A Lower Bound for the Quickest Path Problem

Gianpaolo Ghiani[1] Emanuela Guerriero[1]

[1]Department of Engineering for Innovation
University of Salento (Italy)

Route 2014

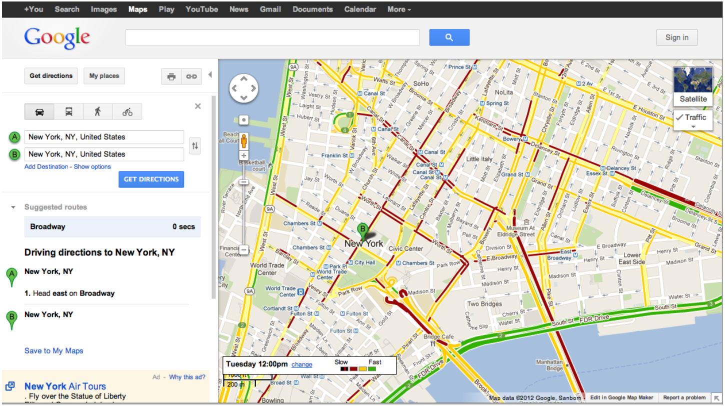
Outline

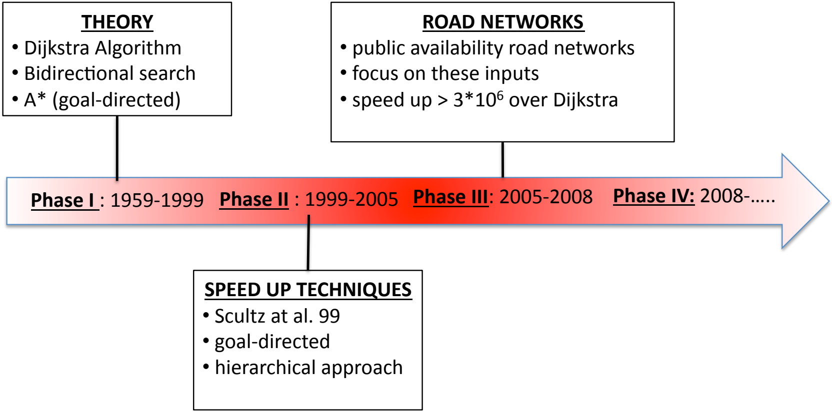
1. Motivation
   - The Quickest Path Problem
2. Our Contribution
3. The Quickest Path Problem
4. A lower bound
5. Computational Results

The Quickest Path Problem

The determination of shortest or quickest paths on road networks is the basic ingredient of driving direction computation as well as of logistic planning and traffic simulation.

Typical application setting





In-depth reviews: Delling et al (2009) and Sommer (2012)

Our Contribution

We can define $\underline{z}(i, d)$: a lower bound on the minimum travel time from node i to target d. This bound can be used in A^* algorithm a best first search

$$h(i, d) = t_{si} + \underline{z}(i, d)$$

The Dijkstra's algorithm can be seen as an A* procedure with a null lower bound.

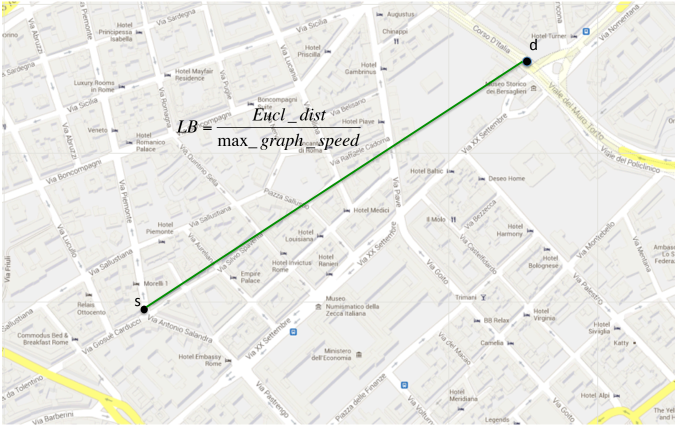
Our Contribution

We can define $\underline{z}(i, d)$: a lower bound on the minimum travel time from node i to target d. This bound can be used in A^* algorithm a best first search

$$h(i, d) = t_{si} + \underline{z}(i, d)$$

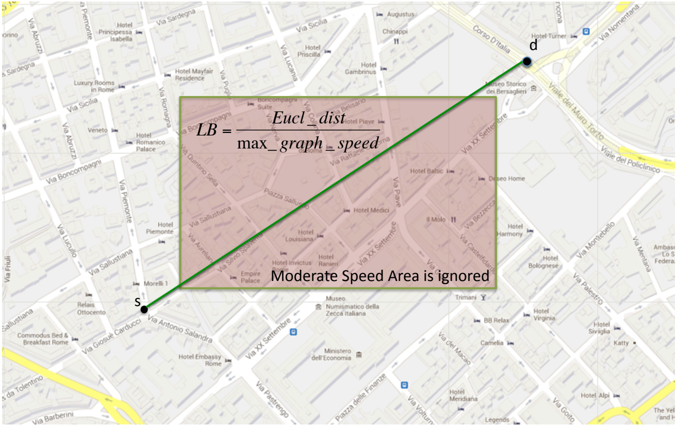
In this research work we focus on a new lower bound $\underline{z}(i, d)$ for the Quickest Path Problem (QPP).

