Kompetitive Analyse für Online-Algorithmen — Eine kommentierte Bibliographie

Teil II

Prof. Thomas Ottmann Dr. Sven Schuierer Christoph A. Hipke

Institut für Informatik
Albert-Ludwigs-Universität Freiburg

August 1994

Inhaltsverzeichnis

Einleitung

Dies ist der zweite Teil der Übersicht über Online-Algorithmen und die kompetitive Analyse, der die Kapitel über geometrische Algorithmen und Graphenalgorithmen, sowie über Online-Algorithmen mit Voraussicht enthält. Aus dem ersten Teil sollte nur die Definition der Kompetitivität bekannt sein, die wir im folgenden kurz wiederholen. Daher kann dieser Teil auch unabhängig vom ersten Teil gelesen werden.

Ein Online-Algorithmus beantwortet eine Folge σ von Anfragen, wobei jede Anfrage beantwortet werden muß, bevor die folgende Anfrage erscheint. Die Art der Anfragen und Antworten sind von Problem zu Problem verschieden, gemeinsam ist jedoch allen Online-Problemen, daß der Algorithmus seine Entscheidungen ohne Kenntnis der Zukunft treffen muß und damit im allgemeinen nicht die optimale Lösung für das Problem erhält. Eine Kostenfunktion C_A ordnet dem Online-Algorithmus A Kosten entsprechend seinen Antworten zu. Diese Kosten sind abhängig von der Anfragefolge σ.

Ein deterministischer Algorithmus A heißt *c-kompetitiv*, wenn

$$C_A(\sigma) \leq c \cdot C_S(\sigma) + a$$

für alle Anfragefolgen σ und alle Offline-Algorithmen S ist. Die additive Konstante a darf dabei nur von der Anfangskonfiguration des Problems abhängen.

Die Definition kann auf randomisierte Algorithmen in unterschiedlicher Art und Weise übertragen werden. Die genauen Definitionen können Kapitel 2 im ersten Teil entnommen werden.

Kapitel 4

Graphentheoretische Online-Probleme und Routenplanung

Dieses Kapitel enthält Ergebnisse zu Online-Problemen, die einen graphentheoretischen oder geometrischen Hintergrund haben. Einen Schwerpunkt werden dabei Suchstrategien in geometrisch modellierten Szenen bilden.

4.1 Graph Coloring

Die Farbe ist der Ort, wo unser Gehirn und das Weltall sich begegnen. Darum erscheint sie den wahren Malern durchaus dramatisch.
— Paul Cézanne *Über die Kunst*

Auf Landkarten sind oftmals die einzelnen verzeichneten Länder oder Regionen durch unterschiedliche Farben voneinander abgesetzt, d.h. Länder, die eine Grenze gemeinsam haben, erhalten verschiedene Farben. Hier wurde bereits früh die Fragestellung betrachtet, welches die minimale Anzahl von Farben ist, um eine beliebige Landkarte zu färben. Daß tatsächlich vier Farben ausreichen, wurde erst kürzlich bewiesen.

Stellt man die Länder durch Knoten dar, die von Kanten, die den gemeinsamen Grenzen entsprechen, verbunden werden, so erhält man einen planaren Graph G. Das *Graph Coloring Problem* besteht nun darin, jedem Knoten eine natürliche Zahl, die Farbe, zuzuordnen, so daß keine zwei benachbarten Knoten die gleiche Farbe haben. Die Anzahl der verwendeten Farben soll minimiert werden. Die *chromatische Zahl* $\chi(G)$ bezeichnet die mindestens benötigte Anzahl von Farben zur Färbung von G. Es wurde gezeigt, daß die Bestimmung von $\chi(G)$ ein NP-vollständiges Problem ist.

Das Graph Coloring Problem kann auf einige praktische Problemstellungen wie Prozessorzuweisung und Registeroptimierung angewandt werden [Pat85].

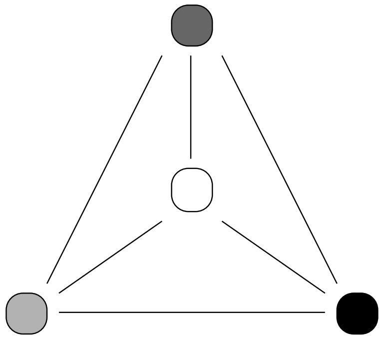

Abbildung 4.1: Ein nicht mit drei Farben färbbarer planarer Graph.

4.1.1 Das Online-Problem

Hier soll der allgemeinere Fall betrachtet werden, daß G nicht planar sein muß. In diesem Fall kann $\chi(G)$ bis zur Anzahl n der Knoten des Graphen wachsen. Es werden *Online*-Algorithmen für das Graph Coloring Problem gesucht. Die Knoten des Graphen G erscheinen einzeln, und nur die Kanten zu bereits erschienenen Knoten werden angegeben. Der Algorithmus muß jedem Knoten eine Farbe zuordnen, bevor der nächste Knoten erscheint. Diese Farbe darf dann nicht mehr verändert werden.

Die Kompetitivität c_A eines Online-Algorithmus A für eine gegebene Klasse Γ von Graphen ist dann

$$c_A = \max_{G \in \Gamma} \frac{A(G)}{\chi(G)},$$

wobei $A(G)$ die Anzahl der von A auf G benötigten Farben bezeichnet. Im allgemeinen Fall ist Γ die Klasse der symmetrischen, nicht-reflexiven, endlichen Graphen; schränkt man diese Klasse ein, erhält man jedoch in vielen Fällen auch verbesserte Aussagen über die Kompetitivität. Im Artikel von Bean [Bea76] findet sich eine noch allgemeinere Betrachtung der Färbung von nur lokal-endlichen Graphen.

4.1.2 Untere Schranken

Für das Graph Coloring Problem im Fall allgemeiner Graphen gab Szegedy [Sze] eine untere Schranke von $n/\log^2 n$ für den kompetitiven Faktor an, indem er zeigte, daß für jeden Online-Algorithmus A und jede Zahl k ein Graph H mit höchstens $k(2^k-1)$ Knoten und chromatischer Zahl k existiert, für den $A(G) \geq 2^k - 1$ ist.

Für Klassen von Graphen mit beschränkter chromatischer Zahl $\chi(G) \leq \chi$ ist das Problem komplizierter. Bean [Bea76] gibt eine Klasse von Bäumen mit n Knoten und $\chi(G) \leq \chi$ an, für die jeder Online-Algorithmus mindestens $\log n$ Farben benötigt. Szegedy und Vishwanathan gaben eine weitere Schranke von $\Omega((\log n/\log\log n)^{\chi/2})$ an und verbesserten diese nochmals auf $\Omega(\frac{1}{\chi-1}(\frac{\log n}{12(\chi+1)} - 1)^{\chi-1})$ [Vis92]. Diese letzte Schranke gilt auch für randomisierte Online-Algorithmen.

4.1.3 Algorithmen für allgemeine Graphen

Benutzt der Online-Algorithmus A für jeden Knoten eine neue Farbe, erhält man die triviale obere Schranke $c_A = O(n)$, wobei n die Anzahl der Knoten des Graphen ist. Das Ziel ist es somit, einen Algorithmus mit sublinearer Kompetitivität zu finden. Der beste bekannte deterministische Algorithmus dieser Art stammt von Lovász, Saks und Trotter [LST88]. Er erreicht eine Kompetitivität von $O(n/\log^* n)$. Schränkt man die Klasse Γ auf Graphen G mit $\chi(G) \leq 3$ ein, so ergibt sich sogar eine Kompetitivität von $O(n \log\log\log n/\log\log n)$.

Ein weitere Verbesserung ergibt sich, wenn man Randomisierung für den Online-Algorithmus zuläßt. Vishwanathan [Vis92] gibt einen solchen Algorithmus an, der eine Kompetitivität von $O(\chi 2^\chi n^{(\chi-2)/(\chi-1)}(\log n)^{1/(\chi-1)})$ erreicht. Für den Spezialfall $\chi = 3$ ergibt sich hier $O((n \log n)^{1/2})$.

Der Algorithmus beruht auf einer teilweisen Benutzung des Greedy-Algorithmus, der immer die kleinste mögliche Farbe vergibt. Eine *partielle Greedy s-Färbung* von G bedeutet, daß der Graph zuerst vom Greedy-Algorithmus mit s Farben $c_1, \ldots, c_s$ gefärbt wird, wobei Knoten, für die diese Anzahl nicht ausreicht, noch frei gelassen werden. Die Menge dieser Restknoten R wird dann gesondert behandelt. Zunächst wählt man eine Greedy-Farbe $c_i \in c_1, \ldots, c_s$ aus. Jeder Knoten in R muß nun zu mindestens einem Knoten der bereits mit c_i gefärbten Knoten $u_1, \ldots, u_k$ verbunden sein. Dabei sind diese Knoten u_j nach der Reihenfolge ihres Erscheinens numeriert. Ordnet man jedem Knoten in R den am frühesten erschienenen Knoten der u_i zu, so erhält man eine Unterteilung von R in Blöcke $B_1, \ldots, B_k$ mit den folgenden Eigenschaften.

- Jeder Block B_j hat hat um eins verringerte chromatische Zahl $\chi - 1$ gegenüber der chromatischen Zahl χ des Graphen.

- Die Zuordnung eines Knotens in R zu einem Block B_i kann online erfolgen.

- Die durchschnittliche Anzahl von Knoten der Färbung c_i, und damit auch die durchschnittliche Anzahl k von Blöcken B_i, ist n/s, wobei s die Anzahl der Greedy-Farben ist.

Jeder Block der Restmenge wird nun mit dem gleichen Verfahren bearbeitet, wobei eine disjunkte Farbmenge benutzt wird, wodurch der Teilgraph separat — aber dennoch online — bearbeitet werden kann. So werden immer kleinere Restmengen und Blöcke erzeugt, bis die entstehenden Teilgraphen 2-färbbar und somit bipartit sind. Diese Teilgraphen können mit dem von Lovász *et al.* [LST88] beschriebenen Verfahren für bipartite Graphen gefärbt werden.

Der Greedy-Algorithmus, in der Literatur auch First-Fit genannt, verhält sich zwar im allgemeinen ungünstig, erzielt jedoch auf verschiedenen eingeschränkten Graphklassen sehr gute Ergebnisse. Dazu gehören Intervallgraphen [Kie88], Splitgraphen, sowie Komplemente von bipartiten Graphen [GL88] und induktive Graphen.

4.1.4 Ein Algorithmus für induktive Graphen

Ein Graph G heißt *d-induktiv*, wenn seine Knoten so numeriert werden können, daß jeder Knoten nur mit höchstens d Knoten mit kleineren Nummern verbunden ist. Diese Ordnung der Knoten wird dann *induktive Ordnung* genannt.

Lemma 4.1.1 *Für d-induktive Graphen G gilt* $\chi(G) \leq d+1$.

Beweis: Färbt man die Knoten in der Reihenfolge der induktiven Ordnung mit dem Greedy-Algorithmus FIRST-FIT, so werden höchstens $d+1$ Farben benötigt. □

Irani [Ira90] zeigt, daß der Greedy-Algorithmus FIRST-FIT höchstens $O(d \log n)$ Farben zur Färbung eines d-induktiven Graphs mit n Knoten benötigt.

Es ergibt sich eine Kompetitivität von $O(\log n)$ für planare Graphen, da diese 5-induktiv sind. Die gleiche Kompetitivität erhält man auch für *Sehnengraphen*. Ein Sehnengraph hat die Eigenschaft, daß es in jedem Zykel von Knoten $v_1, \ldots, v_k$ eine Kante gibt, die zwei Knoten v_i und v_j verbindet, wobei $|i-j| > 1$ und $(i,j) \neq (1,k)$. Diese Kante teilt den Zykel also in zwei kleinere Zykel auf. Man kann zeigen, daß Sehnengraphen $\chi(G)$-induktiv sind. Aufgrund dieser Eigenschaft folgt die behauptete Kompetitivität.

In der gleichen Arbeit zeigt Irani eine passende untere Schranke von $\Omega(\log n)$ für die Kompetitivität jedes Online-Algorithmus auf der Klasse der d-induktiven Graphen.

4.1.5 Färbung der Kanten

In einer Variante des Graph Coloring Problems werden nicht die Knoten, sondern die Kanten gefärbt. Wieder dürfen keine zwei über einen Knoten benachbarten Kanten die gleiche Farbe erhalten. Die minimale Anzahl von Farben für einen Graphen wird nun der chromatische *Index* $\chi'(G)$ genannt.

Vizing zeigte die erstaunliche Tatsache, daß stets $\Delta \leq \chi'(G) \leq \Delta + 1$ gilt, wobei Δ den maximale Knotenrang des Graphen bezeichnet. Der Beweis ist konstruktiv, und ein polynomieller Offline-Algorithmus, der stets eine Färbung mit $\Delta + 1$ Farben erzielt, kann angegeben werden.

Ein einfacher Online-Algorithmus für dieses Problem ist der Greedy-Algorithmus, der jeder Kante die kleinste noch mögliche Farbe zuordnet.

Satz 4.1.2 *Der Greedy-Algorithmus für das Kantenfärbungsproblem ist 2-kompetitiv.*

Beweis: Da jede Kante mit höchstens $2\Delta - 2$ anderen Kanten benachbart sein kann, ist klar, daß der Algorithmus höchstens $2\Delta - 1$ Farben verwendet. Der chromatische Index jedes Graphen ist jedoch Δ oder $\Delta + 1$. □

Bar-Noy, Motwani und Naor zeigen, daß der Greedy-Algorithmus tatsächlich optimal ist [BNMN92]. Sie konstruieren dazu einen Graphen, auf dem jeder deterministische Online-Algorithmus mindestens $2\Delta - 1$ Farben benötigt. Der Knotengrad des Graphen beträgt

$\Delta = O(\log n)$. Der Graph kann variiert werden, so daß auch jeder randomisierte Online-Algorithmus $2\Delta - 1$ Farben benötigt. Der Knotengrad beträgt in diesem Fall nur noch $O((\log n)^{1/2})$.

Eine offene Frage ist, ob es randomisierte Online-Algorithmen gibt, die für Graphen mit großem Knotengrad $\Delta = \Omega((\log n)^{1/2})$ eine bessere Kompetitivität erzielen.

4.1.6 Literatur — Graph Coloring

[Bea76] D. R. Bean. Effective coloration. *J. Symbolic Logic*, pages 469–480, 1976.

[BNMN92] A. Bar-Noy, R. Motwani, and J. Naor. The greedy algorithm is optimal for on-line edge coloring. *Information Processing Letters*, 44(5):251–253, 1992.

[GL88] A. Gyárfás and J. Lehel. On-line and first-fit colorings of graphs. *J. Graph Theory*, 12(2):217–227, 1988.

[Ira90] S. Irani. Coloring inductive graphs on-line. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 470–479, 1990.

[Kie88] H. A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM J. on Discrete Math.*, 1(4):526–530, 1988.

[LST88] L. Lovász, M. Saks, and W. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *SIAM J. Discrete Math.*, 1:526–530, 1988.

[Pat85] D. A. Patterson. Reduced instruction set computers. *Communications of the ACM*, 28(1), 1985.

[Sze] M. Szegedy. Personal communication, transmitted through [LST88].

[Vis92] S. Vishwanathan. Randomized online graph coloring. *J. of Algorithms*, 13:657–669, 1992.

4.2 Matching

Matchmaker, Matchmaker,
match me a match, find me a find, catch me a catch.
— Sheldon Harnick *Fiddler on the roof*

Ein weiteres graphentheoretisches Problem, das sich auf den Online-Fall verallgemeinern läßt, ist die Suche nach einem möglichst großen Matching eines bipartiten Graphen. Dies bedeutet eine Zuordnung von jeweils einem Element der linken und der rechten Seite des Graphen, so daß kein Element mehrfach ausgewählt wird.

Ein *bipartiter* Graph G besteht aus zwei Knotenmengen U und V mit jeweils n Knoten und einer Kantenmenge E, die nur Kanten $e = (u_i, v_j)$ mit $u_i \in U$ und $v_j \in V$ enthält.

Sei nun ein bipartiter Graph $G(U, V, E)$ mit $2n$ Knoten gegeben. Ein *Matching* ist dann eine Menge von Kanten $(u_i, v_i) \in E$, so daß $u_i \in U$ und $u_i \neq u_j$ für $i \neq j$ gilt, sowie $v_i \in V$ und $v_i \neq v_j$ für $i \neq j$. Ein Offline-Problem ist nun zum Beispiel, zu entscheiden, ob ein *perfektes Matching* existiert, d.h. ein Matching der Größe n. Ein *maximales* Matching ist ein Matching, das durch Hinzunahme von Kanten nicht mehr vergrößert werden kann.

4.2.1 Das Online-Problem

Gegeben sei wieder ein bipartiter Graph $G(U, V, E)$ auf $2n$ Knoten. Wir setzen nun jedoch voraus, daß G ein perfektes Matching enthält. Sei B die zu G gehörige $n \times n$ Strukturmatrix. Die Zeilen entsprechen den Elementen aus U, die im folgenden als *Jungen* bezeichnet werden, die Spalten den Elementen aus V, entsprechend *Mädchen* genannt. Jede Kante $e = (u_i, v_j)$ aus E entspricht einer 1 an der Position (i, j) in B.

Ein Online-Algorithmus für das Matching Problem erhält eine Spalte der Matrix nach der anderen und muß sich jeweils nach Erhalt entscheiden, welchem Jungen er das Mädchen zuordnet, das der Spalte entspricht, falls dies überhaupt möglich ist. Die Aufgabe des Algorithmus ist es, möglichst viele Paare zu schaffen.

4.2.2 Der deterministische Fall

Für deterministische Algorithmen ist das Problem sehr einfach zu lösen. Hier führt bereits ein beliebiger Greedy-Algorithmus, der stets nach irgendeinem (deterministischen) Prinzip einen Jungen auswählt, falls dies möglich ist, auf eine optimale kompetitive Lösung.

