# OPTIMAL PATH DETERMINATION OBSERVING TURN RESTRICTIONS

P. H. Richter
OS-Consultancy
10369 Berlin, Hohenschönhauser Str. 1
Germany

## OVERVIEW

The paper gives the mathematical concept as well as an algorithm that efficiently tackles the optimal path problem considering turn restrictions. The internal graph representation remains unchanged apart from the necessary assignment of turn restriction. The algorithm develops ***R**everse **O**ptimal **P**ath Graphs* (*ROPG*s) that are not necessarily cycles-free due to the turn restrictions' requirements. *ROPG*s enable the *successor feature* (in contrast to the conventional *predecessor feature*), i.e. only one path exploration is necessary for a given target $\beta$ that assigns each road uniquely one successor road via the shortest distance towards target $\beta$. Regardless to which road a driver is forced to change, a path recalculation is not necessary if the target $\beta$ remains unchanged. An intensive performance analysis is given. The paper reveals how turn restrictions have to be assigned to traffic points such that the additional time effort proving turn restrictions during the path exploration remains minimum.

## 1 INTRODUCTION

We regard the ***All-to-One problem optimal path algorithm*** that doesn't start its path exploration (as usual) in a specific start point(s) but in the destination $\beta$ scanning the edges in reverse direction and building a *Reverse Optimal Path Graph ROPG denoted here with* $\overline{\mathfrak{T}}(\beta) \subseteq \mathbf{G}$. And we take into consideration comparative ***One-toAll*** algorithms also able to observe turn restrictions. These algorithms start the path exploration in start point $\alpha$ and develop an *Optimal Path Graph OPG* denoted here with $\mathfrak{T}(\alpha)$. Notice, we have to abandon *Optimal Path Trees* because turn restrictions may coerce optimal paths to have cycles, e.g. Fig. 1.

- road net graphs $\mathbf{G}= [V_G, E_G]$
  with traffic points (vertices) $V_G$ and the roads (edges) $E_G \subseteq V_G^2$ and we assume a given
- road cost function $\ell: E_G \rightarrow \boldsymbol{R}_+$
  denoting the effort $\ell(r)$ traversing road r (length, time).

*OPG* and *ROPG* can be seen as a bunch of overlapping optimal paths from all $x \in V_G$ to the target $\beta$ or from $\alpha$ to all $x \in V_G$:

$$OPG\ \overline{\mathfrak{T}}(\beta) = \bigcup_{x \in V_G} \{\mathbf{P}_G(x, \beta)\},\ \text{see} \rightarrow \text{in Fig. 2,}$$

$$OPG\ \mathfrak{T}(\alpha) = \bigcup_{x \in V_G} \{\mathbf{P}_G(\alpha, x)\}.$$

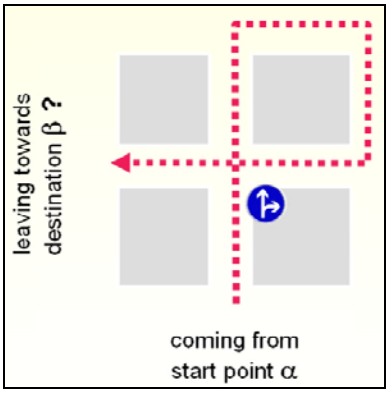



**Fig. 1** Optimal path with a cycle induced by turn restrictions

Concentrating on *ROPG*s, $\overline{\mathfrak{T}}(\beta)$ we introduce an

- edge potential function $\pi: E_G \rightarrow \boldsymbol{R}_+$
  that uniquely assigns to each road r= (x,y) the potential $\pi(r)$ (length or time) of an optimal path $\mathbf{P}_G(x, \beta)$ from x to target $\beta$ ensuring turn restrictions.
- Edge successor function $\sigma: E_G \rightarrow E(\overline{\mathfrak{T}}(\beta))$.
  uniquely assigns each road $r \in V_G$ a successor edge $\sigma(r) \in E(\overline{\mathfrak{T}}(\beta))$ via an optimal path to target $\beta$.
  Related to $\sigma$, there is an implicit
- start edge function $\zeta: V_G \rightarrow E(\overline{\mathfrak{T}}(\beta))$
  assigning each point $x \in V_G$ an outgoing road $\zeta(x) \in E(\overline{\mathfrak{T}}(\beta))$ that is part of an optimal path from x to $\beta$. E.g. $\overline{\mathfrak{T}}(\beta)$ In Fig. 2: $\zeta(5)= 11= (5,4)$. Obviously, $\zeta(x)$ can be called optimal road leaving x:

  $x \in V_G \Rightarrow \pi(\zeta(x)) = \min_{e \in E_G \| \{x\}} \{\pi(e)\}$ with $E_G \| \{x\}$ as the set of edges $E_G$ leaving x. It follows:

  $$e \in V_G \Rightarrow \sigma(e=(u,v)) = \zeta(v).$$

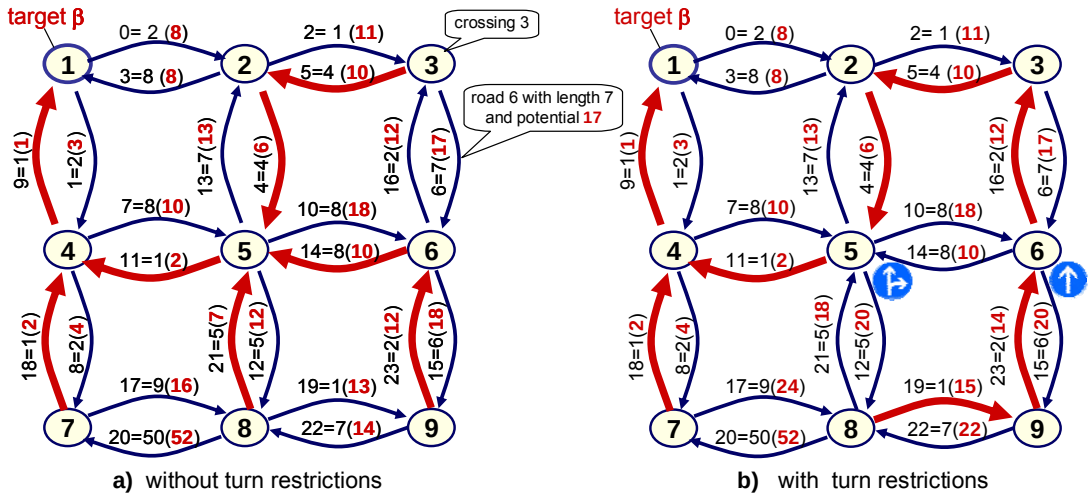
  An optimal path $\mathbf{P}_G(x, \beta)$ leads to $\beta$ via the edges

  $$(\zeta(x), \sigma(\zeta(x)), \sigma(\sigma(\zeta(x))), \ldots, (z, \beta))$$

  with start edge $\zeta(x)$ and the last edge $(z, \beta)$.

We refer to Fig. 2 b, where the optimal path $P_G(\alpha=8, \beta=1)$ observing turn restrictions is 8 units longer than the corresponding path without turn restrictions (Fig. 2 a). For Fig. 2 b) it holds e.g. that $\zeta(8)= 19$ = optimal road leaving crossing 8.
The road sequence of $P_G(\alpha=8, \beta=1)$, (Fig. 2 b), is $\zeta(8)=19$, $\sigma(19)= 23$, $\sigma(23)= 16$, $\sigma(16)= 5$, …$\sigma(11)= 9$.

We state already in advance: If an *ROPG* $\overline{\mathfrak{T}}(\beta)$ does exist and a driver has lost his previous optimal path coming to an unknown road r, he immediately has with $\overline{\mathfrak{T}}(\beta)$ the new successor road $\sigma(r)$ towards the target $\beta$!




**a)** without turn restrictions **b)** with turn restrictions

**Fig. 2** The influence of turn restrictions on the *Reverse Optimal Path Graph* ROPG $\overline{\mathfrak{T}}(\beta)$ (⟶).