In a nutshell...



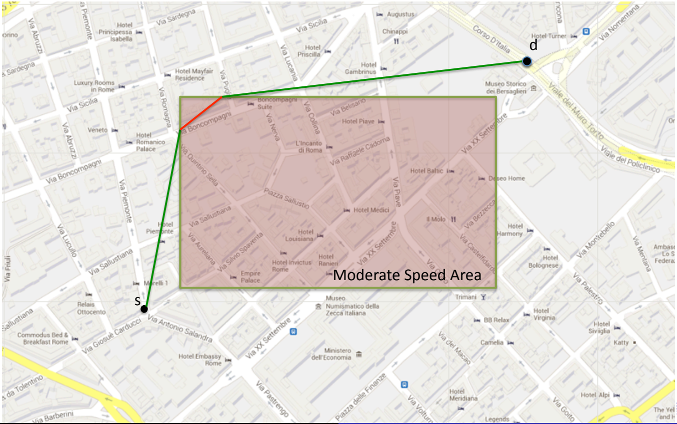
$$LB = \frac{Eucl\_dist}{\max\_graph\_speed}$$

In a nutshell...



In a nutshell...

We propose a new lower bound that takes into account moderate speed areas



Our Contribution

- The proposed new lower bound is 3.88 times faster and more effective then the trivial lower bound
- We embeded the new lower bounding procedure into a unidirectional A*
- We test the resulting algorithm on some metropolitan areas

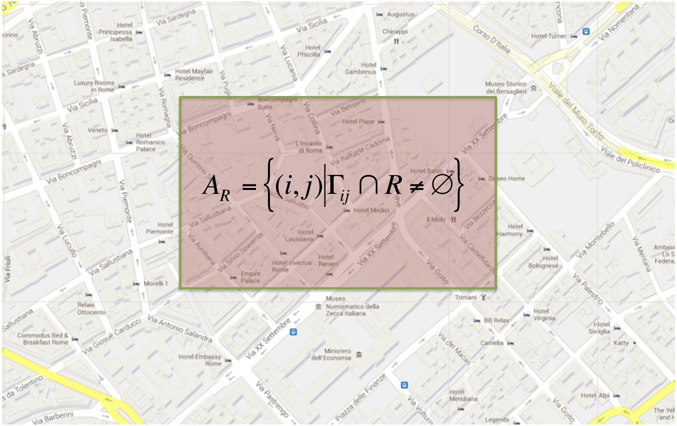
The Quickest Path Problem

- Given a directed graph G(V,A). With each arch $(i, j) \in A$ are associated
  - a non negative constant travel time t_{ij},
  - a positive length l_{ij} and a speed $\nu_{ij} = l_{ij}/t_{ij}$
  - a curve Γ_{ij} (i.e. the *geometry* associated to the road)
- given a source node $s \in V$ and a destination node $d \in V$
- The *traversal time* z_p of a path $p = (s = i_1, i_2, \dots, i_k = d))$ is $\sum_{h=1}^{k} t_{i_h i_{h+1}}$

We aim to determine a path p such that z_p is minimum. Let z^* be such minimum duration.

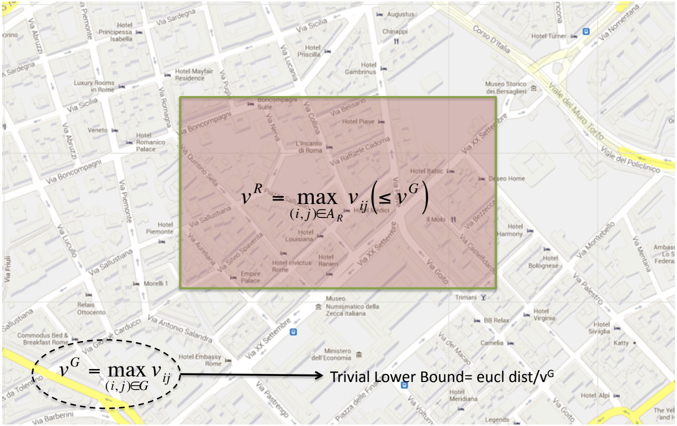
Problem Relaxation

Problem Relaxation



$$A_R = \left\{ (i,j) \middle| \Gamma_{ij} \cap R \neq \emptyset \right\}$$

Problem Relaxation



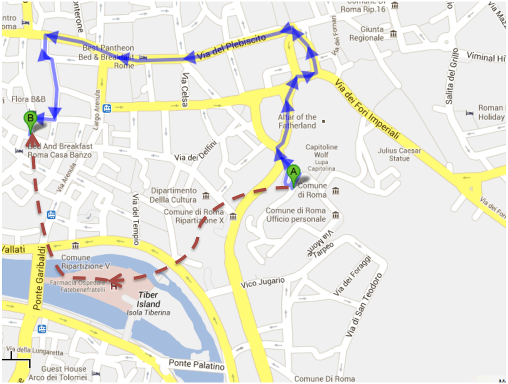
$$v^R = \max_{(i,j)\in A_R} v_{ij} \left(\leq v^G\right)$$

