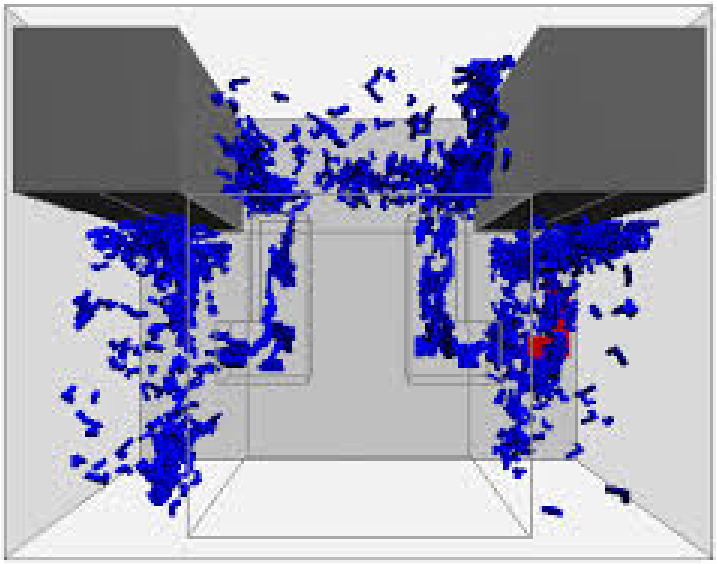
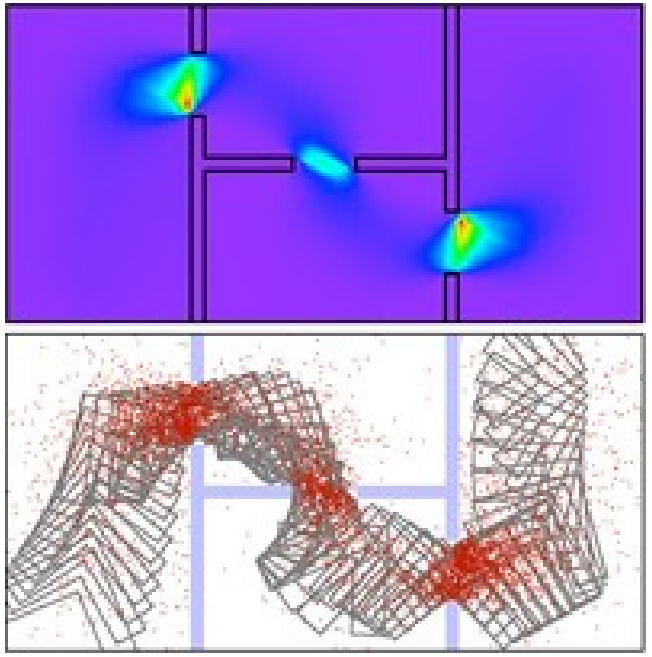
Planejamento baseado em Amostragem





Planejamento baseado em Amostragem

<u>Introdução</u>:

- Métodos determinísticos de planejamento:
  $\Rightarrow$ adequados para espaços de baixa dimensão.
  $\Rightarrow$ obstáculos são descritos por modelos contínuos.

- Planejamento em espaços de dimensão elevada:
  $\Rightarrow$ obstáculos descritos por grade de ocupação
  $\Rightarrow$ métodos probabilísticos mais adequados.

Planejamento baseado em Amostragem

Princípio:

- Amostrar **C** aleatoriamente.
- Testar se as configurações amostradas estão em $\mathbf{C_L}$.
- Tentar ligar pares configurações amostradas em $\mathbf{C_L}$ por caminhos livres.
- Construir uma rede de caminhos livres entre as configurações amostradas em $\mathbf{C_L}$, incluindo q_{ini} e q_{fin}.
- Buscar um caminho entre q_{ini} e q_{fin} na rede de caminhos construída.

Planejamento baseado em Amostragem

- Mapa de Rotas Probabilístico (PRM – *Probabilistic Road Map*): **Grafo**.

- Árvore Aleatória para Exploração Rápida (RRT – *Rapid-exploring Random Tree*): **Árvore**.

Mapa de Rotas Probabilístico

- Mapa de Rotas Probabilístico (PRM – *Probabilistic Road Map*): versão probabilística de Mapas de Rota.

- Executado em duas etapas:

  - Construção do Mapa de Rotas.

  - Busca de Caminho.

Construção do Mapa de Rotas

- Pressuposto: espaço de trabalho é estático.

- Etapa computacionalmente custosa, mas pode ser feita off-line.

- Nesta etapa é construído um grafo R(N,E) $\subset$ $\mathbf{C_L}$, (N é o conjunto de nós e E o conjunto de arestas).

- R(N,E) busca capturar a conectividade de $\mathbf{C_L}$

Construção do Mapa de Rotas

- Configurações aleatórias são geradas em $\mathbf{C_L}$.

