AN ALGORITHM FOR THE PROBLEM OF SHORTEST TRAIL WITH PROHIBITED MANEUVERS

Tomáš MAJER

Faculty of Management and Informatics
University of Zilina
e-mail: tomas.majer@fri.uniza.sk

Abstract

Traditional shortest path algorithms do not respect real life constraints as prohibited left turn, prohibited right turn and prohibited U – turn. More sophisticated street models contain even more complicated prohibited maneuvers. This paper studies shortest path and shortest trail algorithms which can find shortest trail respecting these prohibited motions. Proposed algorithms have to be fast and have to be suitable for restricted computational resources (mobile GPS navigators).

Keywords: *graph theory, shortest path*

1 INTRODUCTION

The part of graph theory concerning shortest path problem seemed to be closed. Many types of shortest path algorithms were published, implemented and succesfully applied in practice. However, usable route planner should propose such instructions which obey traffic rules. Most of them are one way streets and prohibited left turn, prohibited right turn and prohibited U-turn. Effective implementation of all requirements and additional constraints expects development of new exact or suboptimal mathematical methods.

In [4] we presented two algorithms to search feasible shortest trail with respect to prohibited maneuvers based on algoritms published in [6] and [5]. First of them respects a simple prohibited maneuvers consisting of only two arcs. Second algorithm is more sophisticated and respects any prohibited manuever, but is too complex and slow. In this paper we present some modification of the second algorithm.

2 BASIC NOTATION

Graph theory terminology is consedarably nonuniform. That is why we introduce here several fundamental graph definitions. Directed graph will be the fundamental structure for our research.

Definition 1 *A* digraph *(a directed graph) is an ordered pair $G = (V, A)$, where V is a nonempty finite set and A is a set of ordered pairs of the type (u, v) such that $u \in V$, $v \in V$ and $u \neq v$. The elements of V are called* vertices *and the elements of A are called* arcs *of the digraph G.*

Definition 2 *A (v_1, v_k) -* walk *in digraph $G = (V, A)$ is an alternating sequence of vertices and arcs of the form*

$$\mu(v_1, v_k) = (v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \ldots, v_{k-1}, (v_{k-1}, v_k), v_k). \quad (1)$$

A (v_1, v_k) - trail *in G is a (v_1, v_k) - walk in G with no repeated arcs. A (v_1, v_k) -* path *in G is a (v_1, v_k) - walk in G with no repeated vertices.*

Definition 3 *Let* (1) *be a walk μ. A* subwalk *of μ is arbitrary subsequence of* (1) *starting and finishing with a vertex, i.e.:*

$$(v_i, (v_i, v_{i+1}), v_{i+1}, \ldots, v_{j-1}, (v_{j-1}, v_j), v_j), \quad (2)$$

where $1 \leq i \leq j \leq k$.

A simple prohibited left or right turn can be modelled as a prohibited pair of arcs. In more detailed models prohibited left turn, prohibited right turn and prohibited U-turn can be formulated as a prohibited sequence of more vertices and arcs which will be called a prohibited maneuvre. Prohibited maneuvre is a traffic engineering notion – it cannot be derived from graph theory properties of road network. Prohibited maneuvers belong to input data of shortest route problem.

Definition 4 *A* prohibited maneuvre *is a walk declared as prohibited. A walk μ is* feasible with respect to prohibited maneuvers *(or only* feasible*), if no prohibited maneuvre is a subwalk of μ.*

Definition 5 *A* simple prohibited maneuvre *is a walk declared as prohibited containing only two arcs, i.e.*

$$\omega = (i, (i, j), j, (j, k), k), \quad (3)$$

Definition 6 *An* arc weighted digraph *$G = (V, A, c)$ is an ordered triple where $G = (V, A)$ is a digraph and $c : A \to R$ is a real function defined on the arc set A, the value $c(a)$ for $a \in A$ is called the* weight *of the arc a (or sometimes the* arc-weight*, the* length *or the* cost *of the arc a). In this paper we will assume that $c(a) \geq 0$. This condition is fulfilled in many practical applications.*

