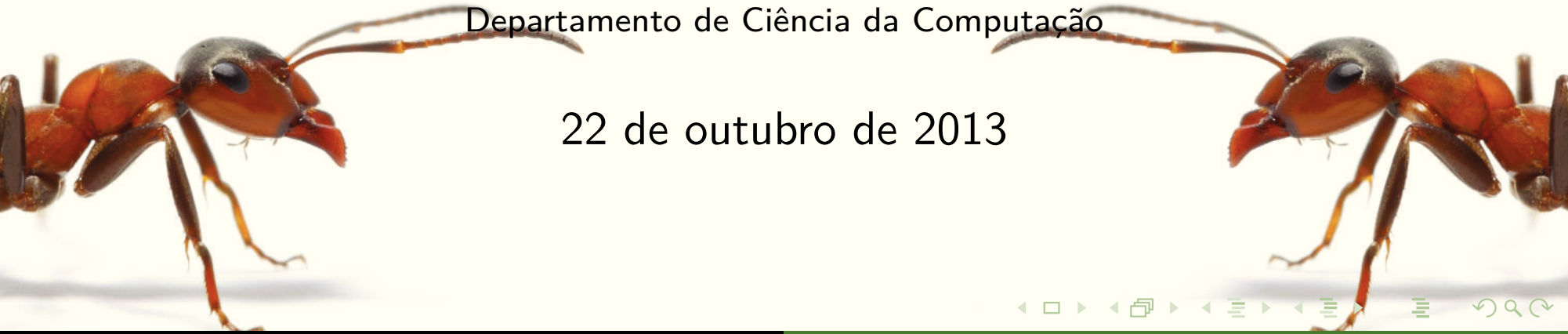
# Otimização por Colônia de Formigas

## Aplicação do Algoritmo Ant System na Solução do Problema do Caixeiro-Viajante Simétrico

Douglas Ferreira de Borba

Universidade Estadual do Centro-Oeste (Unicentro)
Departamento de Ciência da Computação

22 de outubro de 2013



# Sumário



# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\, a, b \in V,\ dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.

# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\, a, b \in V,\ dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.

# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\, a, b \in V,\ dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.

# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\ a, b \in V, dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.

# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\ a, b \in V, dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.

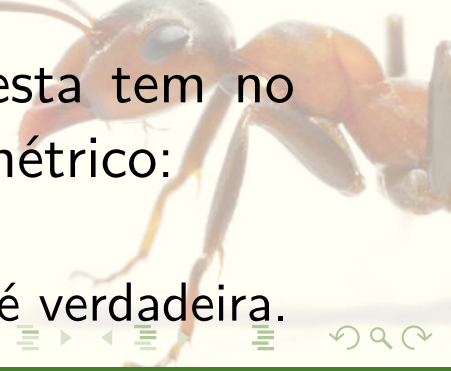
# Introdução

## O Problema do Caixeiro-Viajante

### Descrição

O problema do caixeiro-viajante é um problema combinatorial clássico. Onde dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, busca-se determinar o circuito de menor comprimento que passa por todas as cidades, exatamente uma vez, e termina retornando à cidade de partida [1].

- É um problema NP-Completo de complexidade $O(n!)$;
- Pode ser representado por meio de grafo $G(V, E)$;
- Dependendo da importância que a direção da aresta tem no problema, distingue-se o PCV em simétrico e assimétrico:
  - PCV simétrico: $\forall\ a, b \in V, dist(a, b) = dist(b, a)$;
  - PCV assimétrico: a afirmação acima nem sempre é verdadeira.



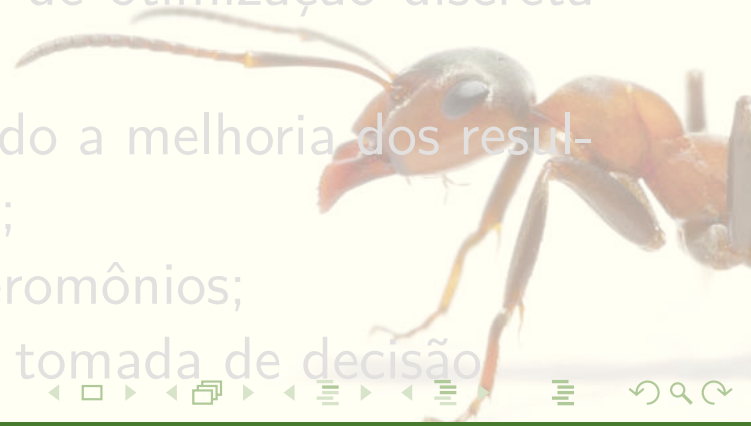
# Introdução

## O Algoritmo Ant System

### Descrição

Proposto por Dorigo e outros [2], o *Ant System* (AS) foi o primeiro método de Otimização por Colônias de Formigas apresentado na literatura. Sendo amplamente indicado para "solução" de problemas combinatoriais, tais como: o PCV e o Problema de Coloração de Grafos [3].

