Gleichmäßige Flächenaufteilung von Polygonen

Sebastian Loder Jan Steffen Jendrny

Januar 2022

Gummibärchen (auch Gummibären bzw. Goldbären) sind Fruchtgummis in Form von etwa zwei Zentimeter großen, stilisierten Bären. Sie werden in unterschiedlichen Farben hergestellt und bestehen im Wesentlichen aus Zucker, Zuckersirup und einer erstarrten Gelatine-Mischung, die ihnen ihre gummiartige Konsistenz verleiht.

Stichwörter: Süßigkeiten, Gelatine, Bär, Lebensmittelfarbe.

1 Einleitung

Die Arbeit basiert auf der Veröffentlichung „Polygon Area Decomposition for Multiple-Robot Workspace Division“ von Susan Hert und Vladimir Lumelsky. In dieser Veröffentlichung wird ein konkretes Problem der Polygonzerlegung, das sogenannte *Problem der verankerten Flächenaufteilung* (eng. *anchored area partition problem*) beschrieben und mittels *sweepline*- und *divide-and-conquer*-Techniken gelöst. Die Lösung erfolgt zunächst für konvexe Polygone und wird anschließend auf nicht konvexe, nicht einfache Polygone erweitert.

Die Polygonzerlegung ist eines der zentralen Probleme in der algorithmischen Geometrie und hat viele Anwendungsfälle, wie beispielsweise in der Kartographie, Bildverarbeitung oder in der Computergrafik. In vielen Fällen wird die Polygonzerlegung benötigt, um aus einem beliebigen Polygon eine Menge aus Teilpolygonen mit bestimmten Eigenschaften zu berechnen. Als Beispiel einer vielfach verwendeten Polygonzerlegung kann die Triangulation genannt werden, bei welcher ein gegebenes Polygon in eine Menge von Dreiecken zerlegt wird. Für die so berechnete Menge von Dreiecken stehen dann effiziente Algorithmen zur Lösung von Problemen zur Verfügung. Anschließend können die Lösungen der Teilpolygone zu einer Lösung für das Ausgangspolygon zusammengefasst werden.

Bei dem hier vorgestellten Problem der verankerten Flächenaufteilung ist die Anforderung an die resultierenden Teilpolygone nicht durch eine bestimmte Geometrie (z. B. ein Dreieck), sondern durch die Lage und Fläche der Teilpolygone gegeben. Bezüglich der Lage besteht die Anforderung darin, dass ein gegebener Punkt (Standort genannt) auf dem resultierenden Polygon liegen muss. Jeder Standort weist als Eigenschaft eine Flächenanforderung auf, welche durch die Größe des Teilpolygons erfüllt werden soll. Die Flächenanforderung kann je Standort den gleichen Wert aufweisen, kann aber auch unter den Standorten variieren. Somit bezieht sich das hier beschriebene Problem sowohl auf eine gleichmäßige, als auch eine ungleichmäßige Flächenaufteilung. Das beschriebene Problem ist unter anderem durch die Flächenerkundung von Robotern motiviert:

Auf einem Polygon werden n Roboter auf Standorten S_i, $i = 1, ..., n$ positioniert, welche die Aufgabe erhalten, zusammen die gesamte Fläche des Polygons zu erkunden. Hierzu muss jede Position innerhalb des Polygons von einem der n Roboter abgefahren werden. Um die Arbeit unter den Robotern aufzuteilen, ist es sinnvoll, jedem Roboter einen Polygonteil zuzuweisen, der jeweils zu bearbeiten ist. Die Teilpolygone sollen sich nicht überlappen, um ein mehrfaches Überfahren zu vermeiden. Bei der Flächenaufteilung muss berücksichtigt werden, dass der Startpunkt eines jeden Roboters auf dem zugewiesenen Teilpolygon beziehungsweise in diesem liegt. Eine unterschiedliche Leistung der Roboter kann über die Flächenanforderung je Standort berücksichtigt werden.

