11. Bewegungsplanung für Roboter bei unvollständiger Information

Problem 1: *Ausweg aus einem Labyrinth* (s. Klein 1997)

Unterschied zwischen

- Berechnung der Lösung (Vorauss.: Information über das Labyrinth vollständig bekannt)
- Suche der Lösung (bei unvollständiger Information, z.B. nur vom aktuellen Standort sichtbare Teile des Labyrinths sind bekannt)

Berechnung der Lösung:
benutze *Sichtbarkeitsgraph*

- Knoten = Startpunkt, alle Hindernisecken, Zielpunkt(e) außerhalb des Labyrinths
- 2 Knoten verbunden durch Kante genau dann, wenn sie sich sehen können

Problem dann reduziert auf Finden eines kürzesten Pfades im Sichtbarkeitsgraphen

Lösung durch Algorithmus von Dijkstra (s. z.B. O'Rourke 1998, S. 297 ff.):

gegeben Graph G mit Kantengewichten ("Längen"), Startknoten s, Zielknoten t
initialisiere Subgraph S mit $\{s\}$
while $t \notin S$
{
Suche eine Kante $e \in G \setminus S$ zu einem Knoten x, dessen Abstand (Pfad-Gewicht) zu s minimal ist
$S = S \cup \{e\}$
}

Zeit- und Speicheraufwand $O(n^2)$ (n = Eckenzahl)

Suche nach Lösung (bei unvollständiger Information):

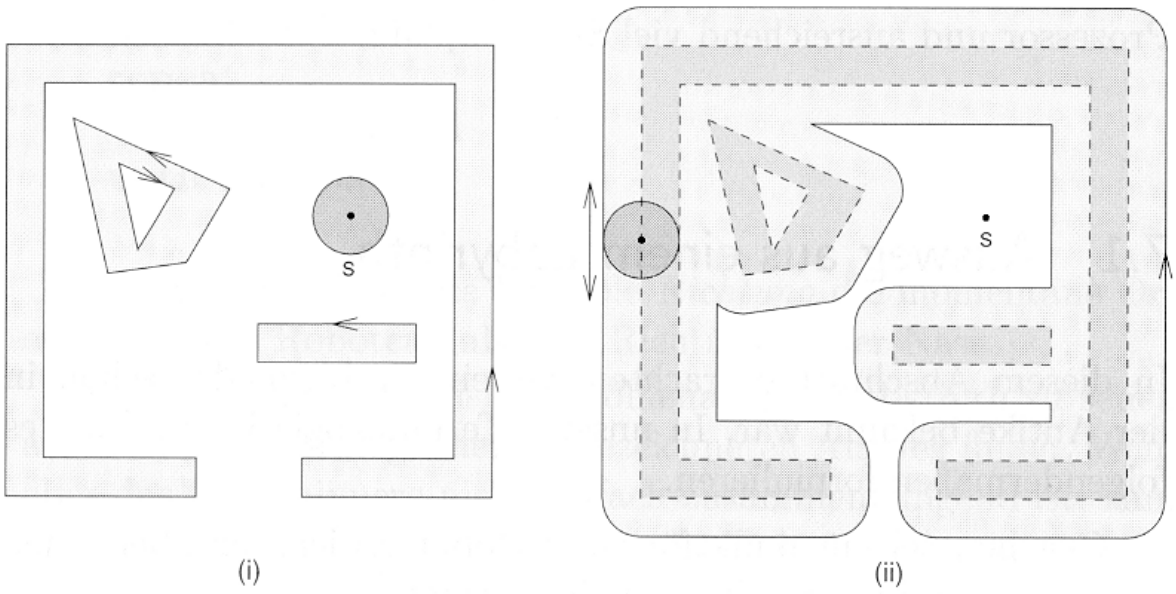
- gefundener Weg wird i. d. Regel sehr viel länger sein als der kürzest mögliche Weg
- kommt durch Versuch und Irrtum zustande
- Effizienzmaß: Länge des Weges, der insgesamt zurückgelegt wird (Rechenzeit dagegen meist vernachlässigbar)
- gibt es überhaupt einen Ausweg? Suchstrategie sollte diesen finden, wenn er existiert!

Labyrinth (bzw. Umgebung des Roboters):
endliche Menge einfacher, geschlossener, paarweise disjunkter polygonaler Ketten (= Ränder der Hindernisse)

Inneres = unzugänglich (Mauerwerk)
Außengebiet = zugänglich
- es kann "Innenhöfe" geben (Abb. (i))
$\Rightarrow$ Ausweg existiert nur, wenn Startpunkt nicht in Innenhof liegt

Idealisierung: punktförmiger Roboter

- im Fall eines kreisförmigen Roboters mit Radius *r* vergrößere alle Hindernisse um *r*; im neuen Labyrinth (ii) genügt ein punktförmiger Roboter



Ausstattung des Roboters:

feste Fahrtrichtung (Richtungsänderung erfordert Rotation; vgl. *turtle geometry*)

Sensoren:

- im einfachsten Fall nur Tastsensor
  - stellt Kontakt mit Hinderniswand fest
  - erlaubt Entlangbewegen an der Wand
- komfortabler: Rundumsicht

Gedächtnis:

- Winkelzähler
  (summiert die Winkel der insges. ausgeführten Drehbewegungen, Linksdrehungen positiv, Rechtsdrehungen negativ; kann auf 0 zurückgesetzt werden)
- komfortabler: Kenntnis der absoluten Koordinaten ("GPS")

für das Labyrinth-Problem:
punktförmiger Roboter, nur mit Tastsensor und Winkelzähler.

Gesucht:

> Strategie, mit der der Roboter aus jedem unbekannten Labyrinth herausfindet, aus dem es überhaupt einen Ausweg gibt.

- Roboter soll Rand der konvexen Hülle der Hindernisse erreichen (oder gleich im Startpunkt stehenbleiben, wenn dieser schon außerhalb der konvexen Hülle liegt)

Annahme: wenn das passiert, wird eine Statusvariable "entkommen" gesetzt und alle Schleifen in den folgenden Algorithmen werden verlassen.

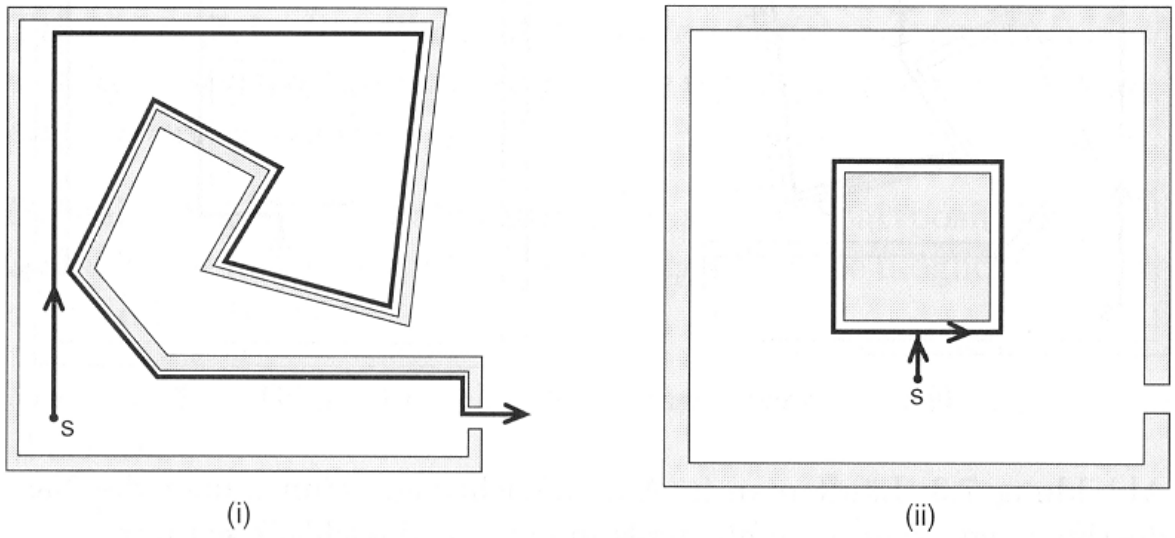
Annahme: bei Wandkontakt dreht sich der Roboter rechtsherum und folgt der Wand so, dass die Wand links von ihm ist

einfachste Strategie:
"*Höhlenforscherstrategie*"
folge immer der ersten Wand, auf die du triffst

genauer:

wähle *Richtung* := beliebig;
repeat
    folge *Richtung*
until *Wandkontakt*;
repeat
    folge der Wand
until *entkommen*

Strategie ist manchmal erfolgreich, manchmal nicht:



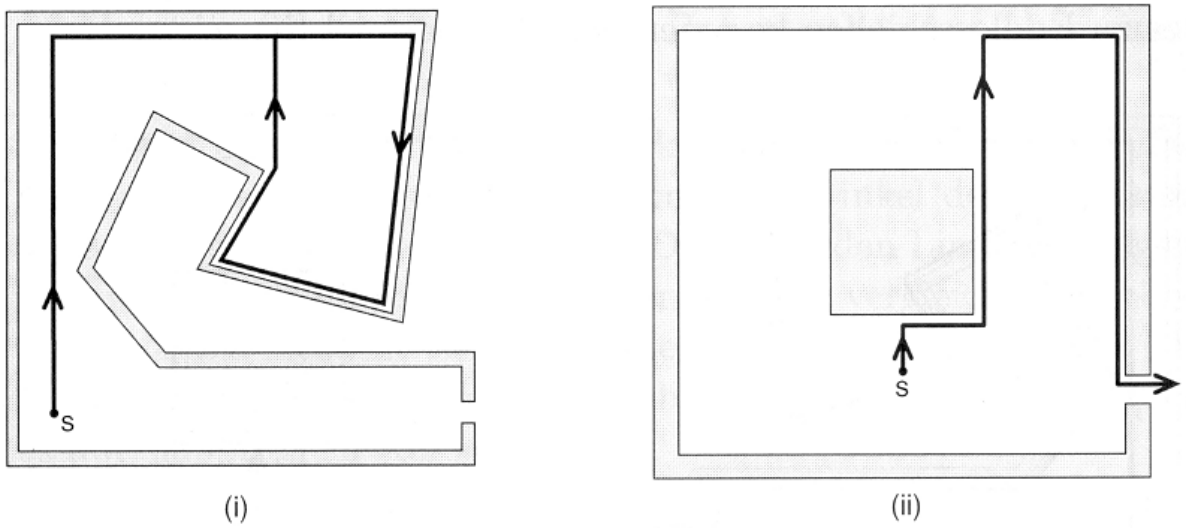
(i) (ii)

⇒ man muss sich von unterwegs angetroffenen Hindernissen auch wieder lösen können!