$$v^G = \max_{(i,j)\in G} v_{ij}$$

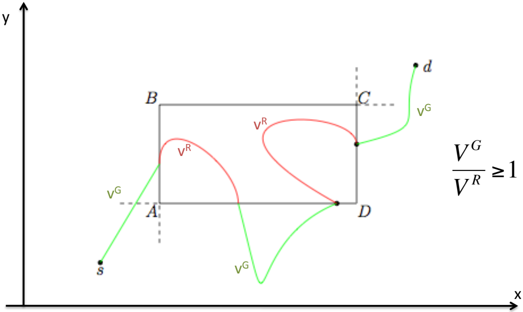
Trivial Lower Bound= eucl dist/v^G

Problem Relaxation

Given two nodes $i, j \in V$ the path can be any curve in the plane.



The proposed Lower Bound



$$\frac{V^G}{V^R} \geq 1$$

Subcurves in R and $\Gamma \setminus R$ are traversed at speeds ν^R and ν^G, respectively.

The proposed Lower Bound

Theorem. A least duration curve from s to d in the plane is a polyline $s - u - v - d$, where $u \in \overline{AB} \cup \overline{AD}$ and $v \in \overline{BC} \cup \overline{CD}$.

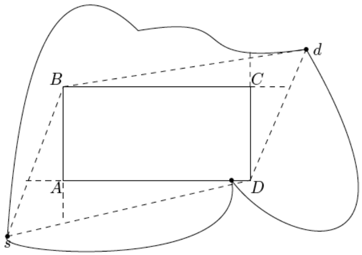


Figure 1: First Case

The proposed Lower Bound

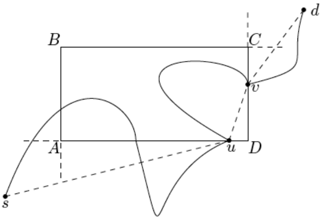


Figure 2: Second Case

- u is the last intersection point with $\overline{AB} \cup \overline{AD}$ *saw from s*
- v is the first intersection point with $\overline{BC} \cup \overline{CD}$ *saw from d* .

The least duration curves: four types

$$T(u,v) = \frac{dist(s,u)}{\nu^G} + \frac{dist(u,v)}{\nu^R} + \frac{dist(v,d)}{\nu^G}$$

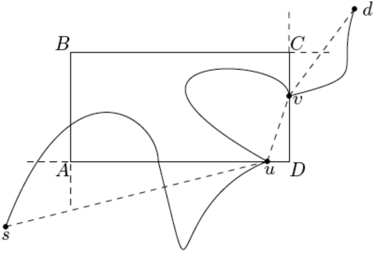


Figure 2: Second Case

The least duration curves: four types

$$\min_{\substack{u \in \overline{AB} \\ v \in \overline{BC}}} T(u,v)$$

$$\min_{\substack{u \in \overline{AD} \\ v \in \overline{BC}}} T(u,v)$$

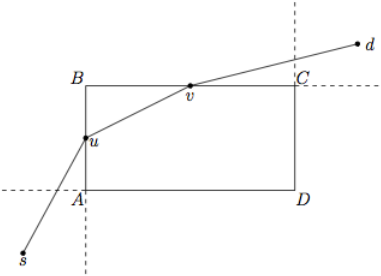


Figure 3: Path Type 1

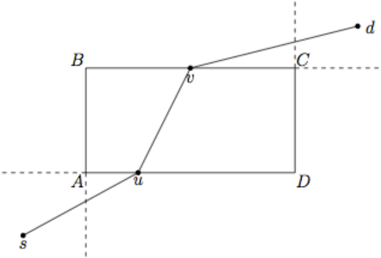


Figure 4: Path Type 2

The least duration curves: four types

$$\min_{\substack{u \in \overline{AD} \\ v \in \overline{CD}}} T(u, v)$$

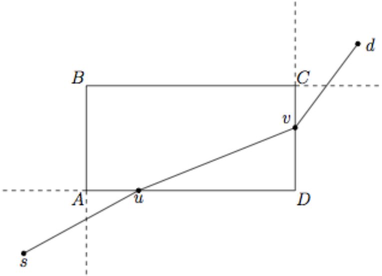


Figure 5: Path Type 3

$$\min_{\substack{u \in \overline{AB} \\ v \in \overline{CD}}} T(u, v)$$

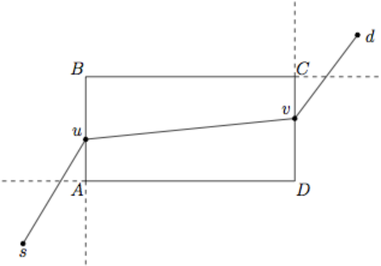


Figure 6: Path Type 4

The Lower Bound

A lower bound $\underline{z}$ can be computed by solving the following four optimization problems:

$$\min_{\substack{u \in \overline{AB} \\ v \in \overline{BC}}} T(u, v) \qquad \min_{\substack{u \in \overline{AD} \\ v \in \overline{BC}}} T(u, v)$$

$$\min_{\substack{u \in \overline{AD} \\ v \in \overline{CD}}} T(u, v) \qquad \min_{\substack{u \in \overline{AB} \\ v \in \overline{CD}}} T(u, v)$$

and then taking the minimum.

