

Navigationsverfahren für Fahrzeuge mit neuartigen Antrieben

White Paper, 21.6.2010

Thomas Kämpke
InMach Intelligente Maschinen GmbH
Kässbohrerstr. 19, 89077 Ulm
kaempke@inmach.de

1 Problemstellung

Es ist zu erwarten, dass Strassenfahrzeuge mit neuartigen Antrieben wie Elektroantriebe in diversen Varianten (Hybrid, plug-in Hybrid, jeweils mit oder ohne range extender, Antrieb mit Brennstoffzellen) oder gar reine Wasserstoffantriebe andere Anforderungen an Navigations- und Dispositionssysteme stellen oder stellen werden als Fahrzeuge mit konventionellem Verbrennungsantrieb. Dies resultiert aus einem oder mehreren der folgenden Gründe: geringe Reichweite pro "Tank"füllung, relativ dünnes Tankstellennetz, lange Ladezeiten und neue Verbrauchscharakteristika z.B. aufgrund von Rekuperieren, d.h. Batterieladen beim Bremsen und bei Bergabfahrten.

Vor allem tatsächliche Reichweiten und Ladezeiten haben einen wesentlichen Einfluss auf Navigationsverfahren. Diese betreffen die Strecken- oder Wegeplanung mit Hilfe eines Navigationssystems für ein einzelnes Fahrzeug, also die Planung einer Verbindung von einem festgelegten Startpunkt- zu einem festgelegten Zielpunkt. Aber auch die Tourenplanung mittels Dispositionssystemen ist betroffen, also die Planung für eine Flotte von Fahrzeugen, die mehrere festgelegte Zielpunkte verbinden oder Teilstrecken abfahren sollen.

In Bezug auf Reichweite geht man zunächst von der maximale Reichweite aus, also von der bei Vollladung ohne Nachladen fahrbaren Distanz. Problematisch hieran ist, dass die tatsächliche Reichweite von Streckenprofil, Zuladung, eingeschalteten Verbrauchern, Verkehrssituation und evtl. auch vom Wetter abhängt. Einige Abhängigkeiten sind wahrscheinlich stärker ausgeprägt als bei Fahrzeugen mit Verbrennungsantrieb.

Grobe Anhaltswerte für den Verbrauch lediglich stadttauglicher Elektro-PKW sind 10 kWh/100km und für autobahnuntaugliche Elektro-PKW 20 kWh/100km, aber es sollen auch größere Werte ermittelt worden sein.

2 Streckenplanung

Exemplarisch wird unter der Vielzahl von neuartigen Navigationsfunktionen hier die Planung einer kürzesten Strecke unter Reichweitenbeschränkung behandelt. InMach verfügt über das know-how, bedarfsweise auch andere Funktionen zu identifizieren und Lösungsansätze unterschiedlicher Komplexität zu entwerfen, zu evaluieren und zu realisieren.