## 2 Problem Formulation

We start from

- a digital road map given as connected, directed, and finite graph G= [ VG, EG ].
- a fixed start point $\alpha \in VG$,
- a fixed target $\beta \in VG$,
- the roads' lengths (or time). :: $EG \to R+$
- turn restrictions (see 3.1 ) $\overline{r}$ .

Let **H** ⊆ **G** be an arbitrary sub-graph in **G**, i.e. $V_H \subseteq V_G$ and $E_H \subseteq E_G \cap V_H^2$. Then we denote with

$C(H)= \sum_{r \in E_H} \ell(r)$ the cost of sub-graph **H**.

**REVERSE OPTIMAL PATH PROBLEM ENSURING TURN RESTRICTIONS:**

We look for sub-graph $H^* \subseteq G$ with cost

$$C(H^*)= \min_{\substack{\forall H \subseteq G \\ x \in V_G \Rightarrow \exists P_G(x,\beta) \subseteq H \\ H \text{ observes } \overline{r}}} \{C(H)\}.$$

We call $H^*= \overline{\mathfrak{T}}(\beta)$ *Reverse Optimal Path Graph* in **G** with respect to target β and turn restrictions $\overline{r}$ .
The formula expresses the following:

Among all sub-graphs **H** in **G** that contain optimal paths $P_G(x, \beta)$ in **G** for all $x \in V_G$ (i.e. $P_G(x, \beta) \subseteq H$) ensuring turn restrictions we look for that one that has minimum cost C(**H**).

### 2.1 Solution Prospect

$\overline{\mathfrak{T}}(\beta)$ uniquely assigns each vertex $x \in V_G$ an optimal path from x to β. Since $P_G(x, \beta) \subseteq \overline{\mathfrak{T}}(\beta)$ it holds $P_G(x, \beta)= P_{\overline{\mathfrak{T}}(\beta)}(x, \beta)$ ensuring turn restrictions.
The solution has to include the calculation of

- $\pi: E_G \to \boldsymbol{R}_+$
- $\sigma: E_G \to E(\overline{\mathfrak{T}}(\beta))$
- $\zeta: V_G \to E(\overline{\mathfrak{T}}(\beta))$ (implicitly)

For an arbitrary vertex $x \in V_G$ the *road* sequence

$$\zeta(x) \to \sigma(\zeta(x)) \to \sigma(\sigma(\zeta(x))) \to \ldots. \to (z, \beta)$$

has to lead via the edges of an optimal path $P_G(x, \beta) \subseteq \overline{\mathfrak{T}}(\beta)$ ensuring turn restrictions.

## 3 SOLUTION – ALGORITHM *A-1*

### 3.1 Turn Restrictions

Regarding a crossing $x \in V_G$ we define restriction as follows:

"if a driver comes via road r1 to crossing x then driving away via road r2 is not allowed!" I.e. the tupelo (r1, r2) is to assign to crossing x as restriction. We may write (r1, r2)$\in \overline{r}(x)$, Thus, we define the turn restrictions as follows:

Turn Restrictions

$$\overline{r}: V_G \to \mathfrak{P}(E_G^2)..$$

The power set $\mathfrak{P}(E_G^2)$ suffices only the exact notation to enable the assignment of several turn restrictions (tupelos) to one crossing. E.g. in Fig. 2, $\overline{r}(5)$= {(21,11), (21,12)}. The deployment is easy because the vertex structure is to enlarge only by a pointer to a restriction structure. The pointer is NULL if the crossing under consideration has no turn restrictions. Otherwise, the pointer refers to the first restriction given, e.g., with the simple structure called Sperr as follows:

```
struct   Sperr {  // e.g. assigned to vertex x ∈ VG
        int       edge1,
        int       edge2;
        Sperr *   p_Next;
}
```

### 3.2 Algorithm *A-1*

Description of ***A-1*** corresponding to Fig. 3:

**1** Set the potential of all edges to infinite and set each edge e itself as successor edge. Set the edge FIFO queue Q empty and mark e* to -1 meaning "the start point $\alpha$ hasn't been reached by the path exploration till now" (backwards from the targt $\beta$). Set the potential $\pi(e)$ of all edges e that directly end in $\beta$ and put them into the queue Q.

