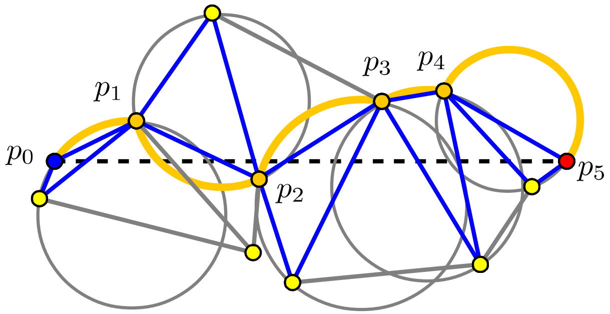
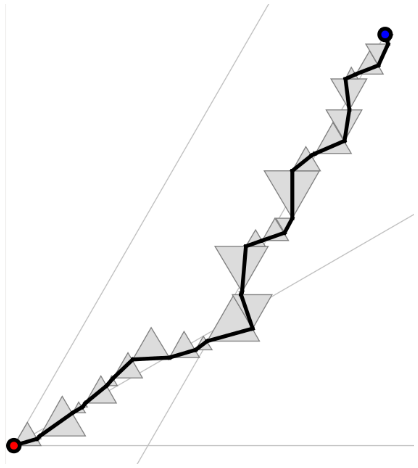
10-15 déc. 2017 Aussois (France)

Routage dans graphes géométriques



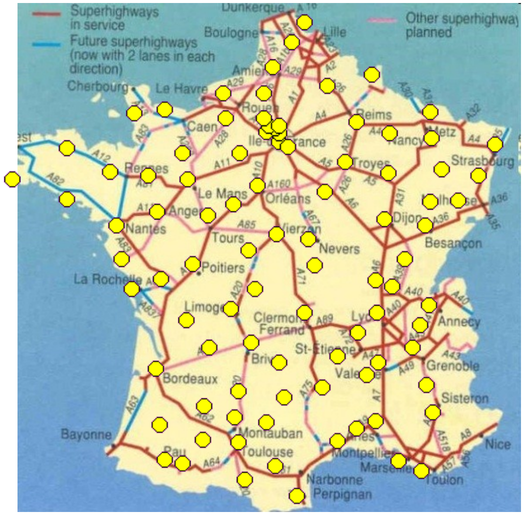


Nicolas Bonichon

LaBRI

université de BORDEAUX

Qu'est-ce qu'un bon réseau ?



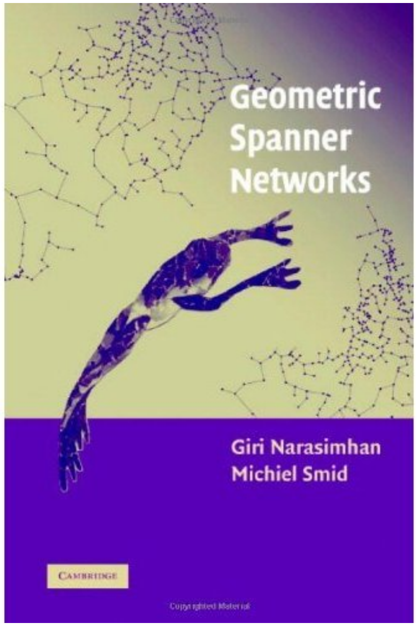
Etirement $:= \max_{a,b} \frac{\text{distance dans le réseau}_{a \to b}}{\text{distance à vol d'oiseau}_{a \to b}}$

Nombre d'arêtes ou somme des longueurs des arêtes

Degré maximal : taille du plus grand rond-point

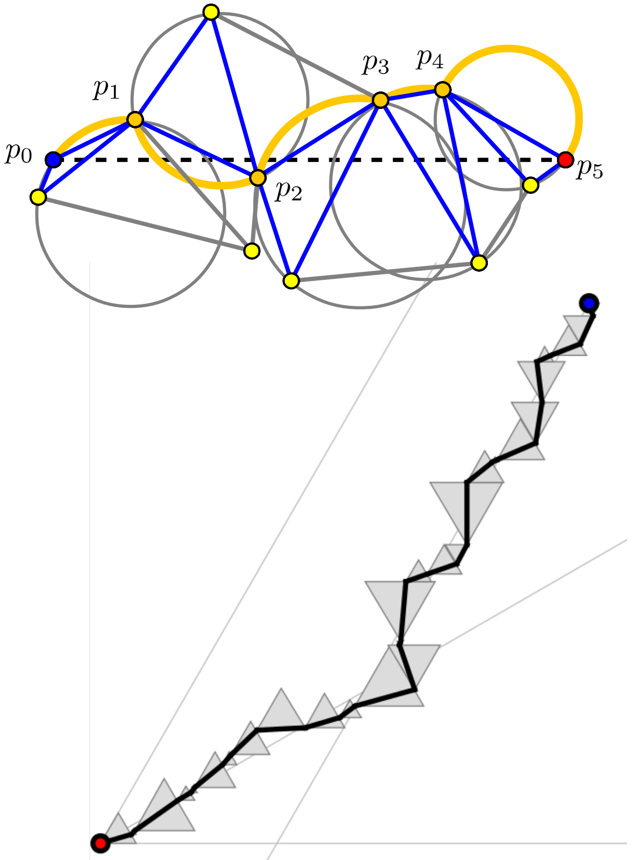
Planaire : ni pont ni tunnel

Facilement Routable



Routing ratio

- Routing algorithm
- Routing in Delaunay
  - Upper & lower bounds
- Routing Theta-Graphs



