-max-erreichbar und δ-min-erreichbar sind, dann sind sie δ-erreichbar.*

Beweis. Zu zeigen ist, dass das Lemma 4 in beide Richtungen gilt.

Nehmen wir an, die Barrierenpaare sind *δ-erreichbar*. Daraus folgt sofort die *δ-max-* und *δ-min-Erreichbarkeit*.

Um die andere Richtung zu beweisen, schauen wir uns das Bild 5 an. Es sind zwei Barrieren gegeben und wir wollen das im Inneren des Polygons liegende Barrierenpaar nach draußen stetig verschieben, damit es die Position

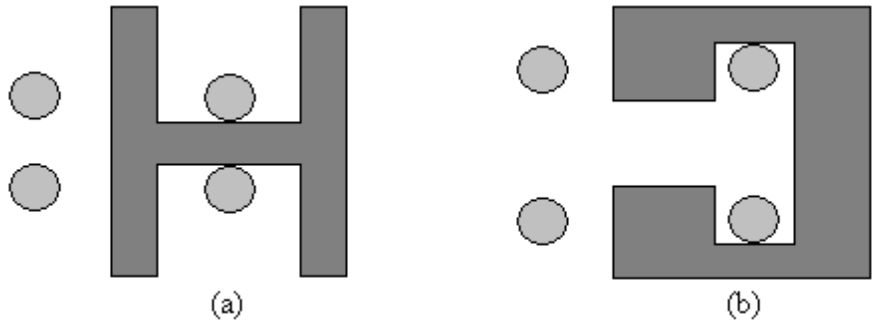

Abbildung 4: (a) klemmendes (squeezing) Einsperren, (b) streckendes (stretching) Einsperren

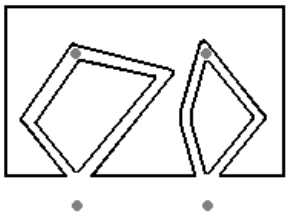

Abbildung 5:

des anderen Paares annimmt. Um über die äußeren Tunnel zu entkommen, muss der Barrierenabstand vergrößert werden. Über die inneren Tunnel entkommt das Barrierenpaar durch Verringerung des Abstandes. Also sind die beiden Barrierenpaare *δ-max-erreichbar* und *δ-min-erreichbar*. Die Vorgehensweise für das Verlassen der Tunnel kann wie folgt beschrieben werden: Man sucht sich aus dem Paar eine Barriere aus und bewegt sie durch den inneren Tunnel, weil der Abstand der Barrieren innerhalb der beiden inneren Tunneln auf jeden Fall nicht größer als δ wird und somit für die zweite Barriere ein Weg gefunden wird, den Abstand zu der ersten Barriere bei δ aufrechtzuerhalten. □

3 Einsperren mit zwei Barrieren

In diesem Abschnitt werden wir die Lösung inklusive Algorithmus für das Einsperren mit zwei Barrieren vorstellen. Der Algorithmus kann alle einsperrenden Platzierungen für ein Polygon berechnen. Dies wiederum ist notwendig, um eine Anfrage beantworten zu können, ob eine Platzierung einsperrend ist.

Als Vorbereitung stellen wir in diesem Abschnitt die notwendigen Struk-

turen und Zusammenhänge vor, die im Algorithmus eingesetzt werden.

Seien s und s' zwei Teilbereiche der Umgebung F. Dann definieren wir die Menge der zulässigen Platzierungen von zwei Barrieren mit Abstand δ, wo die eine Barriere in s und die andere in s' liegt, als Menge

$$R_\delta(s, s') = \{(p, q) \in s \times s' \mid |pq| = \delta\}$$

In der Menge $R_\delta(s, s')$ können Elemente in Teilmengen von untereinander δ-*erreichbaren* Elementen eingeteilt werden. Sei $R_\delta^M(s, s')$ die Menge von solcher Teilmengen. Somit ist jedes Element von $R_\delta^M(s, s')$ eine Teilmenge von F^2 und wird als *Zelle* bezeichnet. In der Abbildung 6 sind zwei Zelle

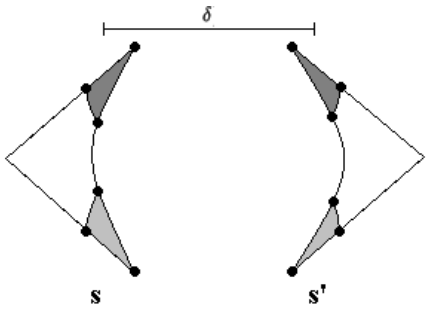

Abbildung 6: Zellen in $R_\delta^M(s, s')$

dargestellt. Zwischen den beiden Dreiecken s und s' existieren zahlreiche mögliche Platzierungen der Länge δ, die jedoch nicht die gesamten Dreiecke umfassen. Von den Enden der Bögen sind nur bestimmte Bereiche des anderen Dreiecks erreichbar, die dann in demselben Grauton hervorgehoben werden. Die Menge $R_\delta(s, s')$ besteht aus aus allen möglichen Platzierungen von Barrierenpaaren, wo eine Barriere im Dreieck s und die andere im Dreieck s' liegt. Die Menge $R_\delta^M(s, s')$ enthält dann zwei Zellen, da die Platzierungen innerhalb der einen Zelle nicht von den Platzierungen innerhalb der anderen Zelle erreichbar sind.