**2** Take an arbitrary edge L= (p, i) out from Q and remove it from there (at the begin: i = $\beta$).

**3** If Q is empty go to **5** (final treatment).

**4** Regard the vertex p of edge L= (p,i) and take all its incoming edges e= (j, p).
Is edge sequence (e, L) via vertex p a turn restriction then go to 2. Are the cost
c= potential of L + length of e
greater or equal than the potential of edge e then go to 2.
Set the potential of e to c.
Take $\sigma(L)$= L as the best successor edge for edge e towards target $\beta$.
Is the start point j= $\alpha$ (remember: reverse edge scanning) and is c < c* then assign the minimum potential edge e* with potential c* to $\alpha$.
Put edge e into queue Q.
Go to 2.

**5** If target $\alpha$ has not been reached then there is no path $\mathbf{P}_G(\alpha, \beta)$ from $\alpha$ to $\beta$ observing turn restrictions. Go to 7.

**6** The *ROPG* $\bar{\mathfrak{T}}(\beta)$ has been developed. The total function $\sigma$ is available and assigns each edge $e \in E_G$ a successor edge $\sigma(e)$ towards target $\beta$. End.

**7** Start point $\alpha$ has not been reached. This may happen if **G** is not connected or if false turn restrictions prevent building a connected path.

Now, a driver can use the *ROPG* $\bar{\mathfrak{T}}(\beta)$ without re-running ***A-1*** as long as the target $\beta$ remains unchanged:

a) The driver is on any road r:
$\sigma(r)$ is the road to optimally drive towards $\beta$.

b) The driver is situated on any crossing x:
$\zeta(x)$= is the road to optimally drive towards $\beta$.

So far an application really needs the total function $\zeta: V_G \rightarrow E_G$, the following statements might be introduced into block 6 of ***A-1***:

$\forall\, x \in V_G$: $\zeta(x)= e^*$ such that $\pi(e^*)= \min_{e \in E_G \| \{x\}} \{\pi(e)\}$. The overall time complexity remains unchanged.