Routing Algorithm

Neighbors

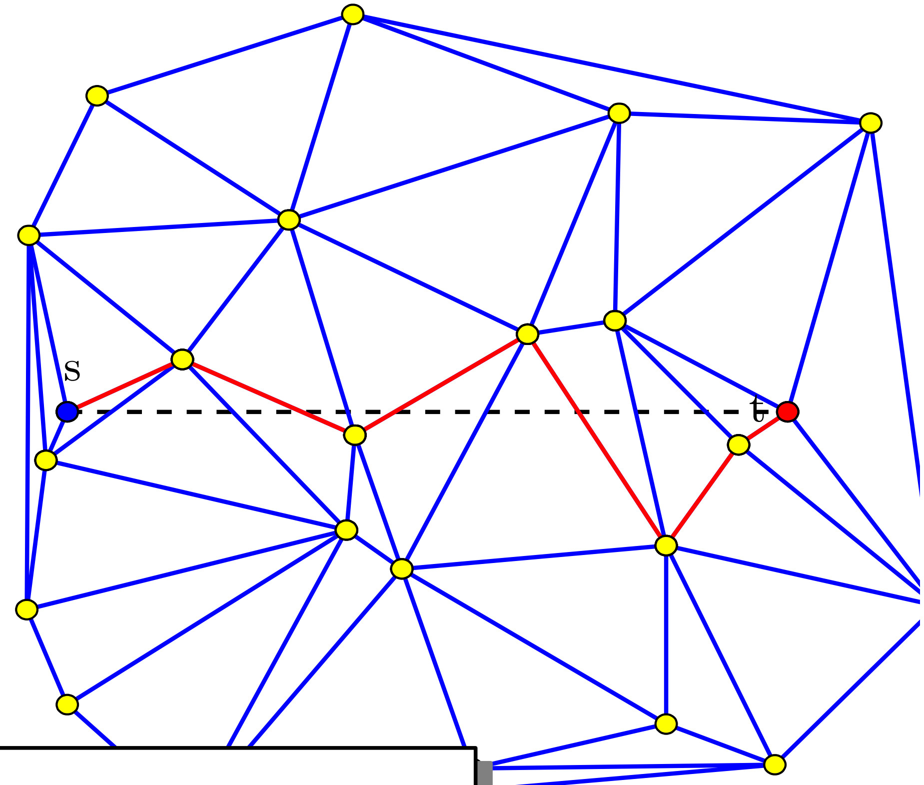
Additional memory

Current Position

Target

```
NextHop(p_i,N(p_i),mem,t)
    (…)
    update(mem)
    return  p_i+1
```

Next Position




```
GreedyNextHop(p_i,N(p_i),∅,t)
    return  neighbor p_i+1 of p_i  such that |p_i+1 t| is min.
```

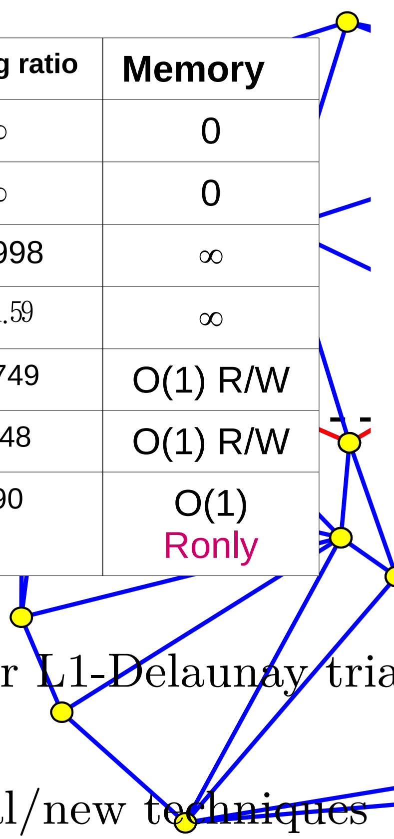
Routing ratio of an algorithm

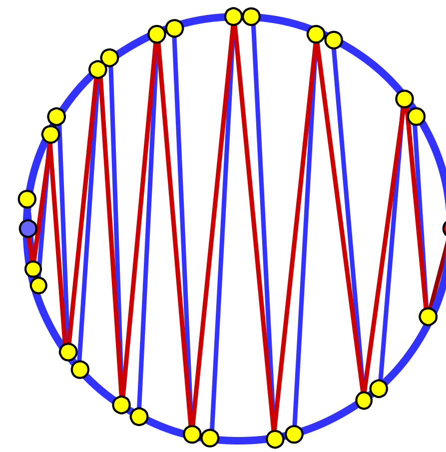
$Route_{\mathcal{A}}(s,t)$: path from s to t computed by $\mathcal{A}$.

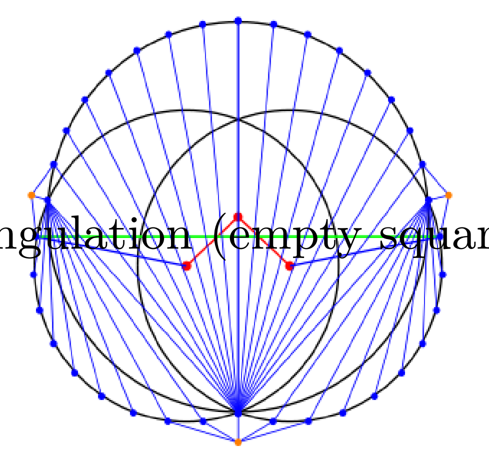
$|Route_{\mathcal{A}}(s,t)|$: sum of length of edges of $Route_{\mathcal{A}}(s,t)$

routing ratio of an algorithm $\mathcal{A}$: $\max_G \max_{s,t} \frac{|route_{\mathcal{A}}(s,t)|}{||st||}$

Algorithm	**Routing ratio**	**Memory**
Greedy routing	∞	0
Compass routing	∞	0
shortest path (non local routing) [Xia 14]	<1.998	∞
shortest path (non local routing) [Xia Zhang 11]	> 1.59	∞
Parallel Voronoi routing [Bose Morin 99]	45,749	O(1) R/W
Searching on the Voronoi path [Bose De Carufel et al 14]	15.48	O(1) R/W
Generalized Chew's algorithm [Bose B. de Carufel Perkovic van Rensen 16]	5.90	O(1) Ronly





