Universidade Federal do Espírito Santo

Planejamento de Caminho para Carros Autônomos

Pedro Henrique Vieira de Oliveira Azevedo - pedro.hvo.azevedo@gmail.com

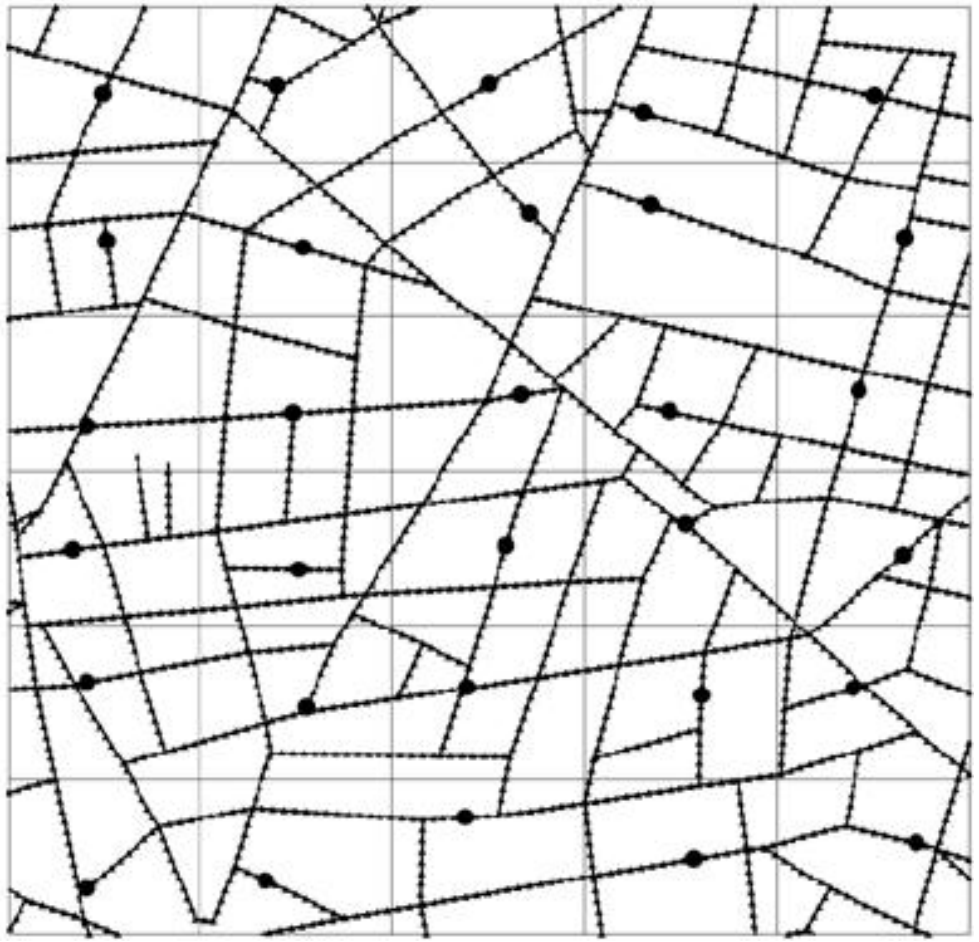
Teoria dos Grafos

Descrição do Problema

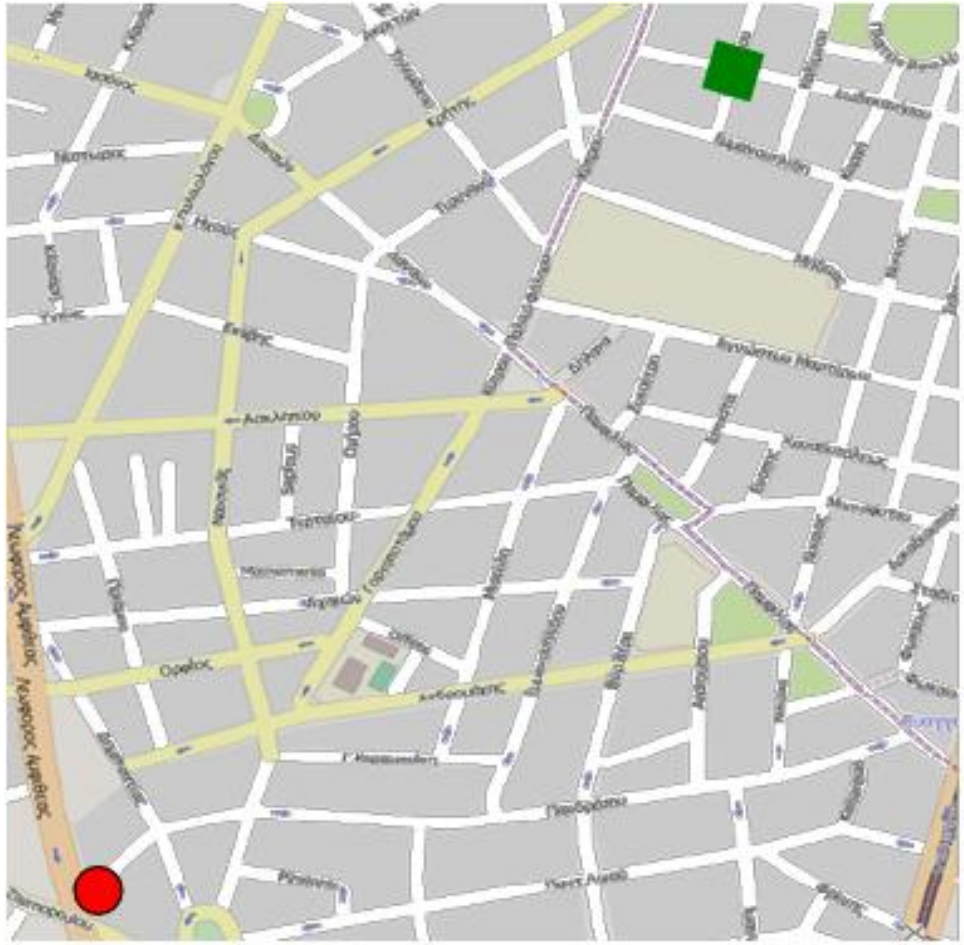
- Planejamento de Caminho consiste em gerar uma sequência de pontos de um ponto inicial s_i até um ponto final s_g.