- Cada configuração q_{nova} gerada é armazenada em um nó.

- Através de planejamento local, verifica-se quais configurações vizinhas q_k às quais q_{nova} pode se conectar.

- Caso não haja colisão entre q_{nova} e q_j:

➢ Se ambas não pertencerem ao mesmo componente conexo de R criar aresta nova em E entre estes dois nós.

➢ Caso contrário, não criar aresta nova, para evitar gerar laços no grafo.

Busca de Caminho no Mapa de Rotas

- Uma vez construído o mapa de rotas, o mesmo pode ser usado para buscar um caminho livre entre q_{ini} e q_{fin}.

- Esta etapa pode ser feita rapidamente, usando o grafo construído na etapa de construção.

Busca de Caminho no Mapa de Rotas

- Liga-se q_{ini} à configuração mais próxima em R, q_{ini}^{R}.

- Liga-se q_{fin} à configuração mais próxima em R, q_{fin}^{R}.

- Inicialmente, verifica-se se q_{ini}^{R} e q_{fin}^{R} estão no mesmo componente conexo de R. caso não, reportar falha.

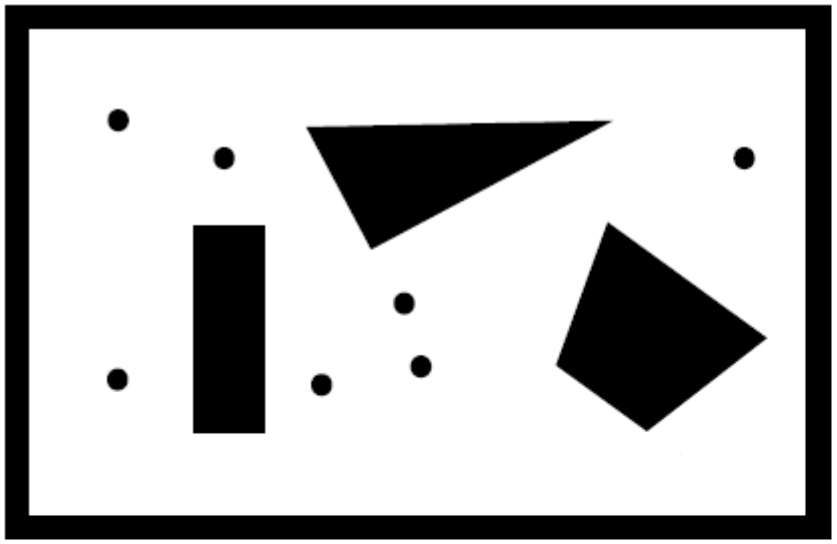
- Caso sim, buscar caminho entre elas em R e retornar o caminho composto pelos nós q_{ini}, q_{ini}^{R}, sequência de nós ligando q_{ini}^{R} a q_{fin}^{R}, q_{fin}^{R} e q_{fin}.

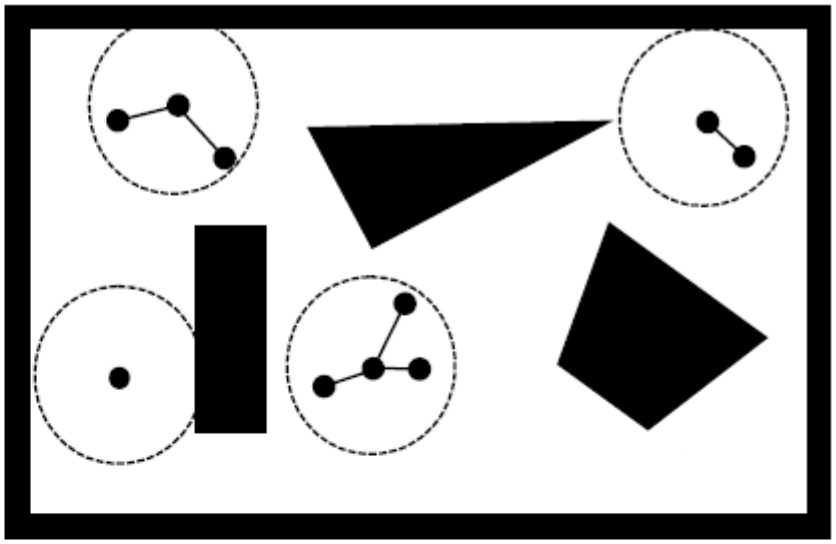
Algoritmo para a Etapa de Construção do PRM

