Colônia de Formigas

Colônia de Formigas (Ant Colony)

O que é Otimização por Colônia de Formigas?

- Colônia de formigas é uma metaheurística baseada em população e inspirada no comportamento forrageiro das formigas.

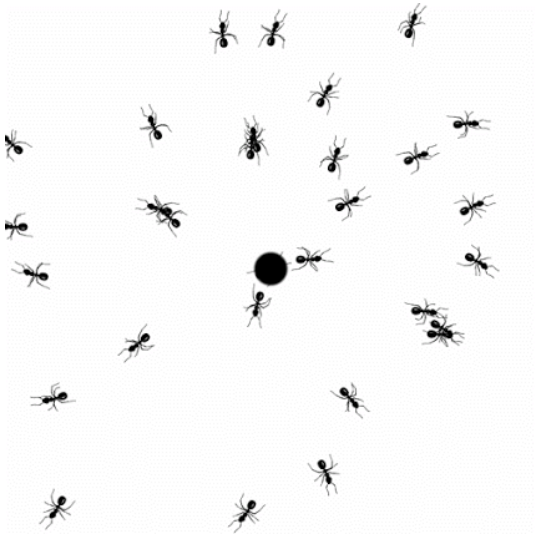
A Inspiração Biológica

- Muitas espécies de formigas são quase cegas.
- A comunicação entre as formigas é realizada através de uma substância química denominada de **feromônio**.
- Em algumas espécies, o feromônio é usado para criar caminhos (**trilhas de formigas**)