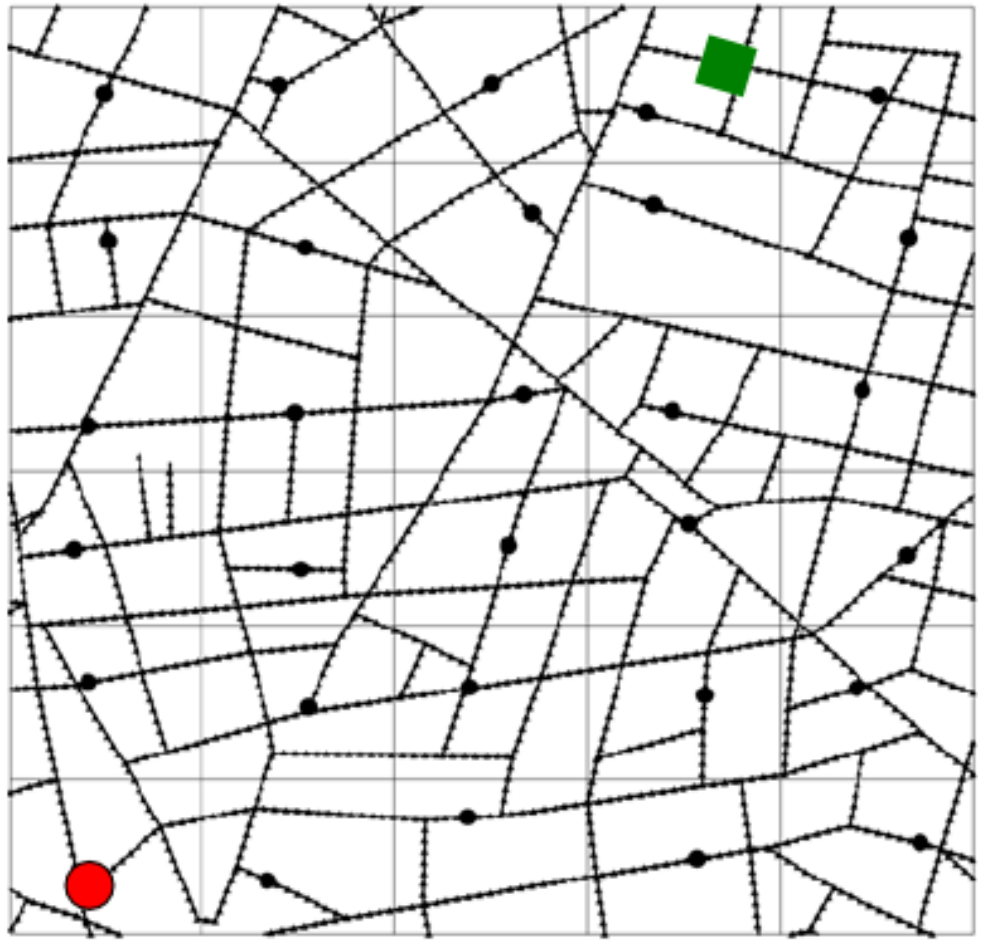
- Planejamento de Caminho Global *vs.* Planejamento de Caminho Local.

Modelagem em Grafos

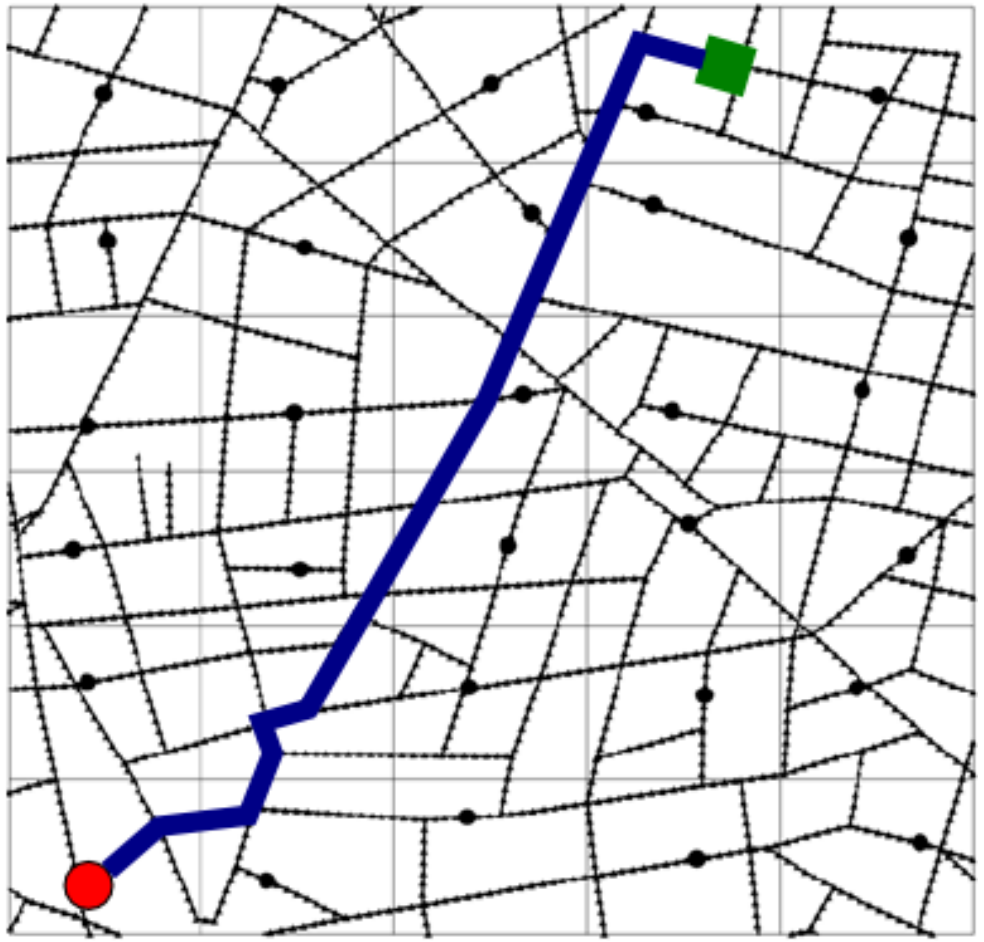


Modelagem em Grafos





Modelagem em Grafos



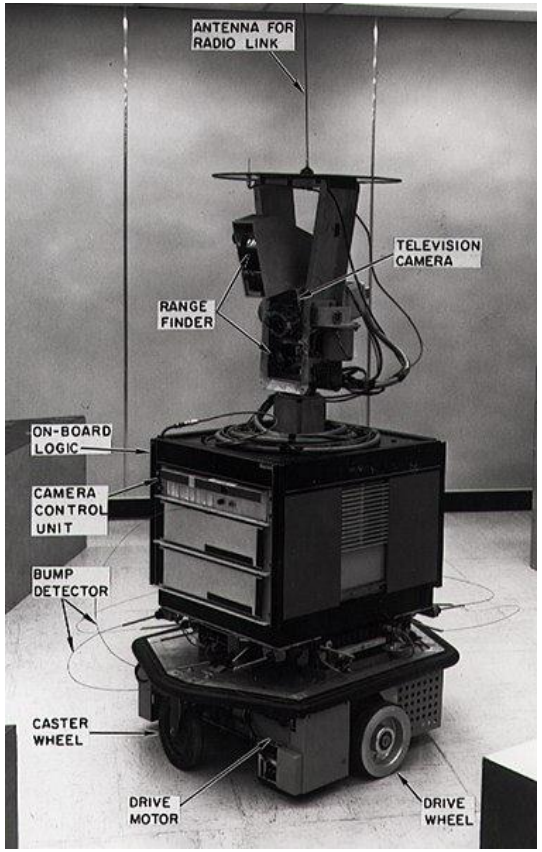
Algoritmos de Planejamento de Caminho

- Muito utilizado na robótica desde o final dos anos 60.

- Shakey, desenvolvido pela DARPA (Agência de Projetos de Pesquisa Avançada de Defesa dos EUA).

- Robô que navega por um ambiente desviando de obstáculos.

- Incrementado com o algoritmo A* [1] em 68.



Algoritmos de Planejamento de Caminho

- Estopim carros autônomos: competições DARPA 2004, 2005 e 2007 [2][3].

- Algoritmos mais utilizados para o planejamento de caminho global são:
  - Dijkstra (Ainda!!) [4][5][6]
  - A* [7][8][9]

- Publicações mais recentes na área estão cada vez mais difíceis pois o mercado está abraçando os pesquisadores.

