Diplomarbeit

Pfadplanung unter Unsicherheit

Uwe Zeisberger

26. April 2005

Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Institut für Informatik

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Uwe Zeisberger

Inhaltsverzeichnis

1. Einführung

Das Canadian Traveller's Problem (CTP) wurde von Christos H. Papadimitriou und Mihalis Yannakakis[6] definiert und ist eine Verallgemeinerung des Single Target Shortest Path Problems. Bei beiden Problemenstellungen ist ein gewichteter Graph gegeben und auf möglichst kurzem Wege soll ein gegebener Zielpunkt t erreicht werden. Beim CTP wird die Aufgabe dadurch erschwert, dass die Kosten der Kanten zu Beginn nicht bekannt sind, sondern es nur eine Wahrscheinlichkeitsverteilung über mögliche Kosten gibt. Die tatsächlichen Kosten einer Kante werden erst bekannt, wenn einer der beiden Kantenendpunkte erreicht wird.

Die namensgebende Anwendung des CTP ist folgende: Ein Autofahrer – beispielsweise in Kanada – möchte zu einem bestimmten Ort fahren. Obwohl ihm eine Straßenkarte zur Verfügung steht, weiss er dennoch nicht, ob alle Teile des geplanten Weges passierbar sind, da möglicherweise ganze Streckenabschnitte etwa durch Schnee blockiert sind. So muss er unter Umständen Umwege in Kauf nehmen.

Die Motivation dieses Problem zu untersuchen entstand aus den Parallelen zu RoboCupRescue[1]. Hier haben die beteiligten Agenten den Auftrag, in einem Katastrophengebiet Hilfe zu leisten, indem sie zunächst einen Weg zu ihrem Einsatzort finden und dort daraufhin z. B. Feuer löschen oder Verletzte bergen. Für RoboCupRescue ist charakteristisch, dass sich die Umgebung kontinuierlich verändert und dadurch auch die Passierbarkeit der Straßen.

Somit stellt das CTP, bei dem diese dynamische Komponente fehlt, eine wesentliche Vereinfachung des Problems dar.

Verwendete Funktionen und Schreibweisen

- $\overline{\mathbb{R}} = \mathbb{R}^+ \cup \{\infty\}$ mit $\mathbb{R}^+ = \{r \in \mathbb{R} : r > 0\}$. Die Menge $\overline{\mathbb{R}}$ besteht gerade aus den zugelassen Kosten einer Kante. Ist der Wert ∞ so ist die Kante nicht passierbar.

- Der Träger $\operatorname{supp} f$ einer reellwertigen Funktion $f : D \to \mathbb{R}$ ist definiert als die Menge von Elementen des Definitionsbereichs von f, für die der Wert der Funktion nicht 0 ist:

  $$\operatorname{supp} f = \{d \in D : f(d) \neq 0\}$$

  Weiterhin sei $\operatorname{fsupp} f$ definiert als $\operatorname{supp} f \setminus \{\infty\}$.

- Mit $\mathrm{dist}(\Omega)$ wird die Menge der Wahrscheinlichkeitsverteilungen über Ω mit endlichem Träger bezeichnet. Eine Wahrscheinlichkeitsverteilung über Ω ist eine Funktion $p : 2^\Omega \to [0,1]$ mit:

$$\begin{aligned} p(\Omega) &= 1 \\ p(A \cup B) &= p(A) + p(B) \quad \text{falls } A \cap B = \emptyset \end{aligned}$$

- Die Menge der Zufallsvariablen mit Wertebereich M und einem endlichen Träger wird mit $\mathrm{ZV}(M)$ bezeichnet. Für den Grundraum der Ereignisse Ω ist eine Zufallsvariable eine Funktion $X : \Omega \to M$. Die Verteilung $P_X \in \mathrm{dist}(M)$ einer Zufallsvariable $X \in \mathrm{ZV}(M)$ hängt wie folgt von der auf Ω zu Grunde gelegten Verteilung P ab:

$$P_X(A) = P(X^{-1}(A)) = P(\{\omega \in \Omega : X(\omega) \in A\})$$

- Um eine Zufallsvariable aus $\mathrm{ZV}(M)$ zu definieren, wird sie – statt als Funktion von Ω nach M – durch eine Funktion gegeben, die jedem Wert aus M eine Wahrscheinlichkeit zuordnet. Das vereinfacht die Schreibweise und ist auch für mehrere Zufallsvariablen sinnvoll, wenn alle unabhängig sind.

- $\mathrm{ZV}(\overline{\mathbb{R}})$ wird mit $\mathbb{K}$ bezeichnet. Eine Zufallsvariable aus dieser Menge steht für das Wissen eines Agenten über die Kosten einer Kante. Wenn sie konstant ist, kennt er die genauen Kosten.

- Für eine konstante Zufallsvariable c bezeichne c^* den eindeutigen Funktionswert. Für $r \in \overline{\mathbb{R}}$ bezeichne r° die konstante Zufallsvariable aus $\mathbb{K}$ mit dem Wert r. In Formeln und mit den beiden vorherigen Vereinbarungen gilt also:

$$\begin{aligned} r^\circ &= \{r \mapsto 1\} \\ \{r \mapsto 1\}^* &= r \end{aligned}$$

  Wird für eine Zufallsvariable c über c^* etwas ausgesagt, ist implizit die Forderung enthalten, dass c konstant ist.

- Für einen gegebenen Graphen (V, E) sei $\mathrm{edge}(v)$ für $v \in V$ die Menge der von v ausgehenden Kanten:

$$\mathrm{edge}(v) = \{e \in E : v \in e\}$$

- Für die Tupel $a = (a_i)_{i \in I}$ und $b = (b_i)_{i \in I'}$ bezeichne $a[b] = (a[b]_i)_{i \in I \cup I'}$ das Tupel, das an den Stellen $i \in I'$ den Wert b_i hat und auf $I \setminus I'$ mit a übereinstimmt.

- Seien A und B zwei Mengen. Dann bezeichne A^B die Menge aller Tupel $(a_b)_{b \in B}$ mit $a_b \in A$ für $b \in B$.

- Für eine Funktion $f : M' \to \mathbb{R}$ und eine Menge $M \subseteq M'$ sei weiterhin gegeben:

$$\operatorname*{argmin}_M f = \left\{ m \in M : f(m) = \min_M f \right\}$$

  Diese Funktion ist nur dann wohldefiniert, wenn $\min_M f$ existiert. Das ist im Folgenden aber immer der Fall und wird deshalb nicht mehr explizit erwähnt.

- Für eine Menge A, $a, b \in A$ und eine Funktion $f : A \to \mathbb{R}$ sei die Funktion $f[a \to b]$ für $x \in A$ wie folgt definiert:

$$f[a \to b](x) = \begin{cases} 0 & \text{falls } x = a \\ f(a) + f(b) & \text{falls } x = b \\ f(x) & \text{sonst} \end{cases}$$

Bemerkung Eine Funktion $f : \Omega \to [0, 1]$ mit $\sum_{\omega \in \Omega} f(\omega) = 1$ impliziert eine Wahrscheinlichkeitsverteilung p_f über Ω wie folgt:

$$p_f(A) = \sum_{\omega \in A} f(\omega) \quad \text{für } A \subseteq \Omega$$

Wenn sich die Funktionswerte von f wie oben zu 1 summieren, gilt das auch für $f[a \to b]$, es ist also auch $p_{f[a \to b]}$ eine Wahrscheinlichkeitsverteilung.

2. Definition des Canadian Traveller's Problem

Definition 2.1 Eine *Instanz* des CTP ist ein Tupel $\langle V, E, c, s, t\rangle$. Es besteht aus einem endlichen, ungerichteten, zusammenhängenden Graphen $\langle V, E\rangle$ mit einer Kantengewichtung c und einem Startpunkt $s \in V$ sowie einem zu erreichenden Zielpunkt $t \in V$. $c = (c_e)_{e\in E} \in \mathbb{K}^E$ ist dabei ein Tupel von unabhängigen Zufallsvariablen mit dem Wertebereich $\overline{\mathbb{R}}$. Die Menge der Instanzen wird mit CTP bezeichnet.

Für eine Instanz des CTP hat ein Agent die Aufgabe, einen Weg von s nach t zu finden. Zu Beginn befindet er sich in s und kennt die Gewichte aller von s ausgehenden Kanten. Von den anderen Kanten kennt er nur die (endlich vielen) möglichen Werte und deren Wahrscheinlichkeiten. Nun wählt er eine von seinem aktuellen Standpunkt ausgehende Kante. Danach befindet er sich am anderen Ende der gewählten Kante und erfährt alle Gewichte der von dort ausgehenden Kanten. Das wird wiederholt, bis der Agent t erreicht. Der gewählte Weg ist dabei um so besser, je kleiner die Summe der Kosten der überquerten Kanten ist.

Definition 2.2 Jedes Element aus dem Wahrscheinlichkeitsraum, welcher der Kantengewichtung einer Instanz zugrunde liegt, heißt eine *Realisierung* der Instanz. Die zugehörige Wahrscheinlichkeitsverteilung wird mit P^I, die Menge der Realisierungen einer Instanz I mit $\mathcal{R}_I$ bezeichnet. Eine Realisierung, in der ein Pfad mit endlichen Kosten zwischen s und t existiert, heißt endlich. Die Menge der endlichen Realisierungen von I wird mit $\mathcal{R}_I^{<\infty}$ bezeichnet.

Bemerkung Für die Kantenkosten ist der Grundraum nicht explizit gegeben, deshalb ist Definition 2.2 rein formal nicht eindeutig. Im weiteren wird angenommen, dass $\mathcal{R}_I$ den einfachsten Wahrscheinlichkeitsraum bezeichne. Das ist der, für den gilt:

$$\forall r_1, r_2 \in \mathcal{R}_I : c(r_1) = c(r_2) \implies r_1 = r_2$$

Er hat die Mächtigkeit $\prod_{e\in E} |\operatorname{supp} c_e| < \infty$.

Definition 2.3 Sind für eine Instanz I alle Realisierungen endlich, d. h.

$$\mathcal{R}_I = \mathcal{R}_I^{<\infty} \quad ,$$

so heißt die Instanz selbst *endlich*.

Während sich der Agent in einer Instanz bewegt, erfährt er von einigen Kanten e den Wert der Zufallsvariable c_e für die tatsächliche Realisierung. In der Wahrscheinlichkeitstheorie ist es üblich, Sigmaalgebren und Messbarkeit bzgl. dieser Sigmaalgebren zu verwenden, und damit eine Struktur zu Grunde zu legen, die Informationen darüber enthält, für welche Zufallsvariablen die Werte schon bekannt sind.

Da hier aber die einzelnen Kantengewichtungen unabhängig sind, kann man das Wissen über die Werte auch einfacher angeben. Für eine Kante bekannten Gewichts wird dazu einfach die ursprüngliche Zufallsvariable c_e durch eine neue ersetzt, die für alle Realisierungen den Wert dieser Kosten annimmt. Somit genügt für die Kodierung der Zustände, in die der Agent kommen kann, die Angabe des Standpunktes und das Wissen in Form von Zufallsvariablen:

Definition 2.4 Ein *Zustand einer Instanz* $\langle V, E, c, s, t\rangle$ ist ein Tupel $\langle v, \gamma\rangle \in V \times \mathbb{K}^E$, für das eine zusammenhängende Menge $\hat{V} \subseteq V$ mit $s, v \in \hat{V}$ existiert, so dass gilt:

$$\forall e \in E : \gamma_e = \begin{cases} c_e & \text{falls } e \text{ mit keinem Knoten aus } \hat{V} \text{ inzidiert} \\ r^\circ & \text{für ein } r \in \operatorname{supp} c_e, \text{ sonst} \end{cases}$$

Die erste Komponente eines Zustandes gibt den Knoten an, in dem der Agent sich befindet. Das Wissen des Agenten wird durch die zweite Komponente γ festgelegt. Hat der Agent in der Realisierung r für eine Kante e schon einen der beiden Endpunkte von e erreicht, so ist $\gamma_e = c_e(r)^\circ$, sonst weiß der Agent noch nicht mehr als zu Beginn über die Kante und deshalb ist $\gamma_e = c_e$.

Als Spezialfall der Bedingung ergibt sich, dass der Agent für die von seinem Standpunkt ausgehenden Kanten die tatsächlichen Kosten kennt.

Diese Definition legt genau alle vom Startzustand aus erreichbaren Zustände fest. Der Zustand $\langle v, \gamma\rangle$ lässt sich erreichen, indem von s aus ein beliebiger Pfad in $\hat{V}$ gewählt wird, der in v endet und alle Knoten aus $\hat{V}$ besucht. Da $\hat{V}$ zusammenhängend ist, existiert ein solcher Pfad. Andererseits ist auch klar, dass diese Bedingung notwendig ist, um $\langle v, \gamma\rangle$ erreichen zu können, weil man genau für die Kanten, die mit einem schon besuchten Knoten inzidieren, die Kosten kennt und für die übrigen Kanten die Kosten unverändert und durch c festgelegt sind.

Einfacher und kompakter als Tupel $\langle V, E, c, s, t\rangle$ kann man eine CTP-Instanz mit Hilfe einer Zeichnung darstellen. Die Wahrscheinlichkeitsverteilung über die Kosten einer Kante wird in der Form $c_1 : p_1; c_2 : p_2; \ldots$ direkt an die Kante geschrieben und bedeutet, dass die Kosten $c_1 \in \mathbb{R}^+$ die Wahrscheinlichkeit $p_1 \in [0, 1]$ haben usw. Falls sich die p_i zu weniger als 1 summieren, hat die Kante in den verbleibenden Fällen die Kosten ∞. Die Beschriftung $c_1 : 1$ wird zu c_1 abgekürzt und eine Kante ohne Kennzeichnung habe immer die Kosten 1.

Abbildung 2.1 zeigt ein Beispiel, das im Wesentlichen aus [6] entnommen ist. Nebenbei sei bemerkt, dass dieses Beispiel nicht sonderlich schwer zu lösen ist, weil für alle Realisierungen des Problems der kürzeste Weg von s nach t immer derselbe – nämlich der über die oberen drei Knoten ist. Ein Agent kann also einfach diesen Weg wählen und hat damit auf jeden Fall einen kürzesten Pfad gefunden.

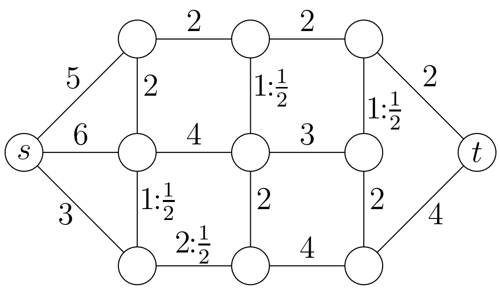

Abbildung 2.1.: Beispielgraph aus [6]

Während der Reise von s nach t erfährt der Agent von immer mehr Kanten deren reale Kosten. Dieses Wissen kann dann selbstverständlich für die weitere Pfadplanung verwendet werden. Ein Agent, der nach k Schritten bisher dem Pfad $v_0 v_1 \dots v_k$ gefolgt ist, befindet sich also im Knoten v_k und kennt die tatsächlichen Kosten aller Kanten, die mit (mindestens) einem v_i inzidieren. Diese beiden Tatsachen – der gegenwärtige Aufenthaltsort v_k und die Kenntnis der Kosten der genannten Kanten – sind die einzigen Informationen die sinnvollerweise die Wahl der als nächstes zu überquerenden Kante beeinflussen. Somit sind die wesentlichen Informationen zur Wahl des weiteren Vorgehens in einem Zustand enthalten.

Definition 2.5 Eine Strategie für eine Instanz $\langle V, E, c, s, t\rangle$ des Canadian Traveller's Problem ist eine Funktion $p : (V \setminus \{t\}) \times \mathbb{K}^E \to V$ für die

$$\{p(v, \gamma), v\} \in E$$

gilt, falls $\langle v, \gamma\rangle$ ein Zustand ist. Die Menge der Strategien für I wird mit $\mathcal{S}_I$ bezeichnet.

Eine Strategie legt fest, wie sich ein Agent verhält. Die Argumente spezifizieren dabei den aktuellen Zustand. $\{v, p(v, \gamma)\}$ ist dann die von $v \in V$ aus gewählte nächste Kante.

Die Kosten, die bei einem Schritt entstehen, entsprechen den Kosten für die überquerte Kante und betragen somit $\gamma^*_{\{v,p(v,\gamma)\}}$. Die Zufallsvariable $\gamma_{\{v,p(v,\gamma)\}}$ ist dabei konstant, weil der Agent bereits ein Ende der betroffenen Kante – nämlich v – erreicht hat. Die Menge der Strategien für eine Instanz I wird mit $\mathcal{S}_I$ bezeichnet.

Um eine Strategie zu definieren, ist es einfacher und verständlicher sie in Worten wiederzugeben. Die Formalisierung ist allerdings hilfreich und notwendig um weitere Begriffe – wie etwa die Kosten einer Strategie – einzuführen. Außerdem genügt es auch, eine Strategie für die Fälle zu definieren, die tatsächlich auftreten können. Für die Instanz aus Abbildung 2.1 genügt somit die Festlegung „Gehe von s aus über die drei oberen Knoten zu t“ um eine Strategie zu konkretisieren. Wie das Verhalten sein soll, wenn sich der Agent z. B. im unteren mittleren Knoten verhält, spielt dann keine Rolle, weil jede beliebige Festsetzung für diese Situationen zum gleichen Ergebnis führt.

Im folgenden Kapitel werden einige Strategien vorgestellt. Um sie vergleichen zu können, wird eine Ordnung auf den Strategien definiert, die erlaubt unter zwei Strategien die „bessere“ zu identifizieren.

Eine solche Ordnung soll Strategien bzgl. der Kosten vergleichen, die einem Agenten beim Befolgen der jeweiligen Strategie entstehen. Dazu werden zunächst die Kosten einer Strategie eingeführt.

Definition 2.6 Sei eine Instanz $I = \langle V, E, c, s, t\rangle$ gegeben. Die *Kosten einer Strategie $p \in \mathcal{S}_I$ auf einer Realisierung $r \in \mathcal{R}_I$ im Zustand $\langle v, \gamma\rangle$* werden mit $\mathrm{cost}_p(r, \langle v, \gamma\rangle)$ bezeichnet und sind rekursiv definiert durch:

$$\begin{aligned}
\mathrm{cost}_p(r, \langle t, \cdot\rangle) &= 0 \\
\mathrm{cost}_p(r, \langle v, \gamma\rangle) &= \gamma^*_{\{v,v'\}} + \mathrm{cost}_p(r, \langle v', \gamma'\rangle) \quad \text{für } v \neq t \text{ mit} \\
v' &= p(v, \gamma) \\
k(\zeta, w) &= \zeta[(c_e(r)^\circ)_{e\in\mathrm{edge}(w))}] \\
\gamma' &= k(\gamma, v')
\end{aligned}$$

$\mathrm{cost}_p(r, \langle s, k(c, s)\rangle)$ wird auch als $\mathrm{cost}_p(r)$ geschrieben und mit *Kosten der Strategie p* bezeichnet.

In Worten ist $k(\zeta, w)$ das Wissen des Agenten, wenn er mit dem Wissen ζ den Knoten w erreicht, also noch Informationen über die von w ausgehenden Kanten erhält. Der erste Zustand des Agenten ist gerade $\langle s, k(c, s)\rangle$, weil er in s startet und nur von den Kanten, die mit s inzidieren, die Kosten in der tatsächlichen Realisierung kennt. $\mathrm{cost}_p(r)$ sind also die Gesamtkosten für das Verfolgen der Strategie p.

Die Kosten dieser Kanten sind gerade die Werte der entsprechenden Zufallsvariablen für die betrachtete Realisierung. γ' ist somit das Wissen des Agenten nach dem Erreichen von v'. Die Kosten im Zustand $\langle v, \gamma\rangle$ setzen sich dann aus den Kosten der Kante, die von p als nächstes traversiert wird, und den Kosten der dadurch entstehenden Situation zusammen.

Etwas einfacher – aber weniger exakt – lassen sich die Kosten auch wie folgt definieren: Jede Strategie p impliziert für eine Realisierung r eine Folge von Knoten $(v_i)_{i=0}^k$, die aus den nacheinander besuchten Knoten besteht. (Wenn die Strategie nicht zum Ziel führt hat die Folge keine endliche Länge und die Kosten werden auf ∞ festgesetzt.) Es gilt dann $v_0 = s$ und $v_k = t$ und die Kosten betragen

$$\mathrm{cost}_p(r) = \sum_{i=1}^{k} c_{\{v_{i-1}, v_i\}} \quad .$$

Diese Darstellung sieht allerdings einfacher aus, als sie tatsächlich ist, da die Folge $(v_i)_{i=0}^k$ im Allgemeinen von r abhängt.

Die folgende Definition legt die minimalen Eigenschaften einer Ordnung über Strategien fest. Sie definiert jedoch noch keine spezielle Ordnung.

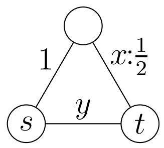

Abbildung 2.2.

Definition 2.7 Eine *Strategieordnung* $\leq_I$ für eine Instanz I des CTP ist eine Ordnung auf den Strategien für I, so dass für alle Strategien $p_1, p_2 \in \mathcal{S}_I$ gilt:

$$\forall r \in \mathcal{R}_I : \mathrm{cost}_{p_1}(r) \leq \mathrm{cost}_{p_2}(r) \implies p_1 \leq_I p_2 \tag{2.1}$$

Wie üblich wird $(p_1 \leq_I p_2 \wedge p_2 \not\leq_I p_1)$ als $p_1 <_I p_2$ und $(p_1 \leq_I p_2 \wedge p_2 \leq_I p_1)$ als $p_1 =_I p_2$ geschrieben. Der Begriff der Ordnung ist dabei so zu verstehen, dass je zwei Strategien vergleichbar sein müssen, die dadurch implizierte Gleichheit ist jedoch nur eine Äquivalenzrelation. D. h. , es ist auch für verschiedene Strategien p_1 und p_2 möglich, dass $p_1 =_I p_2$ gilt. Das entspricht der Intuition für einen derartigen Qualitätsbegriff, da $p_1 =_I p_2$ lediglich bedeutet, dass die beiden Strategien gleich gut sind und dafür müssen sie nicht identisch sein.

Definition 2.7 lässt jedoch noch viel Freiheit bei der Wahl einer Strategieordnung. So legt sie z. B. für Abbildung 2.2 nur im Fall $y < x + 1$ fest, dass die Strategie $p_1 =$ „gehe direkt zu t“ besser ist als die Strategie $p_2 =$ „probiere zunächst den unsicheren Weg“.

Die natürlichste Ordnung ist in diesem Fall die Betrachtung der erwarteten Kosten. Diese sind für p_1 genau y und für p_2 $\frac{1}{2}(1 + x) + \frac{1}{2}(2 + y) = \frac{1}{2}(x + y + 3)$. Somit wäre die Strategie p_1 besser als p_2, wenn $y < x + 3$ gilt.

Eine weitere wünschenswerte Eigenschaft, die eine Strategieordnung haben soll, wird im Folgenden definiert:

Definition 2.8 Eine Strategie p_1 für die Instanz I heißt *strikt besser* als die Strategie p_2 ($p_1 \lessdot_I p_2$), wenn gilt:

$$\forall r \in \mathcal{R}_I : \mathrm{cost}_{p_1}(r) \leq \mathrm{cost}_{p_2}(r) \wedge \exists r \in \mathcal{R}_I : \mathrm{cost}_{p_1}(r) < \mathrm{cost}_{p_2}(r)$$

Eine Strategieordnung $\leq_I$ heißt *strikt*, wenn gilt:

$$p_1 \lessdot_I p_2 \implies p_1 <_I p_2 \tag{2.2}$$

Für eine strikte Strategieordnung wird zusätzlich gefordert, dass eine Strategie, die nie schlechter und für manche Realisierungen besser ist als eine zweite, auch von der Strategieordnung echt besser bewertet werden muss.

In unendlichen Instanzen gibt es Realisierungen, in denen kein Pfad endlicher Länge zwischen s und t existiert. In einer solchen Realisierung erzeugt jede Strategie unendliche Kosten. Folglich sind die erwarteten Kosten für alle Strategien auf einer unendlichen Instanz unendlich, und so bewertet die Strategieordnung, die durch die erwarteten Kosten induziert wird, alle Strategien gleich.

Es liegt deswegen nahe, die unendlichen Realisierungen bei der Bildung des Erwartungswertes zu vernachlässigen. Der resultierende Ordnungsbegriff ist dann nicht in der Lage, alle Strategien so zu unterscheiden, wie es der Intuition entspricht, ermöglicht aber einen Vergleich aller relevanten Strategien.

Definition 2.9 Für eine Strategie p für die Instanz I sind die *erwarteten Kosten für endliche Realisierungen* definiert als

$$\begin{aligned} \mathrm{expcost}_I(p) &= E[\mathrm{cost}_p(r) | r \in \mathcal{R}_I^{<\infty}] \\ &= (P^I(\mathcal{R}_I^{<\infty}))^{-1} \sum_{r \in \mathcal{R}_I^{<\infty}} P(r) \cdot \mathrm{cost}_p(r) \quad . \end{aligned}$$

Ist $P^I(\mathcal{R}_I^{<\infty}) = 0$, so sei $\mathrm{expcost}_I(p) = 0$.

Für den Fall, dass in I sicher ein endlicher Pfad zwischen s und t existiert, also $\mathcal{R}_I = \mathcal{R}_I^{<\infty}$ gilt, entspricht $\mathrm{expcost}_I(p)$ dem normalen Erwartungswert $E[\mathrm{cost}_p(r)]$.

Definition 2.10 Die *kanonische Strategieordnung* für die Instanz I ist definiert als:

$$p_1 \leq_I p_2 \iff \mathrm{expcost}_I(p_1) \leq \mathrm{expcost}_I(p_2)$$

Satz 2.1 *Die kanonische Strategieordnung ist eine Strategieordnung.*

Beweis Reflexivität, Transitivität und Vergleichbarkeit sind klar. Wenn weiterhin $\forall r \in \mathcal{R}_I : \mathrm{cost}_{p_1} \leq \mathrm{cost}_{p_2}$, so gilt auch $\mathrm{expcost}_I(p_1) \leq \mathrm{expcost}_I(p_2)$ und damit $p_1 \leq p_2$, weil der bedingte Erwartungswert eine Linearkombination positiver Werte mit nicht-negativen Koeffizienten ist.

Die gerade definierte Strategieordnung ist nicht strikt. Jedoch ist sie für Strategien aus einer Menge strikt, in der alle interessanten Strategien liegen.

Definition 2.11 Eine Strategie p für I heißt *sinnvoll*, wenn für alle endlichen Realisierungen $r \in \mathcal{R}_I^{<\infty}$ gilt: $\mathrm{cost}_p(r) < \infty$

In Worten heißt dass, dass eine sinnvolle Strategie keine Kante mit unendlichem Gewicht wählt, wenn es noch die Möglichkeit gibt, mit endlichen Kosten zum Ziel zu kommen. Das ist für jede vernünftige Strategie der Fall.

Satz 2.2 *Die Einschränkung der kanonischen Strategieordnung auf die sinnvolle Strategien ist strikt.*

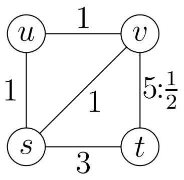

Abbildung 2.3.

	r_1	r_2
$c_{\{v,t\}}$	5	∞
p_1	7	∞
p_2	∞	5
p_3	6	∞

Tabelle 2.1.: Kosten für p_1, p_2 und p_3 in Abb. 2.3

Beweis Gelte $p_1 \lessdot p_2$ für $p_1, p_2 \in \mathcal{S}_I$. Da p_i sinnvoll, gilt $\mathrm{cost}_{p_i}(r) < \infty$ für $i \in \{1,2\}$ und für alle Realisierungen $r \in \mathcal{R}_I^{<\infty}$. Weiterhin gilt

$$\begin{aligned} E[\mathrm{cost}_{p_1}(r) | r \in \mathcal{R}_I^{<\infty}] \cdot P^I(\mathcal{R}_I^{<\infty}) &= \sum_{r \in \mathcal{R}_I^{<\infty}} P^I(r) \cdot \mathrm{cost}_{p_1}(r) \\ &< \sum_{r \in \mathcal{R}_I^{<\infty}} P^I(r) \cdot \mathrm{cost}_{p_2}(r) \\ &= E[\mathrm{cost}_{p_2}(r) | r \in \mathcal{R}_I^{<\infty}] \cdot P^I(\mathcal{R}_I^{<\infty}) \quad , \end{aligned}$$

da $p_1 \lessdot p_2$. Also gilt $p_1 < p_2$.

Für allgemeine Strategien ist die kanonische Strategieordnung jedoch nicht strikt, wie man an der Instanz aus Abbildung 2.3 für folgende drei Strategien sieht:

p_1: Gehe über u und v zu t.

p_2: Gehe zu v. Wenn $\{v,t\}$ die Kosten 5 hat, pendle zwischen s und v, sonst gehe über s zu t.

p_3: Gehe über v zu t.

Für diese Instanz gibt es zwei Realisierungen r_1 und r_2, die beide endlich sind. Der einzige Unterschied der beiden Realisierungen besteht aus den Kosten für die Kante $\{v,t\}$. Die Kosten der drei Strategien sind in Tabelle 2.1 aufgelistet.

Da jede Strategie in einer der beiden endlichen Realisierungen die Kosten ∞ hat, werden alle drei von der kanonischen Strategieordnung gleich bewertet. Für eine strikte Ordnung muss aber $p_3 < p_1$ gelten.

Weiterhin muss für eine strikte Ordnung wegen der Transitivität und $p_3 < p_1$ auch $p_2 < p_1$ oder $p_3 < p_2$ gelten. Eine derartige Festlegung wirkt allerdings sehr künstlich.

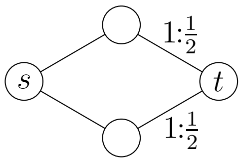

Abbildung 2.4.

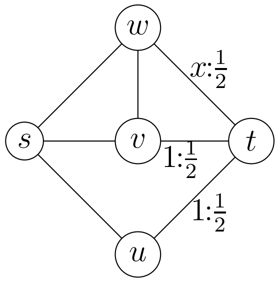

Abbildung 2.5.: Graph G_x

Die kanonische Strategieordnung lässt sich so verfeinern, dass echte Ungleichungen erhalten bleiben und eine strikte Strategieordnung $\preceq_I$ entsteht. Sei dazu $(r_i)_{i\in\mathbb{N}}$ eine Aufzählung von $\mathcal{R}_I^{<\infty}$. Für zwei Strategien p und q, die nicht auf allen endlichen Realisierungen die gleichen Kosten erzeugen, sei dann $\preceq_I$ definiert als:

$$p \preceq_I q \iff p <_I q \vee (p \leq q \wedge \mathrm{cost}_p(r_\iota) < \mathrm{cost}_q(r_\iota)$$

mit $\iota = \min_{i\in\mathbb{N}}\{\mathrm{cost}_p(r_i) \neq \mathrm{cost}_q(r_i)\} < \infty$. Dass mit $\preceq_I$ tatsächlich eine strikte Strategieordnung vorliegt, rechnet man leicht nach.

Durch die Aufzählung der endlichen Realisierungen wird eine Ordnung auf $\mathcal{R}_I^{<\infty}$ definiert, mit deren Hilfe die Kosten auf bestimmten Realisierungen mehr Gewicht in der Bewertung einer Strategie erhalten als andere.

Diese Ordnung ist aber sehr unnatürlich, da damit z. B. für die Instanz aus Abbildung 2.4 die beiden Strategien „Gehe über den oberen Knoten zu t“ und „Gehe über den unteren Knoten zu t“ nicht gleich gut sind.

Da aber sinnvolle Strategien trotzdem bzgl. der kanonischen Strategieordnung besser sind als Strategien, die nicht sinnvoll sind, und es nicht schwer ist, sinnvolle Strategien zu finden, ist es keine Einschränkung, dass im Allgemeinen keine Striktheit vorliegt.

Dadurch, dass bei der Bestimmung der kanonischen Strategieordnung nur das Verhalten auf endlichen Realisierungen relevant ist, ergibt sich, dass sie unter Veränderung der Kostenfunktion nicht stetig ist.

Seien p_1 und p_2 zwei Strategien der Instanz G_x aus Abbildung 2.5 mit

p_1: Gehe über u, s, v, w zu t. Wenn eine der Kanten $\{u, t\}$ und $\{v, t\}$ ein endliches Gewicht hat, verwende jedoch diese sobald u bzw. v erreicht wird.

p_2: analog zu p_1, aber mit der zu Grunde gelegten Reihenfolge v, s, u, s, w.

Für diese gilt mit $x \in \mathbb{R}^+$:

$$\begin{aligned}
\mathrm{expcost}_{G_x}(p_1) &= \frac{8}{7}\left(\frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 4 + \frac{1}{8}\cdot(4+x)\right) = \frac{20+x}{7} \\
\mathrm{expcost}_{G_x}(p_2) &= \frac{8}{7}\left(\frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 4 + \frac{1}{8}\cdot(5+x)\right) = \frac{21+x}{7}
\end{aligned}$$

Somit ist also $p_1 <_{G_x} p_2$ unabhängig von x. Für $x = \infty$ hat die Kante $\{w, t\}$ jedoch sicher die Kosten ∞ und die erwarteten Kosten ergeben sich für $i = 1, 2$ durch:

$$\mathrm{expcost}_{G_\infty}(p_i) = \frac{4}{3}(\frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 4) = \frac{8}{3} \quad .$$

In dieser Instanz gilt demnach $p_1 =_{G_\infty} p_2$.

3. Algorithmen

In diesem Kapitel werden Algorithmen zur Lösung einer beliebigen Instanz des CTP vorgestellt. Algorithmen sind ein allgemeineres Konzept als Strategien, da ein Algorithmus ein Schema ist, aus dem man für jede Instanz eine Strategie ableiten kann.

Im letzen Abschnitt dieses Kapitels wird die Implementierung der einzelnen Strategien beschrieben und mit Testläufen auf zufällig generierten Graphen verglichen.

Definition 3.1 Ein Algorithmus OPT heißt *optimal*, wenn $\mathrm{expcost}_I(\mathrm{OPT})$ für alle Instanzen I minimal ist.

Es gibt nicht nur einen optimalen Algorithmus, jedoch stimmen bei allen die erwarteten Kosten für alle Instanzen überein, deswegen führt es nicht zu Uneindeutigkeiten, wenn im Folgenden von *dem* optimalen Algorithmus OPT gesprochen wird. Dass solch ein Algorithmus existiert wird konstruktiv im nächsten Abschnitt gezeigt.

Definition 3.2 Ein Algorithmus ALG heißt *r-approximativ* für ein $r \in \mathbb{R}^+$, wenn

$$r = \sup_{I \in \mathrm{CTP}} \frac{\mathrm{expcost}_I(\mathrm{ALG}(I))}{\mathrm{expcost}_I(\mathrm{OPT}(I))}$$

Weiterhin heißt ALG *approximativ*, wenn ALG r-approximativ für ein $r \in \mathbb{R}^+$ ist.

In der Beschreibung der Algorithmen werden einige Hilfsfunktionen verwendet:

- Die Nachbarn $\mathrm{neigh}(v)$ des Knotens $v \in V$ sind gegeben durch

$$\mathrm{neigh}(v) = \{v' \in V : \{v, v'\} \in E\}$$

- Die minimale Distanz $d_{\gamma^*}(w, v)$ des Knotens $w \in V$ zum Knoten $v \in V$ bei einer gegebenen Festlegung der Gewichte aller Kanten $\gamma^* : E \to \overline{\mathbb{R}}$ erfüllt für $w \neq v$ die folgenden Bedingungen.

$$\begin{aligned} d_{\gamma^*}(w, v) &= \min_{w' \in \mathrm{neigh}(w)} d_{\gamma^*}(w', v) + \gamma^*_{\{w,w'\}} \\ d_{\gamma^*}(v, v) &= 0 \end{aligned}$$

  Zur Berechnung von d_{γ^*} genügen diese beiden Gleichungen nicht, weil sie die Werte für alle Nachbarn eines Knotens benötigt und für deren Berechnung wiederum der ursprüngliche Wert verwendet wird. Dieses Problem lässt sich lösen, indem man zunächst alle Werte als ∞ annimmt und sie dann an den Stellen, die die Gleichungen verletzen, entsprechend anpasst, bis alle Werte die Gleichungen erfüllen.

Nach diesem Prinzip geht auch der Algorithmus von Dijkstra vor, der diese Funktion d_{γ^*} berechnet, wenn er γ^* als Kostenfunktion verwendet. Für eine Pseudocode-Implementierung siehe Anhang A.

- Die noch möglichen, endlichen Realisierungen von I bei Wissen $\gamma \in \overline{\mathbb{R}}^E$ sind definiert als

$$R_\gamma = \{r \in \mathcal{R}_I^{<\infty} : \forall e \in E : c_e(r) \in \operatorname{supp}\gamma\} \quad .$$

Wenn das Wissen γ, das der Agent angesammelt hat richtig ist, ist die tatsächliche Realisierung – so sie endlich ist – aus R_γ. Für alle anderen (endlichen) Realisierungen wäre das Wissen falsch.

Für die nun folgenden Beschreibung der Algorithmen sei eine feste Instanz $I = \langle V, E, c, s, t \rangle \in \mathrm{CTP}$ gegeben. Die Algorithmen können dann als Strategien in Abhängigkeit von I angegeben werden.

3.1. Die optimale Strategie

Die optimale Strategie wählt ihre Schritte so, dass die erwarteten Kosten für die Lösung minimal sind. Dazu muss für jeden Knoten $v \in V$ und jeden Wissensstand $\gamma \in \mathbb{K}^E$ die minimale erwartete Distanz $\delta_\gamma(v)$ zum Ziel t berechnet werden.

Für $v' \in \operatorname{neigh}(v)$ sei definiert:

$$\begin{aligned}
\psi_\gamma(v, v') &= \gamma^*_{\{v,v'\}} + \frac{1}{P(R_\gamma)} \sum_{r \in R_\gamma} P(r) \cdot \delta_{\gamma[r_{v'}]}(v') \\
\delta_\gamma(v) &= \min_{v' \in \operatorname{neigh}(v)} \psi_\gamma(v, v') \\
\delta_\gamma(t) &= 0 \quad \text{mit} \\
r_{v'} &= (c_e(r)^\circ)_{e \in \operatorname{neigh}(v')}
\end{aligned}$$

$\psi_\gamma(v, v')$ sind dabei die erwarteten Kosten, die einem Agenten entstehen, wenn er von v nach v' geht und sich anschließend optimal verhält, und $\delta_\gamma(v)$ sind damit die erwarteten Kosten der optimalen Strategie.

Für die Berechnung von $\delta_\gamma(v)$ werden nur Werte $\delta_{\gamma'}(v')$ verwendet, für die entweder $\gamma' = \gamma$ gilt oder in denen γ' mehr konstante Zufallsvariablen enthält als γ. Wenn alle Zufallsvariablen aus γ konstant sind, ist $\delta_\gamma(v) = d_\gamma(v, t)$, weil dann die Summe nur noch aus einem Summanden besteht und $\gamma[r_{v'}] = \gamma$ ist. Somit lässt sich $\delta_\gamma(v)$ mit einem Verfahren berechnen, das analog zu dem aus der Berechnung von d_γ ist.

Damit lässt sich nun p_{OPT} für $v \neq t$ so definieren, dass $p_{\mathrm{OPT}}(v, \gamma) = v'$ für ein $v' \in \operatorname{argmin}_{v' \in \mathrm{edge}(v)} \psi_\gamma(v, v')$ ist. Falls es mehr als eine Möglichkeit gibt, einen Knoten zu wählen, wird ein beliebiger Kandidat ausgewählt. Die Qualität der Strategie ist davon unabhängig und die Wahl der Kandidaten stellt den einzigen Unterschied zwischen zwei optimalen Algorithmen dar. Der optimale Algorithmus ist trivialerweise 1-approximativ.

Um für eine Instanz I die optimale Strategie auf diese Weise zu berechnen, müssen exponentiell in $|V|$ viele Berechnungsschritte durchgeführt werden. Nach Papadimitriou und Yannakakis[6] ist die Aufgabe, die optimale Strategie zu finden, #P-hart und in PSPACE lösbar.

3.2. Die optimistische Strategie

Die optimistische Strategie wählt immer eine der Kanten, über die ein kürzester Pfad vom momentanen Standpunkt zum Ziel möglich ist.

Im Knoten v mit dem Wissensstand γ wählt sie eine Kante $\{v, v'\} \in E$ für die

$$v' \in \operatorname*{argmin}_{v' \in \mathrm{edge}(v)} \gamma_{\{v,v'\}} + d_{\overline{\gamma}}(v', t)$$

mit

$$\overline{\gamma} = \left((\min \operatorname{supp} c_e)^\circ\right)_{e \in E}$$

gilt.

In dem Fall, dass allen unsicheren Kanten die minimal möglichen Kosten zugewiesen werden, findet der Algorithmus einen kürzesten Weg. In diesen Fall ist die angenommene Kostenzuweisung $\overline{\gamma}$ für die Kanten, die tatsächlich traversiert werden, richtig. Wenn eine Kante auf dem Weg teurer ist als angenommen, wird vom dann aktuellen Standort von neuem der kürzest mögliche Pfad bestimmt und ausprobiert.

Da der Agent in jedem Knoten nur einmal zusätzliche Informationen bekommen kann und er nur neu planen muss, wenn er entdeckt, dass eine Kante teurer ist als angenommen, muss er für jeden von t verschiedenen Knoten maximal einmal einen Plan erstellen. Deswegen wird er, spätestens nachdem sich $|V| - 2$ kürzeste Wege als zu teuer erwiesen haben, mit Sicherheit den Zielpunkt erreichen. Weiterhin ist die optimistische Strategie sinnvoll, weil erst dann eine Kante mit unendlichen Kosten traversiert wird, wenn der kürzest mögliche Weg vom aktuellen Standpunkt zum Ziel keine endliche Länge hat. Weil bis dahin nur (endlich viele) Kanten endlichen Gewichts verwendet wurden, existiert dann auch kein endlicher Pfad zwischen s und t.

Da spätestens der $(|V| - 1)$-te Plan den Agenten wirklich zu t führt, keine Kante mit unendlichen Kosten verwendet wird, wenn die tatsächliche Realisierung endlich ist und jede Kante in einem möglichen kürzesten Pfad nur einmal verwendet wird, erzeugt die optimistische Strategie auf einer endlichen Realisierung nicht mehr Kosten als

$$(|V| - 1) \cdot \sum_{e \in E} \max \operatorname{fsupp} c_e \quad .$$

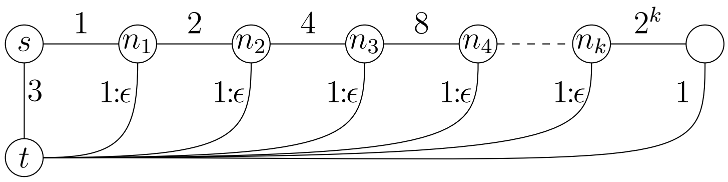

Abbildung 3.1.: Graph $G_{k,\epsilon}$

Für den Graphen $G_{k,\epsilon}$, der in Abbildung 3.1 dargestellt ist, hat der optimale Pfad höchstens Kosten 3.

Wenn ein Agent, der der optimistischen Strategie folgt, sich im Punkt n_i ($i = 1, 2, \dots$) befindet, verwendet er den direkten Weg zu t, so dieser die Kosten 1 hat. Weiterhin haben für $j < i$ die Kanten $\{n_j, t\}$ die Kosten ∞, da der Agent sonst diese Kante gewählt und damit sein Ziel erreicht hätte.

Wenn der direkte Weg zu t von n_i aus blockiert ist, geht er weiter nach links, weil der Pfad $n_i \to n_{i+1} \to t$ möglicherweise die Kosten $2^i + 1$ hat. Dem gegenüber hat der (einzige endliche) Pfad, der n_{i+1} nicht enthält die Länge $\sum_{j=0}^{i-1} 2^j + 3 = 2^i + 2$. Aus dem gleichen Grund fällt in s die Entscheidung, nach n_1 zu gehen anstatt direkt nach t.

Die Kante $\{n_i, t\}$ wird also genau dann verwendet, wenn sie zum einen frei ist und zum anderen die Kanten $\{n_j, t\}$ für $j < i$ blockiert sind. Die Wahrscheinlichkeit dafür ist $(1-\epsilon)^{i-1}\epsilon$ und der zurückgelegte Weg hat dann die Länge 2^i. Wenn alle unsicheren Kanten blockiert sind, hat der gewählte Weg die Länge 2^{k+1}. Die optimistische Strategie hat also erwartete Kosten von

$$\sum_{i=1}^{k} (1-\epsilon)^{i-1}\epsilon \cdot 2^i + (1-\epsilon)^k \cdot 2^{k+1} = k + 2 \text{ mit } \epsilon = \frac{1}{2}$$

Das bedeutet, dass ein Approximationsfaktor r für die optimistische Strategie größer oder gleich $\frac{k+2}{3}$ für jedes $k \in \mathbb{N}$ sein muss. Damit die die Strategie nicht approximativ. Die Lösung, die mit der optimistischen Strategie gefunden wird, kann also im Vergleich zur optimalen Lösung beliebig schlecht werden.

An diesem Beispiel zeigt sich deutlich die Schwäche des optimistischen Algorithmus: Für Weg, der mit einer beliebig kleinen Wahrscheinlichkeit größer als Null kurz ist, aber sonst u. U. sehr lang ist, wird ein sicherer, nur wenig längerer Weg verworfen. Das kommt daher, dass weder die Wahrscheinlichkeit, dass der gewählte Weg tatsächlich minimal kurz ist, noch die Kosten für die Fälle, in denen die Kante länger ist, mit in die Pfadplanung aufgenommen werden.

3.3. Die Greedy-Strategie

Die Greedy-Strategie stimmt weitesgehend mit der optimistischen überein. Der einzige Unterschied ist die verwendete Gewichtsfunktion, mit der die Kosten der Kanten abgeschätzt werden. Dadurch wird versucht, die Schwächen des optimistischen Algorithmus zu kompensieren.

Diese Gewichte werden so gewählt, dass Kanten, die sicher ein bestimmtes Gewicht haben, eher verwendet werden als Kanten, die dieses Gewicht mindestens haben.

Genauer gesagt wird für zwei Parameter $a \in \mathbb{R}^{\geq 1}$ und $b \in \mathbb{R}^{+}$ γ_e^* mit $b_e = \max(\{b\} \cup \operatorname{fsupp} c_e$ wie folgt definiert:

$$\gamma_e^* = \begin{cases} c_e^* & \text{falls } c_e \text{ konstant} \\ a\left(\sum_{x \in \mathbb{R}^+} c_e(x) \cdot x + c_e(\infty) \cdot b_e\right) & \text{sonst} \end{cases}$$

Wenn $a = 1$ und b mindestens $\max \bigcup_{e \in E} \operatorname{fsupp} c_e$ ist, entspricht die Gewichtung gerade den erwarteten Kosten für diese Kante, wobei der Wert ∞ durch b abgeschätzt wird. Für kleinere b wird ∞ mit $\max \operatorname{fsupp} c_e$ bewertet. Durch den Parameter a kann man festlegen, dass sichere Kanten unsicheren vorgezogen werden. Für eine gegebene Instanz kann man a so wählen, dass gar keine unsicheren Kanten in einen Plan aufgenommen werden, wenn es von Beginn an einen endlichen Pfad gibt, der nur aus sicheren Kanten besteht.

Dass diese Strategie schließlich zum Ziel führt, überlegt man sich analog zur optimistischen Strategie.

Wenn man in dem Graphen $G_{k,\epsilon}$ aus Abbildung 3.1 die Kosten der Kante zwischen s und t durch $a(\epsilon + (1 - \epsilon) \cdot \max\{b, 1\}) + 2$ ersetzt, sieht man auch für die Greedy-Strategie, dass sie beliebig schlechte Lösungen produziert, da sie sich mit der gleichen Begründung auf dem veränderten Graphen genauso verhält wie die optimistische Strategie auf $G_{k,\epsilon}$.

3.4. Die Monte-Carlo-Strategie

Die Monte-Carlo-Strategie hat einen Parameter, der die Anzahl der Samples $n \in \mathbb{N}$ angibt. Um in einer Situation den nächsten Schritt zu bestimmen, wird n-mal – gemäß den durch c gegebenen Wahrscheinlichkeiten – für jede noch unbekannte Kante zufällig ein Gewicht bestimmt. Dann wird für alle vom momentanen Standpunkt aus direkt erreichbaren Knoten v' bzgl. des geratenen Wissensstandes die kürzeste Distanz zu t bestimmt. Als nächster Knoten wird dann einer derjenigen gewählt, die im Mittel der n Belegungen den kürzesten Weg zu t versprechen.

Durch die zufällige Komponente ist die Monte-Carlo-Strategie eigentlich keine Strategie, da bei einer Wiederholung eines Zustandes nicht unbedingt noch einmal die gleiche Entscheidung getroffen wird, wie beim ersten Auftreten. Deswegen ist der resultierende Plan keine Funktion von Zuständen auf Knoten. Trotzdem ist klar, dass die Monte-Carlo-Strategie eine definierte Handlungsweise vorschreibt.

Im Zustand $\langle v, \gamma \rangle$ werden also n Funktionen $(\gamma^i)_{i=1}^n$ bestimmt, mit:

$$\begin{aligned} \gamma^i &: E \to \overline{\mathbb{R}} \\ \gamma^i &= x \quad \text{mit Wahrscheinlichkeit } c_e(x) \text{ sonst} \end{aligned}$$

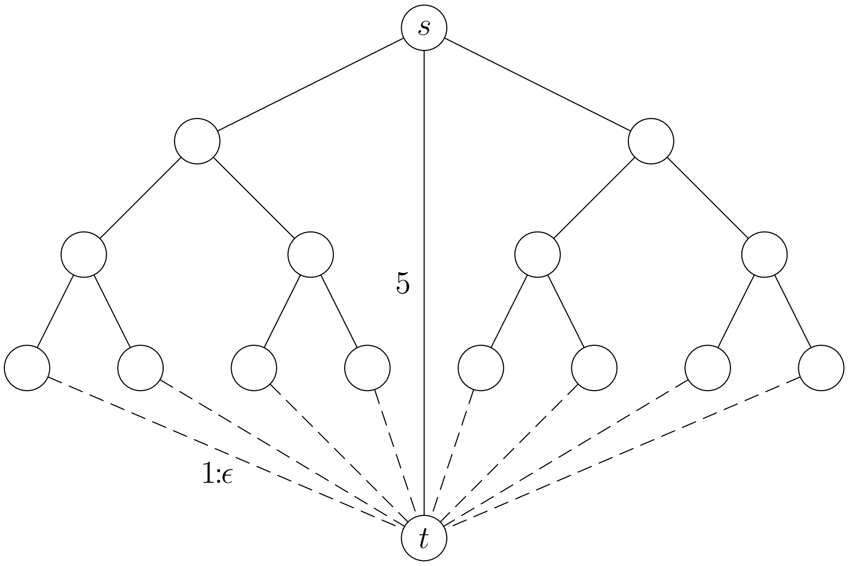

Abbildung 3.2.: $G_{3,\epsilon}$

Als nächstes wird dann eine Kante verwendet, die zu einem Knoten aus

$$\operatorname*{argmin}_{v' \in \text{neigh}(v)} \sum_{i=1}^{n} \left(\gamma_{\{v,v'\}} + d_{\gamma^i}(v', t) \right)$$

führt.

Sei für $k \in \mathbb{N}$ und $\epsilon \in [0, 1]$ die Instanz $G_{k,\epsilon}$ gegeben, in der die Knoten ohne t einen Binärbaum der Höhe k mit der Wurzel s bilden. Die Kanten innerhalb des Baumes haben Kosten 1 und die untersten Knoten sind jeweils mit einer Kante des Gewichts $1 : \epsilon$ mit t verbunden. Weiterhin existiere eine Kante zwischen s und t mit Kosten $k + 2$. Abb. 3.2 zeigt $G_{3,\epsilon}$.

In einem CTP dieser Art hat der optimale Algorithmus erwartete Kosten von nicht mehr als $k + 2$, weil die Strategie „Wähle sofort die sichere Kante der Länge $k + 2$“ immer genau diese Kosten erzeugt. In diesem Graphen kann man, indem man den Baum wie in der Tiefensuche durchläuft, mit Kosten $4 \cdot (2^k - 1)$ zurück zur Wurzel kommen und hat auf dem Weg alle Blätter des Baumes besucht und so für alle unsicheren Kanten deren Länge erfahren. Das ist genau das Vorgehen der optimistischen Strategie, wenn alle unsicheren Kanten blockiert sind. Bei einem solchen Vorgehen wird jede der $2 \cdot (2^k - 1)$ Kanten des Baumes genau zweimal traversiert und somit entstehen einem „optimistischen“ Agenten höchstens Kosten $4 \cdot (2^k - 1) + k + 2$.

Von einem inneren Knoten der Tiefe i geht der kürzeste Weg zu t über einen der beiden Söhne genau dann, wenn für mindestens einen Nachkommen die unsichere Kante die Länge 1 hat. Dieser kürzeste Weg hat die Länge $k + 1 - i$, von s aus also $k + 1$. Wird auf $G_{k,\epsilon}$ die Monte-Carlo-Strategie mit nur einem Sample verwendet, so wird in einem inneren Knoten der Tiefe i also einer der beiden Söhne gewählt, wenn

k	ϵ	s_0
2	$1.81 \cdot 10^{-1}$	4.9
3	$1.67 \cdot 10^{-1}$	9.3
4	$1.18 \cdot 10^{-1}$	21.5
5	$7.84 \cdot 10^{-2}$	65.9
6	$5.11 \cdot 10^{-2}$	$2.81 \cdot 10^{2}$
7	$3.28 \cdot 10^{-2}$	$1.71 \cdot 10^{3}$
8	$2.08 \cdot 10^{-2}$	$1.50 \cdot 10^{4}$
9	$1.30 \cdot 10^{-2}$	$1.97 \cdot 10^{5}$

Tabelle 3.1.: abgeschätzte Kosten der Monte-Carlo-Strategie mit einem Sample

mindestens eine der 2^{k-i} unsicheren Kanten unterhalb des Standortes unblockiert ist. Die Wahrscheinlichkeit dafür ist

$$p_i = 1 - (1-\epsilon)^{2^{k-i}} \quad .$$

Die erwarteten Kosten, die dem Agenten entstehen, wenn er in einem Knoten der Tiefe i startet, lassen sich durch die erwarteten Kosten s_i unter der Annahme, dass der Agent mit Kosten 1 das Ziel erreichen kann, wenn er an einem der unteren Knoten angekommen ist, abschätzen. Für den Fall, dass die entsprechende Kante frei ist, ist die Annahme richtig, sonst entstehen aber noch mindestens Kosten 3 zusätzlich.

Für $i = 1, 2, \ldots k-1$ gilt somit:

$$\begin{aligned}
s_0 &= p_0 \cdot (s_1 + 1) + (1 - p_0) \cdot (k+2) \\
&= (1-\epsilon)^{2^k} \cdot (k+2) + (1 - (1-\epsilon)^{2^k}) \cdot (s_1 + 1) \\
s_i &= p_i \cdot s_{i+1} + (1 - p_i) \cdot s_{i-1} + 1 \\
&= (1 - (1-\epsilon)^{2^{k-i}}) \cdot s_{i+1} + (1-\epsilon)^{2^{k-i}} \cdot s_{i-1} + 1 \\
s_k &= 1
\end{aligned}$$

Dabei gibt der Wert s_0 eine untere Schranke für die erwarteten Kosten der gesamten Instanz an.

Der Tabelle 3.1 können einige numerische Lösungen für s_0 entnommen werden. Der Parameter ϵ wurde dabei mit dem Näherungsverfahren aus Abb. 3.3 bestimmt. Unter der Annahme, dass es nur ein $\epsilon \in [0, 1]$ gibt, für das s_0 ein lokales Maximum annimmt, unterscheidet sich der gefundene Wert um nicht mehr als 10^{-6} davon. Ist $k \geq 6$, ist die optimistische Strategie auch im schlechtesten Fall besser.

Für Werte von k zwischen 2 und 6 zeigt Abb. 3.4, dass tatsächlich nur ein einziges Maximum vorliegt. Das ist auch deshalb plausibel, weil für wachsendes ϵ die Wahrscheinlichkeit innerhalb des Baumes abzusteigen wächst. So wird für $\epsilon = 0$ auf jeden Fall die sichere Kante der Länge $k + 2$ gewählt. Für ein sehr kleines, wachsendes ϵ wächst dann auch die Wahrscheinlichkeit dafür, dass von s nicht die sichere Kante

```
def search_epsilon(k):
  low, high = 0.0, 1.0
  delta = 1.0
  while delta > 1e-6:
    mc_low = mc(samples=1, height=k, epsilon=low + delta / 3)
    mc_high = mc(samples=1, height=k, epsilon=low + 2 * delta / 3)
    if ed_low > ed_high:
      high = low + 2 * delta / 3
    else:
      low = low + delta / 3
    delta = high - low
  return low
```

Abbildung 3.3.: Näherungsverfahren zur Bestimmung von ϵ

gewählt wird. Aber trotzdem findet in den meisten dieser Fälle anschließend wieder eine Rückkehr zu s statt und die sichere Kante wird verwendet. Dadurch entstehen in wenigen Fällen die Kosten $k + 4$ (oder mehr, wenn wiederholt und/oder tiefer abgestiegen wird) und in noch wesentlich weniger Fällen $k + 1$, in denen vollständig und ohne zwischenzeitliches Aufsteigen ein Pfad über eine unsichere Kante gewählt wird. Also steigen die erwarteten Kosten über $k+2$. Zugleich sinkt mit wachsendem ϵ aber die Wahrscheinlichkeit, im Baum wieder aufzusteigen, da es sicherer wird, dass die noch in Frage kommenden unsicheren Kanten in der geratenen Belegung passierbar sind. So konvergieren diese abgeschätzten Kosten – wie auch die tatsächlichen – gegen $k + 1$ für ϵ gegen 1.

Wird in solchen Instanzen jedoch mehr als ein Sample gemacht, fällt der Wert des Maximums sehr schnell, wie in Abbildung 3.5 ersichtlich wird. Hier wurde für eine Instanz, in der die Baumhöhe 3 ist, s_0 gegen ϵ für Samplezahlen zwischen 1 und 6 aufgetragen.

Tatsächlich sind die erwarteten Kosten in einer solchen Instanz jedoch wesentlich höher, beispielsweise erzeugte der Monte-Carlo-Algorithmus in der Implementierung für $\epsilon = 0.2$, $k = 3$ mit 6 Samples in 100 Läufen im Durchschnitt die Kosten 14.2. Mit $k = 5$, $\epsilon = 0.1$ und ebenfalls 6 Samples wurde schon ein Schnitt von 62534 erreicht. Mit $\epsilon = 0.05$, $k = 10$ und 6 Samples lieferte die Implementierung der Monte-Carlo-Strategie in einer akzeptablen Zeit bereits kein Ergebnis mehr. So wurde in 3 Testläufen jeweils abgebrochen nachdem mehr als Kosten 500000 entstanden waren und noch keine unsichere Kante erreicht wurde.

Im Gegensatz zur Greedy- und zur optimistischen Strategie ist die Monte-Carlo-Strategie nicht sinnvoll. Wenn nämlich keines der Samples eine endliche Realisierung rät, obwohl eine solche möglich ist, kann es passieren, dass eine Kante unendlichen Gewichts gewählt wird. Um das zu umgehen, wird in der Implementierung dieser Strategie vor der Generierung der Samples überprüft, ob noch eine endliche Realisierung möglich ist. Wenn daraufhin nach n Samples keine endliche Realisierung

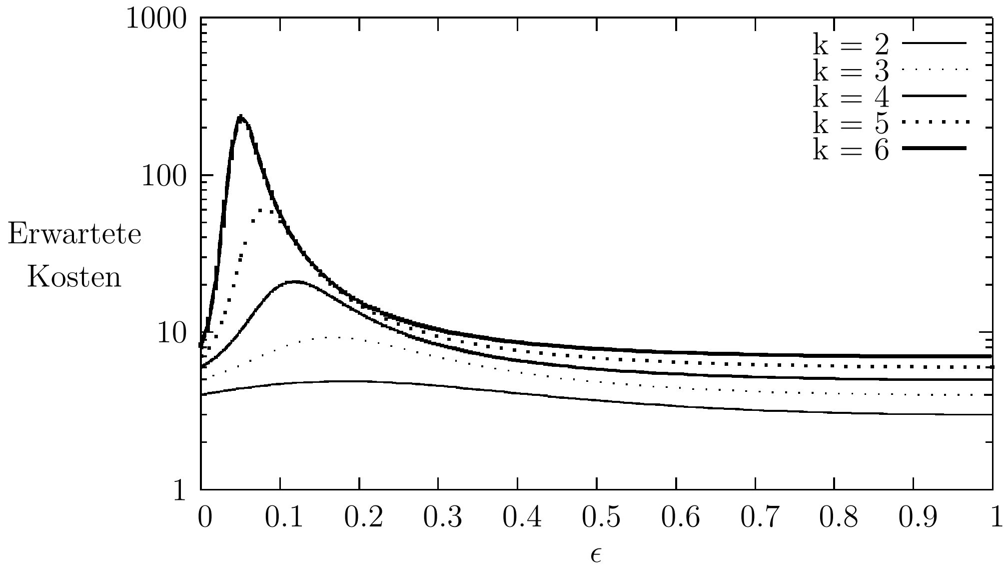

Abbildung 3.4.: Erwartete Kosten mit einem Sample

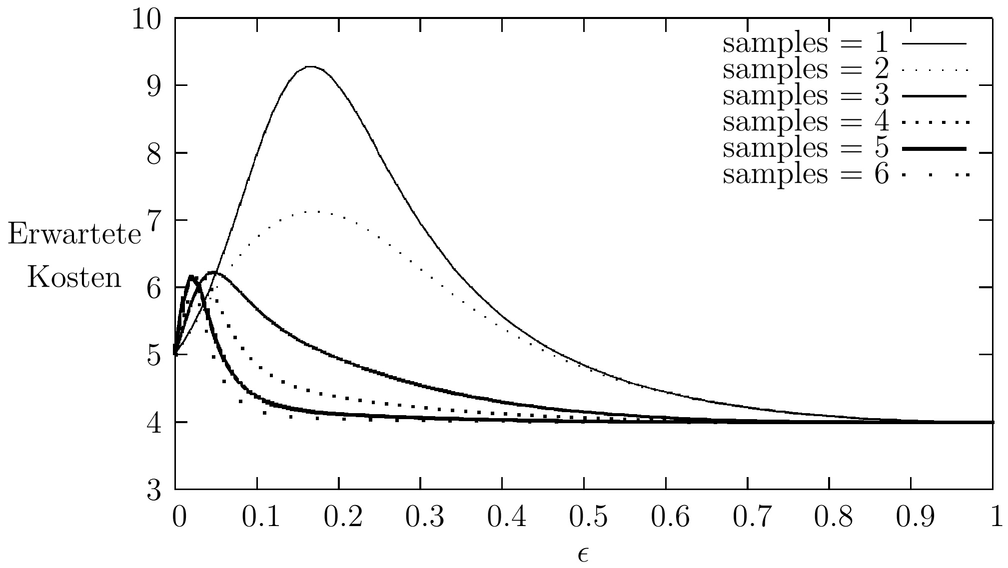

Abbildung 3.5.: Erwartete Kosten in $G_{3,\epsilon}$

erzeugt wurde, werden weitere Samples generiert, bis ein endliches gefunden wird. Ein weiterer Schwachpunkt der Monte-Carlo-Strategie ist die Tatsache, dass der Agent möglicherweise sehr oft zwischen zwei oder mehr Knoten pendelt, ohne dass unbekannte Kanten exploriert werden. In den Tests kam dieses auch sehr häufig vor, so dass diese Strategie sehr viel schlechter abschnitt als die Greedy-Varianten.

Wenn in einer Situation schon so viele Kanten ein bekanntes Gewicht haben, dass $\prod_{\{e \in E\}} |\operatorname{supp} c_e|$ nicht größer als die Anzahl der Samples ist, so liegt die Berechnung der optimalen Strategie in dieser Situation in der gleichen Grössenordnung wie die der Samples. Man kann also ohne einen Mehraufwand an Rechenzeit in diesen Fällen die exakte Lösung berechnen und damit den Algorithmus im Allgemeinen verbessern. Das ist jedoch nur für kleine Instanzen nützlich, da es in großen meistens nicht dazu kommt, dass nur noch für wenige Kanten die Kosten nicht bekannt sind.

Eine Möglichkeit für die Verbesserung der Monte-Carlo-Strategie wäre, in einem Zustand nicht die direkt benachbarten Knoten, sondern all diejenigen zu betrachten, die sicher endlich erreichbar sind und die noch nicht besucht wurden. Dadurch würde sich das Verhalten so verändern, dass auf endlichen Realisierungen spätestens nach $|V|$ Iterationen ein Weg zum Ziel gefunden würde. Ein Pendeln zwischen Knoten wäre dadurch ausgeschloßen.

3.5. Implementierung der Strategien

Die drei vorgestellten Strategien wurden in der Sprache C implementiert und auf zufälligen Graphen getestet.

Die Graphen wurden mit Hilfe der Algorithmenbibliothek LEDA[2] generiert. Diese Bibliothek stellt Funktionen zur Verfügung, mit denen zufällige, ungewichtete Graphen generiert werden können. Um daraus Eingaben für das Canadian Traveller's Problem zu entwerfen, wird für jede Kante eine zufällige, ganze Zahl c zwischen Eins und 100 generiert. Eine weitere Zufallszahl aus dem Intervall $[0, 1)$ legt fest, ob die Kante sicher oder unsicher ist. Ist diese zweite Zufallszahl kleiner als eine vorgegebene Rate, wird der Kante das Gewicht $\{c \mapsto \frac{1}{2}, \infty \mapsto \frac{1}{2}\}$ zugewiesen, ansonsten $\{c \mapsto 1\}$. Dadurch repräsentieren die unsicheren Kanten in etwa den Anteil an der Gesamtzahl der Kanten, der der Rate entspricht.

Getestet wurde auf je zehn Graphem mit 1000 und 10000 Knoten, jeweils 3-, 7- und 20-mal so vielen Kanten und Raten 1, $\frac{3}{4}$ und $\frac{1}{2}$. Um ein zuverlässiges Ergebnis zu erhalten, wurde für jeden Graphen jeder Algorithmus jeweils 10-mal getestet.

Bei Raten von $\frac{1}{2}$ und weniger führte das in allen Fällen zu Graphen, bei denen wie beim ersten Beispiel dieser Arbeit, der kürzeste mögliche Weg zum Ziel auch sicher war. Bei gut einem Viertel der Graphen mit einer Rate von $\frac{3}{4}$ war das ebenso der Fall.

3.5.1. Die optimale Strategie

Der implementierte Algorithmus für die optimale Strategie berechnet nur Werte, die auch tatsächlich in die Bestimmung des Planes einfließen. Dazu wird ausgehend vom Startwissen für alle erreichten Zustände die Reihenfolge der zu expandierenden Möglichkeiten mit einer Heuristik bestimmt. Diese Heuristik entspricht der Berechnung der optimistischen Strategie. Dadurch kann die Suche abgebrochen werden, sobald ein Plan gefunden wurde, dessen erwartete Kosten kleiner oder gleich aller Bewertungen der Heuristik für die noch unexpandierten Möglichkeiten ist. Ausserdem wurden für Kanten erst dann die möglichen Belegungen festgesetzt und der Reihe nach durchprobiert, wenn die Kante von der optimistischen Strategie mit verwendet wurde. Es wurde also zwischen Kanten unterschieden, a) deren Kosten noch nicht beobachtet wurden, b) deren Kosten zwar schon bekannt sind, aber noch nicht relevant waren und c) die bereits verwendet worden sind. Zudem wurden die Ergebnisse der Heuristik gecacht um eine wiederholte Berechnung zu vermeiden.

Die Verwendung der Heuristik und das späte Festlegen und Ausprobieren der Möglichkeiten für die Kostenbelegungen führte dazu, dass der Algorithmus auf den leichten Instanzen, deren kürzester Weg zum Zielknoten nicht blockiert sein konnte, fast die gleiche Ausführungszeit wie die optimistische Strategie erreichte. Auf Instanzen, bei denen alle Kanten (abgesehen von denen, die vom Startpunkt ausgehen) unsicher sind, skalierte der Algorithmus erwartungsgemäß sehr schlecht. So waren nur Instanzen innerhalb einer halben Stunde lösbar, die weniger als 15 Kanten hatten. Diese Grenze ist sehr scharf, vermutlich da kleinere Instanzen auch entsprechend wenige Knoten haben und dadurch der Zielknoten meist schon in zwei Schritten erreicht werden kann.

3.5.2. Die optimistische Strategie und die Greedy-Strategie

Da die optimistische Strategie nur ein Spezialfall der Greedy-Strategie ist, ist für beide zusammen nur ein Modul implementiert worden. Getestet wurden folgende Wertepaare für die Parameter a und b: $\langle 1, 0\rangle$, $\langle 1.1, 0\rangle$, $\langle 1.1, 1000\rangle$, $\langle 2, 1000\rangle$ und $\langle 100, 1000\rangle$. Für das Paar $\langle 1, 0\rangle$ entspricht das gerade dem optimistischen Algorithmus.

Hier hängen die Ergebnisse stark von der Rate der zufälligen Graphen ab. Beträgt sie $\frac{1}{2}$, beschränken sich die Kandidaten mit $b = 1000$ natürlich komplett auf die sicheren Wege und sind damit im Durchschnitt besser als die beiden anderen Varianten. Bei den Graphen mit höherer Rate verschiebt sich das Verhältnis sehr schnell dahingehend, dass die optimistischeren Varianten besser werden. Bei einer Rate von 1 sind die Varianten $(1, 0)$ und $(1.1, 0)$ in etwa gleich gut und 20% besser als die anderen. Bei einer Rate von 0.75 ist $(1.1, 0)$ ebenfalls zirka 20% besser als die anderen, aber 10% schlechter als die $(1, 0)$-Strategie.

Eine Abhängigkeit der Anzahl der Kanten war im Rahmen der Messungen nicht zu erkennen.

3.5.3. Die Monte-Carlo-Strategie

Getestet wurde die Monte-Carlo-Strategie mit 6 und 16 Samples. Es hat sich dabei gezeigt, dass die Qualität der gefundenen Strategien nicht von der Anzahl der Samples abhängt. Auch mit 16 Samples ist die Monte-Carlo-Strategie im Vergleich zu den Greedy-Strategien sehr schlecht. Für Graphen mit mehr als 1000 Knoten reichte eine Bearbeitungszeit von einer halben Stunde nicht aus. Somit wird hier nur über die kleineren Instanzen eine Aussage getroffen. Auf diesen Instanzen liefert die Monte-Carlo-Strategien durchschnittliche Kosten, die im Mittel etwas über 500-mal höher als bei der optimistische Strategie liegen. Wenn man nur die besten Ergebnisse der jeweils 10 Testläufe für die einzelnen Graphen vergleicht, sind die beiden Strategien bis auf wenige Ausreißer der Monte-Carlo-Strategie gleich gut.

Ohne die am Ende des Abschnitts 3.4 über die Monte-Carlo-Strategie vorgeschlagene Optimierung ist diese Strategie also nicht zu empfehlen.

4. Partially Observable Markov Decision Process

Ein Partially Observable Markov Decision Process (POMDP) ist ein sehr allgemeines Modell, um das Handeln eines Agenten unter Unsicherheit zu formalisieren.

In diesem Kapitel wird aufgezeigt, wie sich das CTP als POMDP darstellen lässt. Dann wird den entstandenen POMDP mit dem Programm `pomdp-solve` von Anthony R. Cassandra [3], das für ein POMDP eine optimale Strategie bestimmt, gelöst und die Effizienz mit der dedizierten Lösung verglichen.

Ein POMDP besteht aus einem Tupel $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O \rangle$. Dabei ist $\mathcal{S}$ eine endliche Menge von Weltzuständen und $\mathcal{A}$ eine endliche Menge von Aktionen, die dem Agenten zur Verfügung stehen. Zu jedem Zeitpunkt $t \in \mathbb{N}_0$ wählt ein Agent eine Aktion, die den Weltzustand verändert, dem Agenten einen bestimmten Nutzen bringt und ihm Informationen über den Weltzustand gibt.

Die Übergangsfunktion $T : \mathcal{S} \times \mathcal{A} \to \mathrm{dist}(\mathcal{S})$ gibt an, wie sich der Zustand der Welt durch eine Aktion des Agenten verändert. Ist zum Zeitpunkt t der Zustand gleich $s \in \mathcal{S}$ und wählt der Agent Aktion $a \in \mathcal{A}$ so ist der Weltzustand zum nächsten Zeitpunkt $t + 1$ mit Wahrscheinlichkeit $T(s, a)(s')$ gerade $s' \in \mathcal{S}$.

Für die Aktion $a \in \mathcal{A}$ im Zustand $s \in \mathcal{S}$ erhält der Agent sofort den Nutzen $R(s, a) \in \mathbb{R}$.

Nachdem der Agent eine Aktion ausgeführt hat, macht er eine Beobachtung $o \in \mathcal{O}$, wobei $\mathcal{O}$ die endliche Menge der möglichen Beobachtungen ist. Diese kann ihm Informationen über den aktuellen Weltzustand liefern. Die Beobachtungsfunktion $O : \mathcal{A} \times \mathcal{S} \to \mathrm{dist}(\mathcal{O})$ bestimmt dazu die Wahrscheinlichkeit $O(a, s)(o)$ für die Beobachtung $o \in \mathcal{O}$, wenn die Welt durch die Aktion a in den Zustand s überging. Es ist dabei möglich, dass der Agent aus der Beobachtung nicht auf den Nutzen schließen kann.

Der Agent hat eine eigene Vorstellung davon, in welchem Zustand sich seine Umgebung gerade befindet. Da seine Aktionen keine deterministische Wirkung haben, ist dieser so genannte Wissenszustand eine Wahrscheinlichkeitsverteilung über $\mathcal{S}$. Wenn der Agent zum Zeitpunkt t den Wissenszustand b_t hat, so geht er davon aus, dass er sich mit Wahrscheinlichkeit $b_t(s)$ im Zustand $s \in \mathcal{S}$ befindet.

Mit jeder Handlung des Agenten verändert sich der Zustand seiner Umgebung. Deswegen muss er nach jeder Aktion seinen Wissenszustand anpassen. Dieser lässt sich aus T, O, der zum Zeitpunkt t gewählten Aktion a, der darauf folgenden Beobachtung o und dem Wissenszustand b_t vor der letzten Aktion berechnen:

$$\begin{aligned}
b_{t+1}(s') &= P(s'|a,o,b_t) \\
&= P(s',o|a,b_t) \cdot (P(o|a,b_t))^{-1} \\
&= P(o|s',a,b_t) \cdot P(s'|a,b_t) \cdot (P(o|a,b_t))^{-1} \\
&= P(o|s',a) \cdot P(s'|a,b_t) \cdot (P(o|a,b_t))^{-1} \\
&= O(a,s')(o) \cdot \left(\sum_{\hat{s} \in \mathcal{S}} b_t(\hat{s}) \cdot T(\hat{s},a)(s') \right) \cdot (P(o|a,b_t))^{-1}
\end{aligned}$$

wobei

$$P(o|a,b_t) = \sum_{s' \in \mathcal{S}} O(a,s')(o) \sum_{\hat{s} \in \mathcal{S}} b_t(\hat{s}) \cdot T(\hat{s},a)(s')$$

Die Aufgabe, die der POMDP $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O \rangle$ stellt, ist es eine Strategie für den Agenten zu erarbeiten, die ihm möglichst viel erwarteten Nutzen bringt. Eine Strategie ist dabei eine Abbildung $p : \mathrm{dist}(S) \to \mathcal{A}$ – sie schreibt dem Agenten vor, dass er mit dem Wissenszustand b die Aktion $p(b)$ wählen soll.

Bleibt noch, die Bedeutung von „möglichst viel erwarteter Nutzen“ festzulegen. Dazu gibt es verschiedene Möglichkeiten, der Folge $(r_t)_{t \in \mathbb{N}_0}$ der erreichten Nutzenwerte einen Gesamtnutzen zuzuordnen. Die üblichen Methoden sind:

1. Summe der Nutzen $\sum r_t$
2. Endlicher Horizont für ein $n \in \mathbb{N}_0$: $\sum_{t=0}^{n} r_t$
3. abgezinste Summe für ein $\delta \in [0,1)$: $\sum \delta^t r_t$

Bei der ersten Methode ergibt sich die Schwierigkeit, dass bei viele Probleme die Summe für keine Strategie konvergiert. Um das endliche CTP zu formalisieren, ist diese Methode dennoch geeignet, da die sinnvollen Strategien auf endlichen Realisierungen immer nach einer endlichen Zahl an Schritten mit endlichen Kosten den Zielpunkt erreichen.

Ein POMDP besitzt die Markov-Eigenschaft, die besagt, dass das Ergebnis einer Handlung nur vom Zustand des aktuellen Zeitpunktes abhängt und nicht von der Folge der Aktionen und Zustände, die zu dem Zustand geführt haben.

Wenn für ein POMDP $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O \rangle$

$$\mathcal{O} = \mathcal{S}$$

und

$$\forall a \in \mathcal{A} : O(a,s) = \{s \mapsto 1\}$$

gilt, ist der Zustand für den Agenten vollständig beobachtbar. In diesem Fall gibt es keine Teilbeobachtungen und deswegen bezeichnet man $\mathcal{P}$ als Markov Decision Process (MDP). Das vereinfacht das Modell wesentlich, da sich die Menge der verschiedenen Wissenszustände des Agenten von $\mathrm{dist}(\mathcal{S})$ auf $\mathcal{S}$ verringert und damit endlich ist. Dementsprechend einfacher wird es auch, eine optimale Strategie für $\mathcal{P}$ zu berechnen. Da für einen MDP die beiden letzten Komponenten aus den anderen konstruierbar sind, ist es ausreichend, es als Viertupel $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ aufzufassen.

Man kann ein POMDP als MDP darstellen, wenn man zulässt, dass die Menge der Zustände unendlich ist. Als Zustandsmenge werden dann die Wahrscheinlichkeitsverteilungen über der Zustandsmenge des POMDP gewählt. Ein Zustand entspricht damit dem Wissensstand des Agenten. Die Übergangsfunktion berechnet für eine Aktion, wie oben vorgestellt, den neuen Wissenszustand und der Nutzen wird auf den erwartete Nutzen, ausgehend von den Möglichkeiten, die der Wissenszustand angibt, festgelegt.

4.1. Das endliche CTP als POMDP

Sei eine endliche Instanz $\langle V, E, c, s, t \rangle$ gegeben.

Das CTP lässt sich nun wie folgt als POMDP $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O \rangle$ formalisieren:

- Die Menge der Zustände ist $\mathcal{S} = V \times \bigotimes_{e \in E} \operatorname{supp} c_e$. Die erste Komponente gibt dabei an, in welchem Knoten des Graphen der Agent sich gerade befindet und die zweite besteht aus den tatsächlichen Kosten für die einzelnen Kanten.

- Die Menge der Aktionen ist gerade die Menge der Knoten: $\mathcal{A} = V$. Der Agent kann sich jeden Knoten als direktes nächstes Ziel vornehmen.

- Solange er noch nicht im Ziel angekommen ist, erreicht der Agent den gewünschten Knoten, wenn es eine Kante von seinem momentanen Standort dorthin gibt. Ist er schon im Ziel oder gibt es keine Kante, bleibt er an seinem Standort.

$$T(\langle v, \gamma \rangle, w) = \begin{cases} \{\langle w, \gamma \rangle \mapsto 1\} & \text{falls } \{v, w\} \in E \text{ und } v \neq t \\ \{\langle v, \gamma \rangle \mapsto 1\} & \text{sonst} \end{cases}$$

- Für das Passieren einer Kante muss der Agent deren Gewicht bezahlen. Deswegen ist der Nutzen einer Aktion (außer im Ziel) negativ.

$$R(\langle v, \gamma \rangle, w) = \begin{cases} -\gamma_{\{v,w\}} & \text{falls } \{v, w\} \in E \text{ und } v \neq t \\ 0 & \text{falls } v = t \\ -\infty & \text{sonst} \end{cases}$$

  Statt $-\infty$ könnte im dritten Fall auch ein anderer negativer Wert verwendet werden, weil ein Plan, der eine Aktion ohne Effekt aber mit negativem Nutzen enthält, bzgl. der Nutzensumme sicher nicht optimal ist.

- Für jeden Knoten besteht die Beobachtung, unabhängig von der ausgeführten Aktion, aus den Kosten der inzidierenden Kanten

$$\mathcal{O} := \bigcup_{v \in V} \bigotimes_{e \in \mathrm{edge}(v)} \operatorname{supp} c_e$$

- Wenn der Agent durch eine Aktion $w \in \mathcal{A}$ nach $\langle w', \gamma \rangle \in \mathcal{S}$ kommt, sieht er für alle von w' ausgehenden Kanten deren Kosten:

$$O(w, \langle w', \gamma \rangle) = \{(\gamma_e)_{e \in \mathrm{edge}(w')} \mapsto 1\}$$

Die durch c gegebenen Wahrscheinlichkeiten fließen nur in den anfänglichen Wissenszustand ein. Der Agent weiß, dass er sich zu Beginn in s befindet, kennt die Kosten $(\overline{\gamma}_e)_{e \in \mathrm{edge}(s)}$ der von s ausgehenden Kanten und hat durch c eine Vorstellung davon, welche Kosten für die übrigen Kanten in Frage kommen. Weil die Kosten der unbekannten Kanten unabhängig sind, ist der anfängliche Wissenszustand gerade

$$\left\{ \langle s, \gamma \rangle \mapsto \prod_{e \in E \setminus \mathrm{edge}(s)} c_e(\gamma_e) : (\gamma_e)_{e \in \mathrm{edge}(s)} = \overline{\gamma} \right\} \quad .$$

Für den Agenten ist es sinnvoll solange davon auszugehen, dass die tatsächliche Realisierung endlich ist, bis sich das Gegenteil herausstellt, denn sollte das Gegenteil der Fall sein, so sind durch die anfängliche falsche Annahme dennoch keine Kosten entstanden, da alle Aktionen dann den Nutzen 0 haben.

Dieser POMDP ist in einigen Punkten sehr speziell. Zum einen natürlich, weil nicht die volle Allgemeinheit, die ein Partially Observable Markov Decision Process bietet, ausgeschöft wird, da T und O deterministisch sind. Zum anderen verrät die Beobachtung nach jeder Aktion einiges über den erreichten Zustand. Das führt dazu, dass – unter der Annahme, dass die Realisierung endlich ist – der Agent vor jedem Schritt dessen Kosten kennt, weil er genug über den gegenwärtigen Zustand weiß. Ausserdem kann er auch die erste Komponente des Zustandes vorhersagen, welchen er mit einer Aktion erreichen wird.

Durch die deterministische Übergangsfunktion wird auch erreicht, dass keine Information verloren geht. Das heißt, wenn der Agent zum ersten Mal einen Knoten erreicht und ihm daraufhin die Längen der ausgehenden Kanten offengelegt werden, werden sich diese Informationen nicht ändern, wenn er ein weiteres Mal diesen Knoten erreicht. Das Canadian Traveller's Problem ist also in dem Sinne monoton, dass der Agent nach einer weiteren Aktion höchstens mehr über seine Umgebung weiß.

Außerdem gibt es keine Aktion, die das Erreichen des Zielpunkts unmöglich macht, da jede Handlung reversibel ist. So kann der Agent, wenn er von einem Zustand $\langle v, \gamma \rangle$ mit der Aktion w nach $\langle w, \gamma \rangle$ kommt, mit der Aktion v wieder in den ersten Zustand zurück. Die beiden einzigen Konsequenzen sind, dass er u. U. mehr weiß, weil er mehr Knoten besucht hat, und dass durch das zweimalige Passieren der Kante $\{v, w\}$ zusätzliche Kosten entstanden sind. Wenn sich der Agent entsprechend verhält, kann es ihm folglich in einer endlichen Realisierung nie passieren, dass er in eine Situation kommt, von der aus er nicht mehr oder nur mit unendlichen Kosten zum Ziel kommt. So ein Verhalten entspricht gerade einer sinnvollen Strategie nach Definition 2.11.

Da bei der Formalisierung als POMDP nicht ausgenutzt wird, dass T und O nichtdeterministisch sein können, wird die Berechnung des Wissenszustandes nach einer Aktion vereinfacht. Solange der Agent sein Ziel nicht erreicht hat und eine Aktion wählt, die einer Kante in E entspricht, gilt:

$$b_{t+1} = \left\{ \langle w, \gamma \rangle \mapsto \sum_{\hat{\gamma} \in \Gamma} b_t(\langle v, \hat{\gamma} \rangle) : (\gamma_e)_{e \in \mathrm{edge}(w)} = \overline{\gamma} \right\} \quad ,$$

wobei w die zum Zeitpunkt t vom Knoten v aus gewählte Aktion und $\langle w, \overline{\gamma} \rangle$ die daraus resultierende Beobachtung und

$$\Gamma = \{\gamma[\hat{\gamma}] : \hat{\gamma} \in \mathbb{K}^{\mathrm{edge}(w)}\}$$

ist.

4.2. Das endliche CTP als MDP

Tatsächlich ist das endliche CTP sogar als MDP formalisierbar. Die zweite Komponente besteht dann nicht mehr aus der vollständigen Realisierung der Instanz, sondern nur noch aus dem Wissen, das der Agent bisher gesammelt hat. Die Übergangsfunktion T muss entsprechend angepasst werden, damit sie auch die zweite Komponente gemäß des neu hinzukommenden Wissens anpasst. Die einzelnen Komponenten sehen dann wie folgt aus:

$$\begin{aligned} S &= V \times \bigotimes_{e \in E} (\operatorname{supp} c_e \cup \{\mathit{unknown}\}) \\ T(\langle v, \gamma \rangle, w) &= \bigcup_{\gamma' \in \kappa(w,\gamma)} \left\{ \langle w, \gamma[\gamma'] \rangle \mapsto \prod_{\gamma'_e \in \gamma'} c_e(\gamma'_e) \right\} \\ &\qquad \text{falls } \{v, w\} \in E \text{ und } v \neq t \end{aligned}$$

mit

$$\kappa(w, \gamma) = \bigotimes_{\substack{e \in \mathrm{edge}(w) \\ \gamma_e = \mathit{unknown}}} \operatorname{supp} c_e$$

$\kappa(w, \gamma)$ besteht aus den möglichen Belegungen der in γ noch unbekannten Kanten, die von w ausgehen. γ kann um jede dieser Möglichkeiten erweitert werden; die Wahrscheinlichkeit dafür ergibt sich als das Produkt der einzelnen Wahrscheinlichkeiten über die beteiligten Kanten.

$\mathcal{A}$ und R bleiben unverändert, und entsprechend der Definition für einen MDP ist $\mathcal{O} = \mathcal{S}$ und

$$O(a, \langle v, \gamma \rangle) = \{\langle v, \gamma \rangle \mapsto 1\} \quad .$$

4.3. Das allgemeine CTP als POMDP

Im allgemeinen CTP kann es Realisierungen geben, die nicht endlich sind. Diese wurden in den vorherigen Kapiteln nicht in der Bewertung des Algorithmus berücksichtigt. Damit das hier auch so ist, müssen dem Agenten in einem solchen Fall Kosten entstehen, die nicht von dem gewählten Pfad abhängen. Das lässt sich mit der Markov-Eigenschaft nur so vereinbaren, indem jeder Schritt die Kosten 0 verursachen muss. Da die Kantenkosten in der zweiten Komponente des Zustandes festgelegt sind, ist dieses mit der Formalisierung aus dem vorhergehenden Abschnitt möglich. Das gewählte Modell für das endliche CTP lässt sich also im Wesentlichen genauso auf den allgemeinen Fall anwenden; lediglich die Nutzenfunktion muss wie folgt angepasst werden:

$$R(\langle v, \gamma \rangle, w) = \begin{cases} -\gamma_{\{v,w\}} & \text{falls } \{v, w\} \in E,\ v \neq t \text{ und } \gamma \text{ endlich ist} \\ 0 & \text{falls } v = t \text{ oder } \gamma \text{ nicht endlich ist} \\ -\infty & \text{sonst} \end{cases}$$

Eine Veränderung, die sich dadurch ergibt, ist, dass sich die erwarteten Kosten für eine Strategie in diesem POMDP von dem bedingten Erwartungswert der Kosten für die entsprechende Strategie in I unterscheiden, allerdings num um den Faktor $P^I(\mathcal{R}_I^{<\infty})$. Beim Vergleich zweier Strategien bzgl. der kanonischen Strategieordnung spielt dieser Faktor jedoch keine Rolle, da er entweder positiv ist oder aber die erwarteten Kosten jeweils 0 sind. Um eine vollständige Übereinstimmung zu erreichen, können die von 0 verschiedenen Werte von R durch $P^I(\mathcal{R}_I^{<\infty})$ dividiert werden.

4.4. Das allgemeine CTP als MDP

Auch wenn sich das CTP nicht mit der kanonischen Umformung Umformung eines POMDP in einen MDP überführen lässt, so gibt es dennoch einen Weg, trotzdem einen Markov Decision Process zu konstruieren, der einer Instanz des CTP entspricht.

Im vorherigen Abschnitt wurde für die Konstruktion des POMDP ausgenutzt, dass der Agent den Weltzustand nicht vollständig feststellen kann und dadurch in unendlichen Realisierungen unbemerkt keine Kosten erzeugte.

In einem MDP gibt es die Möglichkeit nicht, dass der Nutzen einer Aktion dem Agenten verborgen bleibt, weil er zu jedem Zeitpunkt den vollständigen Weltzustand kennt und der Nutzen nur von diesem Zustand und der gewählten Aktion abhängt. In einer unendlichen Realisierung sollen dem Agenten aber unabhängig von seinen Aktionen keine unendlichen Kosten entstehen. Dies ist nur möglich, wenn die Kosten

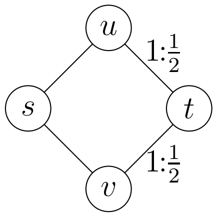

Abbildung 4.1.

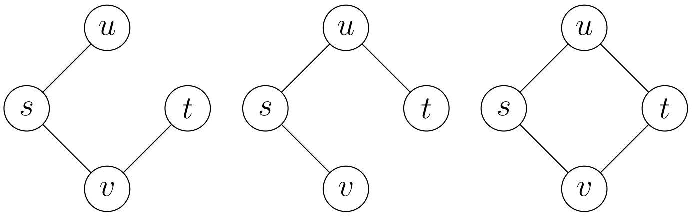

Abbildung 4.2.: Die endlichen Realisierungen von Abb. 4.1

jeder Aktion in solch einer unendlichen Realisierung Null sind. Diese beiden Voraussetzungen lassen sich nur vereinbaren, wenn man fordert, dass der Agent ausschließen kann, dass eine unendliche Realisierung vorliegt. Am folgenden Beispiel sieht man dann, dass die Kosten der Kanten nicht mehr unabhängig sind:

Die endlichen Realisierungen der Instanz aus Abbildung 4.1 sind in Abbildung 4.2 gezeigt. Weil in der Instanz für die unsicheren Kanten die Wahrscheinlichkeit für die Kosten 1 die gleichen sind wie für ∞, haben alle drei Realisierungen die Wahrscheinlichkeit $\frac{1}{3}$. Das heißt die bedingte Wahrscheinlichkeit für die Kosten 1 der Kante $\{u,t\}$ unter der Voraussetzung, dass die Realisierungen endlich ist, $\frac{2}{3}$ beträgt.

Bezeichne P_f dieses bedingte Wahrscheinlichkeitsmaß, dann gilt:

$$P_f\left[c_{\{u,t\}} = 1\right] \cdot P_f[c_{\{v,t\}} = 1] = \frac{4}{9} \neq \frac{1}{3} = P_f\left[c_{\{u,t\}} = 1 \wedge c_{\{v,t\}} = 1\right]$$

Das Problem erhält so einen unnatürlichen Charakter, da der Agent z. B. erfährt, dass die Kante $\{v,t\}$ die Kosten 1 hat, wenn er von s zu u geht und sieht, dass die Kante $\{u,t\}$ blockiert ist. Trotzdem ist es möglich, die Instanz als MDP mit einem endlichen Zustandsraum zu formalisieren:

$$\begin{aligned}
\mathcal{S} &= \{\langle s, \mathit{unknown}, \mathit{unknown}\rangle, \langle s, \mathit{unknown}, 1\rangle, \\
&\quad \langle s, 1, \mathit{unknown}\rangle, \langle s, 1, 1\rangle, \langle s, 1, \infty\rangle, \langle s, \infty, 1\rangle, \\
&\quad \langle u, 1, \mathit{unknown}\rangle, \langle u, 1, 1\rangle, \langle u, 1, \infty\rangle, \langle u, \infty, 1\rangle, \\
&\quad \langle v, \mathit{unknown}, 1\rangle, \langle v, 1, 1\rangle, \langle v, 1, \infty\rangle, \langle v, \infty, 1\rangle, t\} \\
\mathcal{A} &= \{s, u, v, t\} \\
T(\langle s, \mathit{unknown}, \mathit{unknown}\rangle, u) &= \left\{\langle u, 1, \mathit{unknown}\rangle \mapsto \frac{2}{3}, \langle u, \infty, 1\rangle \mapsto \frac{1}{3}\right\} \\
T(\langle s, \mathit{unknown}, 1\rangle, u) &= \left\{\langle u, 1, 1\rangle \mapsto \frac{1}{2}, \langle u, \infty, 1\rangle \mapsto \frac{1}{2}\right\} \\
T(\langle s, a, b\rangle, u) &= \{\langle u, a, b\rangle \mapsto 1\} \quad \text{für } a \neq \mathit{unknown} \\
T(\langle s, \mathit{unknown}, \mathit{unknown}\rangle, v) &= \left\{\langle v, \mathit{unknown}, 1\rangle \mapsto \frac{2}{3}, \langle v, 1, \infty\rangle \mapsto \frac{1}{3}\right\} \\
T(\langle s, 1, \mathit{unknown}\rangle, v) &= \left\{\langle v, 1, 1\rangle \mapsto \frac{1}{2}, \langle v, 1, \infty\rangle \mapsto \frac{1}{2}\right\} \\
T(\langle s, a, b\rangle, v) &= \{\langle v, a, b\rangle \mapsto 1\} \quad \text{für } b \neq \mathit{unknown} \\
T(\langle u, a, b\rangle, s) &= \{\langle s, a, b\rangle \mapsto 1\} \\
T(\langle u, a, b\rangle, t) &= \{t \mapsto 1\} \\
T(\langle v, a, b\rangle, s) &= \{\langle s, a, b\rangle \mapsto 1\} \\
T(\langle v, a, b\rangle, t) &= \{t \mapsto 1\} \\
T(z, a) &= \{z \mapsto 1\} \quad \text{sonst} \\
R(\langle u, a, b\rangle, t) &= -a \\
R(\langle v, a, b\rangle, t) &= -b \\
R(t, a) &= 0 \\
R(z, a) &= -1 \quad \text{sonst}
\end{aligned}$$

Diese Definition ist möglich, da der Agent in einem CTP stets genug Informationen hat, um die Kosten (bzw. den Nutzen) aller Aktionen zu kennen.

Dieser MDP ist in dem Sinne schwächer als der POMDP, als dass die Menge der Wissenszustände des Agenten von den Wahrscheinlichkeitsverteilungen über $V \times \bigotimes_{e \in E} \operatorname{supp} c_e$ auf diejenige Teilmenge von Zuständen eingeschränkt wird, die vom Startzustand erreichbar sind.

Analog lässt sich auch jedes andere CTP als Markov Decision Process formalisieren. Das ist möglich, weil es nur endlich viele verschiedene Wissenszustände gibt, die ausgehend vom Startpunkt tatsächlich erreichbar sind.

4.5. Lösung des CTP mit einem POMDP-Löser

Der POMDP-Löser `pomdp-solve`[3] von A. R. Cassandra bekommt ein POMDP als Eingabe und bestimmt dann mit einer Verallgemeinerung des value iteration-Verfahrens für MDPs eine optimale Strategie für alle möglichen Wissenszustände. Eine solche Strategie lässt sich auf endliche Weise durch einen Strategie-Graphen darstellen[4].

Da es nicht möglich ist, einen Startzustand in der Eingabe des Programms zu definieren, wird mehr berechnet, als notwendig ist, um eine Instanz des CTP optimal zu lösen. Das führt natürlich auch dazu, dass `pomdp-solve` Rechenzeit für die Bestimmung dieser Daten verbraucht, die ein speziell auf das Canadian Traveller's Problem zugeschnittenes Programm nicht benötigt. Bei Instanzen wie in Abb. 2.1 ist dieser Unterschied sehr deutlich zu merken. So brauchte `pomdp-solve` auf dem Testrechner über 5 Minuten, um diese Instanz zu bearbeiten. Das spezialisierte Programm hingegen erkennt sofort, dass der kürzest mögliche Weg sicher ist und benötigt unter 0.2 Sekunden.

Die POMDPs, die Cassandras Programm als Eingabe bekommt, sind so verkleinert, dass nur Tupel $\langle v, \gamma \rangle$ als Zustände spezifiziert werden, die vom Startzustand auch tatsächlich erreichbar sein können.

Für `pomdp-solve`, wie auch für die Implementierung der optimalen Strategie, ist entscheidend, wieviele unsichere Kanten es in der betrachteten Instanz gibt. In den zufällig generierten Graphen kommt das Programm nur bei Graphen, die acht oder weniger unsichere Kanten haben, unter einer halben Stunde zu einer Lösung.

Eine weitere Erklärung dafür, dass `pomdp-solve` bei weitem nicht mit der speziellen Lösung konkurieren kann, ist, dass der so genannte Witness-Algorithmus[4] von seiner Struktur her die spezielle Form der untersuchten POMDPs nicht ausnutzt.

Mit zunehmender Komplexität wird auch das Parsen der Eingabedatei immer aufwendiger. So ist die Beschreibungsdatei für einen zufällig generierten Graphen mit neun Knoten, elf sicheren und vierzehn unsicheren Kanten schon 286 Megabytes (MBs) groß. Der zugehörige POMDP hat 131074 verschiedene Zustände. Der dedizierte Algorithmus ist in der Lage, dieses Problem in unter 0.5 Sekunden zu lösen.

5. Varianten und Spezialfälle des CTP

In diesem Abschnitt werden ein paar Spezialfälle und Varianten des Canadian Traveller's Problem vorgestellt. Einige davon sind im Sinn von Definition 5.4 äquivalent zur allgemeinen Definition.

Um formal über die unterschiedlichen Varianten sprechen zu können, wird zunächst ein Problembegriff benötigt. Es gibt dafür verschiedene Möglichkeiten; hier bietet es sich an, eine Definition zu verwenden, die dem POMDP sehr ähnlich ist.

Definition 5.1 (Planungsproblem) Ein Achttupel $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O, s, g \rangle$ heißt *Planungsproblem*, wenn die ersten 6 Komponenten einen POMDP bilden, $s \in \mathrm{dist}(\mathcal{S})$ und $g \in \mathcal{S}$ ist. s heißt das Startwissen, g das Ziel des Planungsproblems.

Hier wird, wie im vorherigen Kapitel auch, nicht die volle Allgemeinheit des Modells verwendet. Alle vorgestellten Probleme lassen sich auch hier mit einer deterministischen Übergangsfunktion und deterministischen Beobachtungen formalisieren.

Die Lösung für ein Planungsproblem $\mathcal{P}$ besteht aus einem Plan, der jedem Wissenszustand eine Aktion zuordnet. Jeder Lösung p kann ein Nutzen zugeordnet wird. Dieser Nutzen ist der Erwartungswert der Summe der Einzelnutzen bis zum Erreichen des Zustandes g und werde mit $\mathrm{cost}_p(\mathcal{P})$ bezeichnet.

Definition 5.2 (Reduzierbarkeit für Planungsproblemen) Ein Planungsproblem $\mathcal{P}_1$ heißt auf ein Planungsproblem $\mathcal{P}_2$ *reduzierbar* ($\mathcal{P}_1 \lesssim \mathcal{P}_2$), wenn es eine in polynomieller Zeit berechenbare Transformation f der Lösungen von $\mathcal{P}_2$ zu Lösungen von $\mathcal{P}_1$ gibt, so dass für alle Lösungen p von $\mathcal{P}_2$ sowie alle optimalen Lösungen p_1 von $\mathcal{P}_1$ und p_2 von $\mathcal{P}_2$ gilt:

$$\frac{\mathrm{cost}_p(\mathcal{P}_2)}{\mathrm{cost}_{p_2}(\mathcal{P}_2)} \geq \frac{\mathrm{cost}_{f(p)}(\mathcal{P}_1)}{\mathrm{cost}_{p_1}(\mathcal{P}_1)}$$

Wenn man also einen „guten“, polynomiellen Lösungsalgorithmus für $\mathcal{P}_2$ kennt, kann man mit der Transformation eine mindestens genauso gute, ebenfalls polynomielle Lösung für $\mathcal{P}_1$ konstruieren.

Als Spezialfall der Definition ergibt sich, dass die Transformation den optimalen Algorithmus für $\mathcal{P}_2$ auf den optimalen für $\mathcal{P}_1$ abbildet.

Definition 5.3 (äquivalente Planungsprobleme) Seien $\mathcal{P}_1$ und $\mathcal{P}_2$ Planungsprobleme. Sie werden *äquivalent* genannt ($\mathcal{P}_1 \simeq \mathcal{P}_2$), wenn sowohl $\mathcal{P}_1 \lesssim \mathcal{P}_2$ als auch $\mathcal{P}_2 \lesssim \mathcal{P}_1$ gilt.

Im Folgenden sollen nicht nur zwei explizite Probleme, sondern ganze Problemklassen verglichen werden. Die nächste Definition überträgt den Begriff der Reduzierbarkeit und damit auch den der Äquivalenz auf Problemklassen.

Definition 5.4 (Reduzierbarkeit für Problemklassen) Eine Klasse von Planungsproblemen $\mathfrak{P}_1$ heißt auf eine zweite Klasse $\mathfrak{P}_2$ *reduzierbar* ($\mathfrak{P}_1 \lesssim \mathfrak{P}_2$), wenn folgende zwei Funktionen existieren:

- eine polynomielle Transformation g, die jedem Problem aus $\mathfrak{P}_1$ ein Problem aus $\mathfrak{P}_2$ zuordnet und

- eine polynomielle Funktion f, so dass für alle $\mathcal{P} \in \mathfrak{P}_1$ $f(\mathcal{P}, \cdot)$ eine Funktion ist, mit der sich $\mathcal{P}$ auf $g(\mathcal{P})$ reduzieren lässt.

Analog zur Äquivalenz für Probleme, heißen zwei Problemklassen äquivalent, wenn beide auf die jeweils andere reduziert werden kann.

Auch bei Problemklassen bedeutet Reduzierbarkeit intuitiv, dass man mit einer Lösungsstrategie für die Probleme aus der einen Klasse, die der anderen lösen kann. Um ein Problem $\mathcal{P}$ aus $\mathfrak{P}_1$ zu lösen, wenn man für $\mathfrak{P}_2$ ein Lösungsschema hat, erstellt man einen Plan für $g(\mathcal{P})$ und transformiert diesen mit $f(\mathcal{P}, \cdot)$ in einen Plan für $\mathcal{P}$.

Trivialerweise gilt mit $\mathfrak{P}_1 \subseteq \mathfrak{P}_2$ auch $\mathfrak{P}_1 \lesssim \mathfrak{P}_2$. g und f sind dann einfach die Identität bzw. die Projektion auf das zweite Argument. Für Spezialfälle des CTP muss also nur noch gezeigt werden, dass man das allgemeine CTP auf den Spezialfall reduzieren kann um die Äquivalenz der Klassen nachzuweisen.

Die Reduktionen werden im folgenden nicht formal ausgeführt, sondern nur mit Worten begründet. Die Beschreibungen sind aber exakt genug, um daraus die Reduktionen formal konstruieren zu können.

5.1. Zweiwertiges CTP

Beim zweiwertigen CTP werden die Kostenfunktionen soweit eingeschränkt, dass der Träger jeder Funktion c_e nur maximal zweielementig ist.

Wenn man auch Instanzen zulässt, bei denen es mehr als eine Kante zwischen zwei Punkten geben darf, ist das Problem äquivalent zur ursprünglichen Definition.

Um die Äquivalenz zwischen dem zweiwertigen CTP und dem allgemeinen zu zeigen, wird eine Bijektion zwischen den beiden Klassen konstruiert, so dass das Bild einer Instanz des allgemeinen CTP eine zweiwertige Instanz ist, auf die sich die Pläne identisch übertragen lassen und dabei genau die gleichen Kosten erzeugen.

In einem Graphen, der beliebige diskrete Verteilungen für die Kantenkosten verwendet, wird jede Kante so durch andere Kanten mit Kostenfunktionen der einfacheren Form ersetzt, dass alle Abstände zwischen je zwei Punkten die gleiche Wahrscheinlichkeit behalten.

Um eine Strategie auf dieses Modell zu übertragen, wird in dem Fall, dass es mehrere Kanten von dem betrachteten Knoten zu einem anderen gibt, stets die kürzeste Kante gewählt.

Es ist dann klar, dass aus der Existenz dieser Bijektion die Äquivalenz der beiden Klassen folgt.

Die Umformung eines CTP in ein zweiwertiges CTP funktioniert wie folgt: Für eine Kante $e = \{v, v'\}$ ist der Träger von c_e nach Definition endlich. Gelte etwa $\operatorname{fsupp} c_e = \{c_1, c_2, \dots c_k\}$ und o. B. d A. $c_i < c_{i+1}$ für $i \in \{1, 2, \dots, k-1\}$.

Wenn $k < 2$ ist, hat die Kostenfunktion für die betrachtete Kante schon die gewünschte Form. Die im weiteren beschriebene Ersetzung ist also nur für $k \geq 2$ notwendig. Die Kante e kann man nun durch k neue Kanten $\{e_1, \dots e_k\}$ zwischen v und v' ersetzen mit

$$\begin{aligned} c_{e_i}(c_i) &= \frac{c_e(c_i)}{1 - \sum_{j=1}^{i-1} c_e(c_j)} \\ c_{e_i}(\infty) &= 1 - c_{e_1}(c_i) \\ c_{e_i}(x) &= 0 \quad \text{für } x \neq c_i \text{ und } x \neq \infty \end{aligned}$$

Dann ist die Wahrscheinlichkeit, dass die kürzeste direkte Verbindung zwischen v und v' über eine der neuen Kanten e_j geht und das Gewicht c_i hat, die Wahrscheinlichkeit dafür, dass die Kanten mit einem Index kleiner i die Kosten ∞ haben und die Kante mit Index i die Kosten c_i. Für diese Wahrscheinlichkeit gilt:

$$\begin{aligned} & \prod_{j=1}^{i-1} c_{e_j}(\infty) \cdot c_{e_i}(c_i) \\ = \; & \prod_{j=1}^{i-1} (1 - c_{e_j}(c_j)) \cdot c_{e_i}(c_i) \\ = \; & \prod_{j=1}^{i-1} \left(1 - \frac{c_e(c_j)}{1 - \sum_{j'=1}^{j-1} c_e(c_{j'})} \right) \cdot c_{e_i}(c_i) \\ = \; & \prod_{j=1}^{i-1} \frac{\left(1 - \sum_{j'=1}^{j-1} c_e(c_{j'})\right) - c_e(c_j)}{1 - \sum_{j'=1}^{j-1} c_e(c_{j'})} \cdot c_{e_i}(c_i) \\ = \; & \prod_{j=1}^{i-1} \frac{1 - \sum_{j'=1}^{j} c_e(c_{j'})}{1 - \sum_{j'=1}^{j-1} c_e(c_{j'})} \cdot \frac{c_e(c_i)}{1 - \sum_{j=1}^{i-1} c_e(c_j)} \\ = \; & \left(1 - \sum_{j'=1}^{i-1} c_e(c_{j'}) \right) \cdot \frac{c_e(c_i)}{1 - \sum_{j=1}^{i-1} c_e(c_j)} \\ = \; & c_e(c_i) \quad . \end{aligned}$$

Genau dann wenn alle neuen Kanten das Gewicht ∞ haben, hat auch der kürzeste Weg das Gewicht ∞. Die Wahrscheinlichkeit dafür erhält man durch eine ähnliche

Rechnung:

$$\begin{aligned}
& \prod_{j=1}^{k} c_{e_j}(\infty) \\
= \; & \prod_{j=1}^{k} \frac{1-\sum_{j'=1}^{j} c_e(c_{j'})}{1-\sum_{j'=1}^{j-1} c_e(c_{j'})} \\
= \; & 1 - \sum_{j'=1}^{k} c_e(c_{j'}) \\
= \; & c_e(\infty)
\end{aligned}$$

Für alle anderen Werte ist die Wahrscheinlichkeit immer 0, weil ein solcher Wert weder im Träger von c_e noch in einem der Träger c_{e_i} für $i \in \{1, 2, \ldots k\}$ enthalten ist.

Es entsteht also mit den k neuen Kanten ein Ersatz für die Kante e, der zu genau den gleichen Kosten mit gleichen Wahrscheinlichkeiten führt.

So kann man jede Kante einer Instanz des ursprünglichen Modells ersetzen und erhält einen Graphen für das zweiwertige CTP, welches das gleiche Problem repräsentiert.

Die so konstruierten zweiwertigen Probleme haben sogar die Eigenschaft, dass die neu eingeführten Kanten nur Längen der Größe einer einzigen endlichen Konstanten und ∞ annehmen können. Wenn man also auch Kanten, für die die Kostenfunktion zwei endliche Werte im Träger enthält, analog zum obigen Verfahren durch zwei Kanten ersetzt, erhält man ein äquivalentes Problem, in dem für alle Kanten e gilt: $|\text{fsupp}\, c_e| \leq 1$. Aus dem gleichen Grund ist auch dieses speziellere zweiwertige CTP äquivalent zu der ursprünglichen Definition.

5.2. CTP mit endlichen Kosten

Eine weitere alternative Formulierung des CTP ist, dass man den Wert ∞ für die Kostenfunktionen nicht zulässt. Das hat den Vorteil, effektiv mit dem Erwartungswert der Kantengewichte gerechnet werden kann, um z. B. eine Strategie zu entwerfen. Für die Probleminstanzen des CTP, für die es von jedem Knoten aus sicher einen Weg endlichen Gewichts zum Ziel gibt, lässt sich leicht ein äquivalentes Modell dieser Form angeben.

Sei dazu $\hat{c}_e = \max \operatorname{supp} c_e$. Nach Vorraussetzung ist der Graph, der nur die Kanten e enthält mit $\hat{c}_e < \infty$ zusammenhängend. Dann lässt sich mit dem Algorithmus von Dijkstra bzgl. des Gewichts $\hat{c}$ für jeden Knoten $v \in V$ die kürzeste sichere Distanz $d_{\hat{c}}(v, w) \in \mathbb{R}^+$ zum Knoten $w \in V$ berechnen. Wenn man nun die ursprüngliche Kostenfunktion durch $c' : E \to \text{dist}(\mathbb{R}^+)$ mit

$$c'(\{v, w\}) \;=\; c_{\{v,w\}}[\infty \to d_{\hat{c}}(v, w)]$$

ersetzt, erhält man ein äquivalentes Problem ohne unendliche Wegkosten. Die Äquivalenz in die eine Richtung ist klar, weil das endliche CTP ein Spezialfall ist. Die Umkehrrichtung ergibt sich, indem man in einer Strategie auf dem endlichen Graphen diejenigen Schritte, die eine Kante des Gewichts $d_{\hat{c}}$ verwenden, durch die Aktionen ersetzen, die den Agenten auf den Kanten e mit $\hat{c}_e < \infty$ zum Endpunkt bringen. Das führt höchstens zu weniger Kosten, da auf dem Weg u. U. einige Kanten billiger sind, als von $\hat{c}$ angegeben. Um die Strategie anschließend konsistent weiterführen zu können, muss der Agent alle Informationen, die er auf dem Ersatzweg gesammelt hat, wieder „vergessen".

Aber auch im allgemeineren Fall, wenn nur ein sicherer Weg zwischen s und t existiert, lässt sich ein äquivalentes Problem in polynomieller Zeit finden Die Idee dabei ist, eine Kante gesetzt den Fall, dass sie in dem in der zu Grunde liegenden Instanz das Gewicht ∞ hat, nach der Modifikation so teuer zu machen, dass der Agent, anstatt die Kante zu verwenden, besser (oder genauso gut) einen anderen Weg verwenden kann.

Sei dazu κ eine Zahl, die so groß ist, dass es sich nicht lohnt, eine Kante mit Wegkosten κ zu traversieren, weil es dann besser wäre, den sicheren Weg zu t zu wählen.

Der ursprüngliche Graph mit den Kosten $c' : e \mapsto c_e[\infty \to \kappa]$ ist dann ein endliches CTP. Um eine Strategie auf diesem veränderten Graphen auf eine Strategie des ursprünglichen Graphen abzubilden, wird die Strategie einfach identisch übernommen, bis sie eine Kante traversiert, die Kosten κ oder mehr hat. In dem Fall wird im weiteren Verlauf ein sicher endlicher Weg zu t gewählt, der nach Wahl von κ nicht teurer als diese Kante sein kann. Die Existenz eines solchen endlichen Weges zu t ist gesichert, da ein endlicher Weg von s zum aktuellen Standpunkt v und ein weiterer endlicher Pfad von s zu t existiert. Damit existiert auch ein Weg von v nach t.

Es bleibt noch das Problem, ein geeignetes κ zu finden. Ein sicherer Wert ist

$$\sum_{e \in E} \max \operatorname{fsupp} c_e \quad ,$$

weil auf dem bekannten, endlichen Weg von dem Punkt, an dem man die Originalstrategie nicht weiter verfolgt, zum Ziel jede Kante maximal einmal verwendet wird.

5.3. Gerichtetes CTP

Wenn man dem CTP einen gerichteten Graphen zugrunde legt, entsteht ein Problem, das ganz andere Eigenschaften hat als die ursprüngliche Formulierung. So gilt hier etwa nicht mehr, dass alle Aktionen reversibel sind. Betrachtet man z. B. das durch Abbildung 5.1 gegebene gerichtete CTP, so kann ein Agent sehr wohl in eine Sackgasse geraten. Wird nämlich die Kante $\{s, v\}$ gewählt, gibt es von v aus keinen weiterführenden Weg mehr.

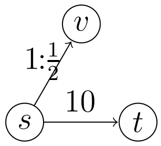

Abbildung 5.1.: gerichtetes CTP

Obwohl man mit einem gerichteten Graphen einen ungerichteten simulieren kann, indem man jede Kante durch zwei unterschiedlich gerichtete Kanten mit den gleichen Endpunkten ersetzt, ist hier eine Reduktion nicht ersichtlich. Denn auch wenn in Abbildung 5.1 zusätzlich die Kanten (v, s) und (t, s) – mit der gleichen Gewichtsfunktion wie die entgegengesetzte Kante – eingefügt werden, ist es trotzdem noch möglich, dass ein Agent zwar v mit Kosten 1 erreicht, aber dennoch nicht mehr zu s zurück kommt ohne eine unendliche Kante zu passieren.

6. Zusammenfassung und Ausblick

Die Definition der Strategieordnungen bildet die Grundlage für eine weitere Analyse von Approximationsalgorithmen für das Canadian Traveller's Problem. Die vorgeschlagene Optimierung der Monte-Carlo-Strategie ist ein vielversprechender Ansatz, um vielleicht noch einen leistungsfähigeren Näherungsalgorithmus als die vorgestellten zu entwickeln. Es bleibt zu untersuchen, wie sehr sich der Algorithmus dadurch tatsächlich verbessert. Auch ist – abgesehen von der optimalen Strategie – keiner der präsentierten Algorithmen approximativ. Im Anschluß an die Untersuchungen dieser Arbeit, kann die Frage gestellt werden, ob überhaupt ein effizient berechenbarer und approximativer Algorithmus existiert.

Die Diskussion in Kapitel 4 hat gezeigt, dass im CTP Unsicherheiten enthalten sind, diese aber eine sehr spezielle Form haben. Diese lässt sich bei der Konstruktion der exakten sowie einer Näherungslösung gut ausnutzen, so dass in kürzerer Zeit ein Ergebnis gefunden wird, als es z. B. mit dem sehr viel allgemeineren Ansatz der Lösung eines POMDP möglich ist.

Die Äquivalenzaussagen für die Varianten des CTP erlauben, durch die Untersuchung dieser Varianten, Rückschlüsse auf das allgemeine CTP zu ziehen. Vor allem das zweiwertige CTP hat den Vorteil, dass sich die Kantenkosten für jede Kante durch drei (bzw. nur zwei) Werte statt durch eine beliebige endliche Anzahl Parameter darstellen lassen.

A. Der Dijkstra-Algorithmus

Für einen ungerichteten, zusammenhängenden Graphen $\langle V, E \rangle$ mit einer Kantengewichtung $c : E \rightarrow \overline{\mathbb{R}}$ lässt sich in polynomieller Zeit ein kürzester, endlicher Pfad zwischen zwei Knoten s und t bestimmen, falls ein solcher Pfad existiert.

```
def min_dist(s, t, V, E, c):
  queue = PriorityQueue()
  for v in V:
    queue.insert(value=v, priority=infinity)

  queue.set_priority(value=t, priority=0)

  while not queue.is_empty():
    # pop gibt immer das Element mit der kleinsten
    # Prioritaet zurueck und entfernt es aus der
    # Warteschlange
    v, dist = queue.pop()

    if dist == infinity:
      raise Exception, "the graph is not connected"
    if v == s:
      return dist

    for w in neigh[v]:
      if w != None and w in queue:
        weight = c(e)
        if queue.priority_of(w) > dist + weight)
          queue.set_priority(value=w, priority=dist + weight)
```

Eine Invariante der while-Schleife ist, dass die Priorität der Knoten in der Schlange immer größer oder gleich der Distanz zu t ist, wobei für den Knoten minimaler Priorität Gleichheit gilt. Hierbei ist vorausgesetzt, dass die Kanten ausschliellich positive Kosten haben und somit auf einem Pfad mit minimalen Kosten von einem Knoten v aus, der Nachfolger v' eine kürzere Distanz zu t hat und deshalb vor v aus der Warteschlange entfernt wird.

Der Algorithmus berechnet somit in aufsteigender Reihenfolge die kürzesten Distanzen von jedem Knoten in V zu t. Sobald eine Distanz zwischen s und t gefunden wird, wird sie zurückgegeben.

Jeder Knoten wird nur einmal in die Warteschlange eingefügt, weshalb die Abbruchbedingung der while-Schleife nach spätestens $|V|$ Iterationen erfüllt ist. Die for-Schleife wird für jede Kante maximal zweimal ausgeführt. Für eine geeignete Implementation der Warteschlange (z. B. als Fibonacci-Heap) ergibt sich damit eine maximale Gesamtlaufzeit von $O(|V| \log |V| + |E|)$.

Index

Literaturverzeichnis

[1] http://www.rescuesystem.org/robocuprescue/.

[2] http://www.algorithmic-solutions.com/enleda.htm, Version 5.0.

[3] Cassandra, Anthony R. http://www.pomdp.org/pomdp/code/, Version 5.2.3.

[4] Cassandra, Anthony R., Leslie P. Kaelbling und Michael L. Littman: *Acting Optimally in Partially Observable Stochastic Domains*. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Band 2, Seiten 1023–1028, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[5] Nebel, Bernhard: *On the Compilability and Expressive Power of Propositional Planning Formalisms*. Journal of Artificial Intelligence Research, 12:271–315, 2000.

[6] Papadimitriou, Christos H. und Mihalis Yannakakis: *Shortest paths without a map*. In: *Proceedings of Theoretical Computer Science*, 84, Seiten 127–150, 1991.