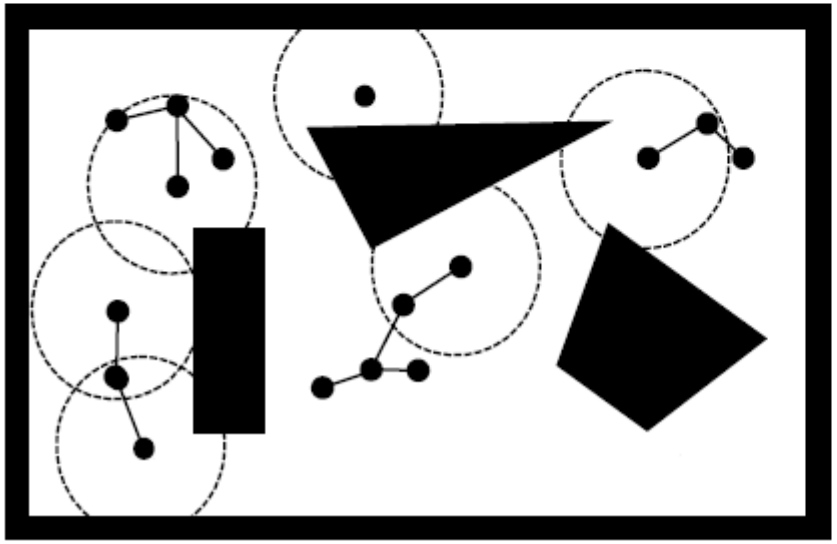
R(N,E)

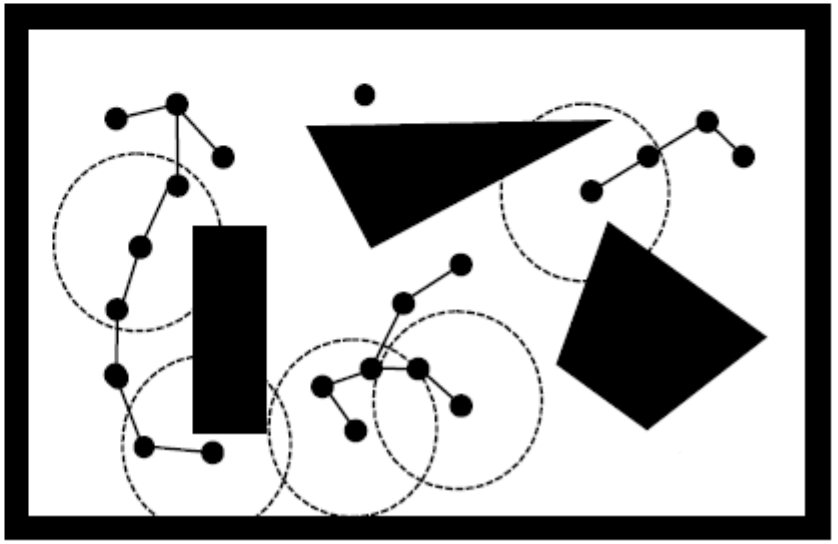
Inicializar:
N_{max} = Número máximo de configurações do grafo.
N_{vmax} = Número máximo de configurações vizinhas.
R_V = Raio de vizinhança em torno de configuração nova q_{nova}.

1 $i \leftarrow 1$
2 $N \leftarrow \varnothing$
3 $E \leftarrow \varnothing$
4 **enquanto** $i < N_{max}$ **faça**
5 $q_{nova} \leftarrow$ configuração aleatória em **C**
6 **se** $q_{nova} \in \mathbf{C_L}$ **então**
7 $i \leftarrow i+1$
8 inserir q_{nova} em N
9 selecionar até N_{vmax} configurações vizinhas q_k ($k = 1, ..., N_{vmax}$), tal que $\|q_k - q_{nova}\| < R_V$
10 **para** $j = 1$ **até** N_{vmax} **faça**
11 **se** ($\exists$ caminho livre entre q_k e q_{nova}) **e** (q_k e $q_{nova} \notin$ mesmo componente conexo de R) **então**
12 inserir em E nova aresta conectando q_k e q_{nova}
13 **fim se**
14 **fim para**
15 **fim se**
16 **fim enquanto**









Observações

Observações:

- Após construído o Mapa de Rotas, pode ser utilizado eficientemente para busca de caminhos (etapa de buscas).

  $\Rightarrow$ Adequado para ambientes estáticos.

- O método pode ser utilizado em espaços de dimensões elevadas, pois, após a etapa de construção, a informação de conectividade é capturada implicitamente em R.

Observações

O número máximo de nós N_{max} depende da topologia de $\mathbf{C_L}$.

N_{max} pequeno pode no garantir conectividade insuficiente, **R** pode não conter a solução, mesmo existindo.

A probabilidade de sucesso (caso exista solução) cresce exponencialmente com o número de amostras.

Método probabilisticamente completo. Caso exista solução, a Pr(sucesso) $\rightarrow 1$ quando N $\rightarrow \infty$.

N_{max} grande pode exigir muitos recursos computacionais.

Observações

- Vários métodos podem ser utilizados para gerar novas amostras (q_{nova} na linha 5 do algoritmo), influenciando bastante no desempenho.

- A seleção das N_{vmax} configurações vizinhas a q_{nova} situadas a uma distância menor que R_V, (na linha 9 do algoritmo), pode não ser simples.

- Em espaços de dimensão elevada, esta seleção pode consumir bastantes recursos computacionais, exigindo a escolha de estruturas de armazenamento e mecanismos de busca eficientes.

Observações

- A busca de um caminho livre entre q_k e q_{nova} (linha 11) pode ser feita de várias maneiras por planejamento local.

- Se apenas caminhos constituídos por segmentos de retas são admitidos, um simples teste de visibilidade resolve o problema, caso contrário, outros métodos de planejamento podem ser utilizados, melhorando o desempenho.

- O algoritmo a seguir executa um planejamento local simples entre duas configurações q_j e q_k tentando conectá-las através de um segmento de reta, parametrizado por $\lambda \in [0,1]$, traçado incrementando λ com resolução $\Delta\lambda$, de q_j a q_k.

Algoritmo de Planejamento Local entre q_j e q_k

PLAN(q_j,q_k)

Inicializar:
$q_j, q_k \in \mathbf{C_L}$
$\Delta\lambda \in (0,1)$, tal que $1/\Delta\lambda \in \mathbb{Z}$
1 $\lambda \leftarrow 0$
2 $q \leftarrow q_j$
3 **enquanto** $\lambda < 1$ **faça**
4 **se** $q \notin \mathbf{C_L}$ **então**
5 **retornar** Status conexão(q_j, q_k) = Falso
6 **Se não**
7 $\lambda \leftarrow \lambda+\Delta\lambda$
8 $q \leftarrow (1-\lambda).q_j + \lambda.q_k$
9 **fim se**
10 **fim enquanto**
11 **retornar** Status conexão(q_j, q_k) = Verdadeiro

Árvore Aleatória para Exploração Rápida

- RRT – *Rapid-exploring Random Tree:* método inicialmente apresentado como um algoritmo de planejamento para buscas rápidas em espaços de alta dimensionalidade.

Árvore Aleatória para Exploração Rápida

Princípio:

- polarizar a busca em regiões pouco exploradas de $\mathbf{C_L}$ amostrando pontos no mesmo e polarizando a busca na direção dessas regiões de forma incremental.

Mecanismo de Busca

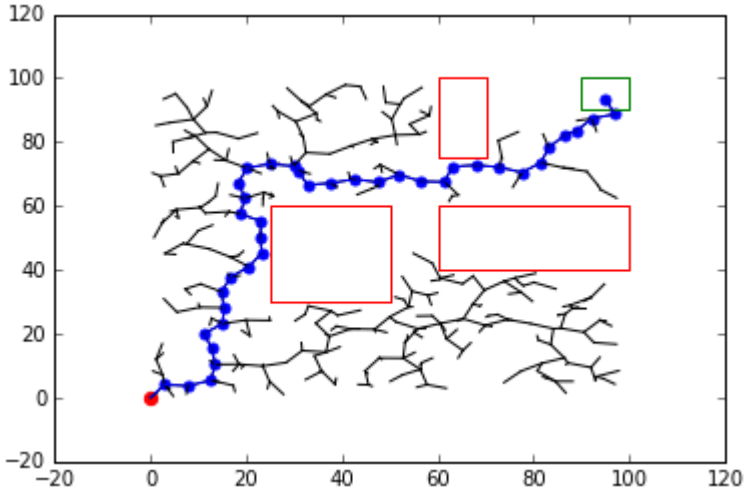
- A partir de q_{ini}, uma árvore T é construída amostrando configurações aleatórias em e tentando conectá-las à arvore corrente, expandindo T.

- Isto leva a uma cobertura uniforme de $\mathbf{C_L}$, permitindo alcançar q_{fin} ou uma vizinhança Q_{fin} de q_{fin}.

- O algoritmo executa a construção incremental da Árvore T.

- A cada passo, o algoritmo tenta expandir T adicionando um novo nó que é polarizado por uma configuração escolhida aleatoriamente q_{al}.

Árvore Aleatória para Exploração Rápida



Observações

- Uma vantagem de RRT sobre PRM é que, enquanto em PRM, o número de nós é determinado *a priori*, no RRT os nós são gerados de forma incremental.

- Versões mais eficientes: busca bidirecional, construindo duas árvores, a partir de q_{ini} e q_{fin}.
  - A busca acaba quando as árvores se encontram.
  - Não precisa cobrir a maior parte de $\mathbf{C}_L$.