- Foi implementado como um algoritmo de otimização discreta para resolver o PCV;
- Passou por diversas modificações visando a melhoria dos resultados obtidos. Ex. *Elitism Ant System*;
- Toda a colônia realiza o deposito de feromônios;
- Faz uso de informações históricas para tomada de decisão.



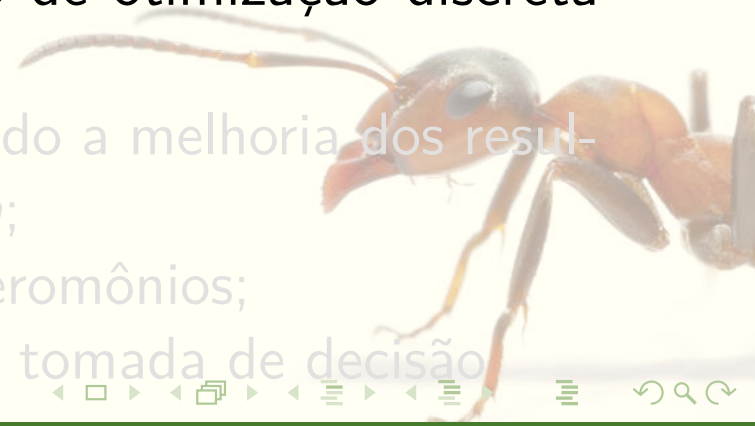
# Introdução

## O Algoritmo Ant System

### Descrição

Proposto por Dorigo e outros [2], o *Ant System* (AS) foi o primeiro método de Otimização por Colônias de Formigas apresentado na literatura. Sendo amplamente indicado para "solução" de problemas combinatoriais, tais como: o PCV e o Problema de Coloração de Grafos [3].

- Foi implementado como um algoritmo de otimização discreta para resolver o PCV;
- Passou por diversas modificações visando a melhoria dos resultados obtidos. Ex. *Elitism Ant System*;
- Toda a colônia realiza o deposito de feromônios;
- Faz uso de informações históricas para tomada de decisão.



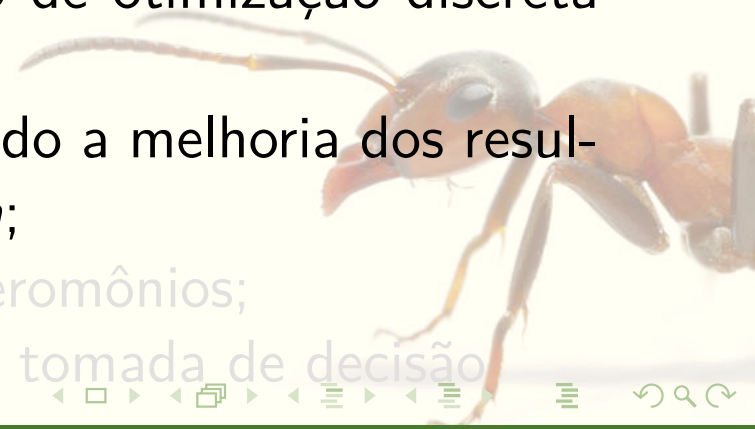
# Introdução

## O Algoritmo Ant System

### Descrição

Proposto por Dorigo e outros [2], o *Ant System* (AS) foi o primeiro método de Otimização por Colônias de Formigas apresentado na literatura. Sendo amplamente indicado para "solução" de problemas combinatoriais, tais como: o PCV e o Problema de Coloração de Grafos [3].

- Foi implementado como um algoritmo de otimização discreta para resolver o PCV;
- Passou por diversas modificações visando a melhoria dos resultados obtidos. Ex. *Elitism Ant System*;
- Toda a colônia realiza o deposito de feromônios;
- Faz uso de informações históricas para tomada de decisão.



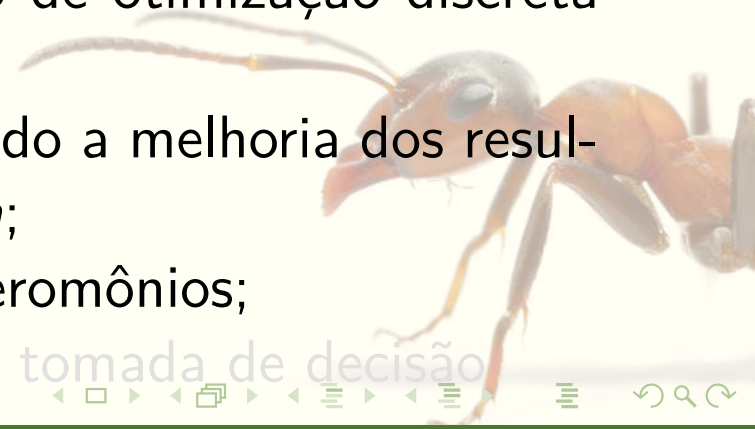
# Introdução

## O Algoritmo Ant System

### Descrição

Proposto por Dorigo e outros [2], o *Ant System* (AS) foi o primeiro método de Otimização por Colônias de Formigas apresentado na literatura. Sendo amplamente indicado para "solução" de problemas combinatoriais, tais como: o PCV e o Problema de Coloração de Grafos [3].