Satz 4.2.1 *Jeder Greedy-Algorithmus für das Matching Problem erzielt ein Matching, das mindestens die Größe $n/2$ hat. Dies ist optimal für deterministische Algorithmen.*

Beweis: Jeder Greedy-Algorithmus erzielt ein maximales Matching. Ein solches muß jedoch stets mindestens die Größe $n/2$ haben, da vorausgesetzt wurde, daß ein perfektes Matching existiert. Zum Beweis nehme man an, es gäbe ein maximales Matching M kleinerer Größe. Betrachtet man dann die Menge U_1 der Jungen, die in M Partnerinnen haben, so ist $|U_1| < n/2$. Weiterhin hat jeder Junge im perfekten Matching eine Partnerin. Dies gilt also insbesondere für alle Jungen, die nicht in U_1 liegen. Es gibt mehr als $n/2$ solcher Partnerinnen, daher ist mindestens eine im Matching M noch frei, denn dort sind erst $|U_1|$ Mädchen zugeordnet. Dies ist jedoch ein Widerspruch zur Voraussetzung, daß M maximales Matching war.

Daß dieses Ergebnis optimal ist, zeigt eine Matrix B, die in den ersten $n/2$ Spalten nur Einsen hat und danach nur in den Zeilen ungleich Null ist, die vom Algorithmus für die ersten $n/2$ Mädchen verwendet wurden. Auf dieser Matrix erzielt kein deterministischer Algorithmus ein größeres Matching als $n/2$. □

Aus Satz 4.2.1 folgt, daß jeder Greedy-Algorithmus 2-kompetitiv ist.

4.2.3 Randomisierte Algorithmen

Betrachtet man randomisierte Algorithmen, so ist klar, daß für Kompetitivität gegen adaptive Offline-Gegenspieler die gleichen Schranken wie für den deterministischen Fall gelten. Von Interesse sind also unwissende und adaptive Online-Gegenspieler.

Adaptive Online-Gegenspieler

Ein adaptiver Online-Gegenspieler kann die Größe eines von einem Online-Algorithmus A erzeugten Matchings durch $n/2 + O(\log n)$ beschränken. Somit kann auch in diesem Fall kein Algorithmus besser als 2-kompetitiv sein.

Diese untere Schranke wird durch folgende Konstruktion bewiesen. In den ersten $n/2$ Spalten wählt der Gegenspieler Einsen genau für die Jungen, die bisher noch nicht von ihm oder dem Algorithmus A zugeordnet wurden. Für sein eigenes Matching wählt der Gegenspieler hier einen der noch freien Jungen zufällig aus.

In der zweiten Hälfte der Spalten schreibt der Gegenspieler eine Eins für jeden Jungen, der bisher noch nicht von ihm selbst zugeordnet wurde. Auch hier wählt er von den noch freien Jungen zufällig einen aus.

Auf diese Weise konstruiert der Gegenspieler ein perfektes Matching. Auf der anderen Seite sei $T(A)$ die Anzahl der Jungen, die in den ersten $n/2$ Spalten von A und dem Gegenspieler zugeordnet werden. Der Erwartungswert von $T(A)$ liegt in $O(\log n)$, und die Größe des Matchings, das A erzeugt, ist beschränkt durch $n/2 + T(A)$.

Unwissende Gegenspieler

Für den Fall unwissender Gegenspieler geben Karp *et al.* [KVV90] den Algorithmus Ranking an. Dieser wählt zunächst eine zufällige Permutation der Jungen aus und erhält damit eine Rangfolge. Jedem Mädchen wird nun unter den verfügbaren Jungen derjenige zugeordnet, der den höchsten Rang hat.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Abbildung 4.2: Eine ungünstige Matrix für Random

Die Autoren vermuten, daß der Algorithmus Ranking etwa $1/(1-\frac{1}{e})$-kompetitiv gegen unwissende Gegenspieler ist. Der kompetitive Faktor ist somit ungefähr $1,58$. Sie beweisen die Vermutung jedoch nur bis auf Terme kleinerer Ordnung, die in $o(n)$ liegen.

Natürlicher mag zunächst der Algorithmus RANDOM erscheinen, der zufällig einen der verfügbaren Jungen auswählt. Dieser Algorithmus erreicht jedoch nicht die gleiche Kompetitivität. Dazu betrachtet man eine Matrix B mit $b_{ij} = 1$ für $1 \leq i, j \leq n/2$ und für $j = n+1-i$ und $b_{ij} = 0$ sonst. Die Matrix ist in Abbildung 4.2 dargestellt. RANDOM konzentriert sich bei dieser Matrix zu stark auf den dichten oberen Teil der ersten Hälfte und wählt zu wenige der günstigeren Zuordnungen auf der Diagonalen. Die erwartete Größe des Matchings liegt daher in $O(n/2 + \log n)$.

Der Algorithmus RANKING hingegen tendiert dazu, eher Jungen zuzuordnen, die in der Vergangenheit seltener verfügbar waren. Dies führt dazu, daß er eine höhere Kompetitivität als 2 erreicht.

4.2.4 Matching mit Gewichten

Eine interessante Verallgemeinerung des Matching Problems ergibt sich, wenn man jedem möglichen Paar (u_i, v_j) ein bestimmtes Gewicht zuordnet. Die Summe der Gewichte soll dann minimiert oder maximiert werden. Genauer wird jeder Kante (u_i, v_j) des bipartiten Graphen G ein Gewicht oder eine Distanz $d(u_i, v_j)$ zugeordnet. G kann als vollständiger bipartiter Graph vorausgesetzt werden, da die Gewichte beliebig groß gemacht werden können. Die Strukturmatrix B hat nun die Form $b_{ij} = d(u_i, v_j)$.

Wieder wird eine Spalte der Strukturmatrix nach der anderen enthüllt, wobei der Online-Algorithmus die Aufgabe hat, dem entsprechenden Mädchen einen Jungen zuzuordnen.

Es läßt sich leicht zeigen, daß für dieses allgemeinere Problem kein kompetitiver deterministischer Online-Algorithmus existieren kann. Dazu betrachtet man für die Suche eines minimalen Matchings zum Beispiel den Fall $n = 2$ und die folgende Matrix B.

$$B = \begin{pmatrix} 1 & 0 \\ 1 & N \end{pmatrix}$$

Dabei stehe die Zahl 0 in der Zeile, die vom Algorithmus für die erste Spalte gewählt wird. Der Algorithmus erhält ein Matching der Größe $N + 1$, optimal ist 1. N kann beliebig groß gewählt werden. Für ein maximales Matching kann ein analoges Beispiel konstruiert werden.

Kalyanasundaram und Pruhs [KP93] diskutieren einen interessanten Spezialfall des gewichteten Matching Problems, der deutliche Parallelen zum k-Server Problem zeigt. Sie betrachten einen allgemeinen metrischen Raum M. Sowohl die Knotenmenge U, die wir nun als *Server* bezeichnen, als auch die Menge V, die *Anfragen*, sind — nicht notwendigerweise disjunkte — Teilmengen von M. Die Anordnung der Server ist dem Algorithmus bekannt, die Anfragen erscheinen nun nach und nach, wobei mit jeder Anfrage auch die Distanzen vom Anfragepunkt zu jedem Server angegeben werden. Die Distanz entspricht dabei der Distanz in M. Die Kosten des Online-Algorithmus sind die Distanzen der einzelnen Server. Es ist klar, daß diese Einschränkung für die Strukturmatrix B den obigen Fall nicht mehr zuläßt.

Der Unterschied zum k-Server Problem ist der, daß ein Server während der Bearbeitung der gesamten Anfragen nur einmal bewegt werden darf und daß die Anzahl der Anfragen bekannt ist.

Für maximales Matching schlagen Sie den deterministischen Algorithmus FARTHEST-NEIGHBOR, der stets den am weitesten entfernten Server für eine Anfrage benutzt, vor und zeigen die folgende Aussage.

Satz 4.2.2 FARTHEST-NEIGHBOR *ist 3-kompetitiv, und dieser kompetitive Faktor ist optimal für deterministische Algorithmen.*

Beweis: Wir beweisen hier nur die Optimalität. Diese ist an folgendem Graphen G_n mit $2n + 1$ Knoten erkennbar. Der Graph besteht aus einem Mittelpunkt c, sowie n Paaren von Punkten a_i, b_i. Die Distanz zwischen a_i und b_j sei 3 für $i = j$ und 1 für $i \neq j$. Die Distanz von a_i zum Zentrum c ist stets 1. Alle übrigen Distanzen ergeben sich als Länge des kürzesten Pfades im Graphen.

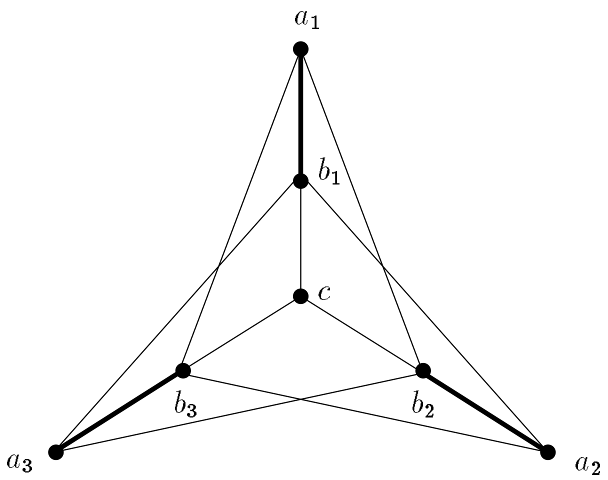

Abbildung 4.3: Der Graph G_3.

Die Server befinden sich zu Beginn auf den Punkten a_i. Die erste Anfrage erfolgt auf c, und die folgenden Anfragen stets an dem Punkt b_i, für den der entsprechende Server auf a_i im letzten Zug bewegt wurde. Die letzte Anfrage erfolgt am Punkt a_j des letzten noch nicht benutzten Servers. Somit kann der Algorithmus keine der Kanten mit Gewicht 3 benutzen. Als Gewicht des Matchings ergibt sich $n - 1$, da die letzte Anfrage kostenlos bearbeitet wird.

Das optimale Matching besteht darin, die erste Anfrage mit dem Server auf a_j zu bearbeiten, und danach stets den Server auf a_i für die Anfrage b_i zu benutzen. Für die Anfrage auf a_j wird der letzte noch freie Server verwendet. Dieses Matching hat die Größe $3(n - 2) + 2 + 1 = 3(n - 1)$. □

Betrachtet man nun das Minimum Matching Problem, zeigt sich, daß sich dieses Ergebnis nicht direkt übertragen läßt. Der Algorithmus NEAREST-NEIGHBOR, der stets den nächsten Server benutzt, ist nämlich nur $(2^n - 1)$-kompetitiv.

Kalyanasundaram und Pruhs geben daraufhin den deterministischen Online-Algorithmus **Permutation** für minimales Matching an und zeigen, daß er $(2n-1)$-kompetitiv ist. Auch dieses Ergebnis ist optimal für deterministische Online-Algorithmen. Dazu betrachte man einen sternförmigen Raum H_n mit $n+1$ Punkten. Alle Punkten sind über einen Mittelpunkt c miteinander verbunden. Die Distanz zu c ist jeweils 1, alle anderen Verbindungen haben die Länge 2. Die Server befinden sich auf den äußeren Punkten. Die erste Anfrage findet im Zentrum statt, alle folgenden Anfragen auf dem Punkt, an dem sich der zuletzt vom Algorithmus bewegte Server befand. Die Kosten sind somit $2n-1$. Das optimale Matching bewegt nur einen Server zum Mittelpunkt. Alle anderen Server bewegen sich nicht, so daß Kosten genau 1 betragen.

4.2.5 Offene Probleme

Für das allgemeine Matching Problem ist die wichtigste offene Fragestellung, ob der von Karp *et al.* vorgeschlagene Algorithmus **Ranking** genau die vermutete Kompetitivität von $1/(1-\frac{1}{e})$ hat, oder ob die Terme kleiner Ordnung tatsächlich auftreten. Weiterhin ließe sich untersuchen, ob dieser Algorithmus auch dann gute Resultate zeigt, falls der Graph G kein perfektes Matching enthält. Interessant wäre auch, das gewichtete Matching Problem für randomisierte Algorithmen zu untersuchen.

4.2.6 Literatur — Matching

[KP93] Kalyanasudraram and Pruhs. On-line weighted matching. *J. of Algorithms*, 14:478–488, 1993.

[KVV90] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 352–358, 1990.

4.3 Routenplanung

Wir suchen niemals die Dinge,
sondern das Suchen nach ihnen.
— Blaise Pascal *Gedanken über die Religion*

Auch auf den Bereich geometrischer Problemstellungen läßt sich die Methode der kompetitiven Analyse anwenden. Wir werden hier einige *Routenplanungsprobleme* betrachten.

Ein Roboter soll dabei so gesteuert werden, daß er sich in einer zunächst unbekannten, zweidimensionalen Umgebung zurechtfindet. Zu diesem Zweck verfügt er über einen Antrieb und ein optisches System, das alle sichtbaren Kanten von Hindernissen, sowie deren Distanz meldet.

Die Literatur zu diesem Gebiet ist vielfältig, wir beschränken uns hier auf Ergebnisse, die kompetitive Aussagen treffen. Weiterhin soll die Geometrie des Roboters vernachlässigt werden, dieser wird stets als punktförmig vorausgesetzt.

Das Modell unterscheidet sich von den vorherigen dadurch, daß hier die fehlende Information nicht durch zeitliche, sondern durch räumliche Trennung entsteht. Der Online-Algorithmus bestimmt durch seine Entscheidungen mit, in welcher Reihenfolge er die fehlende Information erhält. Daher gibt es hier auch keine festgelegte Anfragefolge mehr.

Die Kosten in den Routenplanungsproblemen entsprechen dem vom Roboter zurückgelegten Weg, die Kompetitivität eines Online-Algorithmus wird bestimmt durch das Verhältnis dieses Wegs zum kürzesten Weg, den ein Offline-Roboter wählen könnte, der gleichzeitig von oben auf die Szene sieht und somit bereits eine vollständige Karte zur Verfügung hat.

4.3.1 Das k-Wege Problem

Baeza-Yates *et al.* untersuchen die Suche nach einem Zielpunkt auf einer unbegrenzten Gerade [BYCR91].

Das Problem läßt sich verdeutlichen an einer Kuh, die an einem unendlich langen Weidezaun steht, von dem sie weiß, daß er irgendwo ein Loch hat. Die Kuh möchte zum Loch gelangen, weiß jedoch nicht, ob dieses sich rechts oder links von ihrem Startpunkt befindet.

In diesem Beispiel hat man die Möglichkeit, zwischen zwei Richtungen zu wählen. Allgemein läßt sich das *k-Wege Problem* untersuchen, in dem die Suche auf einer Kreuzung mit k Wegen startet. Nur auf einem Weg findet sich das Ziel, dessen Entfernung unbekannt ist. Das Ziel wird erst erkannt, wenn man sich direkt auf ihm befindet.

Das gleiche Problem kann auch in einem nicht-geometrischen Kontext auftauchen. Baeza-Yates *et al.* geben als Beispiel die Suche nach einem Datensatz an, der auf einem von k Magnetbändern gespeichert ist. Dabei steht nur ein Bandleser zur Verfügung und die Bänder sind zu lang, um sie der Reihe nach vollständig zu durchsuchen.

Ein deterministischer Algorithmus

Betrachten wir zunächst die Suche auf einer Geraden, d.h. $k = 2$. Jeder Algorithmus muß in diesem Fall die Bereiche links und rechts vom Startpunkt abwechselnd besuchen und dabei stets ein Stück weiter vordringen. Somit können deterministische Algorithmen für dieses Problem charakterisiert werden durch die Anzahl von Schritten, die sie zwischen den Wendepunkten ausführen. Sei $f(i)$ der Abstand des i-ten Wendepunkts vom Startpunkt der Suche.

In jedem Fall muß vorausgesetzt werden, daß der Abstand des Zielpunkts d größer ist als der erste Umkehrpunkt $f(1)$, andernfalls ist kein kompetitiver Algorithmus möglich. Denn für die Wahl $d = f(1)/n$ erhält man eine Kompetitivität von n, und n kann beliebig groß gemacht werden. Umgekehrt bedeutet dies, daß dem Algorithmus eine untere Schranke für den Abstand des Ziels bekannt sein muß.

Diese Einschränkung wird nicht benötigt, wenn man für die Definition der Kompetitivität die additive Konstante a aus Abschnitt 2.1.1 benutzt. Wählt man $a = 2$, so ist der im folgenden dargestellte Algorithmus für jeden Abstand d 9-kompetitiv.

Baeza-Yates *et al.* schlagen zur Suche auf einer Geraden die Wahl $f(i) = 2^i$ vor.

Satz 4.3.1 *Der exponentielle Algorithmus $f(i) = 2^i$ erreicht eine Kompetitivität von 9.*

Beweis: Die Distanz d des Ziels t liege zwischen $f(i)$ und $f(i+2)$, d.h. t wird nach dem $(i+1)$-ten Wendepunkt gefunden. Sei $\epsilon = d - f(i)$. Für die Kompetitivität c gilt dann

$$c = \sup_{i \in \mathbb{N}} \frac{2\sum_{j=1}^{i+1} f(j) + f(i) + \epsilon}{f(i) + \epsilon}.$$

Der ungünstigste Wert ergibt sich für sehr kleines ϵ. Für die oben angegebene Wahl von $f(i)$ erhalten wir

$$c = \sup_{i \in \mathbb{N}} \frac{9 \cdot 2^i - 2 + \epsilon}{2^i + \epsilon} = 9.$$

□

Auch Algorithmen für das allgemeine k-Wege Problem lassen sich durch den Abstand $f(i)$ ihrer Umkehrpunkte vom Ursprung charakterisieren. Die einzelnen Wege werden dabei jeweils zyklisch aufgesucht. Ein öfteres Aufsuchen eines einzelnen Weges kann keine Vorteile bringen.

Baeza-Yates *et al.* verallgemeinern in der gleichen Arbeit den Algorithmus für den Fall $k > 2$ durch die Wahl

$$f(i) = \left(\frac{k}{k-1}\right)^i.$$

Sie zeigen, daß dieser Algorithmus eine Kompetitivität von

$$1 + 2\frac{k^k}{(k-1)^{k-1}}$$

erreicht. Dieser Wert nähert sich für $k \to \infty$ asymptotisch dem Wert $(1 + 2ek)$, dabei bezeichnet e die Eulersche Konstante. Insbesondere ergeben sich eine $14,5$-kompetitive Strategie für $k = 3$ und eine ungefähr $19,97$-kompetitive Strategie für $k = 4$.