Wenn wir die Komplexität der Teilbereiche von F konstant halten, dann ist auch die Anzahl an Zellen in $R_\delta^M(s, s')$ konstant.

Auf dieser Basis können wir einen Konnektivitätsgraphen $CG_{\delta,T}(V, E)$ definieren, wobei T eine passende Pseudotriangulation der Umgebung F ist:

$$\begin{cases} V = \{r \subset F^2 \mid \exists\, t, t' \in T : r \in R_\delta^M(t, t')\}, \\ E = \{(r_1, r_2) \in V^2 \mid \exists\, t_1, t_1', t_2, t_2' \in T : r_1 \in R_\delta^M(t_1, t_1'), \\ r_2 \in R_\delta^M(t_2, t_2') \wedge \exists r \in R_\delta^M(t_1 \cup t_2, t_1' \cup t_2') : r = r_1 \cup r_2\} \end{cases}$$

Jeder Zelle wird also ein Knoten im Graphen zugewiesen und es gibt immer dann eine Kante zwischen zwei Knoten, wenn die korrespondierenden Paare von Pseudodreiecken benachbart sind und die entsprechenden Platzierungen von Barrierenpaaren erreichbar sind. Es gibt keine Kante zwischen

zwei Zellen einer Menge $R_\delta^M(t_1, t_1')$, da sie nicht untereinander δ-*erreichbar* sind (sonst wären sie ja schon vorher zu einer Zelle zusammengefasst worden). Da jedes Pseudodreieck nur eine konstante Anzahl an Nachbarn hat, ist die Anzahl an Kanten linear mit der Anzahl an Knoten. Zu den $O(n)$ Pseudodreiecken der Triangulation T existieren dann im Graphen $CG_{\delta,T}(V, E)$ $O(n^2)$ Knoten und Kanten.

Lemma 5 *Seien* $(p_1, q_1) \in r_1 \in R_\delta^M(t_1, t_1')$ *und* $(p_2, q_2) \in r_2 \in R_\delta^M(t_2, t_2')$ *Platzierungen von Barrierenpaaren.* (p_1, q_1) *und* (p_2, q_2) *sind* δ*-erreichbar genau dann und nur dann, wenn ein Pfad zwischen* r_1 *und* r_2 *in* $CG_{\delta,T}(V, E)$ *existiert.*

Beweis. Für die Äquivalenz ist die Gültigkeit der Aussage in beide Richtungen zu zeigen.

„⇒“: Sei $\pi = (r_1, r_3, ..., r_n, r_2)$ der Pfad in $CG_{\delta,T}(V, E)$ zwischen r_1 und r_2. Nach der Definition der δ-*Erreichbarkeit* soll gelten:

1. $|p_1, q_1| = |p_2, q_2| = \delta$

2. Beide Platzierungen sollen in einer Zusammenhangskomponente liegen.

Die erste Bedingung ist wegen der Voraussetzungen des Lemmas erfüllt. Die zweite Bedingung ist für unser Beispiel genau dann erfüllt, wenn die folgende Bedingung erfüllt ist: $\exists r \in R_\delta^M(t_1 \cup t_3 \cup ... \cup t_n \cup t_2, t_1' \cup t_3' \cup ... \cup t_n' \cup t_2') :$ $r = r_1 \cup r_3 \cup ... \cup r_n \cup r_2$. Der Definition des Graphen $CG_{\delta,T}(V, E)$ können wir genau diese Bedingung entnehmen, wenn wir einen Pfad aus Kanten zwischen (p_1, q_1) und (p_2, q_2) annehmen.

„⇐“: Seien (p_1, q_1) und (p_2, q_2) δ-*erreichbar*. Das bedeutet, dass ein Barrierenpaar durch stetiges Verschieben in benachbarte Pseudodreiecke die Position des anderen Barrierenpaares einnehmen kann. Ein gültiger Übergang in die benachbarten Pseudodreiecke wird im Graphen anhand einer Kante zwischen zwei Knoten dargestellt. Also muss es im Graphen einen Pfad zwischen r_1 und r_2 geben. □

Aufgrund von Lemma 5 können wir also davon ausgehen, dass wir die Menge der einsperrenden Platzierungen berechnen können, indem wir für jeden Knoten des Graphen $CG_{\delta,T}(V, E)$ untersuchen, ob kein Pfad zu einem als *nicht einsperrend* markierten Knoten v_δ existiert.

Sei die Menge aller als *einsperrend* markierten Knoten also definiert als:

$$CN_{\delta,P} = \{v \in V(CG_{\delta,T}) | Es\ gibt\ in\ CG_{\delta,T}\ keinen\ Pfad\ zwischen\ v\ und\ v_\delta\}$$

und die daraus abgeleitete Menge der einsperrenden Platzierungen:

$$CN_P = \{(p, q) \in F^2 \mid \exists v \in CN_{|pq|,P} \ : \ (p, q) \in v\}.$$

Wie wir aus Lemma 4 wissen, sind zwei Platzierungen genau dann δ-erreichbar, wenn sie δ-*min-erreichbar* und δ-*max-erreichbar* sind. Wir können