2. Ansatz: der Wand nur solange folgen, bis wieder Anfangsrichtung erreicht ist – dann wieder der Anfangsrichtung folgen

wähle *Richtung* := beliebig;
Winkelzähler := 0;
repeat
    repeat folge *Richtung* **until** *Wandkontakt*;
    repeat folge der Wand **until** *Winkelzähler* **mod** 2π = 0
until *entkommen*

⇒ jetzt Endlosschleife im ersten Beispiel:



(i) (ii)

bei jedem Umlauf um eine Endlosschleife dreht sich der Roboter einmal um seine Achse ⇒ mittels Winkelzähler erkennbar (aber wir brauchen den Gesamt-Drehwinkel, nicht nur den Winkel mod 2π):

"Pledge-Algorithmus"
(benannt nach John Pledge, der das Verfahren im Alter von 12 Jahren erfunden hat):

wähle *Richtung* := beliebig;
Winkelzähler := 0;
`repeat`
    `repeat` folge *Richtung* **`until`** *Wandkontakt*;
    `repeat` folge der Wand **`until`** *Winkelzähler* = 0
`until` *entkommen*

⇒ diese Strategie leistet in beiden Beispielen (Abb.) das Richtige

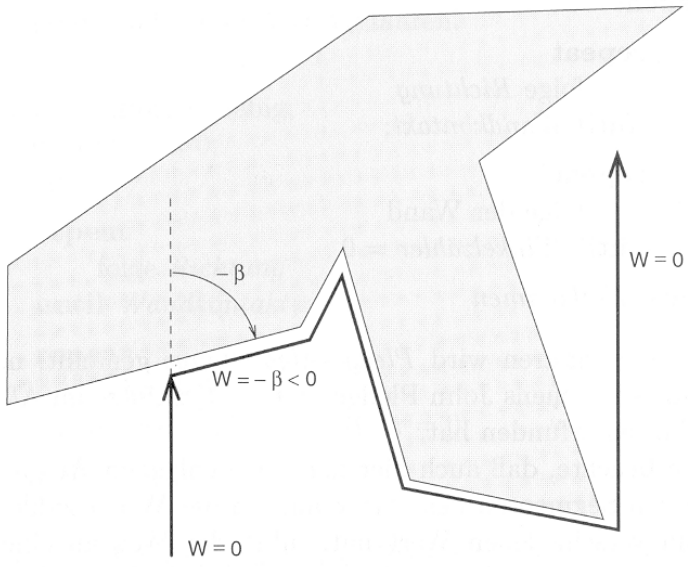
Satz:
Der Pledge-Algorithmus findet in jedem Labyrinth von jeder Startposition aus, von der überhaupt ein Ausweg existiert, einen Weg ins Freie.

Beweis:
Wir zeigen zunächst mehrere Hilfssätze.

Hilfssatz 1: Der Winkelzähler nimmt niemals einen positiven Wert an.
Beweis: Winkelzähler:

- am Anfang = 0
- bei Zurücklegen freier Wegstücke = 0
- bei Auftreffen auf eine Hinderniswand: Rechtsdrehung, um der Wand folgen zu können $\Rightarrow$ Zähler erhält negativen Wert; wenn Zähler wieder 0 wird, löst sich der Roboter vom Hindernis

Drehungen sind stetig $\Rightarrow$ (Zwischenwertsatz) Zähler kann zwischendurch nicht positiv werden.



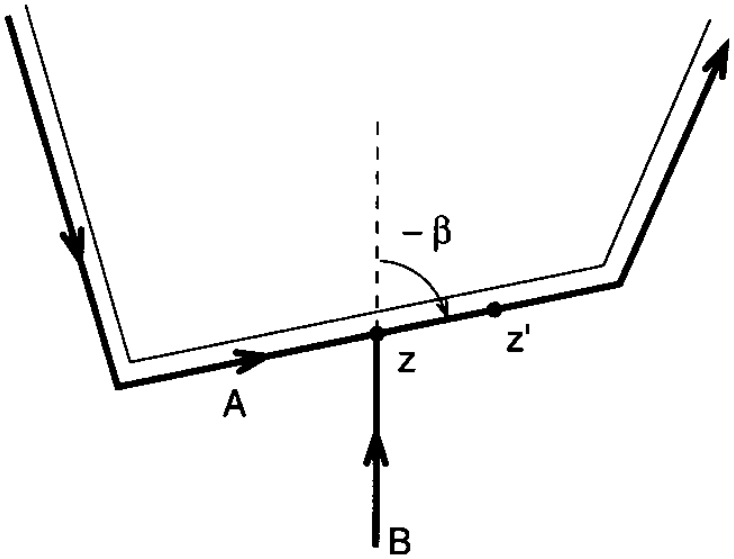
Hilfssatz 2: Wenn der Roboter nicht aus dem Labyrinth herausfindet, besteht sein Weg bis auf ein endliches Anfangsstück (das auch leer sein kann) aus einem geschlossenen Weg, der immer wieder durchlaufen wird.

Beweis: Weg des Roboters ist polygonale Kette; Eckpunkte stammen alle aus der endlichen Menge der Hindernis-Ecken und aus der Menge der Punkte, die von einer Hindernis-Ecke aus in Ausgangsrichtung der erste Auftreffpunkt auf dem nächsten Hindernis sind.
Besucht der Roboter einen solchen Punkt zweimal mit demselben Zählerstand $\Rightarrow$ Weg dazwischen wird endlos iteriert.

Annahme: jeder solche Punkt wird höchstens einmal mit demselben Zählerstand besucht.
$\Rightarrow$ Roboter erreicht nur endlich oft Punkte mit Zählerstand 0
$\Rightarrow$ danach kann er nie wieder eine freie Bewegung machen
$\Rightarrow$ er folgt nur einer einzigen Wand
$\Rightarrow$ Weg wird zyklisch.

Hilfssatz 3: Der Roboter finde nicht aus dem Labyrinth heraus. Es sei *P* der geschlossene Weg, den er nach Hilfssatz 2 schließlich endlos durchläuft. Dann kann sich *P* nicht selbst kreuzen.

Beweis: Annahme, *P* hat doch eine Selbstkreuzung.
Sei *B* die Strecke, die im Punkt *z* auf andere Strecke *A* auftrifft.
Die Wände schneiden sich nicht $\Rightarrow$ eine der beiden Strecken (z.B. *B*) muss frei verlaufen (nicht an einer Wand entlang).



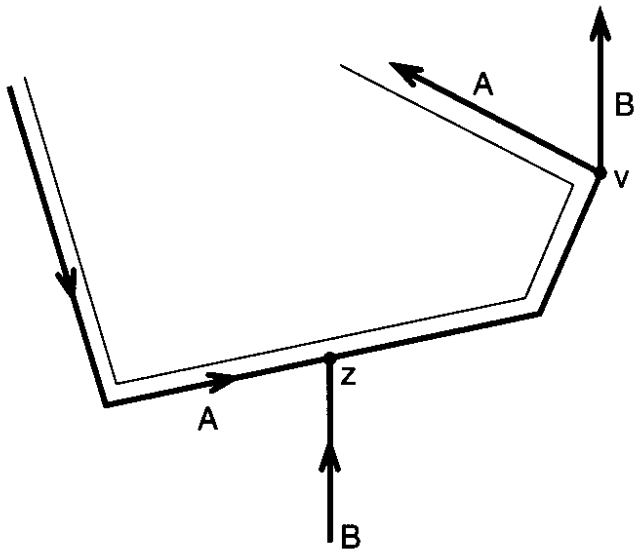
z' Punkt kurz hinter *z*
Anfahrt über *A*: Zählerstand sei $W_A(z')$
Anfahrt über *B*: Zählerstand sei $W_B(z')$

$W_B(z') = -\beta$ mit $0 \leq \beta < \pi$ (da *B* frei)
$W_A(z') = -\beta + k2\pi$ mit ganzzahligem *k*

Ann.: $k \geq 1$
$\Rightarrow W_A(z') = -\beta + k2\pi > -\pi + 2\pi = \pi > 0$, Widerspr. zu Hilfssatz 1.

Ann.: $k = 0$
$\Rightarrow W_A(z') = W_B(z') \Rightarrow$ Wegstücke über A und B trennen sich hinter z nie wieder $\Rightarrow$ höchstens eines der beiden Wegstücke A oder B kann wieder besucht werden
A und B sind aber Teile von P, P wird unendlich oft durchlaufen
$\Rightarrow$ Widerspruch.

Also: $k \leq -1$.
$\Rightarrow W_A(t) < W_B(t)$ für alle Punkte t hinter z' bis zu dem Punkt v, wo sich die Wege wieder trennen
$\Rightarrow$ dort kann nur $W_B(v) = 0$ sein



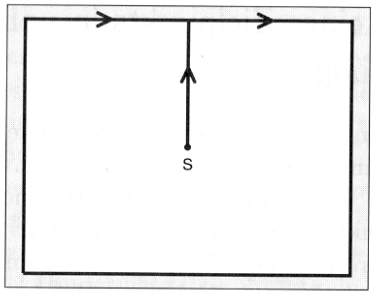
$\Rightarrow$ keine echte Kreuzung (nur "Berührung" der Wege).

Ende des Beweises des Satzes über den Pledge-Algorithmus:

- Weg P lässt sich nach Hilfssatz 3 zu einem einfachen Weg deformieren, ohne die Winkel zu verändern
- P wird entweder im oder gegen den Uhrzeigersinn durchlaufen

wenn P gegen den Uhrzeigersinn durchlaufen wird $\Rightarrow$
Winkelzähler wird bei jedem Durchlauf um 2π erhöht
$\Rightarrow$ wird irgendwann positiv, Widerspr. zu Hilfssatz 1.
Also: P wird *im* Uhrzeigersinn durchlaufen.
$\Rightarrow$ Winkelzähler wird bei jedem Durchlauf um 2π vermindert
$\Rightarrow$ nimmt irgendwann nur noch Werte < 0 an

$\Rightarrow$ *P* enthält kein freies Segment; *P* folgt einer Wand
$\Rightarrow$ wegen des Umlaufsinns kann es sich nur um die Wand eines Innenhofes handeln!

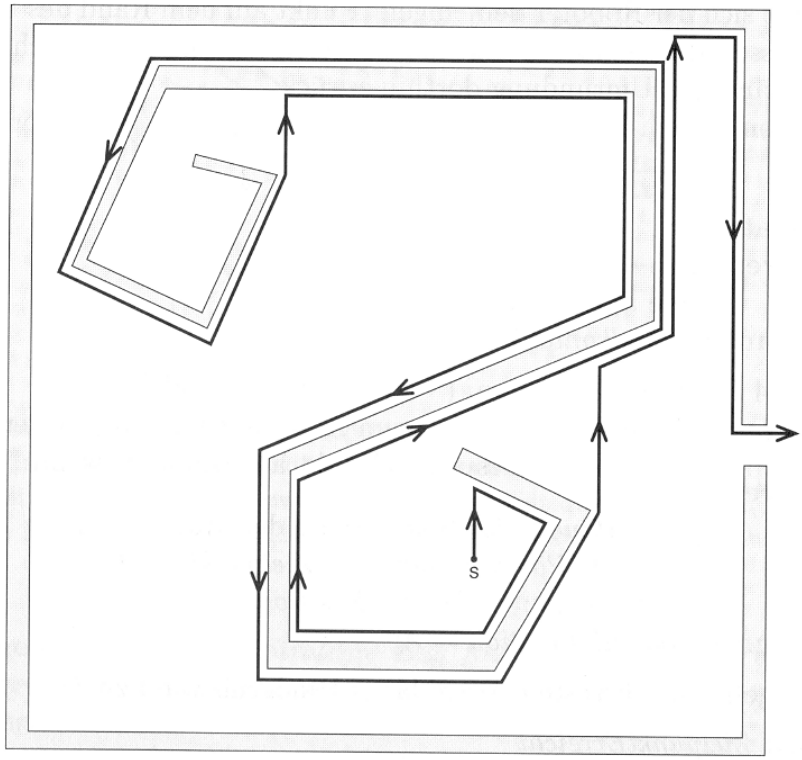


damit gezeigt: wenn der Roboter mit dem Algorithmus nicht aus dem Labyrinth herausfindet, startet er in einem Innenhof, aus dem es keinen Ausgang gibt
$\Rightarrow$ es folgt die Behauptung des Satzes.