Type 1 Path

Two-variable non-linear box-constrained optimization problems (time consuming iterative methods)

$$\min_{\substack{u \in \overline{AB} \\ v \in \overline{BC}}} \frac{dist(s,u)}{\nu^G} + \frac{dist(u,v)}{\nu^R} + \frac{dist(v,d)}{\nu^G}$$

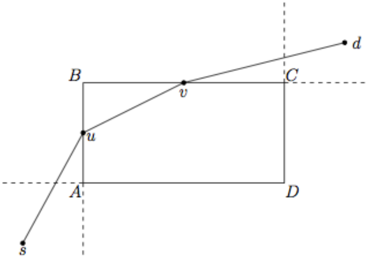


Figure 3: Path Type 1

Type 1 Path: a relaxation

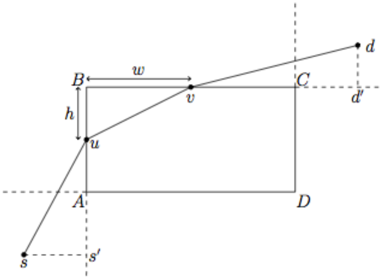


Figure 3: Type 1 path

$$\underline{z}^1 = \min_{\substack{0 \leq w \leq W \\ 0 \leq h \leq H}} \frac{y_B - y_s - h}{\nu^G} + \frac{\sqrt{h^2 + w^2}}{\nu^R} + \frac{x_d - x_B - w}{\nu^G}$$

Type 1 Path: a relaxation

$$\alpha = \frac{\nu_G}{\nu_R} > 1$$

Algorithm 1 Computing an optimal solution (w^*, h^*) for the Path 1 sub-problem

if $(\alpha \geq \sqrt{2})$ **then**
  $w^* = 0;$
  $h^* = 0;$
else
  $w^* = \min(\frac{H}{\sqrt{\alpha^2-1}}, W);$
  $h^* = H;$
end if

$$\underline{z}^1 = \min_{\substack{0 \leq w \leq W \\ 0 \leq h \leq H}} \frac{y_B - y_s - h}{\nu^G} + \frac{\sqrt{h^2 + w^2}}{\nu^R} + \frac{x_d - x_B - w}{\nu^G}$$

$$\alpha = \frac{\nu_G}{\nu_R} > 1$$

Algorithm 1 Computing an optimal solution (w^*, h^*) for the Path 1 subproblem

if $(\alpha \geq \sqrt{2})$ **then**
  $w^* = 0;$
  $h^* = 0;$
else
  $w^* = \min(\frac{H}{\sqrt{\alpha^2 - 1}}, W);$
  $h^* = H;$
end if

Theorem

The solution provided by Algorithm 1 is optimal for the Type 1 path subproblem.

Theorem

The solution provided by Algorithm 1 is optimal for the Type 1 path subproblem.

Proof Sketch

We prove that w^* and h^* satisfy the KKT conditions for the Type 1 path optimization subproblem

Implementation Issue

Type 1 Path is suitable for a preprocessing phase

$$\overline{z}^1 = \overline{z}_a^1 + \left(\frac{y_s + x_d}{\nu^G}\right); \tag{1}$$

$$\overline{z}_a^1 = \left(\frac{y_B - h_1^*}{\nu^G} + \frac{\sqrt{(h_1^*)^2 + (w_1^*)^2}}{\nu^R} + \frac{-x_B - w_1^*}{\nu^G}\right); \tag{2}$$

where $\overline{z}_a^1$ does not depend on s and d

Type 1 Path - Type 3 Path

A Type 3 problem is equivalent to a Type 1 problem where x_i and y_i are swapped for $i \in \{s, d, A, B, C, D\}$.

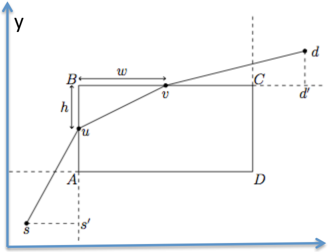


Figure 3: Type 1 path

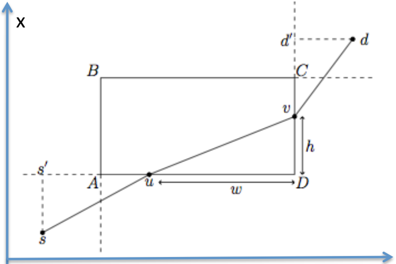
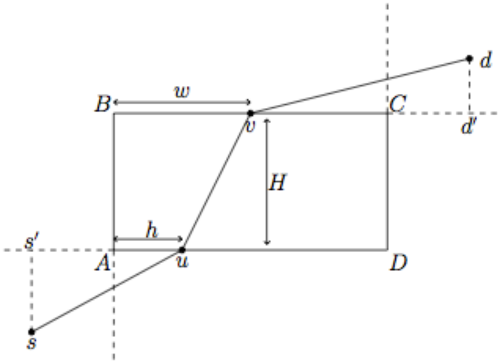


