UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Robótica Móvel

Planejamento de caminhos – Roadmaps

Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

VERLab
Visão e Robótica

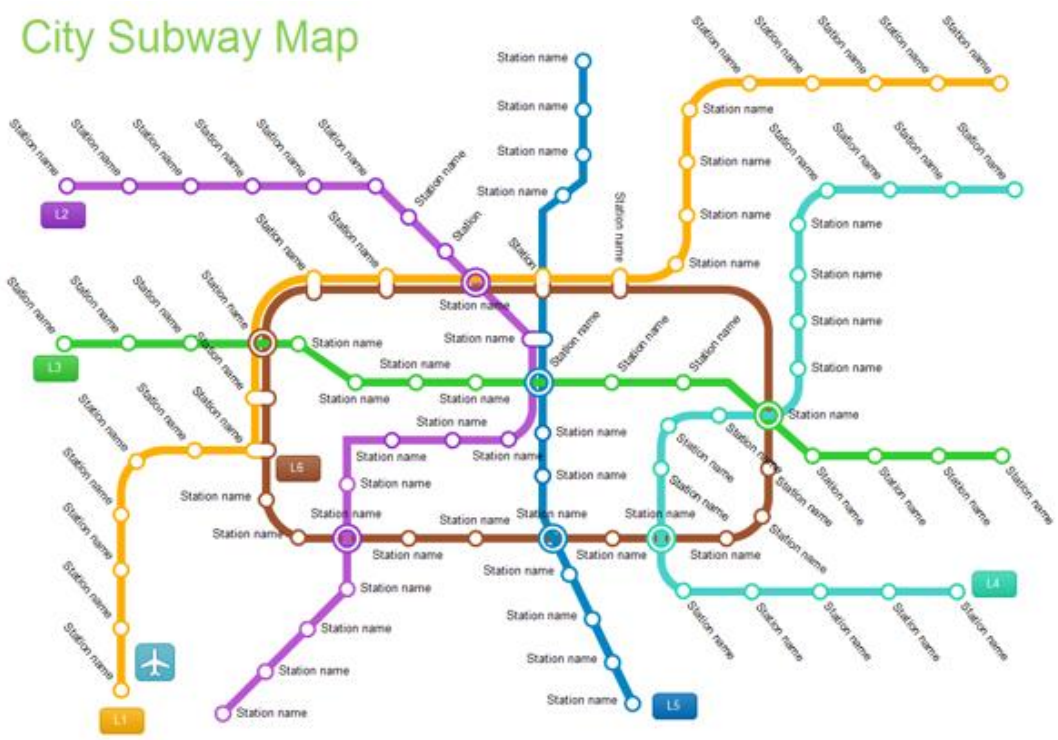
Introdução

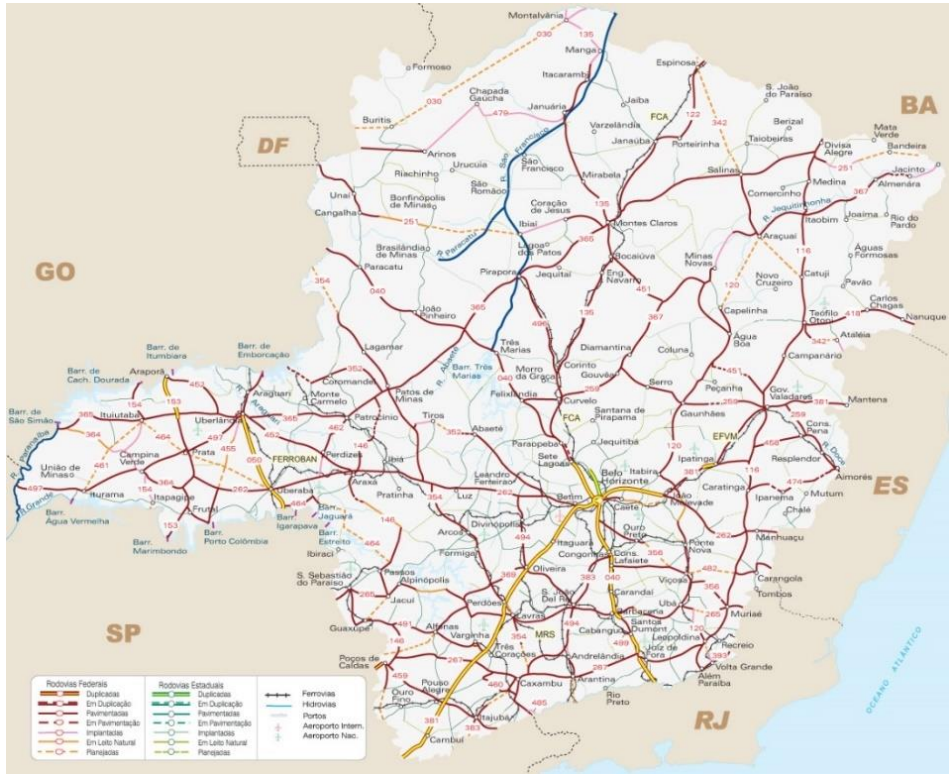
- Estratégias deliberativas
  - Planejamento mais extenso/complexo
  - Mais informação → Melhores decisões

- Qual tipo de informação nos ajudaria?
  - Representação → Mapa/modelo do mundo
  - Totalmente ou Parcialmente observável?
  - Dinâmico ou Estático?

Introdução

ROADMAPS





Introdução
Visão geral

Representação

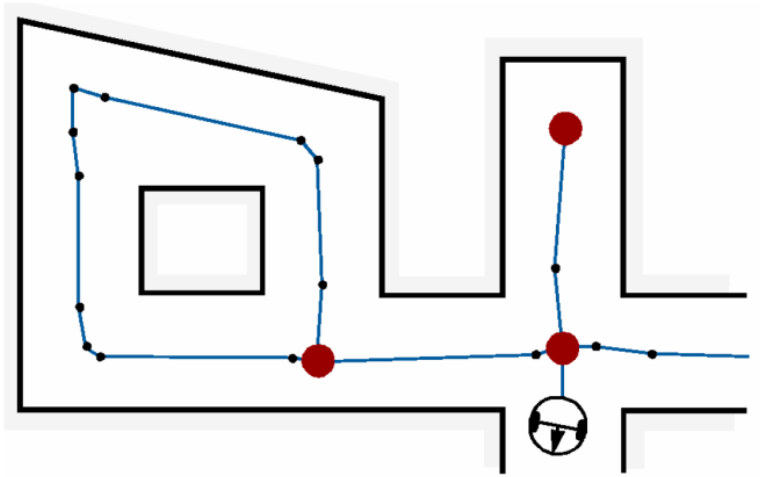
- Gerar um grafo a partir da criação de nós e arestas, ou seja, pontos ligados se houver um caminho livre entre eles.
- Caso necessário, conectar os pontos inicial/final ao grafo original.

Busca

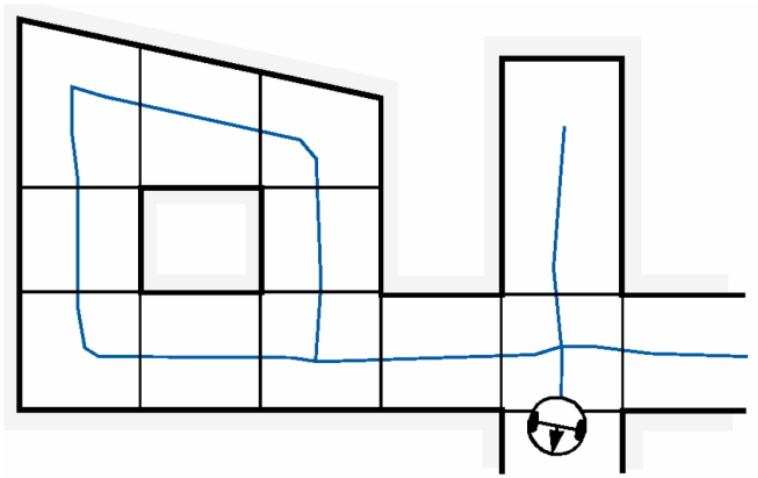
- Determinar o melhor caminho entre os pontos inicial/final dada uma métrica.

Introdução
Representação

- Representar a conectividade entre diferentes configurações em $\mathcal{W}$ ou $\mathcal{C}_{free}$ inserindo pontos ou dividindo o ambiente



Roadmap



Decomposição

Introdução
Representação

- Diferentes métodos para gerar o grafo
- **Determinísticos**
  - Grafo de visibilidade
  - Decomposição em células
- **Probabilísticos**
  - Probabilistic Roadmaps (PRM)
  - Rapidly-Exploring Random Tree (RRT)

Introdução
Busca

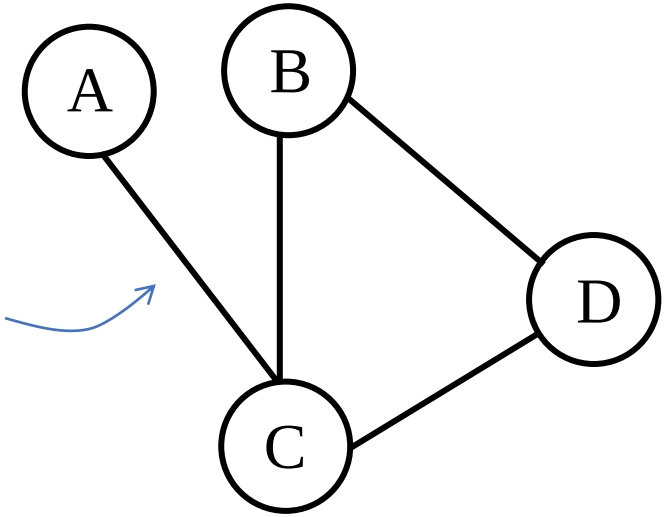
- Algoritmos completos
  - Se existe um caminho no grafo ligando a posição inicial e a posição final desejada esse caminho será encontrado

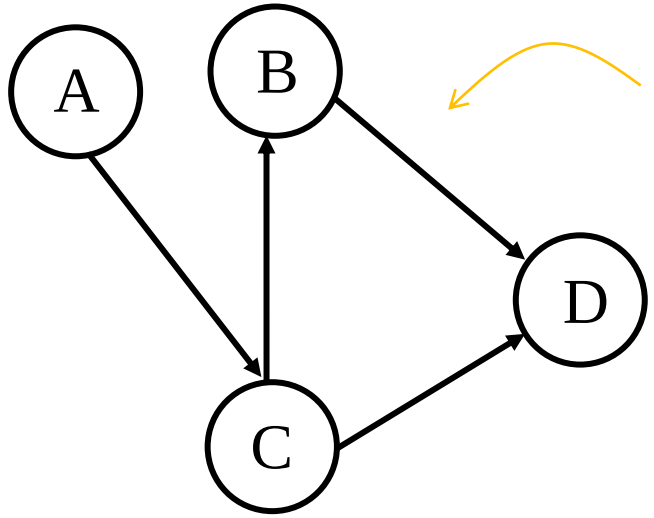
- Algoritmos ótimos
  - O caminho encontrado será o melhor possível (ótimo) de acordo com uma certa métrica escolhida (função de custo)

Grafos

- Grafo G é coleção de vértices (nós) V e arestas E → $G = (V, E)$
- Grafo não-direcionado x Grafo direcionado

As arestas não possuem sentido



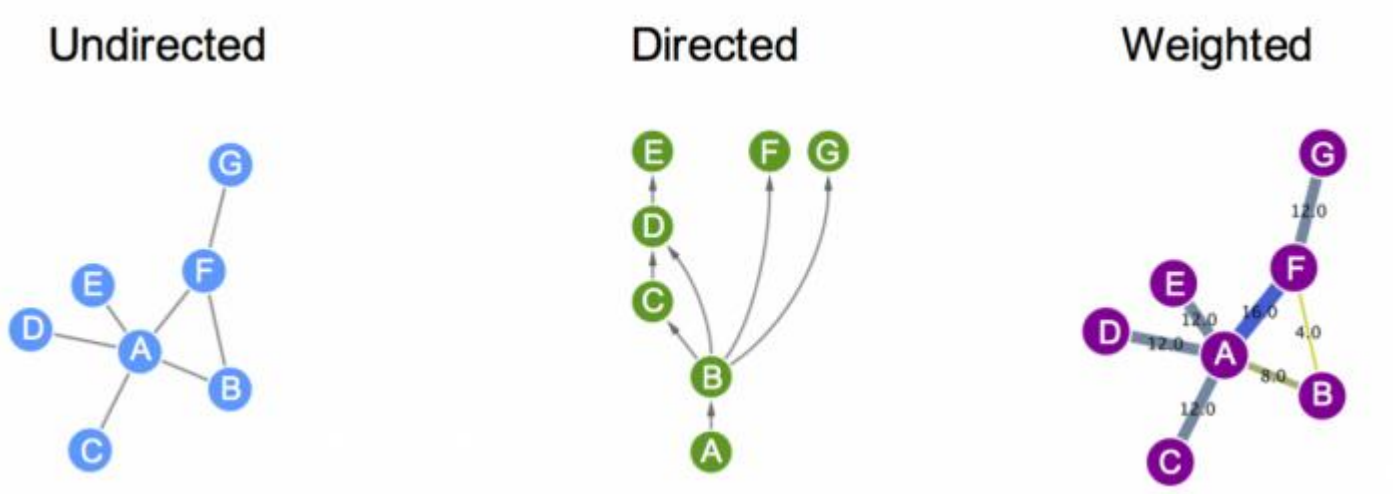


As arestam possuem restrições (sentidos) de conectividade

Grafos

- Arestas podem ser anotadas com valores w
  - Geralmente denominados pesos ou custos
- Um caminho é uma sequência de vértices tal que para dois vértices adjacentes $\{v_i, v_{i+1}\}$ existe uma aresta $e_{i,i+1}$
- Um grafo é dito conexo (conectado) se existir pelo menos um caminho ligando qualquer par de vertices distintos
- Matriz de adjacência: representação onde o valor a_{ij} guarda informações sobre como os vértices v_i e v_j estão relacionados