Untere Schranke

Baeza-Yates *et al.* [BYCR91] zeigen auch, daß ihr Verfahren für die k-Wege Suche optimal ist.

Satz 4.3.2 *Für das Problem der k-Wege Suche kann es für*

$$c < 1 + 2\frac{k^k}{(k-1)^{k-1}}$$

kein deterministisches c-kompetitives Verfahren geben.

Die gezeigte untere Schranke gilt nur für deterministische Verfahren für die k-Wege Suche. Im folgenden Abschnitt werden wir sehen, daß Randomisierung die Kompetitivität in diesem Fall tatsächlich verbessert.

Ein randomisierter Algorithmus

Kao, Reif und Tate [KRT93] untersuchen das k-Wege Problem ebenfalls und geben einen randomisierten Algorithmus an, der eine noch bessere Kompetitivität erreicht.

Für eine zunächst unbestimmte Konstante r läßt sich der Algorithmus wie folgt formulieren.

Algorithmus Smart-Cow
Wähle zufällig eine Permutation $\sigma \in \mathcal{S}_k$ aus.
Wähle zufällig ein ϵ gleichverteilt aus $[0, 1)$.
$d := r^\epsilon$
$p := 1$
repeat
  Gehe auf Pfad $\sigma(p)$ bis zur Distanz d.
  if t nicht gefunden
    then Gehe zurück zum Startpunkt.
      $p := (p \bmod k) + 1$
      $d := d \cdot r$
  end if
until t gefunden ;

Interessant ist, daß dieser Algorithmus die Randomisierung nur in einer schwachen Form ausnutzt. Nur die Reihenfolge σ, in der die Wege aufgesucht werden, und der Startwert ϵ werden zufällig ausgewählt. Während der eigentlichen Suche wird kein Zufallsgenerator mehr benötigt.

Satz 4.3.3 *Die Kompetitivität des Algorithmus* Smart-Cow *gegen unwissende Gegenspieler beträgt für jedes $r > 1$*

$$1 + \frac{2}{k} \cdot \frac{1 + r + r^2 + \cdots + r^{k-1}}{\ln r}.$$

Nun kann die Konstante r abhängig von k so bestimmt werden, daß die Kompetitivität des Algorithmus minimal wird.

Satz 4.3.4 *Die eindeutige Lösung r_k^* der Gleichung*

$$\ln r = \frac{1 + r + \cdots + r^{k-1}}{r + 2r^2 + \cdots + (k-1)^{k-1}}$$

für $r > 1$ ist genau die Wahl von r, die die Kompetitivität des Algorithmus Smart-Cow *minimiert.*

Für spezielle Werte von k ist die Kompetitivität c_k^* des Algorithmus für die beste Wahl r_k^* von r in Tabelle 4.1 wiedergegeben.

k	r_k^*	Kompetitivität c_k^*	Opt. det. Kompetitivität
2	3,59112	4,59112	9
3	2,01092	7,73232	14,5
4	1,62193	10,84181	19,96296
5	1,44827	13,94159	25,41406
6	1,35020	17,03709	30,85984
7	1,28726	20,13033	36,30277

Tabelle 4.1: Näherungen für die Kompetitivität c_k^* des Algorithmus SMART-COW

Die Autoren zeigen, daß der Algorithmus SMART-COW für den Fall $k = 2$ optimal ist, und vermuten, daß dies auch für größere k der Fall ist.

4.3.2 Suche auf einem Gitter

Ein weiteres Problem, bei dem die Koordinaten des Zieles unbekannt sind, ist die Suche nach einem Ziel in der Ebene. Dabei wird angenommen, daß das Ziel ganzzahlige Koordinaten habe, sich also auf einem Gitter befindet. Die Entfernung d des Ziels vom Ursprung, dem Startpunkt der Suche, ist unbekannt. d wird in Schritten auf dem Gitter gemessen. Das Ziel kann wie beim k-Wege Problem erst erkannt werden, wenn der Roboter sich direkt auf ihm befindet.

Baeza-Yates *et al.* [BYCR91] untersuchen dieses Problem. Sie zeigen zunächst, daß jedes Verfahren für die Suche mindestens $2d^2 + 4d + 1$ Schritte benötigt. Somit kann es kein kompetitives Verfahren geben. Sie geben daraufhin die Algorithmen BALANCED und FLIPPED-BALANCED an. Der zweite Algorithmus benötigt höchstens $2d^2 + 5d + 2$ Schritte, bis das Ziel gefunden ist.

Das Problem der Suche auf einem Gitter läßt sich übertragen auf die Suche nach einem beliebigen Punkt in der Ebene, unter der Annahme, daß der Punkt aus einer kleinen Distanz $\delta > 0$ erkannt werden kann.

4.3.3 Suchen in unbekannten Polygonen

In den bisher vorgestellten Modelle zur Suche in unbekannter Umgebung war die Szene, in der gesucht wurde, bis auf die Position des Zieles bekannt. Ein allgemeineres Modell für eine Szene, in der gesucht werden soll, besteht aus einem einfachen Polygon P, d.h. einem Polygon ohne Selbstschnitte.

Auf dem Rand von P liegen zwei ausgezeichnete Punkte s und t. Der Roboter muß einen Weg vom Startpunkt s zum Zielpunkt t finden. Dazu bedient er sich eines visuellen Sensors,

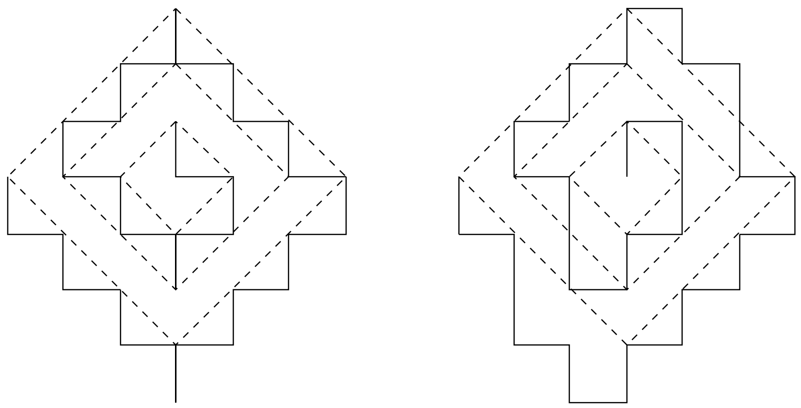

Abbildung 4.4: Zwei Algorithmen zum Suchen auf einem Gitter.

der ihm den von seiner Position p aus sichtbaren Bereich $vis(p)$, das *Sichtbarkeitspolygon* anzeigt. Dabei ist ein Punkt q von p aus *sichtbar*, wenn die Strecke, die p und q verbindet, vollständig innerhalb von P verläuft.

Informationen über das Polygon P erhält der Roboter allein über das Sichtbarkeitspolygon. Die Koordinaten des Ziels t sind dem Roboter nicht bekannt. Er kann das Ziel jedoch erkennen, sobald es sichtbar wird.

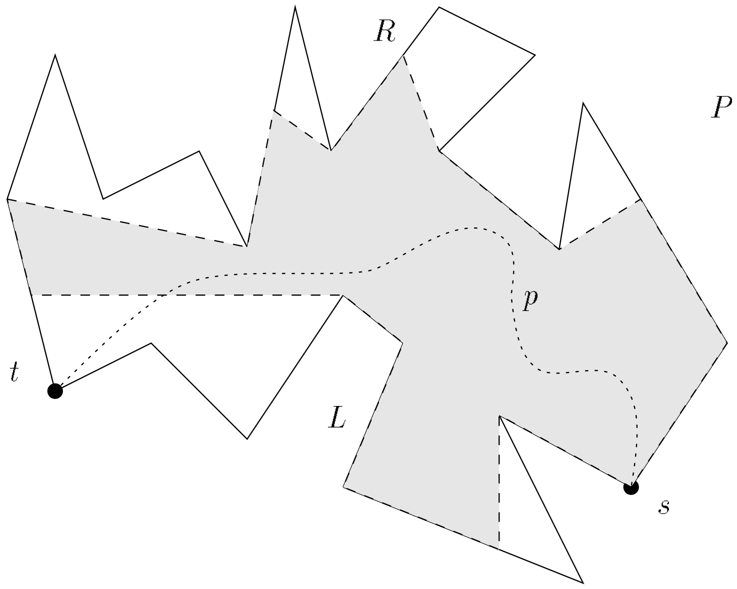

Abbildung 4.5: Ein Beispiel für die Suche in einem unbekannten Polygon. Der von p aus sichtbare Bereich $vis(p)$ ist schattiert wiedergegeben.

Die beiden Punkte s und t unterteilen P in zwei Streckenzüge L und R. L ist die Verbindung von s nach t im Uhrzeigersinn, R entsprechend entgegen dem Uhrzeigersinn.

Definitionen

In diesem Abschnitt fassen wir die Definitionen einiger Begriff zusammen, die wir bei der Beschreibung der Verfahren zum Suchen und Kartographieren in Polygonen verwenden werden.

Zunächst eine Bemerkung zum Sichtbarkeitspolygon $vis(p)$. Die Kanten dieses Polygons entstehen entweder durch Kanten von P oder dadurch, daß der Roboter von p aus nicht um eine Ecke l herum sehen kann. In letzterem Fall bezeichnen wir die Kante von $vis(p)$ als *Eingangskante*, in ersterem als *Wandkante*, siehe Abbildung 4.6. Eine Eingangskante f liegt immer zwischen zwei Wandkanten e_1 und e_2. e_1 und e_2 nennen wir *benachbarte* Wandkanten.

Da eine Eingangskante f immer genau auf die Position p zuläuft, ist einer der Endpunkte von f näher an p als der andere. Diesen Endpunkt bezeichnen wir als *unteren* Endpunkt von f, entsprechend den anderen Endpunkt entsprechend als *oberen* Endpunkt. Hat die Wandkante e_1 mit f den unteren Endpunkt gemein, so sagen wir, e_1 liegt *unterhalb* von seiner benachbarten Wandkante e_2.

Definition 4.3.5 *Einen Randpunkt l des Polygons P heißt* Landmark *bezüglich eines Punktes p, wenn l unterer Endpunkt einer Eingangskante von $vis(p)$ ist.*

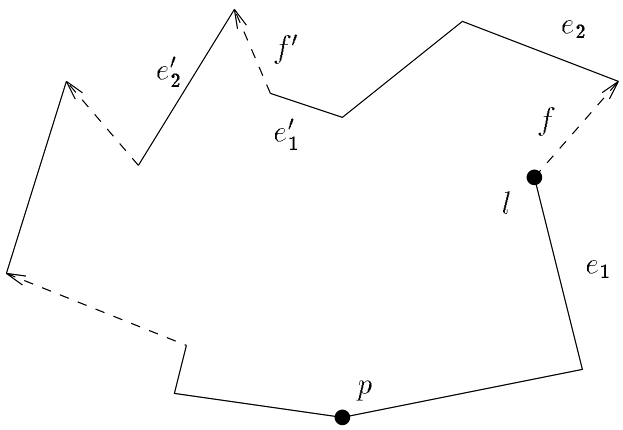

Abbildung 4.6: Ein Sichtbarkeitspolygon, f ist eine Eingangskante, e_1 und e_2 die benachbarten Wandkanten zu f. e_1 liegt unterhalb von e_2.

Landmarks sind wichtige Punkte bei der Suche in unbekannten Szenen, da sie Eckpunkte bezeichnen, hinter denen das Ziel oder der weitere Weg verborgen sein kann. Von den beiden an einem Landmark l angrenzenden Randkanten ist stets eine Kante vollständig unsichtbar. Diese nicht sichtbare Kante bezeichnen wir als von l *verdeckte* Kante.

Das folgende Lemma zeigt eine andere Klassifikation des Begriffes Landmark.

Lemma 4.3.6 *Ein Randpunkt l ist genau dann ein Landmark bezüglich einer Position p, wenn*

- *l sichtbar ist von p und*
- *nur eine der an l angrenzenden Polygonkanten von p aus sichtbar ist.*

Definition 4.3.7 *Eine Strecke c im Inneren des Polygons P wird* Sehne *genannt, wenn die beiden Endpunkte p und q auf dem Rand von P liegen. c heißt LR-*Sehne*, wenn zusätzlich gilt $p \in R$ und $q \in L$, d.h. c den linken und rechten Rand von P verbindet.*

Betrachten wir nun eine Kante e des Polygons mit mindestens einer konkaven Ecke v. Verlängert man die Kante e von v aus, bis die Polygongrenze wieder getroffen wird, so erhält man eine Sehne c, die wir als von e in Richtung v *induzierte Sehne* bezeichnen wollen. Eine Kante e kann somit ein oder zwei induzierte Sehnen haben.

Ist l ein Landmark und e die von l verdeckte Kante, so bezeichnen wir die von e in Richtung l induzierte Sehne vereinfacht als vom Landmark l induzierte Sehne.

Definition 4.3.8 *Ein Punkt p wird von einer Sehne c* dominiert, *wenn jeder Pfad von s nach p die Sehne c schneidet. Eine Kante oder Sehne e wird von einer Sehne c* dominiert, *wenn ein (und damit jeder) Punkt von e durch c dominiert wird.*

Allgemeine Polygone

Schränkt man das Polygon P nicht weiter ein, so ist keine kompetitive Strategie zur Suche nach t mit einer von P unabhängigen Kompetitivität möglich.

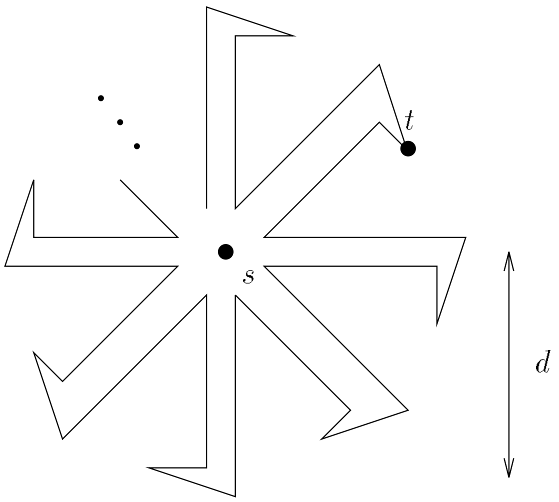

Abbildung 4.7: In allgemeinen Polygonen ist keine kompetitive Suche mit konstanter Kompetitivität möglich.

Dazu ist in Abbildung 4.7 ein Beispiel wiedergegeben. Das Ziel t kann in jedem der von s aus sternförmig abgehenden Eingänge sein. Die Anzahl der Eingänge kann beliebig groß gemacht werden und die Distanz d zwischen s und t muß für jeden Eingang zweimal durchlaufen werden, bis das Ziel gefunden ist.

Satz 4.3.9 *Für jedes deterministische c-kompetitive Verfahren zur Suche in allgemeinen Polygonen mit n Eckpunkten gilt* $c \geq \frac{1}{2}n - 1$.

Beweis: Sind im Beispiel aus Abbildung 4.7 insgesamt m Eingänge vorhanden, so gilt $n = 4m$. Jeder deterministische Algorithmus muß zur Suche nach t mindestens eine Strecke von $(2m-1)d = (\frac{1}{2}n - 1)d$ zurücklegen. Der kürzeste Pfad zum Ziel hat eine Länge von d. □

Auch für die Kompetitivität randomisierter Algorithmen läßt sich eine untere Schranke von $\Omega(n)$ angeben.

Ob es ein $O(n)$-kompetitives Verfahren zur Suche in allgemeinen Polygonen mit n Eckpunkten gibt, ist ein offenes Problem.

Iso-orientierte Polygone

Auch für iso-orientierte Polygone kann es kein Suchverfahren mit konstanter Kompetitivität geben, da sich das oben gezeigte Beispiel leicht verändern läßt, so daß P iso-orientiert ist.

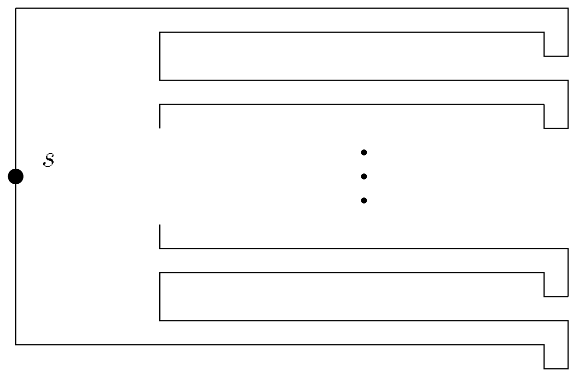

Abbildung 4.8: Ein iso-orientiertes Polygon, das keine kompetitive Suche zuläßt.

Mit dem in Abbildung 4.8 gezeigten Polygon folgt die folgende untere Schranke.

Satz 4.3.10 *Für jedes deterministische c-kompetitive Verfahren zur Suche in allgemeinen iso-orientierten Polygonen mit n Eckpunkten gilt* $c \geq \frac{1}{3}n - \frac{7}{6}$.

Beweis: Analog zum Beweis von Satz 4.3.9 ist der Pfad jedes Online-Algorithmus $\geq 2m - 1$, wobei m die Anzahl der Eingänge bezeichnet, in denen das Ziel versteckt ist. Hier gilt jedoch $m = (n-1)/6$, woraus die untere Schranke folgt. □

Es ist möglich, Verfahren zu konstruieren, deren kompetitiver Faktor von bestimmten geometrischen Eigenschaften des Polygons P abhängt. So gibt Kleinberg [Kle94] ein Verfahren an, das eine Kompetitivität in Abhängigkeit von der Anzahl der *wesentlichen Sehnen* des Polygons erreicht.

Dieser Begriff verlangt eine genauere Erläuterung.

Definition 4.3.11 *Eine von einer Kante e induzierte Sehne c heißt* wesentlich, *wenn c die Kante e, aber keine von einer anderen Kante des Polygons induzierte Sehne c′ dominiert.*

In Abbildung 4.9 ist ein Polygon mit eingezeichneten wesentlichen Sehnen wiedergegeben.