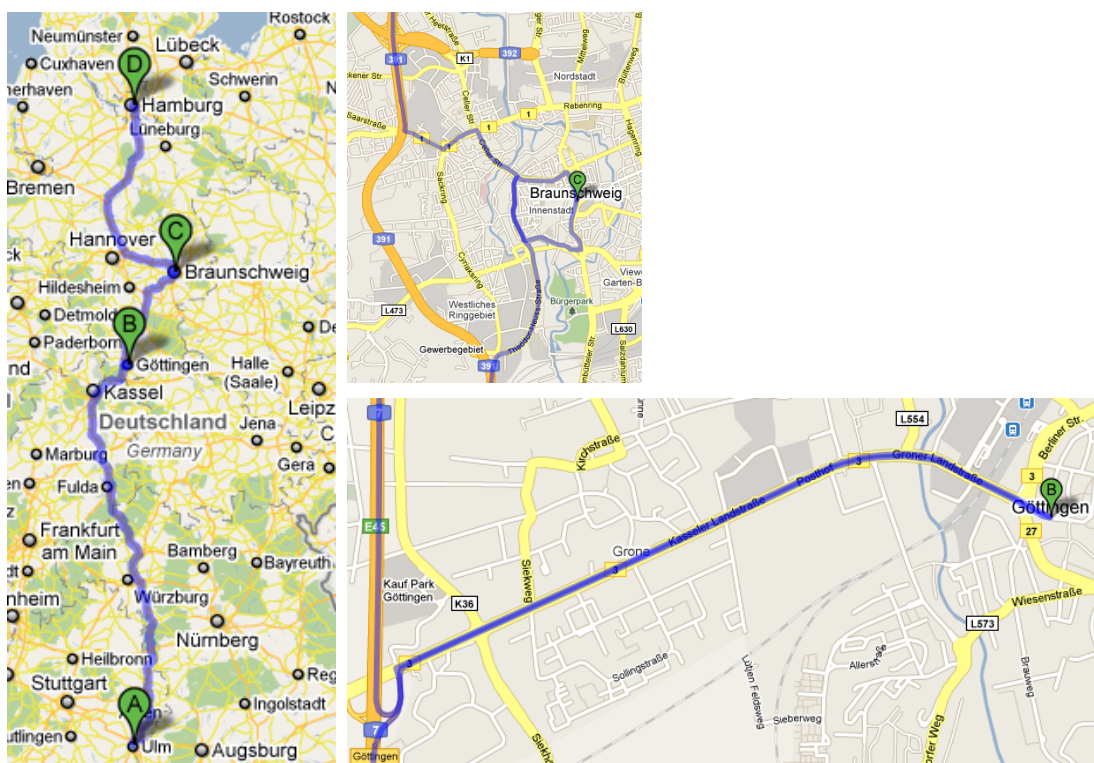
Zentrale Bedingung der Streckenplanung ist, dass die Fahrzeugreichweite kleiner als die zurückzulegende Distanz ist, dass also im Fahrtverlauf nachgeladen werden muss. Dies ist relevant bei Elektrofahrzeugen mit Wechselbatterien, wie von Fa. better place, Palo Alto, entwickelt (www.betterplace.com), der Möglichkeit zur extremen Schnellladung, also Ladung im Minutenbereich und bei Wasserstoffbetankung. Das Tanken von Wasserstoff wird im folgenden unter der Bezeichnung Laden subsummiert.

Die Planung einer günstigen, möglichst kurzen Gesamtstrecke unter dieser Nebenbedingung ist in vielen interaktiven Formen denkbar. Vom Fahrer wird dann typischerweise verlangt, rechtzeitig an das Laden zu denken und in einem dünnen, ihm möglicherweise unbekanntem Netzwerk von Ladestationen eine auszuwählen und anzufahren. Hierbei ist Hilfestellung durch luftliniengestützte Umkreissuche nach Ladestationen mit heutigen Navigationsgeräten oder mit einfachen Erweiterungen möglich.

Dabei können allerdings Fehler auftreten, d.h. der Fahrer entscheidet sich für eine Ladestation, die den kürzesten Weg übermäßig verlängert. Oder es kann als nervenzehrend empfunden werden, dass diese automatisierbare Aufgabe nicht vollständig vom Navigationsgerät gelöst wird. Für letzteres wird hier eine Kernfunktion dargestellt, die um Usabilityaspekte, Rechenzeitminimierung etc. modifiziert werden kann und muss.

Es geht hier um Prinzipialgorithmen. Übliche oder ggf. neuartige Beschleunigungsmethoden für den Algorithmeninsatz in Navigationssystemen schliessen sich an.

Ein wesentliches Merkmal optimaler Streckenführung bei Reichweitenbeschränkung ist die mögliche Wiederholung von Streckenpunkten und -abschnitten. Dies wird von heutigen Routenplanern durchaus beherrscht, allerdings nur bei ausdrücklicher Vorgabe von Zwischenzielen, siehe folgende Abbildung. Im folgenden wird die Ermittlung der erforderlichen Zwischenziele (Anzahl und Position) Bestandteil der automatischen Berechnung optimaler Strecken sein.