Algoritmo para construção da RRT

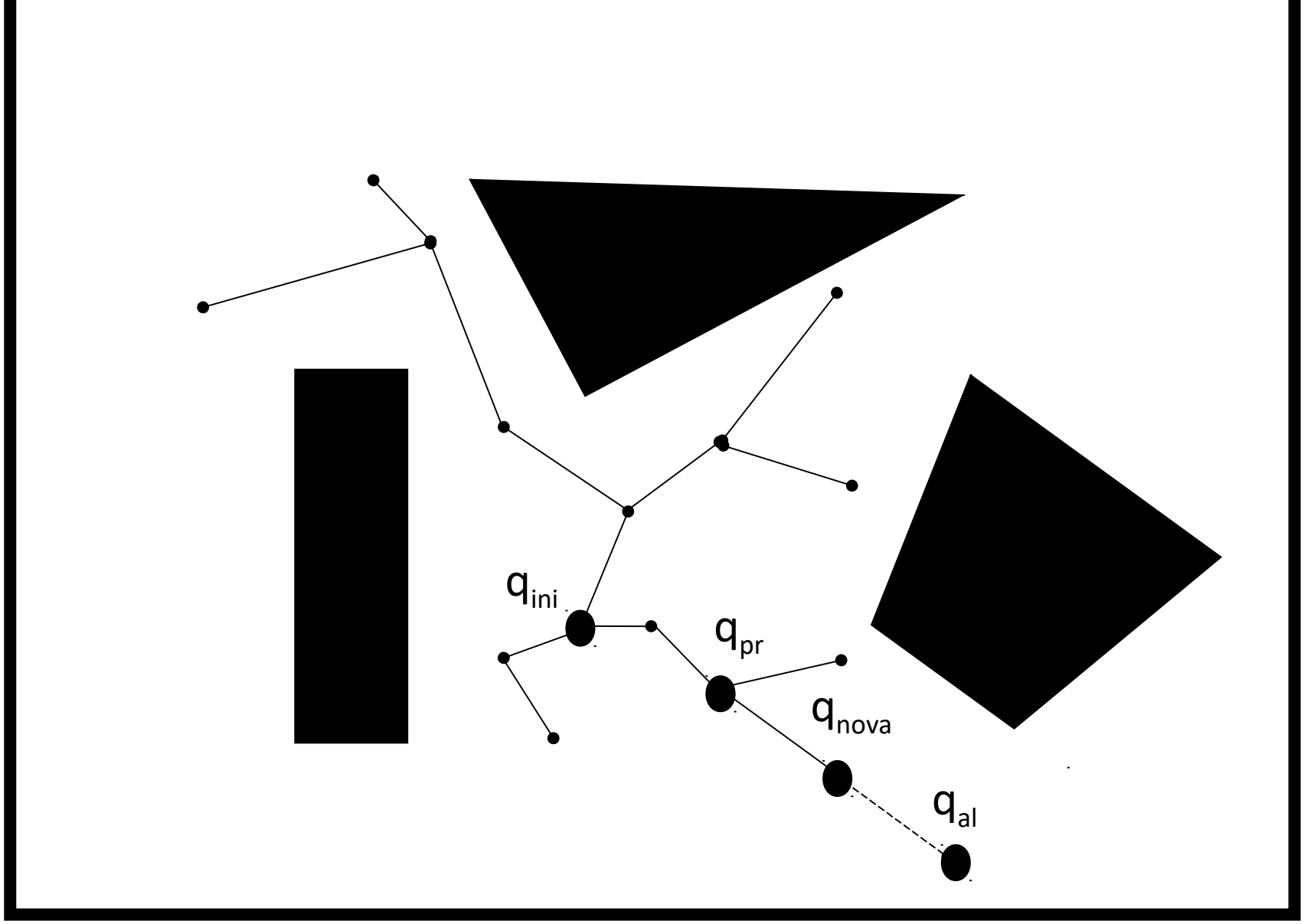
CONSTRUIR_RRT(q_{ini})