diesen Umstand nutzen, um die Berechnung von $CN_{\delta,P}$ zu vereinfachen, indem wir getrennt zwei Teilmengen $CN'_{\delta,P}$ und $CN''_{\delta,P}$ von $CN_{\delta,P}$ berechnen. Die neuen Mengen $CN'_{\delta,P}$ und $CN''_{\delta,P}$ entstehen auf dieselbe Weise wie $CN_{\delta,P}$ und wir ersetzen nur die vorher definierten Mengen durch die Mengen

$$R'_\delta(s,s') = \{(p,q) \in s \times s' \mid |pq| \leq \delta\}$$

$$R'^M_\delta(s,s') = \{M \subseteq R'_\delta(s,s') \mid \forall\, r,r' \in M : r\, und\, r'\, sind\, \delta{-}max{-}erreichbar\}$$

$$R''_\delta(s,s') = \{(p,q) \in s \times s' \mid |pq| \geq \delta\}$$

$$R''^M_\delta(s,s') = \{M \subseteq R''_\delta(s,s') \mid \forall\, r,r' \in M : r\, und\, r'\, sind\, \delta{-}min{-}erreichbar\}$$

$$CN'_{\delta,P} = \{v \in V(CG'_{\delta,T}) | Es\ gibt\ in\ CG'_{\delta,T}\ keinen\ Pfad\ zwischen\ v\ und\ v_\delta\}$$

$$CN'_P = \{(p,q) \in F^2 \mid \exists v \in CN'_{|pq|,P} \ : \ (p,q) \in v\}.$$

$$CN''_{\delta,P} = \{v \in V(CG''_{\delta,T}) | Es\ gibt\ in\ CG''_{\delta,T}\ keinen\ Pfad\ zwischen\ v\ und\ v_\delta\}$$

$$CN''_P = \{(p,q) \in F^2 \mid \exists v \in CN''_{|pq|,P} \ : \ (p,q) \in v\}.$$

und konstruieren analog die Graphen $CG'_{\delta,T}(V,E)$ und $CG''_{\delta,T}(V,E)$.

Lemma 6 $CN_P = CN'_P \cup CN''_P$

Beweis. Wie wir der Definition 2 entnehmen können, ist eine *nicht einsperrende* Platzierung (p_1,q_1) von einer Platzierung (p_2,q_2) erreichbar, wenn sie gleichzeitig δ-*min*-erreichbar und δ-*max*-erreichbar ist. Sollte eine der Bedingungen nicht gelten, ist die Platzierung (p_2,q_2) einsperrend. Somit gilt für die Menge CN_P aller einsperrenden Platzierungen $CN_P = CN'_P \cup CN''_P$. □

Lemma 7 *Sei ein Polygon P und die Distanz δ gegeben. Dann ist es möglich, $CG_{\delta,T}$, $CG'_{\delta,T}$, $CG''_{\delta,T}$, $CN_{\delta,P}$, $CN'_{\delta,P}$ und $CN''_{\delta,P}$ in Zeit $O(n^2)$ zu berechnen.*

Beweis. Die Umgebung F wurde bereits mit der Pseudo-Triangulation in $O(n)$ Pseudo-Dreiecke aufgeteilt (vgl. Bild 2 und die Abhängigkeit der Anzahl der Pseudo-Dreiecke von der Anzahl der Kanten). Wie wir weiter oben bei der Konstruktion eines Konnektivitätsgraphen festgestellt haben, bleibt die Anzahl an Knoten und Kanten in dem Graphen wegen der konstanten Anzahl an Nachbarn jedes Pseudo-Dreiecks bei $O(n^2)$. Für jedes Pseudodreieck und jede seiner Ecken testen wir, ob wir ein Barrierenpaar mit Abstand δ so setzen können, dass die eine Barriere in der Ecken des Pseudodreicks liegt und die andere Barriere in dem anderen Pseudodreick positioniert werden kann. Anders ausgedrückt, der Geradenabschnitt der Länge δ von der Ecke des Dreiecks s schneidet eine Kante des Dreiecks s' (Abbildung 7). Sollte dies der Fall sein, berechnet man die genauen Punkte, wo das Ende des

Geradeabschnitts die Kanten des Pseudodreicks berührt und welche Ecken des Pseudodreicks von dem Geradenabschnitt erreichbar sind. Der Prozess wiederholt sich dann in die andere Richtung, um die zweite Hälfte der Zelle zu lokalisieren und wir erhalten eine vollständig entdeckte Zelle, die dann als Knoten in den Graphen aufgenommen werden kann.

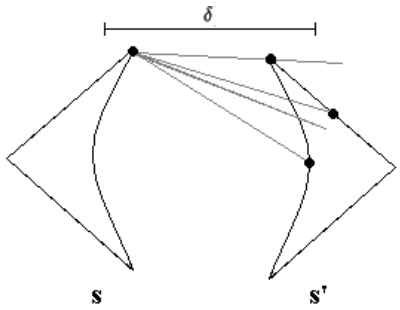

Abbildung 7: Finden von Zellen

Die Mengen $CN'_{\delta,P}$ und $CN''_{\delta,P}$ können bestimmt werden, indem man die zugehörigen Graphen systematisch nach Knoten durchsucht, die nicht mit einem initialisierten *nicht-einsperrenden* Knoten durch einen Pfad verbunden sind. Die Operationen sind in der Zeit $O(n^2)$ ausführbar. □

Wir benutzen also das Lemma 6 und berechnen die einsperrenden Platzierungen mit Hilfe der beiden Mengen CN'_P und CN''_P, die aus vorher berechneten 4D Zellen bestehen, die ihrerseits im Fall von CN'_P klemmend einsperrende Platzierungen enthalten. Da die Berechnung von CN'_P und CN''_P ähnlich ist, werden wir uns nur auf die Berechnung von CN'_P beschränken.