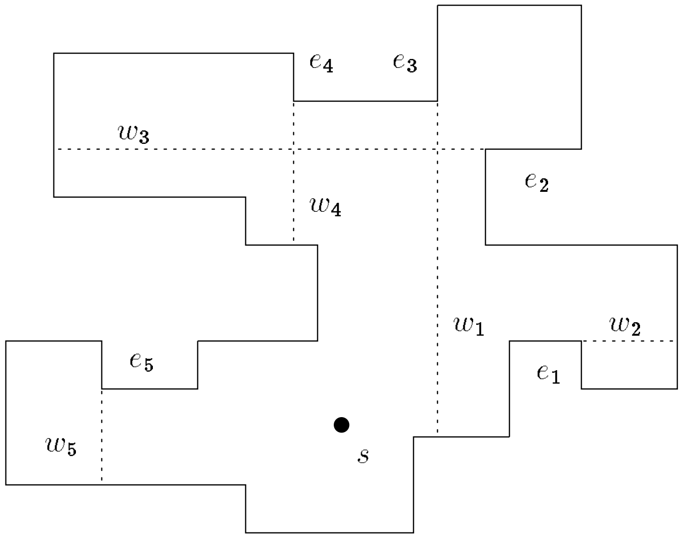

Abbildung 4.9: Ein Polygon und seine wesentlichen Sehnen.

Der Begriff der wesentlichen Sehnen wird bei der Kartographierung von Polygonen in Abschnitt 4.3.4 eine wichtige Rolle spielen, da für die Menge der wesentlichen Sehnen die folgende Eigenschaft gilt.

Lemma 4.3.12 *Von jedem Pfad, der alle wesentlichen Sehnen eines iso-orientierten Polygons P schneidet, sind alle Punkte von P sichtbar.*

Wir sagen in diesem Fall, der Pfad *kartographiert* das Polygon P. Umgekehrt gilt auch

Lemma 4.3.13 *Jeder Pfad, der P kartographiert, schneidet alle wesentlichen Sehnen von P.*

Kehren wir zurück zum Verfahren von Kleinberg zur Suche in allgemeinen iso-orientierten Polygonen. Das Verfahren benutzt den Algorithmus zur Kartographierung eines Polygons mit m wesentlichen Sehnen. Dieser Algorithmus ist in Abschnitt 4.3.4 beschrieben und erzielt eine Kompetitivität von 2 für die Kartographierung von Polygonen.

Sei P^δ das Polygon, das man erhält, wenn man das Innere von P an den induzierten Sehnen abschneidet, die weiter als δ von s entfernt liegen. Man wandelt diese Sehnen also in Wandkanten um und verkürzt so die Begrenzung von P.

Nun gibt es ein kleines δ_0, so daß ganz P^{δ_0} von s aus sichtbar ist. Nehmen wir o.B.d.A. an, daß $\delta_0 = 1$, ansonsten können wir das Polygon P entsprechend skalieren.

Algorithmus EXPLORATION-SEARCH
Setze $\delta := 1$.
while t ist nicht gefunden **do**
  Kartographiere P^δ mit dem Algorithmus aus
    Abschnitt 4.3.4.
  Setze $\delta := 2 \cdot \delta$.
end while;

Der beschriebene Algorithmus hält, da für genügend großes δ gilt $P^\delta = P$ und somit t in P^δ enthalten ist.

Um die Kartographierung von P^δ durchführen zu können, muß der Algorithmus erkennen, wann eine induzierte Sehne c zu weit entfernt vom Startpunkt s liegt. In diesem Fall muß er c als Wand auffassen. Man kann zeigen, daß dem Algorithmus dies online möglich ist.

Satz 4.3.14 *Der Online-Algorithmus* EXPLORATION-SEARCH *zur Suche in iso-orientierten Polygonen mit m wesentlichen Sehnen ist $O(m)$-kompetitiv.*

Da die Anzahl der wesentlichen Sehnen eines Polygons P durch die Anzahl der Kanten und damit der Eckpunkte n beschränkt ist, haben wir somit auch einen $O(n)$-kompetitiven Algorithmus für iso-orientierte Polygone mit n Eckpunkten gewonnen. Dies ist aufgrund von Satz 4.3.10 optimal.

Straßen

Da in allgemeinen Polygonen keine Suche mit konstanter Kompetitivität möglich ist, untersucht man Teilklassen von Polygonen, die diese erlauben. Klein gab als erster eine solche Klasse von Polygonen an, die er *Straßen* nennt [Kle91]. Diese wird definiert über die Streckenzüge L und R aus Abschnitt 4.3.3.

Definition 4.3.15 *Ein Polygon P heißt* Straße, *wenn die beiden Streckenzüge L und R gegenseitig schwach sichtbar sind, d.h. von jedem Punkt aus L ist mindestens ein Punkt aus R sichtbar und umgekehrt.*

Das Beispielpolygon in Abbildung 4.5 auf Seite 17 ist eine Straße.

Der Algorithmus

Wir beschreiben nun den von Klein vorgestellten Algorithmus zur Suche in einer unbekannten Straße. Der Pfad, der vom Algorithmus erzeugt wird, besteht aus einer Folge von Strecken, die wir mit $(l_i)_{1 \leq i \leq m}$ bezeichnen wollen. Die Strecke l_i verbindet jeweils die Punkte p_{i-1} und p_i. Es gilt $p_0 = s$ und $p_m = t$. Zu Beginn jeder Strecke l_i bestimmt der Algorithmus zwei Punkte v_L und v_R, die zur Orientierung dienen. Diese beiden Punkte

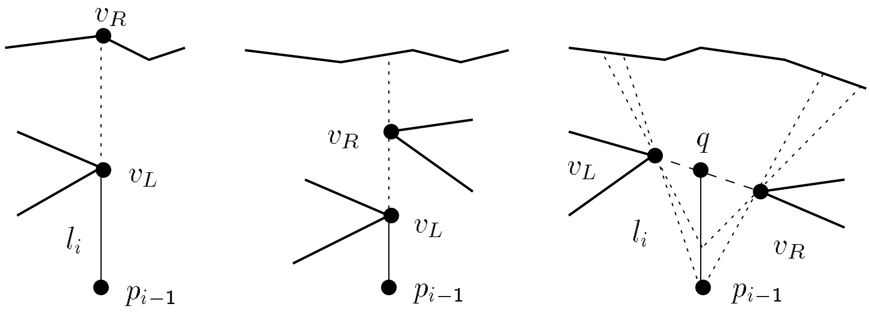

Abbildung 4.10: Die drei Fälle im Algorithmus von Klein.

werden stets so gewählt, daß $v_L \in L$ und $v_R \in R$ gilt und einer der Fälle in Abbildung 4.10 vorliegt. Wird das Ziel t an irgendeiner Stelle der Suche sichtbar, so beginnt die letzte Strecke l_m, die direkt zum Ziel verläuft.

In den Fällen 1.1 und 1.2 endet die Strecke l_i an v_L, bzw. v_R, im Fall 2 wird zunächst ein Punkt q auf der Verbindungslinie von v_L und v_R gewählt. Dieser Punkt wird angestrebt und die Strecke l_i endet, wenn eines von mehreren möglichen Ereignissen eintritt, die weiter unten beschrieben werden.

Es bleibt zu beschreiben, wie die Punkte v_L und v_R bestimmt werden. Anfangs gilt $v_L = v_R = s$.

Fall 1: Verläuft die Strecke l_{i-1} wie im Fall 1.1 oder 1.2 oder ist $i = 0$, und ist t nicht sichtbar, so gilt das folgende Lemma. Dabei benutzen wir den Begriff der Nachbarschaft von Wandkanten, der zu Beginn des Abschnitts 4.3.3 eingeführt wurde. Benachbart heißen Wandkanten in $vis(p)$, wenn sie an eine gemeinsame Eingangskante f grenzen. Die Wandkante, deren Schnittpunkt mit f näher an der Position p liegt, liegt unterhalb der anderen Wandkante. Der *rechte* bzw. *linke* Nachbar einer Wandkante ergibt sich aus der Sicht des Punktes p.

Lemma 4.3.16 *Sei c die Sehne zwischen den alten Orientierungspunkten v'_L und v'_R. Für alle Wandkanten e, die nicht auf der gleichen Seite von c liegen wie die Strecke l_{i-1}, gilt:*

- *Gehört e zu L, so liegt e nicht unterhalb seines linken Nachbarn.*
- *Gehört e zu R, so liegt e nicht unterhalb seines rechten Nachbarn.*

Beweis: Wäre beispielsweise die erste Eigenschaft nicht erfüllt, so wäre eine Situation wie in Abbildung 4.6 auf Seite 18 gegeben, wobei das Ziel t hinter der Eingangskante f liegt. In diesem Fall wäre e'_1 eine Kante in L, die unterhalb ihres linken Nachbarn liegt. Daß dies nicht möglich ist, sieht man daran, daß in diesem Fall von den Punkten hinter f' kein Teil des Streckenzugs R sichtbar ist. Dies gilt für jede Lage von f und f' und widerspricht der Tatsache, daß P eine Straße ist. $\square$

Aufgrund von Lemma 4.3.16 gibt es nur drei mögliche Fälle für die Eingangskanten im Sichtbarkeitspolygon. Entweder liegen alle Wandkanten, die an eine Eingangskante grenzen, unterhalb ihrer linken Nachbarn, oder alle Wandkanten liegen unterhalb ihrer rechten Nachbarn, dann bilden die Endpunkte der am weitesten links bzw. rechts liegenden Eingangskante die neuen Orientierungspunkte v_L und v_R. Oder es gibt sowohl Wandkanten, die unterhalb ihres rechten Nachbarn liegen, als auch solche, die unterhalb ihres linken Nachbarn liegen. In diesem Fall gibt es genau einen Streckenzug aus Wandkanten, der oberhalb seiner beiden Nachbarn liegt. Als v_L und v_R werden die Endpunkte dieser beiden benachbarten Wandkanten gewählt.

Es ist klar, daß bei dieser Wahl $v_L \in L$ und $v_R \in R$ gilt und das Ziel t hinter der Strecke von v_L nach v_R liegen muß.

Fall 2: Verläuft die Strecke l_{i-1} wie im Fall 2 aus Abbildung 4.10, muß unterschieden werden, aufgrund welches Ereignisses diese Strecke endete. Wie bereits oben angedeutet, wird ein Punkt q auf der Verbindungslinie der alten Orientierungsmarken v'_L und v'_R angestrebt. Dabei vergrößert sich der Streckenzug c, der hinter v'_L und v'_R sichtbar ist. Folgende Ereignisse sind an einem Punkt p' möglich

1. *Der wachsende Streckenzug c erreicht v'_L oder v'_R*

   In diesem Fall bleibt oberhalb von v'_L und v'_R nur noch eine Eingangskante im neuen Sichtbarkeitspolygon $vis(p')$ bestehen. Die Endpunkte dieser Kante bilden die neuen Orientierungspunkte v_L und v_R.

2. *Ein Endpunkt des Streckenzugs c und eine neue Eingangskante f erscheint.*

   Hier werden die Endpunkte von f als neue Orientierungspunkte gewählt.

3. *Die Eingangskante links von v_R wird von einer neu auftretenden Ecke v, einem Punkt des Polygons, zweigeteilt.*

   Wir setzen $v_R = v$ und v_L bleibt gleich. Zu diesem Fall gibt es noch den symmetrischen Fall, in dem die Eingangskante rechts von v_L geteilt wird.

Ähnlich wie oben läßt sich für alle Fälle zeigen, daß für die neuen Orientierungspunkte $v_L \in L$ und $v_R \in R$ gilt und das Ziel t oberhalb der Verbindungslinie von v_L und v_R liegt.

Satz 4.3.17 *Sei P eine Straße mit n Kanten. Dann findet das dargestellte Verfahren einen Weg von s nach t.*

Der Satz wird bewiesen, indem man zeigt, daß stets $v_L \in L$ und $v_R \in R$ gilt und daß bei jedem Halt des Algorithmus mindestens einer der beiden Orientierungspunkte weiter nach hinten verschoben wird. Es kann weiter gezeigt werden, daß der erzeugte Pfad nur $O(n)$ Liniensegmente enthält.

Analyse

Für die Kompetitivität des Algorithmus ist es wichtig, genau darzulegen, wie im Fall 2 der Punkt q auf der Verbindungsstrecke von v_L und v_R gewählt wird. Klein zeigt, daß z.B. die Wahl des Mittelpunkts der Strecke nicht zu einem kompetitiven Algorithmus führt. Das gleiche gilt für die Wahl des näheren der beiden Punkte v_L und v_R.

Klein schlägt daraufhin ein Verfahren vor, das den lokalen Umweg minimiert. q wird dabei so gewählt, daß

$$d(v_L, q) = \frac{d(p, v_L) - d(p, v_R) + d(v_L, v_R)}{2}$$

gilt. Werden auf dem Weg zum Punkt q weitere Punkte hinter v_L oder v_R sichtbar, ohne daß eines der oben beschriebenen Ereignisse 1–3 eintritt, so kann der Punkt q entsprechend angepaßt werden.

Die Analyse des Verfahrens soll hier nicht genau wiedergegeben werden. Der wesentliche Teil des Algorithmus ist der Fall 2, wo die Entscheidung zwischen den beiden Punkten v_R und v_L hinausgezögert werden muß, bis mehr Informationen vorliegen. Strecken, die im Fall 1.1 oder 1.2 gewählt werden, verlaufen ohnehin auf dem kürzest möglichen Pfad.

Klein zeigt eine obere Schranke von $1 + \frac{3}{2}\pi$, d.h. ungefähr $5,72$ für die Kompetitivität des Algorithmus bezüglich der L_2 Norm. Er vermutet jedoch, daß die tatsächliche Kompetitivität deutlich niedriger liegt, da kein Beispiel konstruiert werden konnte, in dem das Verfahren eine Kompetitivität von mehr als $1,8$ erreicht.

Ein weiterer Algorithmus

Der Algorithmus von Klein wurde von Kleinberg [Kle94] etwas verändert. Dadurch gelingt ihm eine günstigere Analyse, er zeigt eine Kompetitivität von $2\sqrt{2}$, d.h. ungefähr $2,83$. Es ist jedoch nicht klar, ob der veränderte Algorithmus tatsächlich eine bessere Kompetitivität erzielt.

Der Unterschied zwischen den Algorithmen besteht in der Bearbeitung des Falles 2 aus Abbildung 4.10. Der Algorithmus von Kleinberg wählt zunächst einen beliebigen Punkt q auf der Strecke von v_L nach v_R aus. Zusätzlich zu den beschriebenen Ereignissen 1–3, bei deren Auftreten das Verfahren sich neu orientiert, hält das Verfahren von Kleinberg auch an, falls die Linie, die orthogonal zur Bewegungsrichtung des Roboters durch dessen Position verläuft, entweder v_R oder v_L trifft.

Geschieht dies, so liegt eine Situation vor, wie sie in Abbildung 4.11 dargestellt ist. Dabei nehmen wir o.B.d.A. an, daß der Pfad auf die Höhe von v_L gelangt ist. Wählen wir das Koordinatensystem so, daß der bisherige Pfad des Roboters nördlich gerichtet ist und die jetzige Position p des Roboters im Ursprung liegt. Dann liegt im Bereich links oberhalb von p, d.h. im zweiten Quadranten, ein zusammenhängender Streckenzug c aus Wandkanten.

Der Algorithmus verfolgt nun einen Pfad, der in einem Winkel von 45 Grad zwischen v_L und v_R verläuft, dieser Pfad ist in der Abbildung punktiert wiedergegeben. Falls nötig

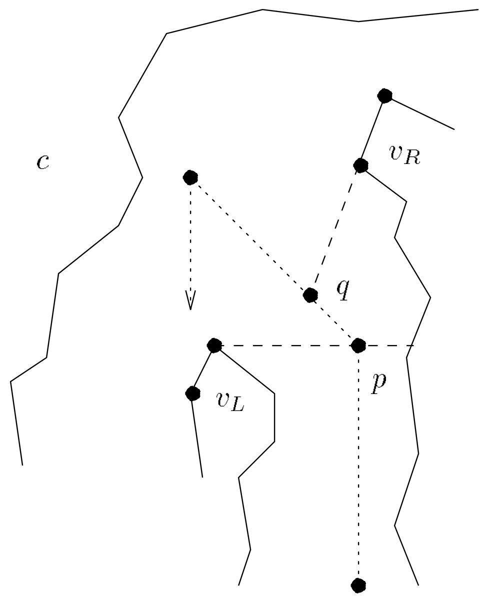

Abbildung 4.11: Der Algorithmus von Kleinberg wählt einen diagonalen Pfad.

werden dabei wie am Punkt q in der Abbildung die Orientierungsmarken v_L und v_R aktualisiert. Dies geschieht solange, bis eines von zwei Ereignissen eintritt.

1. *Der Roboter erreicht die x-Koordinate von v_L oder die y-Koordinate von v_R.*

   Nehmen wir an, der Roboter erreicht die x-Koordinate von v_L. In diesem Fall kann das Ziel nicht rechts von v_R liegen, da dann der Streckenzug c zu L gehören würde und somit von den Punkten hinter v_L kein Punkt aus R sichtbar wäre. Der Roboter kann somit in Richtung v_L neue Orientierungsmarken wählen.

   Der andere Fall ist analog.

2. *Der Teil des Polygons hinter v_L oder v_R wird vollständig sichtbar.*

   Hier läuft der Roboter zum jeweils anderen Orientierungspunkt, da dieser dann auf dem kürzesten Pfad liegt.

Zur Analyse dieses Algorithmus betrachtet man eine Phase, die aus dem Ansteuern eines Punktes q auf der Verbindungsstrecke von v_L und v_R und einem anschließenden Abbiegen in einem Winkel von 45 Grad besteht. Indem man den gelaufenen Umweg des Verfahrens abschätzt erhält man die angegebene Kompetitivität von $2\sqrt{2}$.

Interessant ist, daß der Algorithmus von Kleinberg für iso-orientierte Polygone eine noch wesentlich bessere Kompetitivität erzielt.

Satz 4.3.18 *Der Online-Algorithmus zur Suche in unbekannten Straßen erreicht für iso-orientierte Straßen eine Kompetitivität von $\sqrt{2}$ bezüglich der euklidischen Metrik. Bezüglich der L_1 Metrik ist der kompetitive Faktor 1, d.h. der Algorithmus findet stets einen optimalen Pfad.*

Untere Schranke

Für das Problem der Suche in Straßen läßt sich leicht eine untere Schranke angeben. Dazu betrachtet man die in Abbildung 4.12 dargestellte Situation.