Figure 3: Path Type 3

Type 2 Path: a relaxation



$$\underline{z}^2 = \min_{\substack{0 \le w \le W \\ 0 \le h \le W}} \frac{x_A - x_s + h}{\nu^G} + \frac{\sqrt{H^2 + (w-h)^2}}{\nu^R} + \frac{x_d - x_B - w}{\nu^G} \quad (3)$$

$$\alpha = \frac{\nu_G}{\nu_R} > 1$$

Algorithm 2 Computing an optimal solution (w^*, h^*) for the Path 2 subproblem

$h^* = 0;$
$w^* = \frac{H}{\sqrt{\alpha^2 - 1}};$
if $(w^* > W)$ **then**
  $w^* = W$
end if

Theorem

The solution provided by Algorithm 2 is optimal for the Type 2 path subproblem.

Implementation Issue

Type 2 Path is suitable for a preprocessing phase

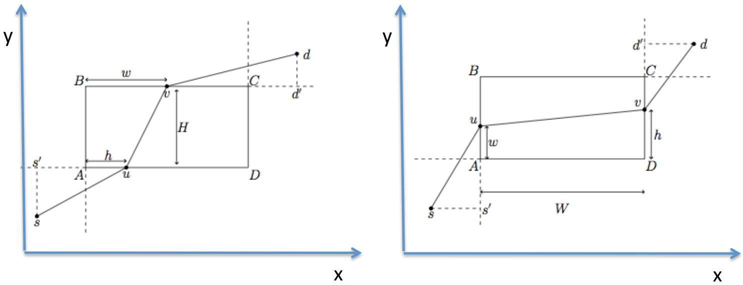
$$\overline{z}^2 = \overline{z}_a^2 + \left(\frac{-x_s + x_d}{\nu G} \right); \tag{4}$$

$$\overline{z}_a^2 = \left(\frac{x_A + h_2^*}{\nu G} + \frac{\sqrt{H^2 + (w_2^* - h_2^*)^2}}{\nu R} + \frac{-x_B - w_2^*}{\nu G} \right); \tag{5}$$

where $\overline{z}_a^2$ does not depend on s and d

Type 2 Path - Type 4 Path

A Type 4 problem is equivalent to a Type 2 problem where x_i and y_i are swapped for $i \in \{s, d, A, B, C, D\}$.



The Lower Bounding Algorithm

$$\overline{z}^1 = \overline{z}_a^1 + \left(\frac{y_s + x_d}{\nu G}\right);$$

$$\overline{z}^2 = \overline{z}_a^2 + \left(\frac{-x_s + x_d}{\nu G}\right);$$

$$\overline{z}^3 = \overline{z}_a^3 + \left(\frac{x_s + y_d}{\nu G}\right);$$

$$\overline{z}^4 = \overline{z}_a^4 + \left(\frac{-y_s + y_d}{\nu G}\right);$$

$$\underline{z} = min\{\underline{z}^1, \underline{z}^2, \underline{z}^3, \underline{z}^4\}$$

Computational Results

The lower bounding procedures were coded in C++ and embedded into a unidirectional A* algorithm implementation of Boost Graph Library.

The codes were run on a PC with a 2.53-GHz Intel Core 2 Duo processor and 4 GB of memory.

Computational Results

We made use of three road networks of large European metropolitan area (OpenStreetMap)

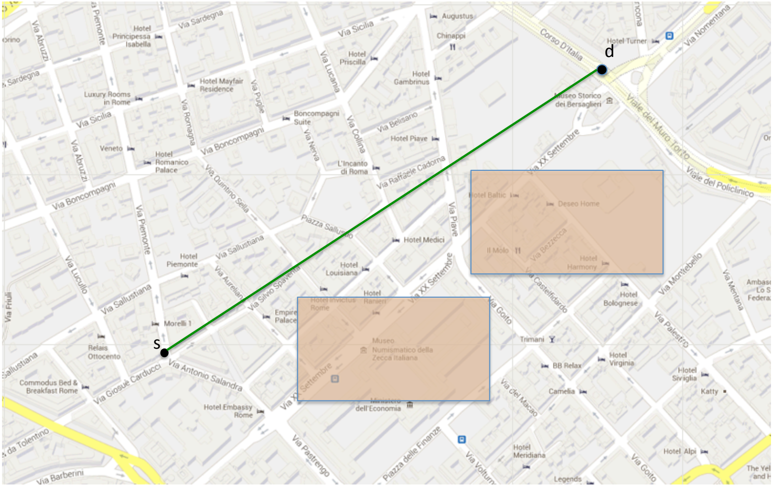
- Paris about 113,000 arcs
- Madrid about 96,000 arcs
- Rome about 52,000 arcs

For each graph we identified n_R (possibly overlapping) rectangles with ν^R equal to 50 km/h.