Mehrfachbenutzung von Kanten und Knoten in einer mit Google maps geplanten Strecke von Ulm nach Hamburg, allerdings mit explizit vorgegebenen Zwischenpunkten in den Zentren von Göttingen (unten rechts) und Braunschweig (oben rechts).

3 Algorithmen

3.1 Ansatz

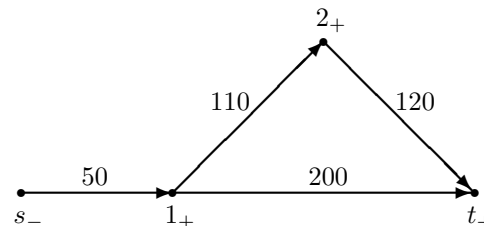
Die Wegeplanung verfolgt letztlich an jedem Entscheidungspunkt die Strategie, nicht zu laden, sofern die Erreichbarkeit des nächsten Entscheidungspunkts inkl. eines möglichen Sicherheitszuschlags gegeben ist. Die Schwierigkeit bei diesem späten laden ist u.a., dass Entscheidungspunkte zur Wegeplanung gewöhnlich keine Ladestationen sind.

Das Streckennetz wird durch einen einfachen gerichteten Graphen $G = (V, E)$ beschrieben, wobei bedarfsweise zwischen Knoten ohne Ladestation v_- und Knoten mit Ladestation v_+ unterschieden wird; $V = V_- \cup V_+$. Jede gerichtete Kante $e \in E$ hat eine positive Länge $l(e) > 0$, wobei die Längen von gegenläufigen Kanten zwischen zwei Knoten verschieden sein können. Eine Kante (v, w) kann von einem Fahrzeug nur befahren werden, wenn seine Reichweite $r(v)$ im Anfangsknoten mindestens so gross ist wie die Kantenlänge plus ggf. einem Zuschlag, d.h.

$$r(v) \geq l(v, w) \text{ bzw. } r(v) \geq l(v, w) + \text{Zuschlag.}$$

Fahrzeugreichweiten kann man nicht dadurch berücksichtigen, dass zu lange Kanten ignoriert werden, denn, ob eine Kante zu lang ist, hängt von Bedingungen ab, unter denen der Anfangsknoten einer Kante erreicht wird oder erreicht werden kann. Das Fahrzeug hat eine maximale Reichweite r_{max} . Nur Kanten mit $l(e) > r_{max}$ können ohne weitere Bedingung aus dem Streckennetz entfernt werden.

Eine der Schwierigkeiten der Bestimmung kürzester Wege unter Reichweitenbeschränkung besteht darin, dass ein Zielknoten nicht erreichbar ist obwohl das Streckennetz zusammenhängend ist oder dass ein Zielknoten zwar erreichbar ist, von dort aus aber keine Ladestation mehr erreicht werden kann. Weitere Schwierigkeiten der Bestimmung eines kürzesten Weges von s nach t unter Reichweitenbeschränkung zeigt folgendes Beispiel mit nur vier Knoten für ein Fahrzeug mit maximaler Reichweite $r_{max} = 220$ und initialer Reichweite $r(s) = 180$.



Es kann in Knoten 1, 2 und t geladen werden, nicht aber im Knoten s . Entlang des Weges $s, 1, t$ wird bei Knoten 1 geladen, d.h. $r(1) = 220$, während entlang des Weges $s, 1, 2, t$ erst bei Knoten 2 geladen werden muss, d.h. $r(1) = 130$ und $r(2) = 220$. Würde entlang des Weges $s, 1, 2, t$ in Knoten 1 geladen, so müsste trotzdem ein weiteres mal in Knoten 2 geladen werden. "Spät laden" ist also sinnvoll, bezieht sich aber jeweils auf einen Weg, nicht auf einen Baum oder einen anderen Teilgraphen, mittels dessen der kürzeste Weg von s nach t berechnet wird. Der kürzeste Weg ist hier offensichtlich $s, 1, t$.