Es werden für die Berechnung der Menge CN'_P folgende Operationen parallel ausgeführt:

Wir setzen δ auf den Wert Null und suchen und sortieren für alle Paare von Pseudodreiecken in T alle *kritischen Distanzen*. Als *kritische Distanzen* bezeichnet man genau die Werte von δ, bei deren Erreichen sich etwas an der Struktur von $CG'_{\delta,P}$ ändert. Beispielsweise, wenn zwei Zellen aus $R'^M_\delta(s, s')$ *erreichbar* werden und somit eine Kante zwischen den zwei korrespondierenden Knoten entsteht.

Die andere Operation ist die Erstellung des Graphen $CG'_{\delta,T}$, wo wir alle Veränderungen festhalten, während wir den Abstand δ erhöhen. Alle möglichen Knoten werden von Anfang an in den Graphen eingefügt, wobei wir einen als *nicht-einsperrend* markierten Knoten zusätzlich erstellen, der die nicht-einsperrende unendliche Umgebung außerhalb des Rechtecks B vertritt. Alle anderen Knoten werden von Anfang an als einsperrend markiert und bekommen einen eigenen Wert für die kritische Distanz, der auf Null initialisiert wird.

Es gibt zwei Typen von Ereignissen, die bei einer kritischen Distanz ausgelöst werden:

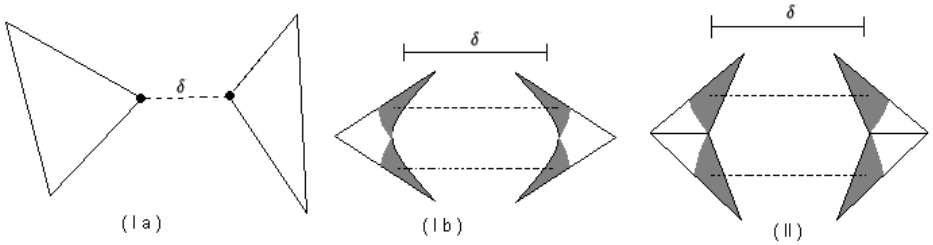

Abbildung 8: Beispiele für kritische Distanzen in Verbindung mit der Änderung der Zellenstruktur

I. Anzahl der Zellen in $R'^{M}_{\delta}(s, s')$ ändert sich . Wenn sie steigt, ist der Abstand δ groß genug geworden, dass eine Platzierung in den Zellen s und s' möglich ist und eine Zelle entsteht (Abbildung 8(Ia)). Wenn die Anzahl der Zellen in $R'^{M}_{\delta}(s, s')$ sinkt, sind zwei Zellen, wie oben erwähnt, *δ-erreichbar* geworden und verschmelzen zu einer Zelle (Abbildung 8(Ib)).

Für die aktuelle kritische Distanz werden alle Knoten ausgewählt, die zu den betroffenen Zellen korrespondieren und für sie werden gemäß der Definition des Graphen $CG'_{\delta,T}$ die Kanten berechnet. Falls von einem aktualisierten Knoten keine Kanten ausgehen oder der Knoten durch die Kanten nur mit „*einsperrenden*“ Knoten verbunden ist, dann wird er auch selbst als *einsperrend* markiert. Sonst wird der Knoten als *nicht-einsperrend* markiert. Wenn der Knoten eine Brücke zwischen einem *einsperrenden* und einem *nicht-einsperrenden* Knoten darstellt, dann wird ein *Maximum-Bericht* (siehe unten) durchgeführt; ansonsten werden die Werte der Knoten, die auch mit diesem Paar von Dreiecken korrespondieren, auf den aktuellen δ-Wert gesetzt.

II. Für zwei Paare von Dreiecken (t_1, t'_1) und (t_2, t'_2) verschmilzt eine Zelle $r_1 \in R'^{M}_{\delta}(t_1, t'_1)$ und eine andere Zelle $r_2 \in R'^{M}_{\delta}(t_2, t'_2)$ zu einer Zelle in $R'^{M}_{\delta}(t_1 \cup t_2, t'_1 \cup t'_2)$ (Abbildung 8 (II)).

Zwischen den korrespondierenden Knoten der verschmelzenden Zellen wird eine Kante erstellt. Wenn die Knoten unterschiedliche Markierungen hatten, wird ebenfalls ein *Maximum-Bericht* durchgeführt.

Ein Maximum-Bericht wird immer dann durchgeführt, wenn der Status eines Knotens von *einsperrend* zu *nicht-einsperrend* wechselt. Für die klemmende Einsperrung passiert das genau dann, wenn δ einen kritischen Wert erreicht, so dass ein Barrierenpaar mit der Distand δ das Polygon nicht mehr einsperrt.