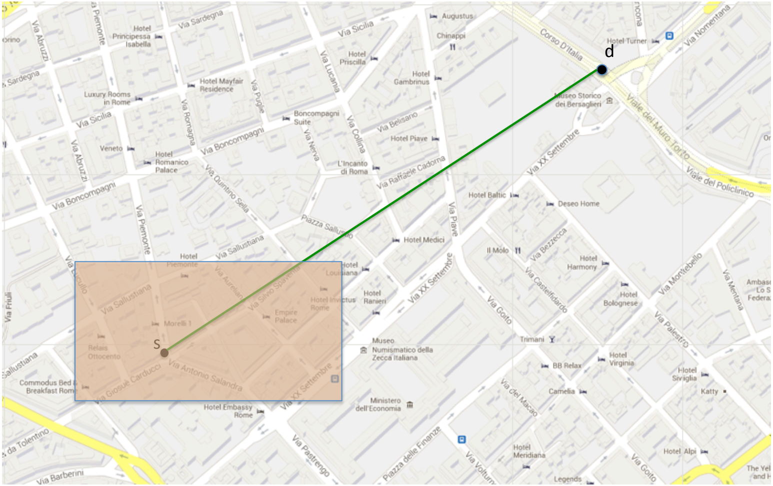
We randomly generated 1500 origin-destination pairs

- Type *a*: neither s or d are located inside a rectangle; moreover the segment $s - d$ does not intercept any rectangle;
- Type *b*: either s or d (but not both) are located inside a rectangle;
- Type *c*: s and d are located inside different rectangles;
- Type *d*: s and d are located inside the same rectangle;
- Type *e*: neither s or d are located inside a rectangle; however, the segment $s - d$ intercepts at least a rectangle.

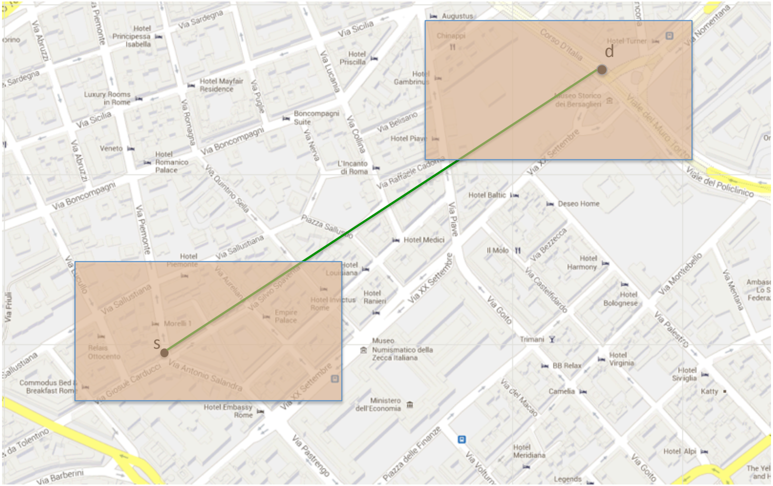
Type a: 300 randomly generated origin-destination pairs



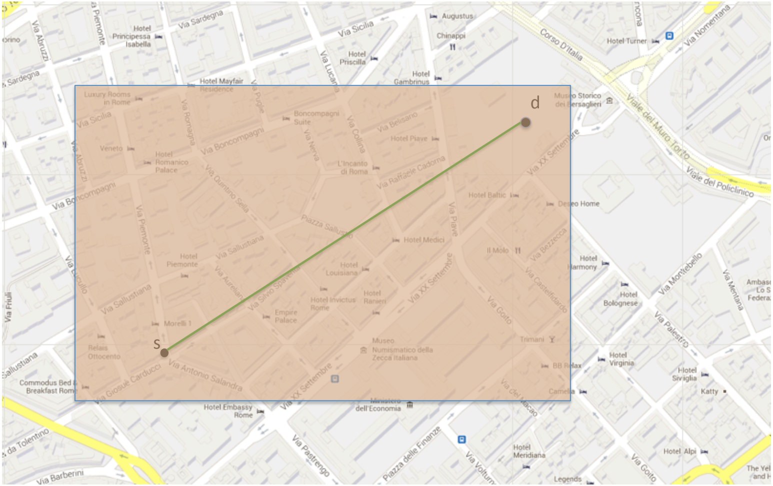
Type b: 300 randomly generated origin-destination pairs



Type c: 300 randomly generated origin-destination pairs



Type d: 300 randomly generated origin-destination pairs



Type e: 300 randomly generated origin-destination pairs

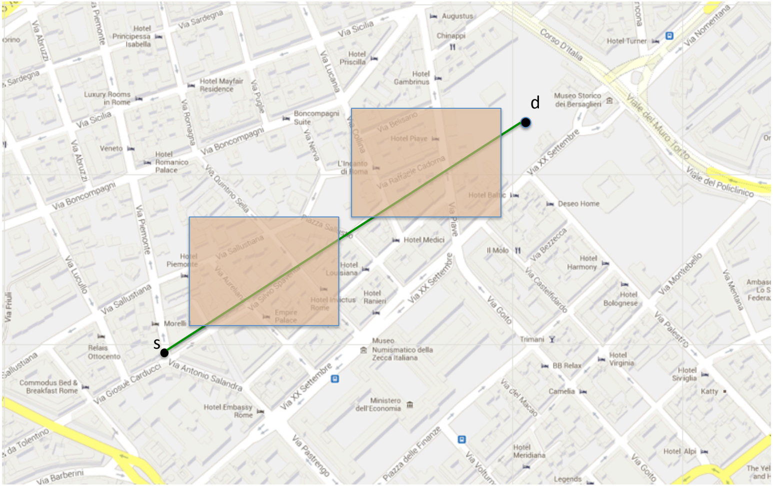


Table 2: Computational results on the Rome instances : $|A| = 52257$ and $n_R = 10$