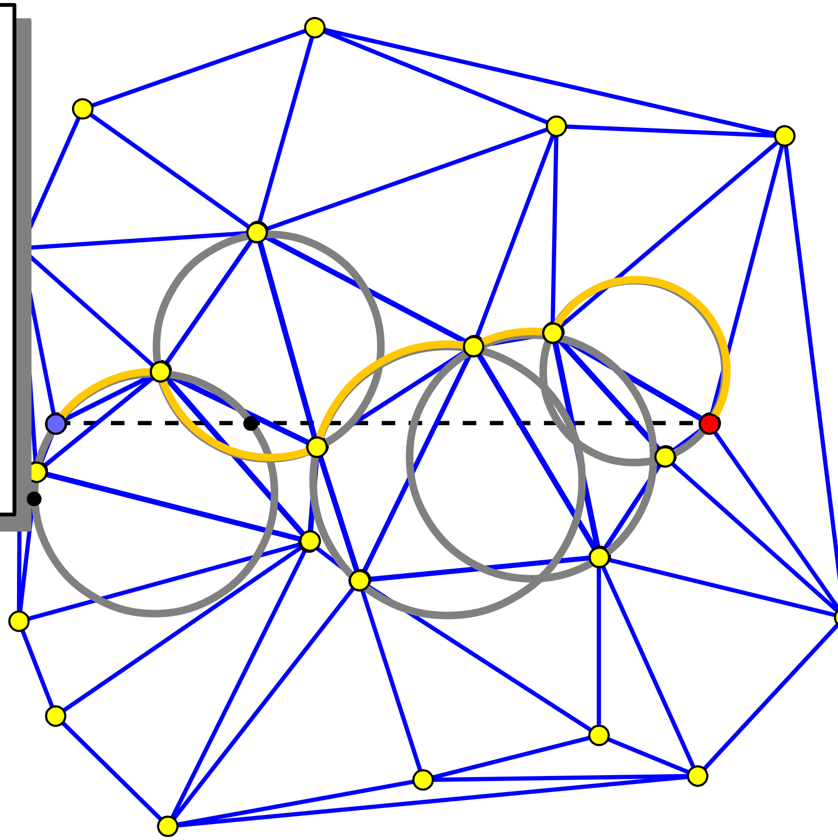


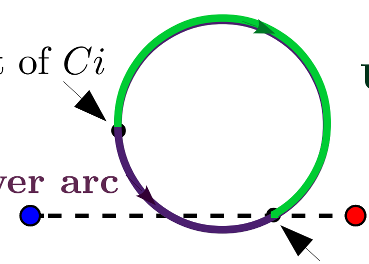
- Chew's algorithm [Chew'89]: designed for L1-Delaunay triangulation (empty square
- Generalization: quite straightforward
- Analysis of the generalization: not trivial/new techniques

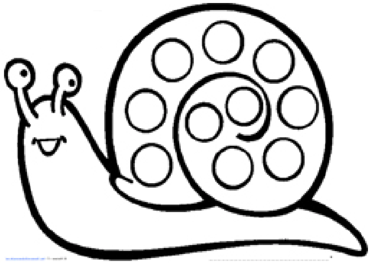
Generalized Chew's algorithm

Consider only triangles that intersect [st]

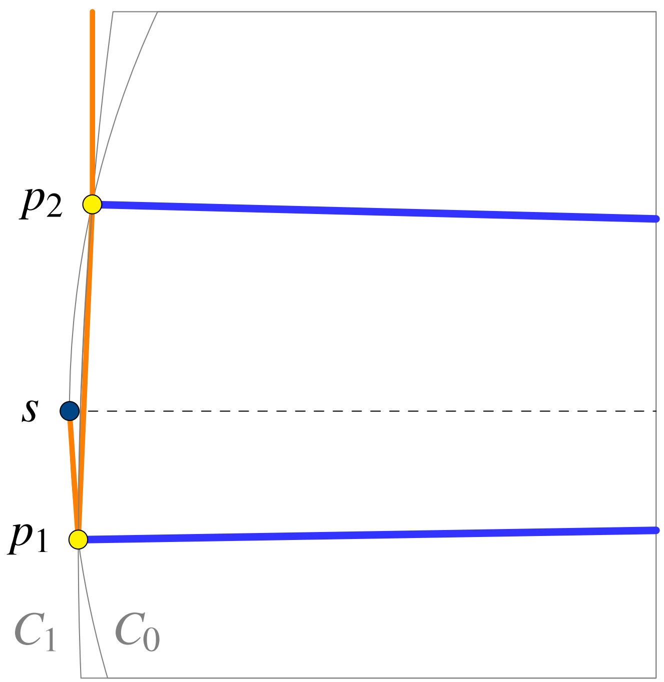
```
ChewNextHop(p_i,N(p_i),s,t)
    T_i ← rightmostTriangle(p_i)
    C_i ← emptyCircle(T_i)
    if p_i ∈ upperArc(C_i)
        move clockwise on C_i
    else
        move counterclockwise on C_i
    return next visited vertex of T_i
```