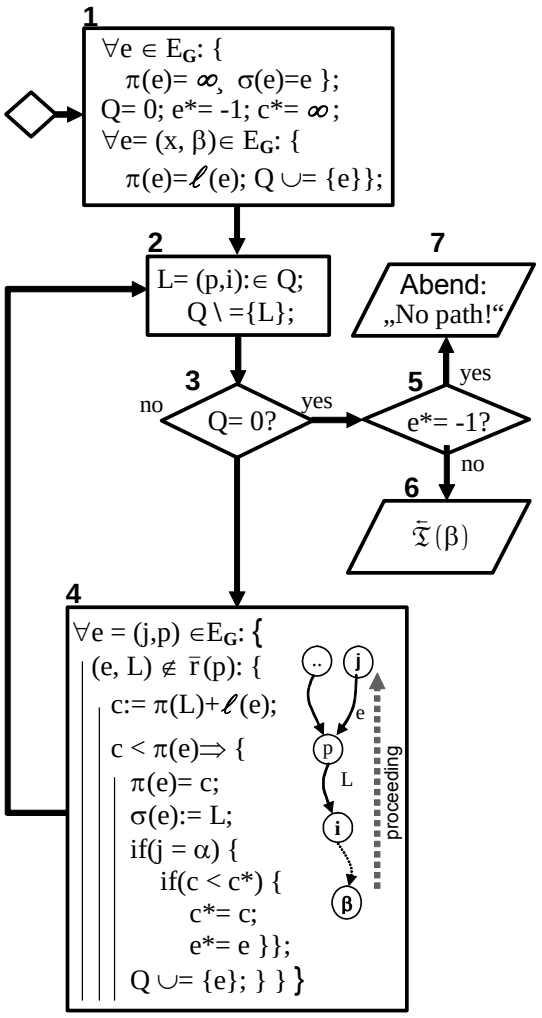



**Fig. 3** Algorithm *A-1* building an *ROPG* $\bar{\mathfrak{T}}(\beta)$ observing turn restrictions

### 3.3 Exactness and Complexity of *A-1*

***A-1*** is an exact algorithm delivering a solution with polynomial time effort $\boldsymbol{O}(m^2)$.

Exactness Proof: (proof by contradiction)
The critical point is the potential underbidding control in block 4 in ***A-1***. We assume that ***A-1*** calculates a path $P_{\alpha,\beta}$ from $\alpha$ to $\beta$ that is not an optimal one. That means that ***A-1*** hasn't discovered the (ore one of several) optimal path $\mathbf{P}_G(\alpha, \beta)$ ensuring turn restrictions. I.e. there is an edge e' within an undiscovered path $\mathbf{P}_G(x, \beta)$, whose potential $\pi(e')$ hasn't been updated during the path exploration of ***A-1***. This contradicts to block 3 and 4: Each edge so far improvable is reached by a temporary path wave started in the target $\beta$ and scanning the edges backwards (their reverse direction). If the graph is connected (it is, because $P_{\alpha,\beta}$ does exist) edge e' must be reached and it takes place the underbidding control improving its potential. Because e' is put into the queue Q (if a potential improvement occurs as assumed), e' is available as successor for further possible improvements applied to the incident edges e'. This contradicts the assumption that $\mathbf{P}_G(\alpha, \beta)$ was undiscovered by ***A-1***. ■

Time Complexity Proof:

We assign all edges to generations g1, g2, …. as follows.

- g1 is the set of the edges $E_G(\beta)$. Among these edges there is one $e_{g1} = (x_{g1}, \beta)$ with minimum potential = minimum length $\ell(e_{g1})$ coming into the queue Q. Clearly, There will be no other edge in the course of the optimization that can reach $v_{g1}$ (reverse direction!) with a smaller path length measured from the target backwards. Thus, this edge will not come again into Q. Only for g1 it holds: The other edges arriving $\beta$ also won't come into Q again because negative cycles cannot occur due to the prerequisite $\ell: E_G \rightarrow \boldsymbol{R}_+$ ( no further improvement of their potential possible).
- g2 are the edges that are successor edges of g1 coming into Q for the first time. It holds also here: there is an edge $e_{g2}$ whose potential (cumulative length through its predecessor edges) is smallest and will not come into the queue Q again.
- g3 to proceed in relation to g2 explained above, … and so on.

It follows that each generation has at least one edge whose potential cannot be improved and won't come again into Q. That means, that the theoretic maximum number of edges each generation steadily decreases at least by 1. Since m+(m-1)+(m-2)+ .. + 3+2+1= $\boldsymbol{O}(m^2)$, m=$|E|$.

Thus the worst case complexity is $\boldsymbol{O}(m^2)$. ■

Remark

The worst-case complexity $\boldsymbol{O}(m^2)$ feigns that ***A-1*** as so called *Label Correcting Algorithm (LC)*, [7], is slower than a comparable *Label Setting Algorithms (LS),* [7], having time complexity $\boldsymbol{O}(m \log m)$. Nevertheless, run time analyses show that LC algorithms outperform LS algorithm if all application cases come into consideration (target and start very fare remote as well as near together, see 5.2).

## 4 Known Solution Approaches Tackling Turn Restrictions

Kirby and Pots [6] solved the turn restriction problem by replacing vertices having turn restrictions with directed sub-graph as shown in Fig. 4. („expanded networks")

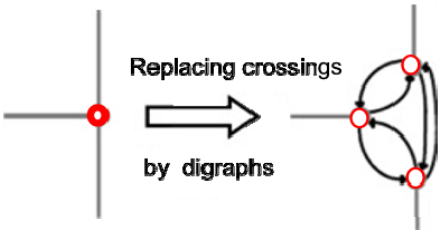



**Fig. 4** Crossing replacement by sub-graphs

In contrast to ***A-1***, this method induces a heavy increase of inefficient vertices and edges for the digital road mapgraph **G** and declines the real time performance.

Jiang et al. [5] proposed an interlinked („link-based") data structure used as vertex connection table „NodeLink" together with an edge connection table „LinkLink" that describes the turn restriction, Fig. 5.

