Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

**Christian Steffes**

# Platzieren von Wachtürmen

**9. Januar 2008**

Seminararbeit im WS 2007/2008

**Zusammenfassung**

Diese Arbeit behandelt das Problem der Platzierung zweier Wachtürme auf einem polyhedralen Gelände im $\mathbb{R}^2$ bzw im $\mathbb{R}^3$, so daß das komplette Gelände von den Spitzen der Türme aus sichtbar ist.

Als Kernliteratur für diese Arbeit diente [2] .

# Inhaltsverzeichnis

# 1 Problemstellung

Gegeben sei ein polyhedrales Gelände $T$ im $\mathbb{R}^2$ bzw im $\mathbb{R}^3$ mit $n$ Ecken.
Gesucht ist die kleinste Höhe $h > 0$, für die zwei Punkte $u, v \in T$ existieren, so daß die Wachtürme mit der Höhe $h$ an den Stellen $u$ und $h$ auf $T$ platziert, das komplett Gelände überblicken können.

Alle hier vorgestellten Algorithmen arbeiten mit einer parametric search. Für jedes Problem suchen wir eine Entscheidungsprozedur, die bei gegebenem $T$ und $h > 0 \in \mathbb{R}$ entscheidet, ob $T$ von zwei Türmen mit Höhe $< h$ überblickt werden kann.
Als nächstes verbinden wir die parametric search mit der Entscheidungsprozedur um einen Algorithmus zu erhalten, welcher berechnet, an welchen Positionen auf $T$ die zwei Wachtürme minimaler Höhe platziert werden müssen, so daß komplett $T$ von den Spitzen dieser Türme aus zu überwachen ist.

Diese Problemstellung teilen wir wie folgt in verschiedene Versionen auf.

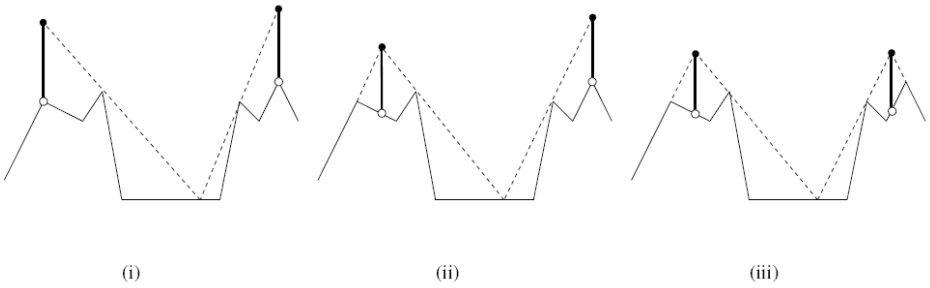


Abbildung 1: Die drei Versionen des Problems im $R^2$: (i) diskret, (ii) semi-diskret, (iii) kontinuierlich

## 1.1 Diskrete Version im $\mathbb{R}^2$

Beide Wachtürme werden auf den Vertices von $T$ platziert.
Der hier vorgestellte Algorithmus braucht $O(n^2 \log^4 n)$ Zeit.

## 1.2 Semi-diskrete Version im $\mathbb{R}^2$

Ein Turm wird auf einem Vertex von $T$ platziert, der zweite Wachturm kann beliebig auf der Ebene $T$ platziert werden.
Der hier vorgestellte Algorithmus braucht $O(n^2 \log^4 n)$ Zeit.

### 1.3 Kontinuierliche Version im $\mathbb{R}^2$

Beide Türme können beliebig auf $T$ platziert werden.
Der hier vorgestellte Algorithmus braucht $O(n^3\alpha(n)\log^3 n)$ Zeit.

### 1.4 Bewachen einer endlichen Teilmenge von $T$ im $\mathbb{R}^2$

Auch hier wird die kontinuierliche Version des Problems betrachtet. Beide Türme können beliebig auf $T$ platziert werden, jedoch gilt es nur eine endliche Teilmenge bestehend aus $m$ Punkten von $P$ zu bewachen.
Der in [2] vorgestellte Algorithmus braucht $O(mn\log^4 n)$ Zeit, wird hier jedoch nicht behandelt.

### 1.5 Diskrete Version im $\mathbb{R}^3$

Wie bei der diskreten Version im $\mathbb{R}^2$ werden hier auch beide Wachtürme auf den Vertices von $T$ platziert.
Der in [2] vorgestellte Algorithmus braucht $O(n^{11/3}polylog(n))$ Zeit, wird jedoch hier nicht behandelt.

## 2 Das diskrete und semi-diskrete Problem im $\mathbb{R}^2$

Sei $T$ einen X-monotone polygonale Kette im $\mathbb{R}^2$ mit $n$ Kanten. Sei $V$ die Menge der Vertices. u(h) bezeichne den vertikal über einem Punkt $u \in T$ mit Distanz $h$ gelegenen Punkt. Das Segment $uu(h)$ (kurz $u(h)$) bezeichnen wir als Wachturm mit Basis $u$ und Höhe $h$.

Wir betrachten nun das diskrete und das semi-diskrete Zwei-Wachtürme Problem im $\mathbb{R}^2$:

Gesucht ist die minimale Höhe $h > 0$, so daß zwei Wachtürme auf Vertices $u$ und $v$ platziert komplett $T$ überwachen können (bei der semi-diskreten Version einer auf einem Vertex, der andere beliebig auf $T$ platziert).

Um das Entscheidungsproblem zu lösen, müssen wir herausfinden, ob diese beiden Punkte $u$ und $v$ existieren. Die Entscheidungsprozedur findet bei gegebenem Wachturm $u(h)$ den Wachturm $v(h')$ minimaler Höhe $h' > 0$, der auf einem Vertex $v \in T$ (beim semi-diskreten Problem beliebig auf $T$) platziert werden kann.

### 2.1 Berechnung von Sichtbarkeitspaaren

Zunächst wird beschrieben, wie Sichtbarkeitspaare auf $T$ berechnet werden: Sei $P$ eine gegebene Menge von Punkten aus $T$ und sei $V$ die Menge der Vertices von $T$. Wir suchen für $v \in V$ einen Punkt $p \in P$ (dieser muss nicht notwendigerweise existieren), der folgenden Bedingungen genügt:

(i) $p$ liegt links von $v$

(ii) $p$ sieht $v$

(iii) $p$ ist der am weitesten rechts gelegene Punkt, der die Bedingungen (i) und (ii) erfüllt. Alternativ: Von den Punkten, die (i) und (ii) erfüllen, hat das Segment $pv$ des gesuchten Punktes $p$ die größte Steigung.

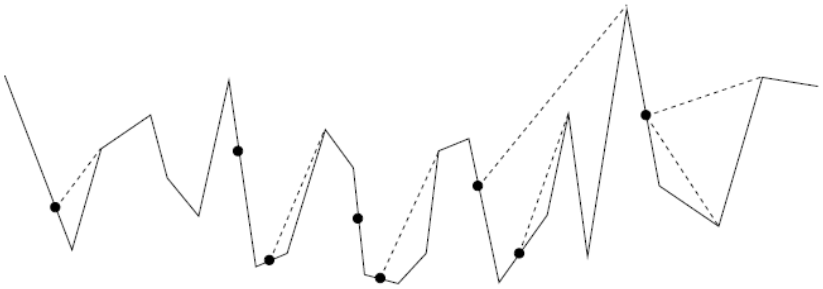

Abbildung 2: Einige Paare $(p, w) \in \Pi(P, V)$.

Sei $\Pi(P, V)$ die Menge der Punktpaare, die die obigen Bedingungen erfüllen. Es gilt: $|P| = O(n)$, da für jeden Vertex $v$ maximal ein solcher Punkt existiert und $|V| \in O(n)$. Somit gilt auch $|P| \in O(n)$.

Wir betrachten die Menge $P \cup V$ . Sei $l$ die vertikale Linie, die $P \cup V$ in zwei gleich große Teilmengen unterteilt. Ohne Beschränkung der Allgemeingültigkeit sei $l$ die Y-Achse, da sich durch Verschiebung der gesamten Szene in X-Richtung nichts verändert.

Wir unterteilen das Problem nun mit folgender Divide and Conquer Strategie: Seien $P_L$ und $V_L$ (bzw $P_R$ und $V_R$) die Teilmengen von $P$ und $V$, die links (bzw rechts) von $l$ liegen. Die Menge der Punkte $v \in V_R$, für die kein $p \in P_R$ existiert, so daß $(p, v) \in \Pi(P_R, V_R)$ bezeichnen wir mit $V'_R$. Sei $n'$ die Mächtigkeit von $V'_R$.

Nun gilt:

$$\Pi(P, V) = \Pi(P_L, V_L) \cup \Pi(P_R, V_R) \cup \Pi(P_L, V'_R).$$

Als nächstes berechnen wir $\Pi(P_L, V_L)$ und $\Pi(P_R, V_R)$ rekursiv, bestimmen die Menge $V'_R$ und berechnen $\Pi(P_L, V'_R)$ wie folgt:

Für alle $p \in P_L$ bestimmen wir alle Geraden $\lambda$, die durch $p$ gehen und $l$ schneiden ($p$ sieht $l$ durch die Gerade $\lambda$ im Punkt $\lambda \cap l$). Nun nutzen wir eine Dualisierung, die die Geraden $\lambda$ auf Punkte abbildet:

$$y = mx + b \mapsto (m, b).$$

$\gamma_p$ ist ein rechtswärts orientierter Strahl durch diese Punkte $(m, b)$, mit Ursprung im Punkt $w_p$, der im Dualen der Geraden von $p$ nach $l$ mit kleinster Steigung entspricht. Abbildung 3 zeigt die Konstruktion eines solchen

Strahls $\gamma_p$. Links die Geraden durch $p$ und deren Schnittpunkt mit $l$, rechts der entsprechende Strahl $\gamma_p$ mit Ursprungspunkt $w_p$.