v^G	α	Type	EUCLIDEAN LOWER BOUND			NEW LOWER BOUND			DEVIATIONS	
			$Time_E$ [ms]	δ_E	LB_E/z^*	$Time$ [ms]	δ	LB/z^*	Δ_V(%)	$\Delta Time$(%)
130	2.6	a	1.772	0.147	0.295	1.549	0.144	0.295	0.470	12.475
		b	6.864	0.556	0.311	5.893	0.552	0.325	3.242	14.913
		c	5.194	0.690	0.298	4.384	0.688	0.321	4.851	16.328
		d	1.983	0.867	0.263	1.572	0.870	0.331	12.352	22.924
		e	6.547	0.387	0.392	5.653	0.382	0.396	2.116	13.922
AVERAGE			4.472	0.530	0.312	3.810	0.527	0.334	4.650	16.151
90	1.8	a	1.517	0.124	0.404	1.331	0.121	0.404	0.534	12.008
		b	5.456	0.592	0.438	4.637	0.585	0.458	5.461	16.366
		c	4.133	0.730	0.428	3.405	0.725	0.461	8.103	18.703
		d	1.524	0.898	0.380	1.124	0.906	0.479	19.453	28.745
		e	6.412	0.438	0.496	5.538	0.431	0.500	2.702	13.926
AVERAGE			3.808	0.557	0.429	3.207	0.554	0.460	7.324	18.014
80	1.6	a	1.443	0.117	0.445	1.179	0.114	0.445	0.517	18.087
		b	4.944	0.604	0.490	3.884	0.594	0.512	6.635	23.124
		c	3.729	0.743	0.481	2.812	0.737	0.517	9.907	25.818
		d	1.366	0.910	0.428	0.902	0.919	0.539	22.632	36.296
		e	6.318	0.456	0.536	5.069	0.449	0.540	2.925	20.069
AVERAGE			3.560	0.566	0.476	2.769	0.562	0.511	8.611	24.751
70	1.4	a	1.338	0.108	0.497	1.138	0.105	0.497	0.534	14.741
		b	4.258	0.616	0.556	3.422	0.602	0.582	5.461	21.932
		c	3.173	0.758	0.550	2.403	0.753	0.592	8.103	25.792
		d	1.161	0.923	0.489	0.748	0.936	0.612	19.453	37.850
		e	5.907	0.474	0.588	4.919	0.467	0.593	2.702	17.053
AVERAGE			3.168	0.576	0.536	2.526	0.573	0.575	10.788	23.571

Table 3: Computational results on the Paris instances : $|A| = 113649$ and $n_R = 6$

v^G	α	Type	EUCLIDEAN LOWER BOUND			NEW LOWER BOUND			DEVIATIONS	
			$Time_E$ $[ms]$	δ_E	LB_E/z^*	$Time$ $[ms]$	δ	LB/z^*	$\Delta_V(\%)$	$\Delta Time(\%)$
130	2.6	a	9.027	0.076	0.357	8.524	0.076	0.357	0.076	5.561
		b	12.173	0.302	0.352	11.420	0.299	0.357	1.518	6.924
		c	5.898	0.499	0.323	5.434	0.498	0.333	3.378	8.681
		d	1.422	0.838	0.288	1.190	0.847	0.360	14.944	19.613
		e	20.620	0.224	0.396	19.449	0.223	0.397	0.424	5.890
AVERAGE			9.828	0.388	0.343	9.205	0.388	0.361	4.107	9.370
90	1.8	a	6.474	0.050	0.503	6.160	0.050	0.503	0.076	4.832
		b	8.914	0.326	0.501	8.364	0.318	0.508	1.518	7.444
		c	4.299	0.544	0.466	3.892	0.541	0.481	3.378	10.777
		d	1.054	0.890	0.416	0.814	0.905	0.521	14.944	26.917
		e	15.136	0.241	0.559	14.359	0.238	0.560	0.424	5.543
AVERAGE			7.175	0.410	0.489	6.718	0.410	0.515	4.107	11.161
80	1.6	a	5.895	0.044	0.551	5.577	0.044	0.551	0.080	5.368
		b	7.848	0.332	0.555	7.285	0.323	0.563	3.758	8.852
		c	3.738	0.560	0.524	3.312	0.556	0.541	8.151	13.012
		d	0.926	0.909	0.468	0.680	0.924	0.586	27.685	31.513
		e	13.468	0.247	0.612	12.688	0.244	0.613	1.100	6.335
AVERAGE			6.375	0.418	0.542	5.908	0.418	0.571	8.227	13.085
70	1.4	a	5.436	0.040	0.608	5.235	0.039	0.608	0.107	3.679
		b	7.014	0.344	0.617	6.574	0.333	0.627	5.168	8.559
		c	3.232	0.585	0.594	2.835	0.584	0.613	11.549	14.712
		d	0.806	0.930	0.535	0.559	0.954	0.666	34.650	36.986
		e	12.665	0.264	0.661	12.140	0.261	0.662	1.245	4.776
AVERAGE			5.831	0.433	0.603	5.469	0.434	0.635	10.636	13.831