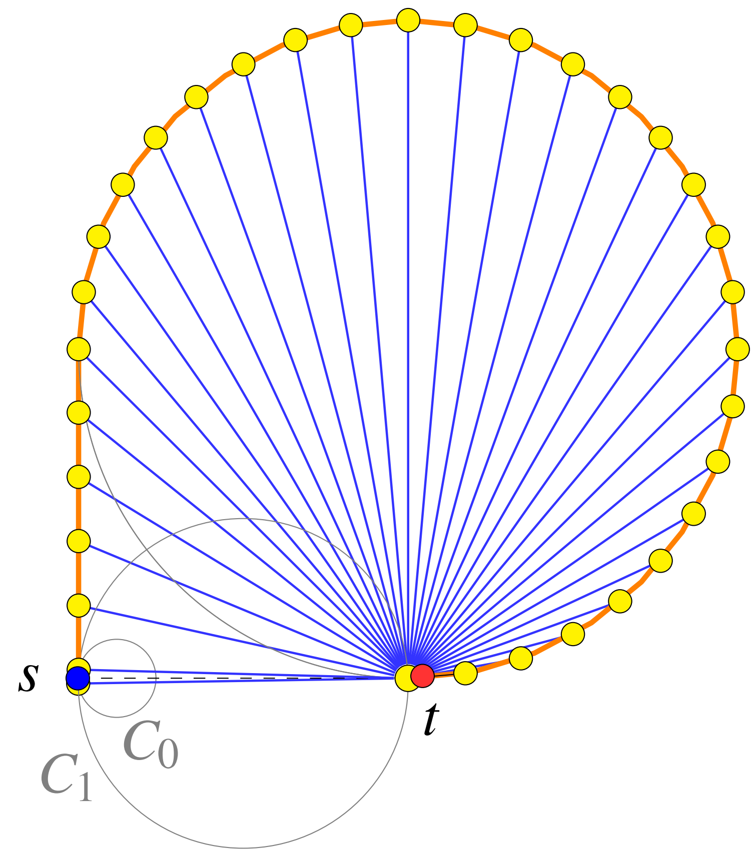






Lower Bound on Chew's routing ratio

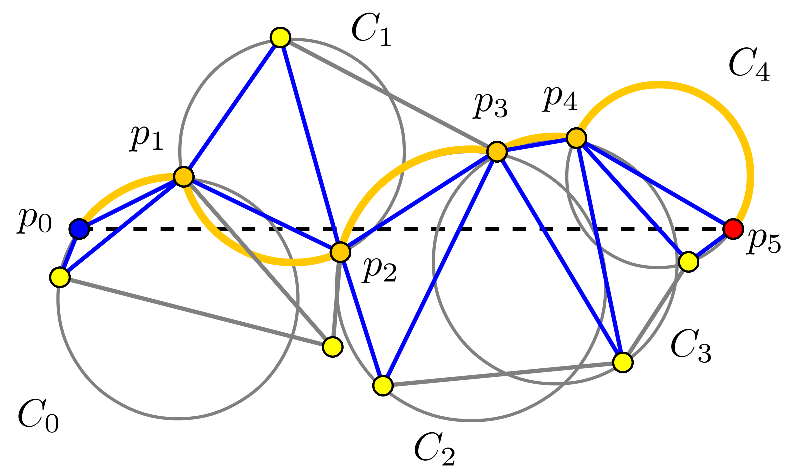




$$\text{routing ratio} \geq 5.7282 > 1 + \frac{3\pi}{2}$$

Analysis of the routing ratio

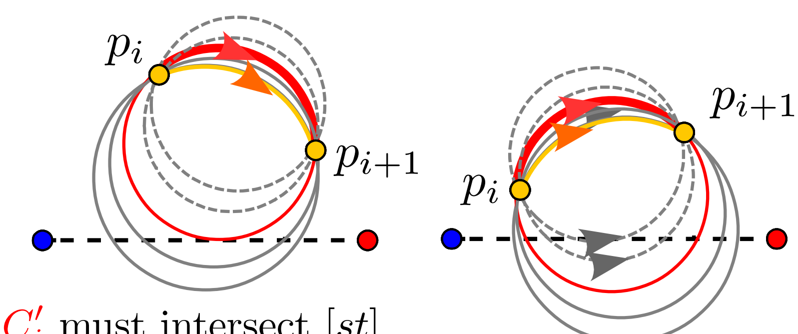
Let's forget irrelevant details:
- my route doesn't depend on the « non-rightmost » triangles
- my route doesn't depend on the position of the third point of the rightmost triangles

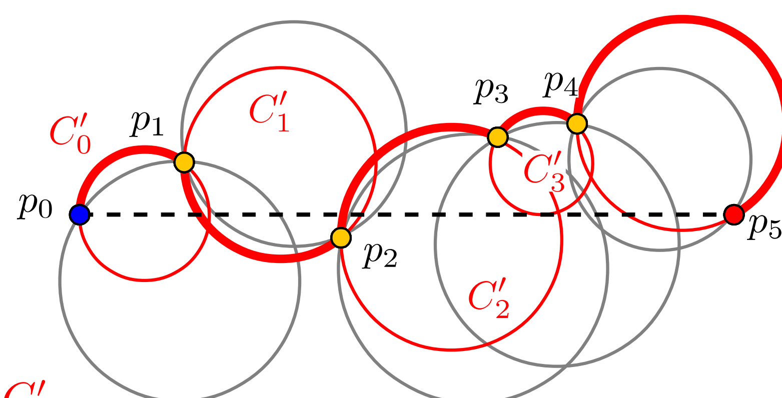


$$|Route_{Chew}(s,t)| \leq |Route_{red}(s,t)|$$

Let's be pessimistic

C'_i: the *worst circle* C_i that goes from p_i to p_{i+1}.



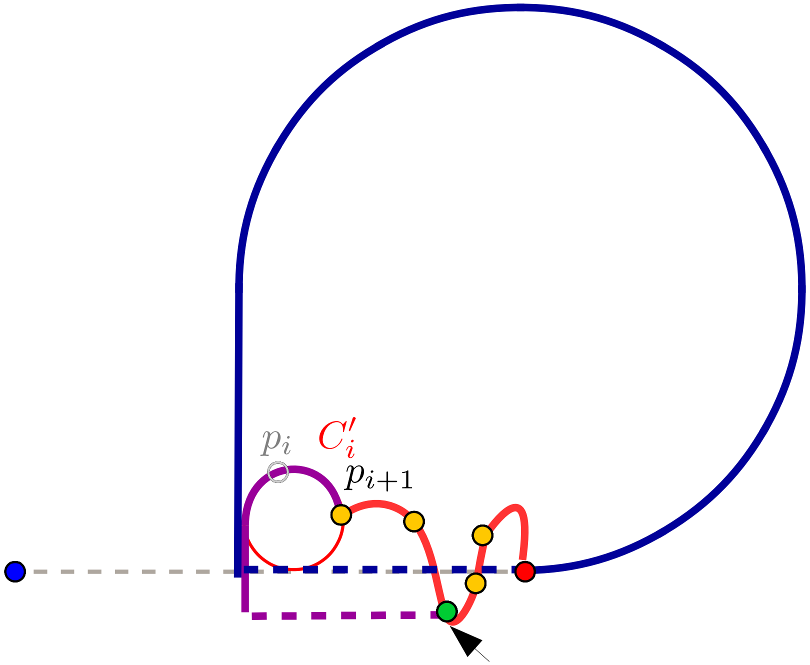


C'_i must intersect $[st]$

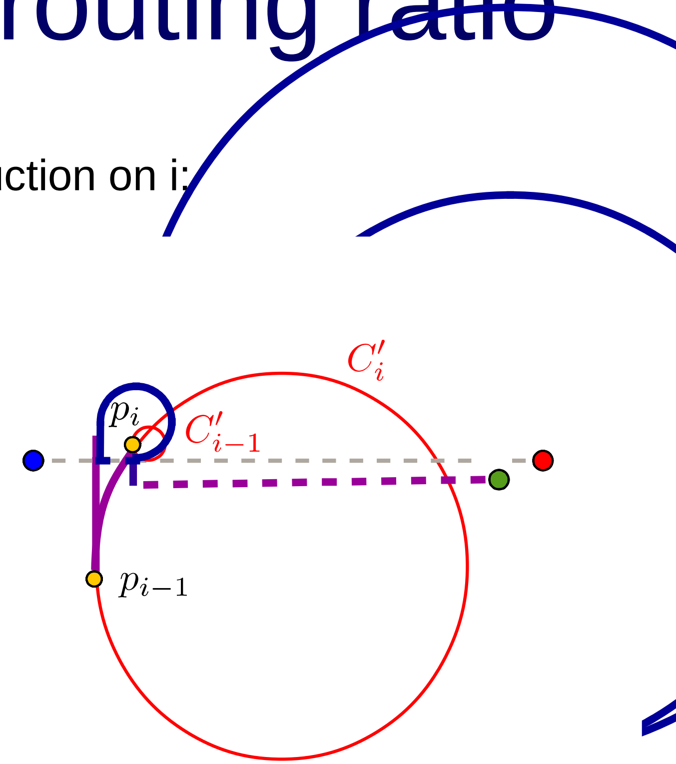
p_i must be on the good arc (upper or lower) of C'_i