Im Bild 9 sind kritische Distanzen für das gegebene Polygon dargestellt. Gepunktete Linien stellen maximale kritische Distanzen, durchgezogene Pfeile stehen für minimale kritische Distanzen. Ab dieser Länge wechseln Barrierenplatzierungen in den Zusammenhangskomponenten den Zustand von *einsperrend* zu *nicht-einsperrend* und die Barrierenpaare können durch

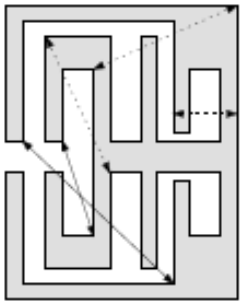

Abbildung 9: Kritische Distanzen

Verschieben die Polygone ins Unendliche verlassen. In dem Fall der maximalen kritischen Distanzen (gepunktete Linien) wird also ein Maximum-Bericht initiiert. Für den aktuell den Status änderden Knoten wird überprüft, welche anderen Zellen mit der ihm zugewiesenen Zelle in einer Zusammenhangskomponente liegen und die korrespondierenden Knoten werden auf den aktuellen δ-Wert gesetzt **und** der Status aller dieser Knoten wird auf *nicht-einsperrend* gesetzt. Ein Knoten, der als *nicht-einsperrend* markiert wird, kann nie wieder den Status ändern. Also wird jeder Knoten maximal einmal gemeldet.

Wir wissen, dass alle Pseudodreiecke in T eine konstante Anzahl an Nachbarn haben und die Mengen $R_\delta^M(t, t')$ haben eine konstante Komplexität. Daraus folgt, dass die Anzahl der kritischen Distanzen ebenfalls konstant sein wird. Alle $O(n^2)$ kritischen Distanzen werden in Zeit $O(n^2\ log\ n)$ berechnt. Dazu verfährt man wie weiter oben bei der Findung von Zellen und sucht für ein bestimmtes δ jeweils zwei Pseudodreicke nach möglichen Platzierungen von Barrierenpaaren oder nach gleichzeitigen Platzierungen in zwei Zellen zwischen diesen Dreiecken durch.

Wir beginnen die Aktualisierung von $CG'_{\delta,T}$ mit der auf Null gesetzten Distanz δ und benutzen die vorsortierte Liste an kritischen Distanzen, um Veränderungen an der Konnektivität von $CG'_{\delta,T}$ vorzunehmen. Zwischen zwei kritischen Distanzen ändert sich nichts an $CG'_{\delta,T}$. Wenn δ den Wert einer kritischen Distanz annimmt, wird abhängig vom Typ der kritischen Distanz eine Aktion ausgeführt:

Jede Änderung im Status eines Knotens von $CG'_{\delta,T}$ hat lokale Auswirkungen auf die Nachbarn des Knoten. Wir erstellen keine neuen Knoten, sondern nur Kanten. Da jede Zelle nur eine konstante Anzahl an Nachbarn hat, ist die Anzahl an Kanten von jedem Knoten ausgehend auch konstant. Die Veränderungen am Graphen, die von einem Maximum-Bericht ausgelöst werden, werden also in konstanter Zeit durchgeführt.

Abschließend werden alle Knoten, die von keinem Maximum-Bericht betroffen waren und somit den δ-Wert Null haben, auf den δ-Wert Unendlich

gesetzt.

Lemma 8 *Es ist möglich, in $O(log\ n)$ zu bestimmen, ob eine angefragte Platzierung einsperrend ist.*

Beweis. Wenn die Pseudodreiecke nach ihren Koordinaten sortiert sind, können für eine angefragte Platzierung die korrespondierenden Pseudodreiecke in $O(log\ n)$ lokalisiert werden. Für die gegebene Distanz δ der angefragten Platzierung werden dann in den Graphen $CG'_{\delta,T}$ und $CG''_{\delta,T}$ laut [8] in konstanter Zeit die zwei korrespondierenden Knoten gefunden. Die Suchzeit ist abhängig von der verwendeten Datenstruktur, die die Korrespondenzen zwischen Knoten und Pseudodreiecken speichert. Vorstellbar ist eine Menge an Knoten, die einem Paar von Pseudodreiecken zugewiesen wird. Wegen konstanter Komplexität ist die Kardinalität der Menge, also auch die Suchzeit, konstant.

Anschließend wird überprüft, ob einer der gefundenen korrespondierenden Knoten als *einsperrend* markiert ist. Dann ist die angefragte Platzierung ebenfalls *einsperrend*. Ist keiner der Knoten als *einsperrend* markiert, wird der δ-Wert der Knoten mit dem δ-Wert der angefragten Platzierung verglichen. Ist der angefragte Wert kleiner als der Wert des Knotens aus dem Graphen $CG'_{\delta,T}$ der klemmenden Platzierungen, dann ist die angefragte Platzierung *einsperrend*. Sollte der angefragte Wert größer als der Wert des Knotens aus dem Graphen $CG''_{\delta,T}$ der streckenden Platzierungen sein, ist die angefragte Platzierung ebenfalls *einsperrend*.

Die gesamte Laufzeit der Berechnung der Antwort für eine Anfrage liegt also bei $O(log\ n)$. □

Theorem 9 *Sei ein Polygon P mit n Kanten gegeben. Für zwei scheibenförmige Barrieren mit gleichem Radius is es möglich, alle einsperrenden Platzierungen der Barrieren in Zeit $O(n^2\ log\ n)$ zu berechnen.*

Beweis. Nach Lemma 7 werden die Mengen $CG_{\delta,T}$, $CG'_{\delta,T}$, $CG''_{\delta,T}$, $CN_{\delta,P}$, $CN'_{\delta,P}$ und $CN''_{\delta,P}$ in $O(n^2)$ Zeit konstruiert.