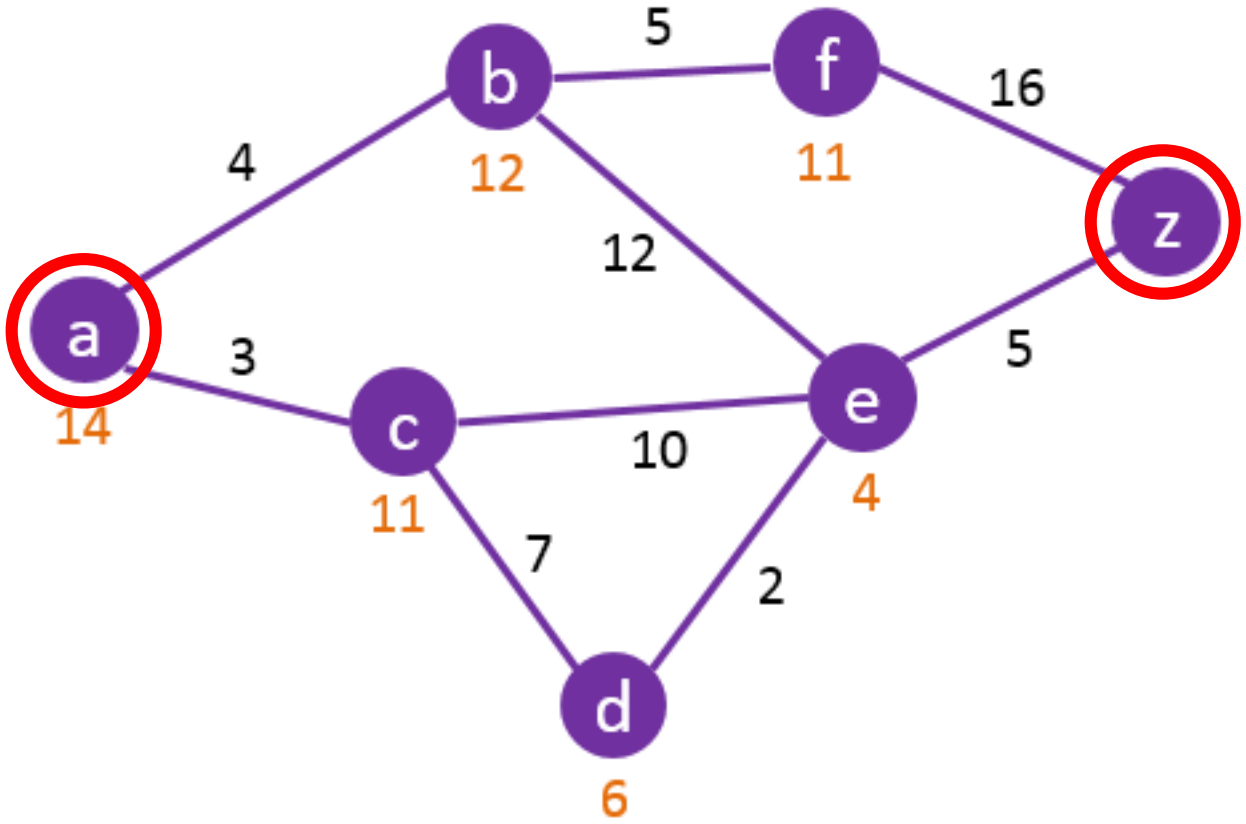
O Algoritmo A*

- O algoritmo A* seleciona um caminho que minimize a função:

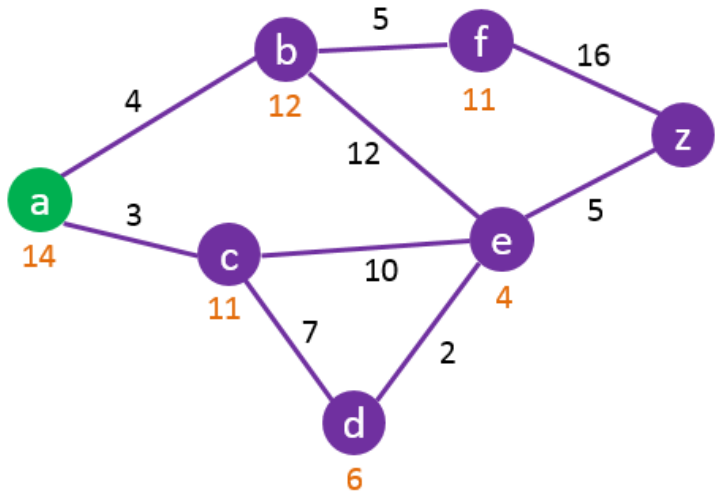
$$f(n) = g(n) + h(n)$$

- Onde:
  - $n =$ nó
  - $g(n) =$ custo de s_i até n
  - $h(n) =$ heurística que estima o custo de n até s_f
    - Mais comuns: Distância Euclidiana e Distância Manhatan

O Algoritmo A*

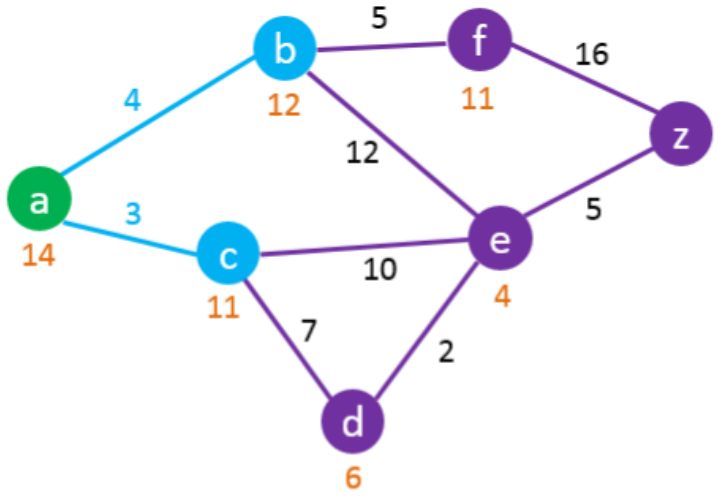


O Algoritmo A*



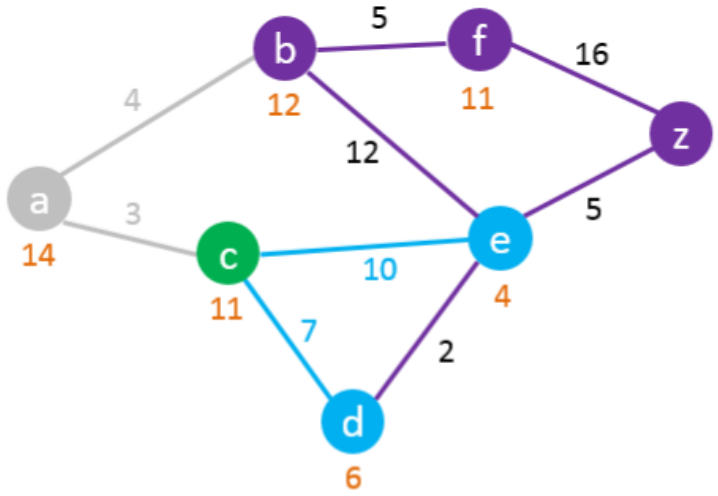
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Current	0	14	14	
B		∞	12		
C		∞	11		
D		∞	6		
E		∞	4		
F		∞	11		
Z		∞	0		