Zur formalen Beschreibung des Problems sind als Eingangsdaten ein Polygon P sowie eine (nicht leere) Liste von Standorten S_P gegeben. Für jeden der n Standorte S_i , $i = 1, ..., n$ ist der benötigte Flächenanteil c_i, $i = 1, ..., n$ mit $0 < c_i < 1$ gegeben, sodass $\sum_{i=1}^{n} c_i = 1,0$ gilt. Das Polygon$P$ soll in n, nicht überlappende Polygone zerlegt werden, sodass jeder Standort S_i auf dem Polygon P_i mit Fläche c_i∗Area(P)liegt. Aus der Fläche des Polygons P kann für jeden der n Standorte die benötigte Fläche mit c_i∗Area(P) bestimmt werden.

2 Aufteilung eines einfachen, konvexen Polygons

2.1 Grundidee

Bei der nachfolgend beschriebenen Lösung des Problems wird das konvexe Eingabepolygon CP mithilfe von Liniensegmenten schrittweise zerlegt. Jedes Liniensegment L ist hierbei vom Startpunkt L_s zum Endpunkt L_e orientiert, wobei beide Punkte auf dem Rand von CP liegen. Wenn für L_s und L_e eine Position gefunden wurde, erfolgt eine Zerlegung in zwei Teilpolygone. Die bei jeder Teilung entstehen Teilpolygone erhalten entsprechend ihrer Lage zum Liniensegment L die Bezeichnungen P_L^r für das rechts und P_L^l für das links des Liniensegments liegenden Polygons. Die Liniensegmente (bzw. L_s und L_e) werden so positioniert, dass die Fläche von P_L^r der benötigten Fläche der auf dem Rand von P_L^r liegenden Standorte entspricht (P_L^l analog). Die Zerlegung wird für jedes Teilpolygon P_L^r und P_L^l rekursiv aufgerufen, bis nur noch 1-Standort Polygone vorliegen. Da aus einer Zerlegung eines konvexen Polygons mit einem Liniensegment immer zwei konvexe Polygone und insbesondere kein nicht konvexes Polygon resultiert, ist dieser Ansatz möglich.

2.2 Aufteilung eines einfachen, konvexen Polygons

Aus CP entstehende, konvexe Polygone werden mit CP_i notiert. Mit den genannten Überlegungen lässt sich ein rekursiver *divide-and-conquer* - Algorithmus zur Flächenaufteilung eines konvexen Polygons - basierend auf n Standorten - nun folgendermaßen skizzieren:

Bei jedem Aufruf von *ConvexDivide* (CP) wird zunächst geprüft, ob das übergebene Polygon nur noch einen Standort enthält. Falls ja, ist der Zielzustand für das übergebene Polygon erreicht und es ist keine weitere Flächenaufteilung erforderlich. Falls das Polygon mehrere Standorte enthält, erfolgt eine weitere Aufteilung des Polygons in zwei Teil-Polygone P_L^r und P_L^l , welche dann rekursiv mit *ConvexDivide* aufgerufen werden.

2.3 Positionierung der Schnittlinie

Aus vorangegangenem Kapitel bleibt offen, wie genau die Aufteilung eines konvexen Polygons CP in die Polygone P_L^r und P_L^l erfolgt, sodass anschließend $Area(P_L^r) = AreaRequired(S_1, ..., S_i)$ und $Area(P_L^l) = AreaRequired(S_{i+1}, ..., S_n)$ gilt. Konkret ist zu klären, wie Anfangs- und Endpunkt der Schnittlinien positioniert werden (siehe Listing XX, Zeile 4).
Initialisierung von L_s und L_e beim Aufruf von *ConvexDivide* :

- Der Startpunkt L_s der Linie L wird mit den Koordinaten des ersten Punkts der Liste $W()$ initialisiert, wobei dieser nach Definition ein Polygonpunkt (und kein Standort) ist. Es gilt daher $w_1 \in V()$.
- Der Endpunkt L_e wird mit den Koordinaten des ersten Standorts in $W()$ initialisiert und mit w_k notiert, wobei k der Index in $W()$ ist, bei dem der erste Standort liegt. Da die Standorte nachihrem Vorkommen auf dem Weg von v_1 nach v_l geordnet sind, ist bei einem konvexen Polygon sichergestellt, dass die Standorte $S_2, \ldots, Sn$ alle links der Linie L liegen.