Die Bestimmung eines kürzesten Weges von s nach t unter Reichweitenbeschränkung erfolgt mit einer Variante des Dijkstra Algorithmus. Zunächst der reine Dijkstra Algorithmus zur Berechnung eines kürzesten Weges ohne Reichweitenbeschränkung, bei dem allerdings alle zu langen Kanten bereits entfernt sind.

Dijkstra

1. Eingabe

Graph $G = (V, E)$ mit Kantenmarkierungen $l : E \rightarrow (0, r_{max})$ und $s, t \in V$.

Initialisierung

$T = V$, $m(s) = 0$ und $m(v) = \infty$ für alle $v \in V - \{s\}$.

2. While $t \in T$ do

(a) Finde $i \in T$ mit $m(i) = \min_{k \in T} m(k)$.

(b) $T = T - \{i\}$.

(c) $\forall k \in T$ mit $(i, k) \in E$ do

If $m(i) + l(i, k) < m(k)$ then $m(k) = m(i) + l(i, k)$ und $pred(k) = i$.

3. Ausgabe

(a) kürzeste Weglänge $m(t)$.

(b) Kürzester Weg in Rückwärtsnotation $t, pred(t), pred(pred(t)), \dots, s$.

Die Werte $m(i)$ für alle Knoten i sind Markierungen, die vom Algorithmus in jeder Phase in vorläufig und endgültig eingeteilt werden. Die vorläufig markierten Knoten sind in der Menge T zusammengefasst, die endgültig markierten ergeben sich als das Komplement $V - T$. Für jeden Knoten bezeichnet die Markierung in allen Phasen des Algorithmus die Länge des kürzesten, bisher gefundenen Weges von s aus. Das Minimum dieser Werte über alle vorläufig markierten Knoten, das in jeder Iteration in Schritt 2(a) gebildet wird, ist monoton steigend im Verlauf der Iterationen. In Schritt 2(c) propagiert der Knoten i seine kürzeste Weglänge zu allen Knoten k , die in einem Schritt erreichbar und noch nicht endgültig markiert sind.

Der Algorithmus besitzt eine Asymmetrie in dem Sinne, dass ein Knoten in mehreren Iterationen Propagierungen empfangen kann aber nur einmal aussendet. Diese Asymmetrie wird zur Symmetrie bei der Berücksichtigung von Reichweitenbeschränkungen. Ein Knoten kann dann in mehreren Iterationen Propagierung empfangen und senden.

Der Dijkstra Algorithmus findet einen kürzesten Weg, sofern es überhaupt einen Weg von s nach t gibt. Wenn der Algorithmus allerdings in Schritt 2(a) in die Situation $m(k) = \infty$ für alle $k \in T$ läuft, dann ist t von s aus nicht erreichbar. Darüber hinaus findet der Algorithmus einen kürzesten Weg von s zu allen Knoten, die bei Terminierung als endgültig markiert angesehen werden, also in $V - T$ liegen. Dies ist die sog. 1:n Eigenschaft ("one to many" Eigenschaft) des Dijkstra Algorithmus.