O Algoritmo A*



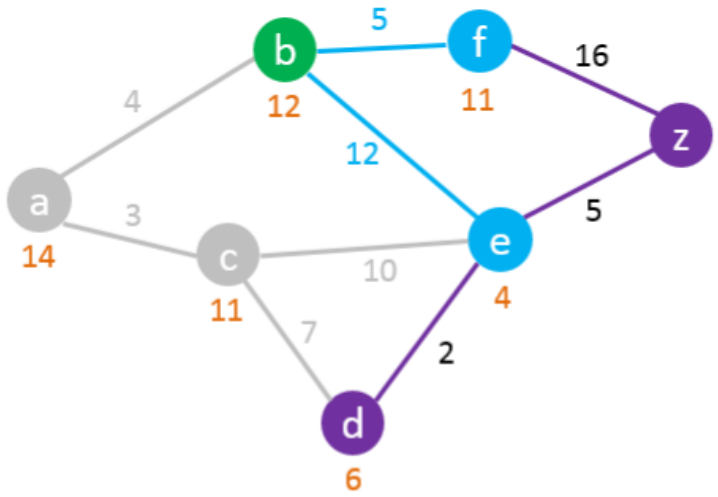
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Current	0	14	14	
B		∞ 4	12	16	A
C		∞ 3	11	14	A
D		∞	6		
E		∞	4		
F		∞	11		
Z		∞	0		

O Algoritmo A*



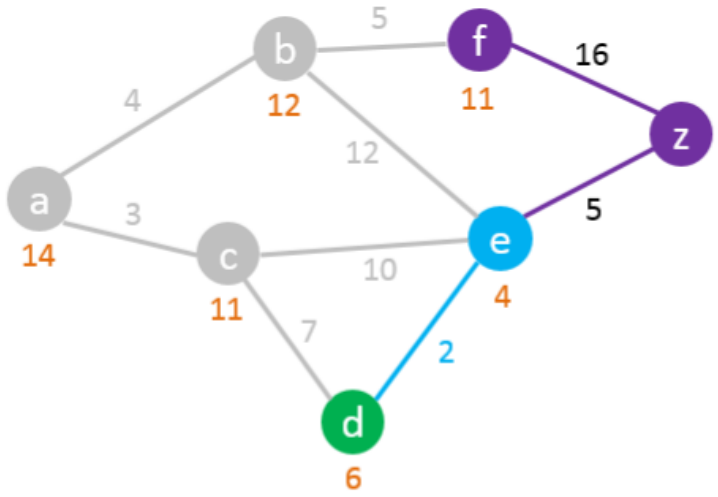
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B		4	12	16	A
C	Current	3	11	14	A
D		∞ 3+7=10	6	16	C
E		∞ 3+10=13	4	17	C
F		∞	11		
Z		∞	0		

O Algoritmo A*



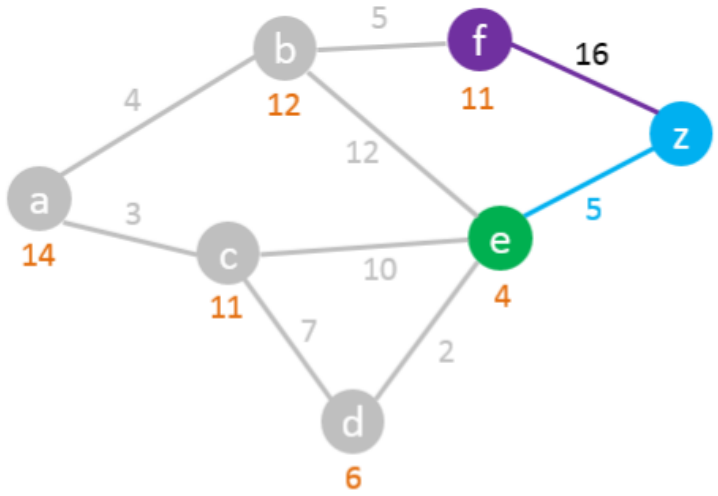
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Current	4	12	16	A
C	Visited	3	11	14	A
D		10	6	16	C
E		13 4+12=~~16~~	4	17	C
F		~~∞~~ 4+5=9	11	20	B
Z		∞	0		

O Algoritmo A*



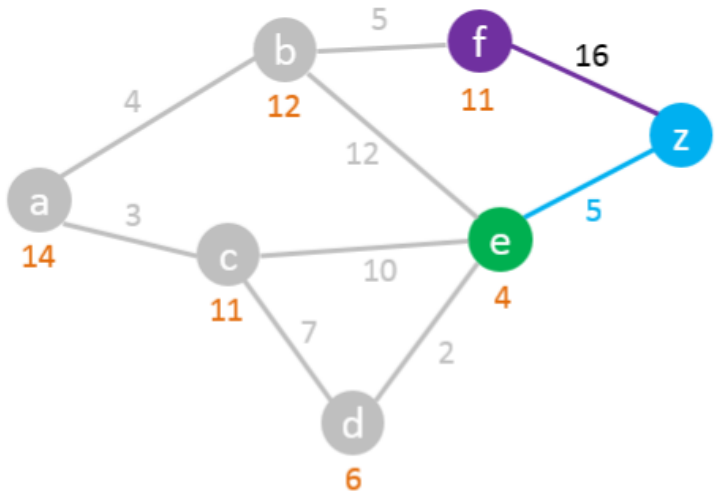
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Current	10	6	16	C
E		~~13~~ 10+2=12	4	16	D
F		9	11	20	B
Z		∞	0		

O Algoritmo A*



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Visited	10	6	16	C
E	Current	12	4	16	D
F		9	11	20	B
Z		∞ 12+5=17	0	17	E

O Algoritmo A*



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Visited	10	6	16	C
E	Visited	12	4	16	D
F		9	11	20	B
Z	Current	17	0	17	E

O Algoritmo A* - Forma Canônica

Entrada: grafo, s_i, s_f

fechado[ ] = 0, anterior[ ] = 0, gScore [ ] = ∞, fScore [ ] = ∞

gScore [s_i] = 0

fScore [s_i] = h_n(s_i, s_f) //Utilizei distância Euclidiana

enquanto (current != s_f)

current = nó não fechado que possui o menor fScore

fechado[current] = 1

para cada vizinho de current

se fechado[vizinho] != 1

t_gScore = gScore[current]+custo[vizinho]

se t_gScore <= gScore[vizinho]

anterior[vizinho] = current

gScore[vizinho] = t_gScore

fScore[vizinho] = gScore[vizinho] + h_n(vizinho, s_f)

O Algoritmo A* - Forma Canônica

Entrada: grafo, s_i, s_f

fechado[ ] = 0, anterior[ ] = 0, gScore [ ] = ∞, fScore [ ] = ∞

gScore [s_i] = 0

fScore [s_i] = h_n(s_i, s_f) //Utilizei distância Euclidiana

enquanto (current != s_f)

current = nó não fechado que possui o menor fScore ← **GARGALO! $O(n)$**

fechado[current] = 1

para cada vizinho de current

se fechado[vizinho] != 1

t_gScore = gScore[current]+custo[vizinho]

se t_gScore <= gScore[vizinho]

anterior[vizinho] = current

gScore[vizinho] = t_gScore

fScore[vizinho] = gScore[vizinho] + h_n(vizinho, s_f)