Grafos



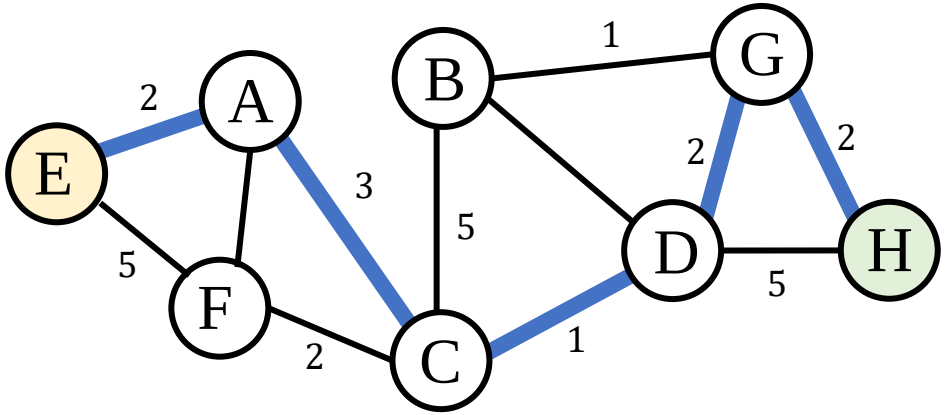
	A	B	C	D	E	F	G
A	0	1	1	1	1	1	0
B	1	0	0	0	0	1	0
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0
E	1	0	0	0	0	0	0
F	1	1	0	0	0	0	1
G	0	0	0	0	0	1	0

	A	B	C	D	E	F	G
A	0	1	0	0	0	0	0
B	0	0	1	1	0	1	1
C	0	0	0	1	0	0	0
D	0	0	0	0	1	0	0
E	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0

	A	B	C	D	E	F	G
A	0	8	12	12	12	16	12
B	8	0	∞	∞	∞	4	∞
C	12	∞	0	∞	∞	∞	∞
D	12	∞	∞	0	∞	∞	∞
E	12	∞	∞	∞	0	∞	∞
F	16	4	∞	∞	∞	0	12
G	12	∞	∞	∞	∞	12	0

Grafos

- O custo total de um caminho no grafo é calculado pelo somatório de todos os pesos das arestas que o formam



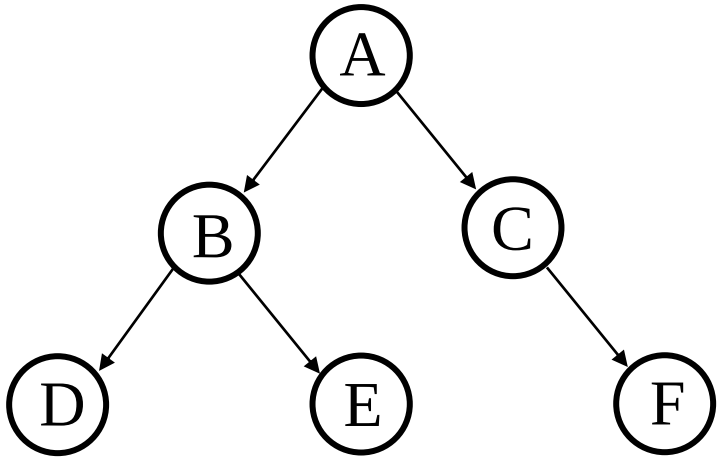
Caminho: E-A-C-D-G-H

Custo: 10

- Algoritmos para determinar caminhos: Dijkstra, A*, etc

Grafos

- Ciclo: caminho onde o primeiro e último vértice são iguais
- Árvore: de forma geral, uma árvore é definida como um grafo que não possui nenhum ciclo entre seus vértices



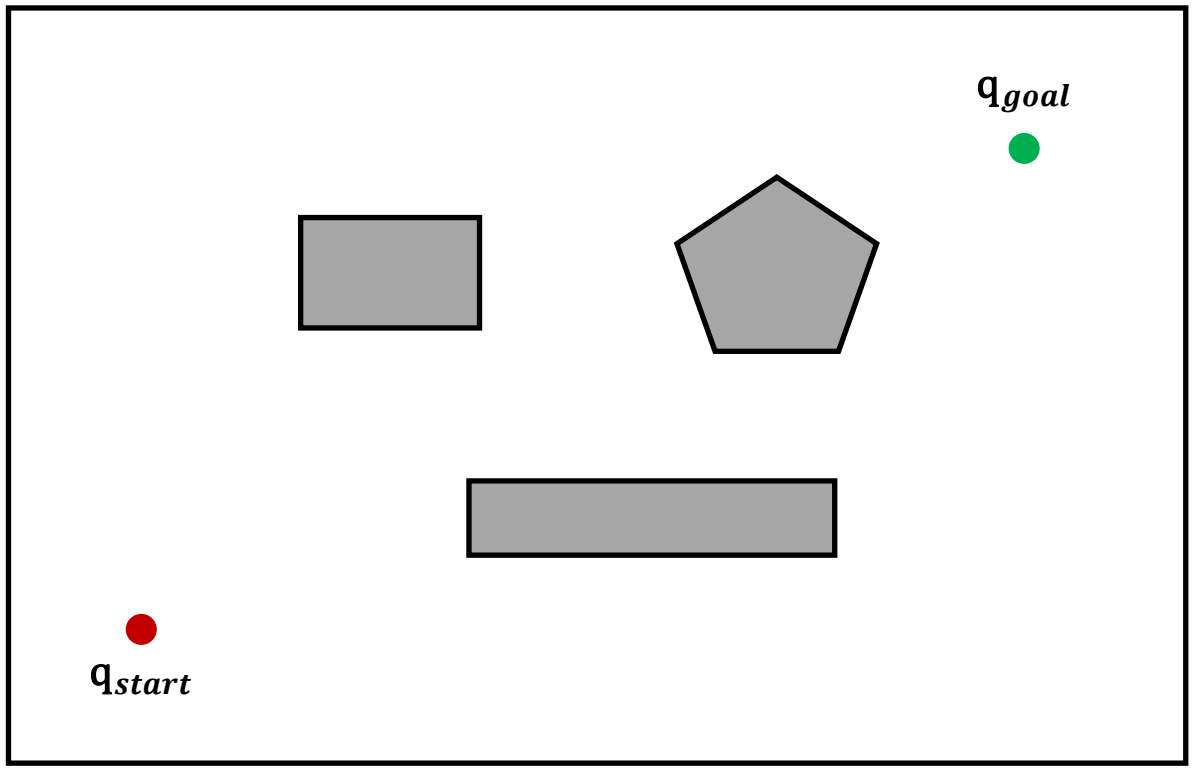
Roadmaps

- Representação do ambiente utilizando um grafo
  - Analogia com um mapa rodoviário (mapa de rotas)
  - Vértices indicam posições/configurações livres
  - Arestas indicam caminhos livres de colisão
- Caso necessário, o robô deve saber "entrar" e "sair" do roadmap, além de conseguir navegar entre os vértices
- Vantagens
  - Facilita a realização de várias consultas no mesmo mapa

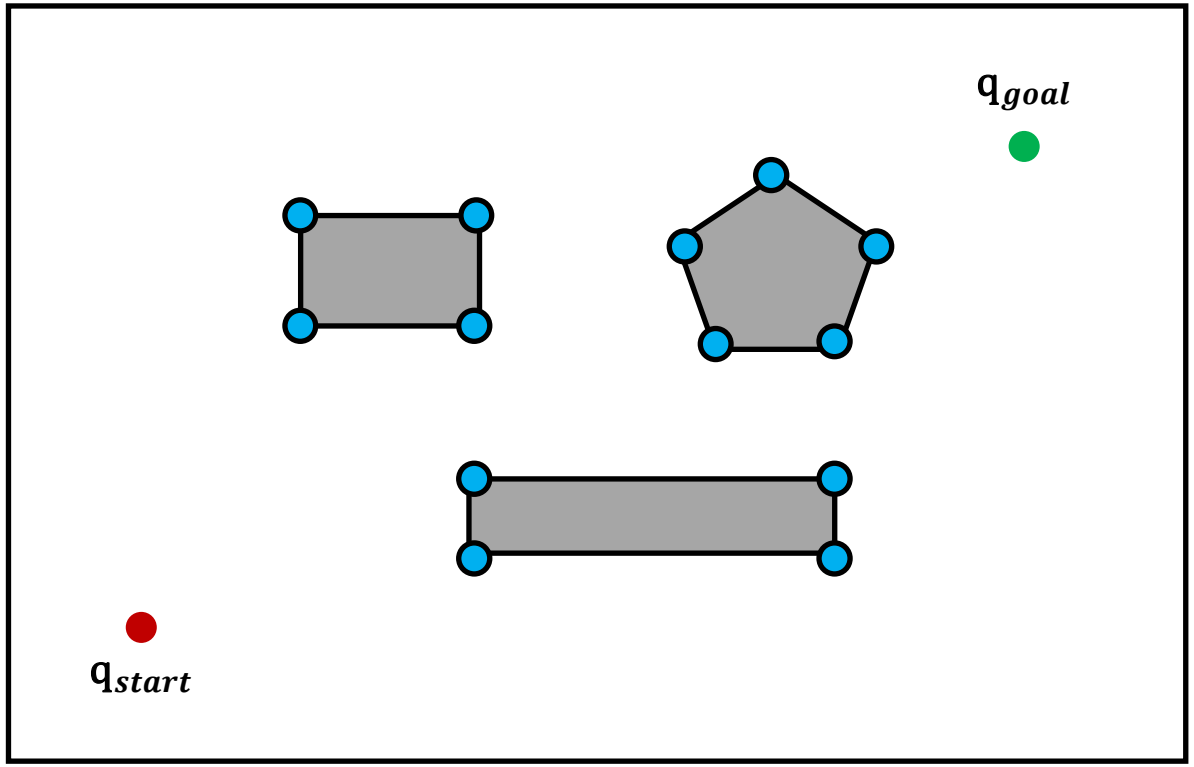
Roadmaps
Grafo de Visibilidade

- Ambiente composto de obstáculos poligonais
- Vértices do grafo correspondem às quinas dos obstáculos
  - Condições para criar uma aresta conectando vértices
    - Se pertencem à mesma borda (lado) do obstáculo
    - Se a aresta não intercepta nenhum outro obstáculo
- Adicionar os vértices start/goal ao grafo inicial
- O caminho encontrado é o menor possível