FORMIGAS FORRAGEIAM ALEATORIAMENTE

FORMIGAS DETECTAM O ALIMENTO

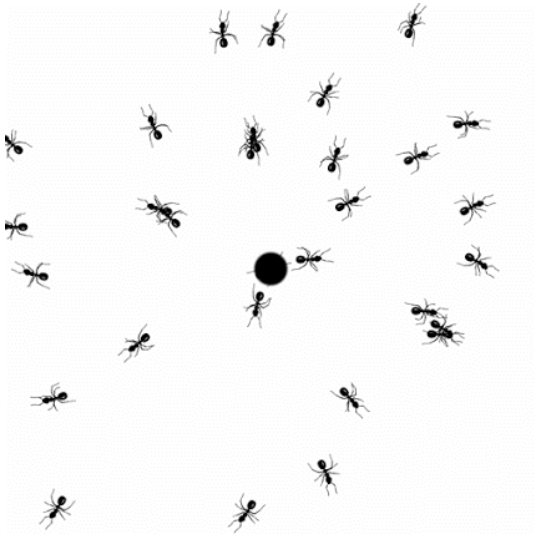
FORMIGUEIRO



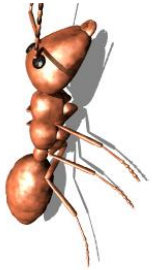
FORMIGAS DETECTAM O ALIMENTO

FORMAÇÃO DA TRILHA

FORMIGUEIRO

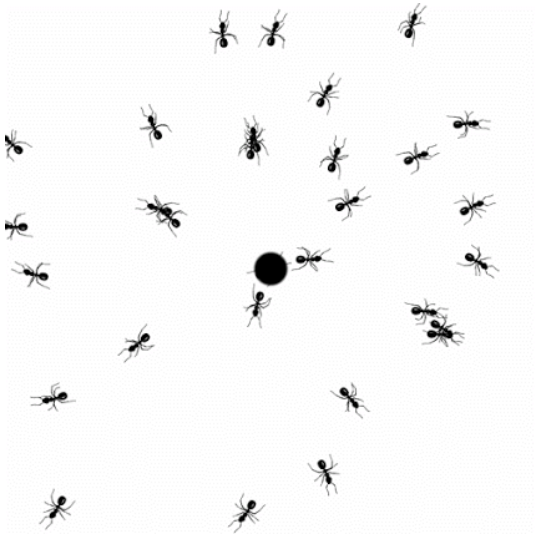


FORMIGAS DETECTAM O ALIMENTO



TRILHA DE FEROMÔNIO

FORMIGUEIRO

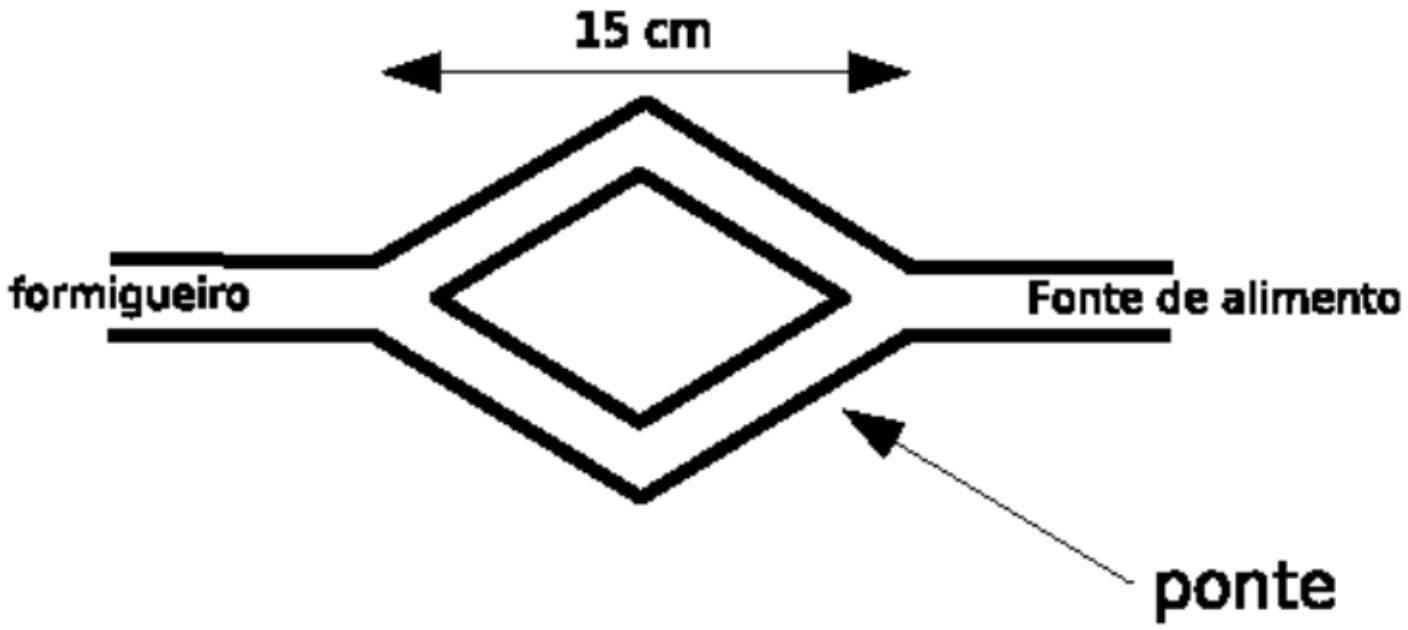


A Inspiração Biológica

- A o caminhar, as formigas depositam no chão o **feromônio**, formando, deste modo, uma **trilha de feromônios**.
- As formigas sentem o cheiro do feromônio, e quando elas têm que escolher um caminho, escolhem, com maior probabilidade, o caminho com **maior quantidade de feromônio (cheiro mais forte).**
- A trilha ajuda a formiga a achar o caminho de volta e as outras formigas a encontrar a fonte de alimentos.