O Algoritmo A* - Forma Canônica

Entrada: grafo, s_i, s_f

fechado[ ] = 0, anterior[ ] = 0, gScore [ ] = ∞, fScore [ ] = ∞

gScore [s_i] = 0

fScore [s_i] = h_n(s_i, s_f) //Utilizei distância Euclidiana

enquanto (current != s_f) ← $O(m), m \leq n$

current = nó não fechado que possui o menor fScore ← **GARGALO!** $O(n)$

fechado[current] = 1

para cada vizinho de current

se fechado[vizinho] != 1

t_gScore = gScore[current]+custo[vizinho]

se t_gScore <= gScore[vizinho]

anterior[vizinho] = current

gScore[vizinho] = t_gScore

fScore[vizinho] = gScore[vizinho] + h_n(vizinho, s_f)

O Algoritmo A* - Forma Canônica

Entrada: grafo, s_i, s_f

fechado[ ] = 0, ante

gScore [s_i] = 0

fScore [s_i] = h_n(s_i

enquanto (current

current =

fechado[c

para cada

se fechado[vizinho] != 1

t_gScore = gScore[current]+custo[vizinho]

se t_gScore <= gScore[vizinho]

anterior[vizinho] = current

gScore[vizinho] = t_gScore

fScore[vizinho] = gScore[vizinho] + h_n(vizinho, s_f)

Complexidade Total:

$$O(m * n)$$

← GARGALO! $O(n)$

Resultados Computacionais

- Instâncias de distância do trabalho anterior.
- Comparar resultados e tempos computacionais do Dijkstra com o A*.
- Cada instância foi dividida em um caminho fácil, médio e difícil.
- Vértice 1 como ponto de partida para todos as instâncias.
  - Difícil (df) = primeiro vértice periférico encontrado
  - Médio (md) = $\frac{df}{2} - 100 < \frac{df}{2} < \frac{df}{2} + 100$
  - Fácil (fa) = $\frac{md}{2} - 100 < \frac{md}{2} < md + 100$

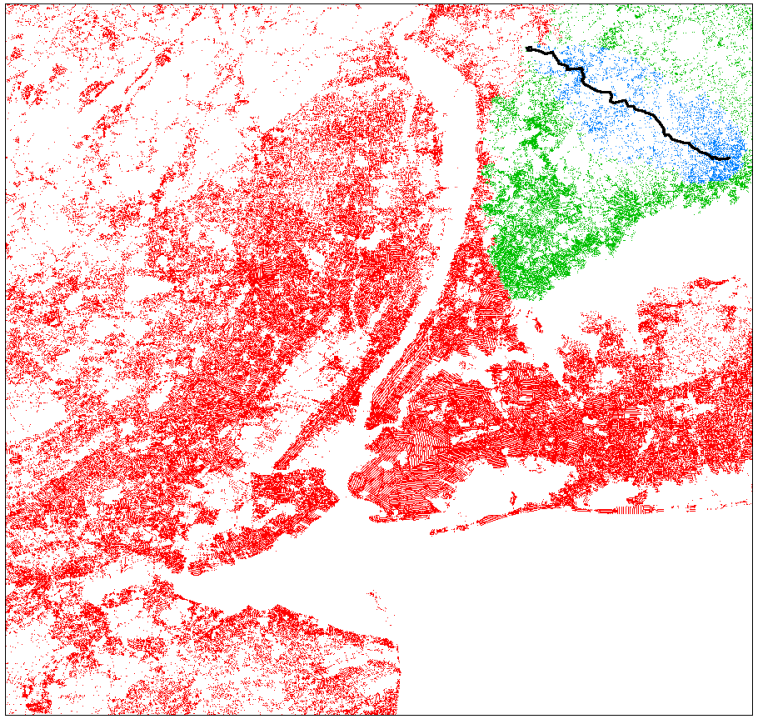
Resultados Computacionais – NY

Vértices: 264346 | Arestas: 733846

NY.d	TEMPO DE EXECUÇÃO (Dijkstra | A*)		CUSTO (Dijkstra | A*)	
Fácil	13.84 secs	2.43 secs	359425	359425
Médio	1 min 7 secs	19.75 secs	719659	720000
Difícil	1 min 52 secs	2 mins 4 secs	1440081	1440241

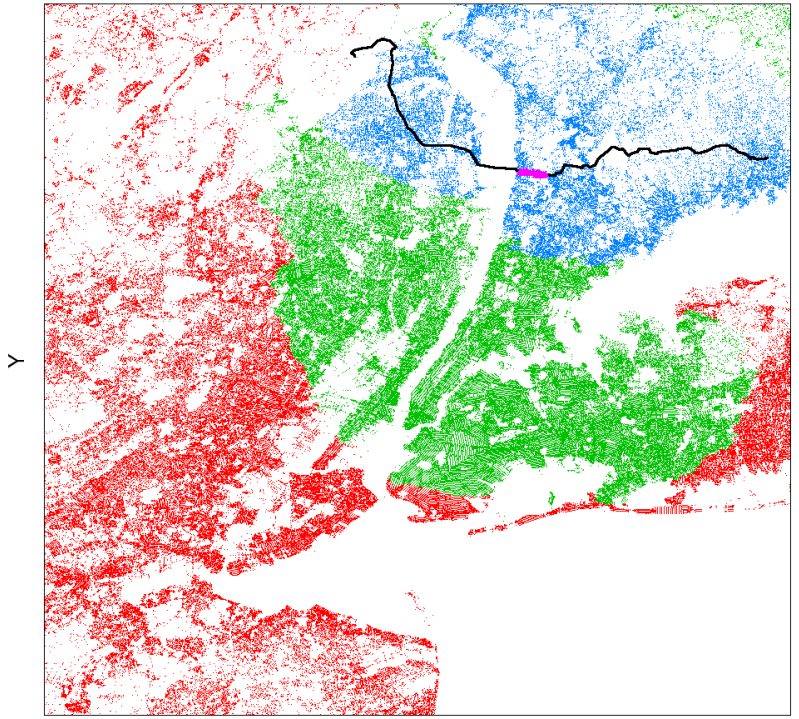
Resultados Computacionais – NY.d

New York Distância Fácil



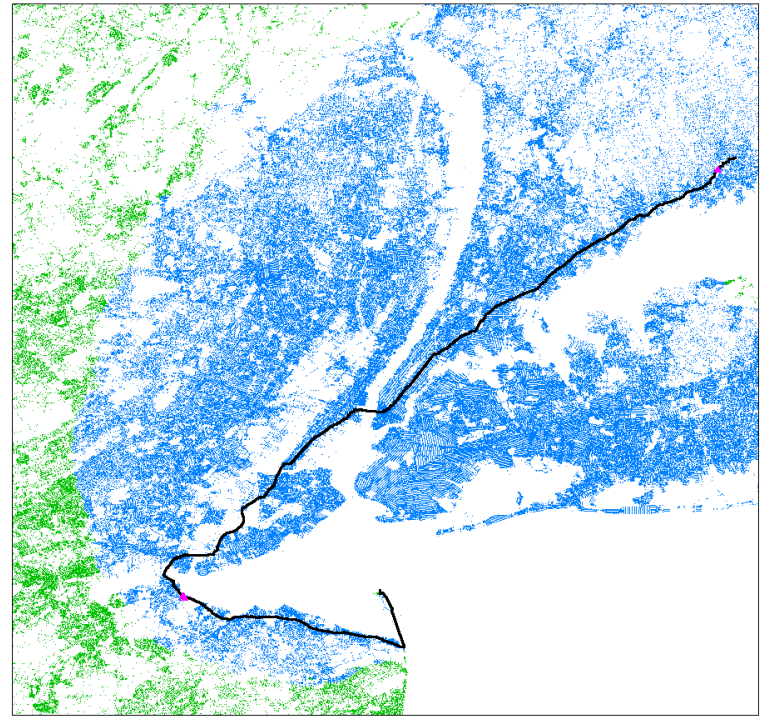
Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
27448	5052	193	193
-22396 pts		0	