Effizienz des Pledge-Algorithmus:
ungünstig!

- gefundener Ausweg kann viel länger sein als der kürzeste Weg ins Freie
- Segmente können darin mehrfach durchlaufen werden

Beispiel:



Problem 2: *Finden eines Zielpunktes in unbekannter Umgebung*

Voraussetzungen jetzt:

- Roboter kann zu jedem Zeitpunkt die eigenen Koordinaten bestimmen
- kann sich einmal besuchte Punkte merken
- zusätzlich Tastsensor, Entlanglaufen an Wänden möglich
- Koordinaten des Zielpunktes sind dem Roboter bekannt!

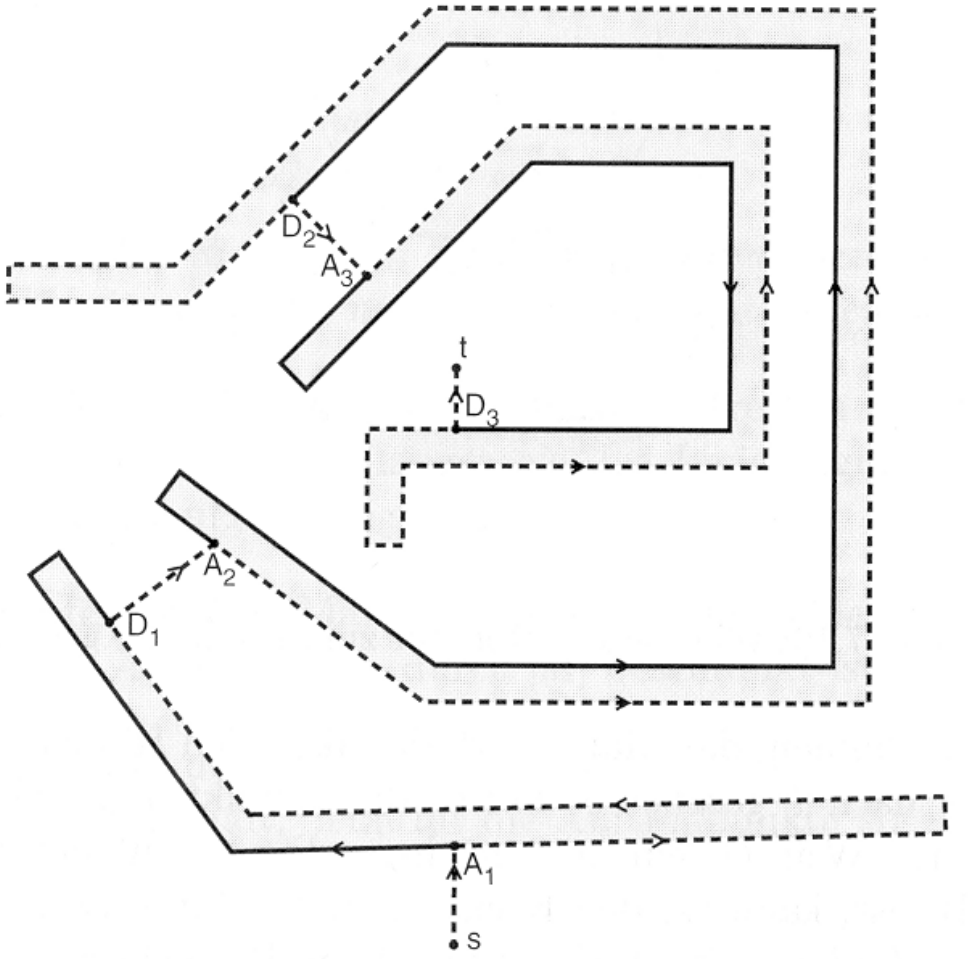
Strategie "*Bug*" (Lumelsky & Stepanov 1987):

- Roboter läuft solange auf den Zielpunkt zu, bis er ein Hindernis erreicht
- dieses wird "untersucht" durch vollständiges Umrunden; der dem Zielpunkt nächste Punkt *D* auf dem Rand des Hindernisses wird gemerkt
- Roboter kehrt zu *D* zurück u. läuft weiter auf den Zielpunkt zu
- iteriere dieses, bis Ziel erreicht.

Als Pseudocode:

```
repeat
    repeat
        laufe auf Zielpunkt zu
    until Wandkontakt;
    A := AktuellePosition; (* auf Hinderniswand *)
    D := AktuellePosition; (* zum Zielpunkt nächster bisher
                              besuchter Punkt auf Hinderniswand *)
    repeat
        rücke AktuellePosition entlang der Wand vor;
        if AktuellePosition näher an Zielpunkt als D
            then D := Aktuelle Position
    until AktuellePosition = A;
    gehe auf kürzestem Wege längs Hinderniswand zu D
until ZielpunktErreicht
```

Beispiel:

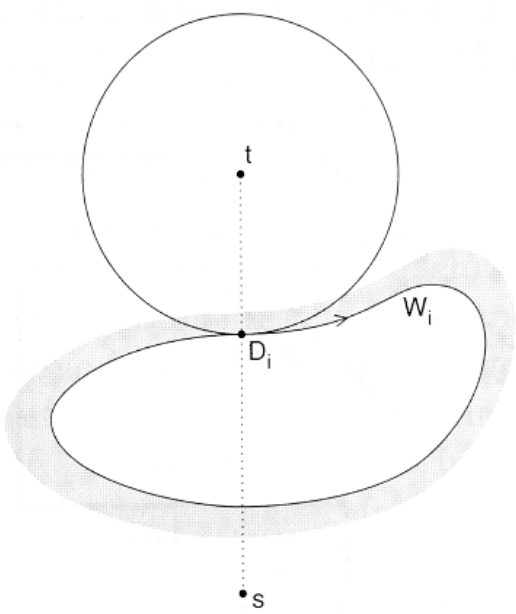


s Startpunkt, *t* Ziel
A_i Punkte, wo Roboter auf Hindernis trifft
D_i Punkte, wo sich Roboter von Hindernis löst

Satz:
Die Strategie "Bug" findet stets einen Weg vom Startpunkt *s* zum Zielpunkt *t*, wenn solch ein Weg überhaupt existiert.

Beweis:
zunächst zu zeigen: der Algorithmus ist durchführbar, d.h. in jedem Punkt D_i ist der Weg in Richtung *t* frei, so dass sich der Roboter dort tatsächlich von der Wand lösen kann.

Ann.: das ist nicht der Fall, d.h. das Mauerwerk (grau) liegt "hinter" D_i (von s aus gesehen):



D_i ist auf dem Hindernisrand W_i dem Punkt t am nächsten
$\Rightarrow$ W_i kann nicht ins Innere des Kreises laufen
W_i kann andererseits s und t nicht trennen (da sonst kein Weg zwischen ihnen existiert)
$\Rightarrow$ W_i kann nur so wie in der Abb. verlaufen
$\Rightarrow$ um von s nach D_i zu gelangen, hätte der Roboter die Wand durchbrechen müssen, Widerspruch.

Korrekte Termination des Algorithmus (t wird erreicht) nun durch streng monotone Abnahme der Abstände:

$$d(A_i, t) \geq d(D_i, t) \underset{\uparrow}{>} d(A_{i+1}, t)$$

da von D_i nach A_{i+1} ein freies Teilstück in Richtung t durchlaufen wird

$\Rightarrow d(D_i, t) > d(D_{i+1}, t)$ für alle i
$\Rightarrow$ keine 2 der D_i können auf demselben Hindernis liegen
$\Rightarrow$ jedes Hindernis wird höchstens einmal besucht

es gibt nur endlich viele Hindernisse $\Rightarrow$ Folge der A_i bzw. D_i ist endlich $\Rightarrow$ t muss nach endlich vielen Schritten des Algorithmus erreicht werden.

Effizienz der Strategie "Bug":

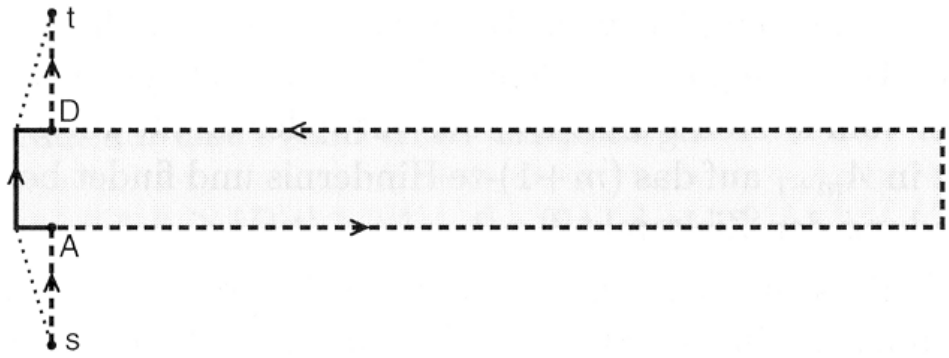
Satz:
Der im Erfolgsfall zurückgelegte Weg ist nicht länger als

$$|st| + \frac{3}{2} \sum_{i=1}^{n} U_i,$$

wobei |*st*| den euklidischen Abstand vom Start- zum Zielpunkt bezeichnet und U_1, ..., U_n die Längen aller Wände sind, auf denen sich Punkte befinden, die näher an *t* liegen als der Startpunkt *s*.

Beweis: s. Klein 1997, S. 327.