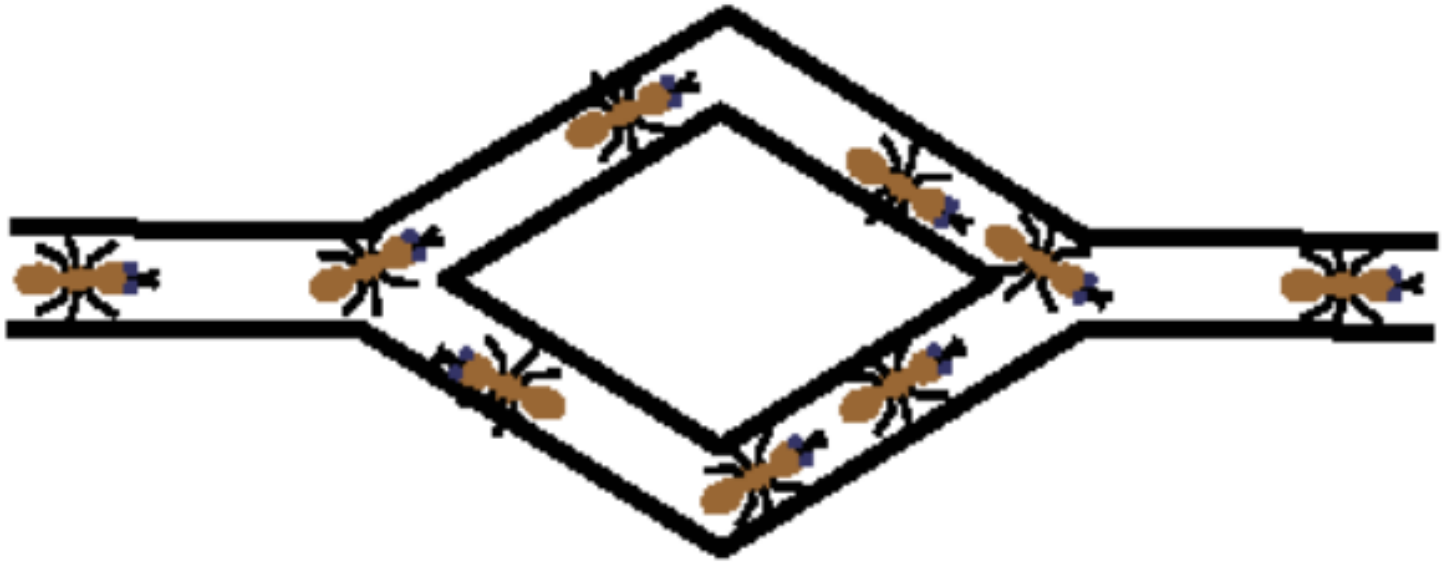
O Experimento da Ponte Binária

- Experimento realizado por Deneubourg et al., 1990, para estudar o comportamento forrageiro das formigas.



O Experimento da Ponte Binária

- No Início:
  - As formigas são deixadas livres para escolher o caminho.
  - Não há feromônio ainda

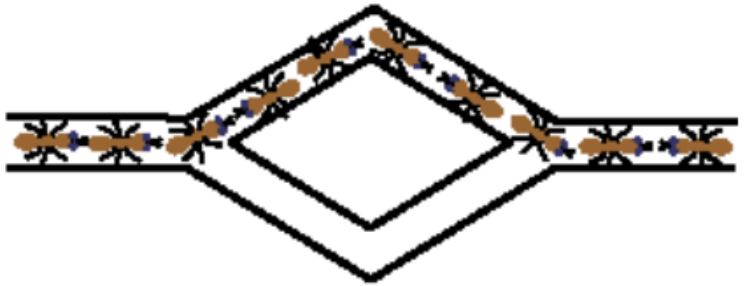


O Experimento da Ponte Binária

- As formigas convergem para um dos caminhos com igual probabilidade.

- Devido a flutuações aleatórias, uma das pontes terá mais feromônio e atrairá as formigas com maior probabilidade.

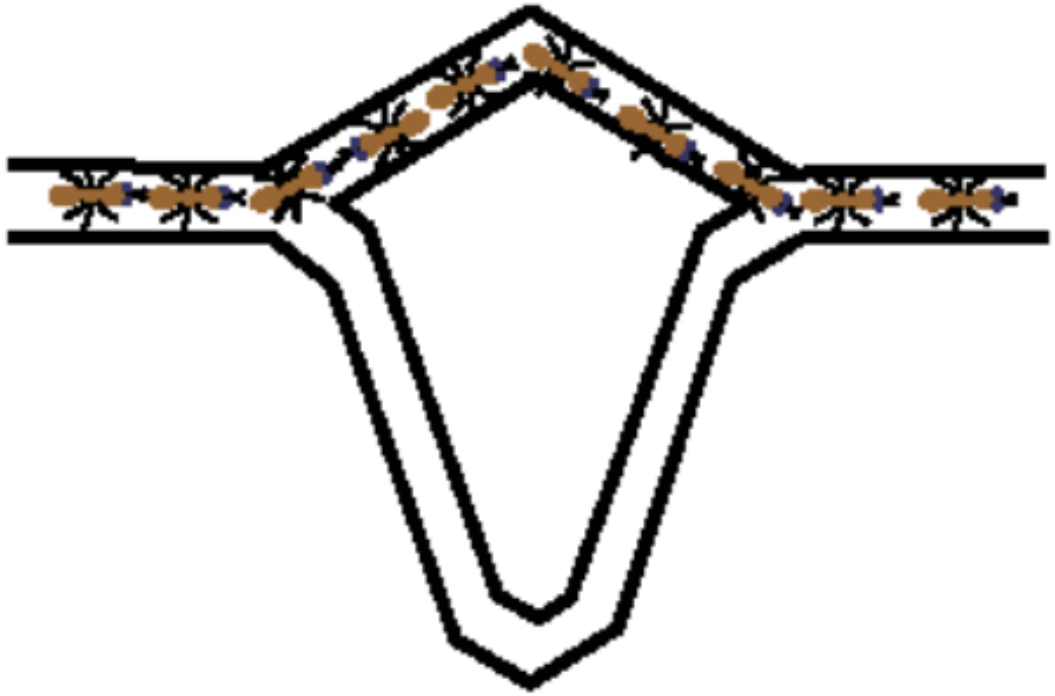
No Fim:



ou

O Experimento da Ponte Binária

- Usando pontes de tamanhos diferentes, as formigas convergem para a ponte mais curta:
  - A ponte curta é percorrida em menos tempo, fazendo com que mais formigas atravessem ela. Logo, mais feromônio é depositado.
  - As formigam escolhem, com maior probabilidade, a ponte curta (com mais feromônio)