Table 4: Computational results on the Madrid instances : $|A| = 96757$ and $n_R = 10$

v^G	α	Type	EUCLIDEAN LOWER BOUND			NEW LOWER BOUND			DEVIATIONS	
			$Time_E$ $[ms]$	δ_E	LB_E/z^*	$Time$ $[ms]$	δ	LB/z^*	$\Delta_V(\%)$	$\Delta Time(\%)$
130	2.6	a	6.287	0.098	0.445	6.015	0.098	0.445	0.087	4.359
		b	7.561	0.294	0.436	7.170	0.290	0.441	1.617	5.824
		c	5.721	0.453	0.380	5.366	0.452	0.391	3.237	7.375
		d	1.803	0.815	0.311	1.606	0.826	0.386	14.228	17.896
		e	12.199	0.227	0.493	11.653	0.226	0.494	0.420	4.678
AVERAGE			6.714	0.378	0.413	6.362	0.378	0.431	3.970	8.077
90	1.8	a	5.262	0.084	0.544	4.997	0.084	0.544	0.075	5.080
		b	6.123	0.313	0.551	5.709	0.306	0.556	2.767	7.636
		c	4.797	0.490	0.494	4.420	0.487	0.508	4.968	9.728
		d	1.598	0.856	0.414	1.364	0.873	0.515	20.819	24.784
		e	10.196	0.243	0.606	9.650	0.241	0.608	0.746	5.717
AVERAGE			5.595	0.397	0.522	5.228	0.398	0.546	5.952	10.663
80	1.6	a	4.907	0.081	0.577	4.608	0.081	0.577	0.080	6.160
		b	5.569	0.321	0.594	5.110	0.312	0.599	3.517	9.388
		c	4.385	0.508	0.544	3.978	0.505	0.559	5.933	11.656
		d	1.468	0.876	0.466	1.219	0.897	0.579	24.857	29.429
		e	9.609	0.254	0.640	8.987	0.251	0.642	0.811	6.846
AVERAGE			5.188	0.408	0.564	4.780	0.409	0.591	7.132	12.783
70	1.4	a	4.544	0.077	0.616	4.350	0.077	0.616	0.119	4.354
		b	4.912	0.329	0.646	4.563	0.319	0.652	4.663	8.706
		c	3.900	0.531	0.607	3.563	0.530	0.625	8.397	12.281
		d	1.328	0.899	0.532	1.098	0.927	0.658	30.616	33.558
		e	8.850	0.269	0.682	8.432	0.267	0.683	0.959	5.158
AVERAGE			4.707	0.421	0.617	4.401	0.424	0.647	9.069	12.924

- $LB_E/z^* = 0.487$, $LB/z^* = 0.515$
- The new lower bounding procedure is 3.88 times faster than Euclidean procedure (thanks to preprocessing phase)
- Average reduction of 28.87% in the number of nodes visited by A* (which is independent of the implementation)
- Average reduction in computing time is 14.36% (This speed-up is valuable for a typical web application setting)

- $LB_E/z^* = 0.487$, $LB/z^* = 0.515$
- The new lower bounding procedure is 3.88 times faster than Euclidean procedure (thanks to preprocessing phase)
- Average reduction of 28.87% in the number of nodes visited by A* (which is independent of the implementation)
- Average reduction in computing time is 14.36% (This speed-up is valuable for a typical web application setting)

- $LB_E/z^* = 0.487$, $LB/z^* = 0.515$
- The new lower bounding procedure is 3.88 times faster than Euclidean procedure (thanks to preprocessing phase)
- Average reduction of 28.87% in the number of nodes visited by A* (which is independent of the implementation)
- Average reduction in computing time is 14.36% (This speed-up is valuable for a typical web application setting)

- $LB_E/z^* = 0.487$, $LB/z^* = 0.515$
- The new lower bounding procedure is 3.88 times faster than Euclidean procedure (thanks to preprocessing phase)
- Average reduction of 28.87% in the number of nodes visited by A* (which is independent of the implementation)
- Average reduction in computing time is 14.36% (This speed-up is valuable for a typical web application setting)

- For type *d* and *c* instances (i.e., whenever the origins and destinations are inside an area with moderate speeds) we obtained even greater computing time reductions (up to 28.06%).

- For type *d* and *c* instances (i.e., whenever the origins and destinations are inside an area with moderate speeds) we obtained even greater computing time reductions (up to 28.06%).

Future Research

- Lower bounding procedure integrated into bidirectional A*
- Quick procedures for the selection of *good rectangles* (we used a simple heuristic)
- Combine our approach with the most recent speed-up techniques.
- Time-Dependent Scenario