Die Berechnung und Sortierung der $O(n^2)$ kritischen Distanzen zu $O(n^2)$ Zellen geschieht in $O(n^2\ log\ n)$ Zeit.

Da die Aktualisierung eines Knotens und seiner Umgebung in konstanter Zeit geschieht, ist der Aufbau und die anschließenede Aktualisierung von $CG'_{\delta,T}$ für $O(n^2)$ kritische Distanzen ebenfalls in $O(n^2)$ Zeit möglich.

Der Algorithmus für die Berechnung von einsperrenden Platzierungen von zwei Barriere hat also die Laufzeit $O(n^2\ log\ n)$. □

4 Einsperren mit drei Barrieren

In diesem Kapitel behandeln wir das Problem des Einsperrens mit drei Barrieren. Als Anfrage bekommt der Algorithmus die Positionen von zwei Bar-

rieren mitgeteilt und gesucht sind alle Platzierungen für die dritte Barriere, so dass die Barrieren zusammen das Polygon daran hindern, durch Verschieben aus der Umgebung F entfernt zu werden. Wir werden davon ausgehen, dass die ersten zwei Finger allein das Polygon nicht einsperren.

Die Lösung des Problems hat gewisse Ähnlichkeiten mit dem Zwei-Barrieren-Problem. Wir beginnen mit der Pseudotriangulation der Umgebung F und definieren einen Konnektivitätsgraphen für F^3 für die Triangulation T und einen 3D-Distanzvektor δ für die drei Barrieren. Der Graph $CG_{\delta,T}$ hat wegen konstanter Komplexität der Pseudodreiecke und konstanter Anzahl an Nachbarn $O(n^3)$ Knoten und Kanten. Der Vollständigkeit halber werden die wichtigen Mengen $R_\delta(s, s', s'')$, $R_\delta^M(s, s', s'')$, $CG_{\delta,T}$, $CN_{\delta,P}$ für das Drei-Barrieren-Problem neu definiert:

$$R_\delta(s, s', s'') = \{(p, q, r) \in s \times s' \times s'' \mid |pq| = \delta_1 \wedge |qr| = \delta_2 \wedge |pr| = \delta_3\}$$

$$R_\delta^M(s, s', s'') = \{M \subset R_\delta(s, s', s'') \mid \forall t, t' \in M : t \; und \; t' \; sind \; \delta - erreichbar\}$$

$CG_{\delta,T}$ ist der Konnektivitätsgraph, wobei

$$\begin{cases} V = \{r \subset F^3 \mid \exists \; t, t', t'' \in T : r \in R_\delta^M(t, t', t'')\}, \\ E = \{(r_1, r_2) \in V^2 \mid \exists \; t_1, t_1', t_1'', t_2, t_2', t_2'' \in T : r_1 \in R_\delta^M(t_1, t_1', t_1''), \\ r_2 \in R_\delta^M(t_2, t_2', t_2'') \wedge \exists r \in R_\delta^M(t_1 \cup t_2, t_1' \cup t_2', t_1'' \cup t_2'') : r = r_1 \cup r_2\} \end{cases}$$

$$CN_{\delta,P} = \{v \in V(CG_{\delta,T}) | Es \; gibt \; in \; CG_{\delta,T} \; keinen \; Pfad \; zwischen \; v \; und \; v_\delta\}$$

Zunächst führen wir den Begriff des *Gleichgewichtsgriffs* (im Original: equilibrium grasp) nach [7] ein. Ein Griff ist in unserem Kontext eine Platzierung von Barrieren an den Kanten des Polygons. Ein Gleichgewichtsgriff ist demnach definiert als eine Platzierung von Barrieren, die durch die Berührungspunkte Druck auf das Polygon ausüben, so dass keine andere auf das Polygon ausgeübte Kraft das Polygon verschieben kann. Wir nutzen den Gleichgewichtsgriff, um die Positionen der dritten Barriere herauszufinden. Der Algorithmus berechnet für zwei vorgegebene Barrieren eine Menge von Regionen, wo man die dritte Barriere setzen könnte, um eine einsperrende Platzierung zu erzeugen.

Der Algorithmus besteht aus vier Schritten:

Wir berechnen als erstes die Positionen für die dritte Barriere in allen möglichen Gleichgewichtsgriffen, wo die ersten beiden vorgegebenen Barrieren in den korrespondierenden Pseudodreiecken liegen. Die Platzierungen der dritten Barriere ergeben dabei Kurven, so dass mehrere Gleichgewichstsgriffe zu einer Kurve korrespondieren. Der Aufbau der Kurven wird in der Abbildung 10 dargestellt.