Bei einer Zerlegung mit einer so initialisierten Linie würde $S(P_L^r) = S_1$ und $S(P_L^l) = S_2, ..., S_n$ gelten, wobei S_1 in einer Ecke von P_L^r liegen würde. Je nach Fläche von P_L^r und $AreaRequired(S_1)$ werden folgende Fälle unterschieden:

Fall 1: $Area(P_L^r) > AreaRequired(S_1)$
Nach der Initialisierung der Linie L wird festgestellt, dass die Fläche von P_L^r größer ist als die benötigte Fläche von S_1. In diesem Fall erfolgt eine Verkleinerung von $Area(P_L^r)$ unter Beibehaltung

von $S(P_L^r) = S_1$. Dies geschieht, indem L_e als Drehpunkt fungiert und L_s inkrementell gegen den Uhrzeigersinn entlang des Polygons verschoben wird, bis $Area(P_L^r) = AreaRequired(S_1)$ gilt. Zur Verdeutlichung dieser Vorgehensweise sollen folgende Punkte nochmals herausgestellt werden:

- Durch die Initialisierung kann auf dem Weg von w_1 zu w_k kein weiterer Standort liegen, d. h. $AreaRequired(S(P_L^r))$ ist konstant.
- $Area(P_L^r)$ wird mit Verschiebung von L_s stetig kleiner. Bei $L_s = S_1$ gilt $Area(P_L^r) = 0$.
- L_e ist fest, d.h. S_1 ist stets Teil von P_L^r .

Wenn die Bedingung $Area(P_L^r) = AreaRequired(S_1)$ eintritt, erfolgt eine Polygonzerlegung. Für P_L^r erfolgt keine weitere Zerlegung beim Aufruf von *ConvexDivide* (siehe Listing XX, Zeile 3), da nur S_1 auf dessen Rand liegt. Falls auf dem Rand von P_L^r mehr als ein Standort verbleibt, erfolgt eine erneute Zerlegung beim Aufruf von *ConvexDivide* (P_L^l) (siehe Listing XX, Zeilen 4 und 5).

Fall 2: $Area(P_L^r) < AreaRequired(S_1)$

Nach der Initialisierung der Linie L wird festgestellt, dass die Fläche von P_L^r kleiner ist als die benötigte Fläche von S_1. In diesem Fall erfolgt eine Vergrößerung von $Area(P_L^r)$ mit dem Ziel, die Anforderung von S_1 zu erfüllen. Hierbei fungiert L_s als Drehpunkt und L_e wird auf den nächsten in $W()$ vorkommenden Polygonpunkt oder Standort w_{k+1} gesetzt. Die Anforderung wird erneut geprüft. In diesem Schritt wird L_e von Polygonpunkt zu Polygonpunkt verschoben. Eine inkrementelle Verschiebung entlang des Polygons erfolgt dann unter Fall 2.1 beziehungsweise 2.2.

Hierbei kann es nun vorkommen, dass L_e auf die Koordinaten eines Punktes w_j in $W()$ gesetzt wird, welcher ein Standort $\neq S_1$ ist. Dieser Standort wird dann beim nächsten Vorrücken (also bei w_{j+1}) zur benötigten Fläche von P_L^r hinzugenommen. Bei Fall 2 kann $AreaRequired(P_L^r)$ demnach ansteigen, sodass ein Vorücken von L_e zwar zu einer größeren Fläche von P_L^r , nicht aber unbedingt zu einem günstigeren Verhältnis aus $Area(P_L^r)/AreaRequired(S(P_L^r))$ führt.
L_e wird so oft verschoben, bis eine der folgenden Bedingungen eintritt:

- $Area(P_L^r) > AreaRequired(S(P_L^r))$
- $L_e = S_n$

Je nachdem, wie weit L_e vorrückt und wie die Fläche von P_L^r zur Flächenanforderung von S(P_L^r) ist, werden nun weiter zwei Fälle unterschieden:

Fall 2.1: $L_e = S_n$ **und** $Area(P_L^r) > AreaRequired(S(P_L^r))$

In diesem Fall wird der Endpunkt L_e inkrementell im Uhrzeigersinn entlang des Polygons bewegt, bis $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt. Hinweis: Angenommen die Ausgangsposition von L_e ist w_j, dann muss es zwischen w_j und w_{j-1} eine Position geben, bei der $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt, da beim Vorrücken $Area(P_L^r)$ bei w_{j-1} zu klein und bei w_j zu groß war. Dieser Zwischenpunkt wird durch Interpolation gefunden.