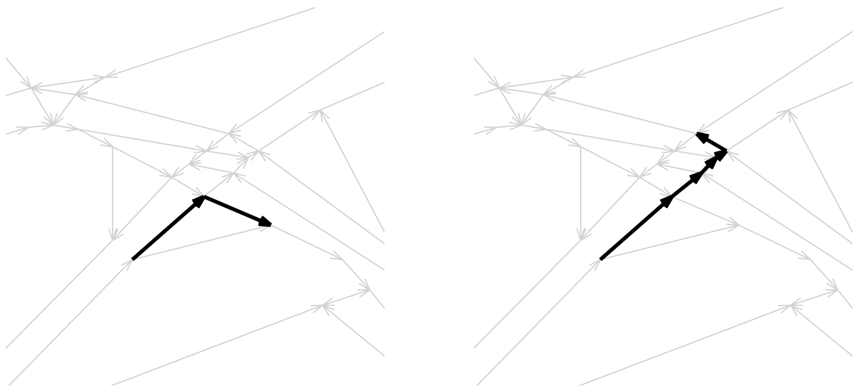

Figure 1 Some of prohibited maneuvers of real junction

Definition 7 *The* length *of the walk $\mu(u, v)$ in a digraph $G = (V, A, c)$ is the total sum of arc-weights of its arcs, whereas the arc weight is added to the total sum so many times how many times it appears in the walk.*

Definition 8 *Denote $V^+(v) = \{w|(v, w) \in A\}$, $A^+(v) = \{(v, w)|(v, w) \in A\}$. The* forward star $\mathrm{Fstar}(v)$ *of the vertex $v \in V$ is the subgraph of $G = (V, A)$ with vertex set $V^+(v) \cup \{v\}$ with arc set $A^+(v)$, i. e.* $\mathrm{Fstar}(v) = (V^+(v) \cup \{v\}, A^+(v))$.

3 AN ALGORITHM TO SEARCH FEASIBLE SHORTEST TRAIL

Palúch and Peško in [6] present a solution how to find shortest trail with respect to simple prohibited meneuvres (prohibited turns) in digraph $G = (V, A)$. In [4] we presented implementation of this algoritm as Label–Set algoritm with labels for arcs instead of vertices.

In [4] we presented an algorithm to search feasible shortest trail with respect to complex prohibited meneuvers as shown on Fig. 1 too. It is based on Palúch's k shortest paths multi label algorithm presented in [5].

Algorithm 1 *Multi label algorithm for shortest u–v trail with respect to prohibited maneuvers in digraph $G = (V, A)$.*

$\forall i \in V$ we remember a set of definitive labels $\mathcal{L}_i$. Every label is in the form

$(k, t, x, x_{\mathrm{k}})$, *where*

k	*is the sequence number of u–i trail*
t	*is the length of u–i trail*
x	*is last but one vertex on u–i trail*
x_{k}	*is the sequence number of the trail into last but one vertex x*

An element of heap $\mathcal{E}$ is in the form $(w, t, x, x_{\mathrm{k}})$, where

w	*is candidate to become pivot vertex*
t	*is length of u–w trail, the priority of element of heap*
x	*is last but one vertex on u–w trail*
x_{k}	*is sequence number of trail into last but one vertex x*

Step 1. Initialization.
Let $\mathcal{L}_u = \{(1, 0, 0, 0)\}$ and $\mathcal{L}_i = \emptyset\ \forall i \in V, i \neq u$.
Let $\mathcal{E} := \emptyset$. $\forall i \in V^+(u)$ let $\mathcal{E} := \mathcal{E} \cup \{(i, c(u, i), u, 1)\}$.

Step 2. *Get an element $(w_{\min}, t_{\min}, x_{\min}, x_{\mathrm{k}\min})$ from heap $\mathcal{E}$ with minimal $t_{\min}$.*

- *Let $k = |\mathcal{L}_{w_{\min}}| + 1$*
- $\mathcal{L}_{w_{\min}} := \mathcal{L}_{w_{\min}} \cup \{(k, t_{\min}, x_{\min}, x_{\mathrm{k}\min})\}$
- $\forall i \in V^+(w_{\min})$*:*
  if arc $(w_{\min}, i)$ is not in u–$w_{\min}$ trail and trail extended by arc $(w_{\min}, i)$ does not contain prohibited maneuver then

$$\mathcal{E} := \mathcal{E} \cup \{(i, t_{\min} + c(w_{\min}, i), w_{\min}, k)\}$$

.