Trotz dieser Schranke kann der zurückgelegte Weg beliebig viel länger werden als der kürzeste Weg von *s* nach *t*:



dieses sollte nicht passieren $\Rightarrow$ man sucht nach Strategien, bei denen die Länge des gefundenen Weges auch im schlimmsten Fall nur um einen festen Faktor größer ist als die optimale Weglänge:

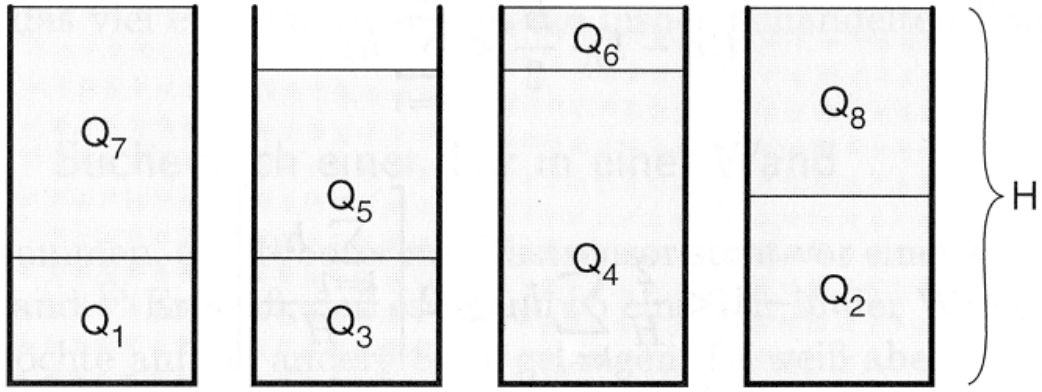
"kompetitive Strategien"

Beispiel "bin packing"-Problem

gegeben: endliche Folge (Q_k) von n 2-dim. Objekten, alle mit gleicher Breite, aber individuellen Höhen h_k.
Außerdem: n leere Behälter (bins), so breit wie die Objekte, mit fester Höhe H ($H \geq h_k$ für alle k).

Ziel: die Objekte in möglichst wenige Behälter packen.

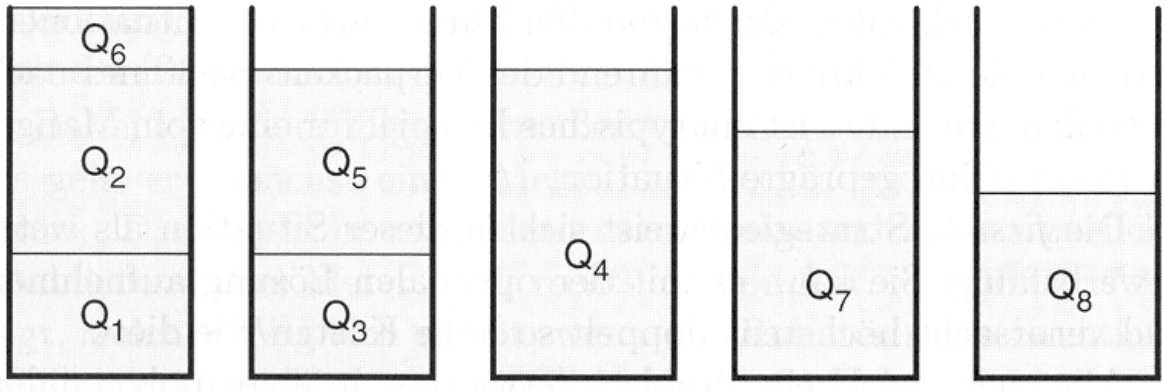
Beispiel für eine optimale Lösung:



Das Problem ist NP-hart
$\Rightarrow$ Optimallösung schwer zu finden; "gute" Heuristiken gesucht

first-fit-Strategie:
Betrachte die Objekte Q_k der Reihe nach und lege jedes in den ersten Behälter von links, in dem zu diesem Zeitpunkt noch Platz ist.

für das obige Beispiel: mit *first fit* 5 statt 4 Behälter



Satz:
Die *first-fit*-Strategie benötigt höchstens doppelt so viele Behälter wie eine optimale Packung.

Beweis:
first fit benötige *m* Behälter für die *n* Objekte.

Es sind alle Behälter bis auf höchstens einen mehr als zur Hälfte gefüllt (wären die Behälter mit den Nummern *i* und *j*, *i* < *j*, nur halb voll, so hätte *first fit* die Objekte aus Behälter *j* schon in Behälter *i* unterbringen können).

$$\Rightarrow \quad (m-1)\cdot\frac{H}{2} < \sum_{k=1}^{n} h_k\,, \quad \text{also } \; m-1 < \frac{2}{H}\sum_{k=1}^{n} h_k \le 2\left\lceil \frac{\sum_{k=1}^{n} h_k}{H} \right\rceil .$$

In der "ceiling"-Klammer: reiner Platzbedarf für die Objekte in Behältereinheiten
$\Rightarrow$ die für optimale Packung benötigte Anzahl m_{opt} ist gleich oder größer als dieser Wert
$\Rightarrow m - 1 < 2\, m_{opt} \qquad \Rightarrow m \le 2 m_{opt}$.

Man sagt:
die Strategie *first fit* ist *kompetitiv* mit Faktor 2.
(Man kann zeigen: *first fit* ist sogar kompetitiv mit Faktor 1,7, aber mit keinem kleineren Faktor; Klein 1997, S. 333.)

Allgemein:
Sei Π ein Problem und *S* eine Strategie, die jedes Beispiel $P \in \Pi$ korrekt löst und dabei die Kosten $K_S(P)$ verursacht.
$K_{opt}(P)$ seien die Kosten einer optimalen Lösung von *P*.
S heißt *kompetitiv* mit Faktor *C*, wenn es eine Zahl *A* gibt, so dass für jedes Beispiel $P \in \Pi$ gilt:

$$K_S(P) \le C \cdot K_{opt}(P) + A.$$

(*A*, *C* $\in$ R; dürfen nur von Π und *S* abhängen, nicht von *P*.)

Problem 3: *Finden einer Tür in einer Wand*

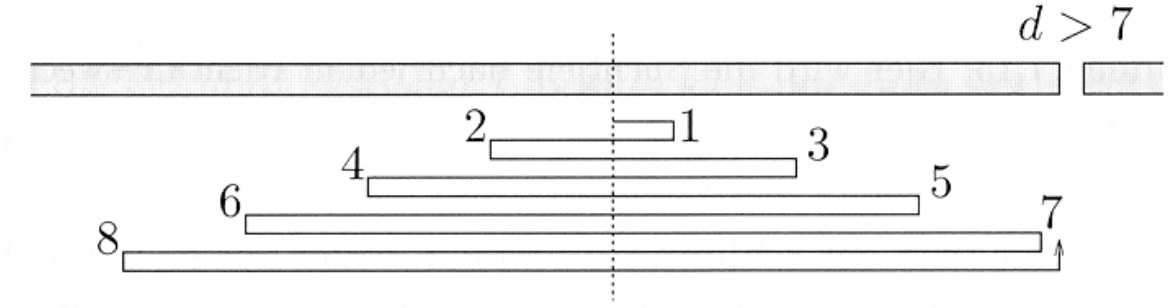
gegeben:

- Roboter mit Tastsensor (kann an Wänden entlanglaufen und Türen identifizieren)
- Wand (sehr lang) mit einer Tür (bzw. Öffnung), Lage der Tür unbekannt

gesucht: Strategie, die den Roboter immer dazu bringt, die Tür zu finden (also nicht: eine Richtung raten und dann endlos in diese Richtung laufen!)

⇒ Roboter muss Suchrichtung irgendwie abwechseln

Strategie: inkrementelles Erhöhen der Suchtiefe



⇒ Strategie korrekt, d.h. Roboter muss die Tür irgendwann erreichen

Annahme: Tür befindet sich bei *d* = l + ε Metern rechts vom Startpunkt *s*, wobei l ganze Zahl und ε > 0 sehr klein

⇒ bei Erkundung des rechten Wandstücks mit Suchtiefe l wird die Tür noch knapp verfehlt

Kürzester Weg zur Tür hätte Länge *d*

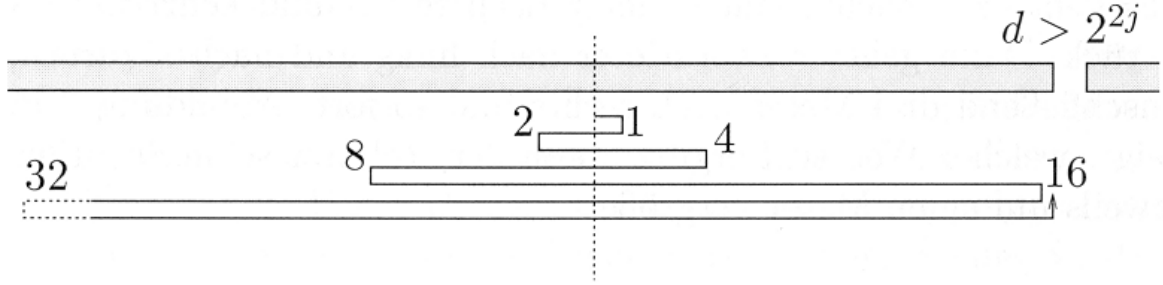
Gesamt-Wegstrecke des Roboters bis zum Finden der Tür:

$$
\begin{aligned}
1+1+2+2+\cdots+l+l+(l+1)+(l+1)+l+\varepsilon & \\
&= 2\sum_{i=1}^{l+1} i + l + \varepsilon \\
&= (l+1)(l+2)+l+\varepsilon \\
&\in \Theta(d^2)
\end{aligned}
$$

(analoge Abschätzung, wenn die Tür links vom Startpunkt)

⇒ Verfahren ist nicht kompetitiv!

Verbesserung:
statt additiver Erhöhung der Suchtiefe diese nach jedem Richtungswechsel verdoppeln



Abschätzung der Gesamt-Weglänge bei knappen Verfehlen der Tür rechts vom Startpunkt, also $d = 2^{2j} + \varepsilon$:

$$
\begin{aligned}
2^0 + 2^0 + 2^1 + 2^1 + \cdots + 2^{2j} + 2^{2j} + 2^{2j+1} + 2^{2j+1} + 2^{2j} + \varepsilon & \\
&= 2 \sum_{i=0}^{2j+1} 2^i + 2^{2j} + \varepsilon \\
&= 2(2^{2j+2} - 1) + 2^{2j} + \varepsilon \\
&= 9 \cdot 2^{2j} - 2 + \varepsilon \\
&< 9d.
\end{aligned}
$$

(Dieselbe Abschätzung gilt, wenn die Tür im Abstand $d = 2^{2j+1} + \varepsilon$ links vom Startpunkt liegt.)

Es folgt:

Die Strategie der abwechselnden Verdopplung der Suchtiefe ist kompetitiv mit Faktor 9.

Baeza-Yates et al. 1993: dieser kompetitive Faktor ist für das Problem optimal, d.h. jede kompetitive Strategie zum Auffinden eines Punktes auf einer Geraden hat einen Faktor ≥ 9.

(Beweis: s. Klein 1997, S. 335 ff.)

Verallgemeinerung des "Tür-Problems":

- *m* Halbgeraden treffen sich in einem gemeinsamen Startpunkt *s*
- eine dieser Halbgeraden enthält den gesuchten Zielpunkt

Verallgemeinerung der Suchstrategie:
man lege unter den Halbgeraden eine Reihenfolge fest und durchsuche sie zyklisch mit den Suchtiefen

$$f_j = \left(\frac{m}{m-1}\right)^j$$

Satz:
Diese Suchstrategie für *m* Halbgeraden ist kompetitiv mit dem Faktor

$$2\frac{m^m}{(m-1)^{m-1}} + 1 \leq 2em + 1;$$

darin ist *e* = 2,718... die Eulersche Zahl.