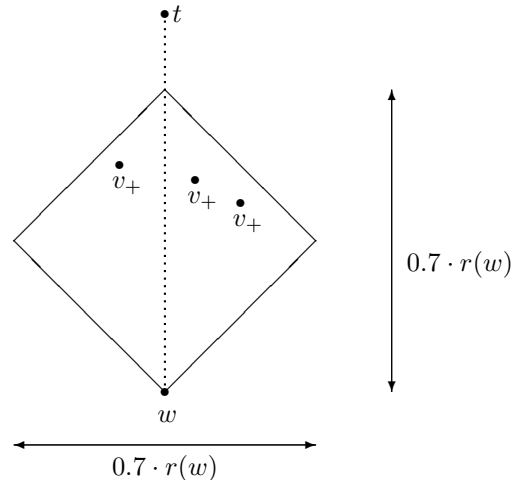
Um Rechenaufwand zu vermeiden, werden Knoten auch dann für endgültig markiert erklärt und nicht weiter betrachtet, wenn sie gemäß einiger Heuristiken als Zwischenknoten eines kürzesten Weges ausgeschlossen werden. Die Minimierung in Schritt 2(a) ersetzt man z.B. durch

$$m(i) + ||i - s|| = \min_{k \in T} m(k) + ||k - s||.$$

Derartige Euklidische Heuristiken begünstigen die tatsächlichen Rechenzeiten wesentlich, aber es kann dazu kommen, dass ein kürzester Weg nicht gefunden wird. Typischerweise wird dann aber immer noch ein guter Weg gefunden. Euklidische Heuristiken werden ergänzt durch ein hierarchisches Vorgehen mit Priorität für Autobahnstrecken. Ein weiteres Beschleunigungselement, das mit vielen anderen kombiniert werden kann, besteht darin, den Algorithmus terminieren zu lassen, sobald $m(t)$ einen endlichen Wert annimmt. Der Algorithmus findet dann generell nicht einen besten, aber einen erst-besten Weg.

3.2 Heuristik mit Suchgebiet bei Reichweitenbeschränkung

Von einem Punkt w auf irgendeiner Strecke nach t kann ein Suchgebiet in Richtung der Luftlinie zum Ziel t aufgespannt werden, in dem nach Ladestationen gesucht wird.



Vom aktuellen Punkt w wird dann ein kürzester Weg zu einigen oder allen diesen Ladestationen bestimmt und von jeder dieser Ladestationen wird ein kürzester Weg ohne Reichweitenbeschränkung zum Ziel berechnet. Die insgesamt günstigste Ladestation aus dem aktuellen Suchgebiet wird dann ausgewählt, d.h. es wird ausgewählt gemäß

$$\min_{v_+ \in S} m_w(v_+) + m_{v_+}(t).$$

Kürzeste Wege vom aktuellen Knoten zu allen Ladestationen im Suchgebiet können aufgrund der 1:n Eigenschaft des Dijkstra Algorithmus durch einmalige Anwendung bestimmt werden.

Kürzeste Wege von den Ladestationen zum Ziel können näherungsweise durch Umkehrung berechnet werden, d.h. durch die Berechnung kürzester Wege vom Ziel zu den Ladestationen. Auch dies ist aufgrund der 1:n Eigenschaft durch eine einzige Anwendung möglich. Eine gröbere Näherung für die geringste Streckenlänge von einer Ladestation zum Ziel ist die Luftliniendistanz, d.h. der Ersatz von $m_{v_+}(t)$ durch $\|v_+ - t\|$. Der Einsatz von Näherungen erfolgt hier nur zur Zwischenzielauswahl und nicht zur tatsächlichen Streckenberechnung, so dass Verzerrungen in Wertevergleichen nur begrenzt verfälschen.

Ggf. muss die Suche nach Ladestationen entlang der weiteren Strecke wiederholt werden. Zwar ist das Vorgehen mit Suchgebieten grundsätzlich einfach, aber ein Problem liegt u.a. darin, dass mehrere Parameter zu setzen sind. Ein Parameter bestimmt den Start des Verfahrens. Z.B. wird das Verfahren gestartet, wenn die Restreichweite bei höchstens 20% der maximalen Reichweite liegt, also $r(w) \leq r_{max}$. Dies kann aber zu früh sein, d.h. es liegen sehr viele Ladestationen im Suchgebiet oder es kann zu spät sein, d.h. es liegt keine Ladestation im Suchgebiet. Dann ist das Suchgebiet dynamisch zu vergrößern.

