Fundamentos de Aprendizagem de Máquina

Hélio Pio

Programação das Aulas

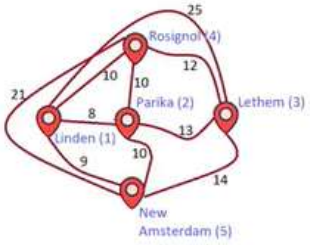
Tópico 1: Introdução a Inteligência Artificial
Tópico 2: Agentes Inteligentes
Tópico 3: Fundamentos de Aprendizagem de Máquina
Tópico 4: Redes Neurais Artificiais
Tópico 5: Atividade em Aula – Primeira Avaliação
Tópico 6: Representação da Incerteza e Lógica Fuzzy
Tópico 7: Redes Bayesianas
Tópico 8: Support Vector Machines
Tópico 9: Atividade em Aula – Segunda Avaliação
Tópico 10: Resolução de Problemas por Meio de Busca e Otimização
Tópico 11: Técnicas de Ensemble
Tópico 12: Atividade em Aula – Terceira Avaliação

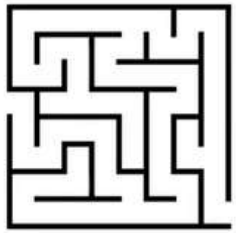
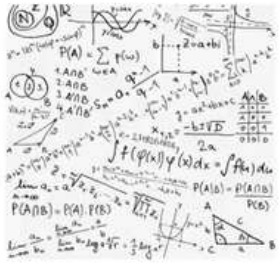
O que é algoritmo de busca e otimização?

Resolução de Problemas - Busca e Otimização

O que é algoritmo de busca e otimização?

- Algoritmos de busca e otimização são técnicas computacionais usadas para encontrar soluções ótimas ou satisfatórias para problemas complexos, considerando um conjunto de possibilidades.





Resolução de Problemas - Busca e Otimização

Por que não testar todas as possíveis soluções?

- Jogo Go
  - Tabuleiro 19x19
  - 2.08168199382.10^170

- Mais que o número de átomos conhecidos no Universo.

Resolução de Problemas - Busca e Otimização

Introdução

- Na IA Clássica, existem os chamados problemas de busca.
- A forma de representação do conhecimento define o espaço de busca e suas propriedades topológicas.
- Em espaços discretos, percorre-se um grafo em busca de um nó.
- Em espaços contínuos, percorre-se o espaço em busca de um ponto.
- Em ambos os casos, a noção de vizinhança do nó ou ponto atual é de grande relevância.

Resolução de Problemas - Busca e Otimização

Técnicas de Busca e Otimização

- Buscas não informadas
  - Exploram o espaço de busca sem informações adicionais.
  - Exemplos: Busca em Largura (BFS) e Busca em Profundidade (DFS).
  - Simples, mas podem ser menos eficientes para problemas complexos.

- Buscas informadas
  - Utilizam informações heurísticas para orientar a busca.
  - Exemplos: Algoritmo A* (A estrela).
  - Mais eficientes e podem encontrar soluções melhores, mas dependem da qualidade das heurísticas.

Resolução de Problemas - Busca e Otimização

Técnicas de Busca e Otimização

- Buscas populacionais
  - Trabalham com uma população de soluções simultaneamente.
  - Usam estratégias de seleção, recombinação e mutação inspiradas na evolução biológica.
  - Exemplo: Algoritmos Genéticos.

- Buscas não populacionais
  - Exploram um único conjunto de soluções iterativamente.
  - Exemplos: Busca Local e Descida de Gradiente.
  - Menos exploratórios, mas podem ser eficazes para problemas bem definidos.

Resolução de Problemas - Busca e Otimização

Exemplo de problema de busca

- Roteamento de Veículos: Empresas de logística e transporte usam técnicas de busca e otimização para determinar a melhor rota para seus veículos entregarem mercadorias, minimizando o tempo de viagem e os custos envolvidos.

- Otimização de Produção: Indústrias usam técnicas de otimização para planejar a programação da produção, minimizando o tempo de espera das máquinas, reduzindo gargalos e maximizando a eficiência geral.

- Otimização de Investimentos Financeiros: Bancos e instituições financeiras aplicam técnicas de otimização para determinar a alocação de ativos em carteiras de investimento, visando maximizar o retorno com base em riscos específicos.

Resolução de Problemas - Busca e Otimização

Exemplo de problema de busca

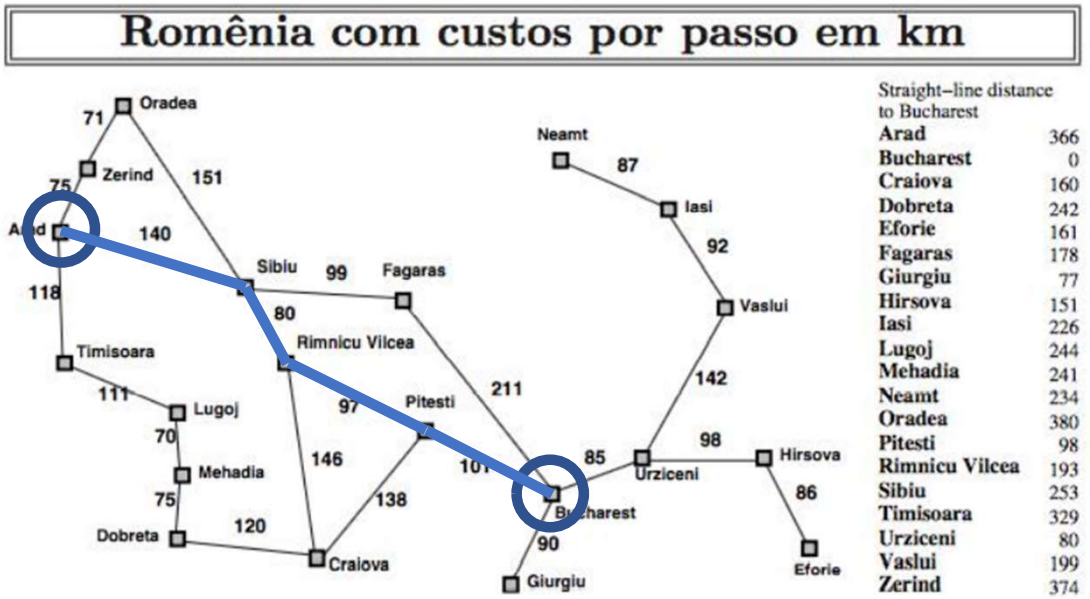
- Planejamento de Redes de Telecomunicações: Empresas de telecomunicações usam técnicas de busca para otimizar a cobertura de rede, encontrar a melhor localização para torres de celular e minimizar as interferências.

- Otimização de Cadeias de Suprimentos: Empresas usam técnicas de otimização para otimizar o fluxo de materiais e produtos ao longo da cadeia de suprimentos, reduzindo custos de armazenagem e transporte.

Resolução de Problemas - Busca e Otimização

Exemplo de problema de busca

- De férias na Romênia, atualmente em Arad.
- Voo sai amanhã de Bucareste.
- Formular objetivo: Estar em Bucareste a tempo.
- Formular problema:
  - Estados: cidades
  - Ações: dirigir entre as cidades
- Encontrar solução:
  - Sequência de cidades: Ex. Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucareste



Resolução de Problemas - Busca e Otimização

Exemplo de problema de busca

- Um problema é definido por quatro itens:
  1. Estado inicial ex., “em Arad"
  2. Ações ou função sucessor S(x) = conjunto de pares estado-ação
     – ex., S(Arad) = {<Arad -> Zerind>, … }
  3. Teste de objetivo, pode ser
     - Explícito, ex., x = “em Bucharest"
     - Implícito, ex., Cheque-mate(x)
  4. Custo de caminho (aditivo)
     - ex., soma das distâncias, número de ações executadas, etc.
     - c(x,a,y) é o custo do passo, que deve ser sempre ≥ 0
- Uma solução é uma sequência de ações que levam do estado inicial para o estado objetivo.
- Uma solução ótima é uma solução com o menor custo de caminho.

Resolução de Problemas - Busca e Otimização

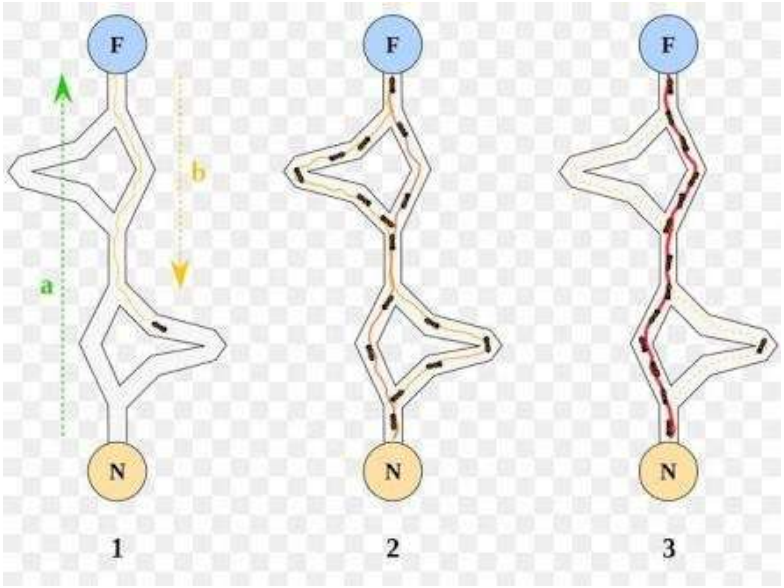
Exemplo de problema de busca

- O mundo real é absurdamente complexo.
- O espaço de estados é uma abstração.
- Estado (abstrato): conjunto de estados reais.
- Ação (abstrata): combinação complexa de ações reais
  - ex.: "Arad a Zerind" representa um conjunto complexo de rotas, desvios, paradas, etc.
- Solução (abstrata): conjunto de caminhos reais que são soluções no mundo real

Resolução de Problemas - Busca e Otimização

Exemplo de técnicas de IA para problemas de busca e otimização

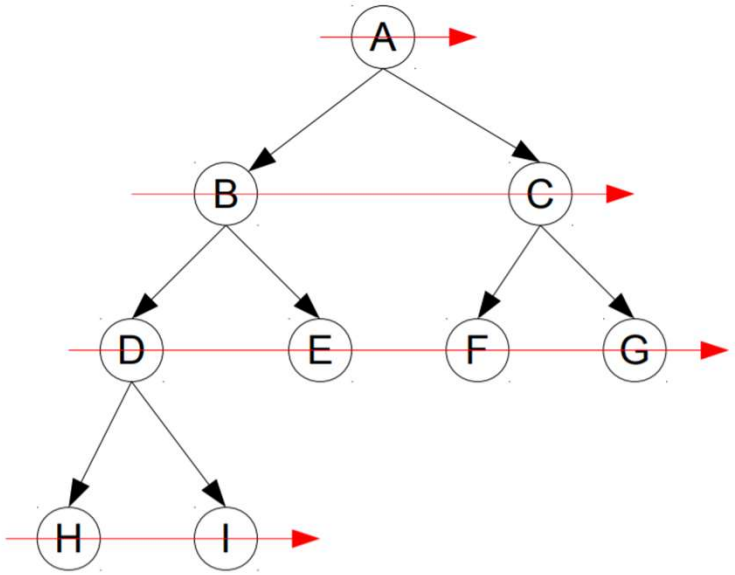
- Busca em Largura (BFS) e Busca em Profundidade (DFS).
- Algoritmo A (A estrela)*.
- *Simulated Annealing*.
- *Genetic Algorithm*
- *Artificial Immune Systems*
- *Particle Swarm Optimization*
- *Ant Colony System*



Resolução de Problemas - Busca e Otimização

Busca cega em amplitude ou largura (*breadth-first search*)

- É um algoritmo de busca em grafos utilizado para realizar uma busca ou travessia num grafo e estrutura de dados do tipo árvore.
- BFS expande todos os nós em um nível antes de passar para o próximo.
- Grande uso de memória em árvores muito horizontais.

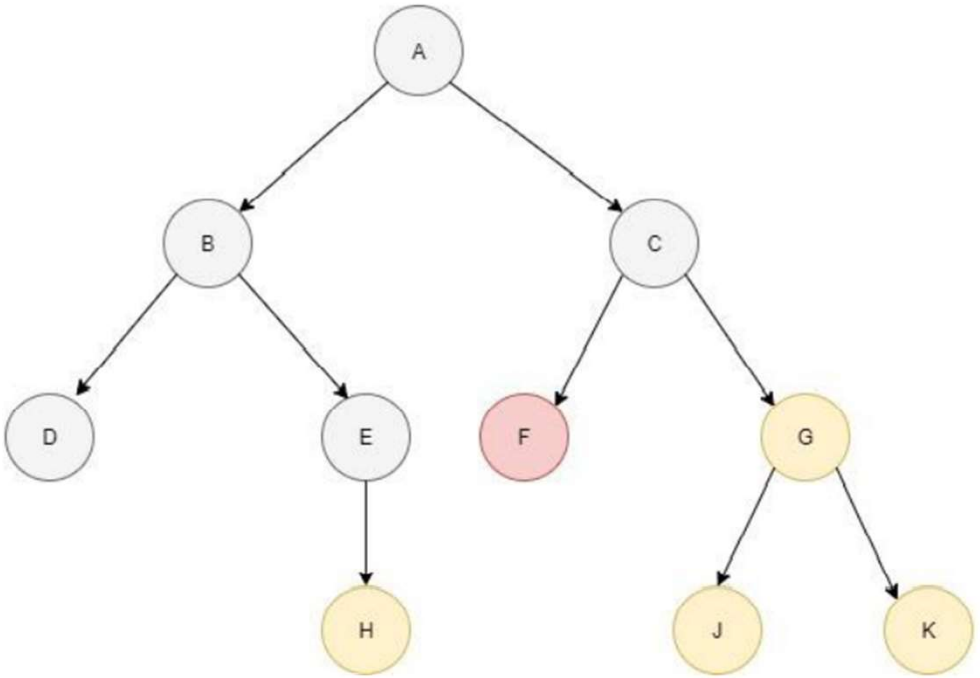


Ordem: A,B,C,D,E,F,G,H,I

Resolução de Problemas - Busca e Otimização

Busca cega em amplitude ou largura (*breadth-first search*)

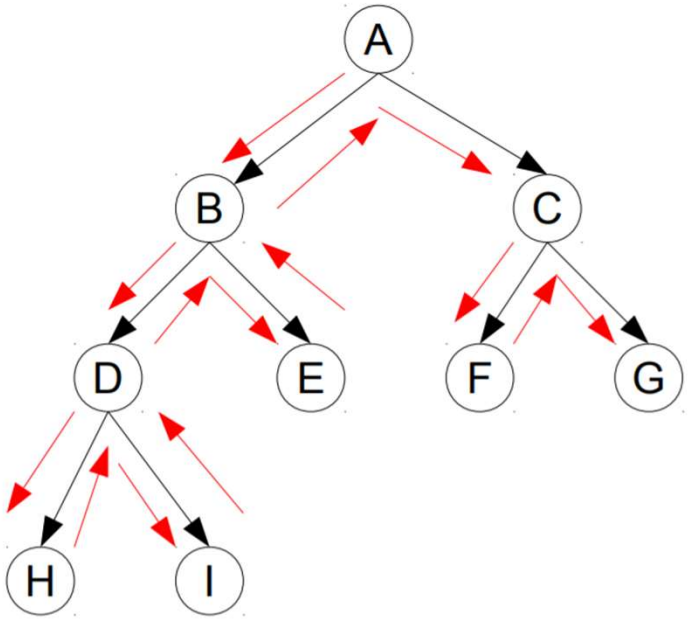
ID	Nó atual	FILA	É objetivo ?	Tem filhos?
1	A	BC	Não	Sim
2	B	CDE	Não	Sim
3	C	DEFG	Não	Sim
4	D	EFG	Não	Não
5	E	FGH	Não	Sim
6	**F**	**GH**	**Sim**	**Não**



Resolução de Problemas - Busca e Otimização

Busca cega profundidade (*depth-first search*)

- É um algoritmo usado para realizar uma busca ou travessia numa árvore, estrutura de árvore ou grafo.
- DFS desce pela árvore de busca o mais profundamente possível antes de retroceder.
- Intuitivamente, o algoritmo começa num nó raiz (selecionando algum nó como sendo o raiz, no caso de um grafo) e explora tanto quanto possível cada um dos seus ramos, antes de retroceder.