Step 3. *If $\mathcal{E} \neq \emptyset$ and $\mathcal{L}_v = \emptyset$, GOTO Step 2.*
Else STOP. We construct u–v trail using labels from $\mathcal{L}_i$:

$$\mu(u, v) = (u = w_1, (w_1, w_2), w_2, \ldots, w_{s-1}, (w_{s-1}, w_s), w_s = v)$$

Let $\mathcal{L}_i[j]$ be a label of vertex i with $k = j$, so if $(k, t, x, x_{\mathrm{k}}) \in \mathcal{L}_i$ then $\mathcal{L}_i[k] = (k, t, x, x_{\mathrm{k}})$. Let $\mathcal{L}_i[j]^{()}$ be a component part of label, $\mathcal{L}_i[k]^{(t)} = t$, $\mathcal{L}_i[k]^{(x)} = x$ and $\mathcal{L}_i[k]^{(x_{\mathrm{k}})} = x_{\mathrm{k}}$. Then

$$
\begin{aligned}
w_s &= v, & k_s &= 1 \\
w_{s-1} &= \mathcal{L}_{w_s}[k_s]^{(x)}, & k_{s-1} &= \mathcal{L}_{w_s}[k_s]^{(x_k)} \\
w_{s-2} &= \mathcal{L}_{w_{s-1}}[k_{s-1}]^{(x)}, & k_{s-2} &= \mathcal{L}_{w_{s-1}}[k_{s-1}]^{(x_k)} \\
&\vdots & &\vdots \\
w_1 &= \mathcal{L}_{w_2}[k_2]^{(x)} = u
\end{aligned}
$$

Condition “if arc $(w_{\min}, i)$ is not in u–$w_{\min}$ trail and trail extended by arc $(w_{\min}, i)$ does not contain prohibited maneuver” in **Step 2.** of this algorithm is complex and slow and need to be optimized.

- Test, if trail contains prohibited maneuver, is easy and with complextity $O(1)$, if the set of prohibited maneuvers is implemented as a dictionary with last vertex in this maneuver as the key.
- Test, if arc $(w_{\min}, i)$ is not in u–$w_{\min}$ trail consists of this steps:
  - If $\mathcal{L}_i$ is empty set, the arc $(w_{\min}, i)$ is not in any u–$w_{\min}$ trail.
  - Else $\mathcal{L}_i[1]^{(t)}$ is a lower bound of length of any u–$w_{\min}$ trail, its can contain arc $(w_{\min}, i)$. So analysing of trail using backtracking can be aborted at label $\mathcal{L}_x[y]$ with $\mathcal{L}_x[y]^{(t)}$ less than this lower bound and must not end at vertex u. This reduces computional complextity, because the difference between the length of shortest and k–th shortest path is low and so backtracing ends after analysing small number of labels.

Average running time of program and number of analysed labels of backtracing without applying lower bounds (Algorithm 1) and with applying lower bounds (Algorithm 2) are shown in Table 1.

Algorithm is implemented using Microsoft Visual Studio 2010, programming language C# with target framework .NET 4.0. Program was started on HP ProBook 6550 with AMD Phenom(tm) II N830 Tripple-Core 2.1 GHz Procesor and 4 GB of main RAM.

Averege values are computed from characteristics of computing 100 feasible shortest trails with randomly choosen start and end vertices. This vertices are junctions of model of road map of Slovakia. This model consists of 237 417 vertices, 516 463 arcs

Algorithm	**Running time [s]**	**Created labels**	**Analysed labels**
Algorithm 1	65.80	517 413	238 968 199
Algorithm 2	2.27	517 413	522 149

Table 1 Comparation of algorithms for feasible shortes trail

and 12 360 prohibited maneuvers. Feasible shortest trail contains in average 628 arcs and its length is in average 196,5 km.

4 REDUCTION OF ARC WEIGHT OF ROAD MAP MODEL

Another speed–up of algorithm can be accomplished using reduced arc weight as proposed by Hart, Nilsson and Raphael in A^*–algoritm in [1].

Let $\mathrm{d}(v_i, v_j)$ be a metric on set of vertices V and it is a lower bound of length of shortest $v_i - v_j$ path. Let $v \in V$ be the end vertex of searched shortest path in digraph G. For every arc $a \in A$ we can compute a reduced weight $\overline{\mathrm{c}}$ of this arc as

$$\forall a = (v_i, v_j) \in A: \ \overline{\mathrm{c}}(a) = c(a) - \mathrm{d}(v_i, v) + \mathrm{d}(v_j, v). \tag{4}$$

Let μ be any $u - v$ walk

$$\mu = (u = v_0, (v_0, v_1), v_1, (v_1, v_2), v_2, \dots, v_{n-1}, (v_{n-1}, v_n), v_n = v).$$

The reduced length of μ, denoted $\overline{\mathrm{c}}(\mu)$, can be computed as

$$\begin{aligned} \overline{\mathrm{c}}(\mu) &= \sum_{i=1}^{n} [\mathrm{c}((v_{i-1}, v_i)) - \mathrm{d}(v_{i-1}, v) + \mathrm{d}(v_i, v)] \qquad (5) \\ &= \mathrm{c}(\mu) - \mathrm{d}(v_0, v) + \mathrm{d}(v_n, v) = \mathrm{c}(\mu) - \mathrm{d}(u, v). \end{aligned}$$

So difference between the length and the reduced length of any path in G is constant. Then any path in G is shortest path with respect to original arc weight c if and only if this path is shortest path with respect to reduced arc weight $\overline{\mathrm{c}}$. This equivalence is true for the second shortest paths, for k–th shotest paths and for feasible shortest trails too.