Ebenso kann das Suchgebiet verschiedene Formen haben und mit entsprechenden Formparameter versehen werden. So kann das angedeutete Suchgebiet auch die Größenparameter $0.8 \cdot r(w)$ in Zielrichtung und $0.6 \cdot r(w)$ senkrecht dazu haben oder das Suchgebiet kann kreisförmig sein.

Wenn mittels Suchgebieten ein Weg zum Zielpunkt tatsächlich gefunden wird, ist nicht garantiert, dass er der kürzeste ist.

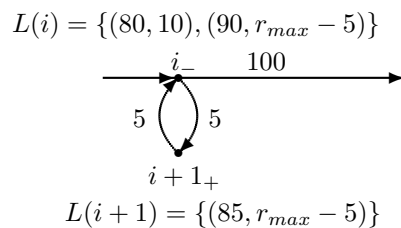
3.3 Algorithmus mit Reichweitenbeschränkung

Berechnungen kürzester Wege mit Reichweitenbeschränkung erfordern kompliziertere Markierungen als Berechnungen gewöhnlicher kürzester Wege. Die einzelne Markierung $m(i)$ des reinen Dijkstra Algorithmus für einen Zwischenknoten wird durch eine Liste von Paaren

$$L(i) = \{(m_1(i), r_1(i)), \dots, (m_{n_i}(i), r_{n_i}(i))\}$$

ersetzt. Jedes Paar entspricht der Länge und der Restreichweite eines bisher gefundenen Weges von s nach i . Die Anzahl n_i der Paare für Zwischenknoten i kann sich im Laufe der Iterationen ändern. Sofern ein Knoten i Ladestation ist, besteht seine Markierungsliste stets nur aus dem einen Paar $(m(i), r_{max})$.

Ein prägnanter Unterschied zwischen kürzesten Wegen mit und ohne Reichweitenbeschränkung besteht darin, dass ein optimaler Weg im Fall von Reichweitenbeschränkungen zu Knoten zurückkehren kann. Sogar mehr als zwei Besuche ein und desselben Knotens können optimal sein, wenn Hin- und Rückwege zu Ladestationen unterschiedlich lang sind. Eine Rückkehrsituation liegt vor, wenn nicht an einem bestimmten Knoten aber in dessen Nähe geladen werden kann und muss, vgl. nachfolgende Abbildung.



Die kleinste Weglänge, die in der Liste auftaucht wird mit

$$m.\min(L(i)) = \min\{m \mid \exists r \text{ mit } (m, r) \in L(i)\}$$

bezeichnet. Es werden nur Markierungspaare von Wegen gespeichert so, dass keiner der bisher gefundenen Wege von s nach i einen anderen dominiert. Das bedeutet $m_k(i) < m_j(i) \Rightarrow r_k(i) < r_j(i)$. Wenn also der Weg Nr. k von s nach i kürzer ist als der Weg Nr. j (Weg k ist besser als Weg j bzgl. Weglänge), dann ist auch die Restreichweite von k kleiner als die von j (Weg k ist schlechter als Weg j bzgl. Restreichweite). Dominierte Wege von s zum Zwischenknoten i werden eliminiert und, falls ein Weg alle anderen dominiert, werden alle anderen eliminiert.

Aus den Beschränkungen, also aus den Nebenbedingungen, wird eine weitere Zielgröße. Dünne einen zum Schluss gefundenen Weg von s nach t mit minimaler Länge gemäß "spät laden" aus. Im Mehrkriterienfall ist die Vorgängerfunktion, mit der ein Weg rückwärts aufgebaut wird, komplizierter als im reinen Dijkstra Algorithmus, da sowohl auf einen Vorgängerknoten als auch auf eine von dessen Markierung verwiesen wird:

$$pred(i, (m_k(i), r_k(i))) = (j, (m_p(j), r_p(j))).$$

Von Tripeln aus Knoten und Markierung wird bedarfsweise die erste Koordinate herausgefiltert, d.h. $comp_1(i, (m_k(i), r_k(i))) = i$.