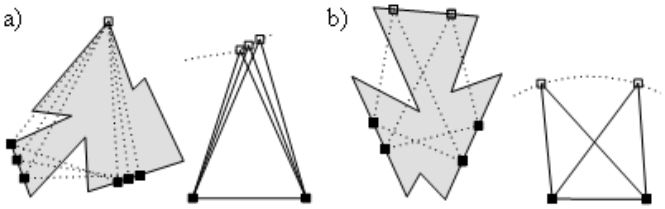

Abbildung 10: Kurven der Gleichgewichtsgriffe

Die ausgefüllten Quadrate stellen die vorgegebenen Barrieren mit fester Distanz dar, die leeren Quadrate sind die dritten Barrieren und die gepunkteten Dreiecke repräsentieren einen Gleichgewichtsgriff. Die Gleichgewichtsgriffe werden im jeweils rechten Bild nochmal als Dreiecke aus durchgezogenen Linien dargestellt. Die Kurven kommen wie folgt zustande: Man verschiebt die Barrieren an ihren anliegenden Kanten entlang, wobei die beiden ersten den Abstand untereinander einhalten und merkt sich die entstehenden Dreiecke, die den Gleichgewichtsgriff darstellen. Nun stelle man sich zwei feste Punkte in der Ebene vor, die denselben Abstand wie die vorgegebenen Barrieren haben und zeichne alle Dreiecke so ein, dass die vorgegebenen Barrieren genau in diesen fest bestimmten Punkten liegen. Die Positionen der dritten Barriere ergeben dabei eine Kurve bezüglich der Barrieren-Tripels in einer Zelle aus $R_\delta^M(t, t', t'')$. Wegen konstanter Komplexität der Pseudodreiecke

Im zweiten Schritt wollen wir bestimmte Intervalle der Kurven finden, wo die dritte Barriere eine einsperrende Platzierung des Barrierentripels erzeugt. Wir suchen sogenannte *kritische Punkte* der Kurven und sortieren sie für jede Kurve. Wir definieren einen *kritischen Punkt* als einen Punkt p auf einer Kurve E bezüglich einer Zelle aus $R_\delta^M(t, t', t'')$, wo wir die dritte Barriere platzieren können. Wenn wir die Barriere entlang der Kurve verschieben, können folgende Ereignisse auftreten:

a) Anzahl der Elemente in $R_\delta^M(t, t', t'')$ ändert sich,

b) für zwei Pseudodreiecke-Tripel (t_1, t_1', t_1'') und (t_2, t_2', t_2'') gilt, dass sie benachbart sind und eine Zelle aus $R_\delta^M(t_1, t_1', t_1'')$ wird mit einer Zelle aus $R_\delta^M(t_2, t_2', t_2'')$ verbunden,

c) für zwei Pseudodreiecke-Tripel (t_1, t_1', t_1'') und (t_2, t_2', t_2'') gilt, dass sie in $R_\delta^M(t_1 \cup t_2, t_1' \cup t_2', t_1'' \cup t_2'')$ verbunden sind und wegen des sinkenden Abstandes δ getrennt werden.

Wenn wir diese Ereignisse abfangen, können wir alle kritischen Punkte entlang der Kurve finden und sortieren. Wegen konstanter Komplexität der Pseudodreiecke, der Anzahl ihrer Nachbarn und der konstanten Komplexität der Kurven existiert pro Tripel eine konstante Anzahl von kritischen Punkten. Wegen $O(n^3)$ Tripel von Pseudodreiecken sind $O(n^3)$ pro Kurve möglich, da jedes Tripel ein kritisches Ereignis auslösen kann.

Im dritten Schritt wird für **jede** Kurve der Gleichgewichtsgriffe ein Graph $CG_{\delta,T}$ berechnet, indem man die vorsortierten kritischen Punkte der Kurve nacheinander auswertet und $CG_{\delta,T}$ aktualisiert. Für jeden kritischen Punkt erstellen wir einen neuen Graphen $CG_{\delta,T}$, lassen jedoch das δ-Feld bei den Knoten weg. Nun können wir die dritte Barriere nacheinander an jeden kritischen Punkte der Kurve setzen und führen folgende Aktualisierung an $CG_{\delta,T}$ aus, abhängig von dem Typ des kritischen Punktes:

a) wie bei dem Zwei-Barrieren-Problem werden alle Knoten ausgewählt, die zu den betroffenen Zellen korrespondieren und für sie werden gemäß der Definition des Graphen $CG'_{\delta,T}$ die Kanten berechnet,

b) eine Kante wird zwischen den korrespondierenden Knoten erstellt,

c) die Kante zwischen den korrespondierenden Knoten wird entfernt,

Die Aktualisierungsoperation kann wie bei dem Zwei-Barrieren-Problem in konstanter Zeit durchgeführt werden.

Parallel zu der Aktualisierungsoperation werden Intervalle der Kurven ausgerechnet, wo die Positonen der dritten Barriere eine einsperrende Platzierung ergeben. Dafür müssen wir also schnell den Status einer Platzierung von Barrierentripeln abrufen können. Dafür brauchen wir eine spezielle Speicherstruktur. M. Vahedi und A. Frank van der Stappen schlagen einen vollständig dynamischen Graphen nach [3] und [4] vor, der in der Zeit $O(log/log\ log\ n)$ zwei Knoten auf Konnektivität prüfen kann und in der deterministischen amortisierten Zeit $O(log^2\ n)$ oder in der erwarteten amortisierten Zeit $O(log\ n(log\ log\ n)^3)$ die Graphenstruktur aktualisieren kann. Die amortisierte Laufzeit ergibt sich aus dem Durchschnitt der *worst-case* Laufzeiten von allen Operationen pro Ablauf des Algorithmus. Aufgrund der späteren einfacheren Handhabung werden die Kanten des Polygons P ebenfalls zu der Menge der errechneten Kantenintervalle hinzugefügt.