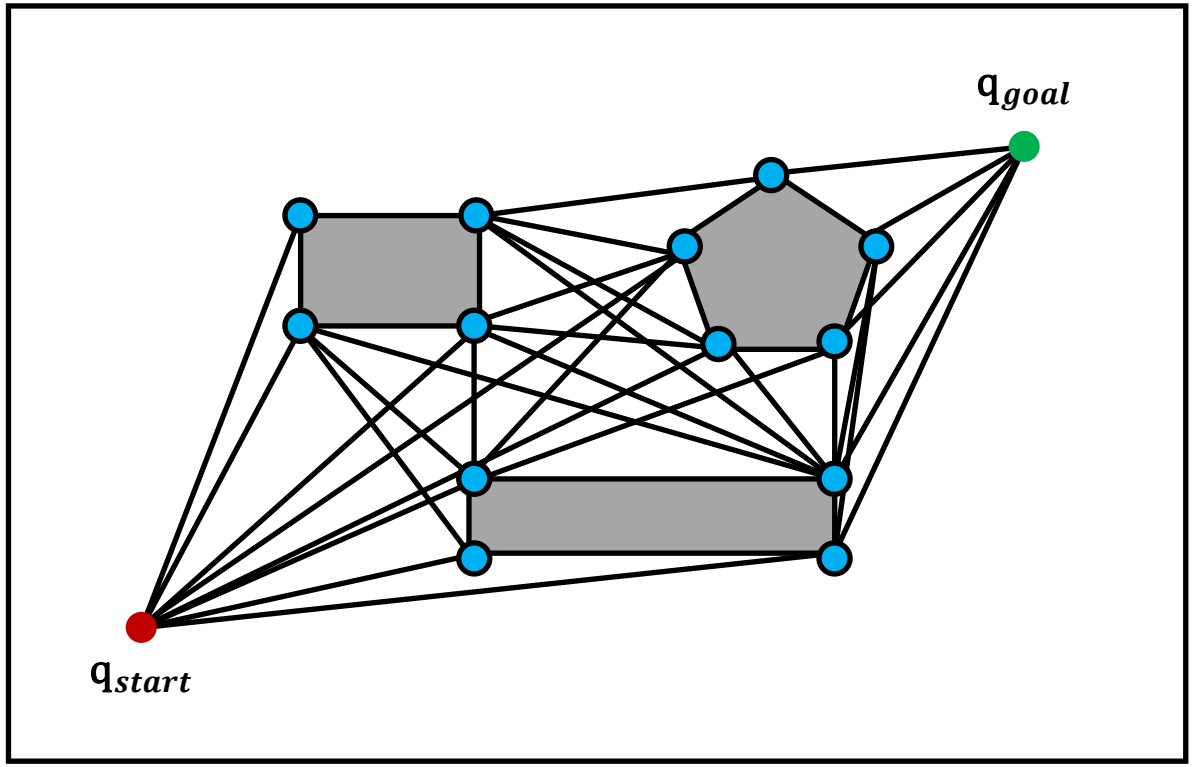
Roadmaps
Grafo de Visibilidade



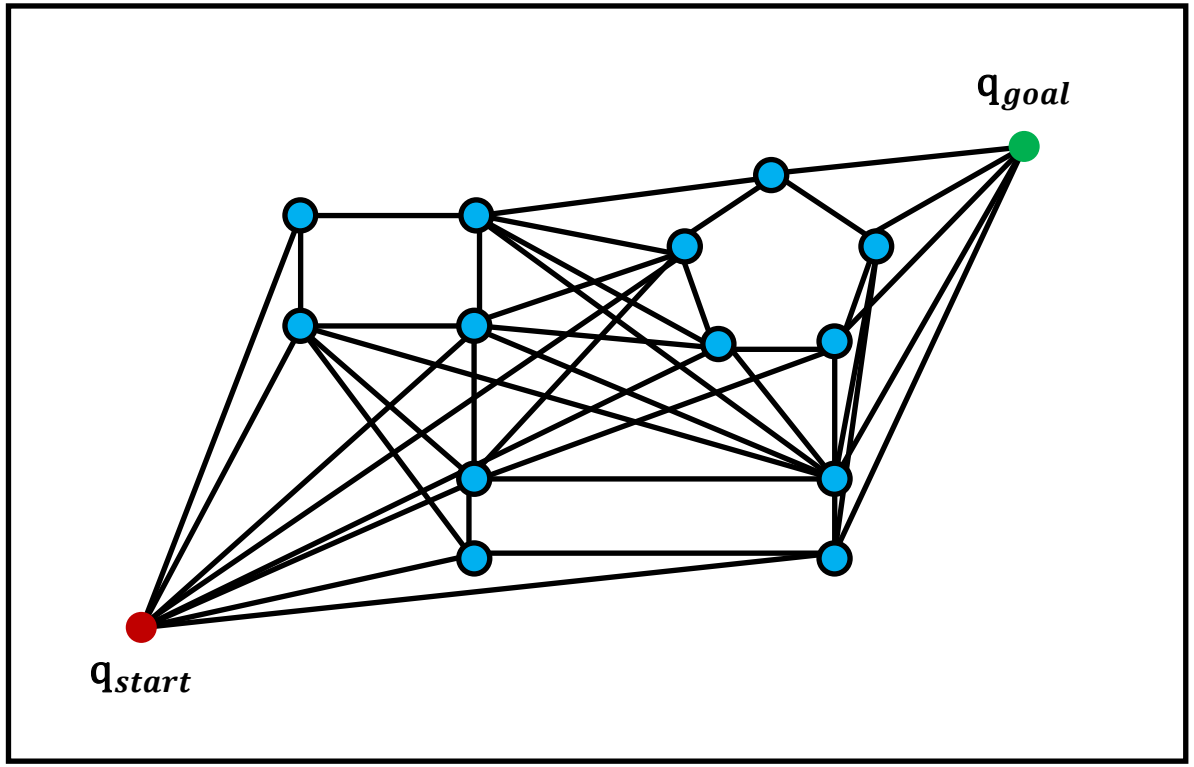
Roadmaps
Grafo de Visibilidade



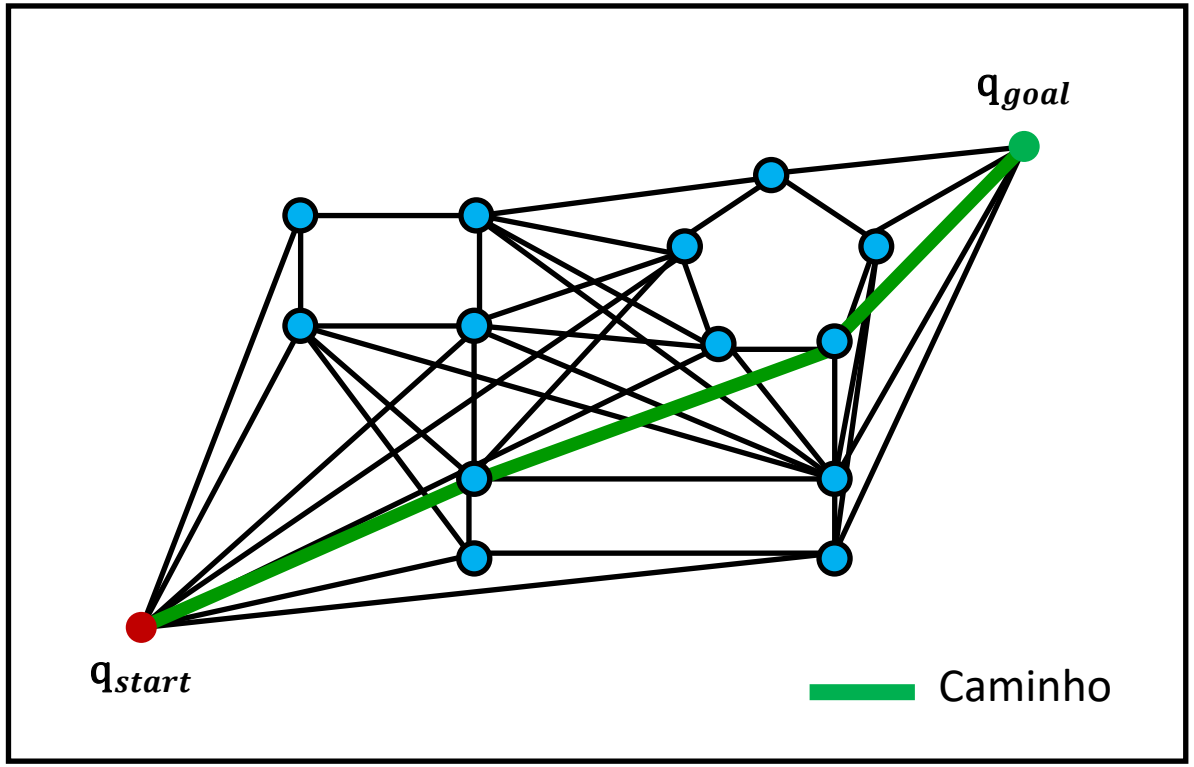
Roadmaps
Grafo de Visibilidade



Roadmaps
Grafo de Visibilidade

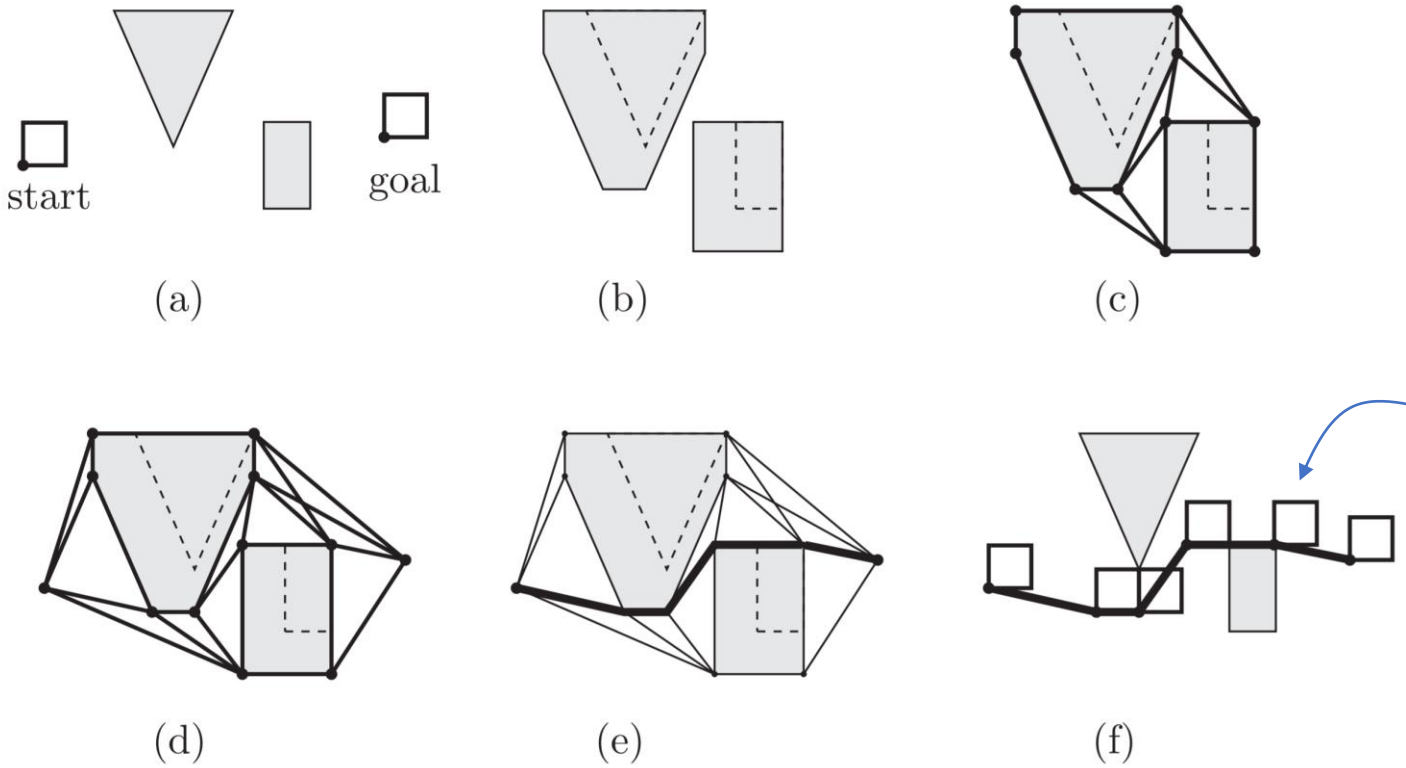


Roadmaps
Grafo de Visibilidade



Roadmaps
Grafo de Visibilidade

- De maneira mais geral podemos considerar o $\mathcal{C}$-space



Lembrando que no $\mathcal{C}$-space o caminho é contínuo e representa uma sequência exata de posições/orientações a seguir.

Fonte: *Modern Robotics Mechanics, Planning, And Control*

Roadmaps
Grafo de Visibilidade

- Vantagens
  - Metodologia simples
  - Completo
  - Ótimo (em termos de distância)

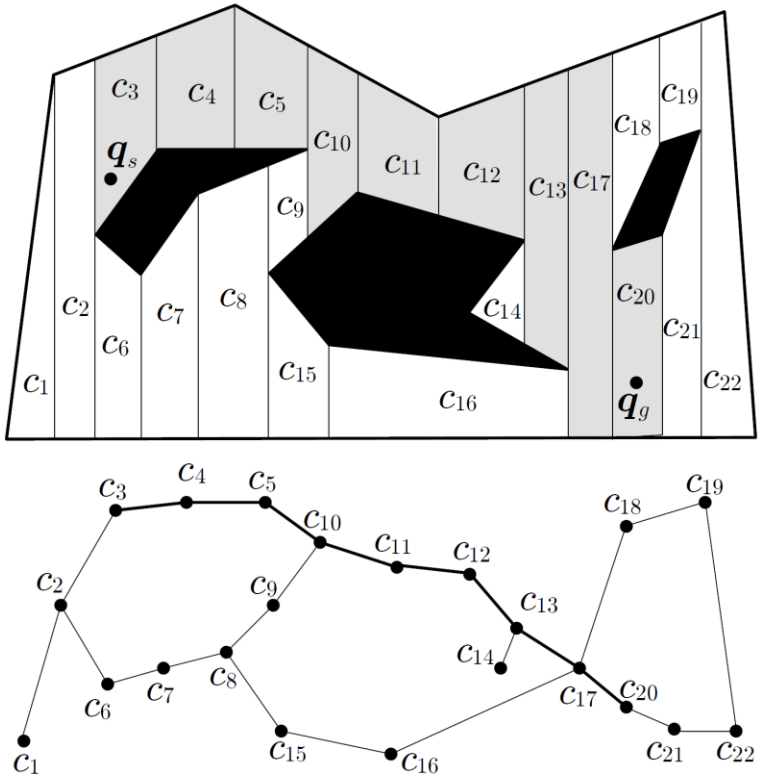
- Desvantagens
  - Grande número de arestas
  - Caminhos muito próximos dos obstáculos

Roadmaps
Decomposição em células

- Dividir o espaço em regiões conectadas
  - Regiões denominadas células, diferentes formatos
- Criar um grafo representando a conectividade
  - Adicionar um vértice para cada célula
  - Determinar as células que são adjacentes (mesma borda)
  - Adicionar as respectivas arestas entres células
- Encontrar as células (vértices) que possuem q_{start} e q_{goal}
- Determinar o caminho que conecta essas células (vértices)

Roadmaps
Decomposição em células (Exata/Trapezoidal)

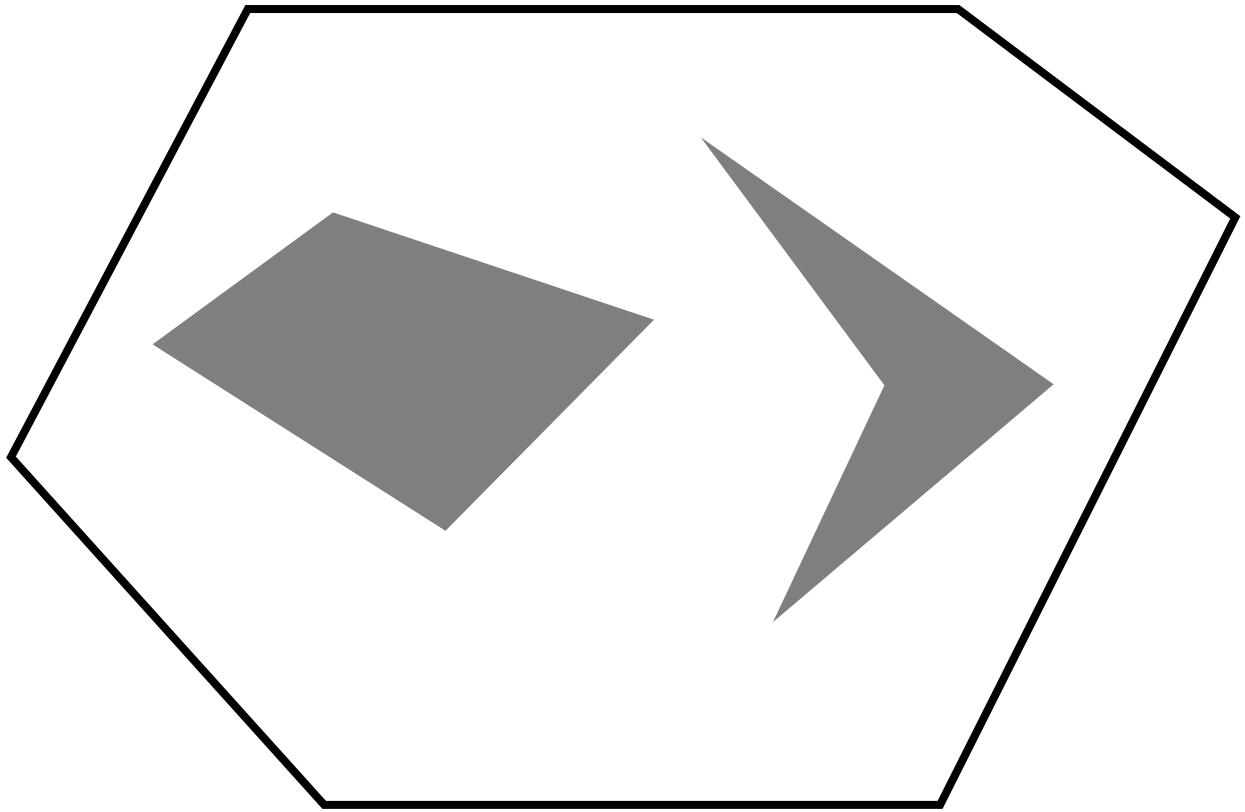
- Divide o espaço traçando linhas verticais superiores/inferiores a partir de quinas
  - Células serão Trapézios ou Triângulos
- Todas as células são espaços livres
- Em ambientes esparsos, o número de células será pequeno, mesmo que o tamanho do ambiente seja muito grande