KW-Reichweite

1. Eingabe

Graph $G = (V, E)$ mit Knotenmenge $V = V_- \cup V_+$ und Kantenmarkierungen $l : E \rightarrow (0, r_{max})$, $s, t \in V$ und initiale Reichweite $r(s) \in (0, r_{max}]$.

Initialisierung

$T = V$, $e(v) = (NAN, NAN)$ für alle $v \in V$, $L(s) = \{(0, r(s))\}$, $L(v) = \{(\infty, 0)\}$ für alle $v \in V - \{s\}$ und $L^*(t) = \emptyset$.

2. While $t \in T$ do

- (a) Finde $i \in T$ und Index d mit $m_d(i) = m_min_{k \in T} L(k)$.
- (b) $L(i) = L(i) - \{(m_d(i), r_d(i))\}$.
- (c) If $L(i) = \emptyset$ then $T = T - \{i\}$. (Ein einmal aus T entfernter Knoten kann in einer späteren Iteration in Schritt (d)ii. wieder eine Markierung erhalten und damit wieder zu T hinzugefügt werden!)
- (d) If $L(i) = \emptyset$ and $e(i) = (NAN, NAN)$ then $e(i) = (m_d(i), r_d(i))$.
- (e) $\forall k \in V$ mit $(i, k) \in E$ do
If $r_d(i) \geq l(i, k)$ then
 - i. If $k \in V_-$ then $(m, r) = (m_d(i) + l(i, k), r_d(i) - l(i, k))$ else
 $(m, r) = (m_d(i) + l(i, k), r_{max})$.
 - ii. If $L(k) \neq \emptyset$ then $L(k) = L(k) \cup \{(m, r)\}$
else if (m, r) nicht durch $e(k)$ dominiert then $T = T \cup \{k\}$ und $L(k) = \{(m, r)\}$.
 - iii. $L(k) = PO(L(k))$.
 - iv. If $(m, r) \in L(k)$ then $pred(k, (m, r)) = (i, (m_d(i), r_d(i)))$.
 - v. If $k = t$ then $L^*(t) = L^*(t) \cup \{(m, r)\}$.

3. Ausgabe

- (a) Kürzeste Weglänge $m(t) = m_min(L^*(t))$.
- (b) Kürzester Weg in Rückwärtsnotation $t, comp_1(A_1), comp_1(A_2), \dots, s$. Dabei ist $pred(t, (m(t), r(t))) = A_1, pred(A_1) = A_2, \dots, pred(A_k) = A_{k+1}$ mit $comp_1(A_{k+1}) = s$ und $pred(A_{k+1}) = \text{void}$.

NAN: not a number. Der Exiteintrag $e(v)$ für Knoten v ist die letzte Markierung, bevor die Markierungsliste von v zum ersten mal leer wird. Eventuelle, später für den Knoten v vorgesehene Markierungen haben größere m -Werte und werden nur bei größeren Reichweiten als der des Exiteintrags berücksichtigt.

Problem bei der Propagierung: das Streichen von Markierungen durch Abarbeiten wie in Schritt 2(b) erschwert das Streichen weiterer Markierungen durch Dominanztest. Die Operation $PO(L(k))$ filtert aus der Markierungsmenge $L(k)$ alle Pareto-optimalen heraus, die anderen entfallen.

4 Erweiterungen

In dünnen Netzwerken von Ladestationen ist nicht nur die Planung einer Einzelstrecke relevant, sondern es muss z.B. auch am Zielort noch mindestens eine Ladestation in Reichweite liegen. Das vorgeschlagene Verfahren kann hierzu mit entsprechenden Eingaben im Hintergrund arbeiten.