- Foi implementado como um algoritmo de otimização discreta para resolver o PCV;
- Passou por diversas modificações visando a melhoria dos resultados obtidos. Ex. *Elitism Ant System*;
- Toda a colônia realiza o deposito de feromônios;
- Faz uso de informações históricas para tomada de decisão.



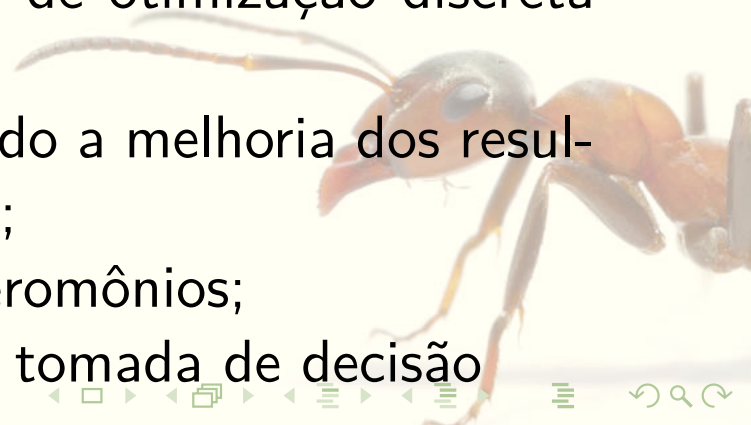
# Introdução

## O Algoritmo Ant System

### Descrição

Proposto por Dorigo e outros [2], o *Ant System* (AS) foi o primeiro método de Otimização por Colônias de Formigas apresentado na literatura. Sendo amplamente indicado para "solução" de problemas combinatoriais, tais como: o PCV e o Problema de Coloração de Grafos [3].

- Foi implementado como um algoritmo de otimização discreta para resolver o PCV;
- Passou por diversas modificações visando a melhoria dos resultados obtidos. Ex. *Elitism Ant System*;
- Toda a colônia realiza o deposito de feromônios;
- Faz uso de informações históricas para tomada de decisão

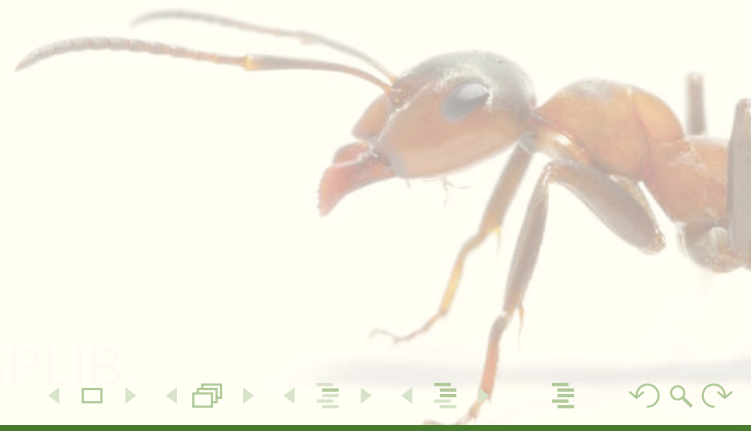


# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5;
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0;
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.



# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2;
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## Recursos Utilizados

Para o desenvolvimento deste trabalho foi feito o uso dos seguintes recursos:

- Ambiente de Execução:
  - Processador Intel i7 (2ª Geração);
  - Memória RAM 6 GB;
  - Sistema Operacional Windows 7 Home Premium (x64);
  - Interpretador Python v.2.7.5.
- Ferramentas Utilizadas:
  - Editor Sublime Text 2 v.2.0.2;
  - Eclipse IDE Kepler;
    - Plugin Pydev v.2.8.2.
  - Biblioteca matplotlib v.1.2.0.
- Instâncias de Teste:
  - Instâncias retiradas do repositório TSPLIB.