Fonte: *Robotics Modelling, Planning and Control*

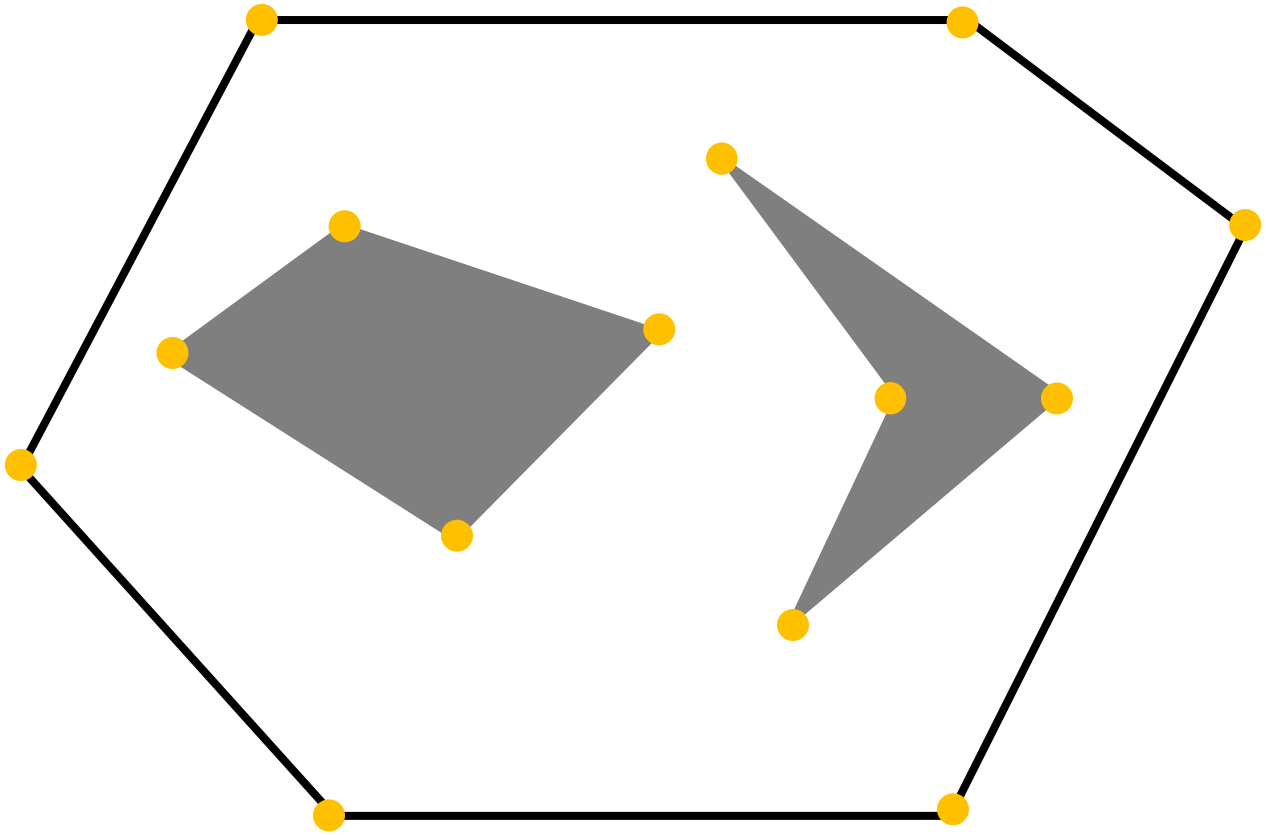
Roadmaps

Decomposição em células (Exata/Trapezoidal)



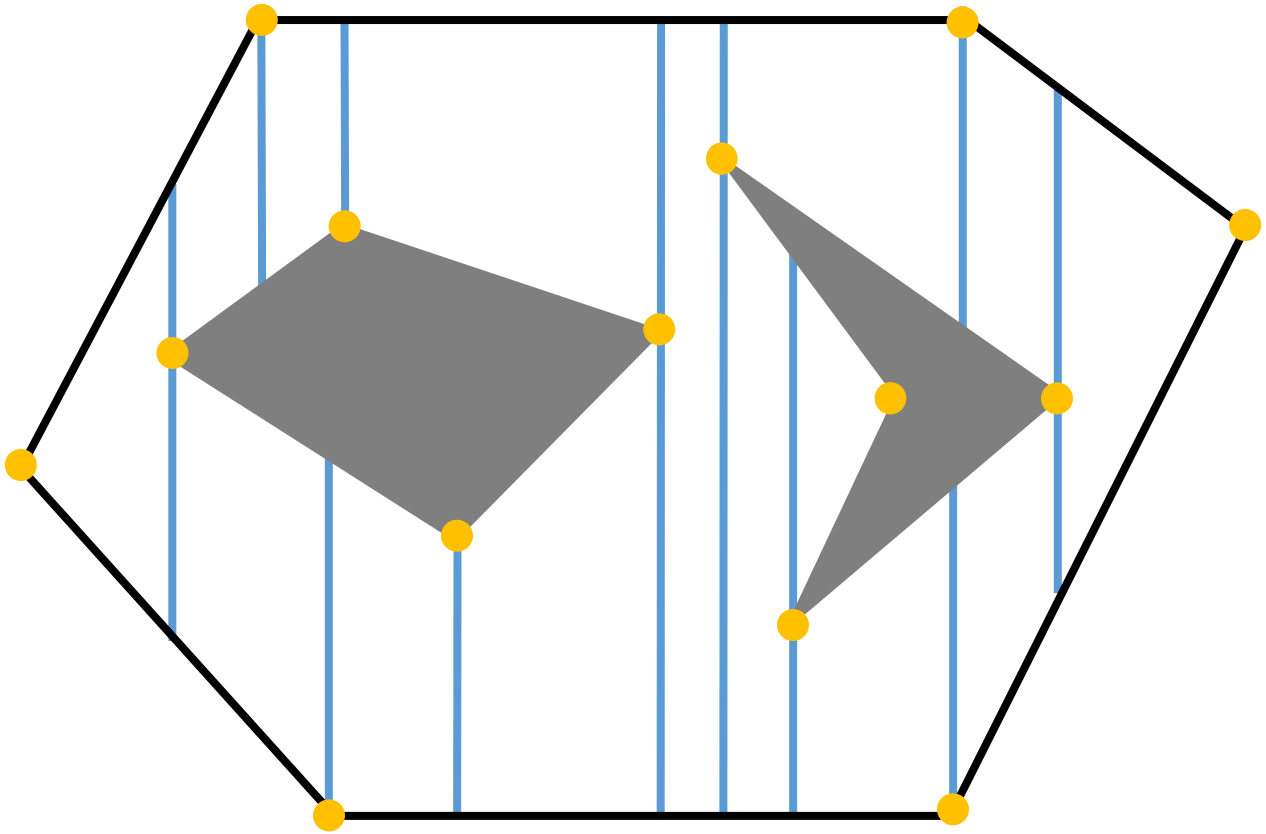
Roadmaps

Decomposição em células (Exata/Trapezoidal)



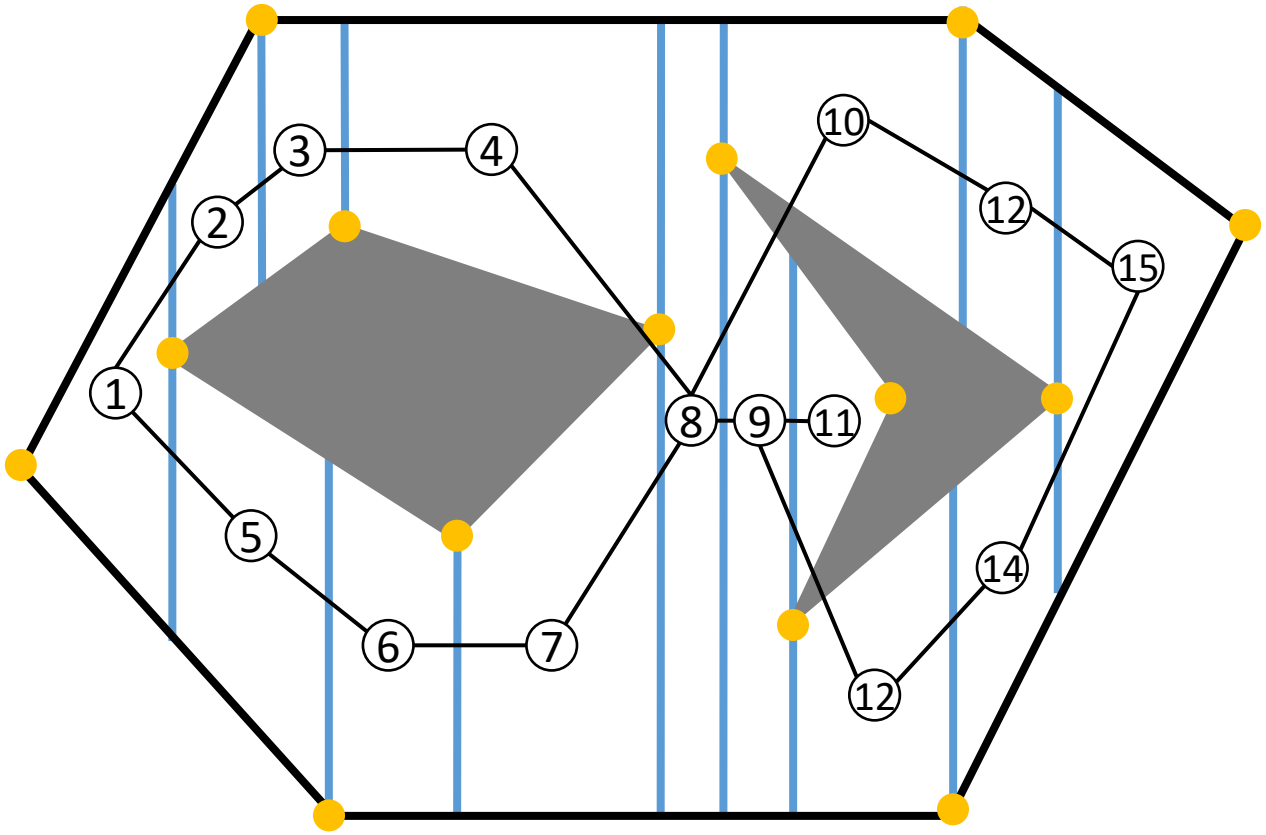
Roadmaps

Decomposição em células (Exata/Trapezoidal)



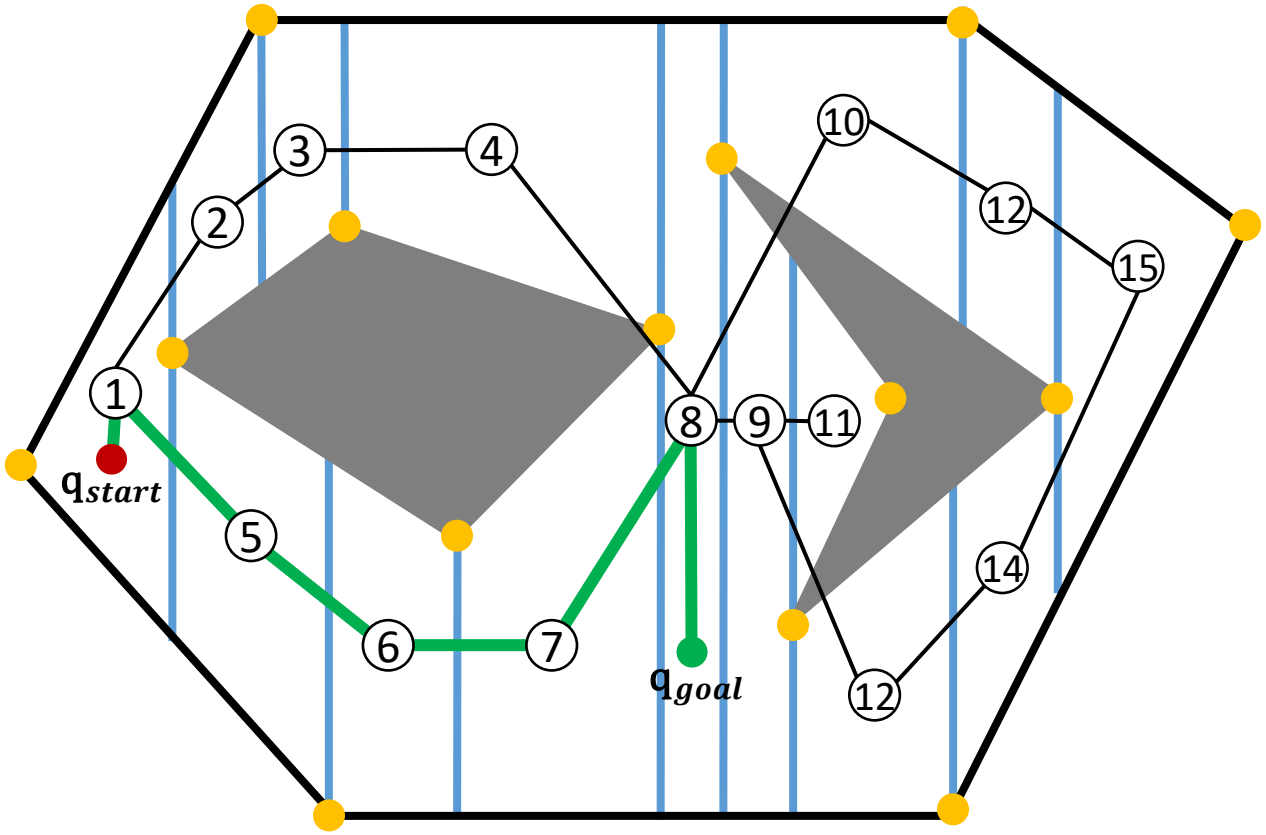
Roadmaps

Decomposição em células (Exata/Trapezoidal)



