# Aufteilung eines einfachen, konvexen Polygons

Die Eingangsdaten zur Berechnung der Flächenaufteilung sind ein konvexes Polygon CP sowie eine (nicht leere) Liste von Standorte S(CP), die auf dem Rand von CP liegen. Für jeden der n Standorte Si ist der geforderte Flächenanteil ci mit 0 < ci < 1 gegeben, sodass gilt. Das konvexe Polygon CP soll in n Polygone zerlegt werden, sodass jeder Standort auf dem Rand eines Polygons mit Fläche ci \* Area(CP) liegt.

## Grundidee

Bei der nachfolgend beschriebenen Lösung des Problems wird das konvexe Eingangs-Polygon CP mithilfe von Liniensegmenten schrittweise zerteilt. Jedes Liniensegment L ist hierbei vom Startpunkt Ls zum Endpunkt Le orientiert. Die beiden - bei der Teilung entstehenden - Polygone, werden entsprechend ihrer Lage zum Liniensegment benannt und erhalten die Bezeichnungen für das *rechts* und für das *links* des Liniensegments liegenden Polygons. Die exakte Positionierung der Liniensegmente wird in Kapitel xx erläutert. An dieser Stelle soll genügen, dass die Liniensegmente so positioniert werden, dass die Fläche von der geforderten Fläche der auf dem Rand von liegenden Standorte entspricht ( analog). Die Teilung wird so oft wiederholt, bis je Polygon nur noch ein Standort auf dessen Rand liegt[[1]](#footnote-1).

Ein Bild, das Text, Gerät, Messanzeige, Anzeige enthält.

Automatisch generierte Beschreibung

Abbildung : Schematische Zerlegung eines konvexen Polygons CP in vier konvexe Polygone CP1 ... CP4 mit c1 ... c4

Mit den genannten Überlegungen lässt sich ein *divide-and-conquer* - Algorithmus zur Flächenaufteilung eines konvexen Polygons - basierend auf n Standorten - nun folgendermaßen skizzieren:

// Input: Convex polygon CP, List of sites S(CP) with #Sites > 1

// Add CP to list ConvexPolygons

while Length(ConvexPolygons) > 0 do

CP = ReadFirst(ConvexPolygons)

PrL, PlL = ConvexDivide(CP) // Divide into two convex polygons

if Length(S(PrL)) > 1 then addToConvexPolygons(PrL) else Output(PrL) end

if Length(S(PlL)) > 1 then addToConvexPolygons(PlL) else Output(PlL) end

Nach einer Teilung eines Polygons durch eine Linie L, wird für die beiden Polygone und jeweils geprüft, ob mehr als 2 Standorte vorliegen. Falls nein, ist der Zielzustand für diesen Standort erreicht und es ist keine weitere Flächenaufteilung erforderlich. Falls ja, wird das Polygon wieder in die Warteschlange der zu teilenden Polygone eingereiht.

# Lösung des Problems (Entwurf)

Aus vorangegangenem Kapitel bleibt noch offen, wie genau die Aufteilung eines konvexen Polygons CP in die Polygone und erfolgt, sodass in jedem Iterationsschritt AreaRequired(S1 ... Si) == Area() und AreaRequired(Si+1 ... Sn) == Area() gilt. Konkret ist zu klären, wie Anfangs- und Endpunkt der Schnittlinien positioniert werden. Dieses Kapitel geht daher auf die konkrete Umsetzung des Algorithmus für ConvexDivide()ein.

Eingangsdaten für den Algorithmus ConvexDivide() sind - wie zuvor - ein konvexes Polygon CP zusammen mit einer Liste von Standorten S(CP), die auf dem Rand von CP liegen. Für jeden der mindestens zwei Standorte Si ist der geforderte Flächenanteil ci mit 0 < ci < 1 gegeben.