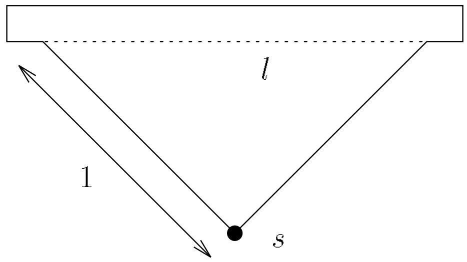

Abbildung 4.12: Eine untere Schranke für die Suche in Straßen.

Das Ziel kann hinter der kleinen Biegung rechts oder links liegen und das optimale Verfahren läuft zunächst gerade bis zur Sichtlinie l und dann zum Ziel. Der zurückgelegte Weg hat damit eine Länge von $\sqrt{2}$ im Gegensatz zum kürzesten Pfad, der eine Länge von 1 hat.

Aus dieser Beobachtung ergibt sich eine untere Schranke von $\sqrt{2}$, bzw. $1,41$ für die Suche in Straßen. Diese Schranke konnte bisher nicht verbessert werden, so daß eine Lücke zu den gezeigten oberen Schranken für allgemeine Polygone bleibt.

Iso-orientierte H-Straßen

Daß es neben Straßen weitere Klassen von Polygonen gibt, die eine kompetitive Suche ermöglichen, zeigen Datta und Icking [DI94].

Definition 4.3.19 *Ein Polygon P heißt* H-Straße*, wenn von jedem Punkt auf dem den Rand von P eine horizontale LR-Sehne sichtbar ist.*

Der Begriff der H-Straßen stellt eine Verallgemeinerung von Straßen dar.

Lemma 4.3.20 *Die Klasse der H-Straßen enthält die Klasse der Straßen und ist strikt größer.*

Abbildung 4.13 zeigt eine H-Straße, die keine Straße ist. Das Ziel kann sich an t_1 oder an t_2 befinden. Befindet es sich beispielsweise an t_1, so ist von $t_2 \in R$ kein Punkt $p \in L$ sichtbar.

Datta und Icking geben nun einen Online-Algorithmus an, der für die eingeschränkte Klasse der *iso-orientierten* H-Straßen eine Kompetitivität von 9 erreicht.

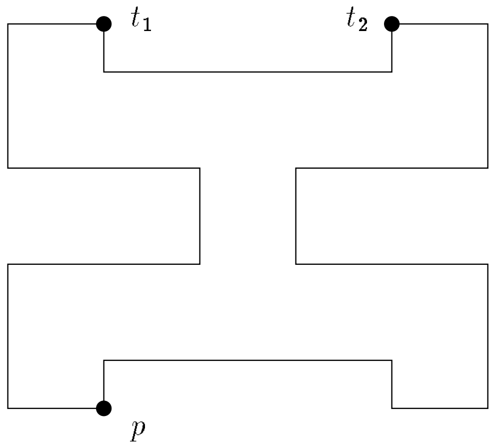

Abbildung 4.13: Ein Beispiel einer H-Straße, die keine Straße ist.

Der Algorithmus

In Abschnitt 4.3.3 haben wir bereits einen Landmark bezüglich eines Punktes definiert. Wir erweitern nun diesen Begriff.

Definition 4.3.21 *Ein konkaver Randpunkt l eines Polygons P heißt* Landmark bezüglich einer Sehne *c, falls l von einem Punkt auf c sichtbar ist, aber eine der an l grenzenden Kanten von keinem Punkt von c sichtbar ist. Diese Kante v heißt die* verdeckte Kante *von l. Die* erweiterte Sehne *von l ist die maximale Sehne, die v enthält, d.h. die erweiterte Sehne enthält beide induzierte Sehnen von v.*

Eine Sehne c in einem iso-orientierten Polygon P unterteilt P in zwei oder drei Gebiete, die wir als *Regionen* von P bezeichnen.

Der Algorithmus zur Suche in H-Straßen benutzt das Verfahren von Baeza-Yates *et al.* für die 2-Wege Suche, das wir in Abschnitt 4.3.1 vorgestellt haben. Die 2-Wege Suche wird dabei stets auf einer horizontalen LR-Sehne c solange durchgeführt, bis ein Landmark bezüglich c oder der Zielpunkt t auf einer Seite von c sichtbar wird. Die einzige Ausnahme bildet der Beginn des Algorithmus, hier wird eine 2-Wege Suche auf der Kante, die s enthält, durchgeführt.

Wir formulieren nun den Algorithmus.

Algorithmus H-Street
Sei c die Sehne, auf der s liegt.
Führe eine 2-Wege Suche entlang c aus, bis ein Landmark
  bezüglich c oder der Zielpunkt t sichtbar wird.
while t ist nicht sichtbar **do**
  Sei c die erweiterte Sehne von l.
  Laufe auf einem xy-monotonen Pfad zum nächsten
    Punkt auf c.

Führe eine 2-Wege Suche entlang c aus, bis ein Landmark
bezüglich c, der nicht in G liegt, oder der Zielpunkt t
sichtbar wird.

end while;

Laufe auf einem xy-monotonen Pfad zum Zielpunkt t.

Satz 4.3.22 *Der Algorithmus* H-STREET *erreicht bezüglich der L_1-Metrik eine Kompetitivität von 9 für die Suche in iso-orientierten Straßen.*

Datta und Icking zeigen weiterhin, daß die Kompetitivität des Algorithmus bezüglich der euklidischen Metrik 9,06 beträgt, also kaum schlechter ist als die Kompetitivität bezüglich der L_1-Metrik.

Untere Schranke

Die Kompetitivität des dargestellten Online-Algorithmus **H-STREET** ist optimal für die Suche in iso-orientierten H-Straßen, wie das in Abbildung 4.14 gezeigte Polygon P zeigt. P ist eine H-Straße, unabhängig davon, in welchem Eingang das Ziel t plaziert wird. Algorithmen, die in dieser Straße suchen, können eindeutig Algorithmen für das 2-Wege Suchproblem aus Abschnitt 4.3.1 zugeordnet werden. Da die Seitenwege beliebig lang gemacht werden können, überträgt sich auch die untere Schranke von 9 für dieses Problem.

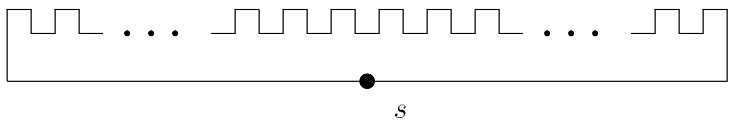

Abbildung 4.14: Eine untere Schranke für die Suche in H-Straßen.

4.3.4 Kartographieren von Polygonen

Neben der Suche nach einem Ziel t gibt es ein anderes interessantes Navigationsproblem, das *Kartographieren* einer unbekannten Umgebung. Hier soll nicht ein bestimmter Zielpunkt erreicht werden, sondern eine Karte der Umgebung erstellt werden.

Untersucht man das Kartographierproblem für Polygone, so ist zu einem Startpunkt s ein Rundweg in P gesucht, der bei s beginnt und endet und von dem aus alle Punkte auf dem Rand des Polygons sichtbar sind.

Vereinfacht gesagt unterscheidet sich das Kartographierproblem vom Suchproblem dadurch, daß alle Seitenwege des Polygons erforscht werden können. Es gibt keinen kürzesten Pfad, auf dem der Roboter bleiben muß. Allerdings ist es dennoch nicht immer einfach, einen kurzen Pfad zur Kartographierung eines Seitenweges zu finden.

Allgemeine Polygone

Ob es zum Kartographieren von allgemeinen Polygonen ein Online-Verfahren mit konstanter Kompetitivität gibt, ist derzeit noch eine offene Fragestellung.

Ein interessantes Teilproblem, das beim Kartographieren von allgemeinen Polygonen durch Online-Verfahren auftaucht, wurde isoliert untersucht. Befindet sich ein Verfahren an einem Punkt p, und $vis(p)$ enthält nur eine Eingangskante, d.h. genau ein Landmark ist sichtbar, hinter dem noch unbekanntes Gebiet liegt, dann muß das Verfahren kompetitiv *um die Ecke sehen.*

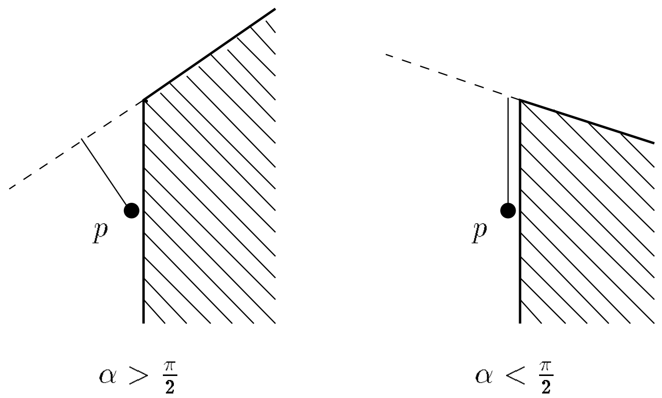

Abbildung 4.15: Situation für spitze und stumpfe Winkel mit kürzestem Pfad.

Icking, Klein und Ma untersuchen dieses Problem [IKM92]. Sie nehmen dabei an, daß sich der Roboter zu Beginn auf der Kante befindet, die an der Ecke endet. Der Roboter hat dann die Wahl, auf die Ecke zuzugehen oder sich seitlich zu bewegen, um eine bessere Übersicht zu erhalten. Schließlich muß er zu einem Punkt gelangen, von dem aus die Kante hinter der Ecke sichtbar ist.

Ist der Winkel α der Ecke unbekannt, so ist das beste Verfahren nicht trivial, da der kürzeste Pfad abhängig von diesem Winkel ist und daher nicht gewählt werden kann. Bei stumpfen Winkeln ist eine seitliche Bewegung günstiger, bei spitzen Winkeln eine vorwärtsgerichtete, wie in Abbildung 4.15 dargestellt ist.

Icking, Klein und Ma beschreiben eine optimale kompetitive Strategie, die darin besteht, sich dem Eckpunkt auf einer gebogenen Linie zu nähern, deren Verlauf in Abbildung 4.16 annähernd wiedergegeben ist. Sobald die Kante auf der anderen Seite der Ecke sichtbar wird, kann angehalten werden. Der kompetitive Faktor beträgt ungefähr $1,21$. Die Kurve wird als Lösung einer Differentialgleichung erhalten. Die Autoren geben auch eine schnell berechenbare Näherung mit ähnlicher Kompetitivität an.

Iso-orientierte Polygone

Deng, Kameda und Papadimitriou [DKP91] geben ein deterministisches Online-Verfahren zur Kartographierung von iso-orientierten Polygonen an.

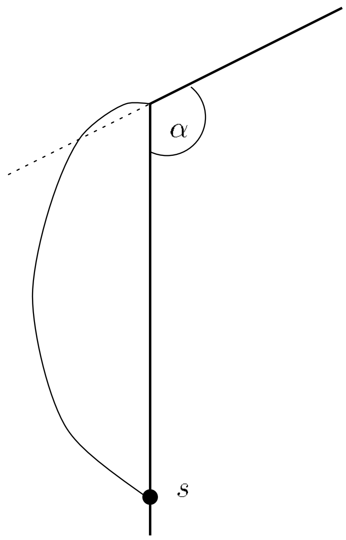

Abbildung 4.16: Der ungefähre Verlauf der optimalen Kurve.

Während des Kartographierens merkt sich der Algorithmus ein Polygon K, das die Teile von P, die der Roboter auf seinem Pfad bereits gesehen hat, enthält. Wie das Sichtbarkeitspolygon zu einem Punkt enthält K Wandkanten und Eingangskanten.

Algorithmus Iso-Exploration
Setze $K := vis(s)$.
Sei C die Kette aus Wandkanten in K, die s enthält.
Sei l der Endpunkt von C im Uhrzeigersinn.
while $K \neq P$ **do**
  Sei p die aktuelle Position des Algorithmus.
  case
    1. *l ist sichtbar von p und Endpunkt einer Kante e.*
      Laufe orthogonal zur von l verdeckten Kante v, bis
      v oder eine von v induzierte Sehne erreicht ist.
      Falls dabei der Polygonrand erreicht wird, laufe
      parallel zu v in Richtung l, bis eine Richtung
      orthogonal zu v wieder möglich ist.
    2. *l ist sichtbar von p und innerer Punkt einer Kante e.*
      In diesem Fall gibt es einen Landmark b, der die Sicht
      auf einen Teil der Kante e versperrt. Wie in Fall 1
      wird ein Pfad zu b oder zur von b induzierten
      Sehne erzeugt.
      Falls das nun sichtbare Gebiet hinter b noch nicht
      vollständig sichtbar und l noch nicht erreicht ist, laufe
      parallel zur von b verdeckten Kante, bis l
      sichtbar wird.

3. *l ist nicht sichtbar von p.*
   Hier unterscheidet man zwei Fälle. Ist l Endpunkt einer Kante e, so ist ein kürzester Pfad zur von l verdeckten Kante oder deren induzierter Sehne bekannt, entlang dem gelaufen wird.
   Sonst ist l innerer Punkt einer Kante e, und es gibt wieder einen Punkt b, der einen Teil von e verdeckt vom letzten Punkt aus, an dem l sichtbar war.
   Laufe in diesem Fall auf einem kürzesten Pfad zu b oder zur von b induzierten Sehne.

end case

Aktualisiere die bisherige Karte K, sowie den Kantenzug C und dessen Endpunkt l.

end while;

Gehe auf dem kürzesten Pfad zurück nach s.

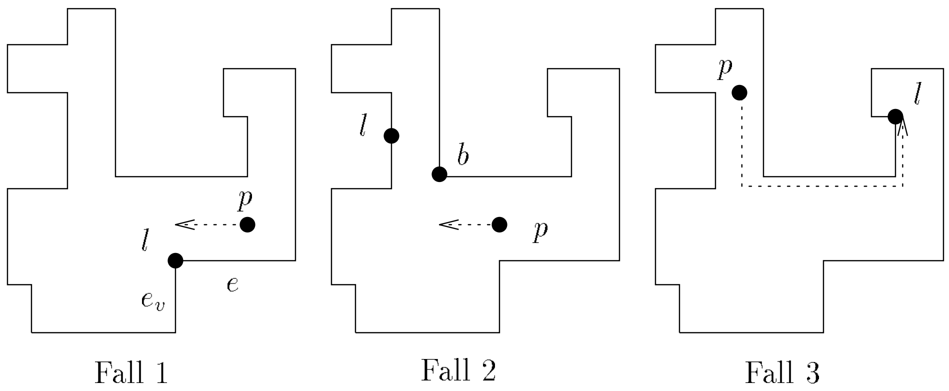

Abbildung 4.17: Die drei Fälle, die während der Ausführung von Iso-Exploration auftreten können.

Satz 4.3.23 *Der Online-Algorithmus* Iso-Exploration *zum Kartographieren eines isoorientierten Polygons ist 2-kompetitiv. Falls der Punkt s auf dem Polygonrand liegt, ist er sogar 1-kompetitiv, d.h. der erzeugte Pfad ist optimal.*

Zum Beweis zeigen die Autoren, daß der Pfad, der von Iso-Exploration erzeugt wird, die gleiche Länge hat, wie der Pfad eines Offline-Verfahrens, das die wesentlichen Sehnen des Polygons im Uhrzeigersinn besucht.

Deng, Kameda und Papadimitriou zeigen weiterhin, daß das Verfahren Iso-Exploration 6-kompetitiv ist, falls das Polygon P nicht am Punkt s, sondern an einem beliebigen Punkt t wieder verlassen werden soll.

Ein randomisiertes Verfahren

Randomisierung kann die Kompetitivität von Online-Algorithmen für dieses Problem erhöhen. Kleinberg gibt ein randomisiertes Verfahren für iso-orientierte Polygone an, das 5/4-kompetitiv ist [Kle94].

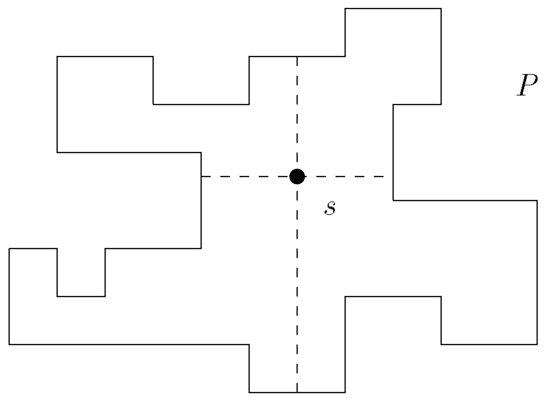

Abbildung 4.18: Das Polygon P wird durch eine von vier möglichen Nadeln, die gestrichelt eingezeichnet sind, erweitert.

Das randomisierte Verfahren benutzt den deterministischen Algorithmus und die Tatsache, daß dieser für den Fall, daß s auf dem Polygonrand liegt, optimal ist. Denkt man sich das Polygon P wie in Abbildung 4.18 um eine *Nadel* vom Rand nach s erweitert, so erhält man ein Polygon, für das s auf dem Rand liegt. Es gibt vier Möglichkeiten, die Nadel parallel zu einer der Koordinatenachsen einzufügen. Bezeichnen wir die vier sich daraus ergebenden Polygone mit P_i für $i \in \{1, 2, 3, 4\}$.

Der Online-Algorithmus Random-Exploration von Kleinberg wählt zufällig eines der Polygone P_1 bis P_4 aus und erzeugt einen Pfad zur Kartographierung dieses Polygons nach dem Verfahren Iso-Exploration von Deng, Kameda und Papdimitriou. Dieser Pfad kartographiert auch das Polygon P.

Satz 4.3.24
Das Verfahren Random-Exploration *zum Kartographieren iso-orientierter Polygone ist* 5/4*-kompetitiv.*

Kleinberg zeigt nicht, daß die erzielte Kompetitivität optimal ist, der kompetitive Faktor von 5/4 ist jedoch optimal für *deterministische* Online-Algorithmen.