Beweis:
schlimmster Fall: *j*-ter Suchvorgang verfehlt knapp den Zielpunkt; dessen Abstand von *s* sei $f_j + \varepsilon$
$\Rightarrow$ Runde vergeblichen Suchens, Suchtiefen f_{j+1}, f_{j+2}, ..., f_{j+m-1} , bevor die richtige Halbgerade wieder an die Reihe kommt
$\Rightarrow$ Gesamtlänge des Weges:

$$\begin{aligned} 2\sum_{i=1}^{j+m-1} f_i + f_j + \varepsilon &= 2\sum_{i=1}^{j+m-1}\left(\frac{m}{m-1}\right)^i + f_j + \varepsilon \\ &< 2\frac{\left(\frac{m}{m-1}\right)^{j+m} - 1}{\frac{m}{m-1} - 1} + f_j + \varepsilon \\ &\leq 2(m-1)\left(\frac{m}{m-1}\right)^{j+m} + f_j + \varepsilon. \end{aligned}$$

Somit ist der kompetitive Faktor wie folgt beschränkt (wobei im letzten Schritt benutzt wird, dass die Folge $(1+1/m)^m$ monoton wachsend gegen e konvergiert):

$$\begin{aligned}\frac{2(m-1)\left(\frac{m}{m-1}\right)^{j+m}+f_j+\varepsilon}{f_j+\varepsilon} &\leq 2(m-1)\left(\frac{m}{m-1}\right)^{m}+1\\ &= 2m\left(1+\frac{1}{m-1}\right)^{m-1}+1\\ &\leq 2me+1.\end{aligned}$$

Bemerkungen:

- der hier gefundene kompetitive Faktor (ohne die *e*-Abschätzung) ist für jedes *m* optimal
- wenn die Suchtiefe bei jedem Halbgeradenbesuch verdoppelt wird (wie im Fall $m = 2$) (also $f_j = 2^j$), so ergibt sich der kompetitive Faktor $2^{m+1}+1$; dieser hängt dann also nicht mehr linear, sondern exponentiell von der Anzahl *m* der Halbgeraden ab!
- wenn dagegen die Suchtiefe nur bei Beginn einer neuen Runde verdoppelt wird, innerhalb einer Runde aber konstant bleibt, ergibt sich ein kompetitiver Faktor von $8m–3$ (linear in *m*).

Anwendung dieses verallgemeinerten Suchproblems auf Zielpunktsuche in Polygonen

Problem 4: *Suche nach einem Zielpunkt in einem einfachen Polygon*

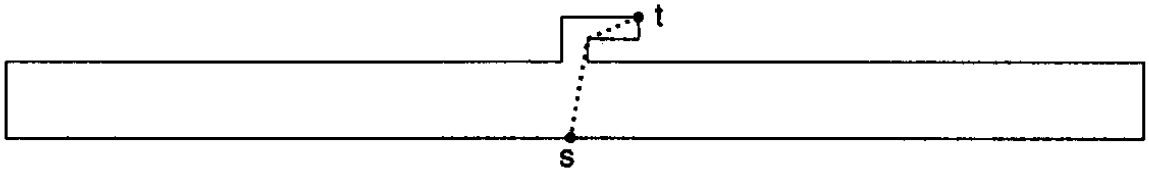
Roboter befindet sich in Startpunkt *s* in einfachem Polygon *P*
Aufgabe: auf möglichst kurzem Weg Zielpunkt *t* erreichen

wenn *s* und *t* beide auf dem Rand von *P* liegen:

mögliche Strategien:

- eine feste Richtung wählen und dem Polygonrand ohne Umkehr folgen, bis *t* erreicht ist
- Rand von *P* abwechselnd nach links und rechts absuchen mit Verdoppelung der Suchtiefe

beide Strategien sind nicht kompetitiv:

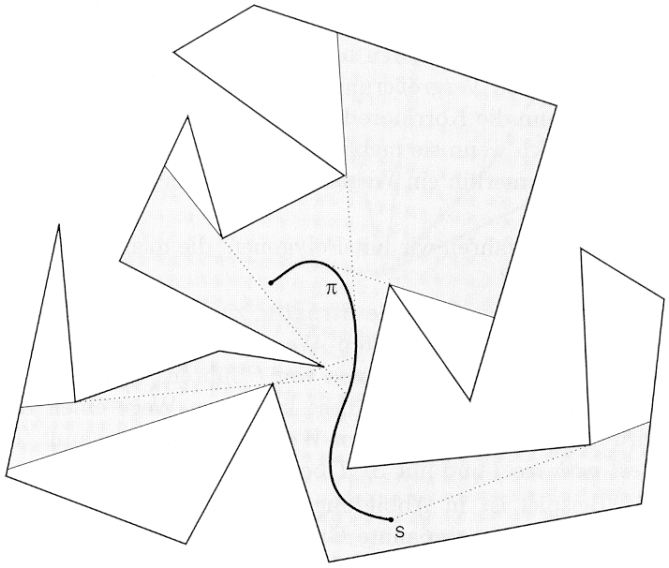


kürzeste Verbindung hier "sehr viel kürzer" als jeder Weg entlang des Randes von *P*

für direktes Ansteuern von *t* bessere Fähigkeiten des Roboters erforderlich

Ausstattung:

- Sichtsystem
- stellt dem Roboter an jeder aktuellen Position *p* in *P* das Sichtbarkeitspolygon *vis*(*p*) zur Verfügung
- Roboter unterhält "partielle Karte" im Gedächtnis, auf der alle Teile von *P* verzeichnet sind, die schon einmal sichtbar waren (Vereinigung der Sichtbarkeitspolygone aller Punkte längs des bereits durchlaufenen Weges)
  Beispiel:

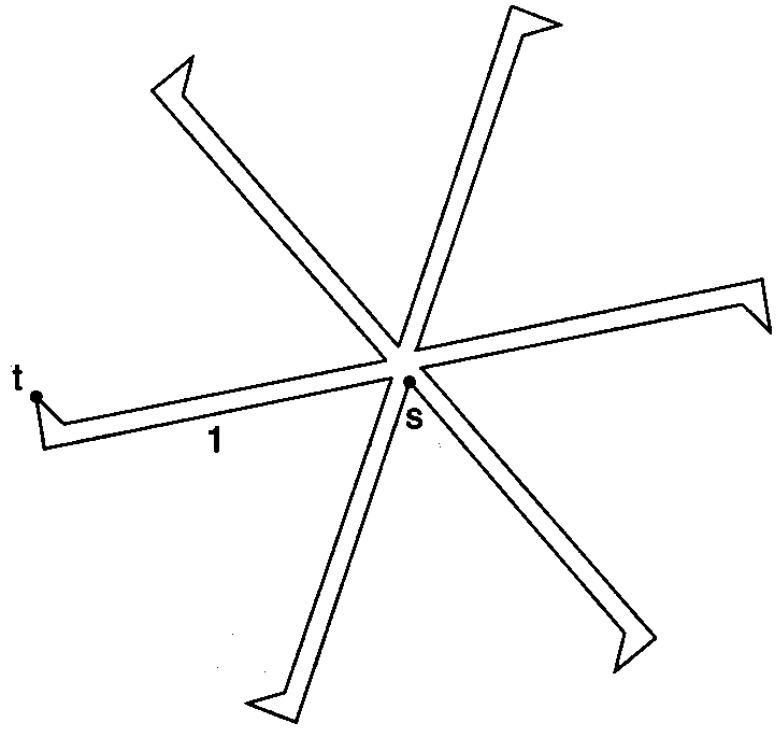


- Zielpunkt *t* ist optisch markiert und wird vom Roboter erkannt, sobald kein Hindernis mehr davor ist

Negatives Ergebnis für Zielpunktsuche, selbst bei dieser "komfortablen" Ausstattung des Roboters:

Satz:
Keine Strategie für die Suche nach einem Zielpunkt kann in beliebigen Polygonen mit *n* Ecken stets einen kleineren Faktor als $n/2 - 1$ für das Weglängenverhältnis garantieren.

Beweis:
betrachte *P* mit *n* Ecken, bestehend aus *n*/4 vielen Korridoren, die beim Startpunkt *s* zusammenlaufen und an ihren Enden abgeknickt sind. Wenn der Zielpunkt *t* am Ende eines der Korridore "versteckt" ist, muss ein Korridor nach dem anderen inspiziert werden.



Annahme: Korridore mit Länge 1, Breiten vernachlässigbar klein, ebenso die Längen der Enden
$\Rightarrow$ Gesamtweg bei der Suche im ungünstigsten Fall:

$$2(n/4 - 1) + 1 = n/2 - 1;$$

kürzester Weg hat Länge 1.

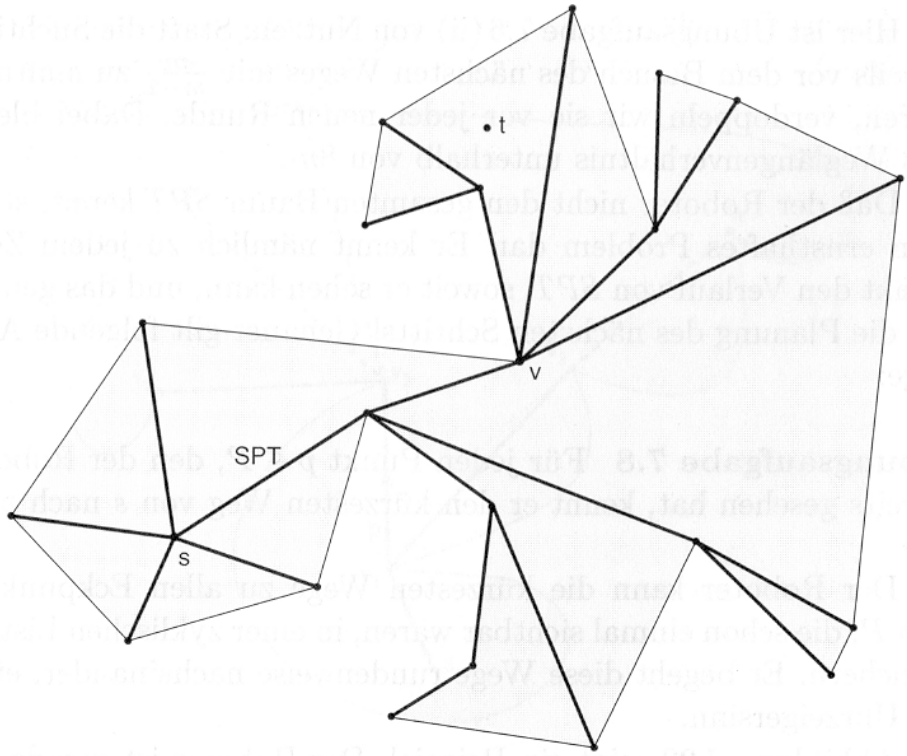
$\Rightarrow$ Weglängenverhältnis $O(n)$ kann in diesem Beispiel immerhin garantiert werden:
verwende Strategie der exponentiellen Vergrößerung der Suchtiefe, wenn *P* aus mehreren in *s* zusammenlaufenden Korridoren besteht.