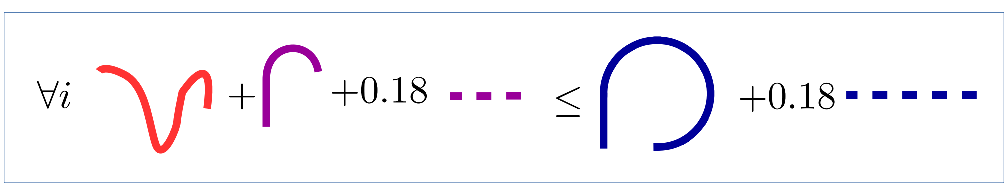
Analysis of the routing ratio

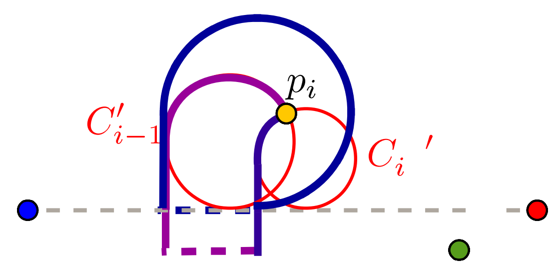
Proof by induction on i:



first point after p_i on the other side of $[st]$







$$i = 0, |Route_{Chew}(s,t)| \leq (1 + \frac{3\pi}{2} + 0.18)|st| = 5.90|st|$$

Routing ratio of an algorithm

$Route_{\mathcal{A}}(s,t)$: path from s to t computed by $\mathcal{A}$.

$|Route_{\mathcal{A}}(s,t)|$: sum of length of edges of $Route_{\mathcal{A}}(s,t)$

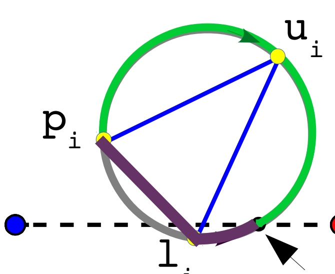
routing ratio of an algorithm $\mathcal{A}$: $\max_G \max_{s,t} \frac{|route_{\mathcal{A}}(s,t)|}{||st||}$

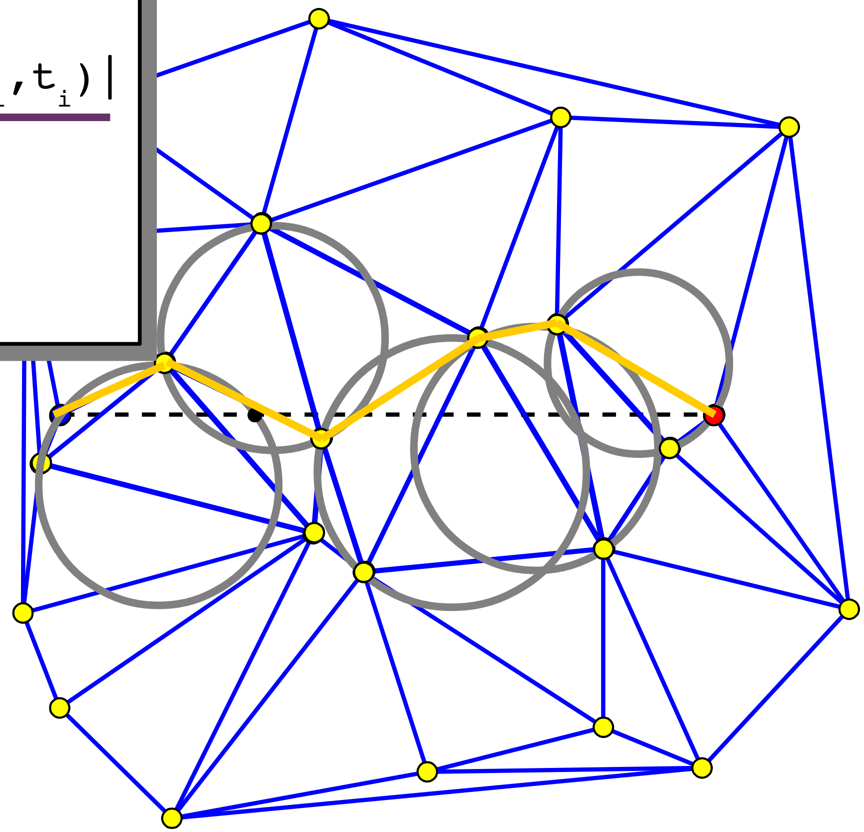
Algorithm	**Routing ratio**	**Memory**
Greedy routing	∞	0
Compass routing	∞	0
shortest path (non local routing) [Xia 14]	<1.998	∞
shortest path (non local routing) [Xia Zhang 11]	> 1.59	∞
Parallel Voronoi routing [Bose Morin 99]	45.749	O(1) R/W
Searching on the Voronoi path [Bose De Carufel et al 14]	15.48	O(1) R/W
Generalized Chew's algorithm [Bose B. de Carufel Perkovic van Rensen 16]	5.90	O(1) Ronly
MixedChordArc algorithm [Bose B. Despré Hill Smid 17+]	3.56	O(1) Ronly

MixedChordArc algorithm

[Bose B. Despré Hill Smid 17+]

```
MixedChordArcNextHop(p_i, N(p_i), s, t)
    T_i ← rightmostTriangle(p_i)
    C_i ← emptyCircle(T_i)
    if |ArcCW(p_i, t_i)| < |p_i l_i| + |ArcCW(p_i, t_i)|
        return u_i
    else
        return l_i
```





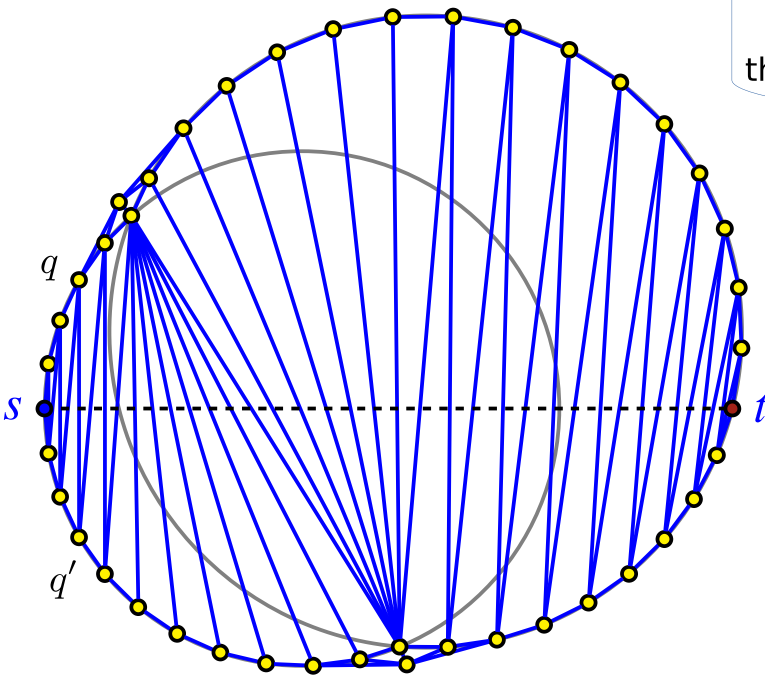
Consider only triangles that intersect [st]