# Material e Métodos

## A Implementação (Pseudo-código)

O algoritmo implementado baseou-se nos pseudo-códigos apresentados em [3] e [4], o que resultou no seguinte pseudo-código:

**Algorithm 6.3.1**: Pseudocode for Ant System.

```
Input: ProblemSize, Population_size, m, ρ, α, β
Output: P_best
1 P_best ← CreateHeuristicSolution(ProblemSize);
2 Pbest_cost ← Cost(S_h);
3 Pheromone ← InitializePheromone(Pbest_cost);
4 while ¬StopCondition() do
5 |   Candidates ← ∅;
6 |   for i = 1 to m do
7 |   |   S_i ← ProbabilisticStepwiseConstruction(Pheromone,
  |   |   ProblemSize, α, β);
```

# Material e Métodos

## A Implementação (Pseudo-código)

```
8        Si_cost ← Cost(S_i);
9        if Si_cost ≤ Pbest_cost then
10           Pbest_cost ← Si_cost;
11           P_best ← S_i;
12       end
13       Candidates ← S_i;
14   end
15   DecayPheromone(Pheromone, ρ);
16   foreach S_i ∈ Candidates do
17       UpdatePheromone(Pheromone, S_i, Si_cost);
18   end
19 end
20 return P_best;
```

# Resultados e Discussão

## Confronto dos Resultados com a Literatura

Ajustando os parâmetros do algoritmo para: $\alpha = 5.0$, $\beta = 5.0$, $\rho = 0.6$, $n_{ants} = 200$ e $n_{colonies} = 500$ em uma série de 10 execuções, obteve-se os seguintes resultados "ótimos":

| **Instância** | **Cidades**$_{Total}$ | **Solução**$_{Best}$ | **Tempo**$_{AS}$ | **Solução**$_{AS}$ |
|---|---|---|---|---|
| oliver30 | 30 | 423.74 | 333.6s | 429.358 |
| eil51 | 51 | 426.00 | 855.6s | 461.811 |
| berlin52 | 52 | 7542.00 | 920.1s | 7823.041 |
| st70 | 70 | 675.00 | 1849.8s | 736.272 |

Tabela : Resultados da Execução do *Ant System* sem arredondamento.

# Resultados e Discussão

## Alterando o Algoritmo

Para melhorar o algoritmo foi adotado o elitismo, alterando o *Ant System* (AS) para *Elitism Ant System* (EAS), obtendo-se os seguintes resultados:

| **Instância** | **Tempo**$_{AS}$ | **Solução**$_{AS}$ | | **Tempo**$_{EAS}$ | **Solução**$_{EAS}$ |
|---|---|---|---|---|---|
| oliver30 | 333.6s | 429.358 | | 264.8s | 423.740 |
| eil51 | 855.6s | 461.811 | | 917.2s | 429.530 |
| berlin52 | 920.1s | 7823.041 | | 973.0s | 7544.365 |
| st70 | 1849.8s | 736.272 | | 1930.0s | 693.542 |

Tabela : *Ant System* vs. *Elitism Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: oliver30

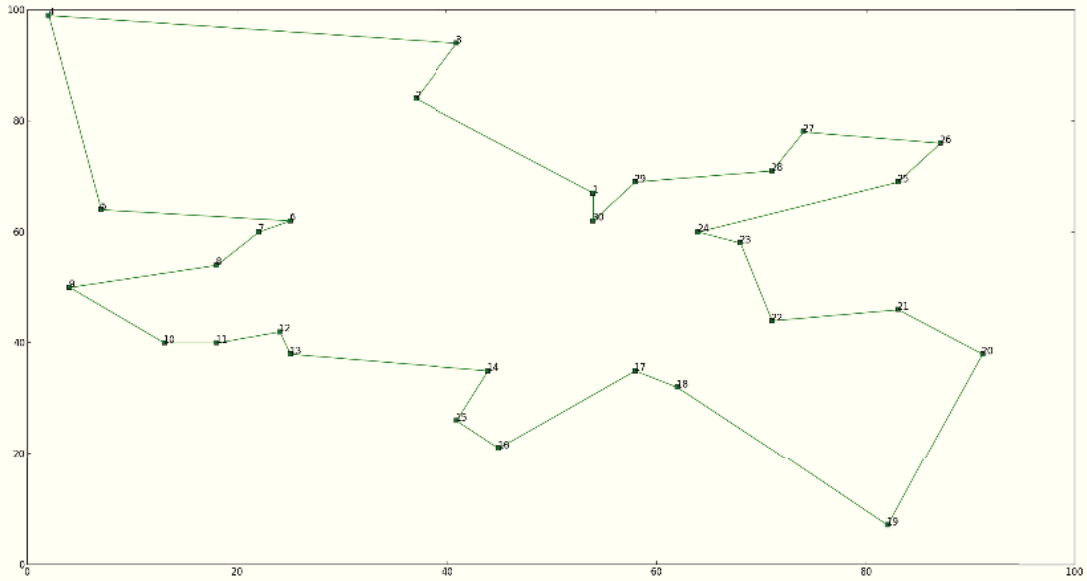



Figura : Melhor solução para instância oliver30.