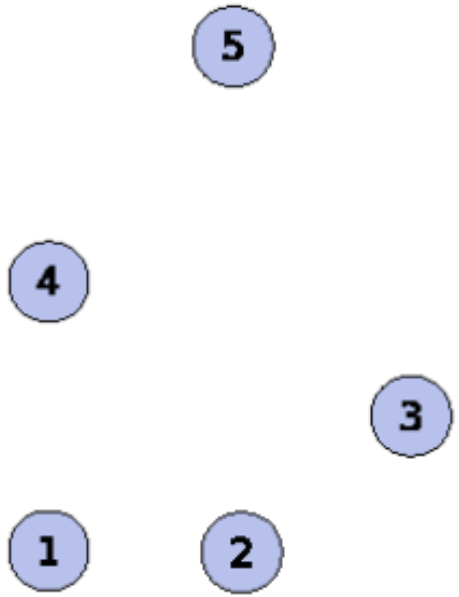
Ant System

- Proposto por **Marco Dorigo** e colaboradores (DORIGO et al., 1991)
- O **Ant System** é o primeiro algoritmo que surgiu inspirado em **colônia de formigas**.
- Peculiaridades do ambiente das formigas utilizadas:
  - Ao tomar um caminho a formiga deixa no mesmo uma certa quantidade de feromônio;
  - Uma formiga escolhe determinado caminho de acordo com uma função probabilística envolvendo a distância deste caminho e a quantidade de feromônio presente neste;
  - As formigas lembram os pontos por onde já passaram e não retornam a estes pontos até que tenham chegado à fonte de alimento

Aplicação do Ant System ao PCV

Matriz Distância do PCV

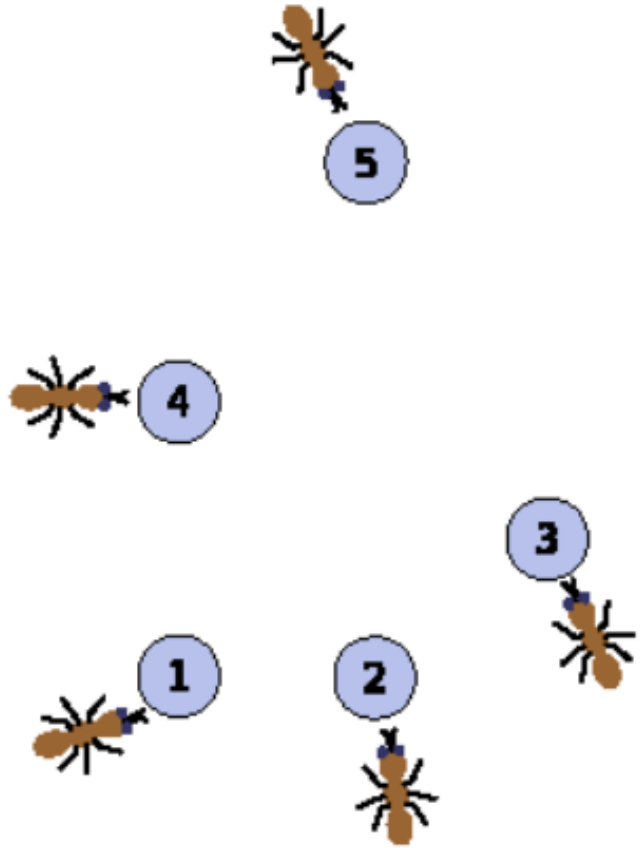
	1	2	3	4	5
1	0,0	1,0	2,2	2,0	4,1
2	1,0	0,0	1,4	2,2	4,0
3	2,2	1,4	0,0	2,2	3,2
4	2,0	2,2	2,2	0,0	2,2
5	4,1	4,0	3,2	2,2	0,0