New York Distância Médio



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
147106	37773	296	296
-109333 pts		0	

New York Distância Difícil



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
264346	227394	1177	1177
-36953 pts		0	

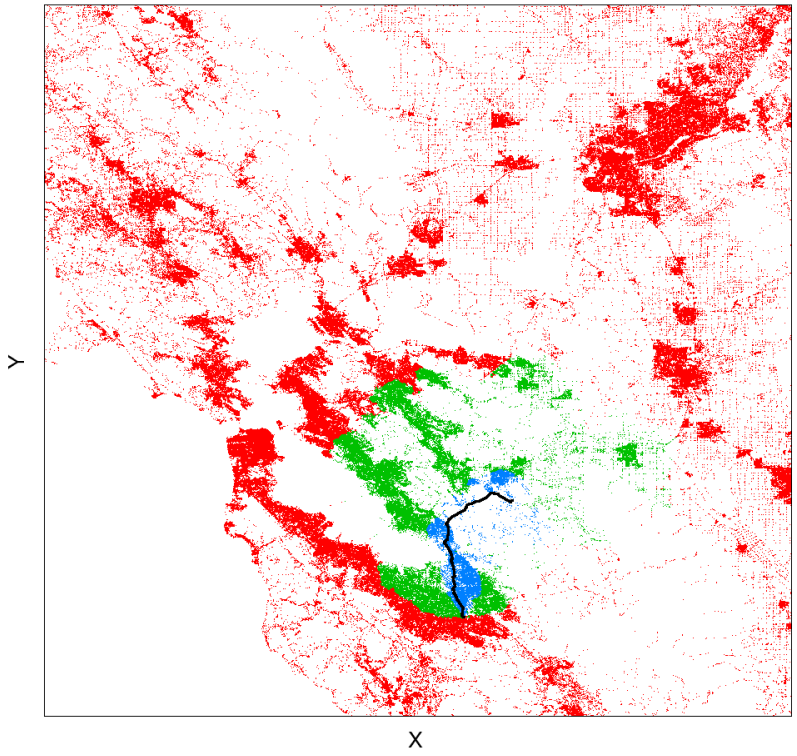
Resultados Computacionais – BAY

Vértices: 321270 | Arestas: 800172

BAY.d	TEMPO DE EXECUÇÃO (Dijkstra | A*)		CUSTO (Dijkstra | A*)	
Fácil	40.98 secs	7.88 secs	539988	539988
Médio	1 min 54 secs	13.71 secs	1082034	1082034
Difícil	2 mins 45 secs	1 mins 23 secs	2165941	2166010

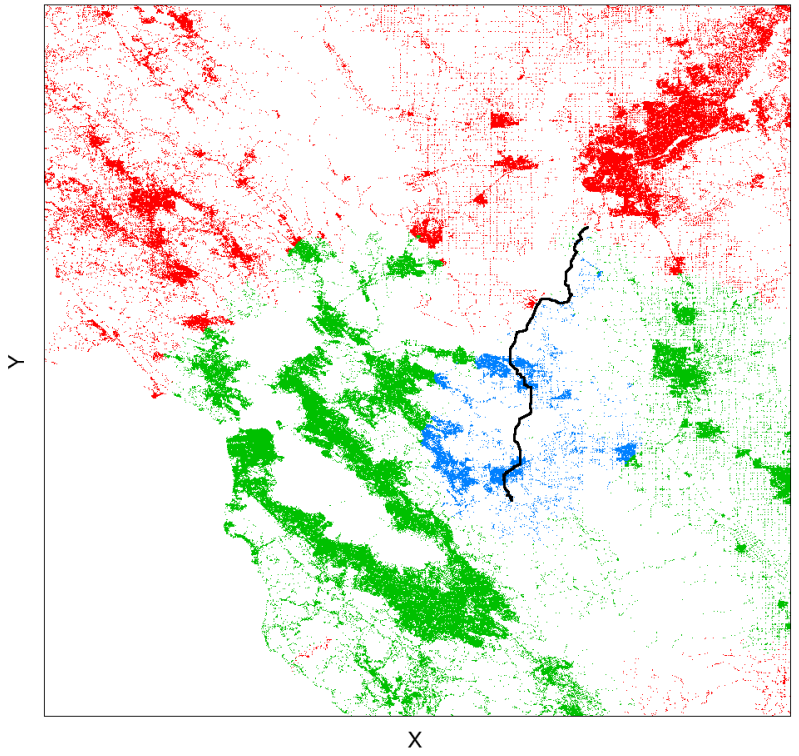
Resultados Computacionais – BAY.d

São Francisco Bay Distância Fácil



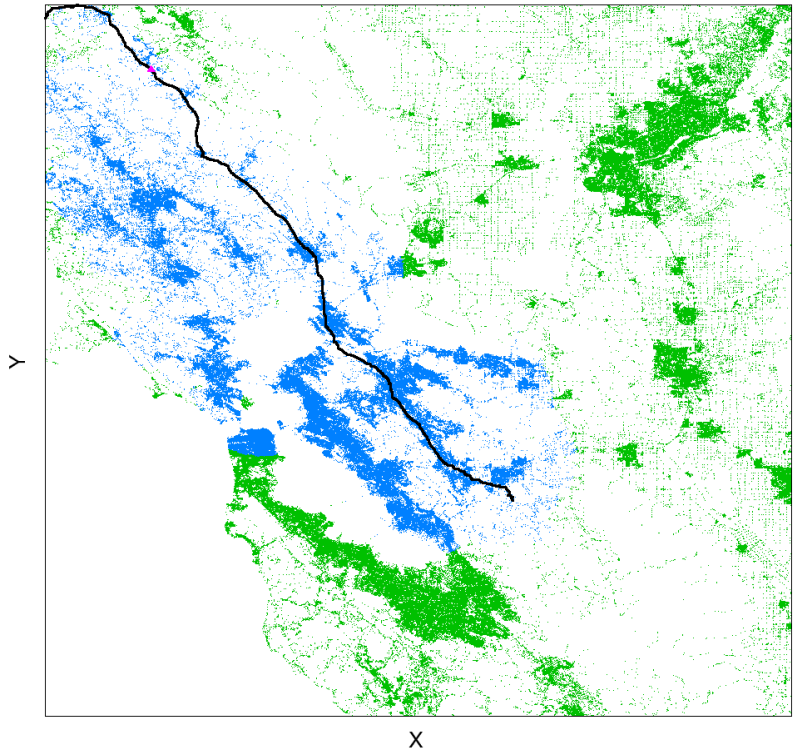
Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
73920	13255	266	266
-60665 pts		0	

São Francisco Bay Distância Médio



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
212411	21707	301	301
-190704 pts		0	

São Francisco Bay Distância Difícil



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
321270	133384	942	946
-187886 pts		+4 pts	