Satz 4.3.25 *Für* $c < 5/4$ *gibt es keinen c-kompetitiven deterministischen Algorithmus zum Kartographieren von iso-orientierten Polygonen.*

Beweis: Betrachten wir das Polygon P in Abbildung 4.19. Die Höhe und Breite des Polygons sei 2, die Abmessungen der kurzen Kanten im Verhältnis dazu sehr klein. An drei der vier Punkte p_i werden wir eine weitere Ecke anfügen, so daß zur Kartographierung der

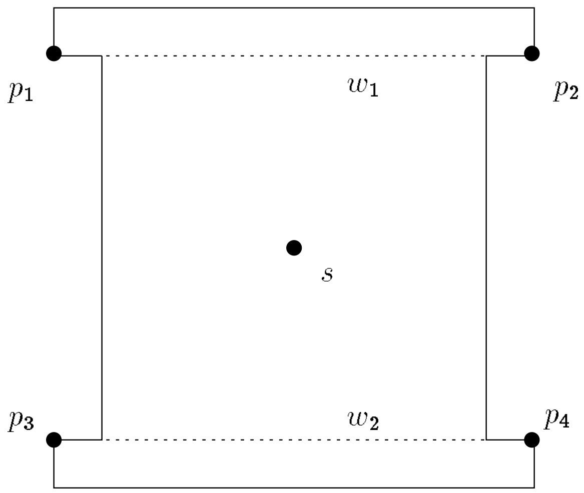

Abbildung 4.19: Eine untere Schranke für die Kartographierung von iso-orientierten Polygonen.

Punkt p_i besucht werden muß. Jeder Pfad zum Kartographieren von P muß eine der beiden wesentlichen Sehnen w_1 und w_2 zuerst schneiden. Nehmen wir o.B.d.A. an, das Verfahren schneidet zuerst w_1. In diesem Fall fügen wir an p_1 und p_2 Ecken ein. Das Verfahren muß nun p_1 und p_2 besuchen. Nehmen wir an, das Verfahren besucht p_1 vor p_2 und vor der zweiten wesentlichen Sehne w_2. Die anderen Fälle können analog behandelt werden. Nun fügen wir einen zusätzliche Ecke an p_3, aber nicht an p_4 ein. Das Verfahren benötigt nun mindestens einen Pfad der Länge 8, um p_2 und p_3 zu besuchen. Die Gesamtlänge ist damit größer gleich 10. Dem gegenüber steht eine optimaler Pfad der Länge 8, der zunächst p_2, dann p_1 und p_3 besucht. $\square$

4.3.5 Szenen mit Hindernissen

Wirkliche Szenen lassen sich im allgemeinen nur unvollständig durch Polygone modellieren. Die wichtigste Einschränkung ist die, daß hier keine Hindernisse auftreten können. Betrachtet man Szenen, in denen Hindernisse enthalten sind, so ist es zunächst wichtig, zwischen konvexen und allgemeinen Hindernissen zu unterscheiden. Schränkt man die Hindernisse nicht auf konvexe Objekte ein, so erhält man das Problem der Suche in einem Labyrinth.

Wir geben hier einige Ergebnisse zum Suchen und Kartographieren in diesem allgemeineren Modell an.

Suchen in Szenen mit konvexen Hindernissen

Für Algorithmen zur Suche nach einem Zielpunkt t in einer Szene mit konvexen Hindernissen wird in der Literatur stets vorausgesetzt, daß die Koordinaten des Zielpunkts bekannt sind, nicht jedoch Umriß und Lage der Hindernisse. Falls die Koordinaten des Zielpunkts nicht bekannt sind, entspricht das Problem dem Problem der Kartographierung der Umgebung, da das Ziel immer am letzten noch nicht untersuchten Punkt versteckt sein kann. Dieses Problem werden wir später in diesem Abschnitt untersuchen.

Der Fall von bekannten Zielkoordinaten wurde unter kompetitiven Gesichtspunkten zuerst von Papadimitriou und Yannakakis [PY91] untersucht. Hier wird gezeigt, daß im allgemeinen kein Algorithmus mit konstanter Kompetitivität möglich ist.

Satz 4.3.26 *Für das Problem der Suche in Szenen mit konvexen Hindernissen gilt eine untere Schranke von $\Omega(\sqrt{n})$ für die Kompetitivität jedes Online-Algorithmus. Diese Schranke gilt auch dann, wenn die Hindernisse iso-orientierte Rechtecke sind.*

Beweis: Betrachten wir die Szene in Abbildung 4.20. Das Ziel t befindet sich im Abstand n auf gleicher Höhe wie der Startpunkt s. Zwischen Start und Ziel liegen n Hindernisse, die eine Breite von $1/2$ und eine Höhe von n haben. Die Hindernisse sind so angeordnet, daß der Online-Algorithmus, nachdem er ein Hindernis umrundet hat, auf die Mitte des nächsten Hindernisses trifft. Die Länge des vom Algorithmus erzeugten Pfades liegt damit in $\Omega(n)$.

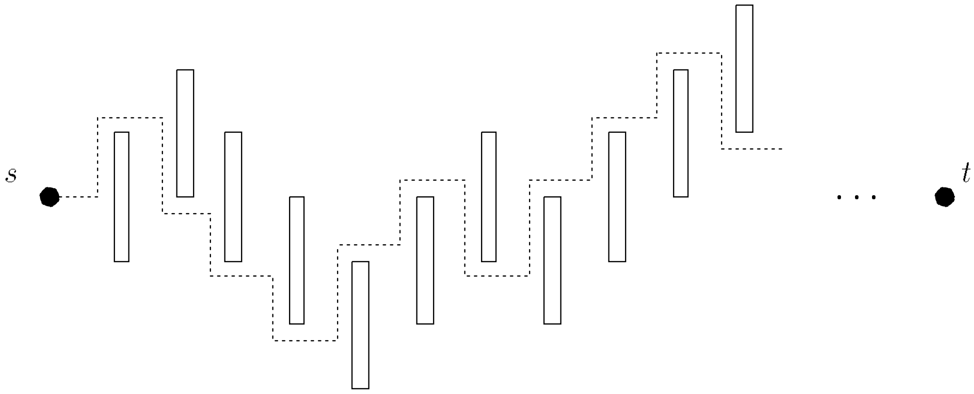

Abbildung 4.20: Eine untere Schranke für die Suche in Szenen mit Hindernissen.

Auf der anderen Seite gibt ein $y < n^{3/2}$, so daß eine horizontale Linie auf Höhe y höchstens $\sqrt{n}$ Hindernisse schneidet. Denn im ungünstigsten Fall liegen $\sqrt{n}$ Schichten mit jeweils $\sqrt{n}$ Hindernissen übereinander, diese haben eine Gesamthöhe von $n^{3/2}$. Der Pfad, der zunächst bis Höhe y läuft, dann auf dem Weg nach rechts die höchstens $\sqrt{n}$ Hindernisse umrundet und wieder nach unten bis zum Zielpunkt t läuft, hat eine Länge, die in $O(n^{3/2})$ liegt.

Damit der Online-Algorithmus keinen günstigeren Pfad durch die Szene finden kann, muß man die Sichtbarkeit zwischen den Hindernissen einschränken. Dies ist durch $O(n)$ zusätzliche Hindernisse möglich. $\square$

Die Autoren untersuchen daraufhin den Fall von iso-orientierten Quadraten. Eine leichte Modifikation des Beweises von Satz 4.3.26 zeigt die folgende untere Schranke.

Satz 4.3.27 *Für $c < 3/2$ gibt es keinen deterministischen c-kompetitiven Online-Algorithmus für die Suche in Szenen mit quadratischen Hindernissen.*

Zwei Spezialfälle dieses Problems führen auf eine kompetitive Strategie. Befinden sich Start und Ziel auf der gleichen Höhe, so ist es günstig, beim Auftreffen auf ein Hindernis stets zur näheren Seite des Quadrats auszuweichen und von dort aus den Pfad weiterzuführen. Dieses Verfahren NEAREST-CORNER ist 2-kompetitiv. Die ungünstigste Szene besteht aus einer Reihe von Quadrate der Seitenlänge 1, die jeweils um die Länge $\frac{1}{2} - \epsilon$ nach oben verschoben sind.

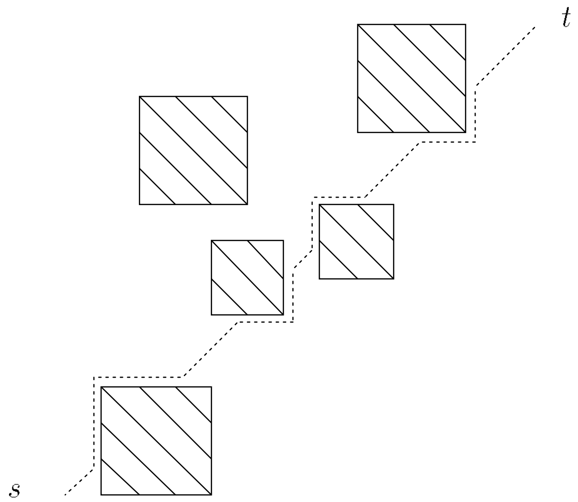

Abbildung 4.21: Start und Ziel liegen in einem Winkel von 45 Grad.

Weiterhin denke man sich Ziel und Start in einem Winkel von 45 Grad gelegen. Hier ist es günstig, bei Konfrontation mit einem Hindernis stets in einem stumpfen Winkel abzubiegen, siehe Abbildung 4.21. In diesem Fall ist es leicht nachzuweisen, daß der spezielle Algorithmus eine Kompetitivität von $\sqrt{2}$ erreicht.

Der gemischte Algorithmus MIXED läuft zunächst nach dem Verfahren NEAREST-CORNER in Richtung des Ziels, bis er einen Punkt erreicht, der in einem Winkel von 45 Grad zum Ziel steht. Nun wendet er die zweite Strategie für den Fall von 45 Grad an, bis er beim Ziel angelangt ist. Papadimitriou und Yannakakis zeigen die Kompetitivität dieses gemischten Verfahrens.

Satz 4.3.28 *Der Online-Algorithmus* MIXED *erreicht eine Kompetitivität von $\sqrt{26}/3$, also ungefähr $1,7$.*

Beweis: Der ungünstigste Fall bei der Ausführung von MIXED ist in Abbildung 4.22 dargestellt.

Ist n der Abstand von s und t, dann hat der Pfad, den der Algorithmus MIXED erzeugt, eine Länge $l(\phi)$ von $(\cos\phi + \sin\phi + 2x)d$. Dabei läßt sich x als $\frac{1}{3}(\cos\phi - \sin\phi)$ bestimmen,

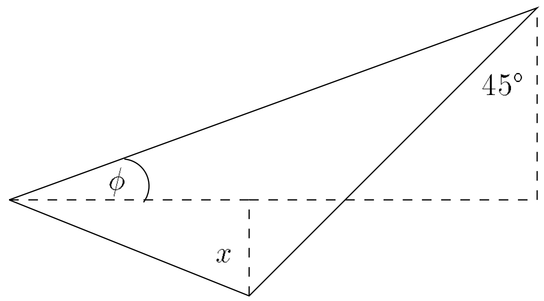

Abbildung 4.22: Der ungünstigste Pfad für den Algorithmus Mixed.

und als Länge des Pfades ergibt sich $l(\phi) = \frac{1}{3}(5\cos\phi + \sin\phi)d$. Dieser Ausdruck wird durch den Winkel $\phi^* = \arctan\frac{1}{5}$ maximiert, und es gilt $l(\phi^*) = \sqrt{26}d/3$. □

Für den Fall von Quadraten der Seitenlänge 1 geben die Autoren ein komplizierteres $\frac{3}{2}$-kompetitives Verfahren an, das somit die untere Schranke erreicht.

Blum *et al.* [BRS91] untersuchen zunächst einige Spezialfälle dieses Problems und greifen dann das allgemeinere Problem für iso-orientierte Rechtecke noch einmal auf. Sie geben einen Algorithmus an, der für Szenen mit iso-orientierten Rechtecken $O(\sqrt{n})$-kompetitiv ist. Die Kompetitivität hängt hier vom Abstand n zum Zielpunkt ab. Die rechteckigen Hindernisse müssen dabei mindestens so groß sein, daß sich ein Einheitskreis einschreiben läßt. Damit erreichen sie die untere Schranke aus Satz 4.3.27.

Dieses Ergebnis verallgemeinern Blum *et al.* in der gleichen Arbeit auf den Fall von iso-orientierten Quadern im dreidimensionalen Raum. Hier beträgt der kompetitive Faktor $O(n^{2/3})$.

Eine weitere Verallgemeinerung des Problems stammt von Blum und Chalasani [BC93]. Sie untersuchen, welche Veränderungen sich ergeben, wenn der Roboter nicht nur einmal von s nach t laufen soll, sondern mehrere Male. Dabei kann er die während der ersten Wege gemachten Erfahrungen nutzen und somit eventuell einen günstigeren Weg finden. Die Autoren geben einen deterministischen Algorithmus für Szenen mit rechteckigen Hindernissen an, dessen mittlere Weglänge nach der k-ten Fahrt $O(\sqrt{d/k})$ beträgt, wobei d die optimalen Weglänge von s nach t bezeichnet. Sie zeigen auch, daß dies die beste mittlere Kompetitivität ist, die ein deterministischer Algorithmus erreichen kann.

Suchen in allgemeine Szenen

Blum, Raghavan und Schieber [BRS91] untersuchen auch den Fall nicht-konvexer Hindernisse. Für diesen Fall gibt es kein kompetitives Verfahren, wenn die Kompetitivität nicht von der Anzahl der Kanten der Hindernisse abhängig gemacht wird. Denn in einem solchen *Labyrinth* kann der Online-Algorithmus im Gegensatz zum Offline-Gegenspieler nicht sofort den richtigen Weg wählen, sondern muß unter Umständen sehr viele andere Wege ausprobieren. Ist m die Anzahl der Hindernisseiten, so gilt der folgende Satz.

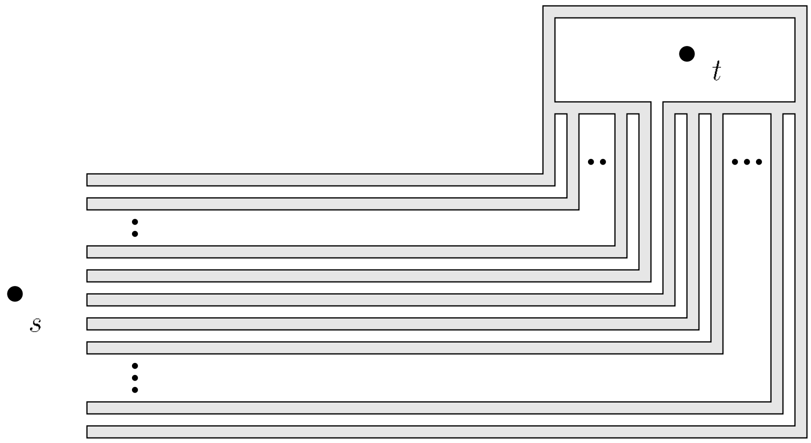

Abbildung 4.23: Eine untere Schranke für das Suchen im Fall von allgemeinen Hindernissen.

Satz 4.3.29 *Kein randomisierter Online-Algorithmus für die Suche nach einem Objekt in einer Szene mit nicht-konvexen Hindernissen kann eine Kompetitivität kleiner als $m/8$ erreichen.*

Beweis: Wir betrachten die Szene, die in Abbildung 4.23 dargestellt ist. Hier gibt es $m/8$ Wege, die von s zum Ziel t führen könnten. Diese sind jedoch bis auf einen Weg versperrt. Ein randomisierter Algorithmus muß durchschnittlich mindestens $(m/8)/2$ Wege ausprobieren, bis er den richtigen Weg findet. Da er dazu jedesmal mindestens eine Distanz von $2d(s,t)$ laufen muß, folgt die Behauptung. □

Blum, Raghavan und Schieber geben einen randomisierten Online-Algorithmus zur Suche in Szenen mit beliebigen Hindernissen mit m Kanten an, der eine Kompetitivität von $50{,}4m$ erzielt.

Kartographieren von Szenen mit Hindernissen

Wie im Fall von Polygonen kann auch für Szenen mit Hindernissen das Problem der Kartographierung betrachtet werden. Gesucht ist ein Pfad, von dem aus alle Hindernisseiten sichtbar sind.

Dieses Problem untersuchen Kalyasundaram und Pruhs [KP92]. Sie betrachten Szenen, die aus k konvexen Hindernissen in der Ebene bestehen. Die Hindernisse können eine beliebige Orientierung haben. Zusätzlich zur Anzahl ziehen sie das Seitenverhältnis α der Hindernisse in Betracht, das Verhältnis des größten einschreibbaren Kreises zum kleinsten Kreis, der das Hindernis enthält. Sie zeigen eine untere Schranke von $\Omega(\min(k, \sqrt{k\alpha}))$ für die Kompetitivität von Verfahren für das Kartographierproblem und geben gleichzeitig einen Algorithmus an, der diese Schranke erreicht.

Dazu untersuchen Sie zunächst den folgenden einfachen Algorithmus NEAREST-NEIGHBOR. Jedes untersuchte Hindernis wird einmal vollständig umrundet. Daraufhin wird das nächste noch nicht untersuchte Hindernis ausgewählt und untersucht. Die Distanz zwischen den Hindernissen ist dabei durch die Länge der kürzesten hindernisfreien

Verbindungsgerade zwischen zwei Punkten der beiden Hindernisse erklärt. Da jedes Hindernis umrundet wird, ist danach das nächste Hindernis bekannt.

Kalyanasundaram und Pruhs zeigen, daß NEAREST-NEIGHBOR bereits eine Kompetitivität von $\alpha \log k \min(k, \sqrt{k\alpha})$ für das Kartographierproblem erzielt und verfeinern das Verfahren, bis es die genannte untere Schranke erreicht. Weiterhin untersuchen Sie den Fall, daß m Roboter gemeinsam ein Gebiet erkunden und zeigen hier, daß im Falle von Hindernissen von beschränkter Größe und beschränktem Seitenverhältnis ein $O(\sqrt{k}/m)$-kompetitives Verfahren möglich ist. In diesem Fall können die Roboter die Arbeit also optimal aufteilen.

Deng, Kameda und Papadimitriou [DKP91] untersuchen die Kartographierung iso-orientierter Szenen mit Hindernissen. Sie erweitern ihren Algorithmus ISO-EXPLORATION für iso-orientierte Polygone, den wir in Abschnitt 4.3.4 vorgestellt haben, auf Szenen, die aus einem begrenzenden iso-orientierten Polygon mit einer beschränkten Anzahl von iso-orientierten polygonalen Hindernissen besteht.