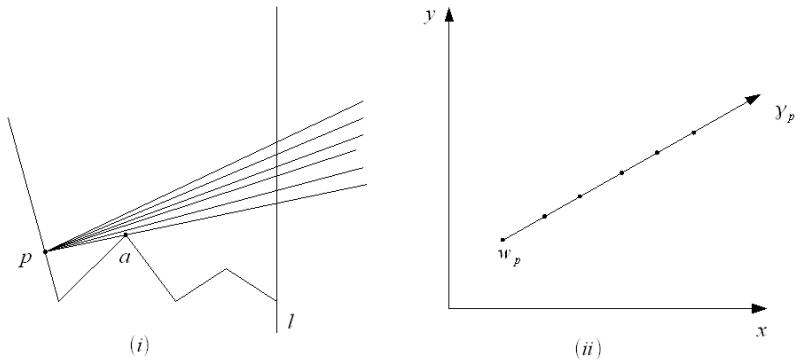


Abbildung 3: (i) Die Geraden $\lambda$ und die Schnittpunkte mit der $Y$-Achse $l$. (ii) Der Strahl $\gamma_p$ wobei $X$-Wert Steigung $m$ und $Y$-Wert der Schnittpunkt $b$ mit $l$.

Die Strahlen $\gamma_p$ sind paarweise disjunkt, denn ein Schnittpunkt von $\gamma_p$ und $\gamma'_p$ wäre dual zu einer Geraden, die durch $p$ und $p'$ geht. Da $p$ und $p'$ beide auf $T$ liegen, ist dies unmöglich.

Analog bestimmen wir für jeden Vertex $q \in V'_R$ alle Geraden $\lambda$ mit obiger Eigenschaft. Die dualen Punkte bilden den Strahl $\delta_q$. Dieser ist linkswärts orientiert, sein Ursprung liegt im Punkt $z_q$, der wieder der dualen Darstellung der Geraden $\lambda$ mit der geringsten Steigung entspricht. Auch hier sind alle $\delta_q$ paarweise disjunkt.

Sei $(p, q) \in \Pi(P_L, V'_R)$. Der duale Punkt zu der Geraden, die durch $p$ und $q$ verläuft, liegt sowohl auf $\gamma_p$ als auch auf $\delta_q$ und ist der am weitesten rechts gelegende Schnittpunkt von $\delta_q$ mit einem $\gamma_p$ (dies entspricht der Geraden mit der größten Steigung aus Bedingung (iii)).

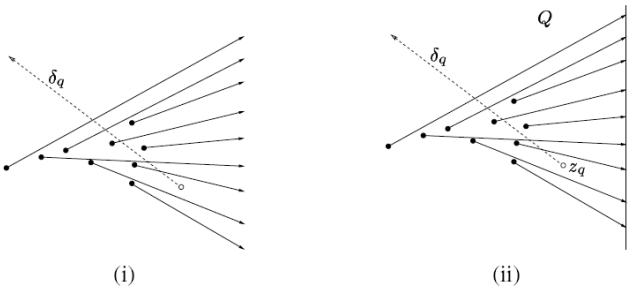


Abbildung 4: Das Polygon $Q$.

Wir nutzen folgendes Preprocessing, um eine Querystruktur aufzubauen: Sei $Q$ das kammartige einfache Polygon dessen Begrenzung aus allen Strahlen $\gamma_p$ und aus der vertikalen Linie $l_0$ bei $x = +\infty$ besteht (siehe Abbildung 4 ). Praktisch positionieren wir $l_0$ bei genügend großer X-Koordinate und schneiden die Strahlen $\gamma_p$ an deren Schnittpunkten mit $l_0$ ab. Der Punkt $p \in P$, der mit $q$ ein Paar in $\Pi(P_L, R'_R)$ bildet ist die Ausgabe einer ray shooting Anfrage in $Q$ durch $\delta_q$.

Als erstes konstruieren wir $Q$. Hierfür müssen wir die Strahlen $\gamma_p$ für $p \in P$ berechnen. Sei $T_L$ bzw $T_R$ die Region, die über $T$ und links bzw rechts von $l$ liegt. Der Schnittpunkt von $T$ mit $l$ sei mit $b_l$ bezeichnet. Wir berechenen die shortest path map in $T_L$ mit $b_l$ als Ursprung, wie in [6] beschrieben. So können wir in linearer Zeit das jeweils letzte Segment des kürzesten Pfades von $b_l$ zu allen Punkten $p \in P_L$ berechnen.

Für alle $p \in P$ ist der Startpunkt von $\gamma_p$ das Duale der Geraden, die das entsprechende letzte Segment $pa$ enthält, wobei $a$ der letzter Vertex auf dem kürzesten Weg von $b_l$ zu $p$ ist. Der Strahl $\gamma_p$ selber ist in der dualen Linie zum Punkt $p$ enthalten, da alle Punkte, die den Strahl bilden, Geraden entsprechen, die durch $p$ gehen. Somit ist $\gamma_p$ nun berechnet.

Als nächstes müssen wir nun die Strahlen $\gamma_p$ nach ihren Schnittpunkten mit $l_0$ sortieren. Diese Sortierung entspricht einer Sortierung nach Steigung der Geraden $\gamma_p$, was wiederum der Sortierung nach X-Koordinaten der entsprechenden dualen Punkte $p \in P_L$ entspricht. Diese Sortierung kann vor Beginn des Divide and Conquer Prozesses berechnet werden (die Berechnung der Sortierung nach X-Koordinaten über $P$ erfolgt in $O(n \log n)$).

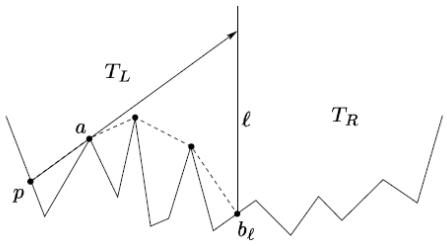


Abbildung 5: Der letzte Segment $pa$ des kürzesten Pfades von $b_l$ nach $p$.

Mit Hilfe des Algorithmus von Guibas et al [6] bzw [3] berechnen wir $Q$ mit linearem Zeitbedarf vor (Preprocessing). Spätere ray shooting Anfragen können dann mit logarithmischem Zeitbedarf pro Anfrage wie folgt berechnet werden. Wir konstruieren die Startpunkte der Strahlen $\delta_q$ für $q \in V'_R$. Dies geschieht symmetrisch zur Berechnung der Strahlen $\gamma_p$ unter Verwendung der Region $T_R$ anstatt $T_L$. Dieser Schritt benötigt wiederum lineare Laufzeit. Als nächstes führen wir ray shooting Anfragen in $Q$ mit den Strahlen $\delta_q$ ($q \in V'_R$) aus. Dabei erhalten wir die Menge $\Pi(P_L, V'_R)$. Somit betragen

die Gesamtkosten des Merge-Schrittes der Rekursion $O(n \log n)$ und $\Pi(P, V)$ kann in $O(n \log^2 n)$ berechnet werden.

## 2.2 Die Entscheidungsprozedur

Sei $H$ die Region der Ebene, die über $T$ liegt. In der Entscheidungsprozedur setzen wir einen Vertex $u \in T$ fest, bauen an dieser Stelle den ersten Wachturm der Höhe $h$ und berechnen das Sichtbarkeitspolygon $W(u(h))$, also den Teil von $T$, der von $u(h)$ aus sichtbar ist (siehe Abbildung 6). $W(u(h))$ kann mit linearem Zeitaufwand, wie in [6] gezeigt, berechnet werden. Der Teil von $\partial W(u(h))$ der auf $T$ liegt besteht aus Teilabschnitten der Kanten von $T$ oder aus Abschnitten von Sichtbarkeitsstrahlen mit Ursprung $u(h)$ welche durch einen Vertex von $T$ geht. In beiden Fällen ist mindestens ein Endpunkt jedes sichtbaren Teilstücks ein Vertex.

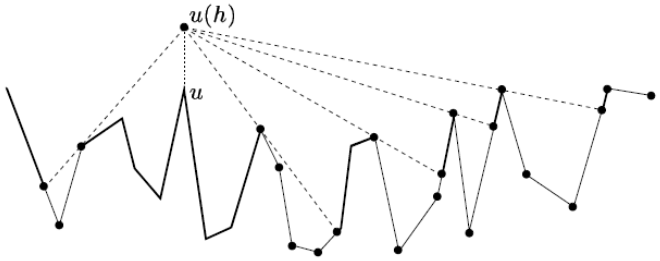


Abbildung 6: Das Sichtbarkeitspolygon von $u(h)$ aus berechnet.

Sei $P = P(u, h)$ die Menge der Endpunkte aller von $u(h)$ aus nicht sichtbaren Teilsegmenten von $T$. Nun ist klar, daß $u(h)$ und ein zweiter Turm $v(h')$, mit eventuell anderer Höhe $h'$, genau dann $T$ vollständig überblicken, wenn $v(h')$ alle Punkte von $P(u, h)$ sieht. Es reicht hier auf Sichtbarkeit der Ecken einer Kante zu prüfen, denn wenn ein Punkt $v(h')$ über $T$ beide Endpunkte eines Teilsegmentes sieht, so sieht er das gesamte Teilsegment.

Das folgende Lemma liefert eine entscheidende geometrische Eigenschaft auf der der Algorithmus beruht.