cidades do PCV

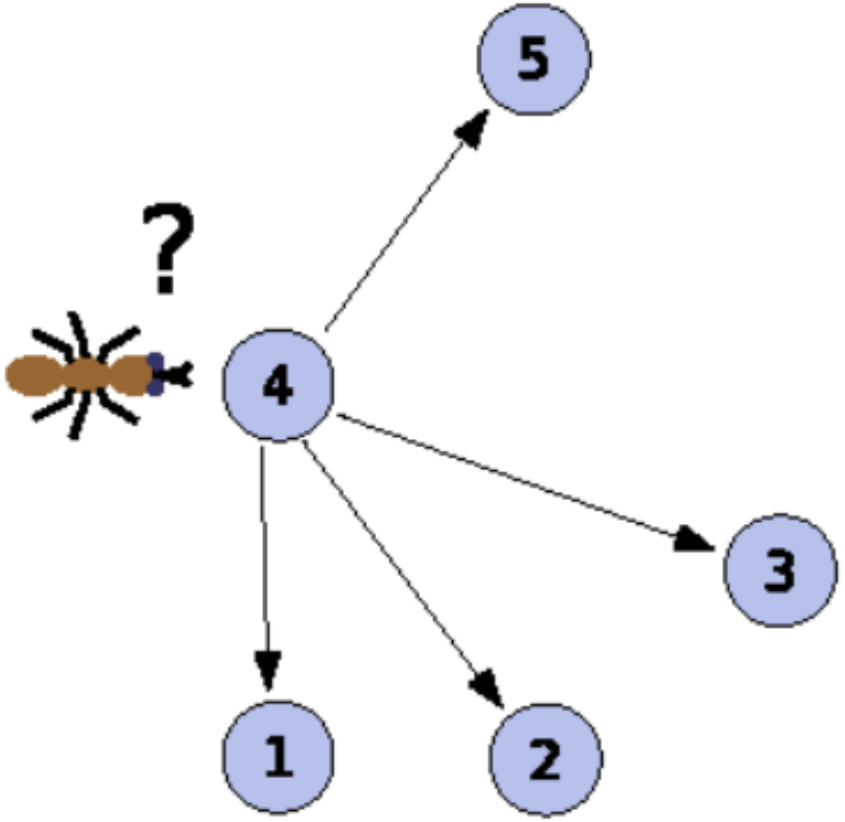
Aplicação do Ant System ao PCV

- Cada formiga irá construir uma solução movendo-se de uma cidade para outra.
- No ínicio, cada formiga é colocada em uma cidade diferente



Aplicação do Ant System ao PCV

- A Construção da Solução pela Formiga
- Começando de uma cidade i, a formiga move-se escolhendo probabilisticamente a cidade vizinha j (entre os vizinhos factíveis)



Probabilidade de Transição

- A probabilidade da formiga k que está na cidade i de escolher a cidade j é dada pela regra:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}, \text{se } j \in N_i^k \\ 0, \text{ caso contrário} \end{cases}$$

(t) **Numero de interação**

onde,

$\tau_{ij}(t)$: quantidade de feromônio presente no caminho (i,j)

$\eta_{ij} = \dfrac{1}{d_{ij}}$: visibilidade da cidade j com relação a cidade i

α e β são parâmetros para determinar a influência do feromônio e da informação heurística,

Feromônio e distância

N_i^k é a vizinhança factível da formiga k (i.e., o conjunto das cidades ainda não visitadas pela formiga k).

A Informação Heurística do PCV

- Associada a aresta (i, j) existe um valor heurístico η_{ij} dado por

$$\eta_{ij} = \frac{1}{d_{ij}}$$

que representa a atratividade da formiga visitar a cidade j depois de visitar a cidade i.

- O valor η_{ij} é inversamente proporcional a distância d_{ij} entre as cidades i e j.

Construção das rotas - Passo 1

formiga	Candidatos / prob. de transição	solução parcial
1	2(45%), 3(21%), 4(23%), 5(11%)	1-2
2	1(41%), 3(30%), 4(19%), 5(10%)	2-1
3	1(23%), 2(37%), 4(23%), 5(16%)	3-4
4	1(27%), 2(24%), 3(24%), 5(24%)	4-5
5	1(19%), 2(20%), 3(25%), 4(36%)	5-2

- A escolha do candidato é de acordo com a probabilidade de transição. É feita de forma similar ao algoritmo da roleta dos algoritmos genéticos.

Construção das rotas - Passo 2

formiga	Candidatos / prob. de transição	solução parcial
1	3(50%), 4(32%), 5(18%)	1-2-3
2	3(38%), 4(42%), 5(20%)	2-1-4
3	1(35%), 2(32%), 5(32%)	3-4-5
4	1(30%), 2(31%), 3(39%)	4-5-2
5	1(46%), 3(33%), 4(21%)	5-2-1

Construção das rotas - Passo 3

formiga	Candidatos / prob. de transição	solução parcial
1	4(59%), 5(41%)	1-2-3-5
2	3(50%), 5(50%)	2-1-4-5
3	1(49%), 2(51%)	3-4-5-1
4	1(58%), 3(42%)	4-5-2-1
5	3(48%), 4(52%)	5-2-1-4

Construção das rotas - Passo 4

formiga	Candidatos / prob. de transição	solução parcial
1	4(100%)	1-2-3-5-4
2	3(100%)	2-1-4-5-3
3	2(100%)	3-4-5-1-2
4	3(100%)	4-5-2-1-3
5	3(100%)	5-2-1-4-3

Término da Primeira Iteração

formiga (k)	solução completa	comprimento da viajem (L_k)
1	1-2-3-5-4-1	9,8
2	2-1-4-5-3-2	9,8
3	3-4-5-1-2-3	10,9
4	4-5-2-1-3-4	11,6
5	5-2-1-4-3-5	12,4

Atualização do Feromônio

- No feromônio τ_{ij} associado a aresta (i, j) ocorre dois eventos:

1. A evaporação;
   - Evita que o feromônio acumulado cresça indefinidamente;
   - Permite esquecer decisões ruins do passado da busca.