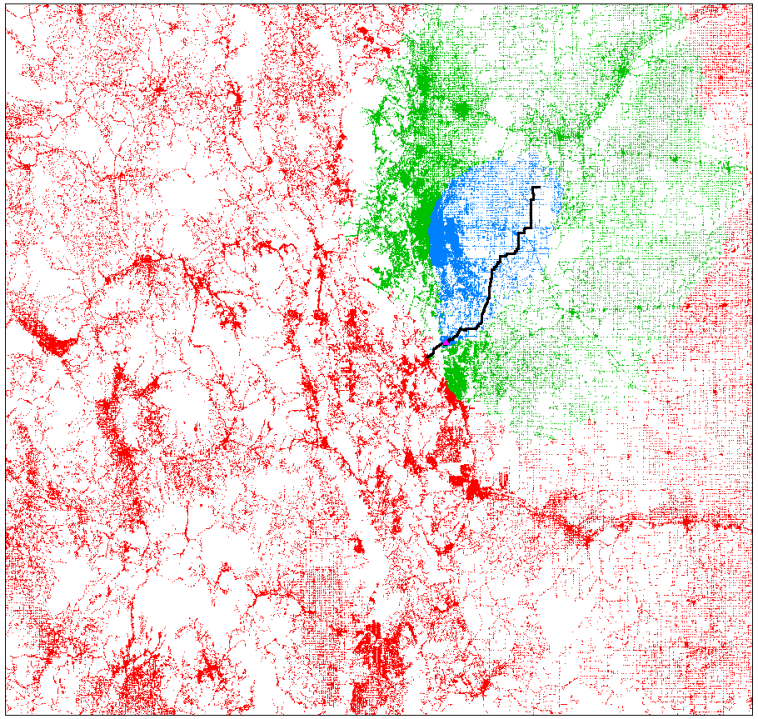
Resultados Computacionais – COL

Vértices: 435666 | Arestas: 1057066

COL.d	TEMPO DE EXECUÇÃO (Dijkstra | A*)		CUSTO (Dijkstra | A*)	
Fácil	3 mins 28 secs	46.38 secs	1804428	1804450
Médio	5 mins 26 secs	2 mins 22 secs	3616936	3619937
Difícil	7 mins 6 secs	5 mins 30 secs	7246198	7246883

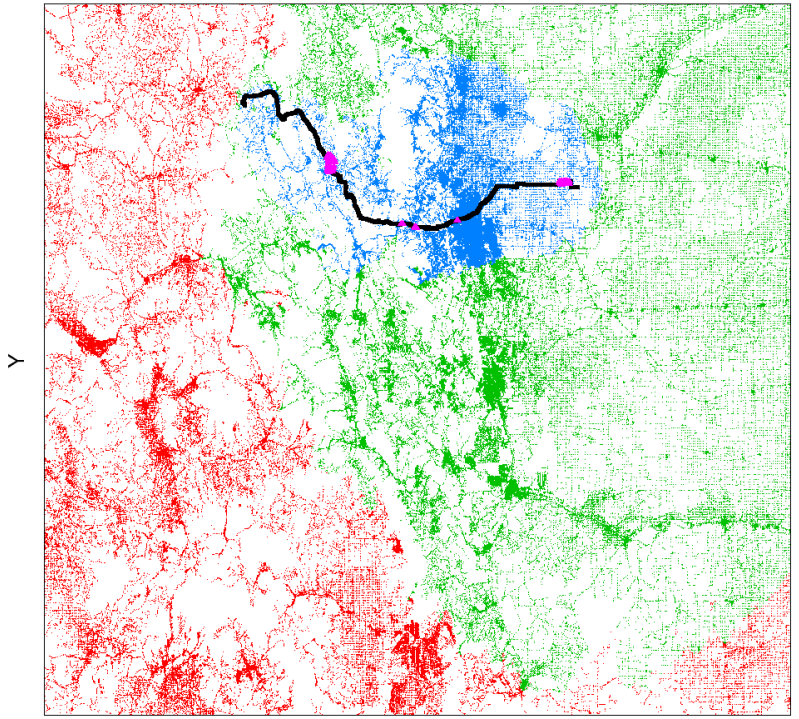
Resultados Computacionais – COL.d

Colorado Distância Fácil



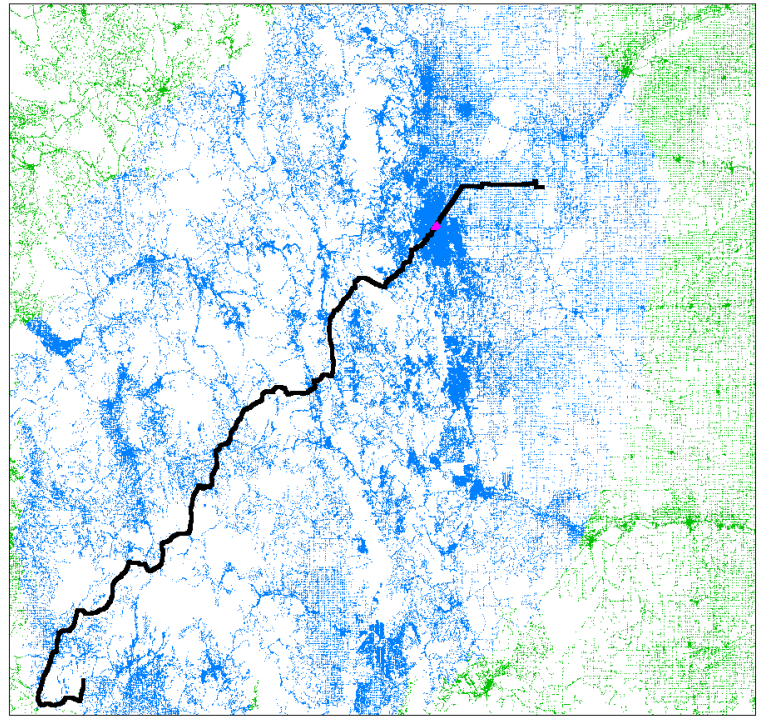
Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
199934	54803	337	336
-145131 pts		-1 pt	

Colorado Distância Médio



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
321101	150916	813	826
-170185 pts		+13 pts	

Colorado Distância Difícil



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
435666	382520	1750	1753
-53146 pts		+3 pts	

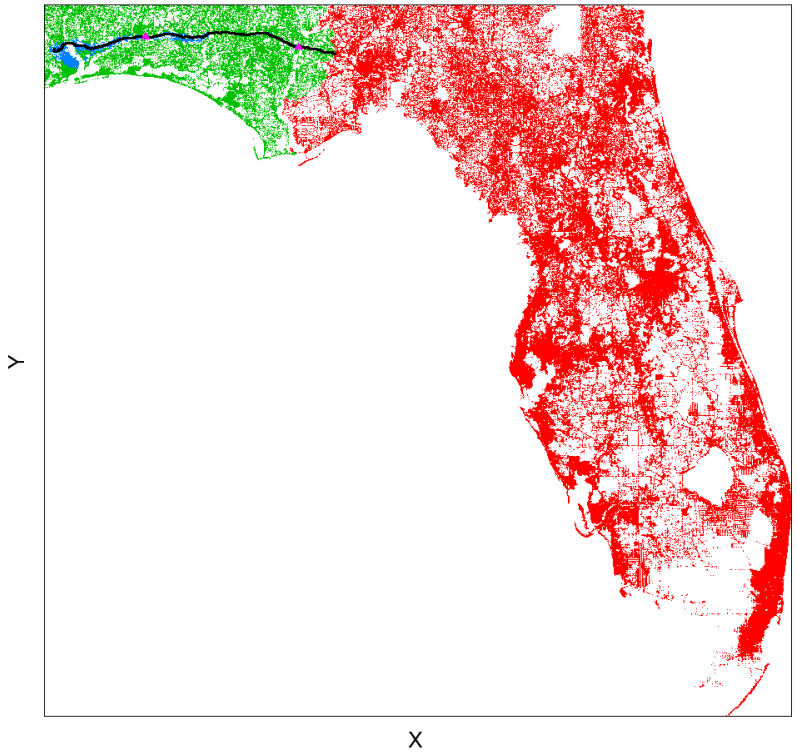
Resultados Computacionais – FLA

Vértices: 1070376 | Arestas: 2712798

FLA.d	TEMPO DE EXECUÇÃO (Dijkstra | A*)		CUSTO (Dijkstra | A*)	
Fácil	3 mins 13 secs	25.65 secs	2907930	2908133
Médio	8 mins 17 secs	6 mins 14 secs	5829987	5830874
Difícil	30 mins 20 secs	37 mins 37 secs	11674195	11674195