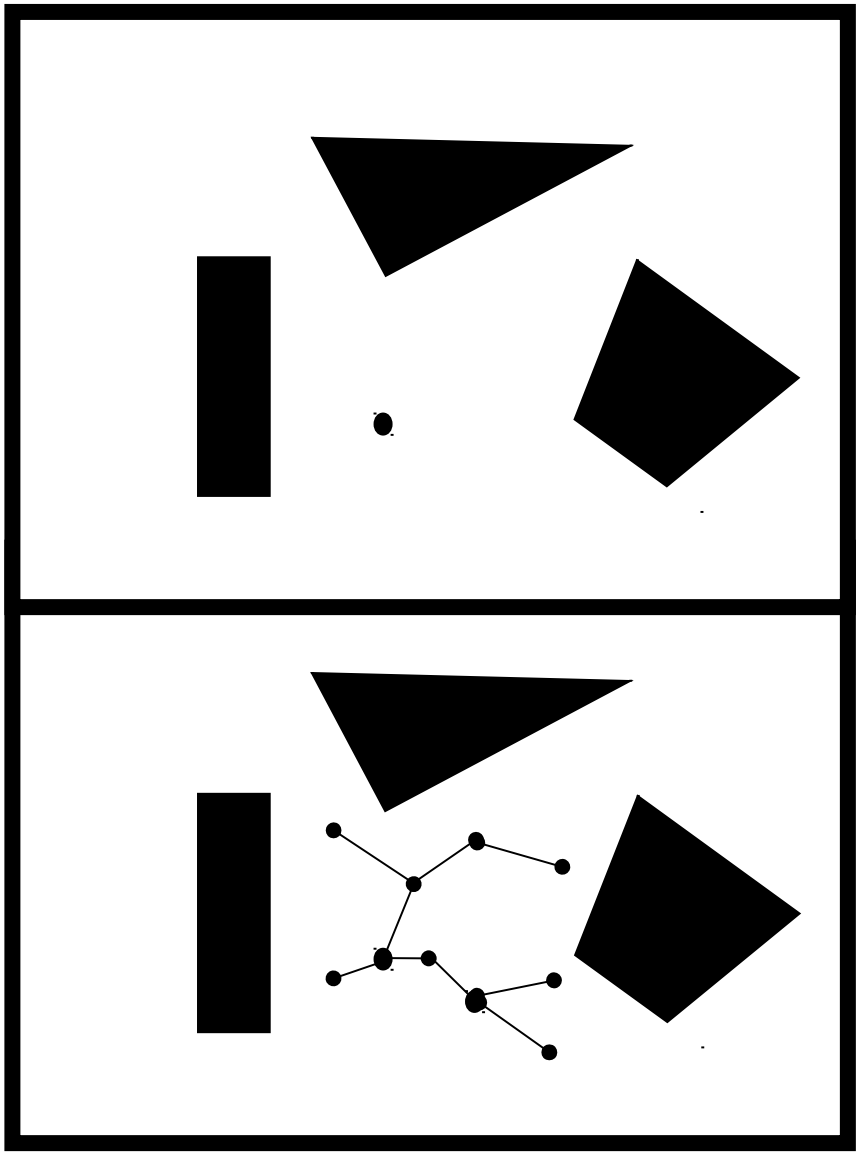
N_{max} = Número máximo de nós na árvore.
T.ini(q_{ini})
para i = 1 **até** N_{max} **faça**
    q_{al} ← configuração aleatória em **C**
    EXPANDIR_RRT(T,q_{al})
Retornar T

Algoritmo para Expansão da RRT

EXPANDIR_RRT(T,q_{al})

q_{pr} ← VIZINHO_MAIS_PRÓXIMO((T,q_{al})
q_{nova} ← PLAN(q_{pr},q_{al})
T.adiciona_nó(q_{nova})
T.adiciona_aresta(q_{pr},q_{nova})
se (q_{nova} = q_{al}) **então**
    retornar Status conexão(q_{pr}, q_{al}) = Alcançada
senão
    retornar Status conexão(q_{pr}, q_{al}) = Avanço
fim se
retornar Status conexão(q_{nova}, q_k) = Bloqueada


q_{ini}
q_{pr}
q_{nova}
q_{al}



Observações

- Para polarizar a busca em regiões pouco exploradas de $\mathbf{C}_L$, a probabilidade de amostrar q_{al}, (linha 3 de CONSTRUIR_RRT (q_{ini})), pode ser inversamente proporcional à densidade de amostras em $\mathbf{C}_L$ medida localmente.

- q_{al} é selecionada com maior probabilidade em áreas menos densas.

- O vizinho mais próximo a q_{al} (na linha 1 de EXPANDIR_RRT(T,q_{al})), é escolhido de acordo com a métrica adotada para o espaço de configuração **C**.

Observações

No planejamento local PLAN(q_{pr},q_{al}), (linha 2 de EXPANDIR_RRT(T,q_{al})), que tenta conectar q_{pr} a q_{al}, temos três situações:

- $q_{nova} = q_{al}$. A configuração selecionada foi alcançada e a mesma é adicionada à árvore. (Linha 6).
- $q_{nova} \neq q_{al}$. O planejador avançou até uma q_{nova}, após a qual há um obstáculo. Neste caso, esta última q_{nova} é adicionada à árvore. (Linha 8).
- O planejador não consegue avançar em direção a q_{al} a partir de q_{pr} (conexão Bloqueada). Neste caso, nenhum novo nó é adicionado a T. (Linha 10).

Planejamento por RRT

- RRT pode ser usado para planejamento de caminhos entre q_{ini} e q_{fin}, uma vez que a medida que a árvore cresce, uma cobertura uniforme de $\mathbf{C}_L$ é alcançada.

- Se q_{fin} é alcançável, a probabilidade de alcançar uma vizinhança de q_{fin} tende a 1 quando o tempo tende a infinito. Assim, q_{fin} deverá ser alcançada, desde que se tenha suficiente tempo de convergência.

- Converge devagar.