Fall 2.2: $L_e = S_n$ **und** $Area(P_L^r) < AreaRequired(S(P_L^r))$

In diesem Fall wird der Anfangspunkt L_s inkrementell im Uhrzeigersinn entlang des Polygons bewegt, bis $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt.

Diese Vorgehensweise entspricht im Wesentlichen Fall 1, wobei L_s (initialisiert mit w_1) nun nicht im Uhrzeigersinn zum ersten Standort S_1 , sondern gegen den Uhrzeigersinn zum letzten Standort S_n bewegt wird. Vergleiche auch Abbildung XX, Fall (c) und (i).

3 Verallgemeinerung: Aufteilung eines nicht einfachen, nicht konvexen Polygons

Es wurde gezeigt, dass ein einfaches, konvexes Polygon rekursiv in n 1-Standort-Polygone aufgeteilt werden kann. Dieses Kapitel dient dazu einen verallgemeinerten Algorithmus zu skizzieren, damit auch für nicht einfache, nicht konvexe Polygone (Abbildung XX) das Problem der verankerten Flächenaufteilung gelöst werden kann. Bevor dieser Algorithmus vorgestellt wird, soll

die Beschreibung der dahinterliegenden Grundidee einen Überblick über die Vorgehensweise verschaffen. Danach werden die vorbereitenden Schritte vorgestellt und die Aufteilung des Polygons erläutert. Ein Beispiel dient anschließend zur Veranschaulichung des vorgestellten Algorithmus und zum Schluss des Kapitels wird der Sonderfall geschildert, dass Standorte im Inneren des Polygons liegen.

3.1 Grundidee

In Kapitel XX wurde bereits erläutert, wie ein einfaches, konvexes Polygon aufgeteilt werden kann. Dieses Vorgehen kann auch bei der Aufteilung nicht konvexer Polygone verwendet werden, muss jedoch in einigen Punkten erweitert werden. Als Voraussetzung wird angenommen, dass ein nicht einfaches, nicht konvexes Polygon P bereits in konvexe Teilpolygone $CP_1, ..., CP_p$ zerlegt wurde. Im ersten Schritt werden die Teilpolygone mithilfe einer Tiefensuche neu geordnet, um eine feste Bearbeitungsfolge für das weitere Vorgehen zu erhalten. Anschließend werden die Teilpolygone rekursiv aufgeteilt, wie es ähnlich bereits in Kapitel XX gezeigt wurde. Allerdings können nun Sonderfälle auftreten, die bei der Zerlegung eines einfachen, konvexen Polygons nicht vorkommen können. Einerseits kann CP_i weniger Fläche ausfüllen, als durch $AreaRequired(S(CP_i))$ gefordert ist. In diesem Fall ist CP_i Flächen-unvollständig und muss Flächen von anderen Teilpolygonen übernehmen. Andererseits kann es sein, dass einzelne Teilpolygone keinen Standort enthalten oder weniger Fläche ausfüllen, als durch $AreaRequired(S(CP_i))$ gefordert ist. In diesem Fall ist CP_i Standort-unvollständig und andere Teilpolygone müssen Flächen von CP_i übernehmen.
Die Neuordnung wird innerhalb der Prozedur *OrderPieces* umgesetzt und die Aufteilung inklusive der Sonderfallbehandlung wird durch die beiden Methoden *NonConvexDivide* und *DetachAndAssign* umgesetzt, die sich gegenseitig rekursiv aufrufen, bis ein n-Standort Polygon in n 1-Standort Polygone aufgeteilt wurde.