**Lemma 1** *Von der Turmspitze $v(h')$ mit $v \in T$ und Höhe $h$ sind alle Punkte $p \in P$, die links von $v$ liegen genau dann sichtbar, wenn $v(h')$ über allen Geraden $pw$ mit $(p, w) \in \Pi(P, V)$ liegt, wobei $w$ links von $v$ liegt oder gleich $v$ ist.*

*Alternativ: $v(h')$ sieht alle Punkte von $P$ die links von $v$ liegen genau dann, wenn $v(h')$ für jedes $p$ über der Geraden $pw$ mit maximaler Steigung liegt. Auch hier sei $(p, w) \in \Pi(P, V)$ und $w$ liegt links von $v$ oder entspricht $v$.*

**Beweis.** Wenn $v(h')$ alle Punkte von $P$ die links von $v$ liegen oder $v$ entsprechen sieht, so muss $v(h')$ über all den Geraden $pw$ aus der ersten Aussage des Lemmas liegen, sonst wäre der entsprechende Punkt $p$ von $v(h')$ aus nicht sichtbar.
Angenommen $v(h')$ liege über all jenen Geraden und $p \in P$ links von $v$. Wenn $v(h')$ $p$ nun nicht sieht, dann muss der kürzeste Pfad von $p$ nach $v(h')$, der über $T$ verläuft an einem Vertex von $T$ abknicken. Sei $w$ der linkeste solche Vertex. Wenn $(p, w) \in \Pi(P, V)$, dann genügt $(p, w)$ den Bedingungen des Lemmas und $v(h')$ liegt unterhalb der Linie $pw$. Dies ist ein Widerspruch zur Annahme.
Wenn $(p, w) \notin \Pi(P, V)$, dann muss ein anderer Punkt $p' \in P$ existieren, der zwischen $p$ und $w$ liegt, so daß $(p', w) \in \Pi(P, V)$. Die Gerade $p'w$ hat jedoch noch größere Steigung als $pw$ und somit liegt $v(h')$ unterhalb dieser Geraden, was einen Widerspruch zur Annahme darstellt.

Die zweite Aussage des Lemmas ist offensichtlich. □

Analog zu Lemma 1 existiert eine symmetrische Version, welche die Sichtbarkeit der Punkte aus $P$, die rechts von $v$ liegen behandelt. Hierfür verwenden wir eine symmetrische Version von $\Pi(P, V)$ mit Paaren $(p, v) \in P \times V$ wobei $p$ rechts von $v$ liegt, welche analog völlig symmetrisch definiert und konstruiert wird.

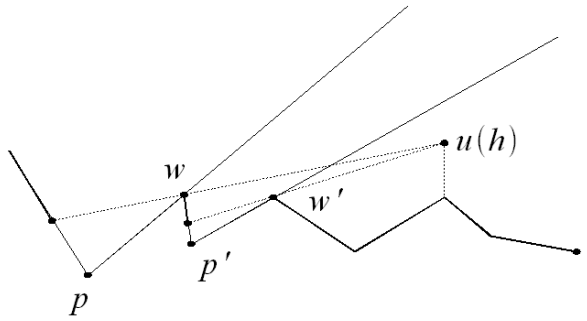


Abbildung 7: Der von $u(h)$ aus sichtbare Teil von $T$ und die Geraden $(p, w)$ und $(p', w')$.

Als nächstes sweepen wir mit einer vertikalen Linie $l$ von links nach rechts durch die Ebene. Wir beginnen bei der X-Koordinate des linkesten Vertex von $T$. Wir brauchen eine Funktion $h_L$, die für jedes $q \in T$ die Höhe des niedrigsten Wachturms, an Position $q$ platziert, der alle Punkte aus $P$ sieht, die links von $q$ liegen, berechnet. Äquivalent: $h_L(q)$ entspricht der Entfernung von $q$ zur oberen Kontur $E$ der Geraden $pw$, für alle Paare $(p, w) \in \Pi(P, V)$, $w$ liegt links bzw entspricht $q$. Beim Sweepen bleibt $h_L(q)$ eine lineare Funktion, die der vertikalen Distanz zwischen einer Kante von $E$ und einer Kante von $T$ entspricht. Wenn wir einen Vertex von $E$ erreichen, gehen wir zu einer anderen Kante von $E$ über und die Aktualisierung von

$h_L$ ist offensichtlich und einfach.
Wenn wir einen Vertex $v$ von $T$ erreichen, kommt zusätzlich zum offensichtlichen lokalen Update von $h_L$ noch ein dynamisches Update von $E$, durch Hinzufügen der Geraden $pv$ ($(p, v) \in \Pi(P, V)$) zur Linienmenge, die $E$ bildet hinzu. Falls das Paar $(p, v)$ nicht existiert, so bleibt $E$ unverändert.
Also bleibt $E$ entweder unverändert oder, da $E$ konvex ist, kann die Erstellung zweier neuer Vertices und das Löschen der alten Vertices dazwischen in $O(\log n)$ erfolgen. Nach jedem Update müssen wir noch den nächsten Vertex aus $E$ finden, der nun rechts von der aktuellen Position von $l$ liegt. Dies können wir in $O(1)$ pro Update erledigen. Somit ist die Berechnung von $h_L$ $\in O(n \log n)$.
Diese algorithmische Analyse impliziert, daß die kombinatorische Komplexität (Anzahl an Vertices des Graphen) der Funktion $h_L$ $O(n)$ ist.
$h_R$ berechnen wir symmetrisch von rechts nach links, beginnend beim rechtesten Vertex von $T$.

Dann konstruieren wir die obere Kontur $h^*$ von $h_L$ und $h_R$ in $O(n)$ Zeit, indem wir von links nach rechts sweepen und dabei die Lage der beiden Konturen zueinander betrachten.

Für die diskrete Version betrachten wir nun die Entfernung aller Vertices $v \in T$ zur oberen Kontur $h^*$ und geben das Minumum aus. Im semi-diskreten Fall, entspricht das globale Minimum von $h^*$ der niedrigsten Höhe des zweiten Turmes, der beliebig auf $T$ platziert, mit $u(h)$ komplett $T$ überblickt. Da wir diesen Schritt für jeden Vertex $u \in T$ wiederholen müssen, ist die Laufzeit der Entscheidungsprozedur $O(n^2 \log^2 n)$.

Wir haben die optimale Höhe $h^*$ gefunden, wenn nun für eine erste Turmposition $u$ eine zweite Positionierung mit Höhe gleich $h$ existiert und für alle anderen $u$ der zweite Türm gleich groß oder größer als $h$ ist. Dementsprechend ist $h$ zu klein gewählt, wenn wir keinen zweiten Turm mit Höhe kleiner oder gleich $h$ finden und zu groß gewählt, wenn ein zweiter Turm existiert, der kleiner als $h$ ist.

Wir fassen diese Ergebnisse in Lemma 2 zusammen:

**Lemma 2**

*(i) Bei gegebenem Gelände $T$ mit $n$ Kanten, einem Vertex $u \in T$ und einer Höhe $h \geq 0$, kann man den zweiten Wachturm minimaler Höhe, der beliebig auf $T$ platziert werden kann und der zusammen mit dem Wachturm $u(h)$ komplett $T$ überwacht, in $O(n \log^2 n)$ ermitteln.*

*(ii) Bei gegebenem $T$ mit $n$ Kanten und einer Höhe $h \geq 0$ kann man in $O(n^2 \log^2 n)$ berechnen, ob $h$ kleiner, größer oder gleich der optimalen Wachturmhöhe für das diskrete bzw das semi-diskrete Zwei-Wachturm-Problem ist.*

## 2.3 Anwendung der paramteric search

In diesem Abschnitt wird beschrieben, wie die optimale Höhe $h^*$ der Wachtürme beim diskreten und semi-diskreten Problem durch parametric search (wie in [8] beschrieben) ermittelt werden kann.

Um das von Megiddo in [8] vorgestellte Verfahren nutzen zu können, brauchen wir eine Entscheidungsprozedur, die bei Eingabe $h$ entscheidet, ob $h < h^*$, $h = h^*$ oder $h > h^*$ gilt. Megiddo beschreibt auch, wie diese Prozedur zur Kostenreduktion zu parallelisieren ist. Bei $T_\Pi$ parallelen Suchschritten und $p$ Prozessoren betragen die Gesamtkosten $O(T_\Pi(p + D \log p))$, wobei $D$ die Kosten eines Aufrufs der Entscheidungsprozedur sind. Hier: $D = O(n^2 \log^2 n)$. Dabei wird die Entscheidungsprozedur mit $h$ als kritischer Variable ausgeführt.

Für Vergleiche mit $h$ werden kritische Werte ermittelt. Als kritische Werte bezeichnen wir Werte von $h$, bei denen die Ergebnisse der Vergleiche mit $h$ sich verändern. Haben wir diese Werte ermittelt, führen wir die Entscheidungsprozedur nun mit diesen konkreten Werten aus, um den Wertebereich, der die optimale Höhe $h^*$ enthält, zu finden.