Planejamento RRT – Goal Bias

- Variante do RRT que converge mais rapidamente para o alvo.
- Na amostragem de q_{al}, (linha 3), em lugar de escolher uma configuração totalmente aleatória, faz-se esta escolha com uma probabilidade alta ou escolhe-se a configuração final q_{fin} com probabilidade baixa (por exemplo: 0,05).
- Acelera bastante a convergência ao alvo.

Planejador RRT Bidirecional

- Melhor desempenho pode ser obtido por busca bidirecional.
- Duas árvores são construídas concorrentemente: T_{ini} partindo de q_{ini} e T_{fin} partindo de q_{fin}.
- A solução é obtida quando as duas árvores se encontram.
- A construção de T_{ini} e T_{fin} deve ser polarizada de modo a assegurar que as duas arvores se encontrem bem antes de que uma cobertura uniforme de todo $\mathbf{C}_L$ seja alcançada.

Algoritmo RRT Bidirecional

RRT_Bidirecional(q_{ini},q_{fin})

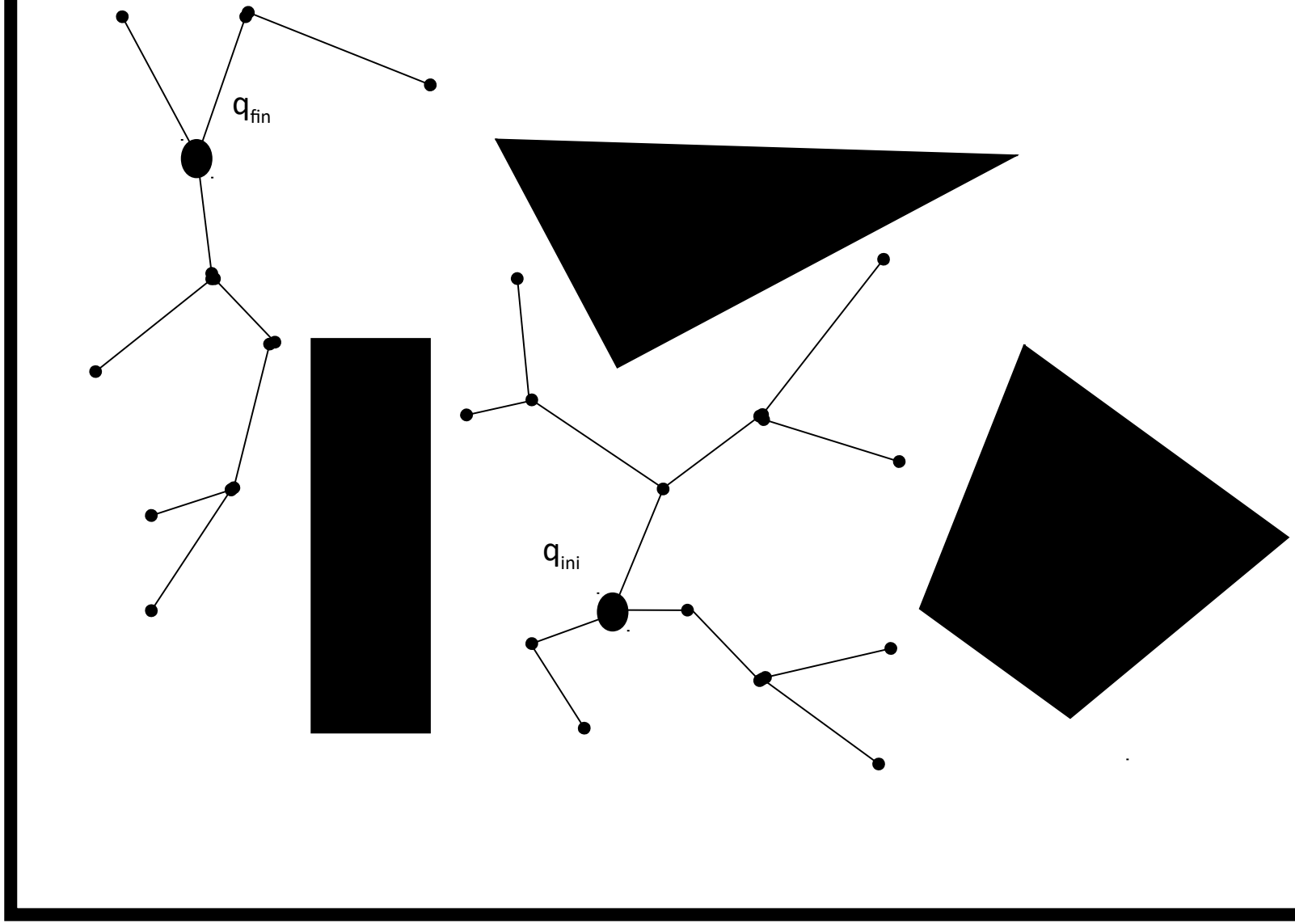
```
1   T.ini(q_ini), T.fin(q_fin)
2   para i = 1 até N_max faça
3           q_al ← configuração aleatória em C
4           se (não(EXPANDIR_RRT(T_ini,q_al) = Bloqueado)) então
5                   se (EXPANDIR_RRT(T_fin,q_nova) = Alcançada)) então
6                           retornar Caminho(T_ini,T_fin)
7                   fim se
8           fim se
9           Alternar(T_ini,T_fin)
10  fim para
11  retornar Falha
```

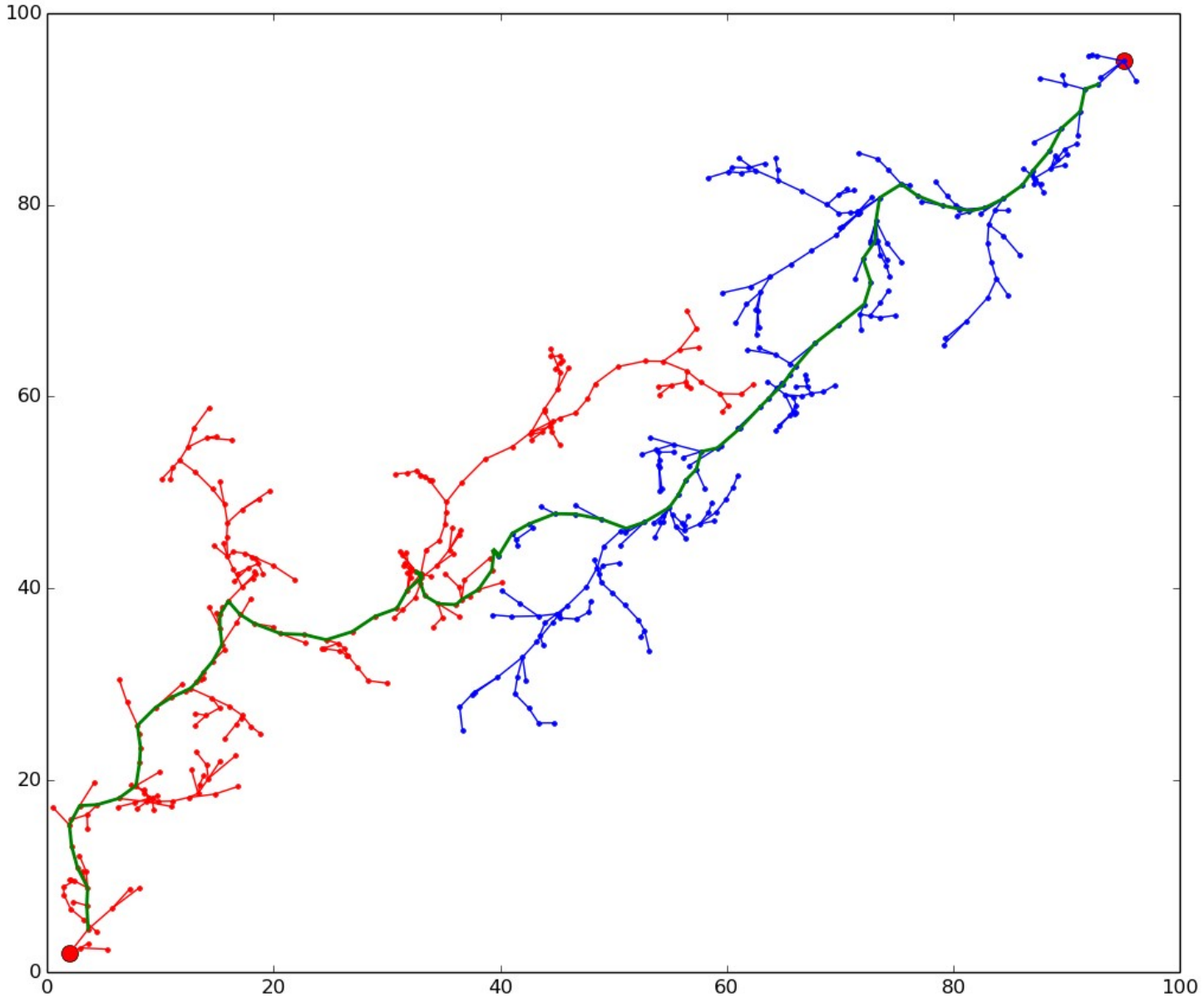
Planejamento baseado em Amostragem

Observações:

- A cada iteração, a árvore corrente é expandida e tenta-se conectá-la ao nó mais próximo da outra árvore.

- A seguir, invertem-se os papeis, alternado as duas árvores.


q_{fin}
q_{ini}


100
80
60
40
20
0
0
20
40
60
80
100

Planejamento baseado em Amostragem

Recursos:

- OMPL: https://ompl.kavrakilab.org/
- OMPL Web: http://omplapp.kavrakilab.org/
- Planner arena: http://plannerarena.org/

Planejamento baseado em Amostragem