3.2 Aufteilung in konvexe Teilpolygone

Als Voraussetzung für die gleichmäßige Aufteilung eines nicht einfachen, nicht konvexen Polygons wird angenommen, dass das Polygon bereits in konvexe Teilpolygone aufgeteilt wurde. In verschiedenen Werken werden Möglichkeiten einer solchen Aufteilung vorgestellt. Ein Vorgehen wäre zum Beispiel, eine Triangulation eines Polygons zu erzeugen. In diesem Fall würde jedoch eine hohe Anzahl von Teilpolygonen entstehen. Um Teilpolygone zusammenzufassen, können nacheinander Kanten der Triangulation entfernt werden, solange das dadurch entstehende Teilpolygon weiterhin konvex ist. Hieraus wird ersichtlich, dass es verschiedene Möglichkeiten gibt, ein Polygon in konvexe Teilpolygone aufzuteilen. Zum Schluss dieser Arbeit wird besprochen, welche Auswirkungen diese vorbereitenden Schritte auf den Verlauf des vorgestellten Algorithmus haben können.

3.3 Neuordnung der konvexen Teilpolygone

Es kann nun davon ausgegangen werden, dass das Polygon P bereits in konvexe Teilpolygone $CP_1, ..., CP_p$ zerlegt wurde. Die Teilpolygone können willkürlich geordnet sein und die Indizes treffen keine Aussage über die tatsächliche Anordnung im Polygon P. Aus diesem Grund werden die Teilpolygone zuerst neu geordnet, was für den anschließend dargestellten Algorithmus erforderlich ist. Dazu wird ein Verbindungsgraph G gebildet und anhand dessen mittels einer Tiefensuche eine Ordnung erzeugt. Jedes Teilpolygon CP_i wird durch einen Knoten N_i in G abgebildet. Für jeden Nachbarn $CP_k (i \neq k)$ des Teilpolygons CP_i wird eine Kante zum jeweils korrespondierenden Knoten N_k eingefügt.
Wir definieren einen Knoten N_i in G als Blatt, wenn N entweder nur einen Nachbarn hat oder alle Nachbarn von N_i als besucht markiert wurden. Die Prozedur *OrderPieces* beschreibt nun die Neuordnung der Teilpolygone. *OrderPieces* wird mit einem beliebigen Knoten N_i von G initial aufgerufen. Zuerst wird geprüft, ob N_i bereits markiert wurde. Ist dies der Fall, kann der Aufruf zurückkehren. Falls N_i noch nicht markiert wurde, wird geprüft, ob N_i nach obiger Definition ein Blatt ist. Falls N_i kein Blatt ist, dann wird der Knoten markiert und für alle Nachbarn N_k von N_i rekursiv *OrderPieces* aufgerufen. Nach dem Rücksprung der Aufrufe aller Nachbarn von CP_i wird CP_i ausgegeben. Falls N_i ein Blatt ist, dann wird N_i markiert und CP_i ausgegeben. Anschließend wird für alle Nachbarn N_k von N_i rekursiv *OrderPieces* aufgerufen. Die neue Ordnung der CP_i ist nun die Reihenfolge, in der die Teilpolygone ausgegeben wurden.

3.4 Aufteilung eines nicht einfachen, nicht konvexen Polygons

Für die Aufteilung wird jedes Teilpolygon $CP_1, ..., CP_p$ betrachtet. Konkret wird das Polygon $PredPoly(CP_i)$ so aufgeteilt, dass ein Teilstück einem Standort in CP_i (falls vorhanden) zugeordnet wird und der Rest dem Polygon $PredPoly(CP_k)$ mit $k > i$ angehangen wird. Diese Aufteilung wird durch die sich gegenseitig rekursiv aufrufenden Prozeduren *NonConvexDivide* und *DetachAndAssign* erreicht. Erstere erzeugt ein Liniensegment, welches $PredPoly(CP_i)$ in zwei Teile aufteilt und letztere ordnet die Teile entweder einem Standort zu oder teilt sie erneut auf.