Roadmaps

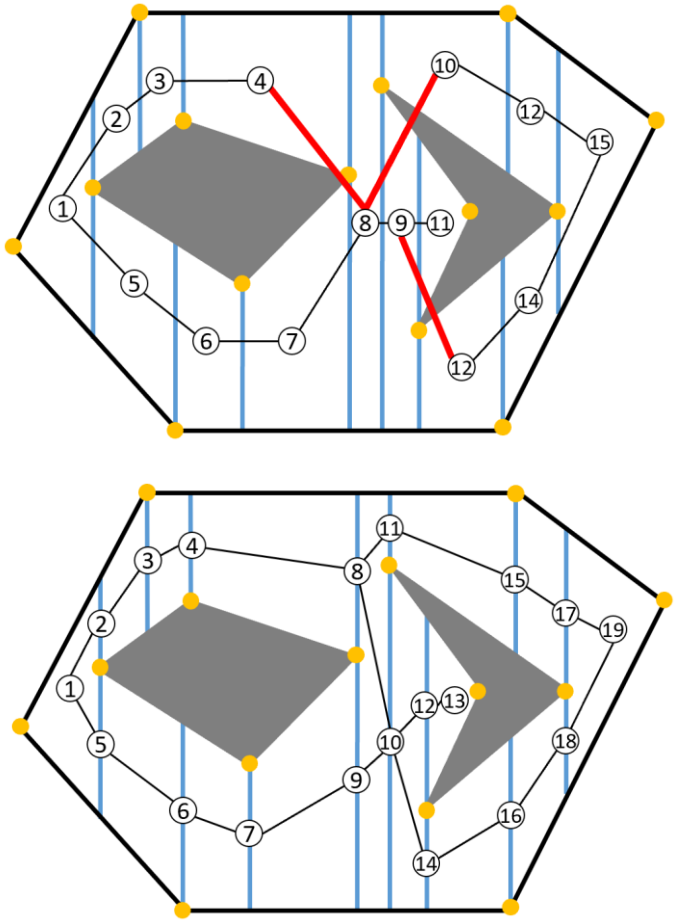
Decomposição em células (Exata/Trapezoidal)



Roadmaps

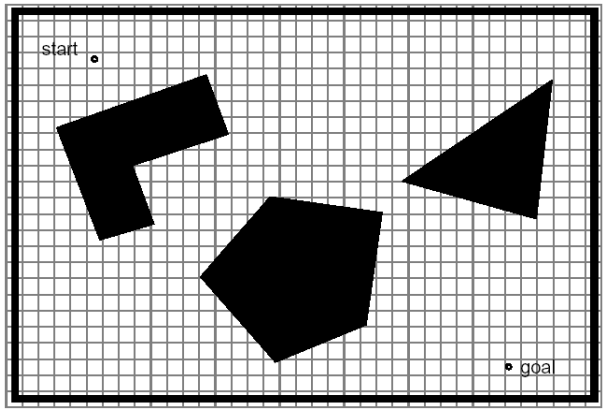
Decomposição em células (Exata/Trapezoidal)

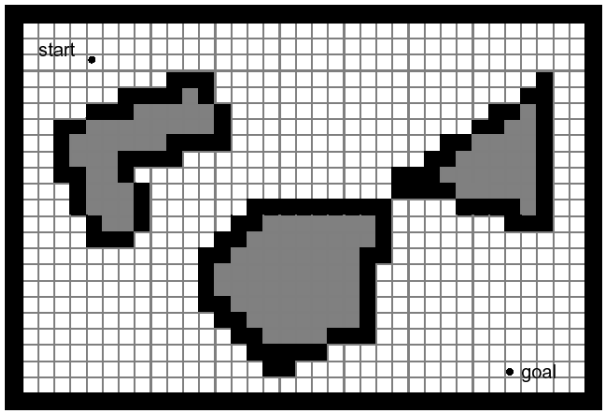
- A posição particular do robô na célula não importa, desde que ele consiga navegar entre células adjacentes
- Entretanto, determinar uma posição específica pode facilitar a navegação
- Pontos médios das bordas das células



Roadmaps
Decomposição em células (Aproximada/Grid)

- Não é feita uma decomposição exata do espaço livre, mas são criadas células (fixas) para o espaço independente dos obstáculos
- Célula marcada como totalmente ocupada mesmo se possui apenas parte do obstáculo
- A definição de célula "vizinha" deve ser escolhida: o robô é restrito a se mover em direções alinhadas aos eixos ou pode mover em várias dimensões simultaneamente?

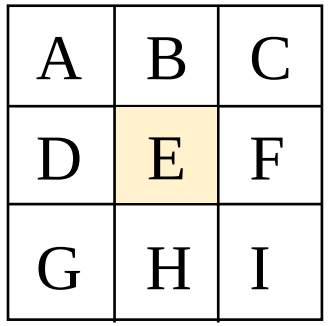




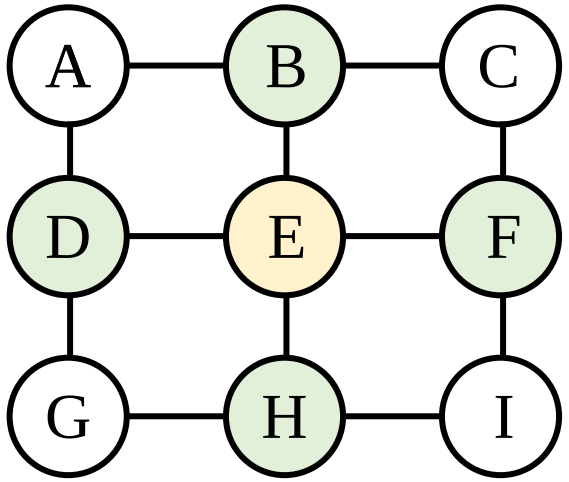
Fonte: *Introduction to Autonomous Mobile Robots*

Roadmaps

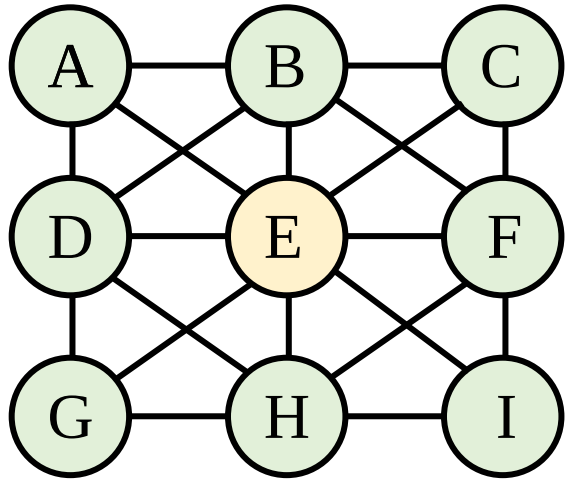
Decomposição em células (Aproximada/Grid)



Grid



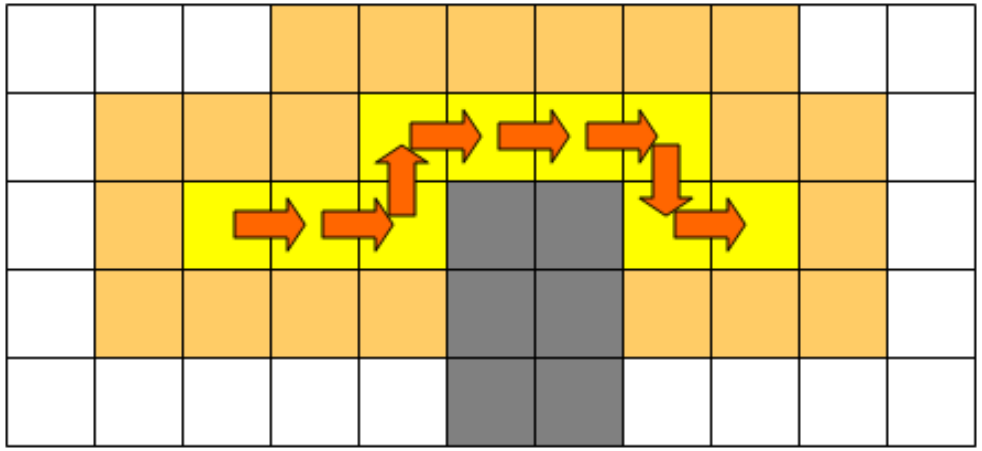
Conectividade 4



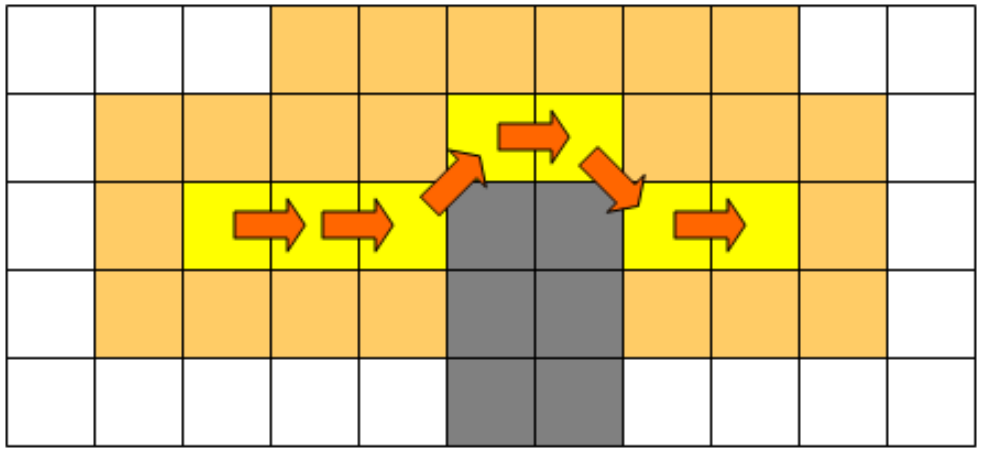
Conectividade 8

Roadmaps
Decomposição em células (Aproximada/Grid)

- Se apenas movimentos alinhados forem usados, o custo de deslocamento deve se basear na distância de Manhattan, não apenas na distância euclidiana
- Questões práticas de movimentos na diagonal também devem ser tratadas



Conectividade 4

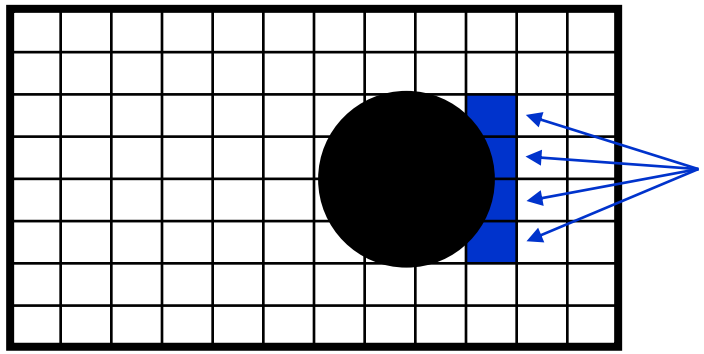


Conectividade 8

https://en.wikipedia.org/wiki/Taxicab_geometry

Roadmaps
Decomposição em células (Aproximada/Grid)

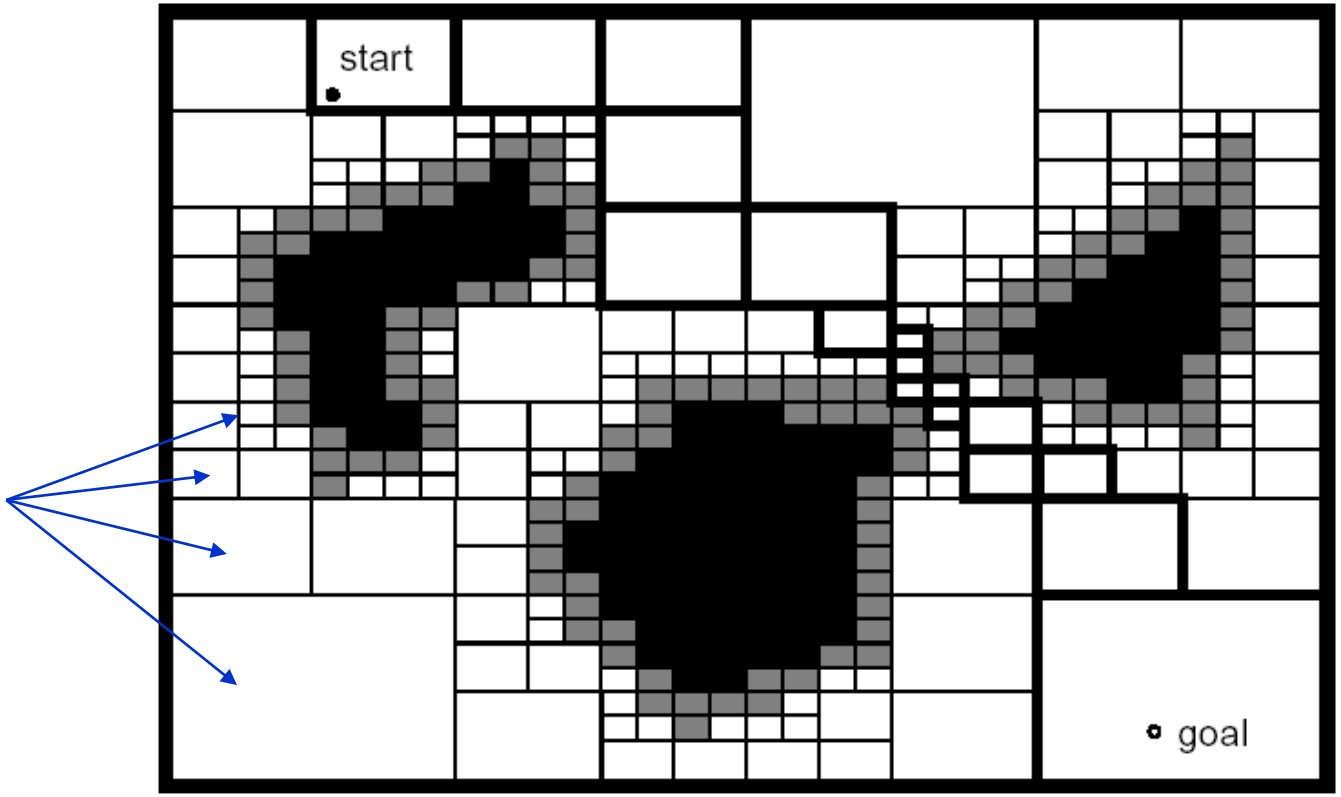
- Problemas
  - Continuidade do trajeto depende da resolução
  - Complexidade computacional / Resolução
  - Inexatidão (simplificação)



Essas células estão ocupadas ou não?