### 2.3.1 Parametric search für die diskrete Problemstellung

Als erstes suchen wir einige kritische Höhen. Für jeden Vertex $u \in T$ berechnen wir die Sichtbarkeitsstruktur für den vertikalen Strahl über $u$ (mit dem Algorithmus aus [6]) in $O(n^2)$, ($O(n)$ pro Vertex $u$). Siehe hierzu Abbildung 8.
Wir erhalten für jeden Vertex $u$ alle Sichtbarkeitspaare $(v, v')$ von Vertices $v$ und $v'$, so daß ein Punkt über $u$ $v$ und $v'$ entlang der Linie $vv'$ sehen kann. Desweiteren erhalten wir den ersten Punkt auf der Geraden von dem Punkt über $u$ aus nach $v$ und $v'$, an welchem diese Gerade $T$ kreuzt und unter $T$ weiter verläuft. Dies kann der von $u$ aus weiter entfernte Vertex $v$ oder $v'$ sein oder ein beliebiger, noch weiter entfernter Punkt sein.
Die Anzahl solcher kritischen Ereignisse ist linear für jedes $u$ (wie in [6] gezeigt). Wir erhalten $O(n)$ kritische Höhen über $u$ - eine für jedes dieser Ereignisse.
Wenn wir diesen Schritt für alle Vertices $u \in T$ wiederholen, liegt uns eine Menge $H_0$ mit $O(n^2)$ kritischen Höhen und eine entsprechende Menge $\mathcal{J}$ mit Schnittpunkten von $T$ und den Sichtbarkeitsstrahlen $vv'$ vor. Wir lassen eine binäre Suche mit der Entscheidungsprozedur auf $H_0$ laufen. Nach $O(\log n)$ Aufrufen der Entscheidungsprozedur mit Gesamtkosten $O(n^2 \log^3 n)$ erhalten wir ein Intervall $I_0$, welches auch die optimale Höhe $h^*$ enthält. Für alle $u \in T$ und für alle $h \in I_0$ hat das Sichtbarkeitspolygon $W(u(h))$ eine feste kombinatorische Struktur. Für jeden sichtbaren Teil einer Kante $e \in T$ steht die Art der Endpunke fest: Jeder Endpunkt ist entweder ein Vertex aus $T$

oder der Schnittpunkt von $e$ mit einem Sichtbarkeitsstrahl, der in $u(h)$ startet und durch einen Vertex von $T$ geht, bevor er $e$ erreicht.
Die Menge der Endpunkte der sichtbaren Segmente von $W(u(h))$ die keine Vertices von $T$ sind, bezeichnen wir mit $P(u,h)$. $P(h)$ ist die Vereinigung von $P(u,h)$ mit allen Vertices $u$ und hat die Größe $O(n^2)$. Wichtig: Diese Mengen sind abhängig von $h$.

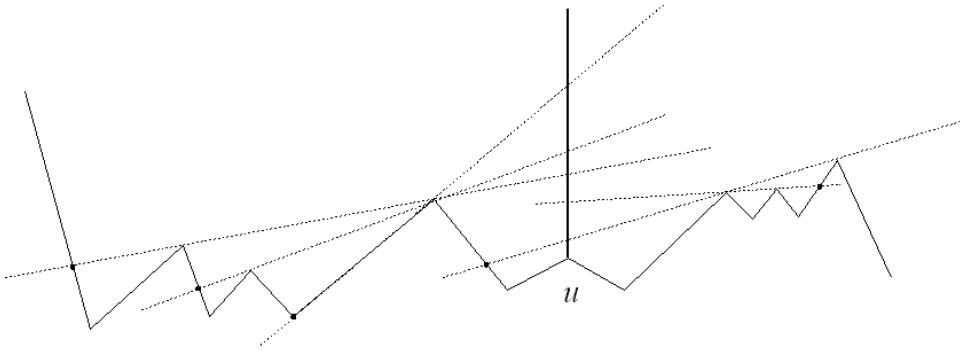


Abbildung 8: Die Schnittpunkte mit $T$ bilden $\mathcal{J}$. Die Schnittpunkte mit der vertikalen Geraden über $u$ bilden die Menge $H_0$.

Als nächstes wollen wir die Punkte von $P(h)$ zwischen den Punkten der Menge $\mathcal{J}$ lokalisieren. Wir betrachten, bei welcher Höhe $h$ der Punkt $P(h)$ seine Lage bezüglich eines Punktes aus $\mathcal{J}$ verändert und können mit dieser Höhe erneut die Entscheidungsprozedur ausführen und das Intervall, welches den optimalen Wert für $h$ enthält, so verkleinern. Dies geschieht, wie folgt:

Sei $S$ eine Aufteilung der Kanten von $T$ in Teilsegmente, welche durch die Punkte aus $\mathcal{J} \cup V$ begrenzt werden. Sei $I_s \subseteq I_0$ Intervall. Für die Startbelegung des Algorithmus zur Lokalisierung der Punkte $P(h)$ sei $S$ die Menge der Kanten von $T$, $I_S := I_0$ und $P_s(h) := P(h) \cap s$ für alle $s \in S$ eine Aufteilung der Menge $P(h)$.

Für jedes $s \in S$ sei $\mathcal{J}_s := \mathcal{J} \cap int(s)$. Segmente $s$, für die $\mathcal{J}_s = \emptyset$ gilt, werden aus $S$ gelöscht und wir geben $P_s(h)$ aus. Für alle übriggebliebenen $s$ sei $q_s$ der mittlere Punkt aus $\mathcal{J}_s$. $\forall p \in P_s(h)$ sei $h_p$ die Höhe, bei welcher $p\, q_s$ entspricht. Wenn diese Höhe nicht existiert oder außerhalb des Intervalls $I_S$ liegt, dann wissen wir nun welche Hälfte von $s$ $p$ enthält und wir ignorieren $p$ im weiteren Verlauf dieses iterativen Schritts. Dies wiederholen wir für alle $s \in S$ und erhalten somit die Menge $H_S$ mit $O(n^2)$ kritischen Höhen (siehe Abbildung 9). Wir nutzen nun wieder binäre Suche mit der Entscheidungsprozedur auf den Höhen $H_S$ und erhalten das Intervall $I' \subseteq I_S$. Wir wissen nun welche Hälfte von $s$ $p$ enthält. Wir erhalten die Menge $S'$ durch Aufteilung der Segmente $s \in S$ an ihrem Mittelpunkt $q_s$ in zwei Teilsegmente. Wir setzen $I_S := I'$ und erhalten eine neue Aufteilung von $P(h)$, indem wir jeden Punkt $p$ einem der neu entstandenen Teilsegmente zuweisen.

Nach $O(\log n)$ Schritten sind alle Segmente aus $S$ entfernt und jeder

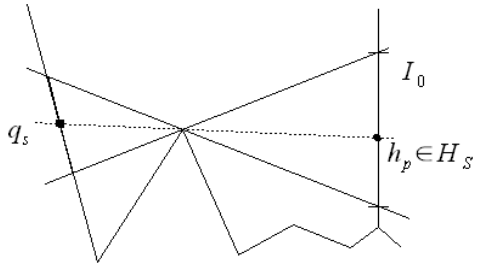


Abbildung 9: Der auf $s$ wandernde Punkt $p$ nimmt an $q_s$ die kritische Höhe $h_p \in H_S$ an.

Punkt $p \in P(h)$ bewegt sich für alle $h \in I_S$ zwischen zwei benachbarten Punkten aus $\mathcal{J} \cup V$. Die Gesamtkosten dieses Algorthmus sind durch $O(\log^2 n)$ Ausführungen der Entscheidungsprozedur $O(n^2 \log^4 n)$.

Sei $I_1$ das letzte erhaltene Intervall $I_S$. Für alle $h \in I_1$ und für jedes $u \in V$ ist die Menge $\Pi(P(u,h),V)$ nun bestimmt. Denn wenn sich die Menge $\Pi(P(u,h),V)$ verändert, so würde ein Punkt $p \in P(h)$ über einen Punkt aus $\mathcal{J} \cup V$ wandern. Dies ist ja nach Ausführung des obigen Algorthmus nicht mehr möglich (siehe Abbildung 10).

Wir speichern nun die Vertices von $T$ in den Blättern eines Binärbaums $\tau$ minimaler Höhe. Jeder innere Knoten $\xi$ repräsentiert die Menge $V(\xi)$ aller Vertices die in Blättern von Teilbäumen von $\xi$ gespeichert sind. Sei $L(\xi)$ die Menge aller Linien $pw$ mit $(p,w) \in \Pi(P,V)$ und $w \in V(\xi)$ und sei $E(\xi)$ die aus den Linien $L(\xi)$ gebildete obere Kontur. Da die obere Kontur $E(\xi)$ aus $|L(\xi)|$ Linien besteht ihre Komplexität $O(|L(\xi)|)$ und somit gilt: $\sum_{\xi \in \tau} |L(\xi)| = O(n \log n)$, d.h. die Komplexität aller Konturen $E(\xi)$ für einen festen ersten Turm $u$ ist $O(n \log n)$.

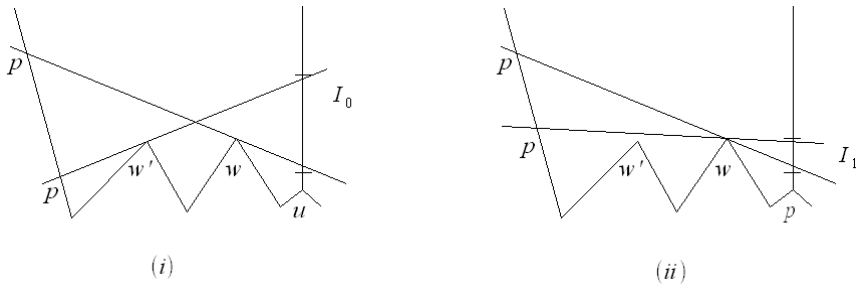


Abbildung 10: (i) Die Paare $(p,w) \in \Pi(P(u,h),V)$ sind nicht eindeutig. (ii) Nach Lokalisierung ist das Intervall $I_1$ klein genug, daher sind die Paare $(p,w)$ nun eindeutig.