Verallgemeinerung auf beliebige einfache Polygone:

betrachte die kürzesten Wege von *s* zu den Eckpunkten von *P*

- diese Wege sind aus Geradenstücken zusammengesetzt
- sie bilden einen Baum, den *Baum der kürzesten Wege* von *s* in *P* (shortest path tree; SPT)

Beispiel:



SPT hat i. allg. n+1 Knoten (n: Eckenzahl von P),
darunter $m \leq n$ Blattknoten.

Idee: Roboter kann die m Wege, die im SPT von der Wurzel s zu den Blättern führen, wie m Halbgeraden behandeln und sie zyklisch mit exponentiell wachsender Suchtiefe absuchen nach einem Punkt v, von dem aus der Zielpunkt t sichtbar ist.

Gibt es im SPT solche Punkte v ? Ja!

- Betrachte den kürzesten Weg von s nach t, der innerhalb von P verläuft
- v sei der letzte Knickpunkt auf diesem Weg (siehe Abb. oben)
- dann ist v Eckpunkt von P und damit Knoten des SPT
- von v ist t sichtbar

(es kann noch andere Knoten im SPT geben, von denen t sichtbar ist: der Roboter kann von dort direkt nach t laufen; dieser günstige Fall bleibt in der Aufwandsanalyse unberücksichtigt)

Es sei
$w(s, p)$ die Länge des vom Roboter zurückgelegten Weges von s zu einem Punkt p
$d(s, p)$ die Länge des (in P) kürzesten Weges von s nach p
$|pq|$ der euklidische Abstand von p und q

Weil v auf dem kürzesten Weg von s nach t liegt, gilt:
$d(s, t) = d(s, v) + |vt|$

Für das Weglängenverhältnis folgt
(wegen $d(s, v) \leq w(s, v) \Rightarrow (w(s, v) + |vt|)d(s, v) \leq (d(s, v) + |vt|)w(s, v)$):

$$\frac{w(s,t)}{d(s,t)} = \frac{w(s,v) + |vt|}{d(s,v) + |vt|} \leq \frac{w(s,v)}{d(s,v)} \leq 2em + 1.$$

(die letzte Abschätzung aus dem Satz über die Suchstrategie für m Halbgeraden).

Somit:
kompetitiver Faktor linear in der Anzahl der Blätter des SPT.

Schwierigkeit: Roboter kennt SPT und dessen Blattzahl m i.allg. nicht! Wie soll der Vergrößerungsfaktor $m/(m–1)$ für die Suchtiefe bestimmt werden?

$\Rightarrow$ wähle Strategie der Verdopplung der Suchtiefe vor jeder neuen Runde
$\Rightarrow$ Weglängenverhältnis bleibt dann unterhalb von $8m$.

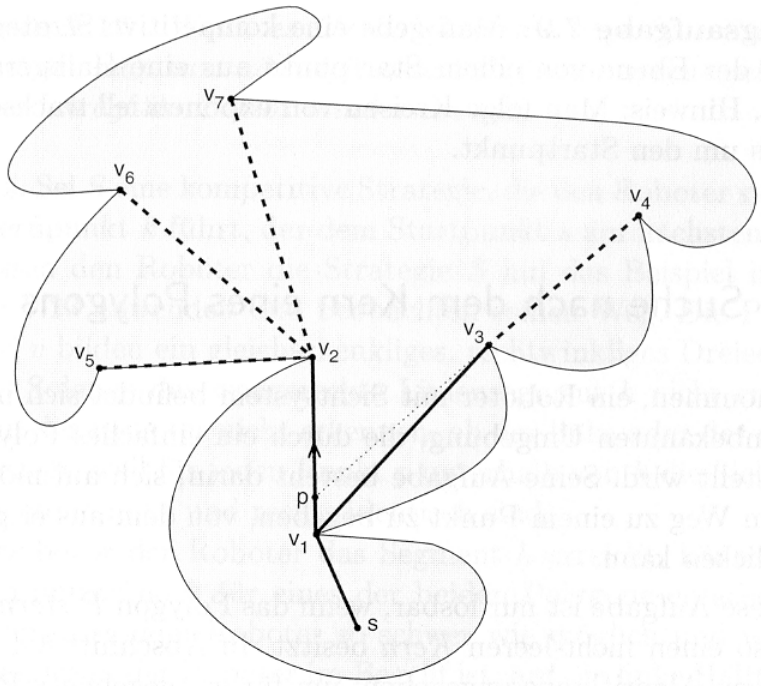
Roboter kennt SPT, soweit er sehen kann:
dies genügt für Planung des nächsten Schrittes!

Genauer:
Für jeden Punkt p in P, den der Roboter bereits gesehen hat, kennt er den kürzesten Weg von s nach p.
Beweis:
p war schon sichtbar $\Rightarrow$ p und s gehören zur aktuellen partiellen Karte P' des Roboters.
Dann enthält P' auch den kürzesten Weg π von s nach p in P:
Annahme: das ist nicht der Fall $\Rightarrow$ π verlässt das Teil-

polygon *P*' über eine Kante *e*, die nicht zum Rand von *P* gehört, und muss über *e* auch wieder in *P*' hineinlaufen
$\Rightarrow$ ersetze diese Schlinge durch ein Stück von *e* und verkürze dadurch π $\Rightarrow$ Widerspruch

Also ist π auch in *P*' der kürzeste Weg von *s* nach *p* und somit dem Roboter bekannt.

Speichere die kürzesten Wege zu allen bereits gesehenen Eckpunkten in einer zyklischen Liste *L* und begehe diese Wege rundenweise nacheinander.



Vereinfachungen:

- nicht jedesmal nach *s* zurücklaufen, sondern bis zur Gabelung der Wege im Baum
- die Kanten zu den Blättern müssen nicht durchlaufen werden, da diese immer schon von Innenknoten des SPT aus sichtbar sind

Ergebnis (schon ohne die Vereinfachungen):
Für Roboter mit Sichtsystem gibt es eine kompetitive Strategie für die Suche nach einem Zielpunkt, die in jedem einfachen Polygon mit *n* Ecken den Weglängenfaktor 8*n* (gegenüber dem kürzesten Weg) garantiert.

(ein Faktor in $O(n)$ ist das Beste, was man erwarten kann)

Problem 5: *Suche nach dem Kern eines Polygons*

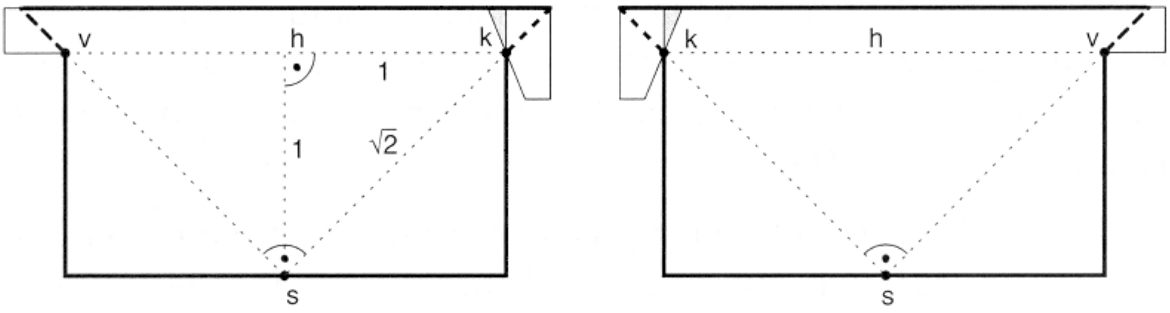
Roboter mit Sichtsystem (wie bei Problem 4) in einfachem Polygon *P*
Aufgabe: Roboter soll sich auf möglichst kurzem Weg zu einem Punkt begeben, von wo er ganz *P* überblicken kann

- nur lösbar, wenn *P* sternförmig, d.h. wenn *P* nichtleeren Kern hat

Aufgabe also: von Startpunkt *s* aus in *P* den nächstgelegenen Punkt *k* von *ker*(*P*) erreichen
(kürzeste Verbindung von *s* nach *k* ist in jedem Fall ein Geradenstück, da *s* von *k* sichtbar ist!)

Obwohl der Roboter schon von *s* aus alle Punkte von *ker*(*P*) sehen kann, hat er dort keine Möglichkeit, sie von den übrigen Punkten von *P* zu unterscheiden!

Beisp.:



aus dem Beispiel ergibt sich untere Schranke für den kompetitiven Faktor:

Satz:
Wenn es überhaupt eine kompetitive Strategie gibt, mit der man zum nächstgelegenen Punkt des Kerns gelangen kann, so beträgt ihr Faktor mindestens $\sqrt{2}$.

Beweis: *S* sei entspr. Strategie; wende *S* auf das obige Bsp. an: unterhalb der Waagerechten *h* ist nicht erkennbar, ob linker oder rechter Fall vorliegt $\Rightarrow$ nach Erreichen von *h* muss Roboter im ungünstigsten Fall immer noch Weg der Länge 1 zum richtigen Endpunkt zurücklegen $\Rightarrow$

Weg von *s* nach *h* hat Länge 1, von dort nochmal Länge 1 = Länge 2, im Vergleich zum euklidischen Abstand (optimaler Weg) $d(s, k) = \sqrt{2} \Rightarrow$