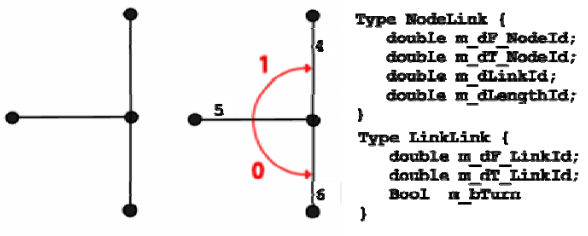
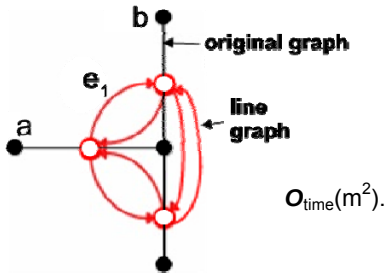



**Fig. 5** Crossings enlarged with connection structures

Algorithm *A-1* doesn't need structures like above. Instead it uses a very simple assignment of turn restrictions to vertices $V_G$ so far necessary.

Cadwell [2], Anez et al. [1], and Winter [8] introduced the concept of „line graph": Roads between crossings are transferred to vertices and crossings are transferred to roads, Fig. 6. The Line graph completely repla ces the original graph **G**. Therefore, run time performance is quite better than that of the approaches des cribed above. Road lengths are to store as vertex cost




$\boldsymbol{O}_{time}(m^2)$.

**Fig. 6** Graph inverting vertices with edges

and turn restrictions are to store as edge cost.The conversion of graph **G** into a new graph like above is always necessary as far as a graph change is necessary. Algorithm ***A-1*** doesn't need such a conversion and is as efficient as the the algorithm of [2].

Flinsenberg [4] proposed an edge-queued algorithm similar to the known A* algorithm. It regards estimated cost from the current edge processed to the target. Flinsenberg takes into account processing all incident road pairs with additional turn cost: Turn cost are set very high to suppress the path exploration via forbidden neighbour roads.

Against it, ***A-1*** considers only those incident road pairs that are really confronted with turn restrictions and an additional introduction of cost to prohibit forbidden turns are not necessary.

## 5 PERFORMANCE ANALYSIS

### 5.1 Test Graphs

We used random grid graphs $\mathbf{G}= [V_G, E_G]$ , $n= |V_G|$, $m= |E_G|$, $n \le 210.000$ (vertices) and $m \le 838.176$ (edges), with road length function $\ell: E_G \rightarrow \boldsymbol{R}_+$ and $0 \le \ell(r) \le 40$ for all $r \in E_G$, **Fig. 7**. The graphs are directed. Each inner vertex $v \in V_G$ has four incoming edges and four outgoing edges. The graphs are compact and shaped as squares with rows and columns corresponding to about $\sqrt{n}$. The correctness of the results has been proved for smaller graphs as they and their optimal paths with turn restrictions were displayed on the screen.

### 5.2 Comparative Algorithms Tackling Turn Restrictions

We consider only those algorithms that explore the paths using an edge queuing strategy (suffix _E). Not vertices but edges are put into the queue Q. This strategy enables algorithms to implement turn restrictions as described in 3.1. ***LS_E_OPG*** and ***LC_E_OPG*** return an edge predecessor list not (as ***A-1***) a successor list. It emerges an Optimal Path Graph *OPG* $\mathfrak{T}(\alpha)$ that can be denoted only then as *Optimal Path Tree* if turn restrictions are not observed.

Algorithm ***LS_E_OPG***
is an edge-queuing ***L***abel-***S***etting algorithm, i.e. the path exploration finishs if the target $\beta$ has been reached. Queue Q is a priority queue organized as a binary heap to efficiently find the minimum potential edge. The path exploration begins at the start point $\alpha$ and ends if a minimum potential edge e= (z, $\beta$) has been fetched from the queue Q ($\beta$ reached). An *OPG* emerges $\mathfrak{T}(\alpha)$. The run time decreases (increases) the nearer (more remote (edge number)) start and target are situated. It follows, that there is an essential performance difference dependent on the remoteness of $\alpha$ and $\beta$. **Fig. 7** shows the great difference between *LS_E_OPG*-max. (great many edges between $\alpha$ and $\beta$) and *LS_E_OPG*-min ($\alpha$ and $\beta$ within the near vicinity) .
Worst case time effort: $\boldsymbol{O}(m \log m)$