# Resultados e Discussão

## Visualizando os Resultados: oliver30

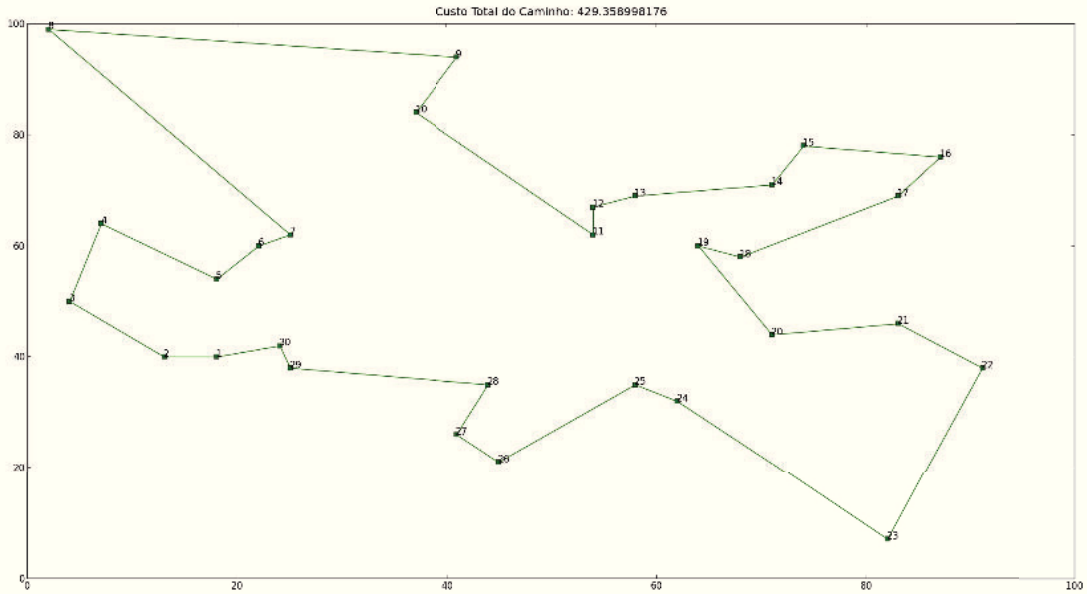



Figura : Solução da instância oliver30 com *Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: oliver30

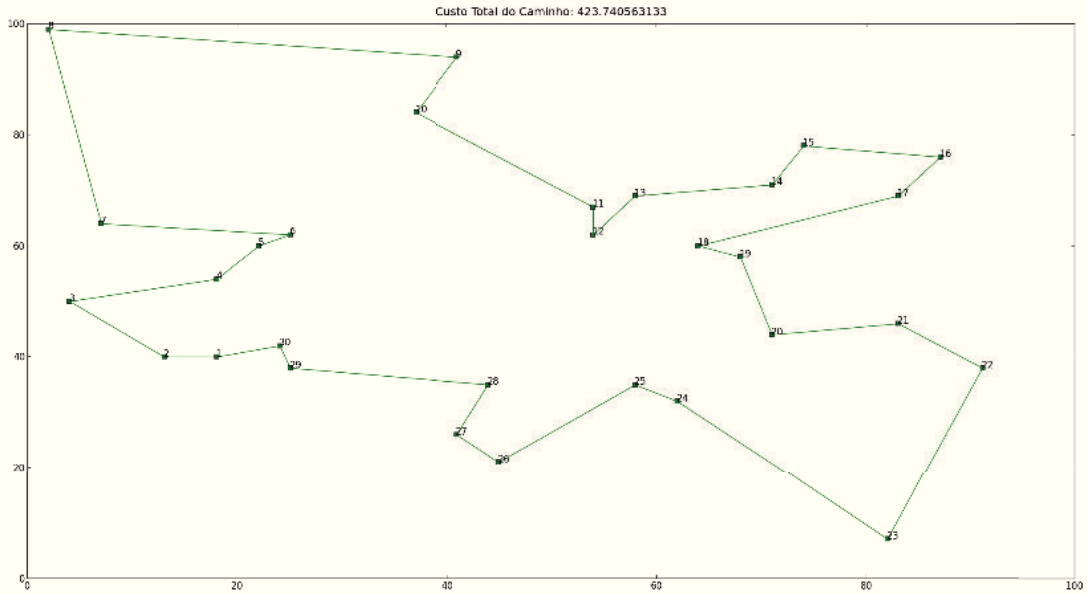



Figura : Solução da instância oliver30 com *Elistism Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: eil51

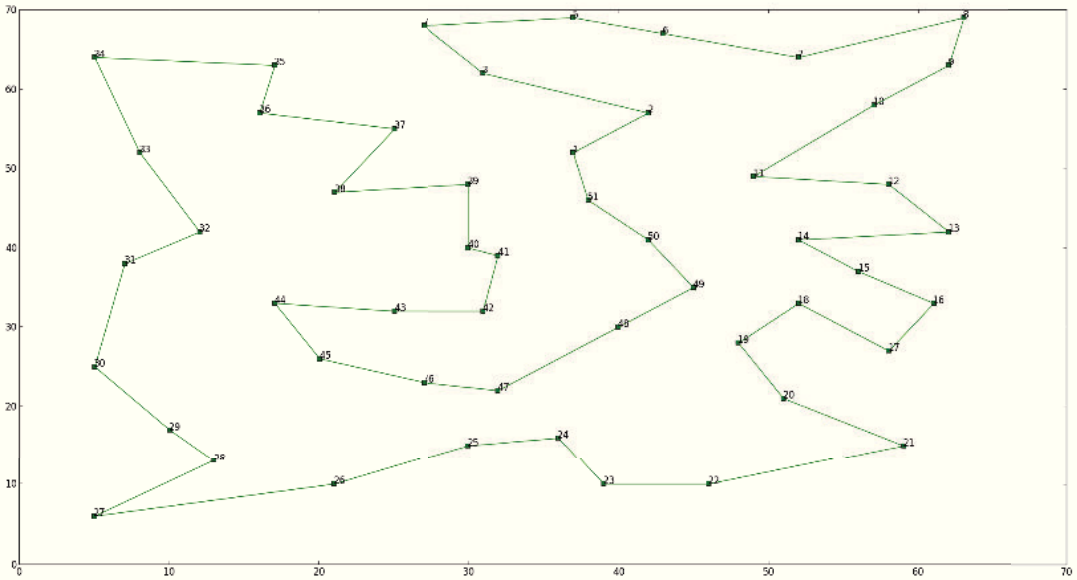


Figura : Melhor solução para instância eil51.

# Resultados e Discussão

## Visualizando os Resultados: eil51

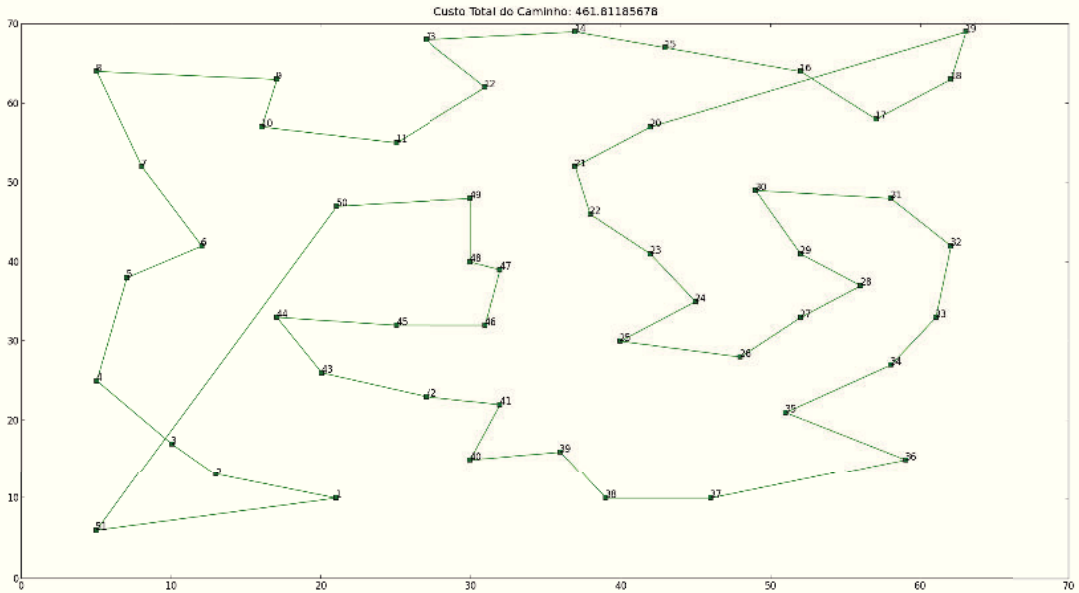



Figura : Solução da instância eil51 com *Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: eil51

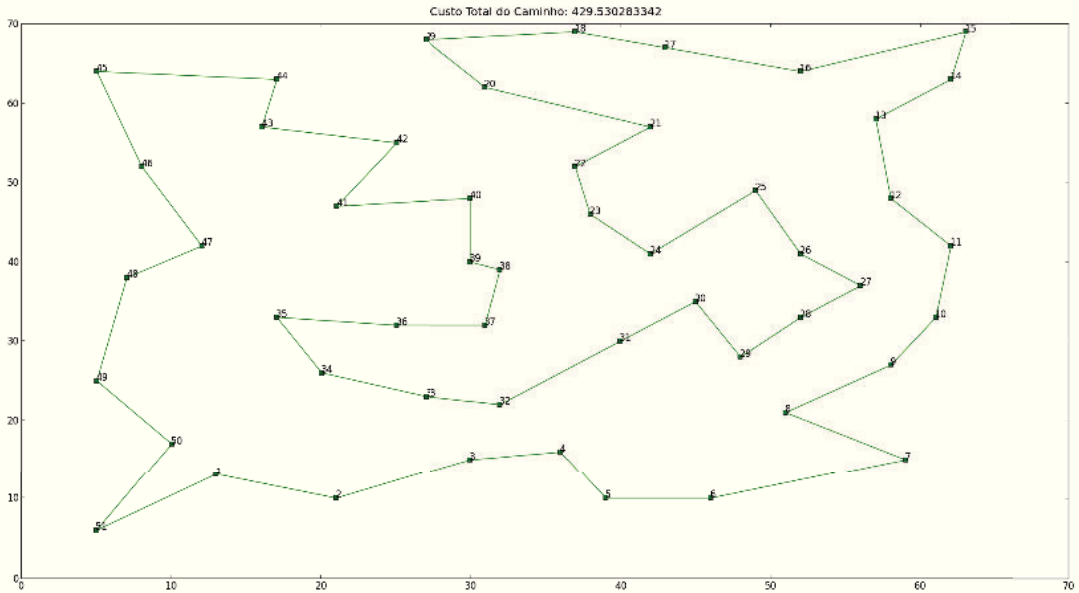



Figura : Solução da instância eil51 com *Elistism Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: berlin52

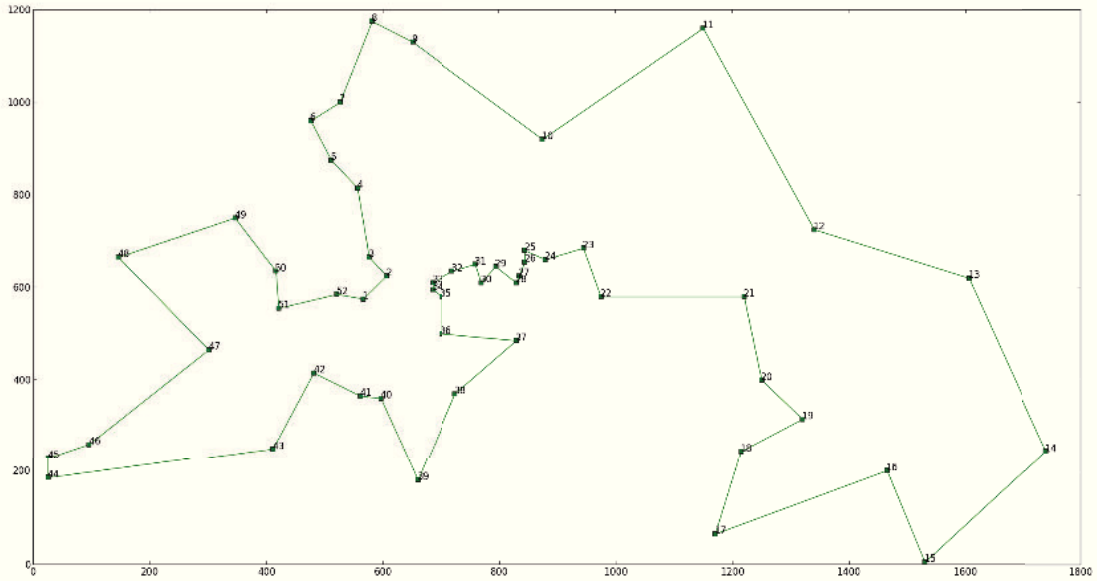



Figura : Melhor solução para instância berlin52.

# Resultados e Discussão

## Visualizando os Resultados: berlin52

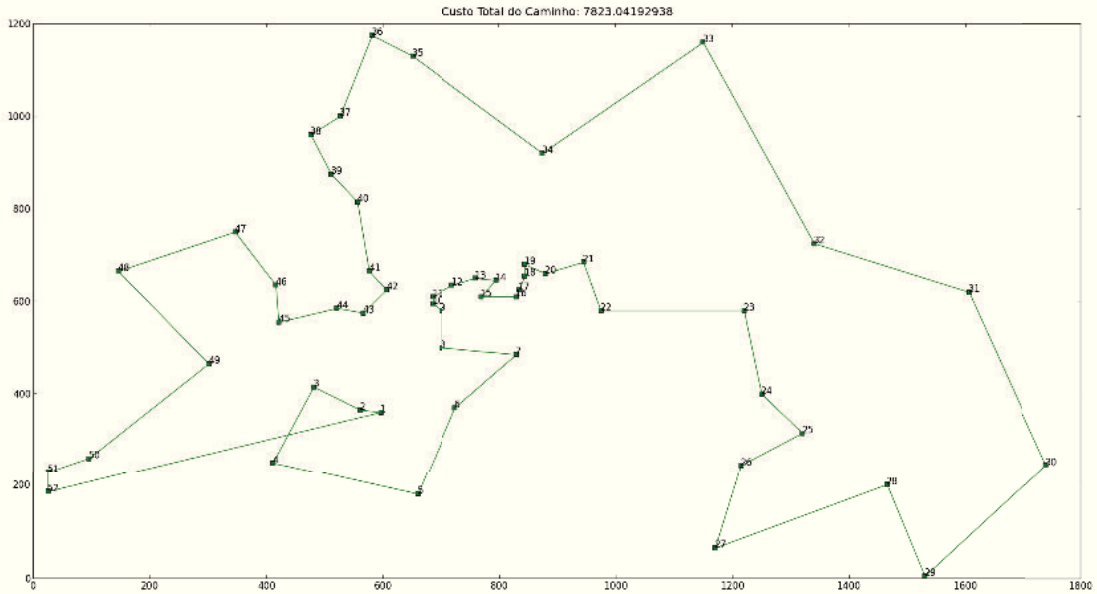



Figura : Solução da instância berlin52 com *Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: berlin52