Ordem: A, B, D, H, I, E, C, F, G

Resolução de Problemas - Busca e Otimização

Comparação

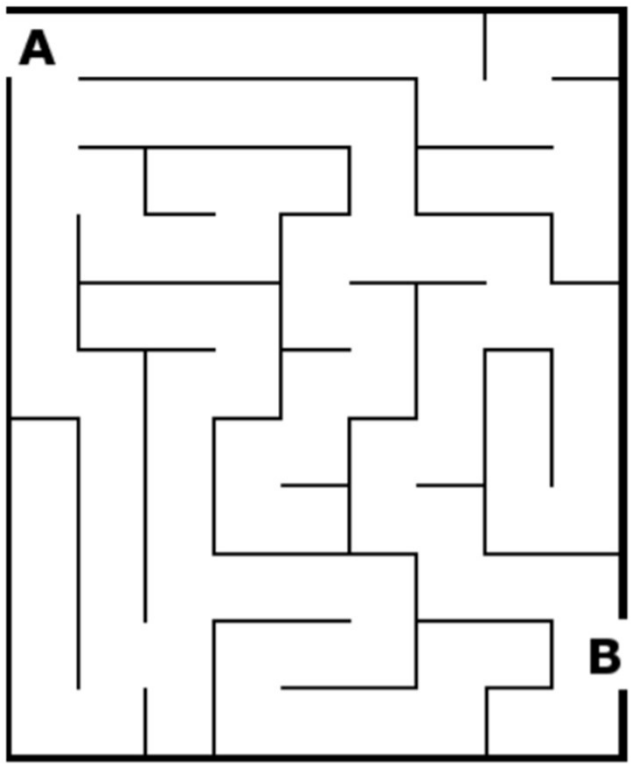
Cenário	Profundidade	Largura
Caminhos muito longos ou infinitos	Funciona mal	Funciona bem
Caminhos com comprimentos parecidos	Funciona bem	Funciona bem
Todos caminhos tem comprimentos parecidos e todos levam a um estado objetivo	Funciona bem	Desperdício de tempo e memória
Alto fator de ramificação	O desempenho depende de outros fatores	Funciona precariamente

Resolução de Problemas - Busca e Otimização

Humanos utilizam busca em profundidade

- É o modo mais fácil e natural.

- Exemplos:
  - Percorrendo um labirinto.
  - Ir a uma loja em um shopping para comprar um presente.



Resolução de Problemas - Busca e Otimização

Busca heurística

- Também conhecida como busca informada ou *best-fit.*

- A utilização de informações que indicam melhor qual caminho a seguir.
  - Ex: Descolcamento em cidade baseado no trânsito.

- De acordo com heurísticas e/ou informações específicas do problema, é possível definir uma ordem de preferência entre os caminhos possíveis a partir da raiz ou estado inicial.

- Logo, recorrendo a um pouco mais de informação inicial sobre o problema, é possível ser mais eficiente.

Resolução de Problemas - Busca e Otimização

Busca heurística

- Em outras palavras, o melhor caminho até a solução é obtido pela adoção de decisões do tipo *best-fit* a cada passo.

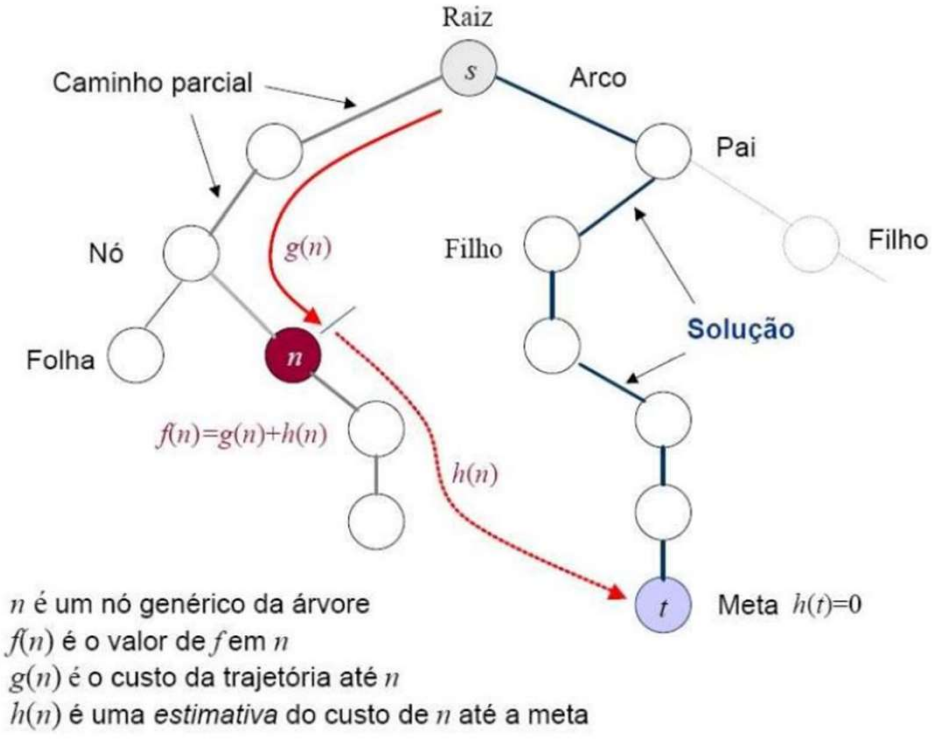
- Para tanto, estimar o custo de se chegar a cada nó e/ou o quão distante se está da solução passam a ser de grande relevância.

- A decisão para que nó seguir é determinada na busca:
  - Custo uniforme: pelo custo de se chegar àquele nó; ou
  - Gulosa: pelo custo estimado de se alcançar a solução a partir do nó corrente;

Resolução de Problemas - Busca e Otimização

Busca heurística

- g(n) : fator de altura, é o custo mínimo entre o nó-raiz e o nó n.
- h(n) : fator heurístico, é o custo mínimo estimado entre o nó n e o nó-solução, considerando todos os possíveis nós-solução e todos os possíveis caminhos entre esses dois nós.
- f(n) = g(n) + h(n) : é o custo mínimo entre o nó-raiz e um nó-solução, considerando todos os caminhos que passam pelo nó n.



Resolução de Problemas - Busca e Otimização

Busca heurística

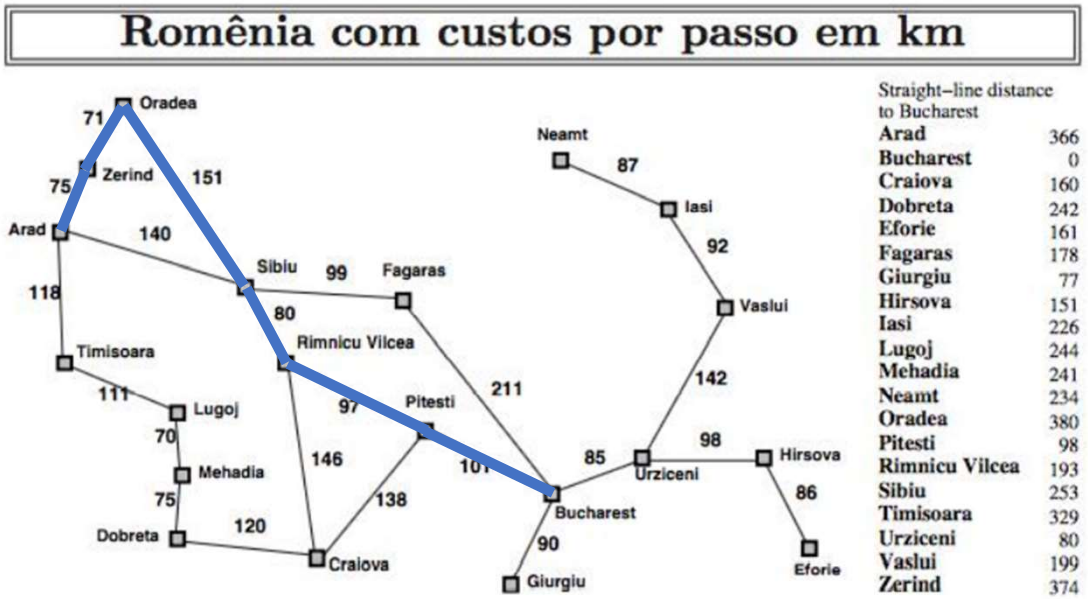
- Só são válidos junto a problemas para os quais, conforme aumenta a altura da árvore, o custo da solução vai aumentando monotonicamente, ou seja, o custo junto a cada aresta da árvore é não-negativo.

- Também, é necessário que o número de filhos de cada nó-pai da árvore de busca seja finito.

Resolução de Problemas - Busca e Otimização

Busca heurística

- Leva em conta apenas o fator de altura g(n), escolhendo o nó com o menor g a cada passo.
- Considere o problema ao lado de encontrar o menor caminho entre Arad e Bucharest num grafo.

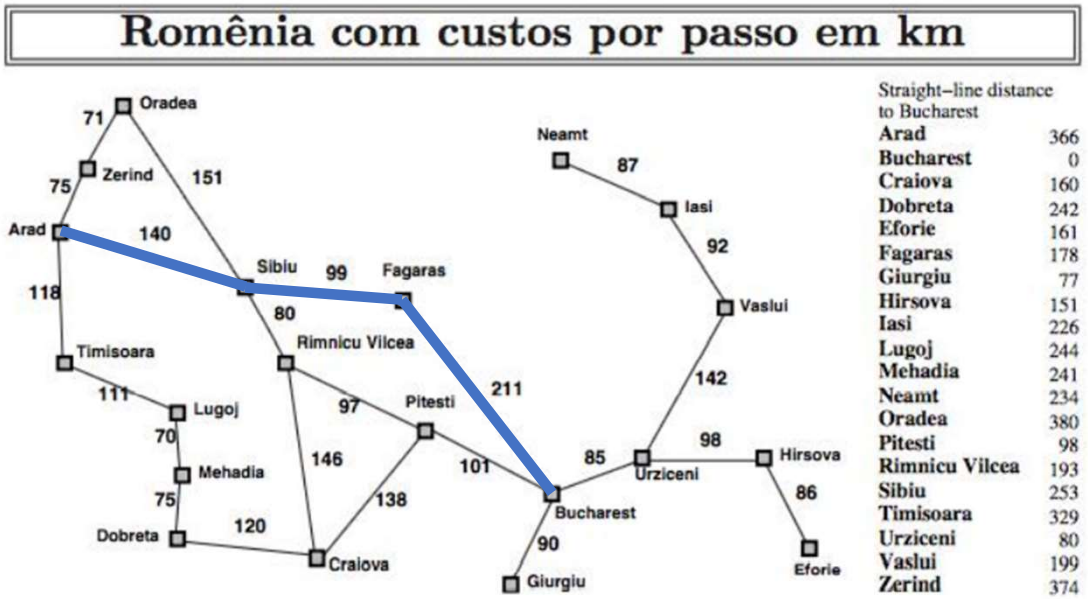


- Solução: Arad – Zerind – Oradea – Sibiu – Rimnicu – Pitesti – Bucharest.
- Total = 575

Resolução de Problemas - Busca e Otimização

Busca heurística gulosa (*greedy search*)

- Leva em conta apenas o fator heurístico h(n).



- Solução: Arad – Sibiu – Fagaras – Bucharest.
- Total = 450

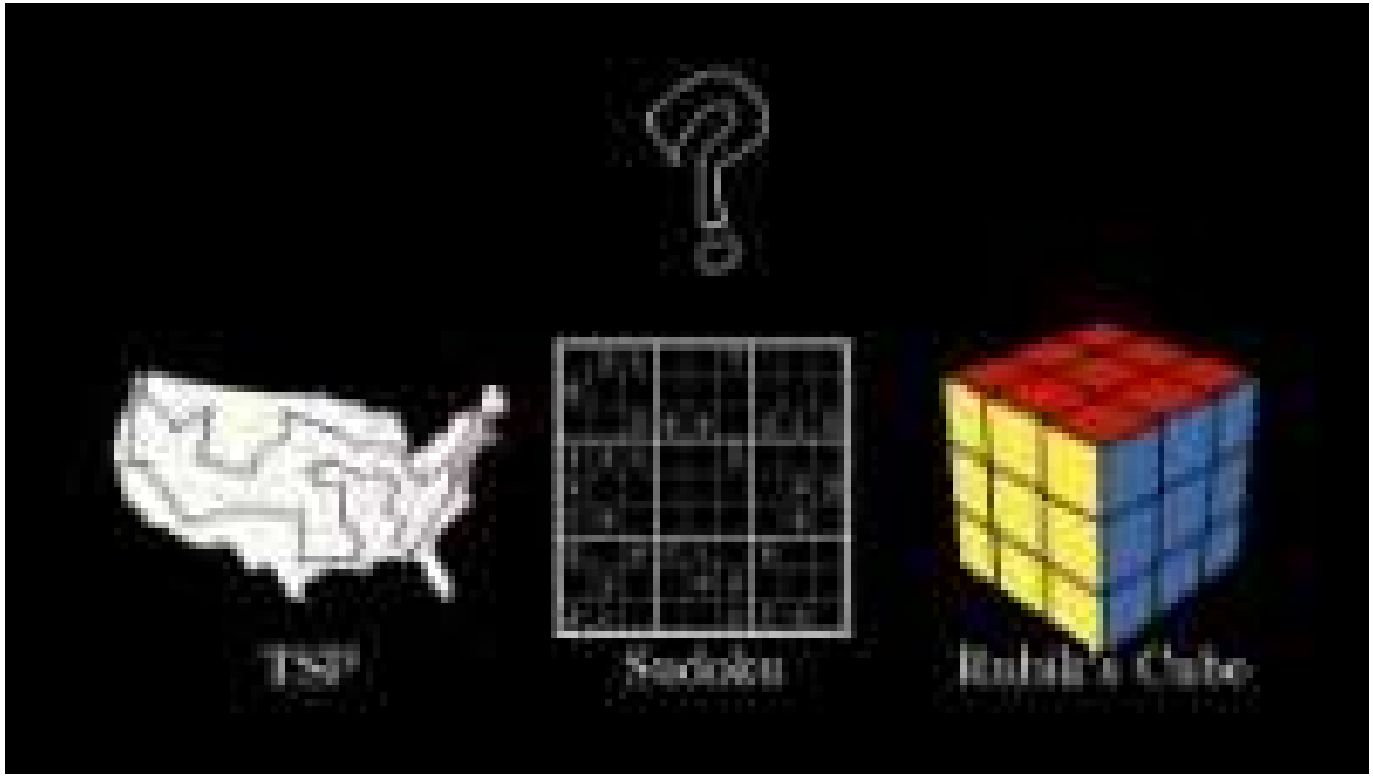
Resolução de Problemas - Busca e Otimização

Recozimento Simulado *(Simulated Annealing)*

- Inspirado no processo de aquecer uma liga e então deixá-la esfriá-la lentamente.
  Os átomos se movimentam bastante inicialmente e gradualmente se fixam em estados de energia mais baixos.
- Em processos metalúrgicos, os materiais atingem estágios ideais de resistência ao sofrer processo gradual de resfriamento.
- SA possui a característica de aceitar soluções piores com uma probabilidade decrescente ao longo das iterações.
- Utiliza uma variável que representa a temperatura.

Resolução de Problemas - Busca e Otimização

Simulated Annealing



Resolução de Problemas - Busca e Otimização

Simulated annealing

- Algoritmo:
- Inicialize a Temperatura T e o ponto de início aleatório;
- Enquanto o esfriamento <= num max de iterações:
  - Enquanto a iteração de temp <= num max de repetições:
    - ✓ Escolha um novo ponto da vizinhança;
    - ✓ Compute o custo atual;
    - ✓ S = custo atual – custo anterior;
    - ✓ Caso S < 0, aceite o vizinho;
    - ✓ Caso contrário, aceite com probabilidade de exp(-S/T);
- T = alfa * T ;
  - (0 < alfa < 1)

Resolução de Problemas - Busca e Otimização

Simulated annealing

- Primeira iteração:
  - Temperatura: 200
  - Custo solução 1: 38
  - Custo solução 2: 36
  - Delta: -2

  - Como Delta < 0, a solução 2 é aceita

Resolução de Problemas - Busca e Otimização

Simulated annealing

- Primeira iteração:
  - Temperatura: 200
  - Custo solução 1: 38
  - Custo solução 2: 40
  - Delta: 2

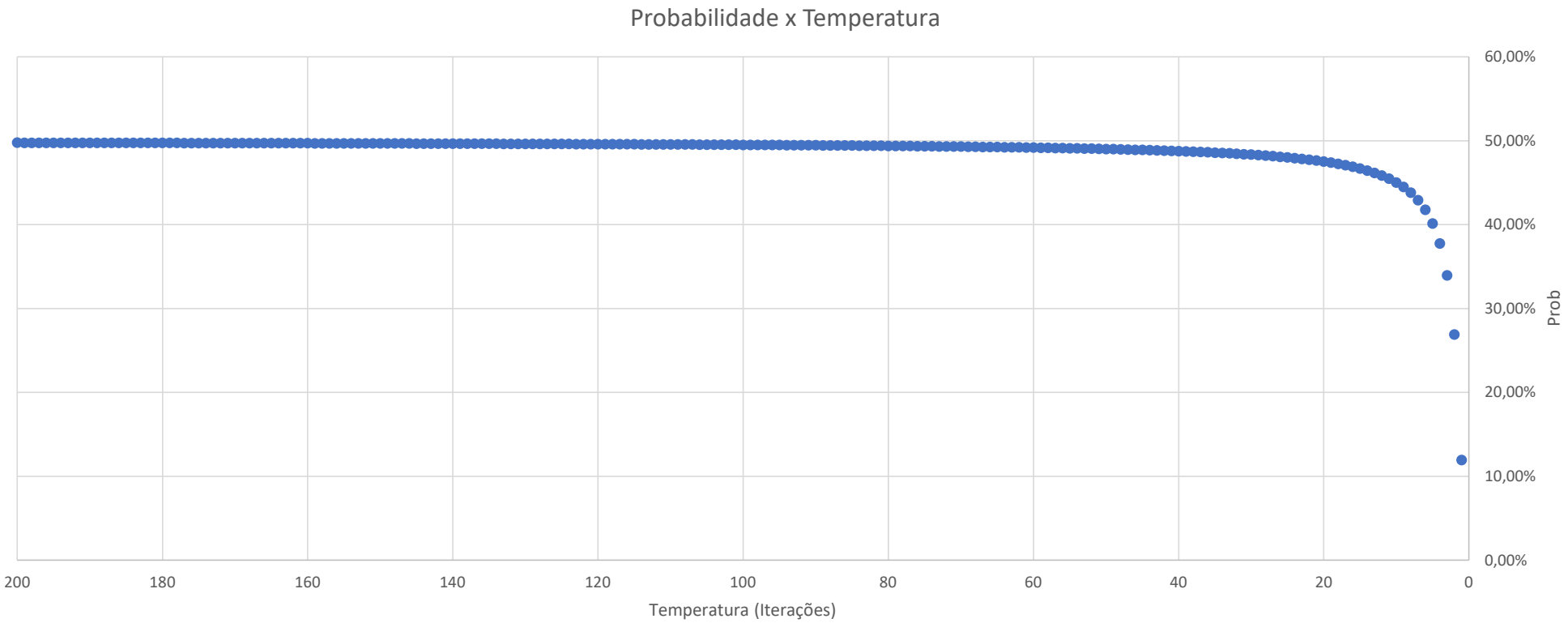
  - A probabilidade de aceitar a solução 2 é dada pela fórmula:

$$p = \frac{1}{1 + e^{\frac{Delta}{Temp}}}$$

  - p = 49.75%

Resolução de Problemas - Busca e Otimização

Simulated annealing



Resolução de Problemas - Busca e Otimização

Simulated annealing

- Primeira iteração:
  - Temperatura: 1
  - Custo solução 1: 38
  - Custo solução 2: 40
  - Delta: 2

  - A probabilidade de aceitar a solução 2 é dada pela fórmula:

$$p = \frac{1}{1 + e^{\frac{Delta}{Temp}}}$$

  - p = 11.92%

Resolução de Problemas - Busca e Otimização

Algoritmos evolucionários

- Algoritmos evolucionários são abordagens baseadas em heurística para resolver problemas que não podem ser facilmente resolvidos em tempo polinomial, como problemas clássicos NP-difíceis e qualquer outra coisa que levaria muito tempo para processar exaustivamente.

- Algoritmos evolucionários usam mecanismos inspirados pela evolução biológica, como reprodução, mutação, recombinação e seleção.

- As soluções candidatas para o problema de otimização desempenham o papel de indivíduos em uma população, e a função de *fitness* determina a qualidade das soluções. A evolução da população então ocorre após a aplicação repetida dos mecanismos descritos anteriormente.

Resolução de Problemas - Busca e Otimização

Algoritmos evolucionários

Resolução de Problemas - Busca e Otimização

Artificial Immune Systems (AIS)

- Sistemas Imunológicos Artificiais, do inglês *Artificial Immune Systems* (AISs), são um conjunto de técnicas computacionais inspiradas por sistemas imunológicos biológicos que solucionam problemas a partir de métodos como clonagem, teoria de rede imunológica ou vacinação.
- Geralmente, essas técnicas funcionam a partir do seguinte pressuposto: Existem indivíduos (anticorpos) que tentam assemelhar-se o máximo possível ao objetivo, o qual é composto por outros indivíduos chamados antígenos.
- A teoria de seleção clonal mostra que quando um animal é exposto a um antígeno, algumas de suas células respondem com a criação e seleção de anticorpos.
- Basicamente, um hemocitoblasto sofre diferenciação e reorganização genética para produzir linfócitos imaturos que se conectam a antígenos, onde muitos desses linfócitos não encontrarão os seus antígenos correspondentes, porém os que encontrarem irão produzir vários clones de si mesmo.

Resolução de Problemas - Busca e Otimização

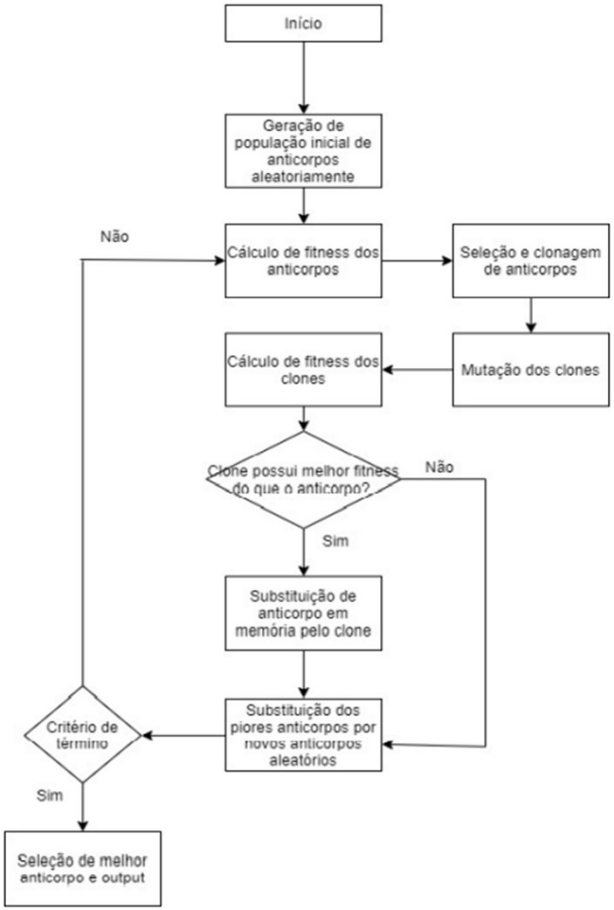
Artificial Immune Systems (AIS)

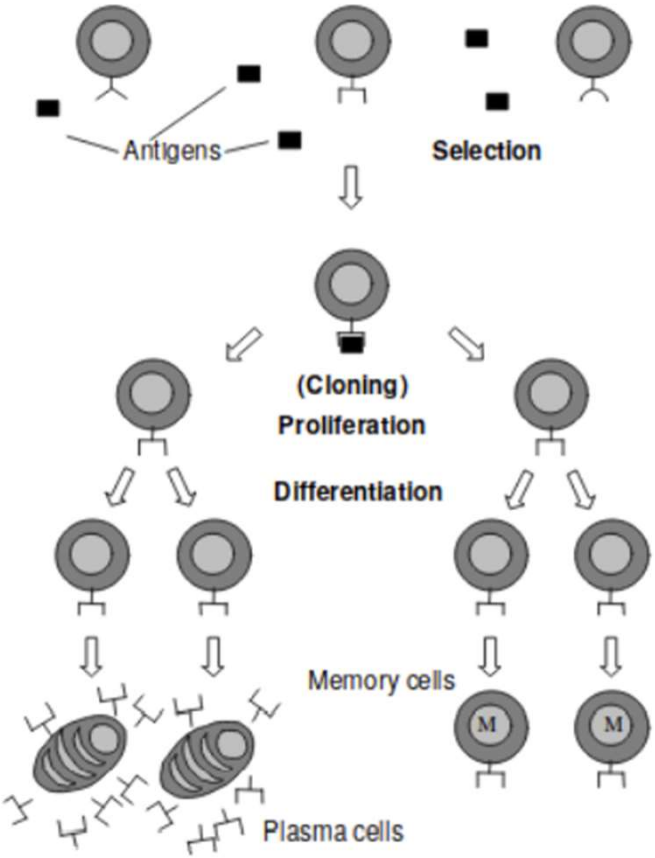
- O algoritmo baseado em clonagem pode ser descrito da seguinte maneira:

1. Anticorpos iniciados com valores aleatórios dos diversos parâmetros de entrada.
2. Calcula-se o custo da operação (solução) para cada um desses anticorpos.
3. Seleciona-se n anticorpos com as melhores soluções encontradas neste momento.
4. Clonam-se os n anticorpos selecionados de maneira independente e com proporções relacionadas a um ranking referente a proximidade com o objetivo.
5. Modificam-se aleatoriamente os C clones de maneira também proporcional ao ranking descrito.
6. Calcula-se o valor correspondente a cada solução para cada clone que sofreu mutação.
7. Substitui-se o anticorpo em memória pelo clone, caso este tenha melhor solução.
8. Substitui-se um número de anticorpos com os piores resultados de solução por novos anticorpos aleatórios.

Resolução de Problemas - Busca e Otimização

Artificial Immune Systems (AIS)

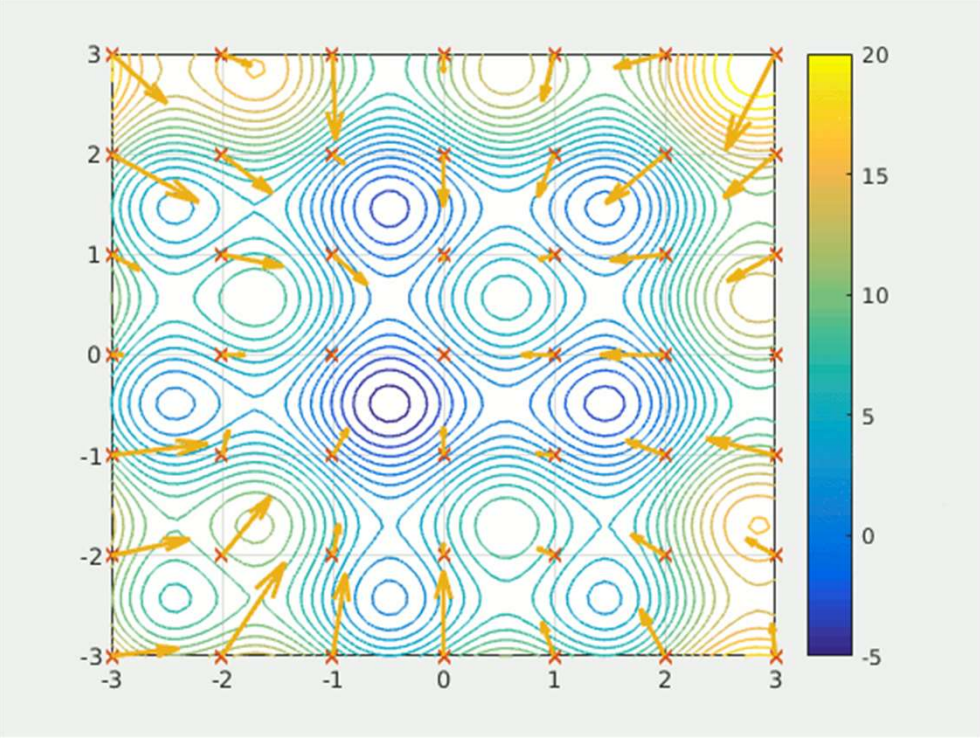




Resolução de Problemas - Busca e Otimização

Particle Swarm Optimization (PSO)

- O PSO foi inicialmente planejado para simular o comportamento social, como uma representação estilizada do movimento de organismos em um bando de pássaros ou cardume de peixes.



Resolução de Problemas - Busca e Otimização

Resolução de Problemas - Busca e Otimização

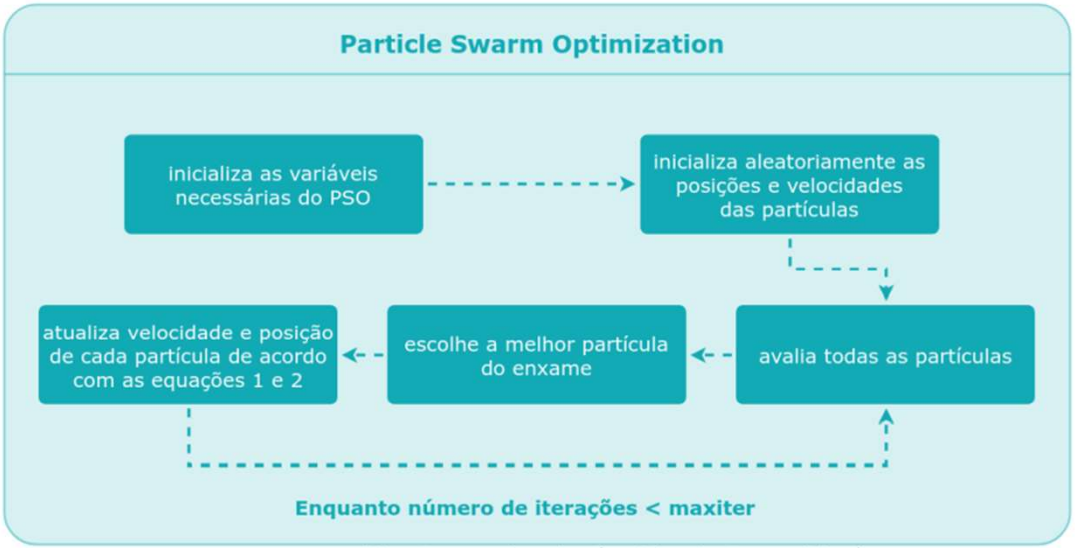
Particle Swarm Optimization (PSO)

- Método computacional que otimiza um problema tentando melhorar uma solução candidata em relação a uma determinada medida de qualidade
  - Iterativamente.
  - Conceito de partículas
    - ✓ Soluções candidatas.
- Abstrações
  - Partícula: vetor numérico representando os parâmetros de entrada.
  - Posição: vetor numérico representando a posição de cada partícula no grid.
  - Velocidade: escalar representando o fator de ajuste dos parâmetros de entrada.
  - *Fitness*: escalar representando o valor da solução de cada partícula.

Resolução de Problemas - Busca e Otimização

Particle Swarm Optimization (PSO)

- w = Constante de peso de inércia (valoriza a velocidade atual na próxima decisão)
- c1 = Constante cognitiva do indivíduo (valoriza as experiências do indivíduo)
- c2 = Constante de aprendizado social (valoriza as experiências do grupo)
- r1 e r2 = Valor aleatório entre 0 e 1 (evita convergências prematuras para um ótimo local)



Eq 1.
$$cognitive\_term = c1 \times r1 \times (position\_best_i - position_i),$$
$$social\_term = c2 \times r2 \times (position\_best_g - position_i),$$
$$velocity_i = w \times velocity_i + cognitive\_term + social\_term$$

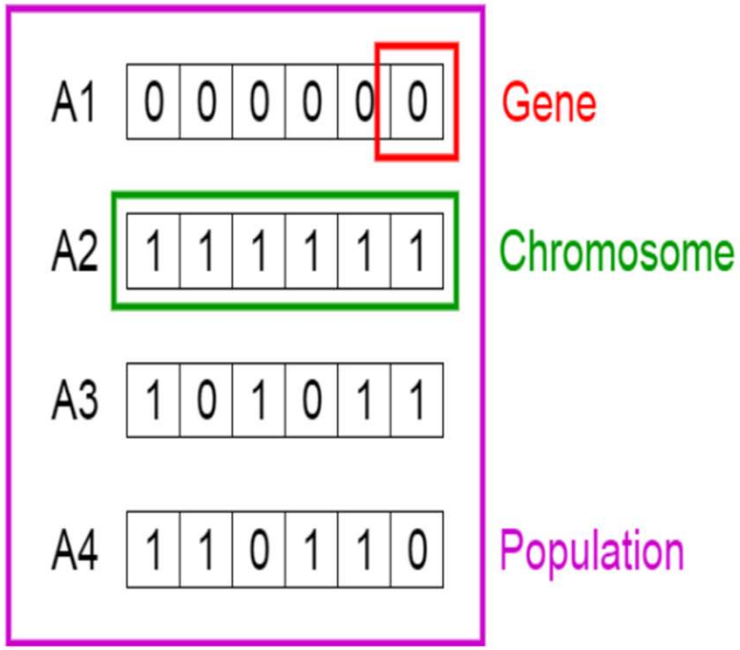
Eq 2.
$$position_i = position_i + velocity_i$$

$$cognitive\_term = c1 \times r1 \times (position\_best_i - position_i),$$
$$social\_term = c2 \times r2 \times (position\_best_g - position_i),$$
$$velocity_i = w \times velocity_i + cognitive\_term + social\_term$$

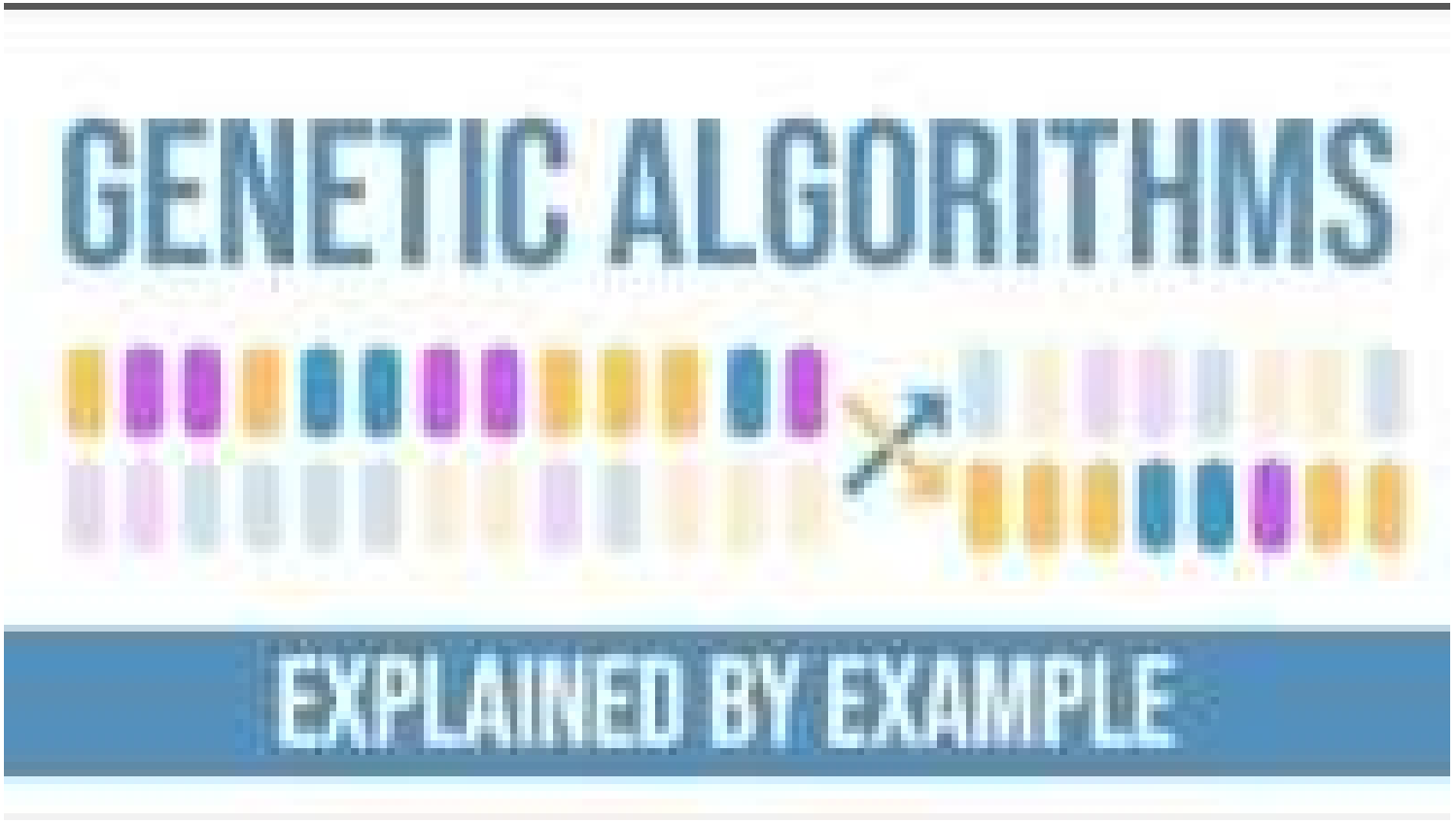
Resolução de Problemas - Busca e Otimização

Genetic Algorithm (GA)

- O Algoritmo Genético, do inglês *Genetic Algorithm* (GA), é uma abordagem computacional baseada na teoria de seleção natural que soluciona problemas de busca ou otimização.
- Neste método os indivíduos de uma população de tamanho arbitrário sofrem operações de cruzamento e mutação e passam por processos de seleção baseados em uma função de aptidão.

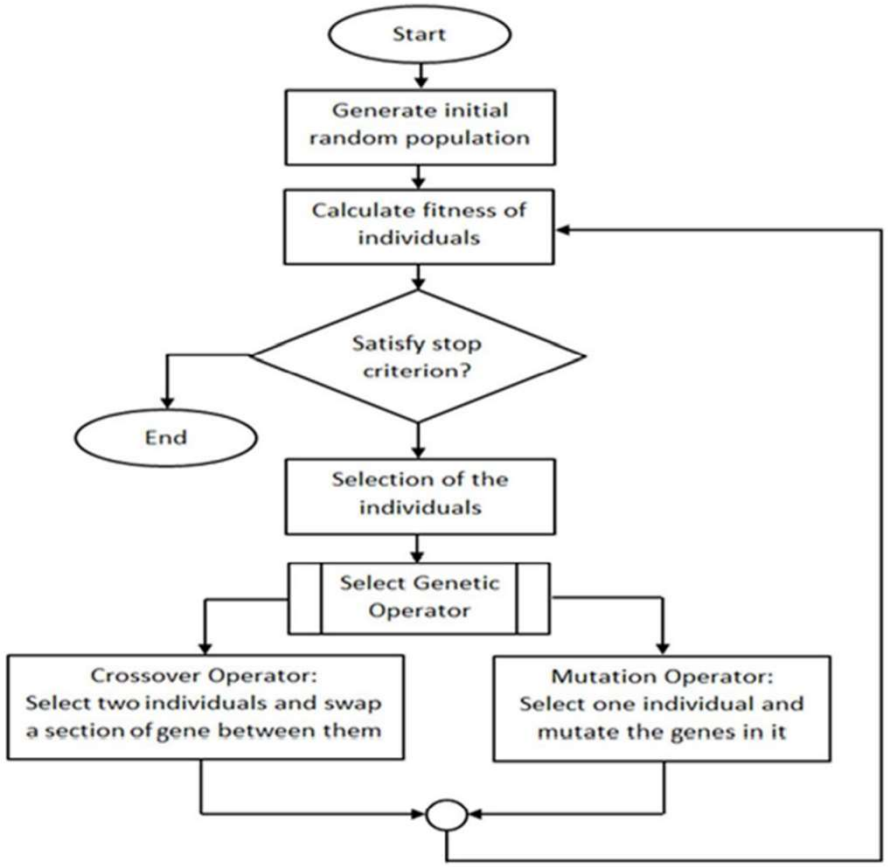


Resolução de Problemas - Busca e Otimização



Resolução de Problemas - Busca e Otimização

Genetic Algorithm (GA)



Resolução de Problemas - Busca e Otimização

Genetic Algorithm (GA)

- O algoritmo genético funciona da seguinte forma:

1. Cria-se uma população inicial com valores aleatórios de parâmetros de entrada.
2. Calcula-se o valor de fitness de cada indivíduo da população.
3. Aplica-se o operador de cruzamento com base na taxa de cruzamento pré-determinada;
4. Aplica-se o operador de mutação com base na taxa de mutação pré-determinada.
5. Aplica-se o operador de seleção do algoritmo.
6. Atualiza-se a população atual com os indivíduos selecionados para a próxima geração.
7. Este procedimento é repetido até que o número máximo de gerações seja atingido.

Resolução de Problemas - Busca e Otimização

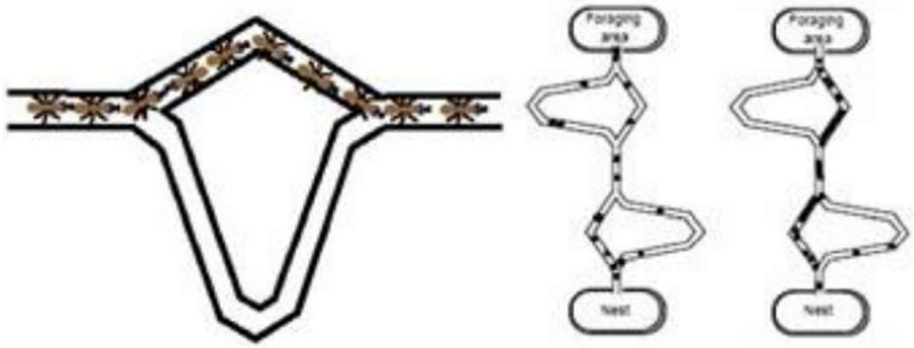
Ant Colony Optimization (ACO)

- Os algoritmos baseados em ACS (Ant Colony System), como o ACO, resolvem problemas NP-difíceis como o problema do caixeiro viajante.
- A formiga sai da colônia e desce os nós onde sempre encontram a comida na folha da árvore, quando é calculada a distância.
- As formigas tendem a seguir os caminhos que têm feromônios fortes, deixando feromônios nos caminhos proporcionalmente à distância percorrida por cada formiga, sendo que quanto menor a distância, bem como quanto mais formigas passam pelo caminho, mais forte ficam os feromônios dos caminhos.