Algorithm ***LC_E_OPG***
is an edge-queuing ***L***abel-Correcting algorithm working with a queue Q being organized as a FIFO queue (first in – first out). The path exploration begins at the start point $\alpha$ and ends if no edges can be improved as to their potential. It emerges an *OPG* $\mathfrak{T}(\alpha)$. In contrast to *LS_E_OPG* that can finish if the target has been reached, *LC_E_OPG* needs more time effort even if $\alpha$ and $\beta$ are situated quite near together. Nevertheless, LC-E outperforms *LS_E_OPG*.
Worst case time effort: $\boldsymbol{O}(m^2)$

Algorithm ***A-1***
Like *LC_E_OPG*, ***A-1*** is an edge-queuing ***L***abel-Correcting algorithm working with a queue Q being organized as a FIFO queue. The path exploration begins at the target $\beta$and proceeds towards the start using the edges' reverse direction. The performance is comparable with that of *LC_E_OPG* but it emerges an *ROPG* $\overline{\mathfrak{T}}(\beta)$. Thus, the very attractive quality "ubiquitous path information" provided by $\sigma$ is available: Each edge $e \in E_G$ has been assigned the successor edge towards the target (via an optimal path using the following successor edges), i.e. a driver can deliberately make detours without a new calculation of the optimal path (*ROPG*). He remains always informed through the edges best successor assigned to by $\sigma$.
Worst case time effort: $\boldsymbol{O}(m^2)$

### 5.3 Evaluation

Algorithm ***A-1*** fulfils the following requirements:

1. ***A-1*** as LC-algorithm is competitive in time and memory compared to algorithm that ensure turn restrictions. ☑
2. The path description of ***A-1*** has to be unique although the vertex predecessor assignment may not be unique. ☑
3. Insertion and deletion of traffic regulations mustn't harm the graphs' vertex and edge table . ☑
4. Traffic regulations have to observe dynamic storage allocation for real-time changes.
   Their internal data representation should abandon concepts based on orientation. ☑

Surprisingly, $\boldsymbol{O}(m \log m)$-algorithm *LS_E_OPG* (average run time, **Fig. 7**, cannot compete with ***A-1*** and *LC_E_OPG* although the latter ones have a worst case time complexity $\boldsymbol{O}(m^2)$. The priority queue Q in *LS_E_OPG* requires more maintenance effort than the FIFO-queue of ***A-1*** or *LC_E_OPG*. This disadvantage of *LS_E_OPG* increases with the circumference of the current path exploration wave: ***A-1*** and *LC_E_OPG* need less queue maintenance effort than *LS_E_OPG*, but their number of queue operations is always very high independent from the $\alpha$--$\beta$- remoteness.

## 6 SUMMARY

***A-1*** outclasses comparative algorithms under consideration observing turn restrictions.

1) The proposed algorithm ***A-1*** outperforms LC_E_OPG and LS_E_OPG as to the trade-in value namely the ubiquitous path information related to $\sigma$.
2) Algorithm ***A-1*** outperforms LS_E_OPG with respect to the run time effort.
3) Regarding the simplicity of implementing turn restrictions proposed here and the arguments of 1) and 2), Algorithm ***A-1*** is a most suitable one for navigation applications.