Wir berechnen $E(\xi)$ indem wir die entsprechenden Linien $pw$ zu Punkten dualisieren und dann mit dem Algorithmus aus [1] die obere konvexe

Hülle dieser Punkte berechnen. Für eine Menge von $m$ Punkten benötigt der Algorithmus $O(m)$ Prozessoren und $O(\log m)$ Zeit. Somit können wir alle Konturen $E(\xi)$ für alle Knoten $\xi \in \tau$ und alle ersten Turmpositionen $u \in V$ parallel mit $O(n^2 \log n)$ Prozessoren und $O(\log n)$ parallelen Schritten berechnen. Die Laufzeit ist daher $O(n^2 \log^4 n)$. Zu beachten ist, dass diese Konturen sich immer noch mit der Höhe $h$ verändern, jedoch ist ihre Struktur für alle $h$ aus dem vorher berechneten Intervall $I_1$ eindeutig bestimmt. Die Paare $(p, w)$ sind eindeutig bestimmt und die zugehörigen Linien, die die Kontur bilden, sind bekannt. Somit wissen wir auch, welche Linien eine Ecke der Kontur bilden.

Als nächstes berechnen wir parallel für alle Konturen $E(\xi)$ und alle Ecken $q \in E(\xi)$, welche Kante von $T$ vertikal unter oder über $q$ liegt. Wir bearbeiten alle Eckpunkte $q$ parallel und nutzen für jeden Eckpunkte eine binäre Suche über seinen parametrischen X-Koordinaten im Vergleich zu den X-Koordinaten der Vertices von $T$. Dieser Schritt benötigt wieder $O(n^2 \log n)$ Prozessoren und $O(\log n)$ parallele Schritte und seine Laufzeit beträgt wieder $O(n^2 \log^4 n)$.

Die bisherigen Vorbereitungsschritte sind sowohl für die diskrete als auch für die semi-diskrete Variante des Problems gültig.

Im diskreten Fall bearbeiten wir nun alle Vertices $v \in T$ für eine feste erste Turmposition $u$ parallel. Für jedes $v \in T$ erhalten wir die Menge aller Vertices, die links von $v$ liegen, durch eine Vereinigung von $O(\log n)$ Teilbäumen von $\tau$. Für jeden dieser Teilbäume $\tau(\xi)$ wollen wir $v$ zwischen den Eckpunkten von $E(\xi)$ ausfindig machen. Da wir die Lage der Eckpunkte aller oberen Konturen zwischen den Vertices von $T$ bereits festgestellt haben, können wir diesen Schritt jetzt explizit (ohne parametric search) ausführen. Der Gesamtzeitbedarf der Berechnung für alle $v \in T$ und für alle ersten Turmpositionierungen $u \in T$ beträgt $O(n^2 log^2 n)$. Wir haben nun für jedes Turmpaar $u$, $v$ eine Menge von $O(\log n)$ Linien (zu den jeweiligen Konturen $E(\xi)$) über $v$ und berechnen dort deren maximale Höhe. Dieser Schritt ist immer noch parametrisch und kann mit $O(n^2 \log n)$ Prozessoren und $O(\log n)$ parallelen Schritten in $O(n^2 \log^4 n)$ Zeit berechnet werden. Wir geben nun ein Paar $u, v \in T$ aus, für welches die berechnete Höhe an der Position $v$ nicht mehr als $h$ ist. Somit beschränken wir $h$ weiterhin und wir geben das Minimum des übriggebliebenen Intervalls aus (außer $h^*$ wurde schon in einem der vorherigen Schritte gefunden). Hieraus folgt nun:

**Theorem 3** *Das diskrete Zwei-Wachturm-Problem im $\mathbb{R}^2$ mit $n$ Ecken kann in $O(n^2 log^4 n)$ Zeit gelöst werden.*

### 2.3.2 Parametric search für die semi-diskrete Problemstellung

Wir können bei der semi-diskreten Variante des Problems die Vorbereitungsschritte bis nach der Lokalisierung der Eckpunkte der Konturen übernehmen, d.h. wir haben schon die Konturen $E(\xi)$ berechnet und deren Eckpunktpositionen über den Kanten von $T$ berechnet.

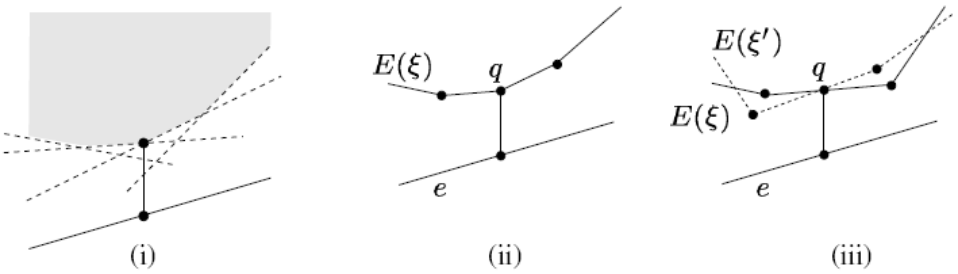


Abbildung 11: (i) Die kleinste Distanz von $e$ zu der entsprechenden oberen Kontur. (ii) Die kleinste Distanz ist an einem Eckpunkt einer einzelnen Kontur. (iii) Die kleinste Entfernung wird an einem Schnittpunkt zweier Konturen erreicht.

Wir wählen einen ersten Turm $u$. Für alle Kanten $e = v'v \in T$ nehmen wir den rechten Eckpunkt $v$ und bestimmen wie oben die Menge der Vertices die links von $v$ liegen (Vereinigung von $O(\log n)$ Teilbäumen von $\tau$). Wir suchen nun die kleinste vertikale Entfernung von $e$ zu der oberen Kontur der $O(\log n)$ Konturen $E(\xi)$ der entsprechenden Teilbäume $\tau(\xi)$.

Diese kleinste Distanz der Kante $e$ zur oberen Kontur kann nur an besonderen Konstellationstypen angenommen werden (siehe Abbildung 11):

(i) An einem Endpunkt von $e$.

(ii) An einem Eckpunkt $q$ einer Kontur $E(\xi)$.

(iii) An einem Schnittpunkt zweier Konturen $E(\xi)$ und $E(\xi')$.

Im Fall (ii) muss die Kante, die links von $q$ liegt, kleinere Steigung (die rechts von $e$ liegende Kante größere Steigung) als $e$ haben, sonst wäre ein Punkt links (bzw rechts) von $q$ noch näher an $e$. Im Fall (iii) ist $q$ ein Schnittpunkt zweier Konturen und die Steigung von $e$ liegt zwischen der Steigung der Kanten aus $E(\xi)$ und $E(\xi')$, die den Schnittpunkt $q$ bilden.

Wir betrachten Fall (iii). Beide Teilkonturen $E(\xi)$ und $E(\xi')$ gehören zu Teilbäumen $\tau(\xi)$ und $\tau(\xi')$, die linke Kinder von Knoten auf dem Pfad in $\tau$ von dem Blatt mit $v$ zur Wurzel sind. Ohne Beschränkung der Allgemeinheit: Sei $\xi$ tiefer im Baum gelegen als $\xi'$. Nun gilt: Wenn $\tau(\xi)$ Teil der Ausgabe für einen Vertex $v \in T$ ist, dann ist $\tau(\xi')$ auch Teil dieser Ausgabe. Wir

nennen $\xi'$ in diesem Fall linken Großonkel von $\xi$. Unabhängig von $v$ kann eine Teilkontur $E(\xi)$ Schnittpunkte vom Typ (iii) nur mit $O(\log n)$ anderen Teilkonturen $E(\xi')$ haben, wobei $\xi'$ höher im Baum gelegen ist und alle diese Knoten $\tau'$ linke Großonkel von $\xi$ sind (siehe Abbildung 12).

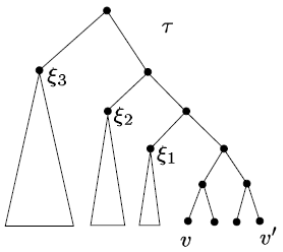


Abbildung 12: Der Baum $\tau$. Wenn ein Teilbaum $\tau(\xi_1)$ Teil der Ausgabe für ein $v$ ist, so sind alle Teilbäume, die linke Kinder von Knoten auf dem Weg von $\xi_1$ zur Wurzel sind, auch Teil der Ausgabe.

Wir fixieren die erste Turmposition $u$ und erhalten die entsprechende Menge $P = P(u, h)$ und den Baum $\tau$. Wir fixieren ein Paar $(\xi, \xi')$ von Knoten aus $\tau$, so dass $\xi'$ linker Großonkel von $\xi$ ist. Dann bauen wir die obere Kontur $E(\xi) \cup E(\xi')$ durch Einfügen der Vertices und Kanten von $E(\xi)$ in $E(\xi')$ wie folgt: Lokalisiere die Lage aller Ecken $q \in E(\xi)$ zwischen den Ecken von $E(\xi')$ entsprechend der X-Koordinate mit parametrischer binärer Suche. Für alle Kanten $e \in E(\xi)$ suchen wir nun, sofern diese existieren, die Schnittpunkte mit $E(\xi')$. Da $E(\xi')$ eine konvexe polygonale Kette ist, kann es für jede Kante $e \in E(\xi)$ maximal zwei Schnittpunkt geben. Diese können durch parametric binary search wie folgt gefunden werden. Für jedes Paar $(\xi, \xi')$ weisen wir jedem Eckpunkt und jeder Kante aus $E(\xi)$ einen Prozessor zu. Da wir insgesamt $O(n \log n)$ Konturen $E(\xi)$ haben und jede dieser Konturen $O(\log n)$ linke Großonkel hat, brauchen wir für diesen Schritt $O(n \log^2 n)$ Prozessoren und $O(\log n)$ parallele Schritte. Diese Prozedur lassen wir für alle ersten Turmpositionen $u$ parallel laufen und brauchen somit $O(n^2 \log^2 n)$ Prozessoren und $O(\log n)$ parallele Schritte. Daher braucht dieser Schritt insgesamt wieder $O(n^2 \log^4 n)$ Zeit.