Let $G = (V, A, \mathrm{c})$ be a digraph that represents a road map (i.e. road map of Slovakia). Let $\mathrm{GPS_e}(v_i)$ be a longitude and $\mathrm{GPS_n}(v_i)$ be a latitude of vertex $v_i \in V$. Then distance $\mathrm{d}(v_i, v_j)$ of two vertices $v_i, v_j \in V$ can be computed using haversine method as

$$\begin{aligned} \mathrm{d}(v_i, v_j) &= R * \arccos(\cos(\mathrm{GPS_n}(v_i)) * \cos(\mathrm{GPS_n}(v_j)) + \qquad (6) \\ &\quad \sin(\mathrm{GPS_n}(v_i)) * \sin(\mathrm{GPS_n}(v_j)) * \cos(\mathrm{GPS_e}(v_i) - \mathrm{GPS_e}(v_j))), \end{aligned}$$

where $R = 6,372,797.0$ is radius in meters of Earth.

Average running time of program and number of created labels by searching feasible shortest trail in digraph G with respect to original arc weight and with respect to reduced arc weight using optimized algorithm (Algorithm 2) are shown in Table 2. Graphical views of labeled vertices in digraph G are on Figures 2 and 3.

Algorithm	**Running time [s]**	**Created labels**	**Analysed labels**
original	2.27	517 413	522 149
reduced	1.42	189 534	207 975

Table 2 Comparation of searching feasible shortest trail with respect to original and reduced arc cost

Figure 2 Labeled vertices by Algorithm 2 in directed graph with original arc weight

5 CONCLUSION

Authors in [2, 3] present another fast algorithms to search shortest path. Principes used in this algorithms can be used to speedup algorithm to search feasible shortest trail with respect to prohibited maneuvers too. We want to proof this modification in the future work.

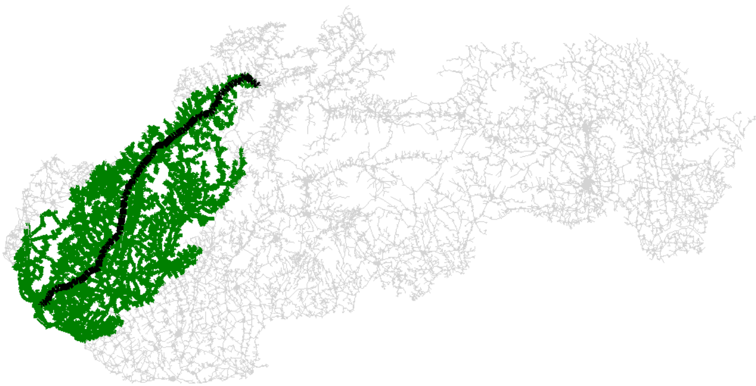

Figure 3 Labeled vertices by Algorithm 2 in directed graph with reduced arc weight

REFERENCES

[1] Hart,P.E – Nilsson,N.J. – Raphael,B.: *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions of Systems Science and Cybernetics, Vol. SSC-4, No. 2, 1968

[2] Goldberg,A.V. – Harrelson,Ch.: *Computing the Shortes Path: A^* Search Meets Graph Theory*, 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’05), Vancouver, Canada, 2005

[3] Gutman,R.: *Reach–based routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks*, 6th International Workshop on Algorithm Engineering and Experiments, pp. 100–111, 2004

[4] Majer, T.: *Shortest Trail Problem with Respect to Prohibited Maneuvers*, Mathematical Method in Economics 2010, České Budějovice, pp. 418–422, 2010

[5] Palúch, S.: *A Multi Label Algorithm for k Shortest Paths Problem*, Communications **3** (2009), pp. 11–14.

[6] Palúch, S., Peško, Š.: *Kvantitatívne metódy v logistike*, Žilinská univerzita v Žiline, Žilina, 2006.

Acknowledgement

The author is pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic VEGA under the grant No. 1/0374/11.