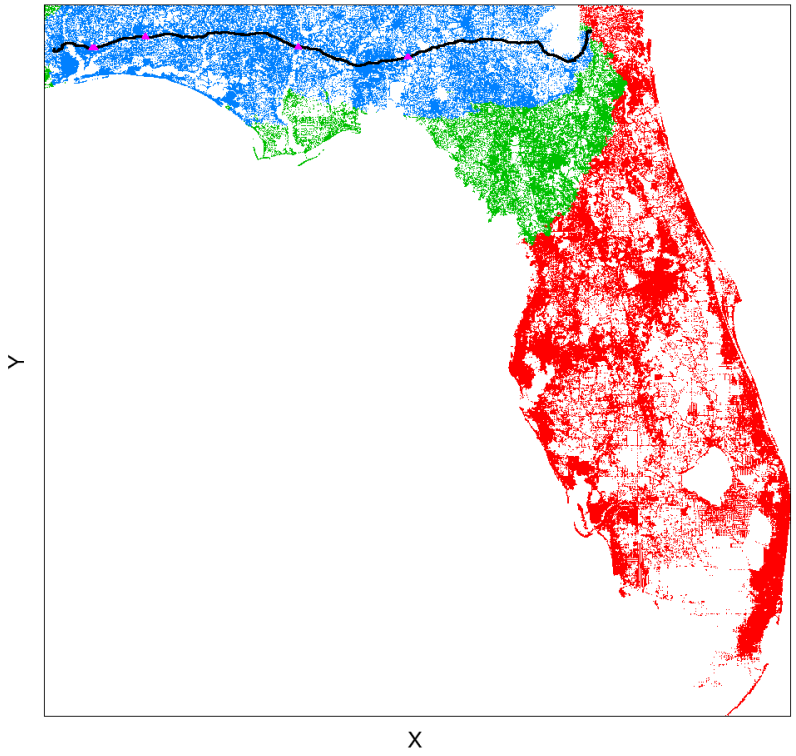
Resultados Computacionais – FLA.d

Flórida Distância Fácil



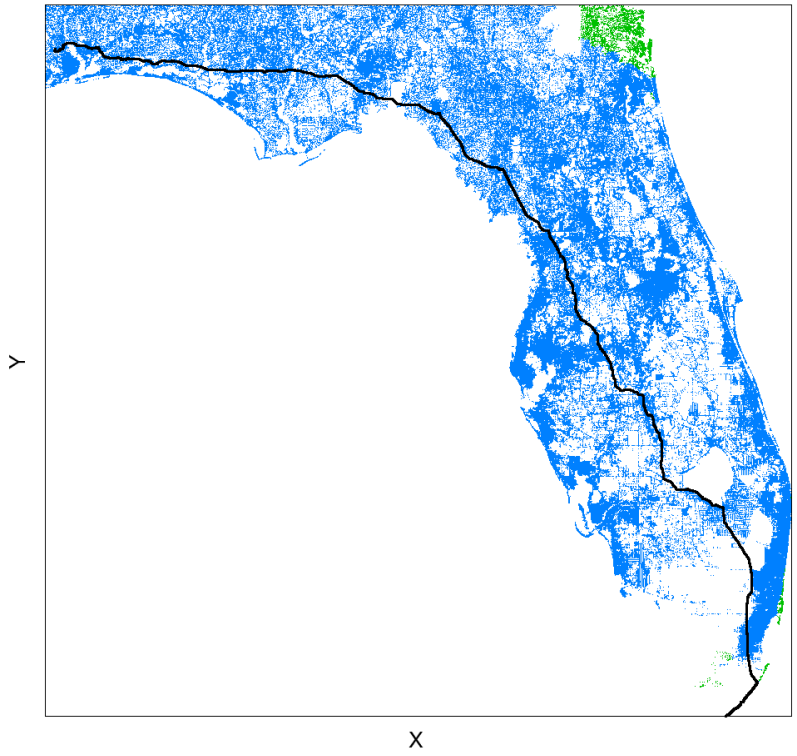
Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
110581	13309	768	766
-97272 pts		-2 pts	

Flórida Distância Médio



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
281082	186903	1363	1356
-94179 pts		-7 pts	

Flórida Distância Difícil



Pontos Visitados (Dijkstra | A*)		Tamanho Caminho (Dijkstra | A*)	
1070376	1053326	3529	3529
-17050 pts		0	

Conclusões Obtidas

- Em 83,3% dos casos, o A* reduziu significativamente o tempo computacional, menos em FLA.d e NY.d difícil.
- A visitação de nós para chegar ao nó de origem tem uma redução significativa em todos os casos.
- Quando executado de um s_i para todos os outros, utilizando $h(n)$ como distância euclidiana é garantido o resultado ótimo.
- Uma melhoria para o código seria a implementação da minHeap binária.
- Código e instâncias se encontram disponíveis em: https://github.com/pepeaze/aStar

Referências

[1] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths”, IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[2] M. Buehler, K. Iagnemma, and S. Singh, Eds., The 2005 DARPA grand challenge: the great robot race, Springer-Verlag Berlin Heidelberg, 2007.

[3] M. Buehler, K. Iagnemma, and S. Singh, Eds., The DARPA urban challenge: autonomous vehicles in city traffic, Springer-Verlag Berlin Heidelberg, 2009.

[4] R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta, “Safe and reliable path planning for the autonomous vehicle verdino”, IEEE Intelligent Transportation Systems Magazine, vol. 8, no. 2, pp. 22–32, 2016.

[5] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. Van Covern, and M. Webster, “Odin: team victortango’s entry in the darpa urban challenge,” Journal of Field Robotics, vol. 25, no. 8, pp. 467–492, 2008.

[6] R. Kala and K. Warwick, “Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization,” Journal of Intelligent & Robotic Systems, vol. 72, no. 3–4, pp. 559–590, 2013.

[7] S. Yoon, S. Yoon, U. Lee, and D. H. Shim, “Recursive path planning using reduced states for car-like vehicles on grid maps”, IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 5, pp. 2797–2813, 2015.

[8] K. Chu, J. Kim, K. Jo, and M. Sunwoo, “Real-time path planning of autonomous vehicles for unstructured road navigation”, International Journal of Automotive Technology, vol. 16, no. 4, pp. 653–668, 2015.

[9] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments”, The International Journal of Robotics Research, vol. 29, no. 5, pp. 485–501, 2010.

DÚVIDAS???