Dazu geben sie zunächst ein Verfahren zur Kartographierung eines einzelnen polygonalen Hindernisses an. Der kombinierte Algorithmus erreicht eine Kompetitivität von $O(k)$, wobei k die Anzahl der Hindernisse bezeichnet.

4.3.6 Unbekannte Graphen

Läßt man bei der Betrachtung eines unbekannten Terrains den geometrischen Aspekt ganz weg, erhält man das Problem der Kartographierung eines unbekannten Graphen. Auch das Problem der *Suche* in unbekannten Graphen zu betrachten, ist zunächst nicht sinnvoll, da der Online-Algorithmus dazu zu wenig Informationen hat und eine Breitensuche die einzig mögliche Strategie wäre. Wir werden jedoch ein verändertes Modell angeben, indem auch das Suchproblem untersucht werden kann.

Kartographieren

Beim Problem der Kartographierung von unbekannten Graphen ist anfangs ist nur der Startknoten und die Anzahl der abgehenden Kanten bekannt. Durch Bewegung innerhalb des Graphen erhält der Online-Algorithmus mehr Information. Zu jedem der bereits untersuchten Knoten werden die Kanten zu ebenfalls untersuchten Knoten und die Anzahl der noch nicht untersuchten abgehenden Kanten enthüllt. Der Graph wird stets als zusammenhängend vorausgesetzt.

Die meisten hier gezeigten Resultate entstammen der Arbeit von Deng und Papadimitriou [DP90]. Der folgende einfache Backtracking-Algorithmus erzielt für nicht gerichtete Graphen bereits gute Resultate.

Algorithmus GRAPH-SEARCH
v := Startknoten
while $v \neq$ Startknoten **or**
        Es gibt nicht untersuchte Kanten an v **do**

if Es gibt eine nicht untersuchte Kante e an v
  then Gehe entlang e, setze v auf den neuen Knoten.
  else Gehe zurück zum zuletzt besuchten Knoten,
  setze v auf diesen Knoten.
end if
end while;

Dieser Algorithmus ist 2-kompetitiv für das Kartographieren von Graphen, da jede Kante höchstens zweimal durchlaufen wird. Denn eine bereits untersuchte Kante wird nur gewählt, wenn der dahinter liegende Teilgraph bereits vollständig durchsucht wurde.

Dieses Verfahren läßt sich von ungerichteten Graphen auf gerichtete Euler-Graphen übertragen. Euler-Graphen haben die Eigenschaft, daß die Anzahl der ankommenden Kanten an jedem Knoten gleich der Anzahl der abgehenden Kanten ist. Folgt man in einem Euler-Graph so lange nicht untersuchten Kanten, bis dies nicht mehr möglich ist, so befindet man sich zuletzt am Startknoten wieder. Andernfalls gäbe es zu der vorher unbekannten Kante, auf der man zu diesem Knoten gelangt ist, eine weitere unbekannte ausgehende Kante.

Aufgrund dieser Eigenschaft läßt sich das sukzessive Zurückgehen zum letzten Knoten ersetzen durch ein erneutes Durchlaufen der gesamten Schlaufe, wobei an jedem Knoten der Algorithmus rekursiv aufgerufen wird für noch nicht untersuchte ausgehende Kanten.

Dieses neue Verfahren für gerichtete Euler-Graphen erzielt eine Kompetitivität von 2, und dies ist für diese Graphenklasse sogar optimal, wie man leicht am in Abbildung 4.24 dargestellten gewichteten Graphen sieht. Dabei sollen stets zuerst die teuren Kanten ausgewählt werden. Auf diese Wahl hat der Algorithmus keinen Einfluß. Die Gewichtung läßt sich durch mehrere Kanten hintereinander simulieren.

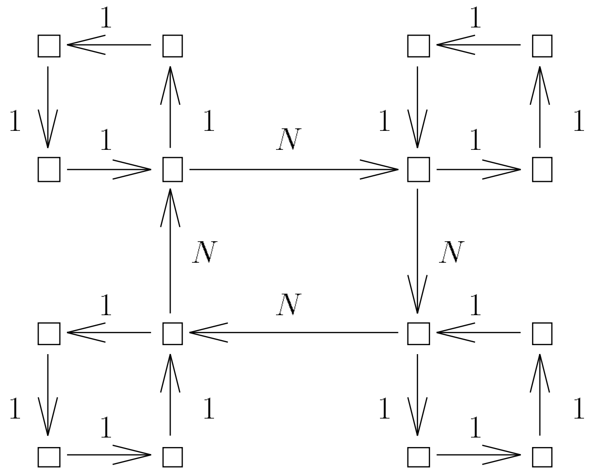

Abbildung 4.24: Ein ungünstiger Euler-Graph.

Betrachtet man allgemeine gerichtete Graphen, ist der Begriff der *Schwäche* h eines

Graphen $G = (V, E)$ hilfreich. Diese ist definiert als

$$h := \frac{1}{2} \sum_{v \in V} |\mathrm{outdegree}(v) - \mathrm{indegree}(v)|,$$

die minimale Anzahl von Kanten, die dem Graphen zu einem Euler-Graphen fehlen.

Für Graphen mit unbeschränkter Schwäche h kann es keinen kompetitiven Online-Algorithmus geben. Dies zeigt der folgende Satz.

Satz 4.3.30 *Für jedes $h > 0$ gibt es eine Familie von Graphen mit Schwäche h, so daß für jeden c-kompetitiven Online-Algorithmus gilt $c > h/(2 \log h)$.*

Für Graphen mit Schwäche 1 geben Deng und Papadimitriou ein optimales 4-kompetitives Verfahren an und ein einfacheres 9-kompetitives Verfahren. Dieses zweite Verfahren können Sie verallgemeinern für allgemeine Graphen mit durch h beschränkter Schwäche. Die Kompetitivität hängt jedoch exponentiell von der Schranke h ab, sie beträgt $3(h+1)^{2h+1}$. Die Schranke muß dem Algorithmus nicht bekannt sein.

Geschichtete Graphen

Ein Modell zur Suche in unbekannten Graphen untersuchten zuerst Papadimitriou und Yannakakis [PY91]. Hier handelt es sich tatsächlich um ein Suchproblem, da ein Zielpunkt t gegeben ist und die Kompetitivität gegen den kürzesten Pfad von s nach t gemessen wird.

Ein *geschichteter Graph* ist ein gewichteter, zusammenhängender Graph, dessen Knoten in Schichten $L_0 = \{s\}, L_1, \ldots, L_k$ eingeteilt sind, wobei die Kanten nur zwischen benachbarten Schichten verlaufen. Die *Breite* w eines geschichteten Graphen ist als $\max\{|L_i|\}$ definiert.

Ausgehend vom Startknoten s in Schicht L_0 soll ein Zielknoten t in der letzten Schicht k aufgesucht werden. Dabei ist der Wert k dem Online-Algorithmus A jedoch nicht bekannt. Weiterhin werden die Knoten der Schicht L_i, sowie deren Verbindungen zu Knoten der Schicht L_{i-1} erst enthüllt, sobald A einen Knoten der Schicht L_{i-1} erreicht.

Papadimitriou und Yannakakis untersuchen den Fall $w = 2$, hier besteht jede Schicht also aus höchstens zwei Knoten. Sie zeigen, daß jeder deterministische Algorithmus A in diesem Fall den schlechtesten kompetitiven Faktor auf dem folgenden Graphen G erreichen. G enthält in jeder Schicht genau zwei Knoten, und es verlaufen stets zwei Kanten von diesen Knoten zu den nächsten beiden Knoten. Die Kante e_i, die vom Knoten v_i ausgeht und in Schicht L_{i+1} endet, hat Gewicht 1, falls A die Schicht L_i zum ersten Mal am Knoten v_i betreten hat. Sonst hat die Kante das Gewicht 0. Betritt A eine neue Schicht, hat somit stets die andere Kante das günstigere Gewicht 0.

Daß dieser Graph tatsächlich den ungünstigsten Fall darstellt, führt zu einer interessanten Folgerung. Jeder deterministische Online-Algorithmus kann aus diesem Grund nämlich dargestellt werden durch die Anzahl $f(i)$ der Kanten, die er auf einem Zweig des Schichtgraphen ausführt, bevor er umkehrt und den anderen Zweig wählt. Die Kosten, die für den Algorithmus entstehen, sind dann $2\sum_{i=1}^{n} f(i) + f(n-1)$ und die optimalen Kosten $f(n-1)$. Damit entspricht das Problem genau der 2-Wege Suche aus Abschnitt 4.3.1.

Der Algorithmus von Baeza-Yates *et al.* [BYCR91] für dieses Problem läßt sich also übertragen, und man erhält einen optimalen 9-kompetitiven deterministischen Online-Algorithmus für die Suche in Schichtgraphen der Breite 2. Die Analogie zwischen dem w-Wege Problem und Suchen in Schichtgraphen der Breite w gilt jedoch nur für den Fall $w = 2$. Ist $w > 2$, dann wird der größte kompetitive Faktor nicht stets auf einem Schichtgraphen mit disjunkten Pfaden erreicht.

Für Schichtgraphen mit beliebiger Breite w zeigen Fiat *et al.* [FFK+91] eine untere Schranke von 2^{w-2} für deterministische Online-Algorithmen. Weiterhin geben Sie einen deterministischen $O(9^w)$-kompetitiven Online-Algorithmus an.

Sie zeigen zusätzlich eine untere Schranke von $w/2$ für die Kompetitivität randomisierter Online-Algorithmen. Für den Spezialfall, daß der Graph aus w disjunkten Pfaden besteht, die nur den Startknoten s gemeinsam haben, geben sie einen optimalen $O(\log w)$-kompetitiven randomisierten Online-Algorithmus an.

Ramesh [Ram93] verbessert diese Resultate noch weiter. Er beschreibt einen deterministischen Online-Algorithmus für geschichtete Graphen mit Breite w, der eine Kompetitivität von $O(w^3 2^w)$ erreicht, und einen randomisierten Online-Algorithmus für das gleiche Problem, der die polynomielle Kompetitivität $O(w^{15})$ erzielt. Weiterhin verbessert Ramesh die untere Schranke für randomisierte Algorithmen auf $\Omega(w^2/(\log^{1+\epsilon} w))$ für beliebiges $\epsilon > 0$.

Eine interessante Beobachtung bei diesem Problem ist, daß Task Systeme[1] durch geschichtete Graphen simuliert werden können. Schichten repräsentieren dabei die Tasks, und jeder Knoten einer Schicht ist einem möglichen Zustand zugeordnet. Dennoch können die Algorithmen nicht direkt übertragen werden, da die Breite des entstehenden geschichteten Graphen von der Anzahl der Zustände abhängt. Im Spezialfall des k-Server Problems ist die Kompetitivität somit nicht nur von der Anzahl der Server, sondern auch von der Größe des metrischen Raums abhängig, auf dem sich die Server bewegen.

4.3.7 Offene Probleme

Für das Kartographierproblem in der Definition von Deng, Kameda und Papadimitriou ergibt sich die Frage, ob es ein kompetitives Verfahren für rechteckige Polygone mit unbegrenzt vielen rechteckigen Hindernissen gibt. Die Autoren vermuten, daß ein solcher Algorithmus existiert.

Weiterhin wäre es interessant zu untersuchen, ob weitere Klassen von Polygonen, die kompetitives Suchen erlauben, angegeben werden können. Auch die Frage, ob es eine $O(m)$-kompetitive Suchstrategie für allgemeine Polygone mit m Kanten gibt, steht noch offen.

[1] siehe Abschnitt 2.2.1.

4.3.8 Literatur — Routenplanung

[BC93] A. Blum and P. Chalasani. An on-line algorithm for improving performance in navigation. In *Proc. 34th IEEE Symp. on Foundations of Computer Science*, pages 2–11, 1993.

[BEBFY92] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. On-line navgation in a room. In *Proc. 3rd SIAM Symp. on Discrete Algorithms*, pages 237–249, 1992.

[BRS91] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 494–504, 1991.

[BYCR91] R. A. Baeza-Yates, J. C. Culberson, and G. J. Rawlins. Searching in the plane. *Information and Computation*, 1991.

[CN88] W. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28:39–44, 1988.

[DI94] A. Datta and C. Icking. Competitive searching in a generalized street. In *Proc. of the 10th Annual ACM Symp. on Computational Geometry*, pages 175–182, 1994.

[DKP91] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment (extended abstract). In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 298–303, 1991.

[DP90] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 355–361, 1990.

[FFK+91] A. Fiat, D. P. Foster, H. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan. Competitive algorithms for layered graph traversal. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 288–297, 1991.

[IKM92] C. Icking, R. Klein, and L. Ma. An optimal competitive strategy for looking around a corner. Manuskript, FernUniversität Hagen, 1992.

[Kle91] R. Klein. Walking an unknown street with bounded detour. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 304–313, 1991.

[Kle94] J. M. Kleinberg. On-line search in a simple polygon. In *Proc. 5th SIAM Symp. on Discrete Algorithms*, pages 8–15, 1994.

[KP92] B. Kalyanasundaram and K. Pruhs. A competitive analysis of nearest neighbor based algorithms for searching unknown scenes. In *Proc. STACS*, volume 577 of *LNCS*, 1992.

[KRT93] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. 4th SIAM Symp. on Discrete Algorithms*, pages 441–447, 1993.

[LS87] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[LS90] V. J. Lumelsky and A. A. Stepanov. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462–472, 1990.

[MI94] A. Mei and Y. Igarashi. Efficient strategies for robot navigation in unknown environment, 1994. manuscript.

[PY91] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[Ram93] H. Ramesh. On traversing layered graphs on-line. In *Proc. 4th SIAM Symp. on Discrete Algorithms*, pages 412–421, 1993.

[RKSI93] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical report ORNL/TM-12410, Oak Ridge National Laboratory, 1993.

[ST85] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

4.4 Weitere Anwendungsgebiete

In den vorangegangenen Abschnitten wurden nur einige der wichtigsten untersuchten Probleme auf dem Gebiet der Online-Algorithmen dargestellt. Es folgt an dieser Stelle eine kurze Liste mit weiteren Arbeiten, wobei natürlich kein Anspruch auf Vollständigkeit erhoben werden kann.

4.4.1 Verteilte Systeme

Einige neuere Arbeiten übertragen die Ergebnisse, die für verschiedene Probleme auf Einzelsystemen erzielt wurden, auf verteilte Systeme.

Alon, Kalai, Ricklin und Stockmeyer untersuchen zwei Probleme auf einem Kommunikationsnetzwerk [AKRS92]. Zum ersten das Verfolgen eines Benutzers, der seine Position im Netzwerk verändern kann. Zum zweiten eine Variante des Prozeßplanungsproblems aus Abschnitt 3.4.1, in der die Prozesse auf allen Knoten des Kommunikationsnetzwerks ausgeführt werden können. Dabei entstehen jedoch Verzögerungen durch das Senden der Prozeßdaten. Als Kostenfunktion werden diese Verzögerungen verwendet.

Ein ähnliches Modell für Prozeßplanung in verteilten Systemen untersuchen auch Awerbuch, Kutten und Peleg [AKP92].

Dateiverwaltung innerhalb eines verteilten Systems, wobei eine Datei in mehreren Knoten repräsentiert sein darf, jedoch nur in einem Knoten repräsentiert sein muß, bildet das Thema der Arbeit von Bartal, Fiat und Rabani [BFR92]. Dies ist eine Verallgemeinerung des Paging Problems. Die Autoren geben einen kompetitiven deterministischen Online-Algorithmus an. Ein verbessertes Resultat erzielen Awerbuch, Bartal und Fiat [ABF93b] mit einem randomisierten Online-Algorithmus. Eine weitere Arbeit dieser Autoren [ABF93a] befaßt sich mit dem gleichen Thema.

Das verteilte k-Server Problem wurde bereits in Abschnitt 3.1.4 beschrieben.

4.4.2 Weitere Arbeiten

Optimierungsprobleme, bei denen mehrere unabhängige *Agenten* ihre Aktionen koordinieren müssen, um ein bestimmtes Ziel zu erreichen, wurden von Irani und Rabani [IR93] untersucht. Sie wenden ihre Ergebnisse auf das Load Balancing Problem und Virtual Circuit Routing an, siehe dazu Abschnitt 3.4.

Azar, Broder und Manasse geben ein deterministisches Online-Verfahren MIN an, das mehrere Online-Algorithmen $A_1, \ldots, A_k$ für ein gemeinsames Problem P, das als Task-System formuliert wird, kombiniert [ABM93]. Man nimmt dabei an, daß jeder Algorithmus A_i jeweils für eine Teilmenge der möglichen Anfragefolgen c_i-kompetitiv ist, so daß die Vereinigung der Teilmengen alle Anfragefolgen enthält. Die Autoren zeigen, daß der kombinierte deterministische Online-Algorithmus MIN eine optimale Kompetitivität von $O(\sum c_i)$ erreicht.

Lipton und Tomkins [LT94] untersuchen das *Online Interval Scheduling Problem*, eine Variante der Prozeßplanung aus Abschnitt 3.4.1. Die Anfragen bestehen aus Intervallen auf der positiven reellen Achse, der Online-Algorithmus muß entscheiden, ob ein Intervall angenommen werden soll oder nicht. Angenommene Intervalle dürfen sich nicht überschneiden, und die Gesamtlänge der angenommenen Intervalle soll maximiert werden. Lipton und Tomkins geben einen 2-kompetitiven randomisierten Online-Algorithmus für den Fall an, daß die Intervalle eine Länge von 1 oder $l \gg 1$ haben. Für den allgemeinen Fall konstruieren sie einen $O((\log \Delta)^{1+\epsilon})$-kompetitiven Algorithmus und zeigen, daß eine Kompetitivität von $O(\log \Delta)$ nicht möglich ist. Dabei steht Δ für das Verhältnis der größten zur kleinsten Länge eines Intervalls.

Das *Online Routing Problem*, also das Bereitstellen von Kommunikationskanälen in einem Netzwerk, untersuchen Awerbuch, Azar und Plotkin [AAP93]. Auch die Implementierung paralleler Programme kann als Online-Problem formuliert und kompetitiv analysiert werden [DK94].