Spanning ratio vs routing ratio

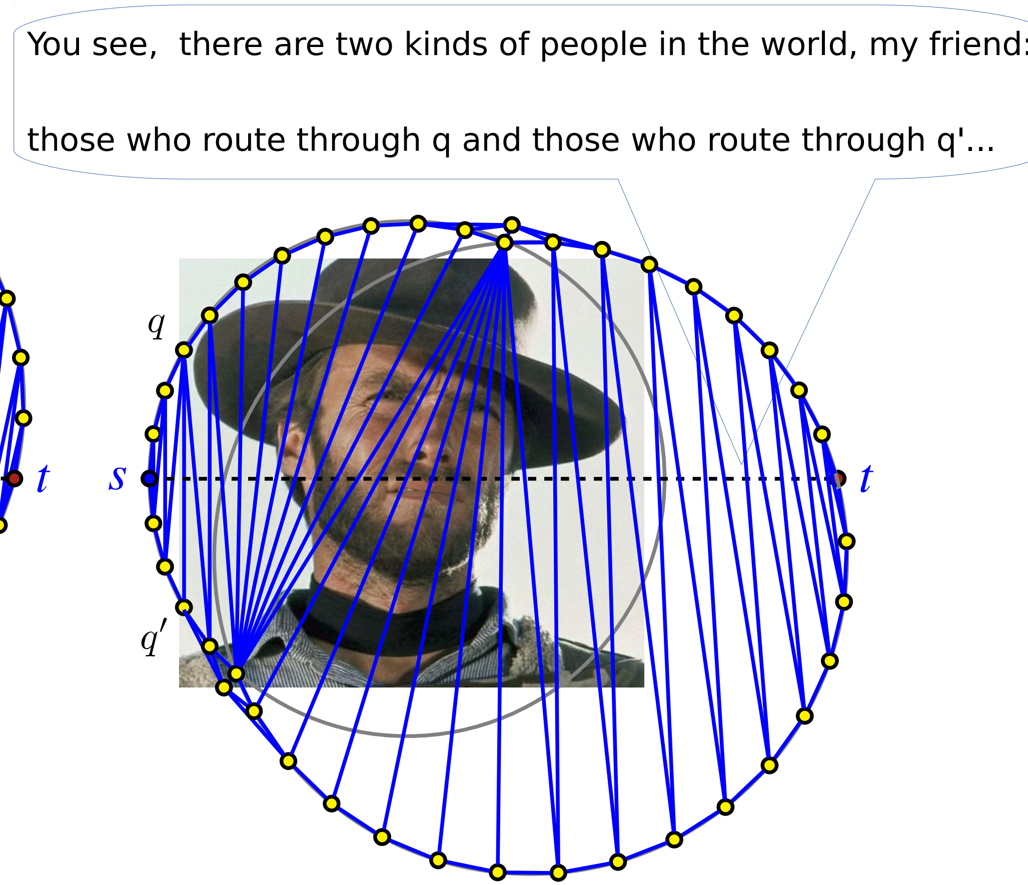
(the cost of locality)

Graph family	Spanning ratio (shortest path)	Routing ratio
TD-Delaunay triangulations △	2 [Chew 86]	2.89 [Bose Fagerberg van Renssen Verdonshot 12]
L1-Delaunay triangulations ◇	2.61 [B. Gavoille Hanusse Perković 14]	2.71 <c < 3.16 [Chew 89]
Delaunay triangulations ○	[Xia Zhang 11] 1.59<c<1.998 [Xia 14]	[B. Bose De Carufel Perkovic van Rensen 14] 1.70<c<3.56 [Bose B. Despré Hill Smid 17+]

Lower bound on the routing ratio



$dist_G(s, q) + dist_G(q, t) = 1.70$



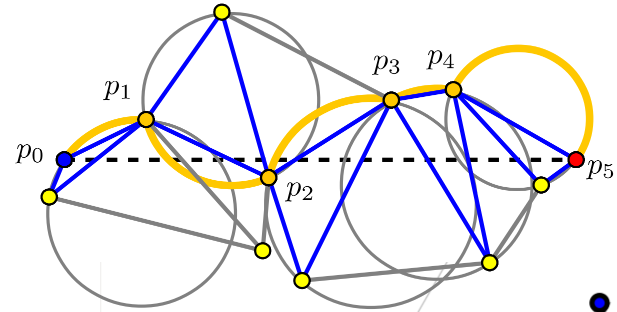
There is no deterministic k-local routing algorithm with routing ratio < 1.70