Nun konstruieren wir die Menge $B$, bestehend aus allen Vertices von $T$, allen Eckpunkten aller einzelnen Konturen $E(\xi)$ und den Schnittpunkten aller Konturpaare $E(\xi)$ und $E(\xi')$ für alle ersten Turmpositionen $u$. Aus der vorhergehenden Analyse folgt, dass $B$ die Mächtigkeit $O(n^2 \log^2 n)$ hat. Wie vorher lokalisieren wir nun auch die Eckpunkte der Konturpaare zwischen den Vertices von $T$ und kennen somit die Lage aller Elemente von $B$. Dieser Schritt kostet auch wieder $O(n^2 \log^4 n)$ Zeit.

Wir bearbeiten nun für jede feste erste Turmposition $u$ alle Kante $e \in T$.

Wie im diskreten Fall suchen wir für den linken Vertex $v$ von $e$ wieder die Menge $L(v)$ der Linien $pw$ durch Vereinigung von $O(\log n)$ Teilbäumen von $\tau$. Sei $B(v)$ die Teilmenge von $B$, die aus den zwei Endpunkten $v'$ und $v$ von $e$ und aus Eckpunkten von Konturen und Eckpunkten von Paaren von Konturen, die über $e$ liegen, besteht. Außer den enthaltenen Vertices aus $T$ sind die Mengen $B(v)$ paarweise disjunkt. Alle Mengen $B(v)$ für alle ersten Turmpositionen $u$ können in $O(|B|) = O(n^2 log^2 n)$ Zeit errechnet werden. Dieser Schritt ist nicht mehr abhängig von $h$.

Mit fixierten $u$ und $v$ weisen wir nun jedem Paar von Punkt $q$ $inB(v)$ und zur Ausgabe von $v$ gehörenden Kontur $E(\xi'')$ einen Prozessor zu. Die Gesamtzahl an Prozessoren ist $O(n^2 \log^3 n)$. Jeder Prozessor muss, mit Hilfe von parametrischer binärer Suche über die Ecken von $E(\xi'')$, bestimmen, ob $q$ unter $E(\xi'')$ liegt. So können wir mit $O(n^2 \log^3 n)$ Prozessoren und $O(logn)$ paralleler Suchtiefe alle Punkte $q \in B$ finden, die nicht von oben durch eine andere Kontur verdeckt werden. Von diesem Punkten existiert genau einer, für jedes Paar $u$ und $v$, der die Eigenschaft besitzt, dass die Steigung von $e$ zwischen den Steigungen der beiden Konturkanten liegt, welche zu $q$ gehören. Dieser Punkt ist der am nächsten vertikal über $e$ gelegene Punkt der oberen Kontur.

Am Ende testen wir (parametrisch) welcher der übriggebliebenen Punkte näher als $h$ von der entsprechenden Kante $e$ entfernt liegt. Wir geben wieder das Minumum des so erhaltenen Intervalls aus.

Die Gesamtlaufzeit des Algorithmus ist $O(n^2 \log^4 n)$.

**Theorem 4** *Das semi-diskrete Zwei-Wachturm-Problem im* $\mathbb{R}^2$ *mit n Ecken kann in* $O(n^2 log^4 n)$ *Zeit gelöst werden.*

# 3 Das kontinuierliche Zwei-Wachturm-Problem im $\mathbb{R}^2$

In diesem Abschnitt wird die kontinuierliche Version des Zwei-Wachturm-Problems behandelt. Wir suchen wiederum die niedrigste Höhe $h$ zweier Wachtürme, die zusammen komplett $T$ überblicken können. Diese beiden Wachtürme dürfen beliebig auf dem Gelände $T$ platziert werden. Auch hier erlaubt uns die parametric search Technik ein Entscheidungsproblem in ein Suchproblem zu überführen. Als erstes wird nun die Entscheidungsprozedur vorgestellt, die bei vorgegebener Höhe $h$ berechnet, ob zwei Wachtürme dieser Höhe auf $T$ platziert werden können, die komplett $T$ überwachen. Dann betrachten wir die parametric search, mit welcher wir die optimale Höhe für

ein solches Wachturmpaar bestimmen.

### 3.1 Die Entscheidungsprozedur

Seien $e_1$ und $e_2$ zwei unterschiedliche Kanten aus $T$. Seien $p \in e_1$ und $q \in e_2$ Punkte auf diesen Kanten und sei $s$ der $X$-Wert von $p$ bzw $t$ der $X$-Wert von $q$. Wir betrachten nun die Frage, ob es zwei Punkte $p$ und $q$ gibt, so dass die Turmspitzen $p(h)$ und $q(h)$ mit Höhe $h$ komplett $T$ überblicken können.

Hierzu sei $e_1(h)$ das Segment, das wir erhalten, wenn wir die Kante $e_1$ um die Höhe $h$ nach oben projizieren. Nun wollen wir die Sichtbarkeitsstruktur von $T$ ausgehend von $e_1(h)$ berechnen. Um diese effizient zu berechnen, müssen wir aus $T$ und dem Segment $e_1(h)$ ein einfaches Polygon konstruieren: Als erstes schliessen wir den Bereich über $T$ ab, indem wir am linken und rechten Ende von $T$ jeweils eine vertikale Kante nach oben einfügen und diese in ausreichender Höhe durch eine horizontale Kante $D$ verbinden. Als nächstes verbinden wir $e_1(h)$ ausgehend von dem linken Endpunkt des Segments mit $D$. Die dabei entstehende Kante nennen wir $f$. So entsteht das einfache Polygon $P'$ wobei $f$ und $e_1(h)$ als Doppelkanten anzusehen sind.

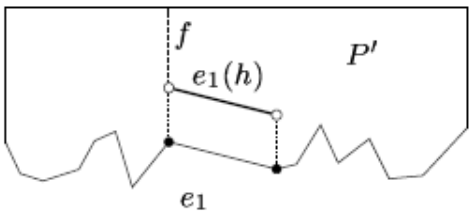


Abbildung 13: Das einfache Polygon $P'$ bestehend aus $T$ und $e_1(h)$

Nun können wir mit dem Algorithmus aus [4] in Zeit $O(n \log n)$ die Sichtbarkeitsstruktur von $P'$ von der Kante $e_1(h)$ aus berechnen. Allerdings stört uns hier noch die Kante $f$, die die Sichbarkeit einschränken könnte. Um dieses Problem zu lösen, konstruieren wir ein zweites einfaches Polygon $P''$, in dem wir $e_1(h)$ ausgehend von dem rechten Endpunkt des Segments mit $D$ verbinden und berechen nun für $P''$ die Sichtbarkeitsstruktur. Die beiden Ergebnisse verbinden wir und erhalten die Sichtbarkeitsstruktur von $T$ von $e_1(h)$ aus gesehen. Genauer wird hier $e_1(h)$ in $O(n)$ Intervalle unterteilt. Jedes dieser Intervalle wird durch Punkte begrenzt, von welchen aus zwei Vertices von $T$ auf einem Sichtbarkeitsstrahl liegen.
Sei $\sum(e_1, g)$ die Sequenz der Schnittpunkte solcher Strahlen mit einer Kante $g \in T$. Wenn die Linie, die $e_1$ enthält, einen Schnittpunkt mit der Kante $g$ besitzt, so fügen wir diesen Schnittpunkt auch in die Sequenz $\sum(e_1, g)$ ein. Es existieren insgesamt maximal zwei solche Schnittpunkte für alle Kanten

$g \in T$.

Bezeichne $p(s)$ den Punkt $p$ auf $e_1$ mit dem $X$-Wert $s$ und $p(s,h)$ den entsprechenden Punkt auf dem Segment $e_1(h)$. Sei $g$ nun eine Kante aus $T$ und $g \neq e_1$. Der Punkt $p(s,h)$ sieht einen Teil von $g$, welcher, wenn nicht leer, von dem von $e_1$ weiter entfernten Eckpunkt von $g$ und von einem Punkt $z(s) = z_g(s)$ begrenzt wird. Wenn wir den Punkt $p$ nun von links nach rechts auf $e_1$ laufen lassen, so bewegt sich $z_g(s)$ fortlaufend mit $p$. Solange $p(s,h)$ keine kritischen Sichtbarkeitsereignisse kreuzt entspricht $z(s)$ entweder dem näheren Eckpunkt von $g$ oder einem Schnittpunkt eines Sichtbarkeitsstrahls von $p(s,h)$ durch einen festen Vertex $v \in T$ mit $g$. Wenn $p(s,h)$ ein kritisches Sichtbarkeitsereignis und $z_g(s)$ einen entsprechenden Punkt in $\sum(e_1, g)$ passiert, so kann sich der Vertex $v$, der den Achspunkt der Sichtbarkeitslinie $p(s,h)z_g(s)$ darstellt, ändern. Die Bewegung von $z(s)$ bleibt jedoch stetig und monoton. Die Richtung hängt davon ab, ob $g$ links oder rechts von $e_1$ liegt und ob der Achspunkt $v$ über oder unter der Linie liegt, die das Segment $e_1(h)$ enthält. Falls der Achspunkt sich, während $p$ von links nach rechts wandert, verändert, so liegt der neue Achspunkt $v'$ auf der selben Seite dieser Linie. Somit bleibt die Bewegung von $z_g(s)$ fortlaufend in die selbe Richtung.