Eine Online-Version des Steiner Problems, in dem die regulären Punkte nach und nach enthüllt werden, untersuchten zunächst Imaze und Waxman [IW91]. Westbrook und Yan verbesserten die Resultate und untersuchten zusätzlich eine Verallgemeinerung des ursprünglichen Steiner Problems [WY93].

4.4.3 Literatur

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proc. 34th IEEE Symp. on Foundations of Computer Science*, pages 32–40, 1993.

[ABF93a] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 164–173, 1993.

[ABF93b] B. Awerbuch, Y. Bartal, and A. Fiat. Heat & dump: Competitive distributed paging. In *Proc. 34th IEEE Symp. on Foundations of Computer Science*, pages 22–31, 1993.

[ABM93] Y. Azar, A. Z. Broder, and M. S. Manasse. On-line choice of on-line algorithms. In *Proc. 4th SIAM Symp. on Discrete Algorithms*, pages 432–440, 1993.

[AKP92] B. Awerbuch, S. Kutten, and D. Peleg. Competitive distributed job scheduling. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 571–580, 1992.

[AKRS92] N. Alon, G. Kalai, M. Ricklin, and L. Stockmeyer. Lower bounds on the competitive ratio for mobile user tracking and distributed job scheduling (extended abstract). In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 334–342, 1992.

[BFR92] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management (extended abstract). In *Proc. 24th ACM Symp. on Theory of Computing*, page 39, 1992.

[DK94] X. Deng and E. Koutsoupias. Competitive implementation of parallel programs. In *Proc. 5th SIAM Symp. on Discrete Algorithms*, pages 455–465, 1994.

[EYFKT92] R. El-Yaniv, A. Fiat, R. Karp, and G. Turpin. Competitive analysis of financial games. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 327–333, 1992.

[IR93] S. Irani and Y. Rabani. On the value of information in coordination games. In *Proc. 34th IEEE Symp. on Foundations of Computer Science*, pages 12–21, 1993.

[IW91] M. Imaze and B. M. Waxman. Dynamic steiner tree problem. *SIAM J. Disc. Math.*, 4(3):369–384, 1991.

[LT94] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proc. 5th SIAM Symp. on Discrete Algorithms*, pages 302–311, 1994.

[WY93] J. Westbrook and D. C. K. Yan. Greedy algorithms for the on-line steiner tree and generalized steiner problems. In *Proc. Workshop on Algorithms and Data Structures*, pages 622–633, 1993.

Kapitel 5

Voraussicht

Seltsam ist Prophetenlied;
Doppelt seltsam, was geschieht.
— J. W. Goethe *Weissagungen des Bakis*

Anders als in den vorangegangenen Abschnitten betrachten wir nun den Fall, daß der Online-Algorithmus nicht nur die aktuelle Anfrage erhält, sondern zusätzlich eine begrenzte Anzahl der weiteren Anfragen. Soweit nicht anders vermerkt, entstammen alle genannten Ergebnisse der Arbeit von Susanne Albers [Alb92].

Dabei unterscheidet man zwei Arten von Voraussicht. Der Algorithmus hat *schwache* Voraussicht l, wenn er von der aktuellen Anfrage $\sigma(i)$ und zusätzlich den Anfragen $\sigma(i+1), \ldots, \sigma(i+l)$ Kenntnis hat. Er hat *starke* Voraussicht l, wenn er neben der aktuellen Anfrage $\sigma(i)$ die nächsten l *verschiedenen* Anfragen kennt, d.h. die Anfragen $\sigma(i+1), \ldots, \sigma(i+\lambda(i))$ mit

$$\lambda(i) = \min\{s > i \mid |\{\sigma(i), \ldots, \sigma(s)\}| = l + 1\}.$$

Die Anzahl der tatsächlich vorausgesehenen Anfragen ist bei starker Voraussicht l somit immer größer oder gleich der Anzahl l bei schwacher Voraussicht l.

Schwache Voraussicht scheint zunächst das natürlichere Modell zu sein, in einigen Fällen kann jedoch der Gegenspieler diese Voraussicht unbenutzbar machen, indem er jeweils l gleiche Anfragen erzeugt. Ein Beispiel hierfür ist das k-Server Problem, und somit auch im Spezialfall das Paging Problem, wie das folgende Lemma zeigt.

Lemma 5.0.1 *Schwache Voraussicht kann die Kompetitivität eines Online-Algorithmus für das k-Server Problem nicht erhöhen.*

Beweis: Sei A ein Online-Algorithmus für das k-Server Problem, c der kompetitive Faktor von A und $c_0 < c$. Dann gibt es eine unendliche Folge σ von Anfragen, so daß $c_A(\sigma_n) > c_0 \cdot c_{opt}(\sigma_n)$, wobei σ_n die Folge der ersten n Anfragen von σ ist.

Ist τ nun die Folge, die man erhält, wenn man jede Anfrage der Folge σ l-mal ausführt, so gilt $c_A(\tau_{l \cdot n}) \geq c_A(\sigma_n)$ und $c_{opt}(\tau_{l \cdot n}) = c_{opt}(\sigma_n)$. Daher folgt $c_A(\tau_n) \geq c_0 \cdot c_{opt}(\tau_n)$. Für die Folge τ bedeutet eine Voraussicht von l jedoch keinen Vorteil, d.h. auch mit Voraussicht gilt für den kompetitiven Faktor c die Beziehung $c > c_0$. $\square$

Will man in solchen Fällen dennoch die Auswirkungen von schwacher Voraussicht untersuchen, muß man die Kompetitivität von der Länge n der Anfragefolge σ abhängig machen.

5.1 Beispiele

Im folgenden wird gezeigt, daß Voraussicht zwar nicht immer zur Verbesserung des bestmöglichen kompetitiven Faktors führt, daß dies jedoch bei einigen Problemstellungen durchaus der Fall ist.

5.1.1 Paging mit Voraussicht

Wir betrachten zunächst das Paging Problem aus Abschnitt 3.3. Für den Fall von starker Voraussicht l mit $l \leq k-2$ führt eine Variante von **LRU** zu guten Ergebnissen. Der deterministische Online-Algorithmus $\mathbf{LRU}(l)$ entfernt bei vollem Cache immer die Speicherseite, die den am weitesten zurückliegenden letzten Zugriff hat und nicht unter den Seiten in der Voraussicht vorkommt.

Satz 5.1.1 *Der Algorithmus* $\mathbf{LRU}(l)$ *mit starker Voraussicht* l *mit* $l \leq k-2$ *ist* $(k-l)$*-kompetitiv für das allgemeine Paging Problem.*

Dieses Ergebnis ist optimal, denn

Satz 5.1.2 *Ist ein deterministischer Online-Algorithmus* A *mit starker Voraussicht* $l \leq k-2$ c*-kompetitiv, dann gilt* $c \geq k-l$.

Verändert man den Algorithmus $\mathbf{LRU}(l)$ so, daß er nicht immer die volle Voraussicht benutzt, sondern immer *Blöcke* der Länge $\lambda(i)$ betrachtet, so erhält man den Algorithmus BLOCK-$\mathbf{LRU}(l)$. Dieser Algorithmus betrachtet also die nächsten l verschiedenen Anfragen und bearbeitet dann diese, ohne weitere Voraussicht zu benutzen. Erst nach der Bearbeitung dieses *Blocks* nutzt er wieder die Voraussicht und betrachtet die nächsten l verschiedenen Anfragen. Interessanterweise ist dieser Algorithmus nur wenig schlechter als der ursprüngliche und somit nahezu optimal.

Satz 5.1.3 *Der Algorithmus* BLOCK-$\mathbf{LRU}(l)$ *mit starker Voraussicht* l *mit* $l \leq k-2$ *ist* $(k-l+1)$*-kompetitiv für das allgemeine Paging Problem.*

Auch zwei Resultate für randomisierte Algorithmen aus Abschnitt 3.3.3 lassen sich erweitern.

Man modifiziert zunächst den Algorithmus MARK aus Abschnitt 3.3.3 so, daß vor Beginn jeder Phase folgendes ausgeführt wird. Sei U die Menge der Seiten in der Voraussicht, die nicht im Cache liegen. Dann wählt der neue Algorithmus MARK(l) $|U|$ viele Seitenrahmen im Cache zufällig unter denen aus, die Seiten enthalten, die nicht in der Voraussicht enthalten sind und liest dort die Seiten aus U ein. Nach diesem Schritt läuft die Phase wie beim Algorithmus MARK ab.

Satz 5.1.4 *Sei $l \leq k-2$. Der Algorithmus* $\textsc{Mark}(l)$ *mit starker Voraussicht l ist $2H_{k-l}$-kompetitiv.*

Die untere Schranke von H_k für Kompetitivität gegen unwissende Gegenspieler kann ebenso übertragen werden.

Satz 5.1.5 *Kein Online-Algorithmus mit starker Voraussicht l, wobei $l \leq k-2$, kann c-kompetitiv gegen unwissende Gegenspieler sein für $c < H_{k-l}$.*

5.1.2 List Update mit Voraussicht

Wir betrachten nun das Problem der selbstanordnenden Listen aus Abschnitt 3.2. Dabei bezeichne m die Länge der Liste. Starke Voraussicht der Länge l verringert den kompetitiven Faktor eines Online-Algorithmus für das List Update Problem nur, falls $l = m$ — hier sieht der Algorithmus alle Anfragen voraus — oder l linear in m ist. Dies zeigt der folgende Satz.

Satz 5.1.6 *Sei A ein deterministischer Online-Algorithmus mit starker Voraussicht $l \leq n-1$. Falls A c-kompetitiv ist, dann gilt*

$$c \geq 2 - \frac{l+2}{m+1}.$$

Für schwache Voraussicht muß die Voraussicht sogar noch größer sein.

Satz 5.1.7 *Sei A ein deterministischer Online-Algorithmus mit schwacher Voraussicht l. Ist A c-kompetitiv, dann gilt*

- $c \geq 2$, *falls* $l = o(n^2)$ *und*
- $c \geq 2 - 2\sqrt{4K^2 + 2K} + 4K$, *falls* $(l+1) = Kn^2$.

Albers gibt auch einen Online-Algorithmus an, der starke Voraussicht l ausnutzt und für $l \geq \lfloor m/2 \rfloor$ einen kompetitiven Faktor $c < 2$ erzielt. Für $l = m-1$ beträgt c ungefähr 15/8. Der Algorithmus basiert auf der **MTF**-Strategie und unterteilt die Anfragefolge in Blöcke.

5.1.3 Graph Coloring mit Voraussicht

Irani [Ira90] betrachtet das Graph Coloring Problem auf d-induktiven Graphen, siehe auch Abschnitt 4.1.4.

Satz 5.1.8 *Ein d-induktiver Graph G mit n Knoten kann bei schwacher Voraussicht l online mit $O(\min\{d \log n, \frac{d \cdot n}{l}\})$ Farben gefärbt werden.*

In diesem Fall verringert somit schwache Voraussicht der Größe l die Kompetitivität, falls $l \geq n/\log n$. Irani zeigt auch, daß es keinen Online-Algorithmus mit kleinerer Voraussicht geben kann, der eine bessere Kompetitivität erzielt.

Für $l \geq n/\log n$ ist $\Omega(\frac{n}{l})$ eine untere Schranke für die Kompetitivität jedes Online-Algorithmus.

5.1.4 Matching mit Voraussicht

Kao und Tate [KT91] untersuchen das Online Matching Problem aus Abschnitt 4.2 im Fall, daß die Spalten der Strukturmatrix in Blöcken der Größe k enthüllt werden. Dies ist eine etwas schwächere Voraussetzung als schwache Voraussicht der Länge k.

Sie zeigen, daß diese Voraussicht die Kompetitivität nur erhöhen kann, wenn k in $\Theta(n)$ liegt. Konstante Voraussicht führt also wie beim List Update Problem zunächst nicht zu verbesserten Resultaten. Für den Fall $k = \Theta(n)$ zeigen Sie eine untere Schranke für die Kompetitivität von randomisierten Algorithmen gegen unwissende Gegenspieler. Sie beweisen, daß die Kompetitivität gegenüber einem Matching Algorithmus ohne Blöcke höchstens um den Faktor k kleiner wird.

Die Autoren schlagen folgenden Algorithmus für Matching mit Blöcken vor, der im wesentlichen den Online-Algorithmus RANKING von Karp *et al.* verallgemeinert.

Zunächst wird zufällig eine Permutation der Jungen ausgewählt. Diese Rangfolge wird nun während der Bearbeitung der Anfragen beibehalten. Für jeden Block wird mit einem entsprechenden Offline-Algorithmus ein maximales Matching ausgewählt, das zusätzlich möglichst viele Jungen von hohem Rang benutzt. Dazu wird jedem Jungen ein von der Rangfolge abhängiges Gewicht zugeordnet und ein gewichtetes Offline-Matching durchgeführt.

Kao und Tate vermuten, daß der angegebene Algorithmus optimal ist, konnten jedoch die Kompetitivität des Algorithmus nicht in Abhängigkeit von der Blockgröße bestimmen. Klar ist, daß der Algorithmus für $k = o(n)$ mindestens so gut ist wie der Algorithmus RANKING und daß er für $k = n$ das optimale Matching findet. Der Algorithmus erreicht nach Experimenten die gezeigte untere Schranke für den Fall $k = n/2$, nicht aber für $k = n/3$ oder kleiner. Die Autoren vermuten, daß die untere Schranke dort noch erhöht werden kann.

5.2 Keine Verbesserung

Nicht immer verbessert starke Voraussicht die Kompetitivität von Online-Algorithmen. Dies gilt insbesondere für die beiden wichtigen Fälle k-Server Problem und metrische Task Systeme. Natürlich kann auch schwache Voraussicht dann die Kompetitivität nicht verbessern, dies haben wir für das k-Server Problem in Lemma 5.0.1 bereits gezeigt.

Satz 5.2.1 *Sei $s > 0$ und $0 < \epsilon < 1$. Dann existiert ein Graph mit $k + s + 1$ Knoten, für den kein deterministischer k-Server Algorithmus mit starker Voraussicht $l \leq k + s - 1$ einen kompetitiven Faktor von $\leq \min\{k, (k - l + s)\} - \epsilon$ erreichen kann.*

Starke Voraussicht ist also nur von Vorteil, wenn der gegebene Graph weniger als $(k + l + 1)$ Knoten hat und somit sehr klein ist. Für metrische Task Systeme gilt ein ähnlicher Satz.

Satz 5.2.2 *Sei A ein deterministischer Online-Algorithmus für metrische Task Systeme, er habe starke Voraussicht l. Dann kann die Kompetitivität von A nicht kleiner sein als $(2n-1)$.*

Aufgrund des Algorithmus aus Abschnitt 2.2.1 kann der kompetitive Faktor hier also nicht verbessert werden.

5.3 Der Zugriffsgraph beim Paging Problem

Eine andere Möglichkeit der Voraussicht wird von Borodin *et al.* [BIRS91] vorgeschlagen. Hier wird die Menge der möglichen nächsten Anfragen eingeschränkt durch den sogenannten *Zugriffsgraphen*. Jeder Knoten des Graphen enthält eine Anfrage, wobei verschiedene Knoten die gleiche Anfrage enthalten können. Kanten geben mögliche folgende Anfragen an. Zulässig sind nun nur solche Anfragefolgen, die einen Weg durch den Zugriffsgraphen beschreiben. Sind keine Einschränkungen bekannt, ist der Zugriffsgraph somit der vollständige Graph.

Die Autoren zeigen anhand verschiedener Beispiele, daß die Annahme, daß folgende Anfragen auf diese Weise eingeschränkt sind, in der Praxis in vielen Fällen durchaus berechtigt ist. Sie konstruieren einen *universellen*, d.h. unabhängig von G definierten, Algorithmus **Far**, dessen kompetitiver Faktor für jede Cachegröße k, also für jede Anzahl von Seiten, die das Cache enthalten kann, und für jeden Zugriffsgraphen G in $O(c_k(G))$ liegt, wobei $c_k(G)$ der optimale kompetitive Faktor für den Graphen G darstellt.

5.4 Literatur — Voraussicht

[Alb92] S. Albers. The influence of lookahead in competitive on-line algorithms. Technical Report MPI-I-92-143, Max-Planck-Institut für Informatik, 1992.

[BIRS91] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 249–259, 1991.

[Ira90] S. Irani. Coloring inductive graphs on-line. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 470–479, 1990.

[KT91] M.-Y. Kao and S. R. Tate. Online matching with blocked input. *Information Processing Letters*, 38(3):113–116, 1991.

Kapitel 6

Zusammenfassung

Seit der Einführung durch Sleator und Tarjan im Jahre 1985 erhielt die kompetitive Analyse schnell Einzug in die unterschiedlichsten Anwendungsgebiete. Noch immer sind eine Fülle von Problemen offen, darunter so wichtige wie das k-Server Problem für beliebige metrische Räume. Darüber hinaus gibt es zahlreiche weitere Online-Probleme, die noch nicht unter dem Aspekt der Kompetitivität von Algorithmen betrachtet wurden. Eine wichtige Richtung bilden hier in jüngster Zeit Untersuchungen von Problemstellungen in verteilten Systemen.

Wir haben in dieser Arbeit die Grundlagen der kompetitiven Analyse recht ausführlich dargelegt, einschließlich der Unterschiede, die sich ergeben, wenn man von deterministischen zu randomisierten Online-Algorithmen übergeht. In welchen Gebieten das Qualitätsmaß der Kompetitivität angewandt werden kann, wurde dann anhand zahlreicher Beispiele deutlich gemacht. Hierbei war es uns vor allem wichtig, eine Übersicht über die wichtigsten erzielten Ergebnisse zu geben. Wo es möglich war, haben wir jedoch auch die Beweise einfacherer Sachverhalte mit in den Text aufgenommen, um die Anwendung der theoretischen Begriffe aus dem zweiten Abschnitt transparent zu machen.

Als ein besonders interessantes Anwendungsgebiet der kompetitiven Analyse erschien uns die Routenplanung, da hier die dem Online-Algorithmus fehlende Information nicht rein zeitlich, sondern räumlich von ihm entfernt ist. Damit hat dieser die Möglichkeit, durch gezielte Bewegung in der unbekannten Umgebung selbst zu beeinflussen, welche zusätzliche Information er erhält. Auch auf diesem Gebiet gibt es zahlreiche Möglichkeiten, vorhandene Ergebnisse zu erweitern oder neue Fragestellungen zu untersuchen.