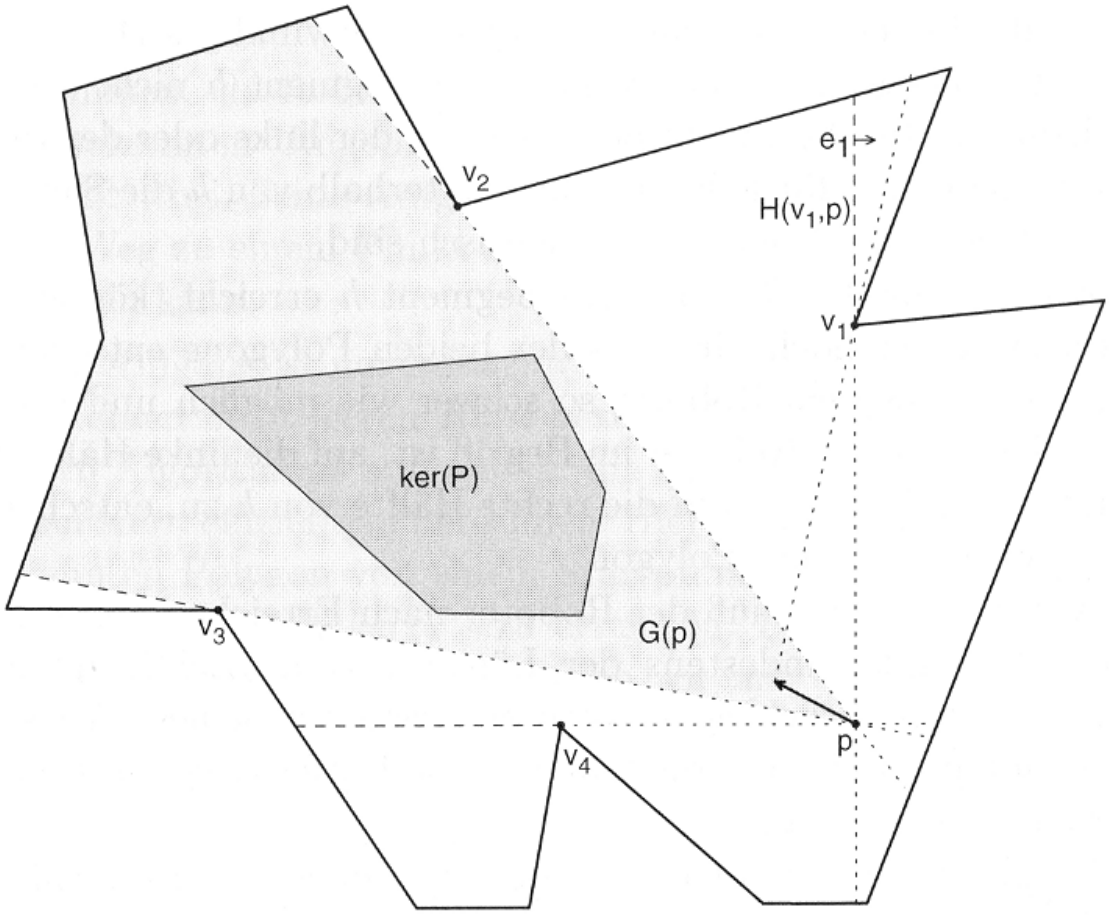
Weglängenverhältnis $\frac{w(s,k)}{d(s,k)} \geq \frac{1+1}{\sqrt{2}} = \sqrt{2}.$

- Bei einer geradlinigen Bewegung von *s* zu einem Punkt im Kern würde das Sichtbarkeitspolygon des Roboters monoton wachsen
- die "richtige" Richtung ist allerdings vorher nicht bekannt

Idee: Roboter soll sich so bewegen, dass das Sichtbarkeitspolygon wächst

Strategie CAB (continuous angular bisector)

- von aktueller Position *p* sind bestimmte spitze Ecken nicht einsehbar
- damit der sichtbare Bereich größer wird, muss der Roboter in die "richtige" Halbebene bzgl. der entspr. "künstlichen Kante" von *vis*(*p*) hineinlaufen



$\Rightarrow$ bewege Roboter in den "Gewinnbereich"

$G(p)$ = Durchschnitt($H(v_i, p)$),

wobei der Durchschnitt über alle spitzen Ecken v_i von P gebildet wird, die eine künstliche Kante e_i von $vis(p)$ erzeugen.

Dieser Durchschnitt ist Winkelbereich in p, der von nicht mehr als 2 Geraden begrenzt wird (s. Abb.: schraffiert)

- die spitzen Ecken, die zusammen mit p diese Geraden def.: "wesentliche Ecken" von $vis(p)$

$\Rightarrow$ Strategie: folge der *Winkelhalbierenden* in diesen Bereich

p (und damit auch die Winkelhalbierende) verändern sich stetig; daher der Name "continuous angular bisector"

Beispiel für Anwendung von CAB:

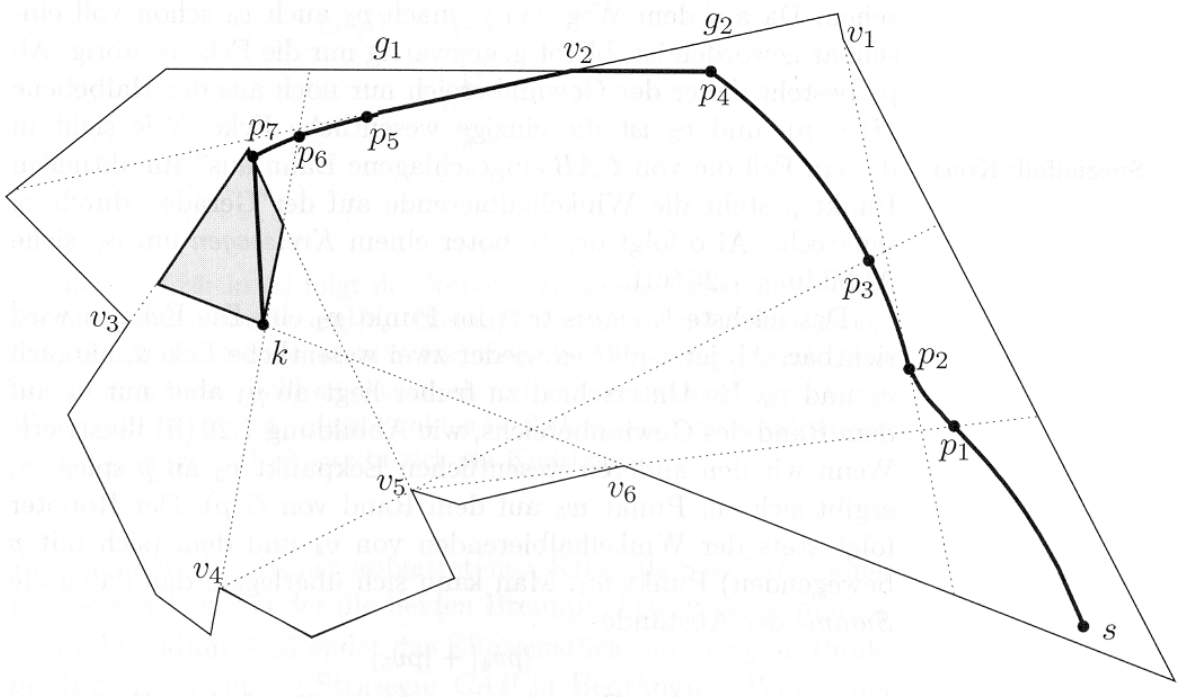


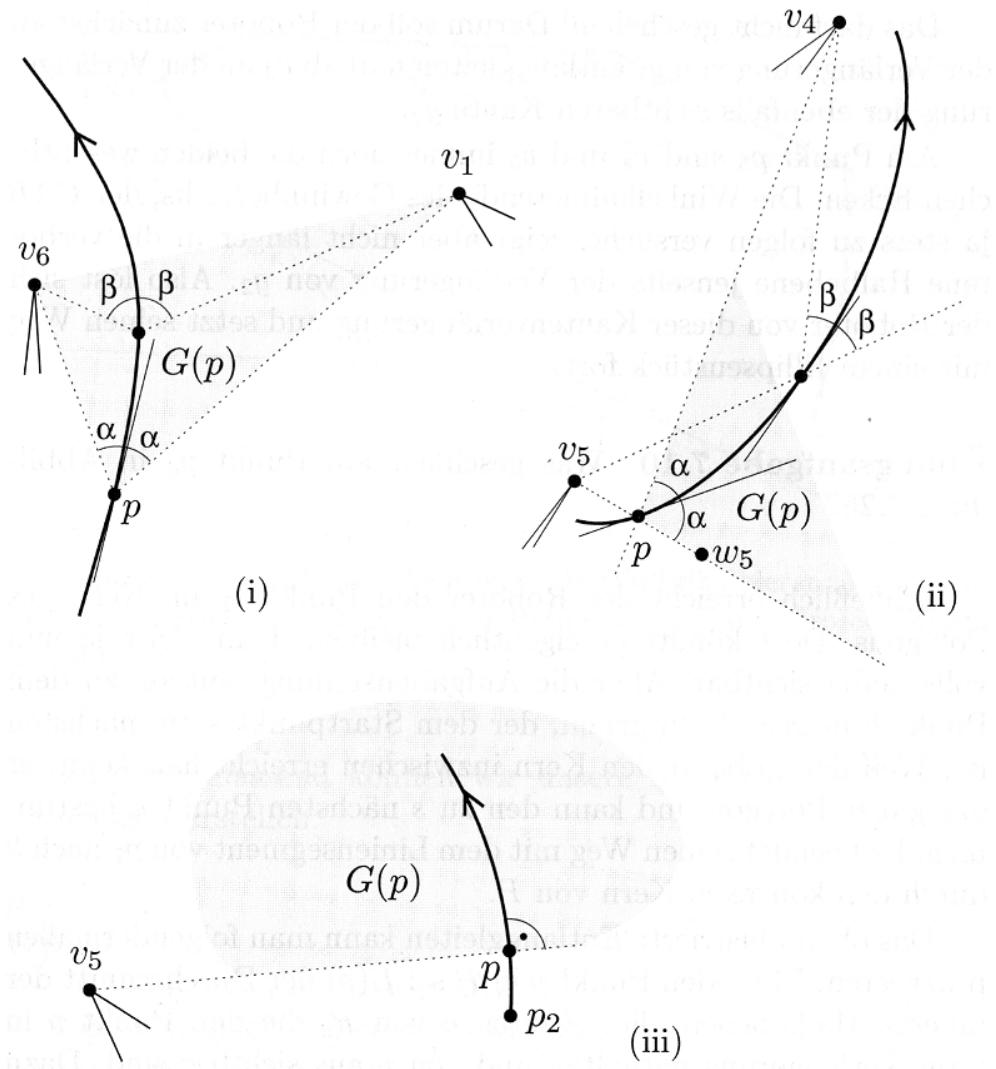
Abb. (*)

von s aus sind nur v_1 und v_6 nicht voll einsehbar (andere spitze Ecken sind in diesem Moment z.T. noch gar nicht bekannt!)