2. O depósito de feromônio de todas as formigas que passaram sobre (i, j).

Atualização do Feromônio

- Depois que todas as formigas construíram suas rotas, o feromônio é atualizado.

- $\Delta\tau_{ij}^{k}$ é a quantidade de feromônio que a formiga k deposita sobre a aresta (i, j). É dado por:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q / L_k, & \text{se a aresta } (i,j) \text{ pertence a rota } S_k \\ 0, & \text{caso contrário} \end{cases}$$

onde

Q : quantidade de ferômonio excretada por uma formiga a cada iteração

Atualização do Feromônio

- O feromônio τ_{ij} associado a aresta (i, j) é atualizado pelo fórmula:

$$\tau_{ij}(t+1) = \underbrace{(1-\rho)\tau_{ij}(t)}_{\text{evaporação}} + \underbrace{\sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)}_{\text{depósito}}$$

onde

$\rho \in [0,1]$ é a taxa de evaporação de feromônio

Exemplo de Atualização do Feromônio

Atualização do Feromônio da aresta (3,5)

- Apenas as formigas 1, 2 e 5 depositam feromônio nesta aresta. Suponha Q = 1, 0. A contribuição de cada formiga:

$$\Delta\tau_{3,5}^{(1)} = 1/L_1 = 0{,}102$$
$$\Delta\tau_{3,5}^{(2)} = 1/L_2 = 0{,}102$$
$$\Delta\tau_{3,5}^{(5)} = 1/L_5 = 0{,}081$$

k	viagem	L_k
1	1-2-3-5-4-1	9,8
2	2-1-4-5-3-2	9,8
3	3-4-5-1-2-3	10,9
4	4-5-2-1-3-4	11,6
5	5-2-1-4-3-5	12,4

Suponha $\rho = 0{,}5$

$$\begin{aligned}\tau_{3,5} &= (1-\rho)\tau_{ij} + \Delta\tau_{3,5}^{(1)} + \Delta\tau_{3,5}^{(2)} + \Delta\tau_{3,5}^{(5)} \\ &= (1-0{,}5)1{,}0 + 0{,}102 + 0{,}102 + 0{,}081 \\ &= 0{,}785\end{aligned}$$

Critérios de Parada

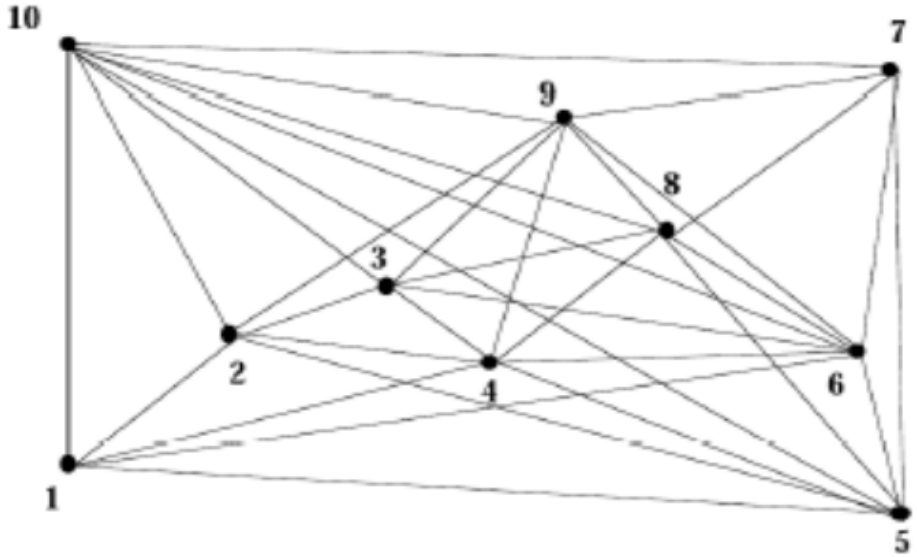
- Número máximo de iterações;
- Estagnação