Roadmaps
Decomposição em células (Aproximada/Grid)

Grid multi-resolução



Fonte: *Introduction to Autonomous Mobile Robots*

Pesquisa em grafos

- Algoritmos básicos
  - Pesquisa em largura
    - Expande todos os filhos
  - Pesquisa em profundidade
    - Expande o primeiro filho

- Algoritmos específicos de caminho mínimo
  - Dijkstra
  - A* (D*)

Algoritmo A*

- Busca informada
- Utiliza conhecimento específico do problema na escolha do próximo nó a ser expandido
  - Auxílio de uma função heurística
  - Indica o quão promissor é um determinado nó

Dica: https://youtu.be/-L-WgKMFuhE

Algoritmo A*

- Combinação de diferentes estratégias
  - Busca gulosa
    - Eficiente
    - Não é completa e não é ótima
  - Busca de custo uniforme
    - Ineficiente
    - Completa e ótima

Algoritmo A*

- Escolha do nó a ser expandido é dado por

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

  - $g(n)$: custo de chegar até o vértice atual
  - $h(n)$: estimativa do custo do vértice atual ao goal
- Logo, $f(n)$ fornece uma estimativa do custo da melhor solução que passa pelo vértice n

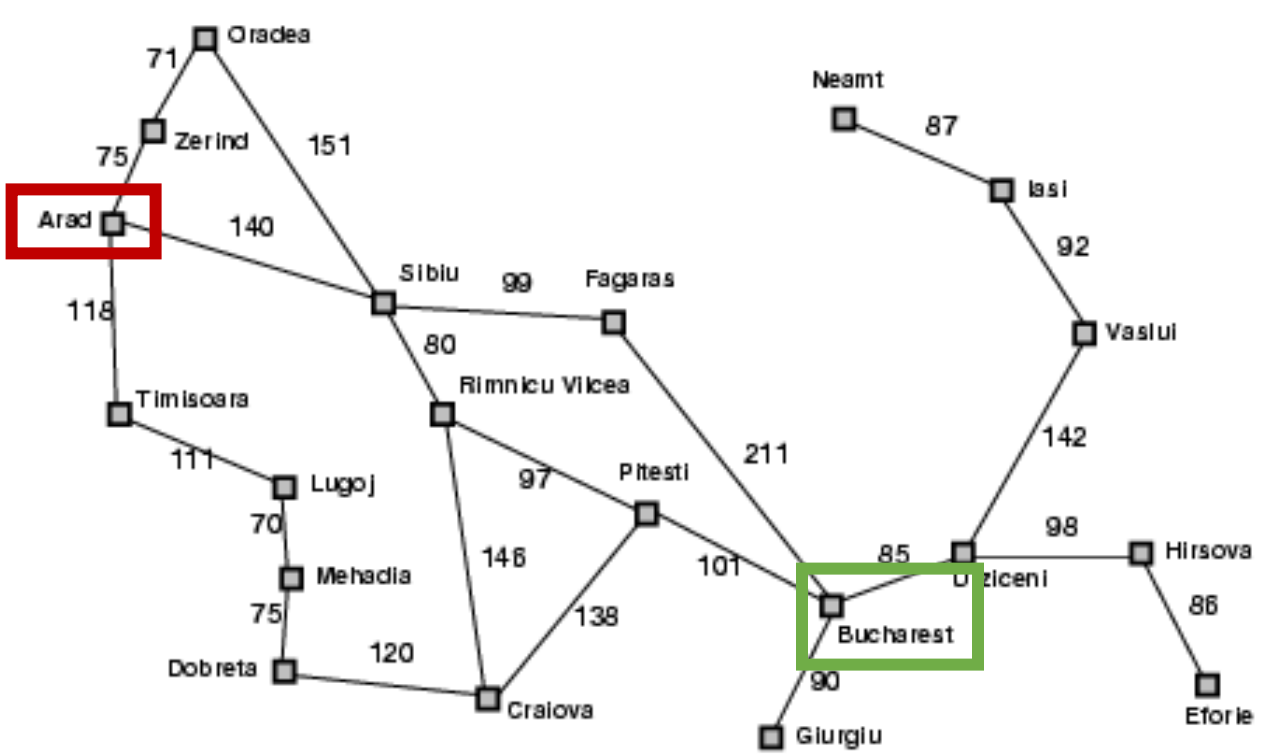
Algoritmo A*
Função $h(n)$

- Heurística
  - Tenta adivinhar o custo de n ao goal
- Pode-se escolher qualquer valor?
  - Não!
  - Necessário que a heurística seja <u>admissível</u>
    - $h(n) \leq h^*(n)$: $h^*(n)$ representa o custo real de n até o goal

Algoritmo A*
Função $h(n)$

- Heurística admissível
  - Nunca superestimar o custo real da solução
  - Garante a otimalidade da solução encontrada
  - Ex: distância euclidiana (linha reta)
    - Caminho real nunca menor, mantém admissibilidade
- Complexidade depende da heurística
  - Pior caso é exponencial no tamanho da solução

Algoritmo A*

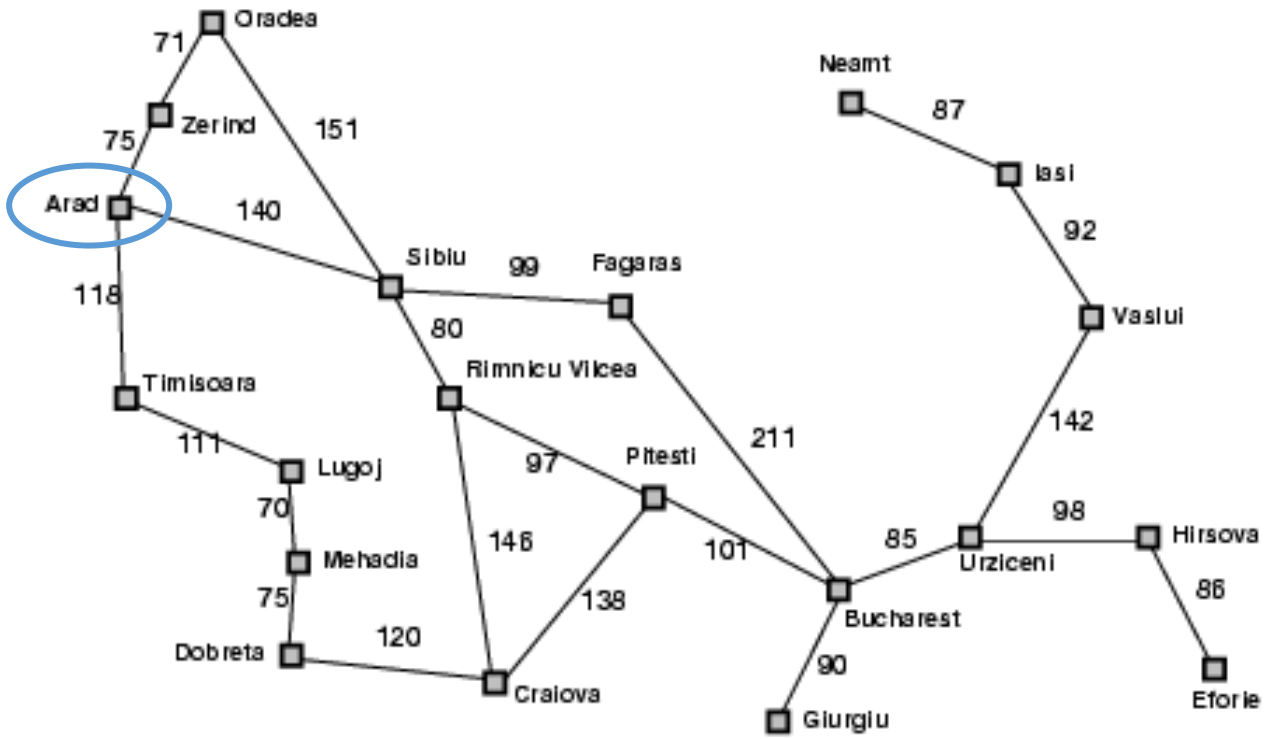


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*

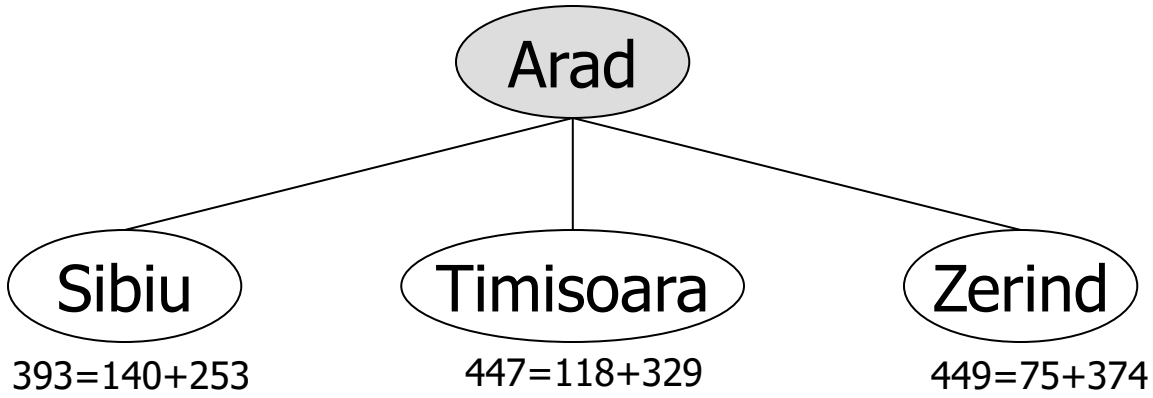
Algoritmo A*



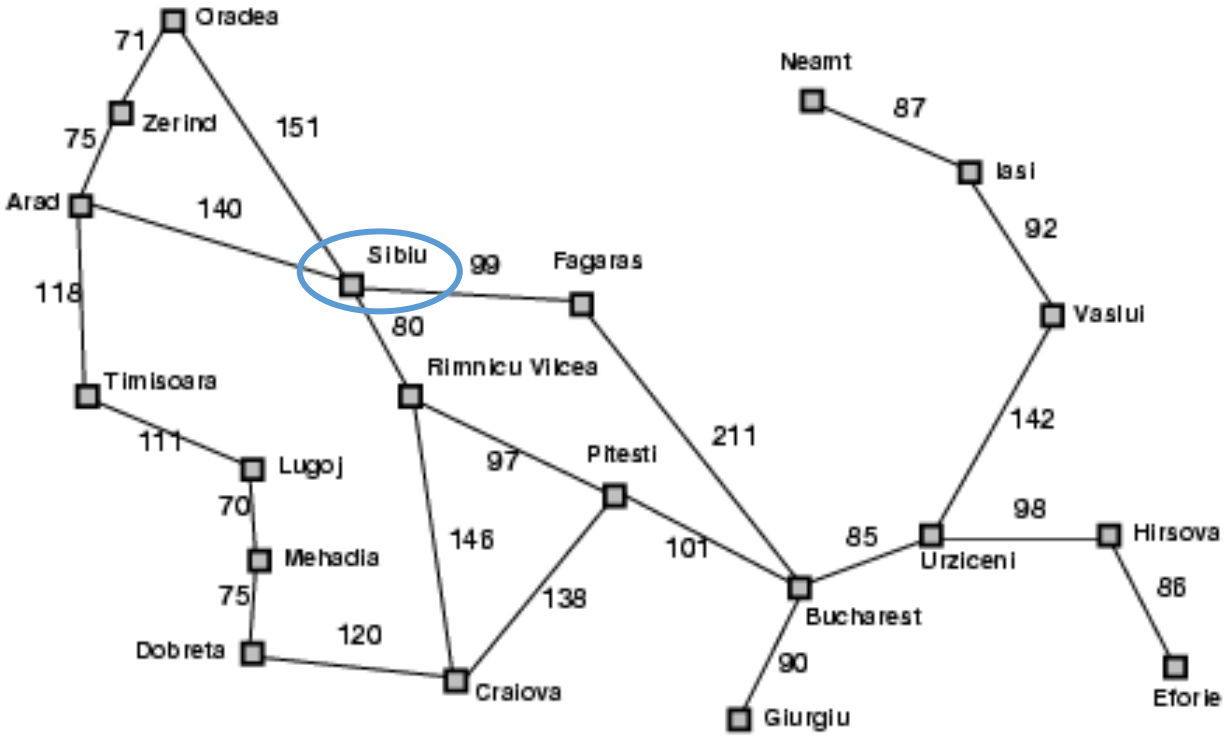
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*