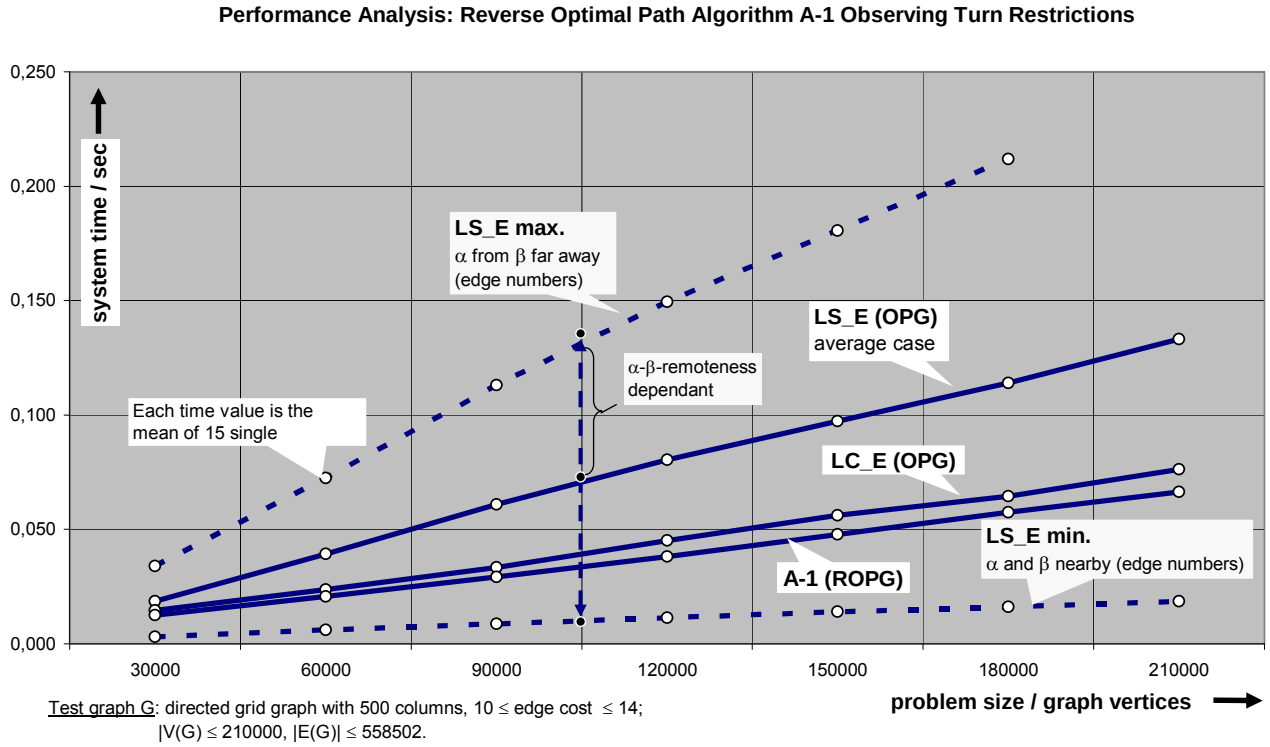



**Fig. 7 Run time effort of comparative algorithms efficiently tackle turn restrictions**

## 7 REFERENCES

[1] Anez, J., T. de la Barra, and B. Perez (1996), 'Dual Graph Representation of Transport Networks'. Transportation Research 30(3), p. 209–216.

[2] Caldwell, T. (1961), 'On finding minimum routes in a network with turn penalties'. Communications of the ACM 4(2), p. 107–108.

[3] Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs'. Numerische Mathematik 1, p. 269–271.

[4] Flinsenberg, Ingrid C.M. Route planning algorithms for car navigation / by Ingrid C.M. Flinsenberg. - Eindhoven: Technische University Eindhoven, 2004. Proefschrift. - ISBN 90-386-0902-7

[5] Jiang J., Han H. and Chen J. (2002), 'Modeling turning restrictions in traffic network for vehicle navigation system'. IAPRS, Vol. XXXIV, part 4.

[6] Kirby R.F and Potts R.B. (1969), 'The minimum route problem for networks with turn penalties and prohibitions'. Transport Research 3, p. 397-408.

[7] Richter P.: Efficient Shortest Path Algorithms for Road Traffic Optimization, 27th International Symposium on Advanced Transportation Applications (ISATA): Dedicated Conference on Advanced Transport Telematics, ISBN 0947719652, Aachen, 31.10.-4.11.1994, 79-86

[8] Winter, S. (2002), 'Modeling Costs in Turns of Route Planning'. GeoInformatica 6(4), p. 345-360.