Estagnação

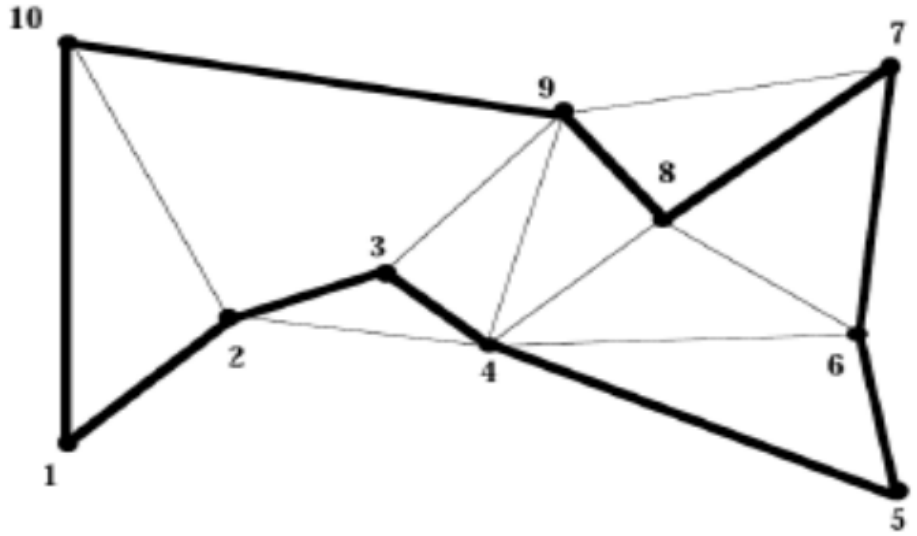
- Estagnação é a situação na qual todas as formigas seguem sempre o mesmo percurso.
- A Estagnação é causado pelo excessivo crescimento de feromônio nas arestas de uma rota sub-ótima.

Estagnação

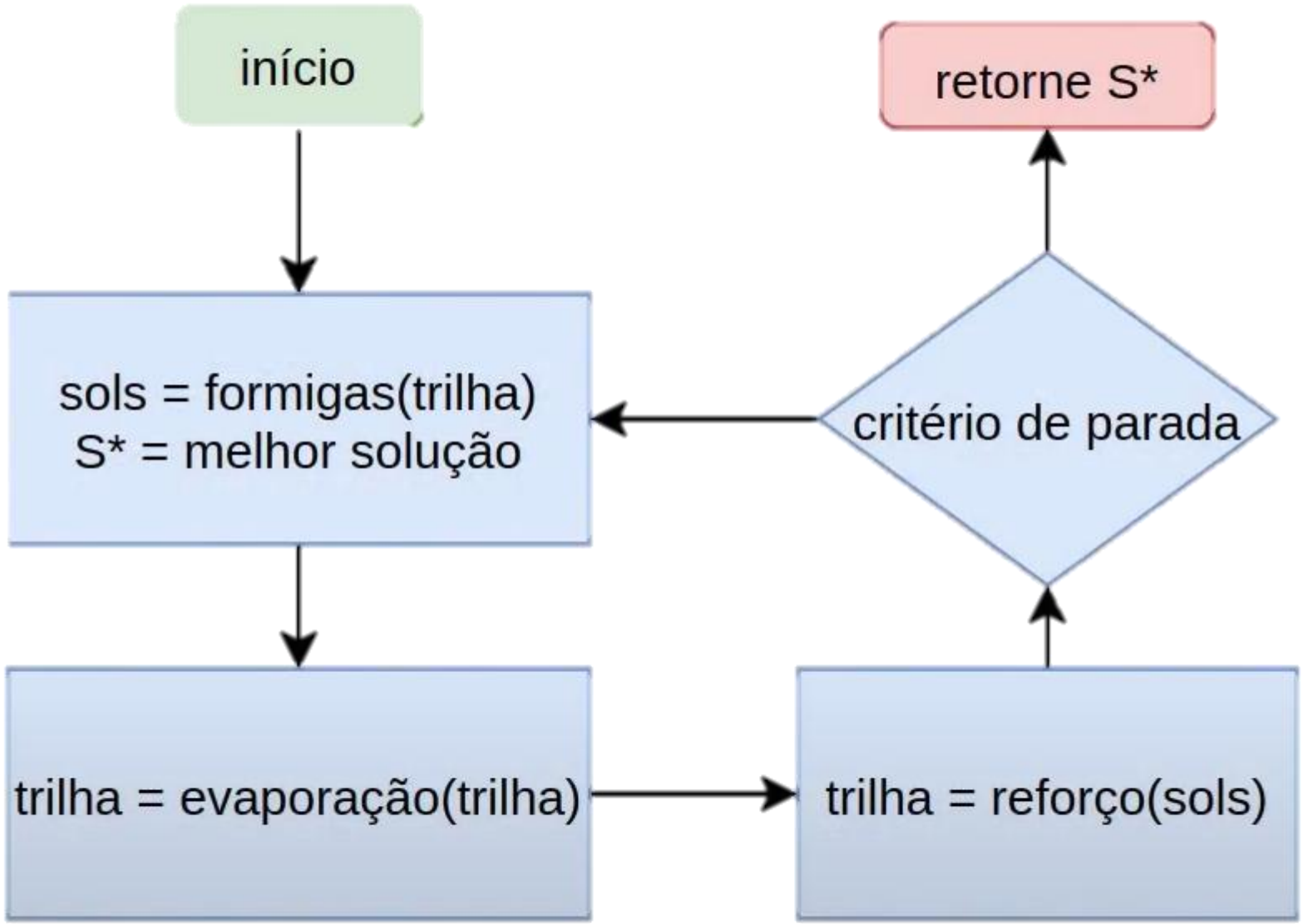
- Apesar da natureza estocástica do algoritmo, a forte concentração de feromônio nas arestas força a formiga a fazer sempre o mesmo percurso.