* Beim ersten Aufruf, d.h. mit dem Ausgangs-Polygon CP, gilt weiterhin: , da hier alle Standorte auf dem Rand des (noch ungeteilten) Polygons vorhanden sind.
* Für Aufrufe von ConvexDivide() „nach der ersten Teilung“, d.h. mit konvexen Polygonen CPj, gilt dann , wobei i die Zählvariable über alle Standorte von S(CPj) ist.

Initialisierung:

* Der Startpunkt Ls der Linie L wird mit den Koordinaten des ersten Punkts in der Liste W initialisiert. Dieser ist nach Definition ein Polygonpunkt und insbesondere kein Standort.
* Der Endpunkt Le der Linie L wird mit den Koordinaten des ersten in W vorkommenden Standorts initialisiert.
* Hiermit ist sichergestellt, dass „rechts der Schnittlinie kein weiterer Standort als S1 liegt. Würde man das Polygon mit dieser Schnittlinie teilen, würde gelten S(PrL) == S1.

Je nach Fläche von PrL und der in S1 geforderten Fläche werden nun 3 verschiedene Fälle unterschieden:

Algorithmus, Fall 1 „Area(PrL) > AreaRequired(S1)“:

* Nach der Initialisierung wird festgestellt, dass die Fläche von PrL größer ist als die geforderte Fläche von S1.
* Der Anfangspunkt Ls wird nun so lange auf dem Rand des Polygons im Gegenuhrzeigersinn verschoben, bis Area(PrL) < AreaRequired(S1) gilt.
* Auf diesem Weg von w1 zu S1 kommt per Definition kein weiterer Standort vor, sodass die Fläche von PrL sich stetig verringert und gegen 0 strebt. Der gewünschte Zustand tritt also zwingendermaßen ein.
* Ist die Bedingung erfüllt, wird mit L der Teil von PrL abgeschnitten. PlL wird - sofern 2 oder mehr Standorte auf dem Rand liegen – wieder in ConvexPolygons eingereiht und erneut geteilt.

Algorithmus, Fall 2 „Area(PrL) < AreaRequired(S1)“:

* Der Endpunkt Le wird zum nächsten Punkt in W bewegt (CCW) und es wird erneut geprüft, ob die ggf. vergrößerte Fläche nun der Anforderung AreaRequired(S1) entspricht.
* Falls ja, ist bekannt, dass die Fläche von PrL unter Schnitt mit wk zu klein und unter Schnitt mit Le = wk+1 zu groß ist. Der Endpunkt Le muss nun nur solange CW von wk+1 zu wk bewegt werden, bis Area(PrL) < AreaRequired(S1) gilt
* Falls nein, wird wie unter Punkt 1 vorgegangen und Le wird zum nächsten Punkt in W bewegt (CCW). Sofern k-1 ein Standort ist, wird dieser zur Anforderung von S1 hinzuaddiert. Mit koninuierlicher Vergrößerung der Fläche von PrL muss daher die Anforderung AreaRequired(S(PrL)) nicht zwingend erfüllt werden.
* Für die Beschreibung von Fall 2 soll bei einem Punkt von W jedoch gelten Area(PrL) > AreaRequired(S(PrL)), sodass dann Punkt 2 „Falls ja“ eintritt.

Algorithmus, Fall 3 „Area(PrL) < AreaRequired(S1) nach Initialisierung” + Vorgehen unter Fall 2 führt dazu, dass Le = Sn gilt.

* Der Anfangspunkt Ls wird nun so lange auf dem Rand des Polygons im Uhrzeigersinn verschoben, bis Area(PrL) = AreaRequired(S(PrL)) gilt.

// Algorithm ConvexDivide()

// Input: Convex polygon CP, List of sites S(CP)

// Initialize: LS = w1, LE = wk (wk is at position of S1)

while Area(PrL) < AreaRequired(S(PrL)) and Le != Sn do

if (k > 1) and (wk-1 element of S(P)) then

S(PrL) = S(PrL) + wk-1

end

k = k + 1

Le = wk

...

## Beispiel

1. 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. [↑](#footnote-ref-1)