Resolução de Problemas - Busca e Otimização

Ant Colony Optimization (ACO)

- Usando caminhos de tamanhos diferentes, as formigas convergem para o melhor caminho (mais curto e/ou menos custoso)
- O melhor caminho é percorrido em menos tempo/custo, fazendo com que as formigas o atravessem.
- Mais feromônios são depositados sobre esse caminho.
- As formigas, por fim, escolhem, com maior probabilidade, o melhor caminho, que contém mais feromônios



Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante

- O problema do caixeiro viajante é um problema que tenta determinar a menor rota para percorrer uma série de cidades (visitando uma única vez cada uma delas), retornando à cidade de origem.
- Inspirado na necessidade dos vendedores em realizar entregas em diversos locais (as cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.
- O problema do caixeiro viajante é NP-Completo.
  - Problemas NP são um conjunto de problemas cujas soluções são fáceis de verificar e são resolvidos por Máquina Não-Determinística em tempo polinomial.
  - Um problema é NP-difícil se for difícil de resolver ou encontrar uma solução.
  - Um problema é NP-completo se for NP e NP-difícil. Portanto, existem duas propriedades essenciais e intuitivas para NP-completo: Fácil de verificar, mas difícil de encontrar soluções.

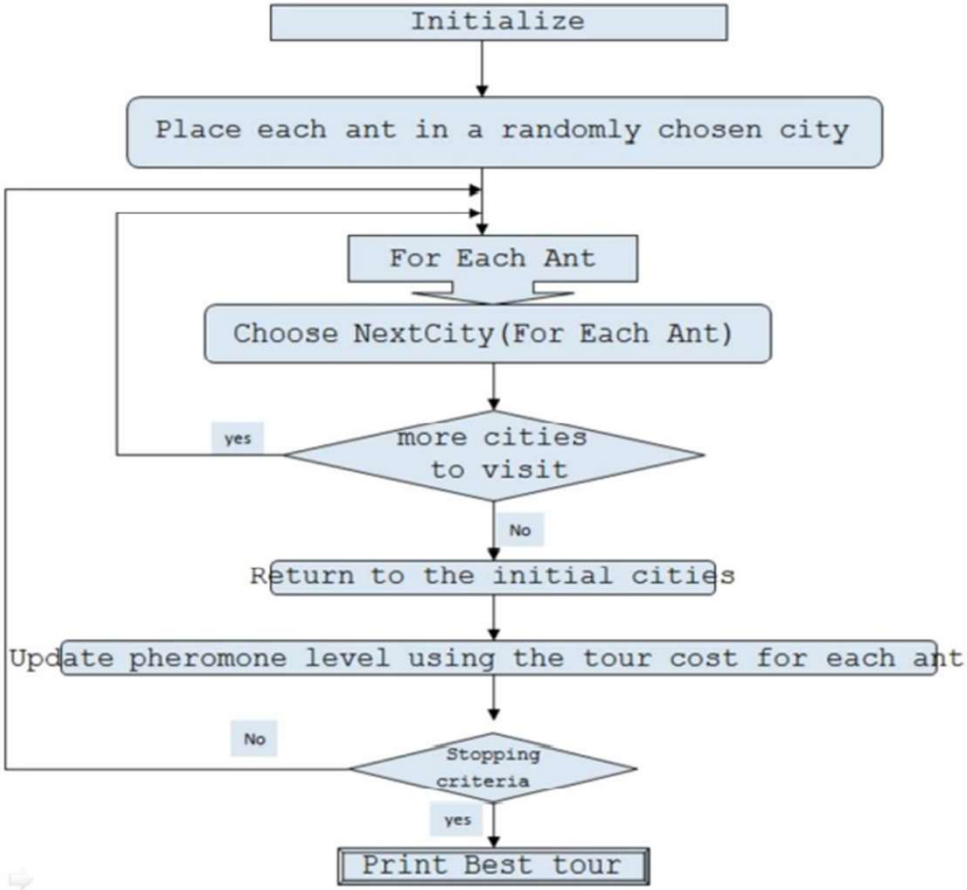
Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante

- As possibilidades são definidas por (n-1)! / 2
- Por exemplo, no caso de 4 cidades, consideramos 3 soluções possíveis.
- Porém, no caso de 31 cidades, teremos 30!/2 soluções. Considerando que um computador consegue realizar 106 operações por segundo, levaria 6.3*10^13 anos para percorrer todas as combinações possíveis de rotas. O que é mais do que o tempo de vida da Terra.

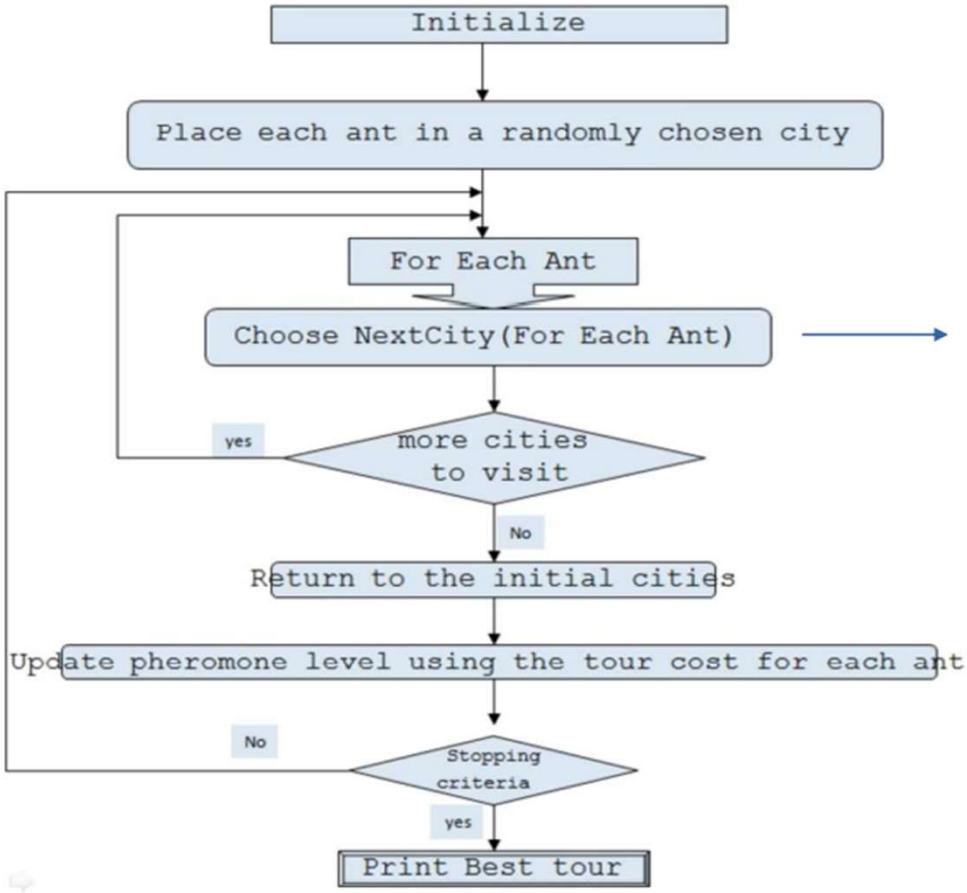
Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante



Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante

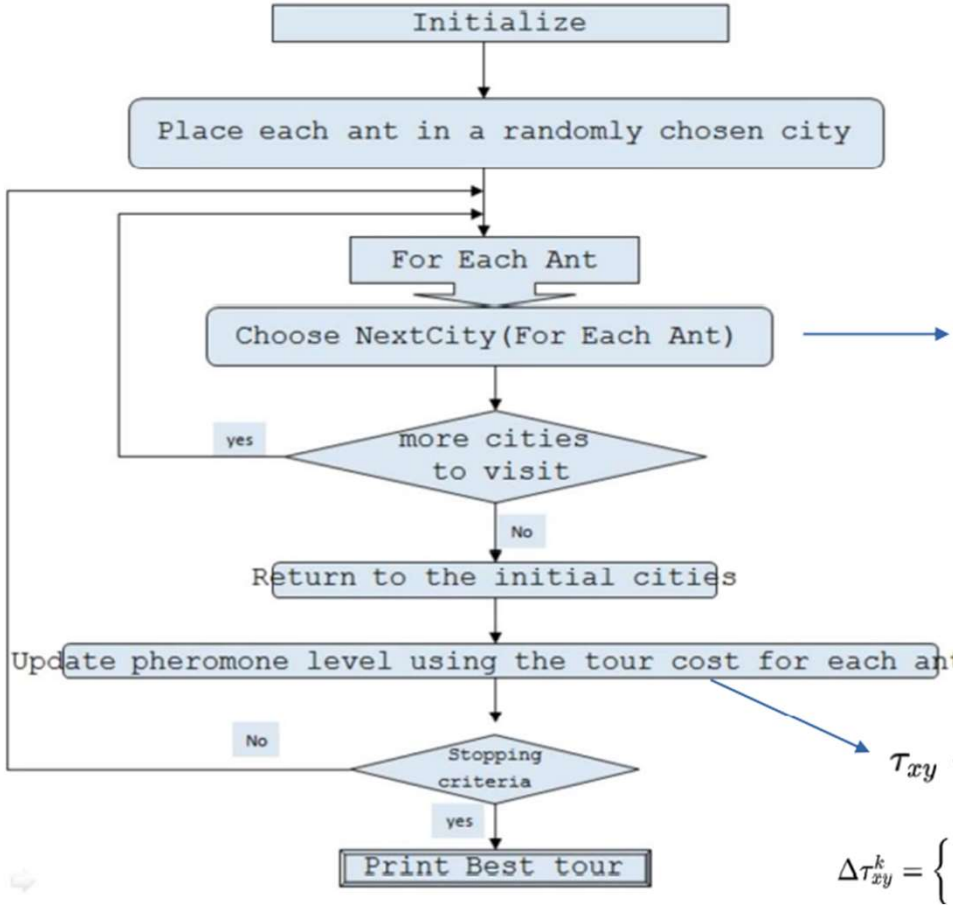


Probabilidade de escolha de caminho:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \text{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante



Probabilidade de escolha de caminho:

$$p_{xy}^k = \frac{(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}{\sum_{z \in \text{allowed}_x} (\tau_{xz}^\alpha)(\eta_{xz}^\beta)}$$

Atualização de feromônios

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^k$$

$$\Delta\tau_{xy}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Resolução de Problemas - Busca e Otimização

Problema do caixeiro viajante

$$p_{xy}^k = \frac{(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}{\sum_{z \in \text{allowed}_x} (\tau_{xz}^\alpha)(\eta_{xz}^\beta)}$$

τ_{xy} -> Quantidade de feromônios depositada na transição do estado x para o estado y, sendo 0 <= α o parâmetro que controla a influência desta quantidade.

η_{xy} -> Desejabilidade de transição do estado x para o y (geralmente é $1/d_{xy}$ onde d é a distância), sendo β >=1 o parâmetro que controla a influência desta desejabilidade.

τ_{xz} e η_{xz} -> Representam a quantidade de feromônios e desejabilidade de transição de outros possíveis estados.

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^k$$

$$\Delta\tau_{xy}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

ρ -> Coeficiente de evaporação de feromônio.

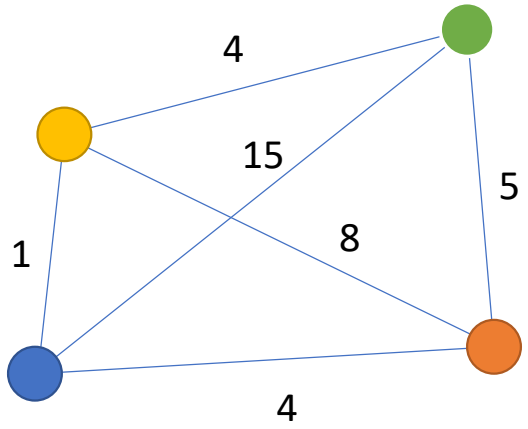
$\Delta\tau_{xy}^k$ -> Quantidade de feromônios depositados pela formiga *k*.

L_k -> Custo da formiga *k*.

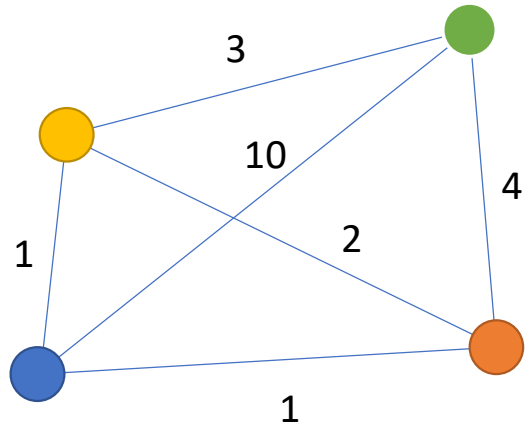
Q -> Constante arbitrária.

Resolução de Problemas - Busca e Otimização

Matriz de Custo
(Distância)



Matriz de Ferormônio

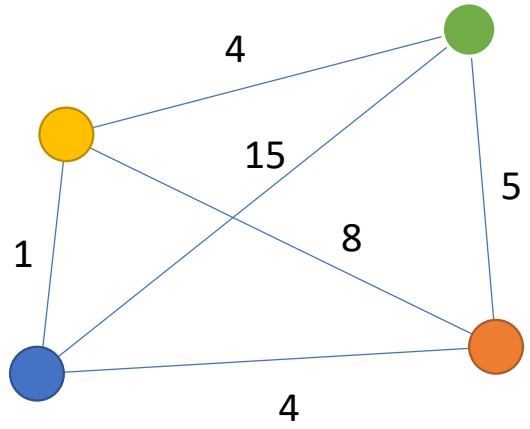


Resolução de Problemas - Busca e Otimização

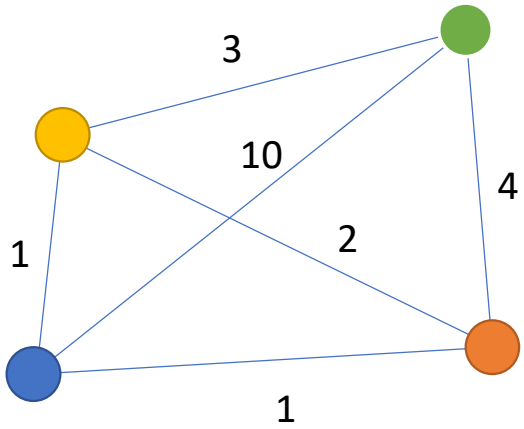
$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k} \\ 0 \end{cases}$$

Para o caminho entre I e J, a formiga k deposita:

- Zero se não passar no caminho
- O inverso do tamanho do caminho (Lk) encontrado



Matriz de Custo (Distância)

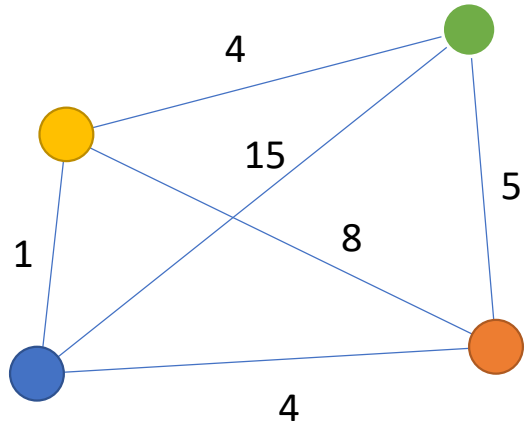


Matriz de Ferormônio

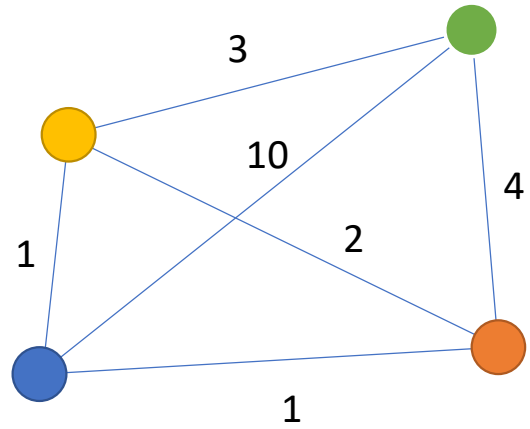
Resolução de Problemas - Busca e Otimização

$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

Quantidade de ferormônio sem evaporação é a soma do depositado por cada formiga.



Matriz de Custo (Distância)

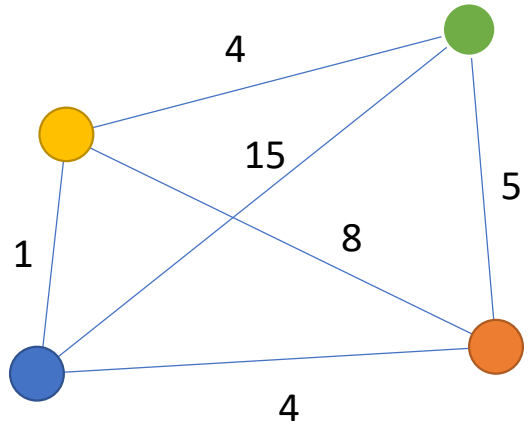


Matriz de Ferormônio

Resolução de Problemas - Busca e Otimização

$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

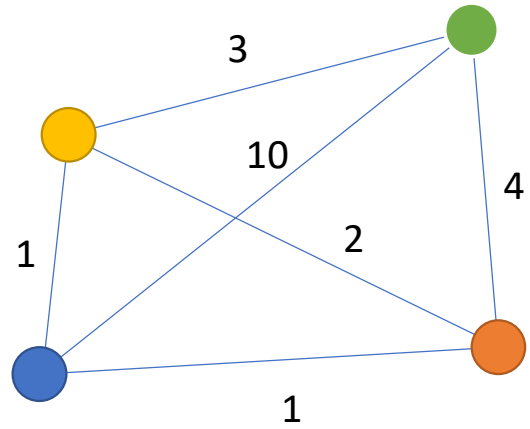
Quantidade de ferormônio sem evaporação é a soma do depositado por cada formiga.



Matriz de Custo (Distância)

$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

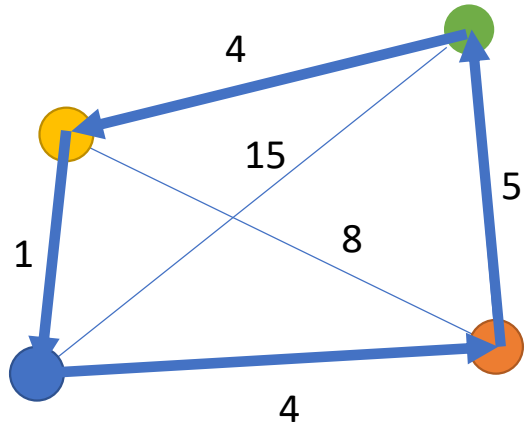
Quando há evaporação, considera-se uma nova parcela composta pela quantidade corrente de ferormônio e um fator rô ([0,1]), associado a um fator de evaporação.



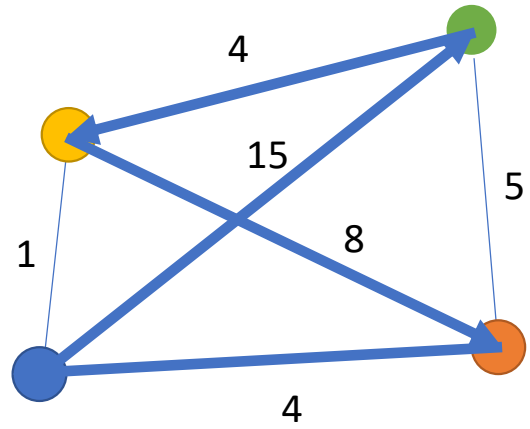
Matriz de Ferormônio

Resolução de Problemas - Busca e Otimização

$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$

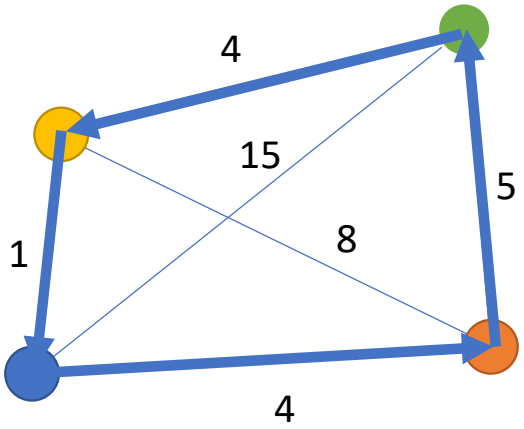


$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$

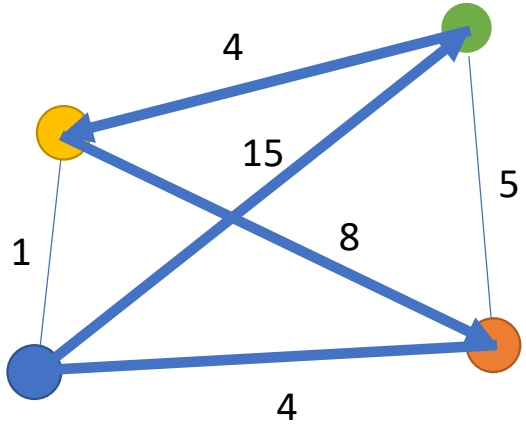


Resolução de Problemas - Busca e Otimização

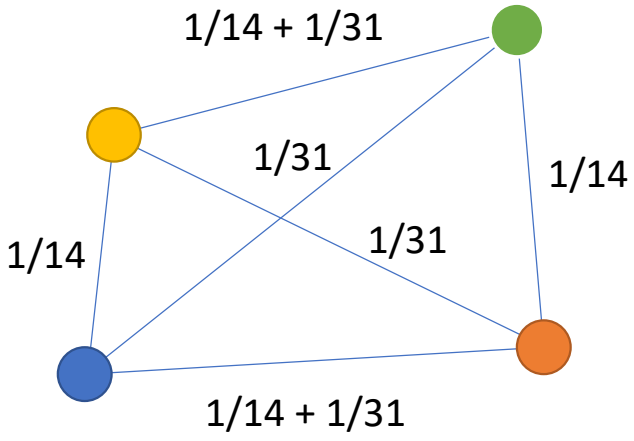
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



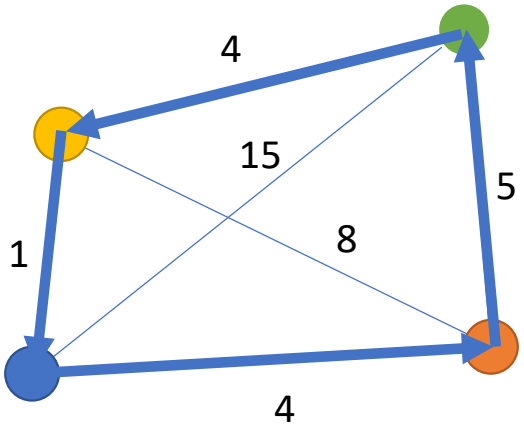
Soma dos Ferormônios



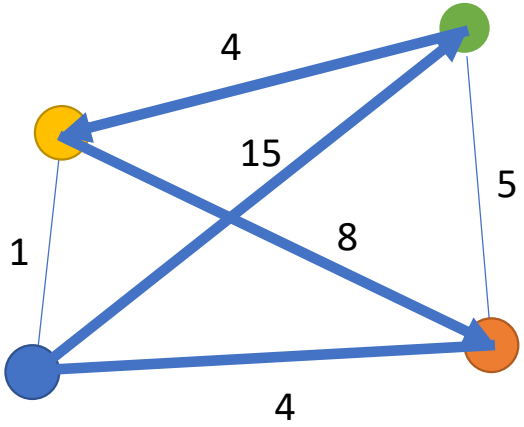
$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

Resolução de Problemas - Busca e Otimização

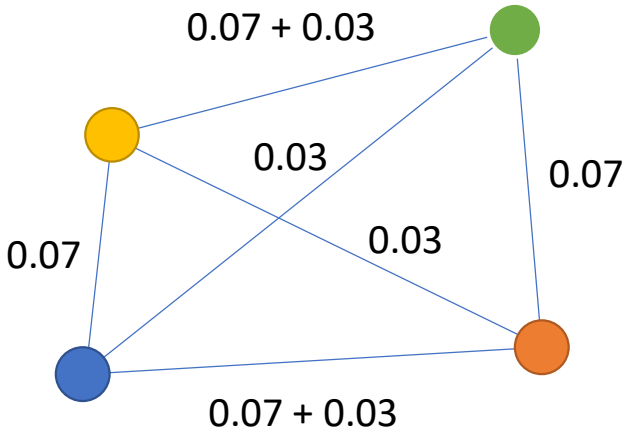
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



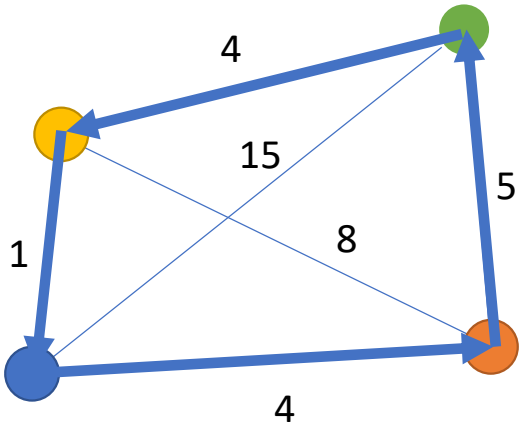
Soma dos Ferormônios



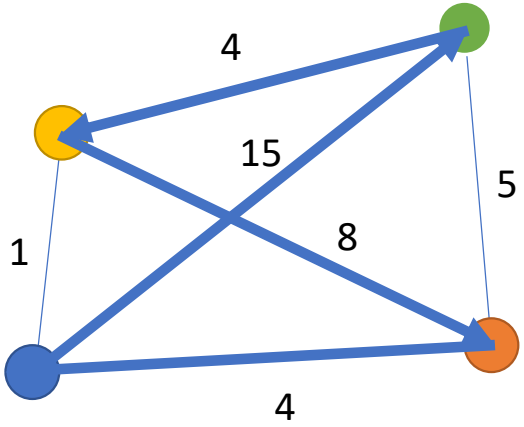
$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

Resolução de Problemas - Busca e Otimização

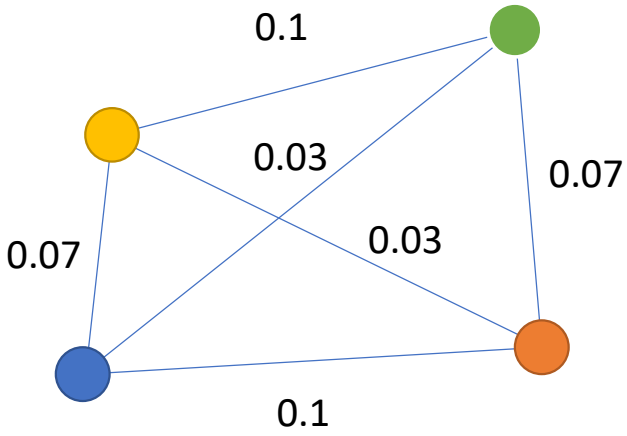
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



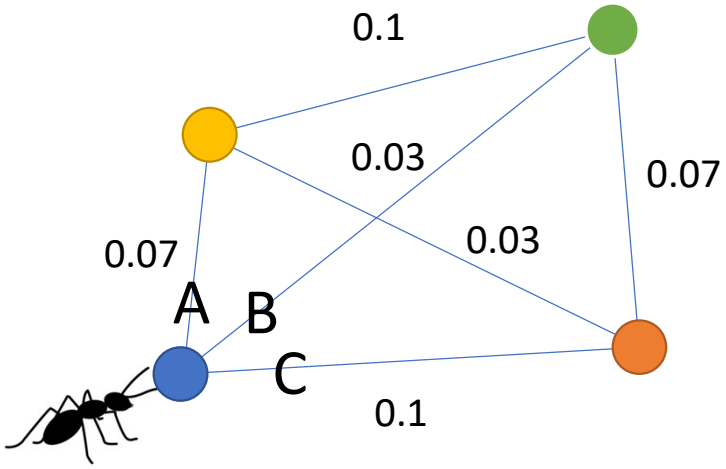
Soma dos Ferormônios



$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

Resolução de Problemas - Busca e Otimização

Soma dos Ferormônios

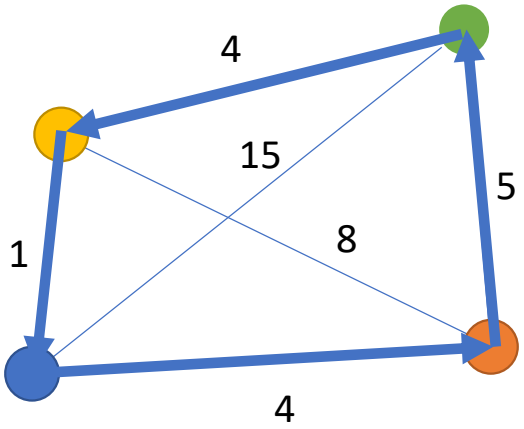


$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

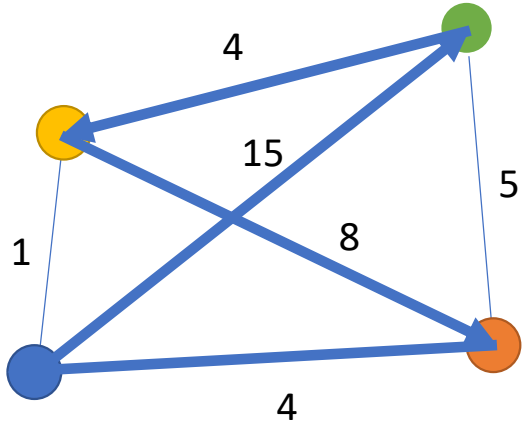
Saindo do ponto azul, qual o caminho a ser escolhido? A, B ou C?

Resolução de Problemas - Busca e Otimização

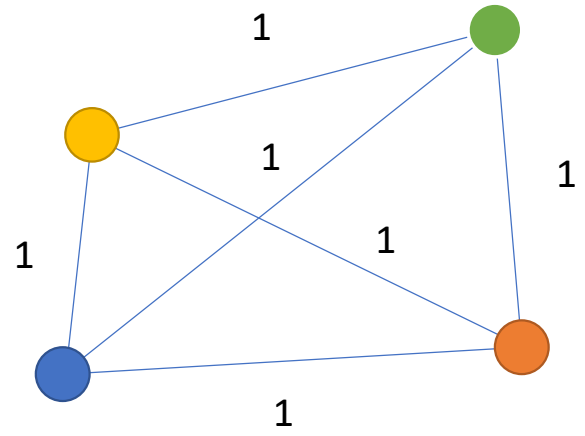
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



Soma dos Ferormônios

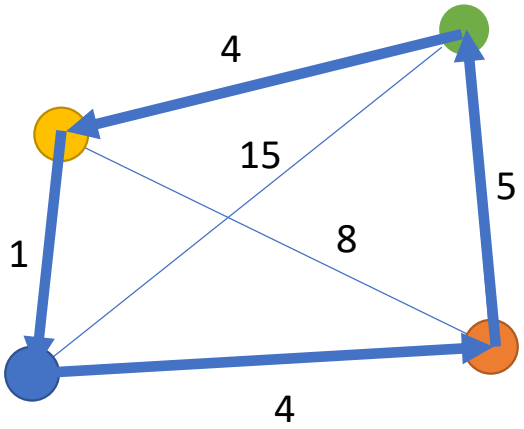


$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

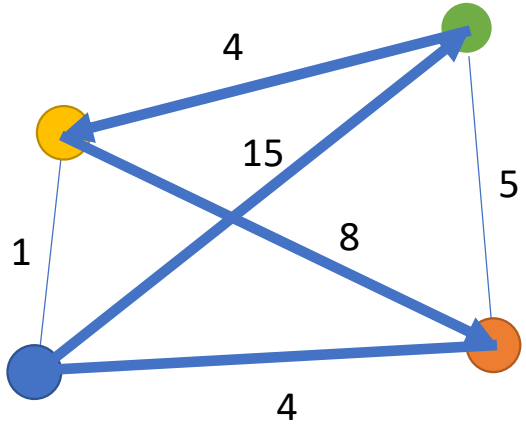
$$\rho = 0.5$$

Resolução de Problemas - Busca e Otimização

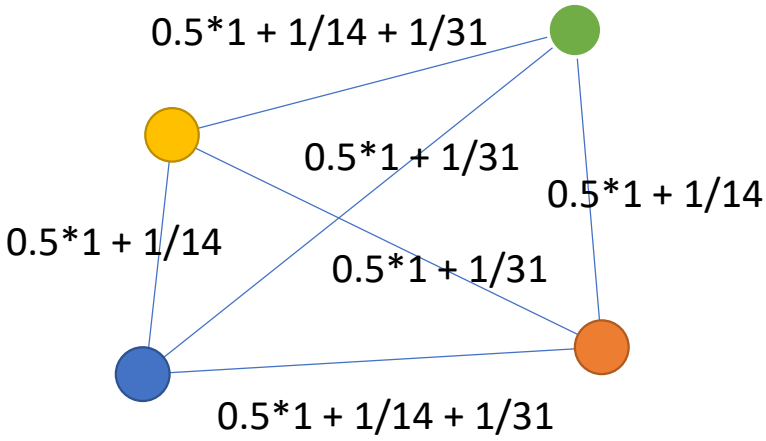
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



Soma dos Ferormônios

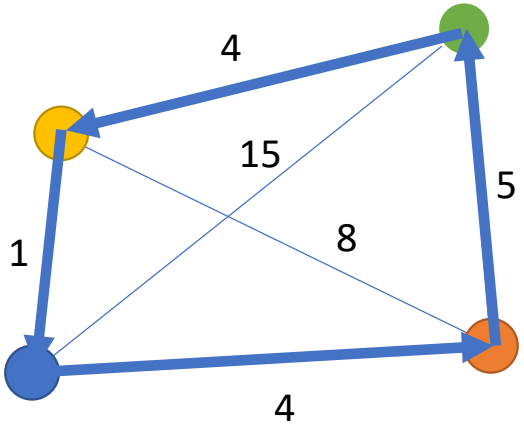


$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

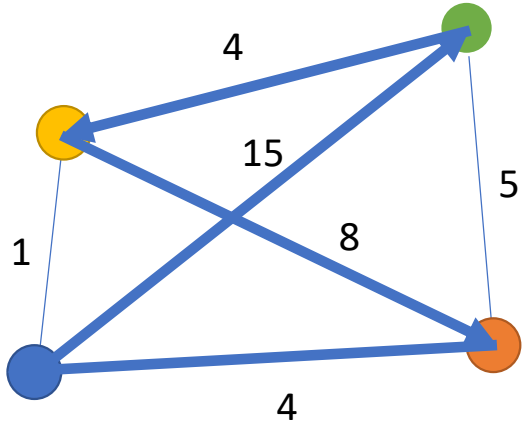
$$\rho = 0.5$$

Resolução de Problemas - Busca e Otimização

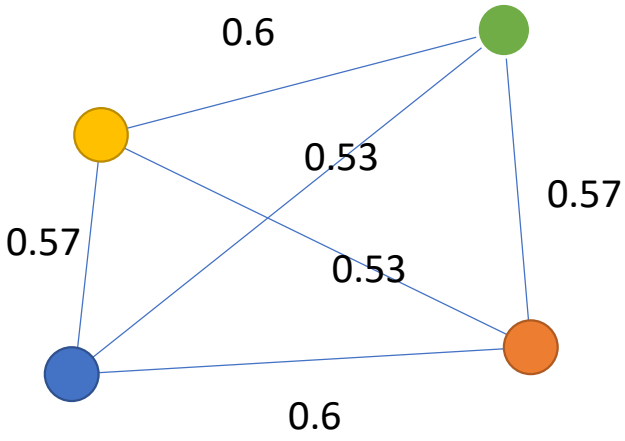
$$L1 = 4 + 5 + 4 + 1$$
$$L1 = 14$$
$$\frac{1}{L1} = \frac{1}{14}$$



$$L2 = 15 + 4 + 8 + 4$$
$$L2 = 31$$
$$\frac{1}{L2} = \frac{1}{31}$$



Soma dos Ferormônios

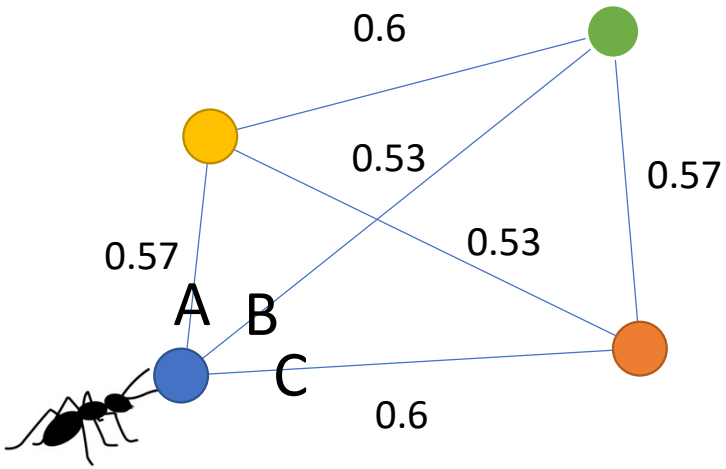


$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

$$\rho = 0.5$$

Resolução de Problemas - Busca e Otimização

Soma dos Ferormônios



$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

Saindo do ponto azul, qual o caminho a ser escolhido? A, B ou C?

Resolução de Problemas - Busca e Otimização

Como funciona no código?

O modelo da quantidade de ferormônio e do custo é representado por matrizes que serão manipuladas durante a execução do código.

Resolução de Problemas - Busca e Otimização

Como escolher o próximo passo?

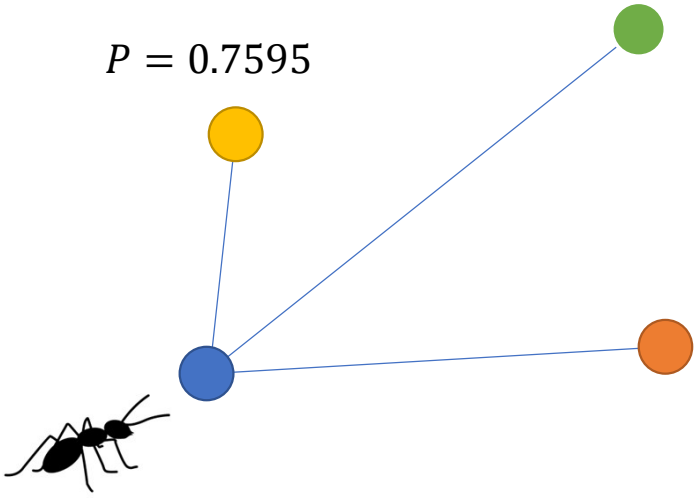
$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\sum\left((\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}\right)}$$

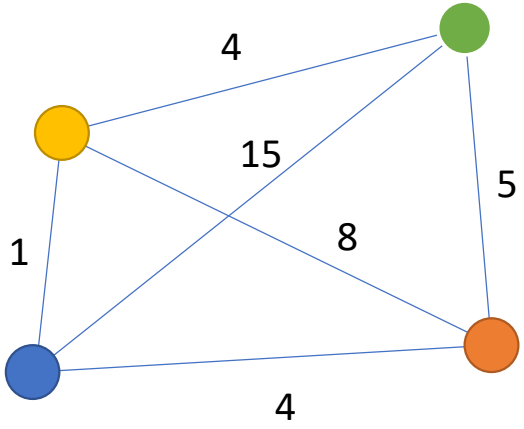
$$where: \eta_{i,j} = \frac{1}{\mathrm{L}_{i,j}}$$

Resolução de Problemas - Busca e Otimização

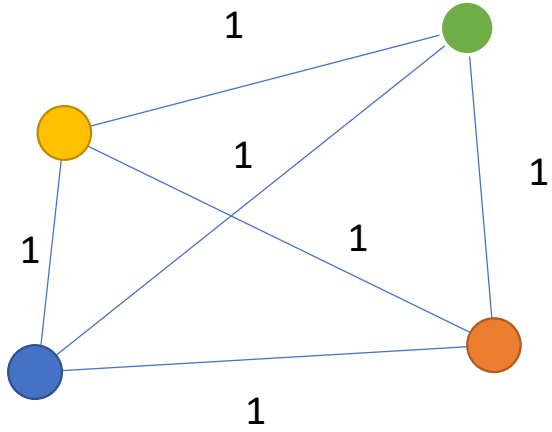
$$P = \frac{1 * \frac{1}{1}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 1 * \frac{1}{4}}$$

$$P = 0.7595$$



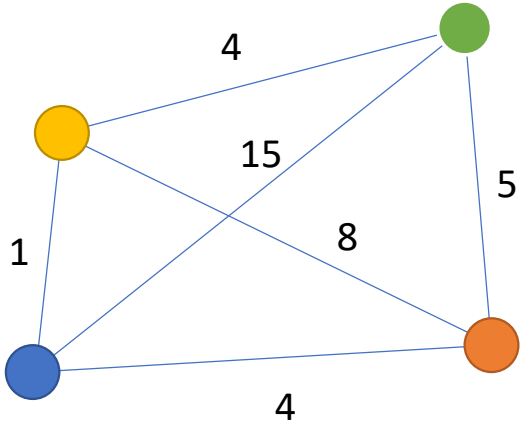


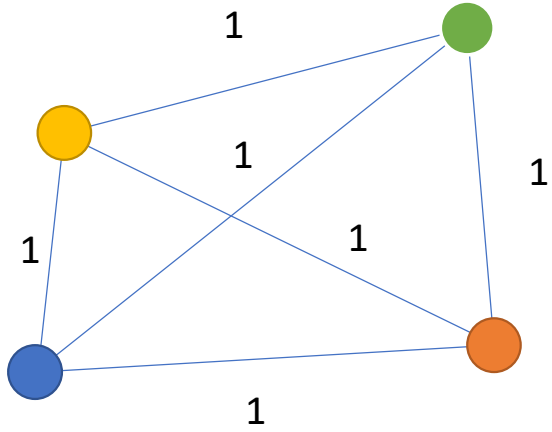
Distância



Ferormônios

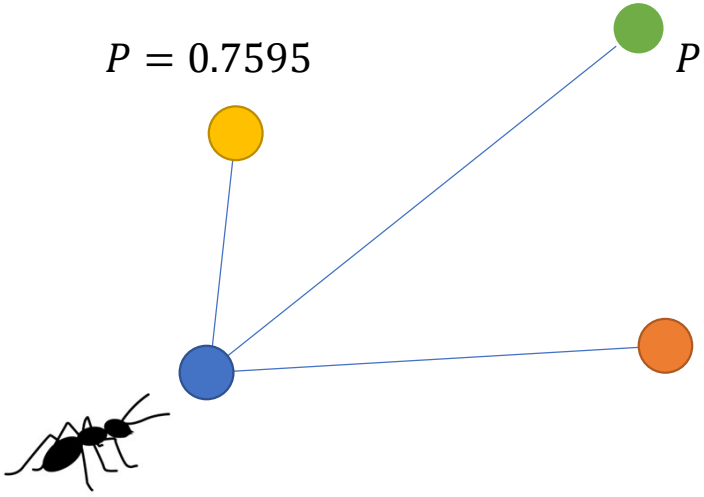
Resolução de Problemas - Busca e Otimização





$$P = \frac{1 * \frac{1}{1}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 1 * \frac{1}{4}}$$

$$P = 0.7595$$



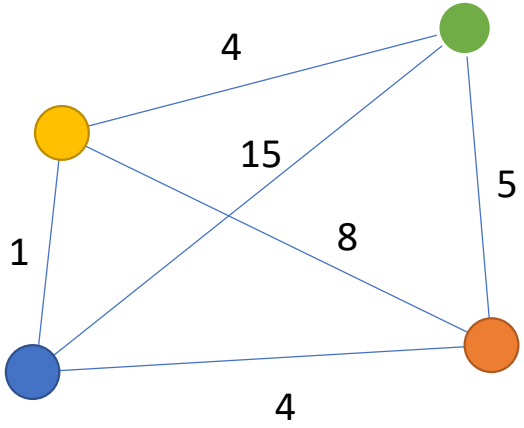
$$P = \frac{1 * \frac{1}{15}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 1 * \frac{1}{4}}$$