$\Rightarrow$ Roboter folgt Winkelhalb. von pv_1 und pv_6 (p = jeweils aktuelle Position)

bei Bewegung auf der Winkelhalb. verändert sich die Differenz der Abstände von p zu v_1 und v_6 nicht

$\Rightarrow$ diese Bedingung def. einen Hyperbelzweig mit Brennpunkten v_1 und v_6
siehe Abb. (i):



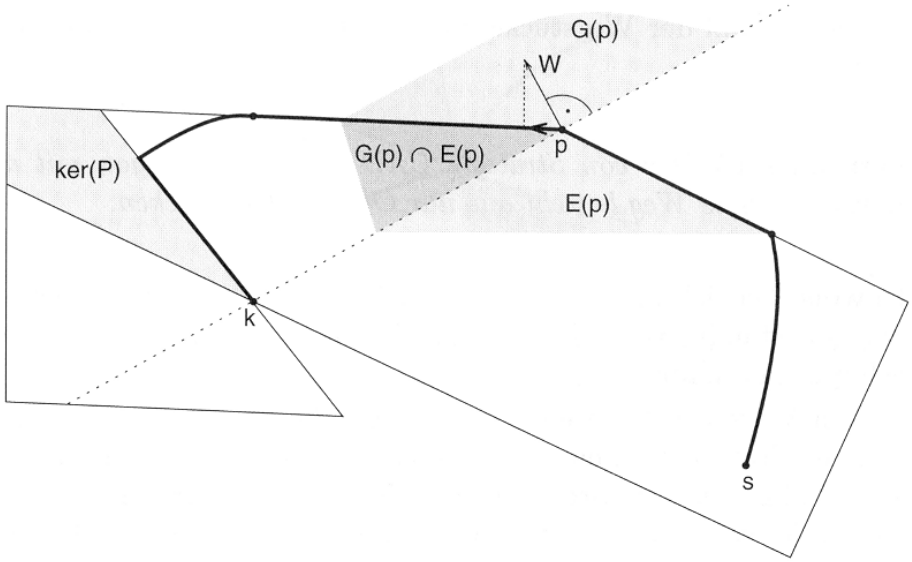
(wenn beide Abstände gleich: Gerade (Bisektor der Ecken))
weiter in Abb. (*):
In Punkt p_1 wird neuer Eckpunkt v_5 entdeckt und löst v_6 als wesentliche Ecke ab $\Rightarrow$ Roboterbahn ist von da an wieder Hyperbelstück, krümmt sich aber zum (näheren) Eckpunkt v_1.

Ab Punkt p_2 kann der Roboter hinter die Ecke v_1 sehen
$\Rightarrow$ nur noch v_5 bleibt jetzt als wesentl. Ecke übrig;
Gewinnbereich ist Halbebene
Bahn des Roboters jetzt: Kreisbogen um v_5
Abb. (iii).

nächstes Ereignis: im Punkt p_3; v_4 wird sichtbar $\Rightarrow$ es gibt wieder 2 wesentl. Ecken (v_4 und v_5), aber nur v_4 liegt auf dem Rand des Gewinnbereiches
Abb. (ii)
w_5 = an p gespiegelter Punkt v_5
Roboter folgt der Winkelhalb. zwischen den Halbgeraden zu v_4 und w_5
dabei bleibt Summe der Abstände zu v_4 und v_5 konst.
$\Rightarrow$ Roboter folgt einem Ellipsenbogen

weiterer Fall, der noch auftritt: ab p_4 würde bei Weiterfolgen der Ellipse Sicht auf Kante g_1 verlorengehen
$\Rightarrow$ der Verlängerung dieser Kante folgen (= Geradenstück)

- allgemein:
  "Haltebereich" $E(p)$, der nicht verlassen werden darf
  $E(p)$ = Durchschnitt der inneren Halbebenen aller Kanten von P, die p in ihrer Verlängerung enthalten und von p aus sichtbar sind
  wenn die Winkelhalb. von $G(p)$ nicht in $E(p)$ enthalten ist, gehe am Rand von $E(p)$ entlang in Richtung der Projektion der Winkelhalbierenden:



weiter in Abb. (*):
Strategie weiter folgen, bis Punkt p_7 im Kern des Polygons erreicht
nun ist P vollständig sichtbar
$\Rightarrow$ Roboter kann Punkt k bestimmen, der im Kern liegt und s am nächsten liegt, und sich geradlinig dorthin bewegen.

Zusammenfassung der Strategie CAB:

p := s;
repeat
    berechne Gewinnbereich $G(p)$;
    W := Winkelhalbierende von $G(p)$;
    berechne Haltebereich $E(p)$;
    if $W \subseteq E(p)$
    then folge W
    else folge der Projektion von W auf den Rand von $E(p)$
until $vis(p) = P$; *(Kern erreicht)*
gehe direkt zum Punkt $k \in ker(P)$, der s am nächsten ist.

Abschätzung des Aufwands:

Satz:
Der von Strategie CAB in einem Polygon mit n Ecken erzeugte Weg besteht aus nur $O(n)$ vielen Stücken.

Beweis:
Sichtbarkeitspolygon vergrößert sich $\Rightarrow$ Bahn enthält keine Schlingen.
Wann endet ein Wegstück?

- wenn eine neue wesentl. Ecke entdeckt wird (kann für jede Ecke nur einmal geschehen $\Rightarrow O(n)$)
- wenn eine schon entdeckte Ecke vollständig sichtbar wird (dito)
- wenn der Roboter auf den Rand des Haltebereichs (Kante von P oder Kantenverlängerung) stößt: beim Lösen vom Rand folgt er konvexer Bahn $\Rightarrow$ dieselbe Kantenverlängerung kann nur dann nochmal getroffen werden, wenn sich zwischendurch die Bahnkurve wieder ändert, also wenn sich eine wesentl. Ecke ändert $\Rightarrow O(n)$ mal
- wenn der Kern erreicht wird (1 mal)
- wenn der Roboter danach zum Punkt k geht (1 mal).

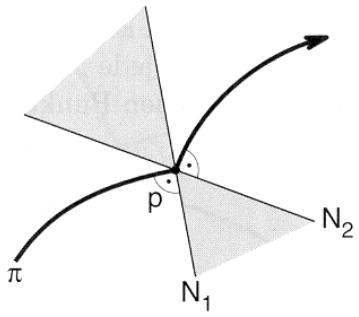
Längenberechnung des von CAB gefundenen Weges:

scheinbar schwierig, da u.a. Ellipsenstücke enthalten
$\Rightarrow$ Integral für die Weglänge hat keine geschlossene Darstellung.

Für eine Abschätzung der Weglänge def. wir eine spezielle Eigenschaft von Wegen, die von allen von CAB gefundenen Wegen erfüllt ist.

Def. "Normale" zu einem Weg in einem Punkt *p* auf dem Weg:

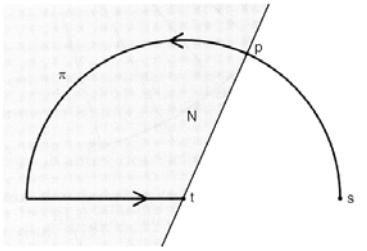
- in einem glatten (d.h. differenzierbaren) Stück: Gerade senkrecht zur Tangente in *p*
- in einem Punkt *p*, wo 2 glatte Stücke zusammenstoßen: jedes Element des Geradenbündels, das von den beiden einseitigen Normalen begrenzt wird:



Def.:
Ein stückweise glatter, von *s* nach *t* orientierter Weg heißt *selbstnähernd*, wenn für jeden Punkt *p* und jede Normale *N* in *p* gilt: Das restliche Wegstück von *p* bis *t* liegt ganz in der abgeschlossenen Halbebene "vor" der Normalen *N* (d.h. in der Halbebene, die der Roboter im Punkt *p* vor sich sieht).

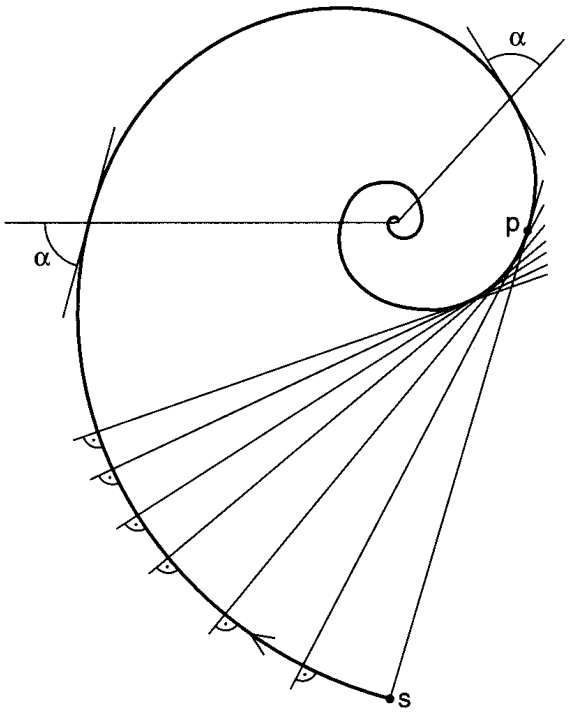
Ein selbstnähernder Weg kommt in jedem Punkt *p* all seinen "zukünftigen" Punkten näher (sind *a*, *b*, *c* drei aufeinanderfolgende Punkte des Weges, so liegt *b* näher an *c* als *a*).

Beispiel:



Weg selbstnähernd; bei umgekehrter Orientierung nicht!

Beispiel: logarithmische Spirale mit Winkel α = 74,66...°:
jede Normale ist gleichzeitig Tangente an einen später durchlaufenen Punkt $\Rightarrow$ Weg (nach innen) "gerade noch" selbstnähernd!

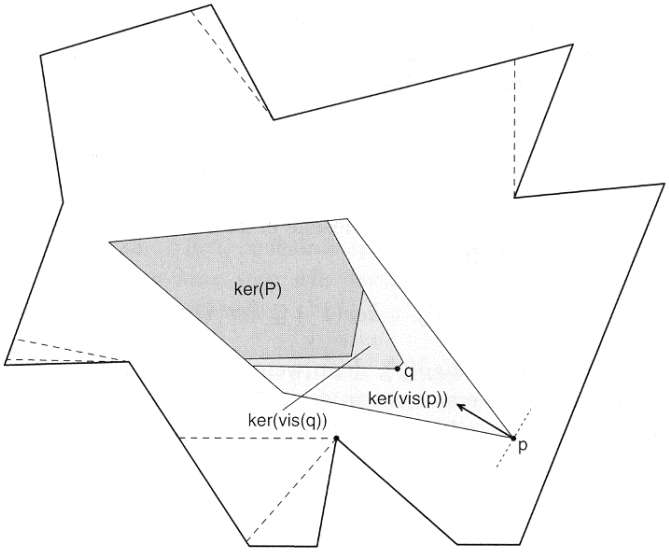


Satz:
Jeder von der Strategie CAB erzeugte Weg ist selbstnähernd.

Beweis: s. Klein 1997, S. 360 ff.

Im Beweis werden die folgenden Eigenschaften der von CAB erzeugten Wege verwendet (Beweise s. Klein 1997):

- wenn *p* den Weg durchläuft, dann wächst das Sichtbarkeits-polygon *vis*(*p*), während *ker*(*vis*(*p*)) schrumpft



- für jeden Punkt *p* auf dem Weg des Roboters ist der Rest des Weges ab *p* ganz in *ker*(*vis*(*p*)) enthalten.

Zur Abschätzung der Weglänge können nun folgende Sätze dienen (Icking & Klein 1995):

- Ein selbstnähernder Weg ist höchstens so lang wie der Umfang seiner konvexen Hülle
- Ein selbstnähernder Weg kann höchstens 5,3331... mal so lang sein wie der Abstand seiner Endpunkte. Diese Schranke ist scharf.

Folgerung:
Die Strategie CAB zum Finden des nächstgelegenen Punktes im Kern ist kompetitiv mit Faktor 5,3331... .

(Es ist nicht gesagt, dass diese Schranke scharf ist, denn nicht jeder selbstnähernde Weg kann bei CAB vorkommen.)