Zuerst wird die Prozedur *NonConvexDivide* beschrieben, die die Teilpolygone in zwei Teile aufteilt. Listing XX beschreibt diese Prozedur. Als Eingabe dient ein konvexes Teilpolygon, beschrieben durch die Liste W(CP_i) (mit $wk, k = 1, \ldots, m$) mit allen Polygonpunkten inklusive Steiner-Punkten und die Liste S(CP_i) mit den Standorten des Teilpolygons inklusive der jeweils benötigten Fläche. Anders als bei *ConvexDivide* aus Kapitel XX ist für die Bearbeitung relevant, welche Punkte w_1 und w_m in W(CP_i) ist. Die Kante, die durch die Polygonpunkte (w_m, w_1) erzeugt wird, sei nun die Kante zu $NextNeighbor(mcpi)$. Hat CP_i keinen nächsten Nachbarn, muss wm gleich einem Standort sein. Ansonsten gilt, dass W(CP_i) wieder gegen den Uhrzeigesinn geordnet ist.

Wie es auch schon bei *ConvexDivide* der Fall war, lässt die Prozedur erneut ein Liniensegment L gegen den Uhrzeigersinn durch das Polygon CP_i wandern, wobei L_s als Drehpunkt dient. L wird durch $(L_s\ ,\ L_e\ ) = w_1, S_i$ initialisiert, wobei S_i der erste Standort aus S(CP_i) ist.
Nun können zwei Fälle eintreten, in denen die Schleife stoppt:

- Die Fläche rechts der Linie L ist größer oder gleich der benötigten Fläche der Standorte, die sich in diesem Gebiet befinden. Es gilt: $Area(P_L^r) \geq AreaRequired(S(CP_L^r))$
- Das Ende des Polygons wird erreicht, also $L_e = w_m$.

Durch die Bearbeitung von vorherigen Teilpolygonen kann es sein, dass nicht zugewiesene Teile dieser Polygone in die Aufteilung von CP_i miteinbezogen werden müssen. Außerdem kann nun der Fall eintreten, dass die Fläche des Teilpolygons kleiner ist als $AreaRequired(S(CP_i))$. Aus diesem Grund müssen die oberen beiden Fälle noch feingranularer aufgeteilt werden.

Fall 1: Wie auch in der Prozedur *ConvexDivide* wird ein Ende der Linie L entlang des Polygons bewegt, um die Fläche $Area(P_L^r)$ zu verkleinern. Hierbei unterscheiden wir zwei Fälle. Falls $L_e = S_i$ für einen beliebigen Wert für i gilt, dann wird der Startpunkt L_s gegen den Uhrzeigersinn bewegt, ansonsten wird der Endpunkt L_e im Uhrzeigersinn bewegt.

Nun seien L_1 und L_2 zwei Liniensegmente, die einen gemeinsamen, festen Endpunkt haben. Dieser gemeinsame Endpunkt ist für beide Linien entweder L_s oder L_e und damit das Gegenstück zum oben bestimmten Punkt, welcher entlang des Polygons bewegt wurde. Die Linien sind so positioniert, dass $Area(P_{L_1}^r) < AreaRequired(S(CP_{L_1}^r))$ und $Area(P_{L_2}^r) > AreaRequired(S(CP_{L_2}^r))$ gilt. Die Linie L_2 wird demnach durch (w_1, w_k) und die Linie L_1 durch (w_1, w_{k-1}) beschrieben. Dadurch entsteht ein Dreieck $T = (t_1, t_2, t_3)$, das die Differenz von $CP_{L_1}^r$ und $CP_{L_2}^r$ bildet. Außerdem sei (t_1, t_2) das Liniensegment von CP_i , das L_1 und L_2 verbindet. Der gemeinsame Endpunkt von L_1 und L_2 ist demnach t_3.

Nun muss $CP_{L_1}^r$ mit einer Teilfäche des Dreiecks T und gegebenenfalls mit Teilflächen der Reste der Vorgängerpolygone vereinigt werden, damit die Flächenanforderungen der Standorte in $CP_{L_1}^r$ erfüllt ist. Dabei entstehen 3 Fälle: