Der Pledge-Algorithmus

Benjamin Kahl

22. Juni 2017

Inhaltsverzeichnis

1 Einführung

1.1 Modelldefinition 2

2 Naiver Ansatz

2.1 Zufällige Wegwahl 3

2.2 Rechte-Hand-Methode 3

3 Pledge-Algorithmus

3.1 Pseudocode 4

3.2 Beispiele 4

3.3 Korrektheitsbeweis 5

3.4 Fazit 7

4 Literatur

1 Einführung

Lösungsalgorithmen für Irrgärten stellen Methoden dar, die automatisiert einen Ausweg aus einem Irrgarten finden. Sie unterscheiden sich zu Algorithmen aus der Graphentheorie dabei, dass die erforderliche Information zum Finden der Lösung nicht vollständig vorhanden ist oder abgespeichert wird. Ein mobiler Agent der nur die direkt umliegende Umgebung wahrnehmen kann, soll zu einem Ziel gesteuert werden, ohne dass die Position des Zieles a-priori bekannt ist.
Der kürzeste Weg kann nicht, wie in der Graphentheorie, *berechnet* werden, sondern muss stattdessen *gefunden* werden.
Die Qualität eines Lösungsalgorithmus ist im Allgemeinem von drei Faktoren abhängig:

- Erfolgsrate: Ob eine Lösung immer gefunden wird wenn sie existiert.
- Weglänge: Die Länge des Weges den der Algorithmus gefunden hat.

Ein solcher Algorithmus, der Pledge-Algorithmus, steht in dieser Arbeit und dem zugehörigen Vortrag im Zentrum.

1.1 Modelldefinitionen

Zunächst definieren wir die erforderlichen Elemente um das Problem darzustellen:

Definition 1.1. *Wand / Hindernis*
Wände bilden die Hindernisse des Irrgartens. Sie bestehen aus einer endlichen Menge von polygonalen Ketten $p_0, \dots, p_n$ (verbundene Liniensegmente) die sich nicht schneiden.

Definition 1.2. *Irrgarten*
Ein Irrgarten besteht aus einer endlichen Menge Wände die in einer Ebene liegen.
Die Fläche des kleinstmöglichen Rechtecks, dass alle Wände des Irrgartens enthält ist das Irrgarten-Innere. Ein Punkt der außerhalb dieses liegt hat, sozusagen, den Irrgarten verlassen.

Definition 1.3. *perfekter Irrgarten*
Ein Irrgarten wird als perfekt (oder zusammenhängend) bezeichnet, wenn man in diesem nicht im Kreis gehen kann. Solch ein Irrgarten ist äquivalent zu einem Baum in der Graphentheorie.

Definition 1.4. C_{frei}
C_{frei} ist die Menge aller Punkte in der Ebene die aus dem Irrgarten-Äußerem erreichbar sind ohne eine Wand zu durchqueren.

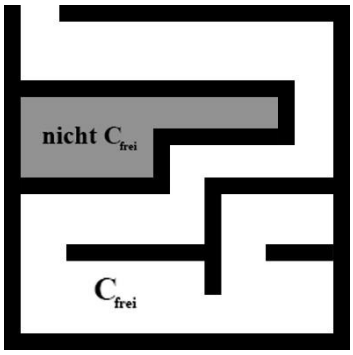

Abb. 1.5. C_{frei} in einem perfekten Irrgarten

Es gibt unterschiedliche Varianten an Navigationsproblemen die von der Art des Irrgartens und der Ausstattung des Agenten abhängen.
In unserem Fall, suchen wir nach einem Algorithmus der einem punktförmigen Roboter den Ausgang aus einem allgemeinem Irrgarten findet.
Der Agent (fortan auch *Roboter* genannt) verfügt über einen Tastsensor und einem Winkelzähler.
Drehungen nach rechts werden vom Winkelzähler als negative Drehungen wahrgenommen, Drehungen nach links als positive.
Der Roboter ist mit folgenden rudimentären Funktionen ausgestattet:

- Vorwärts bewegen
- Drehen
- Eine Wand direkt vor sich zu erfassen
- Sich an einem Hindernis entlangtasten

Mit den gegebenen Funktionen und Hilfsmitteln kann der Roboter nicht autonom prüfen, ob er den Irrgarten verlassen hat. Deshalb gehen wir des Weiteren davon aus, dass der Roboter von außen ein Signal erhält sobald er sich außerhalb des Irrgartens befindet.

2 Naiver Ansatz

2.1 Zufällige Wegwahl

Bei dem Algorithmus der zufälligen Wegwahl handelt es sich um eine äußerst triviale Methode den Ausgang aus einem Irrgarten zu finden.
Wie dem Namen zu entnehmen ist, besteht der Algorithmus darin, sich solange geradeaus zu bewegen, bis ein Hindernis erreicht wird. Dort entscheidet man sich zufällig für eine Richtung.
Mit dieser Methode wir der Ausgang zwar immer erreicht, jedoch besteht die Möglichkeit, dass derselbe Pfad mehrmals beschreitet wird.
Die Erfolgsrate ist somit maximal, jedoch kann keine obere Schranke für Weglänge oder Berechnungszeit bestimmt werden.

2.2 Rechte-Hand-Methode

Die Rechte-Hand-Methode ist der bekannteste Lösungsalgorithmus für Irrgärten und garantiert auf deterministische Weise in endlicher Zeit den Ausgang zu einem beliebigen, perfekten Irrgarten zu finden.
Hierbei wird der Roboter nach vorne bewegt, bis dieser auf eine Wand stößt. Dann legt der Roboter sozusagen „die rechte Hand auf die Wand“ und tastet sich an dieser entlang bis der Ausgang gefunden wird.
Besteht der Irrgarten aus nur einem zusammenhängenden, polygonalen Hindernis, ist die Methode garantiert erfolgreich.

Die Erfolgsrate wird dadurch garantiert, dass bei einem perfekten Irrgarten die Segmente der Wände zu einer zusammenhängenden, geraden Kette verformt werden können.
Bei Anwendung der Rechten-Hand-Methode tastet sich der Roboter so lange an der Kette lang, bis er das Ende dieser erreicht, welches dem Ausgang entspricht.
Es folgt, dass die resultierende Weglänge bei dieser Methode im *worst case* die summierte Länge aller Segmente des Irrgartens entspricht.

Sobald ein nicht perfekter Irrgarten betrachtet wird, ist es möglich, dass der Algorithmus scheitert.
Das Beispiel in Abb. 2.1.(rechts) zeigt wie der Roboter sich bei Anwendung der Rechten-Hand-Methode ewig im Kreis bewegt, wenn er auf einem alleinstehenden Hindernis stößt.
Der Pledge-Algorithmus erweitert diese Methode auf beliebige Irrgärten.

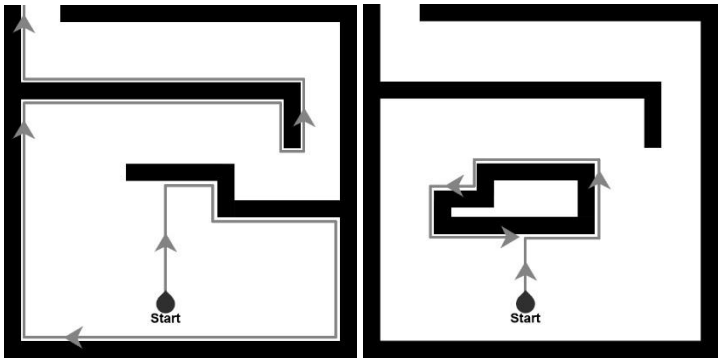

Abb. 2.1. Rechte-Hand-Methode bei einem perfekten Irrgarten (links) und einem nicht perfekten Irrgarten (rechts)

3 Pledge-Algorithmus

Der Pledge-Algorithmus, konzipiert von dem damals 12-jährigem John Pledge, benötigt eine zufällig gewählte Zielrichtung in die sich der Roboter beim Start des Algorithmus bewegt. Stößt er auf ein Hindernis, dreht er sich zunächst nach rechts und beginnt dieses zu umrunden. Der Winkelzähler verfolgt hierbei die Drehungen die durchführt werden. Wobei Drehungen nach rechts als negativ zählen und Drehungen nach links als positiv. Sobald der Winkelzählerwert wieder gleich Null ist, löst sich der Roboter wieder vom Hindernis.
Auf dieser Weise begibt sich der Roboter in eine bestimmte Richtung und umrundet alle alleinstehenden Hindernisse die dabei im Weg stehen, bis die „äußerste“ Wand erreicht wird (die Wand die mit dem Ausgang verbunden ist). An dieser kann sich der Roboter wie bei der Rechten-Hand-Methode zum Ausgang entlangtasten.

Es reicht hierbei nicht sich von einem Hindernis zu lösen sobald der Roboter sich in seiner anfänglichen Ausrichtung befindet, weil Fälle wie in Abb. 3.1. (Mitte) ansonsten versagen würden.

3.1 Pseudocode

```
while(Ausgang nicht erreicht)                                  (1)
      if(Wandkontakt und Winkelzähler != 0)                    (2)
            folge Wand;                                        (3)

      else                                                     (4)
            Geradeaus bewegen;                                 (5)
```

3.2 Beispiele

Bei Irrgärten in denen alle Wände zueinander parallel oder orthogonal stehen, genügt es die Anzahl Umdrehungen mitzuzählen. Ansonsten muss der genaue Winkel der Umdrehungen berücksichtigt werden.

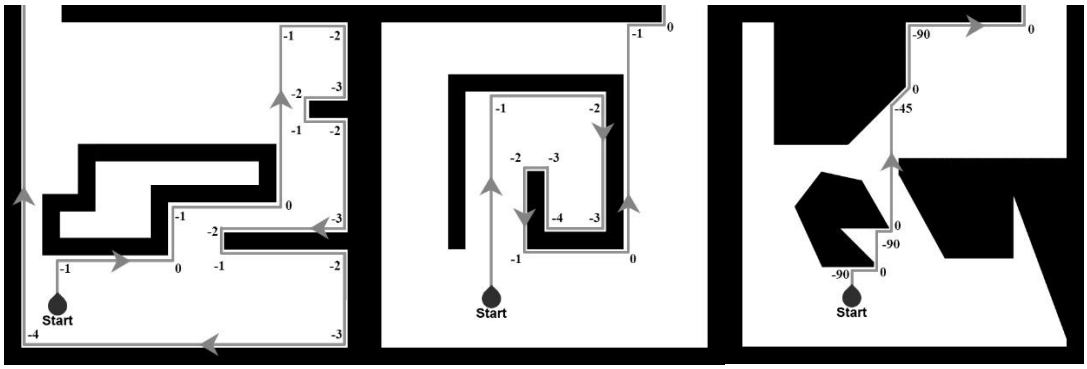

Abb. 3.1. Pledge Algorithmus Beispiele

3.3 Korrektheitsbeweis

Theorem 3.1: *Der Pledge-Algorithmus findet für einen beliebigen Irrgarten von jeder Startposition in C_{frei} einen Weg ins Freie, solange dieser existiert.*

Wir wollen hierbei zeigen, dass wenn der Pledge-Algorithmus keinen Weg findet, dieser auch nicht existiert.
Aus 3.1 ist zu entnehmen, dass wenn kein Ausgang erreicht wird, der Algorithmus nicht terminiert.
Wir zeigen zunächst, dass der Algorithmus in diesem Falle in einen Zyklus geraten ist.

Lemma 3.2: *Wenn der Pledge-Algorithmus für einen gegebenen Irrgarten L und einen Startpunkt S nicht terminiert, bewegt sich der Agent ab einem bestimmten Zeitpunkt auf einem geschlossenen Weg, welcher ewig wiederholt durchlaufen wird.*

Beweis: Solange der Roboter sich auf einer freien Fläche befindet, bzw. die Bedingung in Zeile 2 im Pseudocode als falsch ausgewertet wird, kann es keine Richtungsänderung geben. Nur an Ecken und Hindernissen dreht sich der Roboter.
Ebenso kann keine Richtungsänderung stattfinden, wenn der Roboter sich an einer Wand entlang tastet.
Daraus folgt, dass der Roboter einen Weg zurücklegt, der durch eine polygonale Kette beschrieben werden kann deren Eckpunkte zu einer der folgenden Mengen gehören:

- M_e: Menge aller Eckpunkte aller Wände im Irrgarten
- M_s: Menge aller Projektionspunkte von M_e auf das nächste Hindernis in Richtung der Startrichtung des Roboters.
  Für Startrichtung $\vec{S}$: $M_s = \{ P \mid \vec{P} = \overrightarrow{P_e} + \gamma\vec{S},\ \text{für ein } \gamma \in \mathbb{R} \text{ wobei } |\gamma| \text{ möglichst klein},$
  $P_e \in M_e,\ P \notin C_{frei}\}$

Wenn der Roboter einen Eckpunkt $P \in M_e \cup M_s$ ein zweites Mal mit demselben Winkelzählerwert besucht, wiederholt er den ganzen Weg zyklisch, da der Algorithmus deterministisch ist und aus gleichen Umständen immer derselbe Weg folgt.
Damit kein Zyklus entsteht, darf der Agent jeden dieser Eckpunkte also höchstens einmal mit dem Winkelzählerstand gleich Null besuchen.
Sind alle diese Besuche erfolgt, kann der Roboter keinen Wandkontakt mit Winkelzähler gleich Null mehr machen, da dies sonst eine Wiederholung wäre. Daraus folgt, dass der Roboter sich entweder ewig geradeaus bewegt (also den Irrgarten verlassen hat), oder ewig dieselbe Wand umrundet.
Im ersten Falle terminiert der Algorithmus, sodass der Roboter sich entweder tatsächlich in einem Zyklus befindet, oder ewig eine Wand umrundet und sein Weg in diesem Fall auch zyklisch ist.

■

Lemma 3.3: *Der Winkelzähler des Agenten nimmt nie einen Wert größer als Null an.*

Beweis: Der Winkelzähler ist beim Start des Algorithmus gleich Null. Stößt der Roboter gegen eine Wand, führt er immer eine Rechtsdrehung durch, sodass der Winkelzähler einen negativen Wert annimmt. Bei Linksdrehungen erhöht sich der Wert des Winkelzählers, sobald dieser jedoch wieder Null erreicht, löst sich der Roboter wieder von der Wand ohne weitere Linksdrehungen durchzuführen. Ab diesem Zeitpunkt gilt dieselbe Aussage wiederholt, bis der Algorithmus endet oder in einen Zyklus geriet.

■

Lemma 3.4: *Befindet sich der Agent in einem Zyklus wie im Beweis von 3.2 beschrieben, kreuzen sich die Segmente dieses Zyklus nicht.*

Beweis: Nach Definition 1.1 dürfen sich die Segmente von Wänden nicht kreuzen. Zwei freie Segmente können sich ebenfalls nicht kreuzen, da diese immer in die Startrichtung des Roboters zeigen. Es folgt, dass wenn sich zwei Segmente des Weges des Agenten kreuzen, genau eines dieser Segmente frei sein muss (Winkelzähler gleich Null).
Seien S_f, S_w ein freies und nicht freies Segment im Zyklus des Agenten die sich kreuzen. Wir definieren P als den Punkt der Kreuzung und Z als einen Punkt kurz hinter P entlang desselben Wandsegments.
$W_{S_f}(Z)$ und $W_{S_w}(Z)$ seien die Winkelzählerwerte bei Punkt Z bei entsprechender Anreise über S_w oder S_f.
Wenn der Agent auf eine Wand trifft wird eine Rechtsdrehung von maximal 180° durchgeführt. Es gilt

$$W_{S_f}(Z) \in \ ]-\pi, 0]$$
$$W_{S_w}(Z) = W_{S_f}(Z) + k * 2\pi \ mit \ k \in \mathbb{Z}$$

da nach Z der Roboter stets in dieselbe Richtung zeigt.
S_f und S_w sind nach ihrer Definition Teil desselben Zyklus, liegen aber nicht aufeinander, sodass $W_{S_w}(Z)$ nicht gleich $W_{S_f}(Z)$ sein kann, da der Algorithmus sonst nach Z stets denselben Pfad durchlaufen würde, der nicht über S_w und S_f laufen kann (in einen Zyklus ohne S_w oder S_f geriet).
k kann somit nicht Null sein.
Ebenso kann *k* nicht größer als Null sein, da der Winkel sonst positiv wäre, was Lemma 3.3 wiederspricht. Es bleibt $k < 0$, bzw. $W_{S_w}(Z) \ < \ W_{S_{f(Z)}}$.
Hieraus folgt, dass sich bei Linksdrehungen der Pfad der über S_f führt, sich ‚früher' von der Wand wieder trennt als der über S_w führt. Somit kann der Weg über S_f nie ‚hinter' dem über S_w liegen und diese sich daher nicht kreuzen, sondern nur berühren.

■

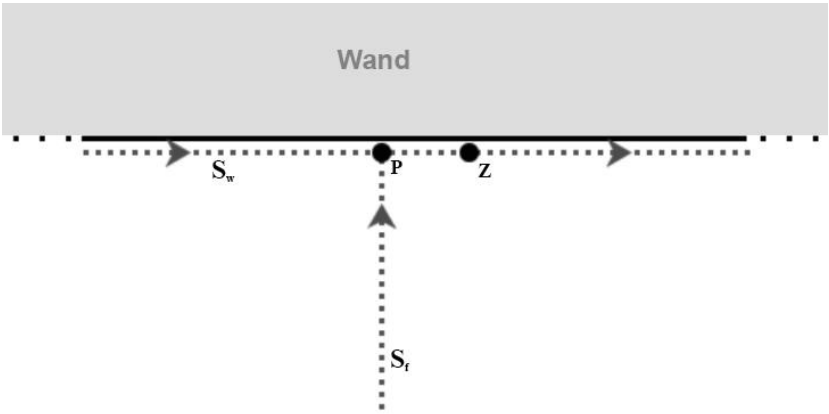

Abb. 3.2. S_f trifft auf S_w

Um Theorem 3.1 zu beweisen, betrachten wir folgendes: Der Pledge-Algorithmus terminiert alleine dann, wenn auch ein Ausgang gefunden wird. Bei Annahme, dass es einen Irrgarten gibt der einen Ausgang besitzt den der Pledge-Algorithmus allerdings nicht finden kann, bewegt sich der Roboter nach Lemma 3.2 ab einem bestimmten Zeitpunkt in einem Zyklus.
Es muss sich hierbei um einen Zyklus *im* Uhrzeigersinn handeln, da sonst bei jedem Rundgang sich der Winkelzählerwert erhöhen würde und eventuell positive Werte annehmen würde, was nach 3.3 nicht passiert.

Daraus folgt, dass der Winkelzählerwert sich nach jedem Rundgang um 2π verringert, sodass anzunehmen ist, dass dieser eventuell nur negative Werte annehmen wird. Wenn der Winkelzähler stets negativ ist, bedeutet das, dass sich der Roboter nie von einem Hindernis mehr lösen wird. Zusätzlich geht hervor, dass es sich bei diesem Hindernis nicht um eine Säule bzw. einem alleine stehenden Hindernis handeln kann, da der Pledge-Algorithmus diese immer gegen den Uhrzeigersinn umrundet (weil immer mit einer Rechtsdrehung begonnen wird).
Aus der Tatsache, dass sich der Agent in einem Zyklus ohne Kreuzungen bewegt der keine Säule umrundet, geht hervor, dass sich der Agent nicht in C_{frei} befindet, bzw. in einem Innenhof gefangen ist.

■

3.4 Fazit

Bei einem endlichen Irrgarten der einen Ausgang besitzt, kann der Pledge-Algorithmus für jeden beliebigen Startpunkt in C_{frei} garantiert in endlicher Zeit einen Ausweg finden. Die Erfolgsrate ist somit der Rechten-Hand-Methode für allgemeine Irrgärten überlegen. Die Weglänge besitzt dieselbe obere Schranke.
Jedoch sind beide Algorithmen nicht kompetitiv, da ihre Weglänge verglichen mit dem kürzesten Weg, beliebig lang werden kann.

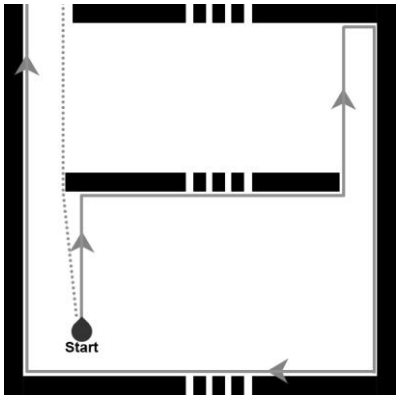

Abb. 3.3. Das Hindernis kann beliebig lange sein, sodass der Quotient zwischen Lösungsweg und kürzester Weg (gepunktete Linie) beliebig groß sein kann. Dh. es gibt keinen konstanten kompetitiven Faktor für den Pledge-Algorithmus oder die Rechte-Hand-Methode.

Ebenso scheitert der Algorithmus bei der umgekehrten Aufgabe, vom Eingang aus ein Ziel zu finden welches sich im Inneren des Irrgartens befindet. Dieses Problem ist im Allgemeinen nicht ohne weitere Hilfsmittel in endlicher Zeit lösbar.

Andere Lösungsalgorithmen für Irrgärten, wie der Trémaux-Algorithmus, sind mächtiger oder effizienter, benötigen jedoch auch mehr Hilfsmittel (zum Beispiel Markierungen am Irrgarten).

4 Literatur

[1] Bernd Brüggemann Entkommen aus einem Labyrinth mit Einbahnstraßen
http://tizian.cs.uni-bonn.de/publications/b-eaulm-06.pdf, 2006

[2] Rolf Klein Algorithmische Geometrie, Springer-Verlag Berlin Heidelberg
http://www.springer.com/de/book/9783540209560, 2005

[3] Bernd Brüggemann, Tom Kamphans, Elmar Langetepe
Leaving an Unknown Maze with One-Way Roads
https://www.researchgate.net/publication/241276366_Leaving_an_unknown_maze_with_one-way_roads, 2007