Distribuição de feromônio no inicio da busca.



Distribuição de feromônio após 100 iterações.


início
sols = formigas(trilha)
S* = melhor solução
trilha = evaporação(trilha)
trilha = reforço(sols)
critério de parada
retorne S*

Importação de Pacotes e Módulos

```
!pip install ACO-Pants  # Instalação da biblioteca ACO-Pants
import pants
import math
import random
import numpy
```

Explicação:

•**!pip install ACO-Pants:** Comando usado para instalar a biblioteca **ACO-Pants**, que implementa o algoritmo de Colônia de Formigas (Ant Colony Optimization).

•**import pants:** Importa o módulo principal da biblioteca **Pants**, que fornece as ferramentas para criar o ambiente do algoritmo (ex: cidades, caminhos e formigas).

•**import math:** Fornece funções matemáticas padrão, como raiz quadrada, logaritmo etc.

•**import random:** Usado para gerar números aleatórios, essenciais para simular o comportamento estocástico das formigas.

•**import numpy:** Biblioteca poderosa para trabalhar com **vetores, matrizes e operações numéricas eficientes**.

Geração do Grafo - Matriz de Distâncias

```python
def graphTSP(numCities, minDist, maxDist):
  cities = numpy.zeros((numCities, numCities), dtype=int)
  for i in range(numCities):
    for j in range(numCities):
      if j > i:
        cities[i, j] = random.randint(minDist, maxDist)
      elif j < i:
        cities[i, j] = cities[j, i]
  return cities
```

Explicação:

- Cria uma **matriz simétrica** onde cada posição [i][j] representa a **distância entre a cidade i e a cidade j**.
- As distâncias são sorteadas aleatoriamente entre minDist e maxDist.
- A simetria garante que a distância de ida seja igual à de volta.

Entrada do Usuário e Função de Distância

```
while True:
  numCities = int(input('Digite o número de cidades: '))
  if numCities > 4:
    break
  else:
    print('O número de cidades deve ser maior que 4!')

cities = graphTSP(numCities, 10, 100)

def dist(cid1, cid2):
  return cities[cid1][cid2]
```

Explicação:

•O usuário informa quantas cidades deseja simular (mínimo de 5 para tornar o problema mais interessante).

•A função dist(cid1, cid2) retorna a **distância entre duas cidades**, usando a matriz gerada.

•Essa função será usada pelas formigas para decidir os próximos passos com base nas distâncias.

Representação do Mundo e Execução do Algoritmo

```
nodes = list(range(numCities))
world = pants.World(nodes, dist)
solver = pants.Solver()
solution = solver.solve(world)

print('Caminho:', solution.tour)
print('Custo Total:', solution.distance)
```

Explicação:

•nodes = list(range(numCities))

Cria uma lista de cidades representadas por números (ex: [0, 1, 2, 3, 4]).

•world = pants.World(nodes, dist)

Cria o "mundo" onde as formigas vão atuar, ou seja, define o problema com os **nós (cidades)** e a **função de distância** entre eles.

•solver = pants.Solver()

Inicializa o solucionador usando a implementação da Colônia de Formigas da biblioteca **Pants**.

•solution = solver.solve(world)

Executa o algoritmo de otimização para encontrar o **melhor caminho** passando por todas as cidades uma única vez (problema do caixeiro viajante).

•solution.tour

Mostra a sequência de cidades visitadas (caminho encontrado).

•solution.distance

Mostra o **custo total (soma das distâncias)** do caminho.

https://colab.research.google.com/drive/1L4s1n1F4mUTKZfpWVUTRnc88wYB7v4RO?usp=sharing