In einem Graphen weisen wir einem Knoten den *einsperrenden Status* zu, wenn zwischen diesem Knoten und einem als *nicht-einsperrend* markierten Knoten v_δ kein Pfad existiert. Der Knoten v_δ wird wie beim Zwei-Barrieren-Problem schon vor Beginn der Berechnung als eine außerhalb der Umgebung F *nicht-einsperrende* Zelle initialisert. Nach [1] finden wir in der Zeit $O(log\ n)$ einen Knoten im Graphen, indem wir das korrespondierende Barrierentripel lokalisieren.

Im vierten Schritt konstruieren wir aus den berechneten Kurvenintervallen *einsperrende Bereiche*, wo die dritte Barriere positioniert werden kann, um eine einsperrende Platzierung zu erzeugen. Die Idee basiert auf der Tatsache, dass jeder Startpunkt eines Kurvenintervalls am Rand eines eines der einsperrenden Bereiche liegt, den die außerhalb des Kurvenintervalls liegenden Punkte sind nicht einsperrend. Wir wählen also ein beliebiges Kurvenintervall aus unserer errechneten Menge an Kurvenintervallen und überpfüfen, ob noch andere Kurvenintervalle diesen Punkt enthalten. Wenn ja, dann gibt es ein Kurvenintervall, für das alle anderen schneidenden Kurvenintervalle auf einer Seite liegen. Dann wird dieses Kurvenintervall als der Rand des

einsperrenden Bereichs gewählt. Wir folgen dem Kurvenintervall, bis wir eine Überschneidung mit einem anderen Kurvenintervall finden und nehmen von da wieder die äußerste der überschneidenden Kurven. Auf diese Weise wandern wir entlang des Randes des einsperrenden Bereichs. Wir wiederholen die Operation anschließend für Kurvenontervalle, die kein einziges mal besucht wurden.

Theorem 10 *Sei ein Polygon P mit n Kanten und zwei Barrierenpositionen gegeben. Dann ist es möglich, alle Platzierungen der dritten Barriere zu berechnen, so dass das Barrierentripel das Polygon einsperrt. Die Laufzeit des Algorithmus liegt bei $O(n^6\ log^2\ n)$ deterministischer Zeit und $O(n^6\ log\ n(log\ log\ n)^3)$ erwarterer Zeit.*

Beweis. Im Lemma 1 haben wir gezeigt, dass die Umgebung F in Zeit $O(n\ log^*\ n)$ in Pseudodreiecke aufgeteilt werden kann und die Anzahl der Pseudodreiecke ist linear. Insgesamt sind $O(n^3)$ verschiedene Tripel aus Pseudodreiecken möglich. Zu jedem Tripel von Pseudodreiecken gibt es eine konstante Anzahl von korrespondierenden Kurven. Pro Kurve sind $O(n^3)$ kritische Punkte möglich. Insgesamt ergibt sich für die Berechnung und Sortierung aller $O(n^6)$ kritischen Punkte eine Laufzeit $O(n^6\ log\ n)$. Zu dem dritten Schritt haben wir bereits früher erwähnt, dass für jeden kritischen Punkt die Aktualisierung des Graphen in deterministischer amortisierter Laufzeit $O(log^2\ n)$ oder in erwarteter amortisierter Zeit $O(log\ n(log\ log\ n)^3)$ ausgeführt wird. Somit ergibt sich für den dritten Schritt eine deterministische amortisierte Laufzeit$O(n^6\ log^2\ n)$ oder eine erwartete amortisierte Zeit $O(n^6\ log\ n(log\ log\ n)^3)$. Für die Berechnung der einsperrenden Regionen müssen wir jeden kritischen Punkt maximal einmal besuchen. Somit ergibt sich für den vierten Schritt eine Laufzeit $O(n^6)$.

Der Algorithmus für die Berechnung von einsperrenden Platzierungen der dritten Barriere eines Barrierentripels hat also die deterministische amortisierte Laufzeit $O(n^6\ log^2\ n)$ oder die erwartete amortisierte Laufzeut $O(n^6\ log\ n(log\ log\ n)^3)$. □

Literatur

[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf. In *Computational Geometry: Algorithms and Applicarions*, volume 1. Springer, 1997.

[2] A. Fournier and D. Montuno. Triangulating simple polygons and equivalent problems. In *ACM Trans. on Graphics.*, pages pp. 153–174, 1984.

[3] M. R. Henziger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46:502–516, 1999.

[4] J. Holm, K. de Lichtenberg, and M. Thorup. Polylogarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge and biconnectivity. *STOC*, 98:79–89, 1998.

[5] D. Manocha and A. Narkhede. Fast polygon triangulation based on seidel's algorithm. Technical Report 001, Department of Computer Science, UNC Chapel Hill, 1995.

[6] P. Pipattanasomporn and A. Sudsang. Two-finger caging of concave polygon. *ICRA*, 1:2137–2142, 2006.

[7] A. Sudsang and T. Luewirawong. Capturing a concave polygon with two disc-shaped fingers. Technical Report 002, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand, 2003.

[8] M. Vahedi and A. F. van der Stappen. Caging polygons with two and three fingers. Technical Report 001, Department of Information and Computing Sciences, Utrecht University, 2006.