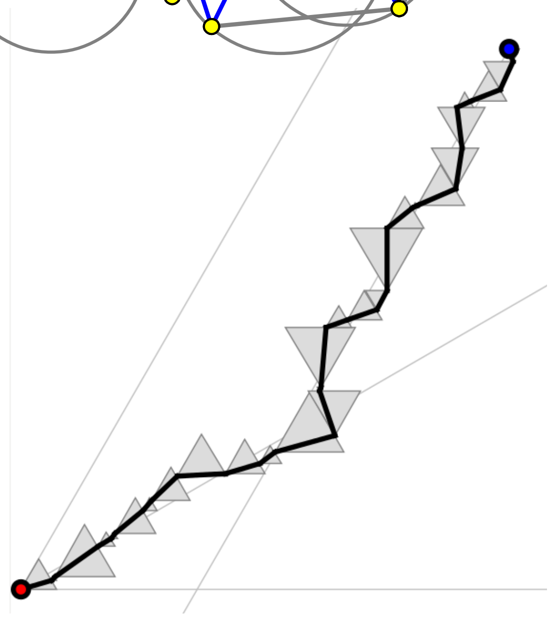
Routing ratio

- Routing algorithm

- Routing in Delaunay
  - Upper & lower bounds

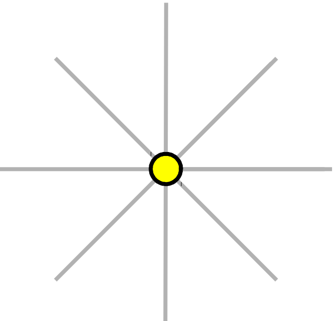
- Routing Theta-Graphs



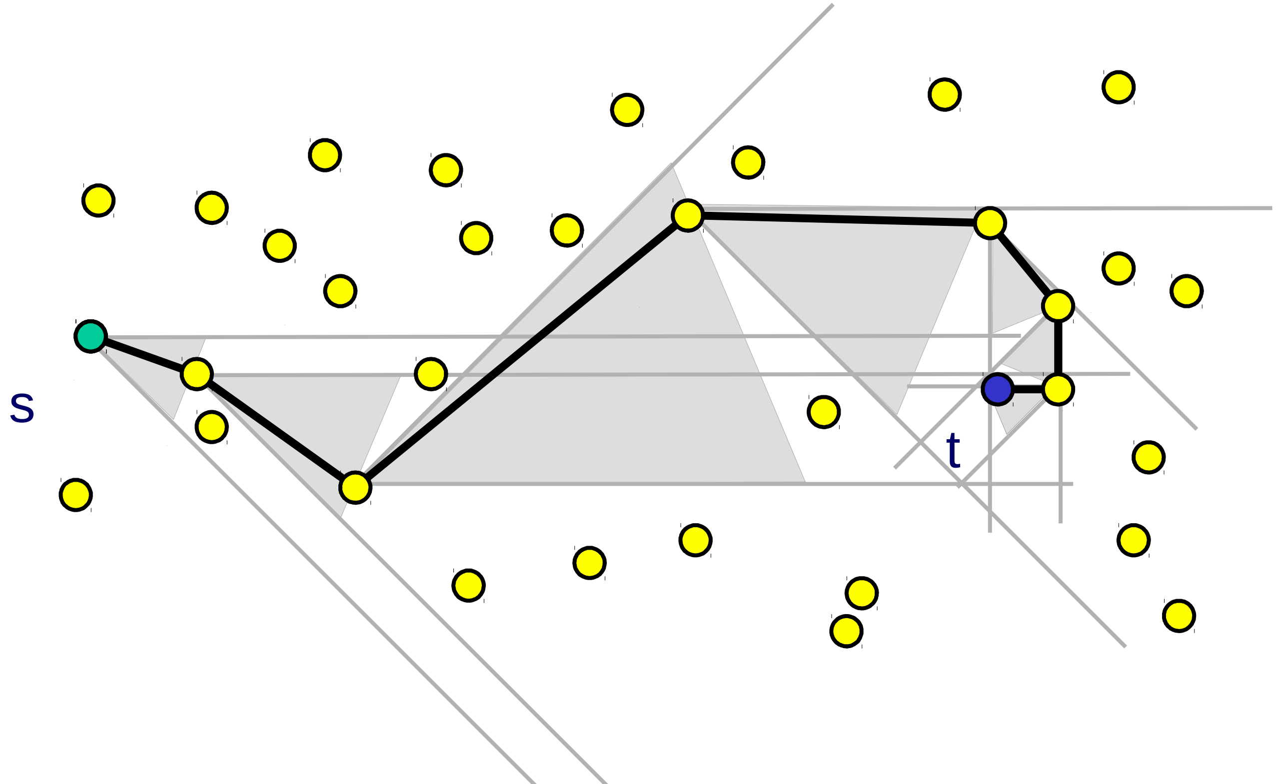


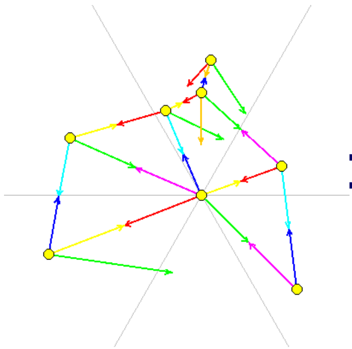
Compass routing

- Select the neighour that is in the direction of the destination.



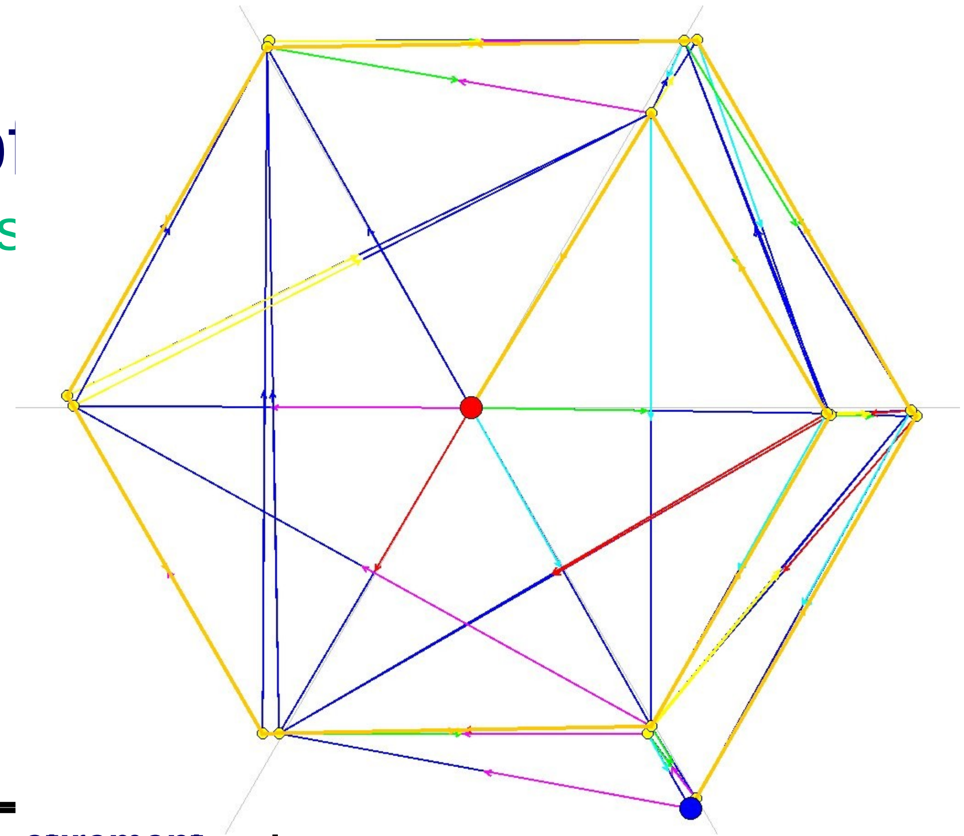
$k=8$





Routing ratio of

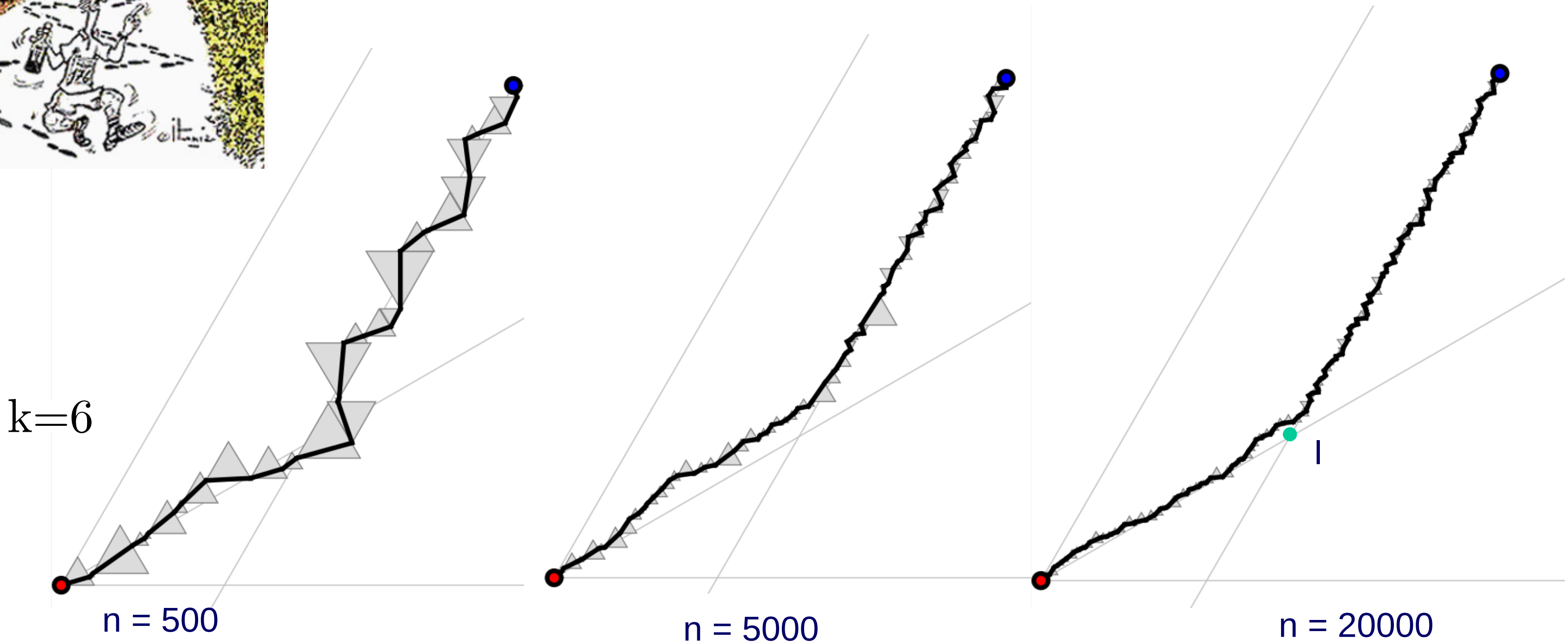
[Clarks



papier	k	étirement
[El Molla 09]	2,3	∞
	4	
	5	
	6	∞
[Clarkson 87][Keil 88]	≥ 7	$\frac{1}{1-2\sin(\pi/k)}$

k	stretch
7	7.56
8	4.26
…	…
12	2.07
13	1.91

Quand le nombre de points devient grand…
[Marckert B. 11]



$$c_1 = \frac{1}{2}\left(\frac{1}{\cos(\pi/k)} + \frac{\operatorname{arcsinh}(\tan(\pi/k))}{\tan(\pi/k)}\right)$$

$$c_2 = \frac{1}{2}\left(\frac{1}{\cos^2(\pi/k)} + \frac{\operatorname{arcsinh}(\tan(\pi/k))}{\sin(\pi/k)}\right)$$

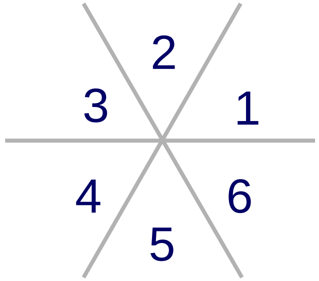
$$\text{chemin(s,t)} \to [s, I] \cup [I, t]$$

$$|\text{chemin}(s,t)| \to 1,05||s,I|| + 1.22||I,t||$$

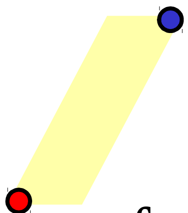
$$|\text{chemin}(s,t)| \to \quad c_1||s,I|| + \quad c_2||I,t||$$

$$Pr(\sup_{s,t}(|\text{chemin}(s,t)| - (c_1||s,I|| + c_2||I,t||)) > \frac{1}{n^\alpha}) < \frac{1}{n^\beta}$$

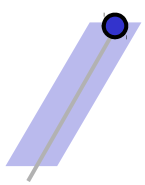
Valable aussi pour des densités de points non-uniformes



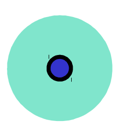
Decomposition of the trajectory in 3 phases



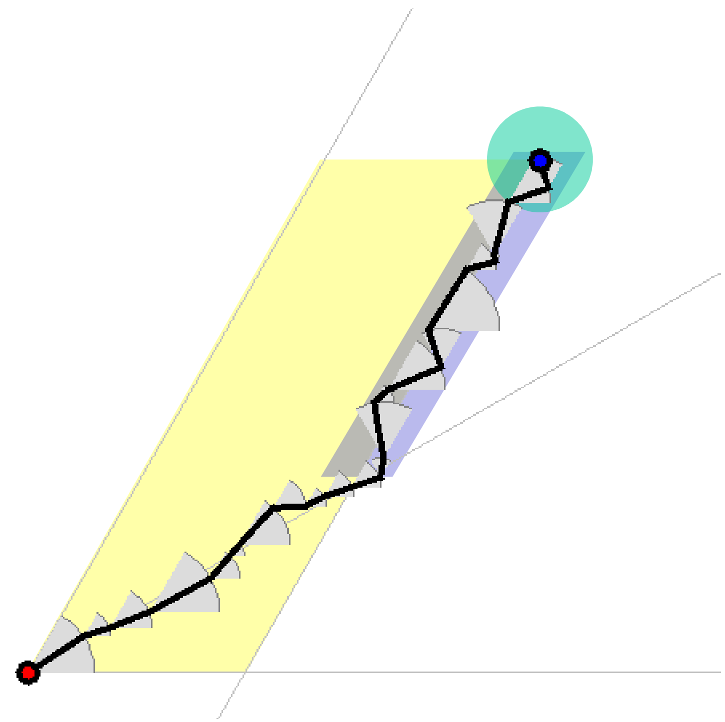
- Phase 1:
  - ⇒ only steps of type 1.



- Phase 2:
  - Only steps of type 1 and 2



- Phase 3:
  - Final approach
  - Steps of any type



Compass routing algorithm

(*straight*)

- Compute a path by selecting at each step the neighbor in the direction of the target.