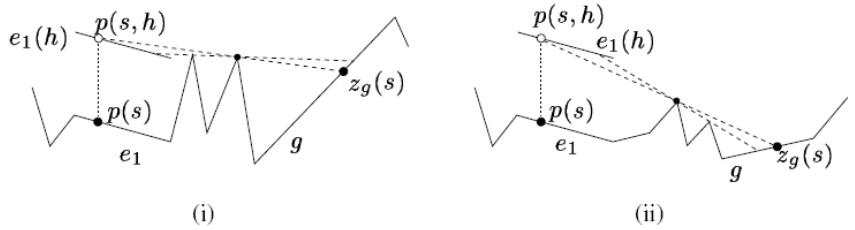


Abbildung 14: $p$ und entsprechend $p(s,h)$ bewegen sich von links nach rechts. Der entsprechende Punkt $z_g(s)$ läuft in (i) nach rechts und in (ii) nach links.

Wenn wir mit $z_g(s)$ die $X$-Koordinate dieses Punktes bezeichnen, so ist die Funktionsschar $z_g(s)$ über alle Kanten $g \in T$ eine Schar stetiger, stückweise linearer, rationaler Abbildungen von $s$.

Seien $a$ und $b$ zwei Punkte auf $e_1(h)$ und $c$ und $d$ die entsprechenden Endpunkte der von $a$ und $b$ aus sichtbaren Segmente von $g$, wobei $ac$ und $bd$ über den selben Vertex $o \in T$ peilen. Wenn wir $o$ als Ursprung annehmen, können wir $c = -\xi a$ und $d = -\eta b$ für positive Skalare $\xi$ und $\eta$ schreiben. Wählen wir einen Punkt $p = p(s,h) \in e_1(h)$ und den entsprechenden Endpunkt $z = z_g(s)$ des sichtbaren Intervalls von $g$. Wir können $p$ und $z$ schreiben als: $p = \lambda a + (1-\lambda)b$ für $\lambda \in [0,1]$ und $z = \mu c + (1-\mu)d$

für $\mu \in [0,1]$. Da die Punkte $p$, $o$ und $z$ kollinear sind, sind die Vektoren $\lambda a + (1-\lambda)b = \mu c + (1-\mu)d = -\mu\xi a - (1-\mu)\eta b$ parallel.
Da $a$ und $b$ affin unabhängig sind, gilt:

$$\begin{aligned} \frac{\mu\xi}{\lambda} &= \frac{(1-\mu)\eta}{1-\lambda} \\ \mu\xi(1-\lambda) &= (1-\mu)\eta\lambda \\ \mu &= \frac{\eta\lambda}{\xi(1-\lambda)+\eta\lambda} \end{aligned}$$

Da $\lambda$ eine lineare Funktion über $s$ und $\mu$ eine lineare Abbildung von $z$ folgt, dass $z_g(s)$ eine lineare, rationale Abbildung von $s$ im Intervall $[a,b]$ auf $e_1(h)$ ist. Insgesamt ist die Zahl der linear rationalen Teilbereiche dieser Funktionen auf allen Kanten $g \in T$ gleich $O(n)$. Jeder dieser Teilbereiche endet entweder an einem Endpunkt von $e_1(h)$ oder an einem kritischen Sichtbarkeitspunkt auf $e_1(h)$. Die Anzahl dieser Punkte auf $e_1(h)$ ist $O(n)$.
Eine analoge Konstruktion gilt für die Kante $e_2$, wobei der Punkt, der über $e_2$ wandert, mit $q = q(t)$ bezeichnet wird. $q(t,h)$ entspricht den Turmspitzen. Das Gegenstück zur Funktionsschar $z_g(s)$ sei $w_g(t)$.

Sei $C$ das Rechteck $e_1^* \times e_2^*$ in der $st$-Ebene. $e_1^*$ bzw $e_2^*$ sind die X-Projektionen der Kanten $e_1$ respektive $e_2$. Wir teilen $C$ nun durch vertikale und horizontale Linien an den X-Koordinaten der kritischen Sichtbarkeitspunkte auf $e_1(h)$ bzw $e_2(h)$ in $O(n^2)$ Teilrechtecke auf. Sei $g \in T$ eine von $e_1$ und $e_2$ verschiedene Kante. Wir zeichnen die Kurve $\beta_g$ für die Kante $g$ gegeben durch $z_g(s) = w_g(t)$ in $C$ ein. In jedem Teilrechteck von $C$, das $\beta_g$ durchläuft, ist $\beta_g$ ein Hyperbel der Form:

$$\frac{\alpha_1 s + \beta_1}{\gamma_1 s + \delta_1} = \frac{\alpha_2 t + \beta_2}{\gamma_2 t + \delta_2}$$

für geeignete Koeffizienten $\alpha_i, \beta_i, \gamma_i, \delta_i, i = 1,2$. Diese Gleichung können wir in $Ast + Bs + Ct + D = 0$ umformen. Dies stellt eine Hyperbel in der $st$-Ebene dar.
$\beta_g$ ist eine zusammenhängende Kurve, die sowohl $s$- als auch $t$-monoton ist und deren Endpunkte auf $\partial C$ liegen.
Für jeden Punkt $s$ existiert maximal ein Punkt $t$, der $w_g(t) = z_g(s)$ erfüllt, denn da sich $w_g(t)$ monoton bewegt, kann dieser maximal einmal mit $z_g(s)$ übereinstimmen. Analog gilt dies für $t$.
Daraus folgt, dass $\beta_g$ keinen Endpunkt im Inneren von $C$ haben kann und dass $\beta_g$ beim Übergang von einem Teilrechteck zum nächsten zusammenhängend bleibt. Außerdem ist die Anzahl der hyperbolischen Bögen, die die Kurven $\beta_g$ erzeugen für alle $g \in T$ gleich $O(n)$. An einem Endpunkt eines $\beta_g$ liegt entweder an der $s$-Koordinate oder an der $t$-Koordinate auf $e_1(h)$ bzw auf

$e_2(h)$ ein kritischer Sichtbarkeitspunkt vor. Die Anzahl solcher Punkte ist $O(n)$. Desweiteren kann nur ein $\beta_g$ so einen Punkt an der selben $s$- oder $t$-Koordinate haben, nämlich die Kurve $\beta_g$, die zu der Kante $g$ gehört, welche vom Sichtbarkeitsstrahl gekreuzt wird, der dem Sichtbarkeitsereignis auf $e_1(h)$ oder $e_2(h)$ entspricht. Diese Anzahl ist $O(n)$.

Jedes $\beta_g$ teilt $C$ daher in zwei Teile: Ein Teil $MV_g$ bestehend aus allen Punkten $(s,t)$, die Turmplatzierungen auf $e_1$ und $e_2$ entsprechen und so komplett $g$ überblicken. Das Komplement der Menge $MV_g$ enthält genau die Punkte, die Turmplatzierungen entsprechen, von denen aus $g$ nicht komplett zu überwachen ist.
Diese Aufteilung von $C$ läßt sich in vier Typen klassifizieren (Abbildung 15) und hängt von drei Faktoren ab: (i) Die links-rechts-Reihenfolge der Kanten $e_1, e_2, g$, (ii) die Lage des Achsvertices des Sichtbarkeitsstrahls $p(s,h)z_g(s)$ bezüglich der Linie, die das Segment $e_1(h)$ enthält, (iii) die Lage des Achsvertices des Sichtbarkeitsstrahls $q(t,h)w_g(t)$ bezüglich der Linie, die das Segment $e_2(h)$ enthält.

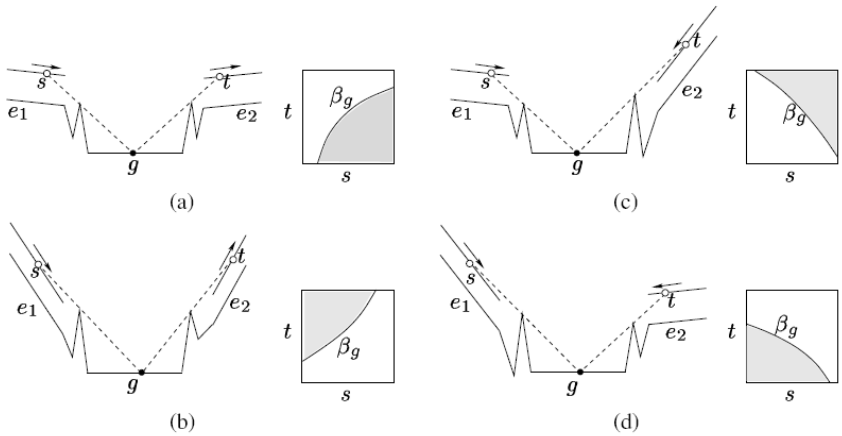


Abbildung 15: Klassifizierung der $MV_g$ in $C$. Der dunkle Teil stellt $MV_g$ dar.

Als nächstes konstruieren wir in linearer Zeit die Kurven $\beta_g$ bzw die Mengen $MV_g$ für alle Kanten $g \in T$, bei festen $e_1$ und $e_2$. Hierzu erstellen wir aus $\sum(e_1, g)$ und $\sum(e_2, g)$ eine gemeinsame Sequenz $\sum(e_1, e_2, g)$. Wir bearbeiten nun die Teilstücke $I$ der Kante $g$, die durch Punkte aus $\sum(e_1, e_2, g)$ begrenzt werden. Jedes Intervall $I$ beschreibt einen Teil von $\beta_g$, in welchem sich der gemeinsam sichtbare Punkt $z_g(s) = w_g(t)$ im Intervall $I$ bewegt. Hier ist zu beachten, dass nur Intervalle $I$, die von beiden Funktionen aus sichtbar sind, zur Konstruktion von $\beta_g$ beitragen. Die Berechnung aller $MV_g$ für feste Kanten $e_1$ und $e_2$ erfolgt in $O(n \log n)$ Zeit.