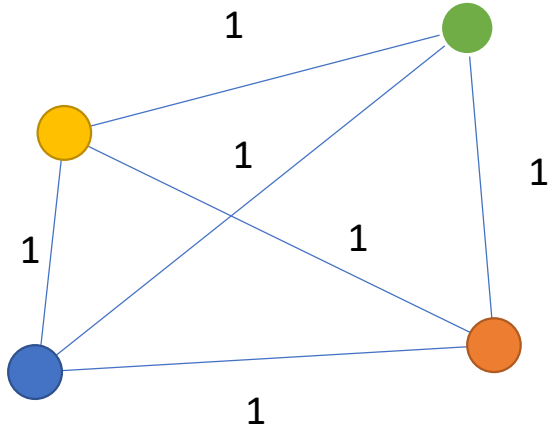
$$P = 0.0506$$

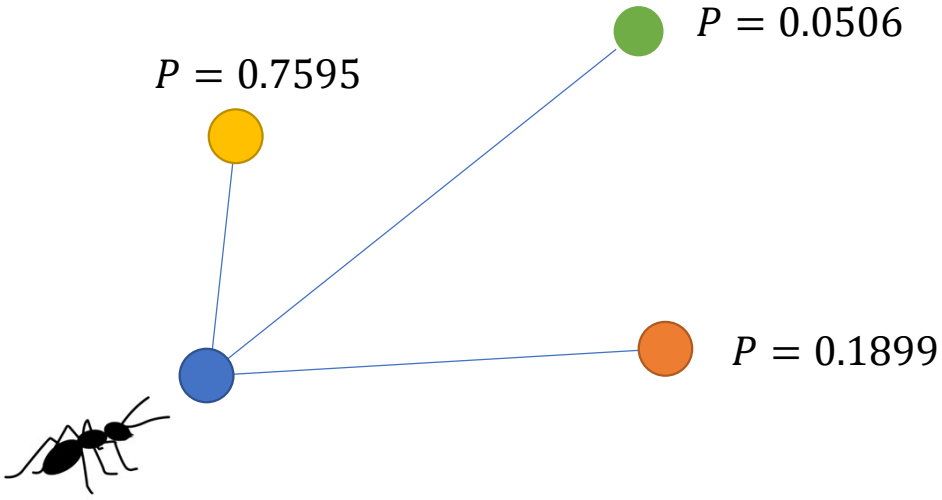
$$P = \frac{1 * \frac{1}{4}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 1 * \frac{1}{4}}$$

$$P = 0.1899$$

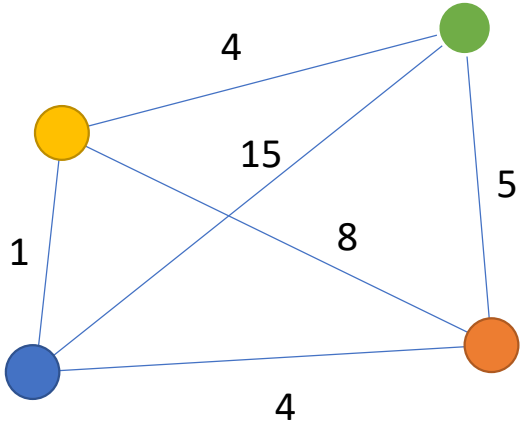
Resolução de Problemas - Busca e Otimização

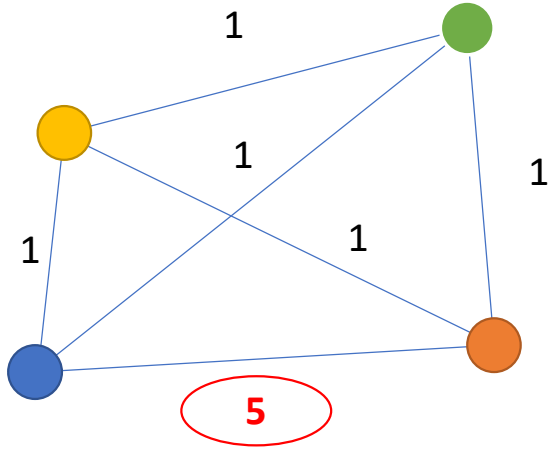


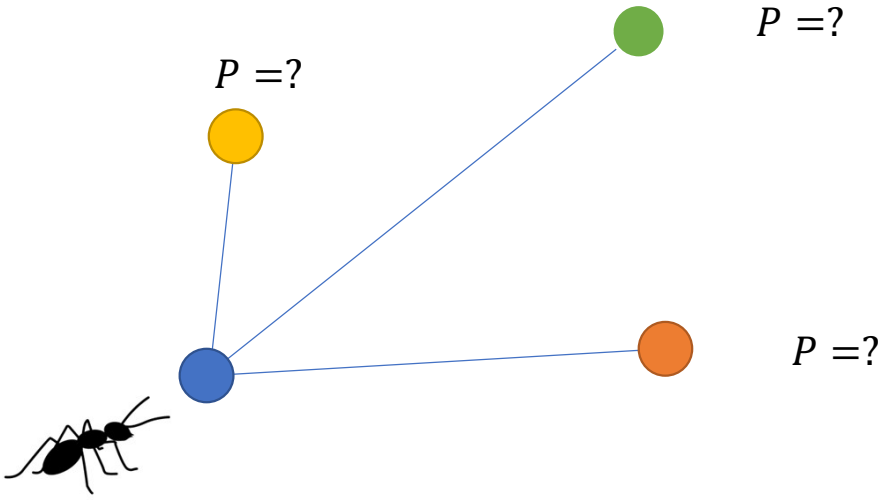




Resolução de Problemas - Busca e Otimização

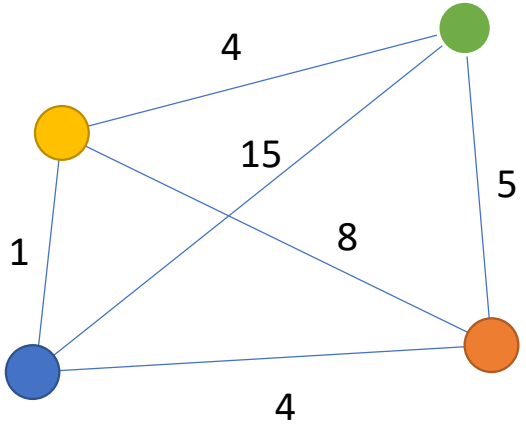


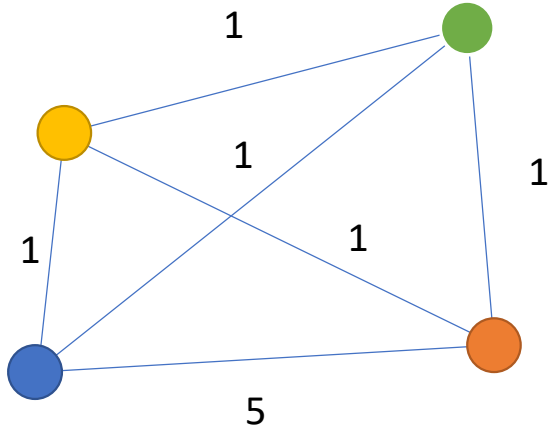




Vamos checar o impacto do aumento do ferormônio em um dos trechos

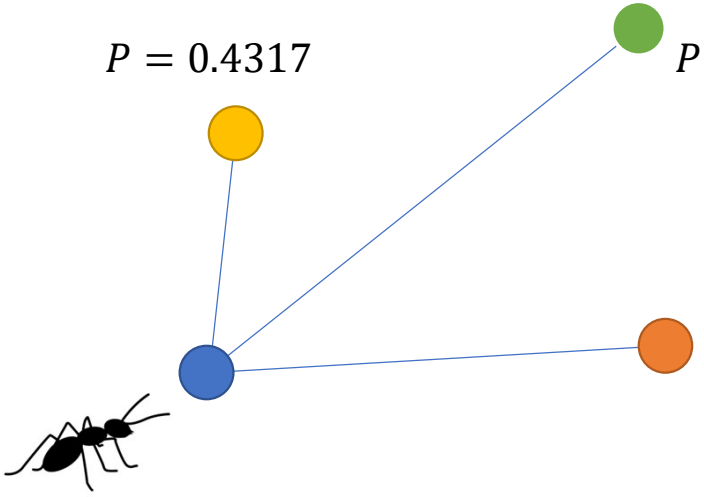
Resolução de Problemas - Busca e Otimização





$$P = \frac{1 * \frac{1}{1}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 5 * \frac{1}{4}}$$

$$P = 0.4317$$



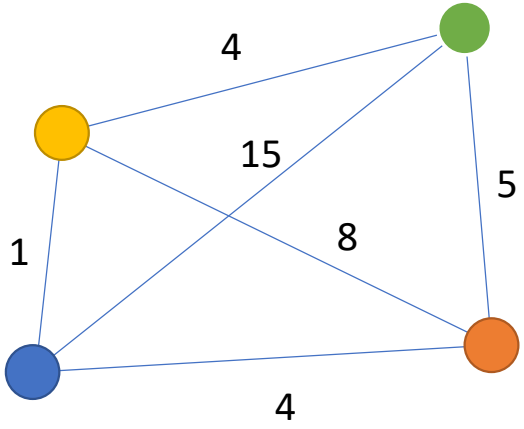
$$P = \frac{1 * \frac{1}{15}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 5 * \frac{1}{4}}$$

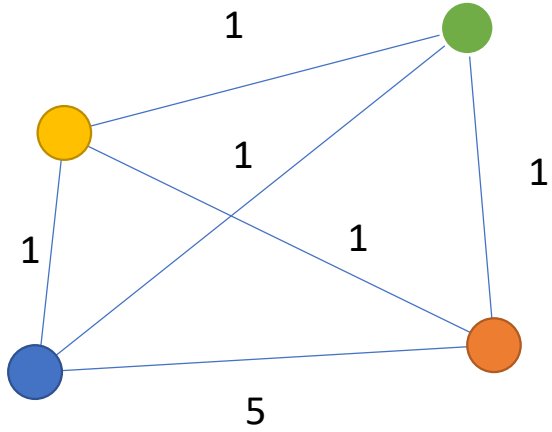
$$P = 0.0288$$

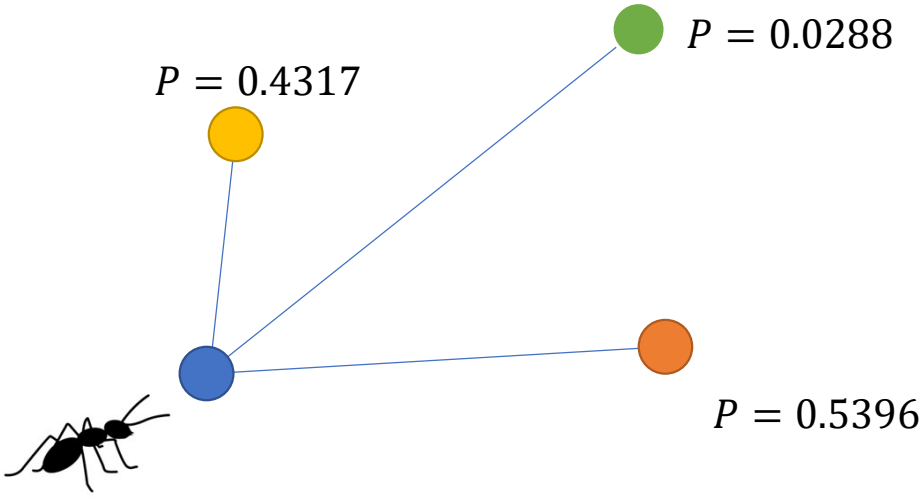
$$P = \frac{5 * \frac{1}{4}}{1 * \frac{1}{1} + 1 * \frac{1}{15} + 5 * \frac{1}{4}}$$

$$P = 0.5396$$

Resolução de Problemas - Busca e Otimização



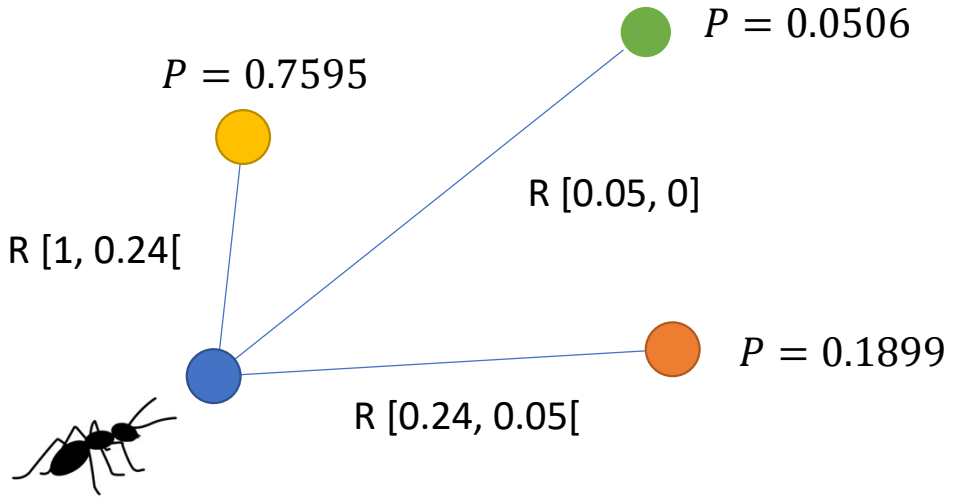




A probabilidade deste trecho aumentou consideravelmente.

Resolução de Problemas - Busca e Otimização

Roleta



Probabilidade	0.76	0.19	0.05
Acumulado	1	0.24	0.05

Sorteia-se um número R [0, 1]

E agora?

Comentários