Arad
Sibiu
Timisoara
Zerind
393=140+253
447=118+329
449=75+374

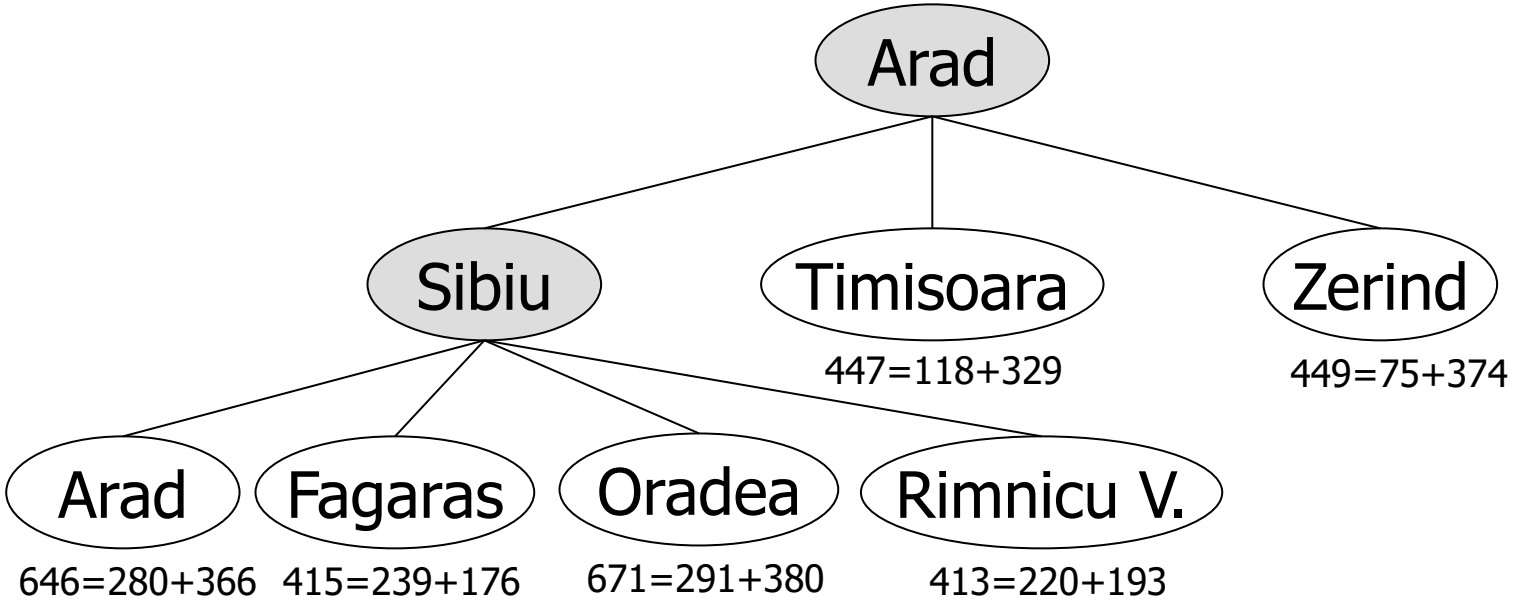
Algoritmo A*



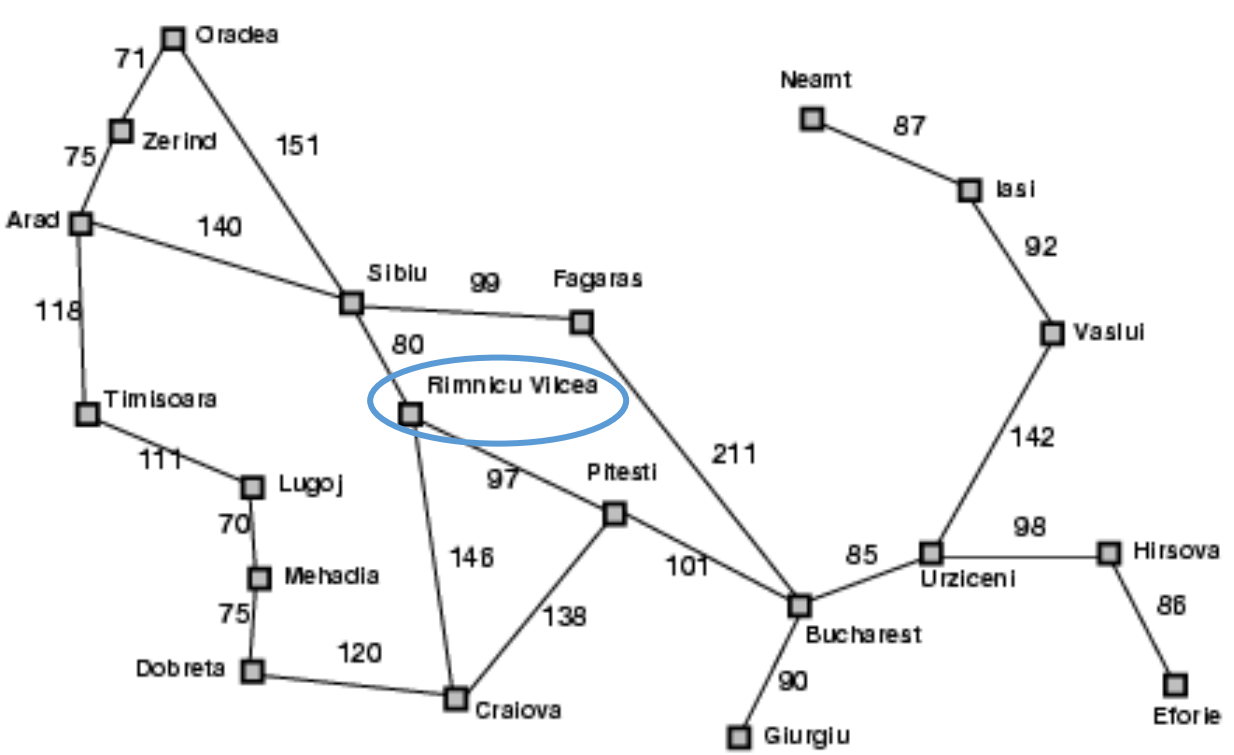
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*


Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu V.
646=280+366
415=239+176
671=291+380
413=220+193

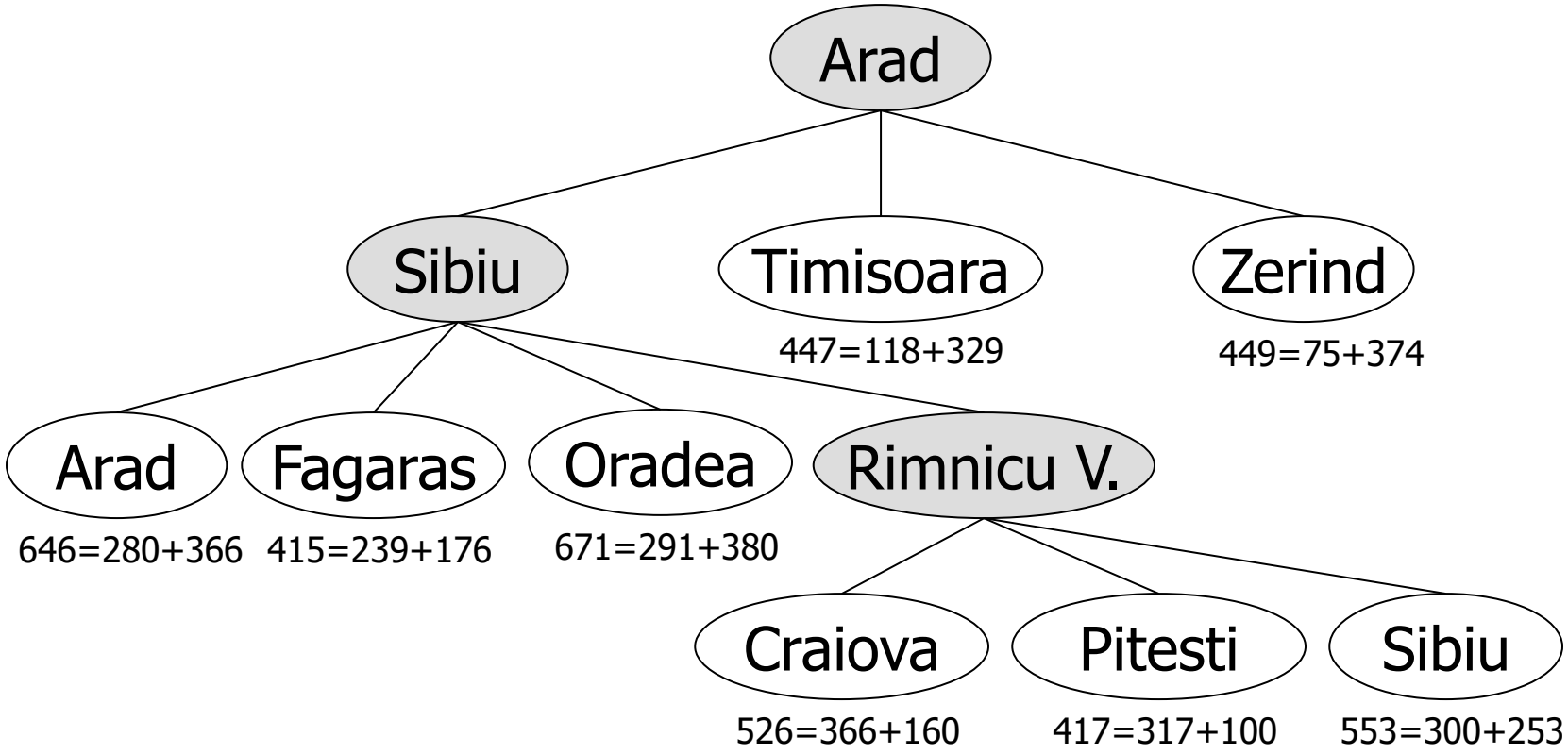
Algoritmo A*



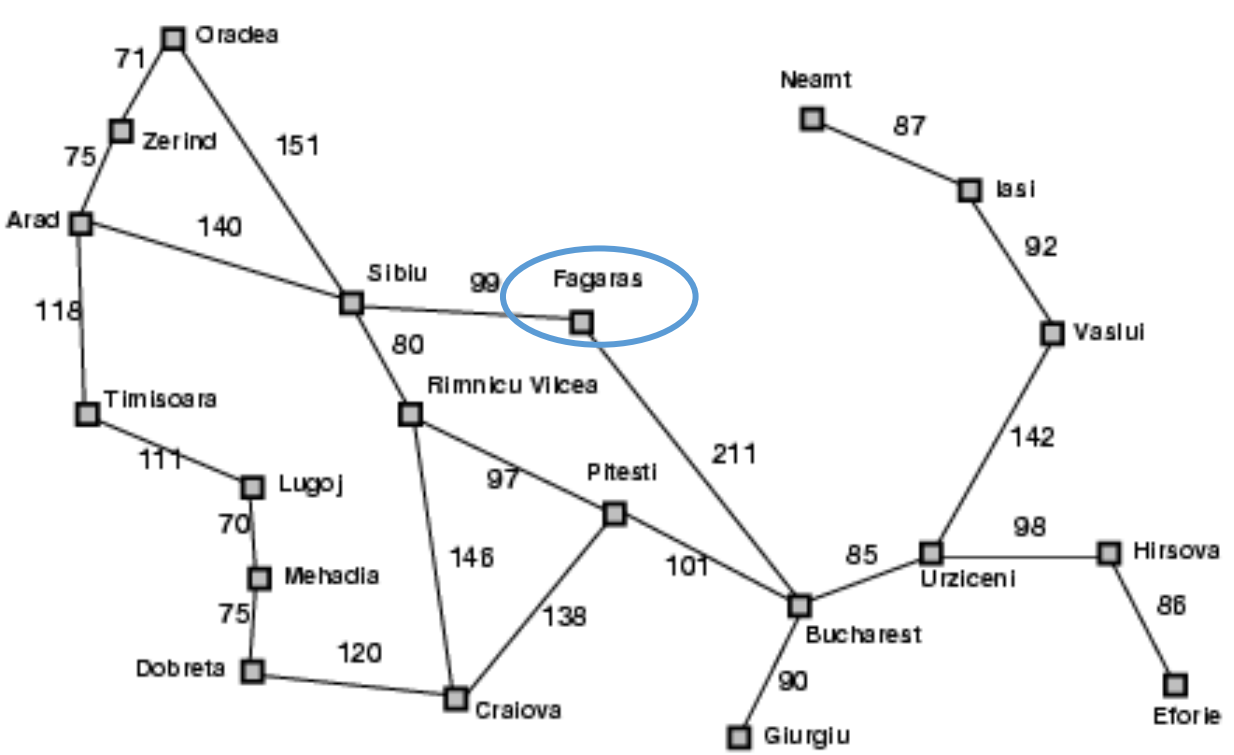
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu V.
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