Als nächstes berechnen wir die Vereinigung $\bigcap_g MV_g$ über allen Kanten $g \in T$. Wenn diese Vereinigung leer ist, so existiert kein Wachturmpaar, welches bei gegebenen Kanten $e_1$, $e_2$ und bei gegebener Höhe $h$ komplett $T$ überwacht. Sonst entsprechen alle Punkte $(s,t) \in \bigcap_g MV_g$ einem Wachturmpaar welches komplett $T$ überblicken kann.
Wir nutzen folgende Eigenschaft von $\bigcap_g MV_g$ zur Berechnung aus: $\bigcap_g MV_g$ liegt zwischen der oberen Kontur der Kurven $\beta_g$, für welche $MV_g$ (in $t$-Richtung) über $\beta_g$ liegt (Abbildung 15 (b,c)), und der unteren Kontur der Kurven $\beta_g$, bei denen $MV_g$ unter $\beta_g$ liegt (Abbildung 15 (a,d)). Da jedes Paar hyperbolischer Bögen der Form $Ast + Bs + Ct + D = 0$ maximal zwei Schnittpunkte hat folgt, dass die Konturen, und somit auch $\bigcap_g MV_g$, die Komplexität $O(\lambda_4(n))$ haben. Wir können $\bigcap_g MV_g$ nun mit dem Algorithmus von Hershberger (siehe [7] und [9]) in $O(\lambda_3(n) \log n)$ Zeit berechnen.

Wir wiederholen diese Berechnung für alle Kantenpaare $e_1, e_2 \in T$ und stoppen, sobald wir ein Paar gefunden haben, welches eine nichtleere Menge $\bigcap_g MV_g$ liefert und geben einen mögliche Positionierung $p(s), q(t)$ aus. Falls alle $\bigcap_g MV_g$ leer sind, so existiert kein Wachturmpaar mit Höhe $h$, das komplett $T$ überwacht.

Die Gesamtkosten der Entscheidungsprozedur betragen also $O(n^3 \alpha(n) \log n)$ mit $\lambda_3(n) \in \theta(n\alpha(n))$, wobei $\alpha(n)$ die Inverse der Ackermann-Funktion ist.

### 3.2 Anwendung der parametric search

In diesem Abschnitt wird nun beschrieben, wie wir mit Hilfe der Entscheidungsprozedur die optimale Höhe $h^*$ durch parametric search bestimmen können. Wie bei der (semi-)diskreten Version des Problems verkleinern wir sukzessive das Intervall, welches die optimale Höhe $h^*$ enthält. Die Ausgabe der optimalen Höhe erfolgt entweder nach einer der Ausführungen der Entscheidungsprozedur oder wir geben am Ende das Minimum des finalen Intervalls aus.

Sei $p_v$ bzw $p_u$ der vom Vertex $v$ bzw $u$ ausgehende, nach oben gerichtete, vertikale Strahl. Wir bilden für alle Kanten $e = uv \in T$ das nach oben geöffnete Trapez $\tau_e$, begrenzt durch die Kante $e$ und durch die Strahlen $p_u$ und $p_v$. Nun bestimmen wir alle in $\tau_e$ liegenden Schnittpunkte $q$ von Sichtbarkeitsstrahlen, die durch zwei Vertices von $T$ verlaufen (siehe [6]). Für alle solche Punkte $q$ berechnen wir den Abstand zur jeweiligen Kante $e$. Die Abstände bilden die Menge der kritischen Höhen. Pro Kante $e$ existieren $O(n^2)$ solcher Abstände. Somit erhalten wir insgesamt $O(n^3)$ kritische Höhen. Zusätzlich fügen wir in diese Menge noch die Höhen $h$ ein, bei denen die Kante $e(h)$ kollinear mit einem Vertex $v \in T$ ist. Dies sind nochmal $O(n^2)$ kritische Höhen, da wir pro Kante $O(n)$ solcher Punkte haben.

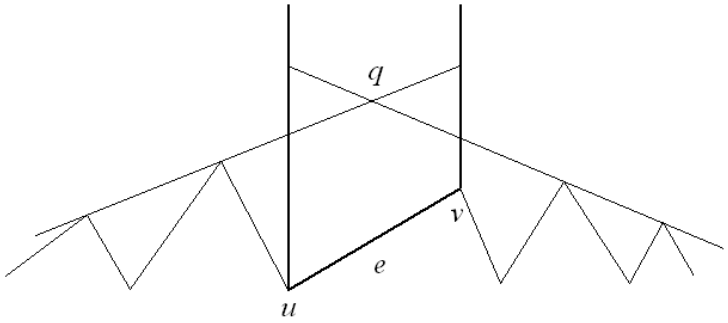


Abbildung 16: Das Trapez $\tau_e$ und ein Schnittpunkt $q$ zweier Sichtbarkeitsstrahlen, die durch zwei Vertices von $T$ gehen.

Wir führen eine binäre Suche mit Hilfe der Entscheidungsprozedur durch die kritischen Höhen durch. Hierfür benötigen wir $O(\log n)$ Schritte und da eine Ausführung der Entscheidungsprozedur $O(n^3\alpha(n)\log n)$ Zeit kostet, ergeben sich für diesen Schritt Gesamtkosten von $O(n^3\alpha(n)\log^2 n)$. Wir erhalten das Intervall $I_0$, welches durch zwei kritische Höhen begrenzt ist und die optimale Höhe $h^*$ enthält.

Die kritischen Ereignisse auf den Kanten $e(h)$ sind nun für alle $h \in I_0$ kombinatorisch bestimmt und deren Abfolge von links nach rechts ist fest. Desweiteren sind die Sichtbarkeitsstrahlen, die von einem Punkt auf $e(h)$ ausgehen, durch einen Achsvertex $v \in T$ gehen und $g$ schneiden, nun auch kombinatorisch bestimmt. Das heißt vor allem, dass alle $\sum(e_1, g)$ und $\sum(e_2, g)$ für alle Kantenpaar nun eindeutig sind. Die Punkte können sich auf $g$ zwar noch bewegen, die Reihenfolge bleibt aber gleich. Wir sammeln alle Schnittpunkte der kritischen Sichtbarkeitsstrahlen ausgehend von allen verschobenen Kanten $e(h)$ mit jeder Kante $g$. Für jedes Kantenpaar $e_1$, $e_2$ vereinen wir die beiden Sequenzen $\sum(e_1, g)$ und $\sum(e_2, g)$ in $\sum(e_1, e_2, g)$ mit Hilfe eines Sortiernetzes der Tiefe $O(\log n)$. Die Gesamtzahl an Prozessoren, die dafür benötigt werden ist $\sum_{e_1,e_2,g}(|\sum(e_1, g)| + |\sum(e_2, g)|) \leq 2n\sum_{e1,g}|\sum(e_1, g)| = O(n^3)$. Mit Cole's Verbesserung der parametric search (siehe [5]) braucht dieser Schritt $O(n^3\alpha(n)\log^2 n)$ Zeit.

Im nächsten Schritt konstruieren wir parallel für jedes Paar $e_1$, $e_2$ die Kurven $\beta_g$ für alle anderen Kanten $g$. Bei gegebenen $e_1$, $e_2$, $g$ bearbeiten wir die Teilintervalle von $g$, die durch Punkte aus $\sum(e_1, e_2, g)$ begrenzt werden. Jedes dieser Intervalle $I$ beschreibt einen Teil von $\beta_g$, in welchem der gemeinsam sichtbare Punkt $z_g(s) = w_g(t)$ auf $g$ in $I$ wandert. Wir berechnen all diese Teilstücke von $\beta_g$ parallel. Da $\sum(e_1, e_2, g) = O(n^3)$, braucht dieser Schritt $O(n^3)$ Zeit und ist nicht mehr von $h$ abhängig.

Wir berechnen nun parallel für jedes Kantenpaar aus $T$ die Region zwischen den oberen und unteren Konturen wie folgt: Wir berechnen die untere Kontur von $m$ Hyperbeln, indem wir parallel den Algorithmus aus [9] anwenden. Wir haben hier $O(\log n)$ parallele Schritte und jeweils müssen zwei Konturesequenzen miteinander verbunden werden. Dies kann in $O(\log n)$ Schritte erledigt werden. Somit haben wir insgesamt $O(\log^2 n)$ parallele Schritte und daraus folgen mit [5] Kosten von $O(n^3\alpha(n)\log^3 n)$. Dieselbe Laufzeit benötigen wir nun noch, um die Regionen zwischen den oberen und unteren Konturen zu berechnen.

**Theorem 5** *Das kontinuierliche Zwei-Wachturm-Problem im* $\mathbb{R}^2$ *mit* $n$ *Ecken kann in* $O(n^3\alpha(n)log^3n)$ *Zeit gelöst werden.*

## Literatur


[1] A. Agarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel computational geometry. *Algorithmica*, 3:293–328, 1988.

[2] P. K. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. Ntafos, M. Sharir, and B. Zhu. Guarding a terrain by two watchtowers.

[3] B. Chazelle and L. Guibas. Visibility and intersection problems in plane geometry. *Discrete Comput. Geom.*, 4:551–581, 1989.

[4] D. Chen and O. Daescu. Maintaining visibility of a polygon with a moving point of view. *Inform. Process. Lett.*, 65:269–275, 1998.

[5] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[6] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[7] J. Hershberger. Finding the upper envelope of n line segments in $o(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.

[8] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[9] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, New York, 1996.