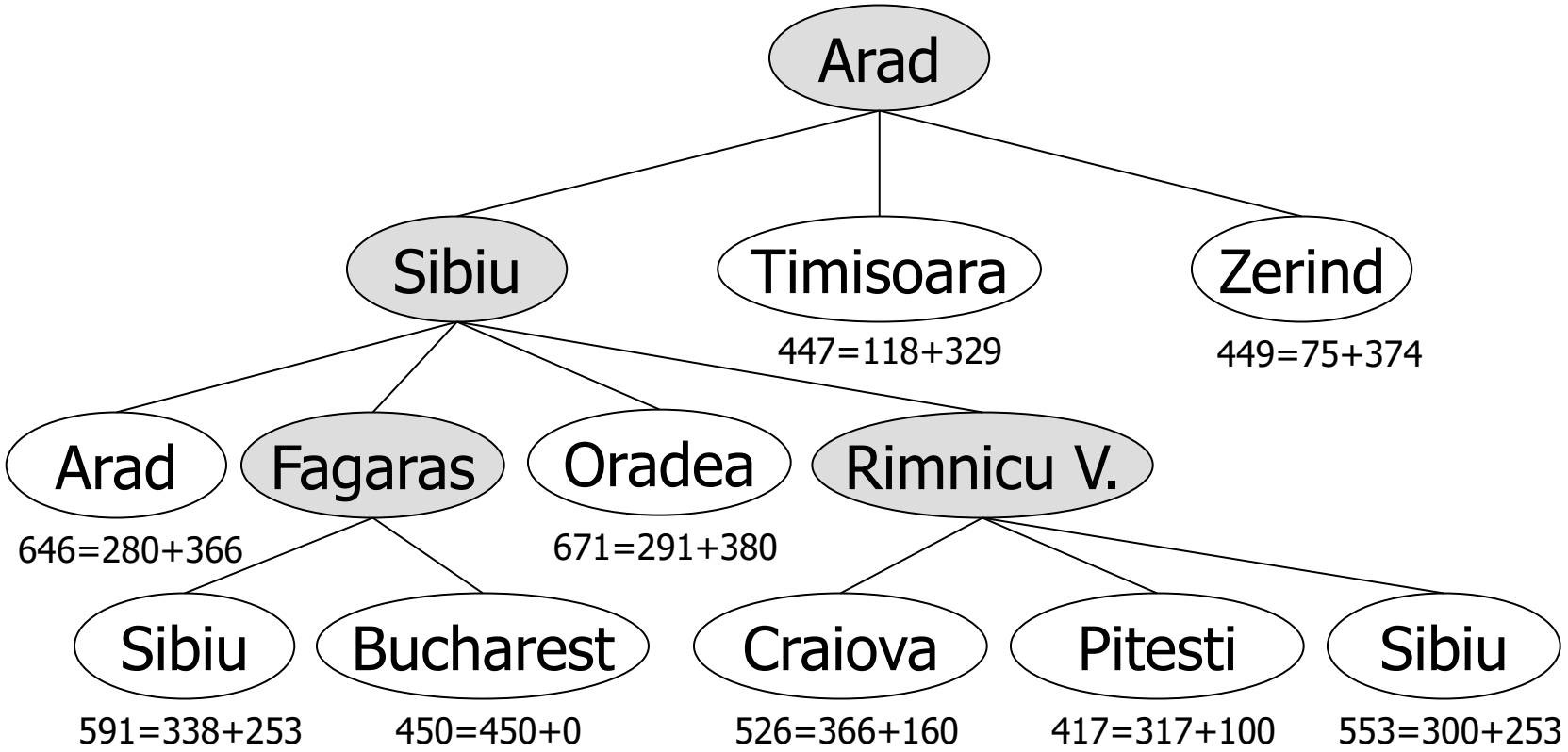
Algoritmo A*



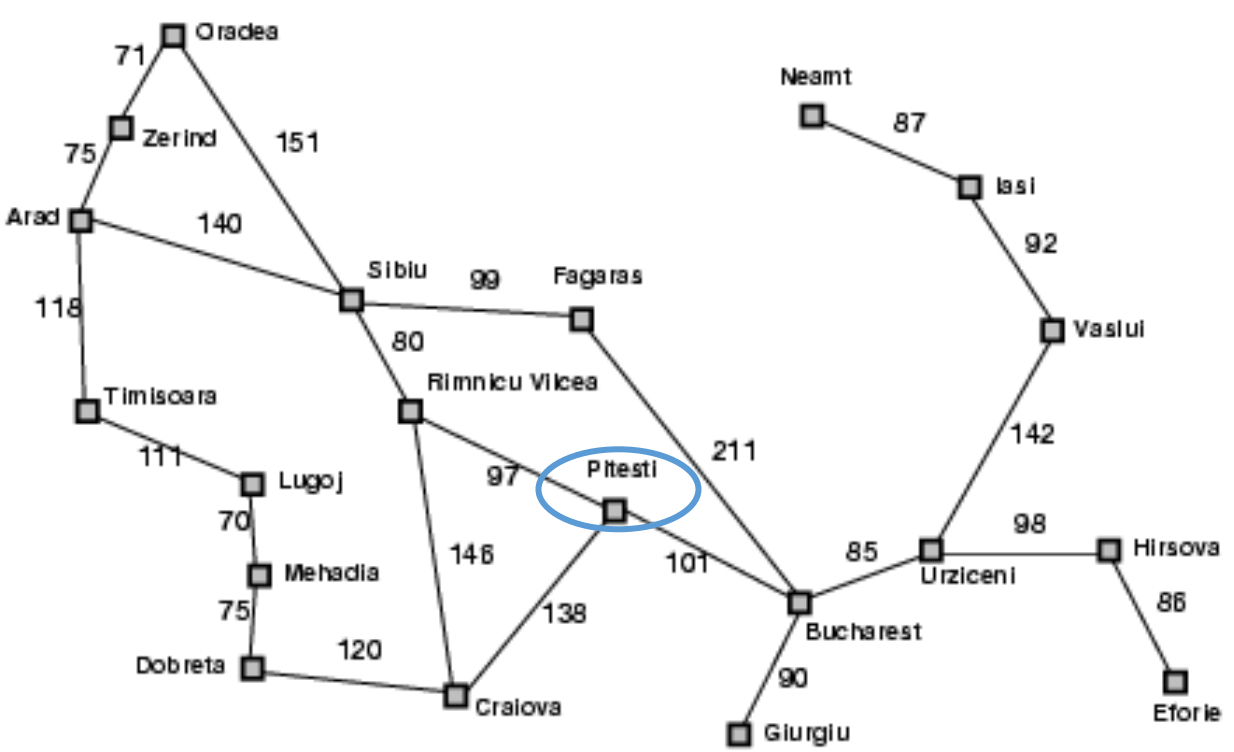
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*


Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu V.
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=300+253

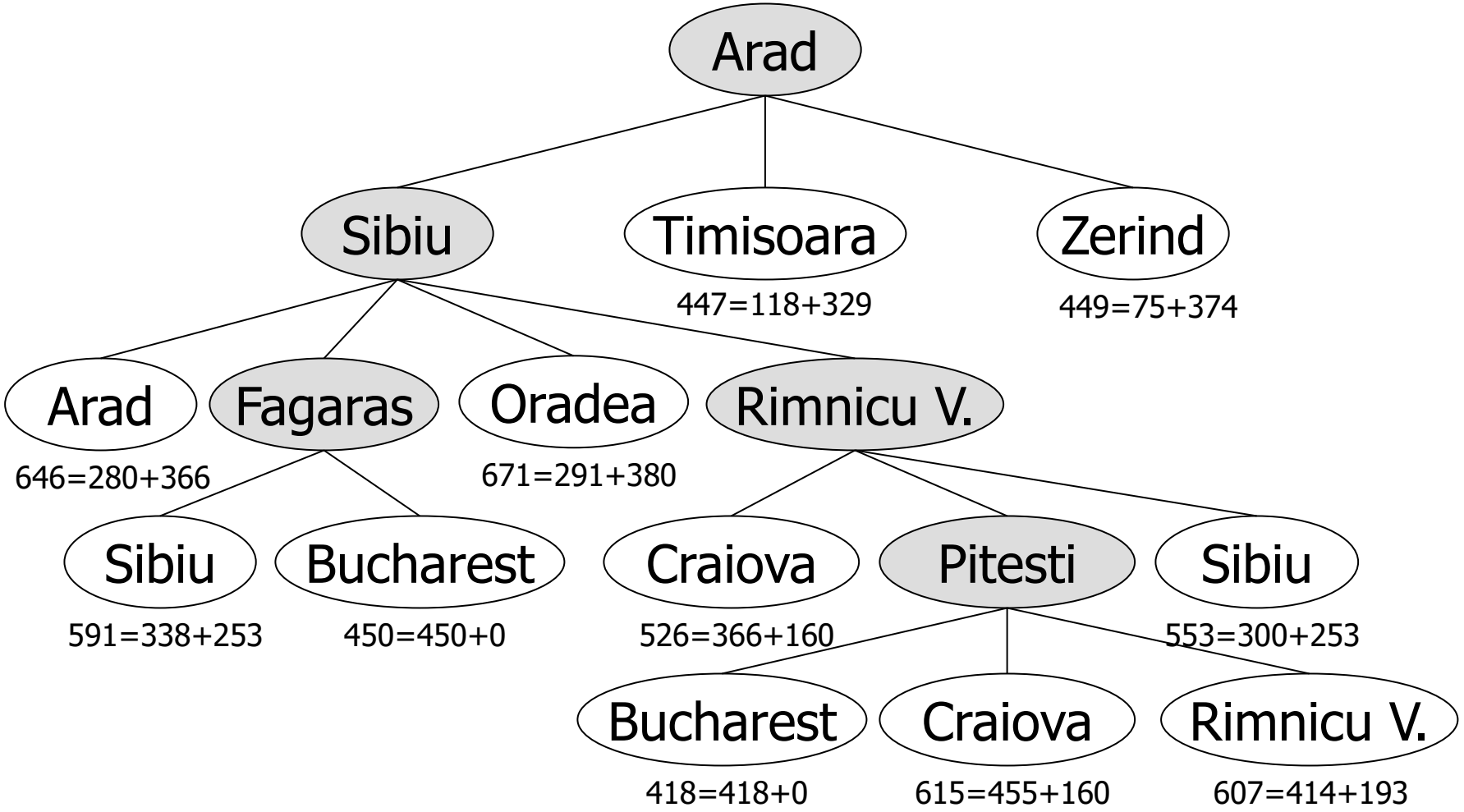
Algoritmo A*



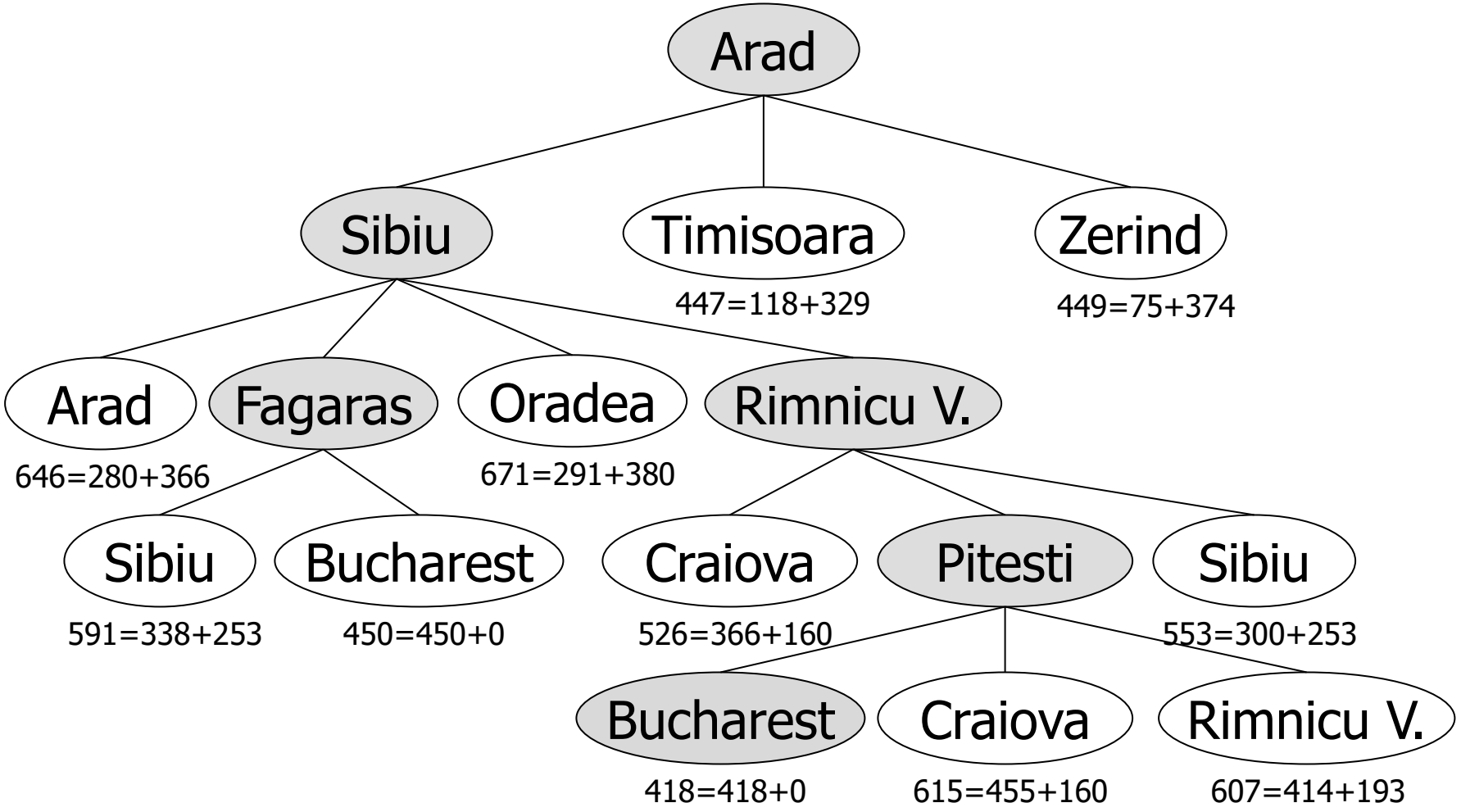
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Algoritmo A*


Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu V.
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu V.
418=418+0
615=455+160
607=414+193

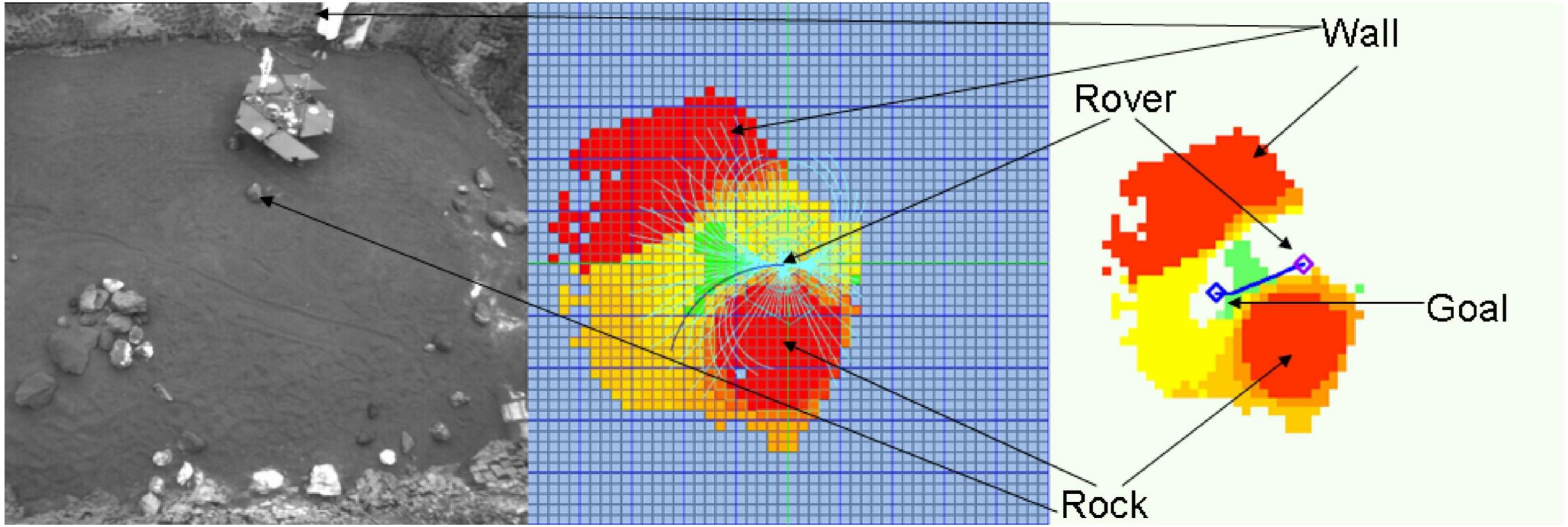
Algoritmo A*


Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu V.
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu V.
418=418+0
615=455+160
607=414+193

Algoritmo D*

- Como considerar mudanças no ambiente?
  - Dynamic A*
- Caminho inicial calculado a partir do A*
- Continuamente recalcula o caminho
  - Atualiza o mapa com novas informações
  - Altera o caminho considerando o novo mapa
- Planejamento Local e Global

Algoritmo D*



Fonte: http://www.cs.ait.ac.th/~mdailey/cvreadings/Carsten-MarsPath.pdf