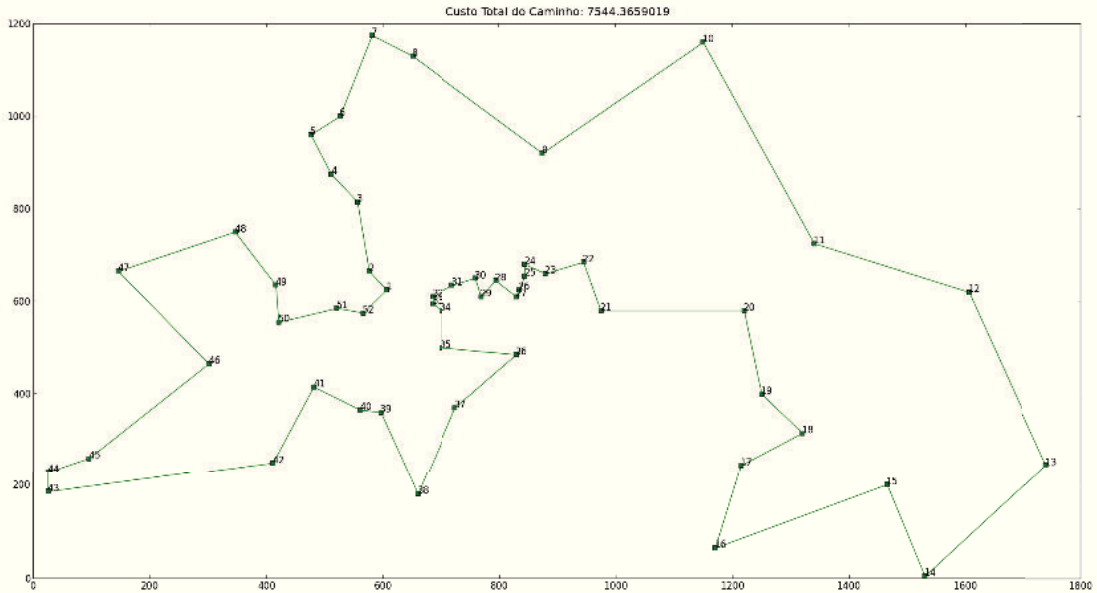



Figura : Solução da instância berlin52 com *Elistism Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: st70

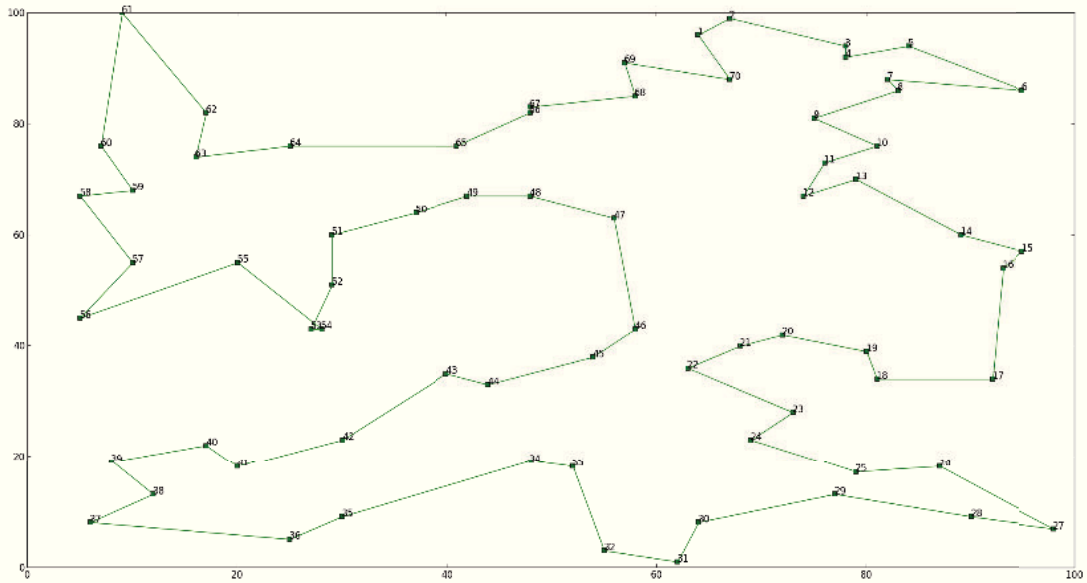


Figura : Melhor solução para instância st70.

# Resultados e Discussão

## Visualizando os Resultados: st70

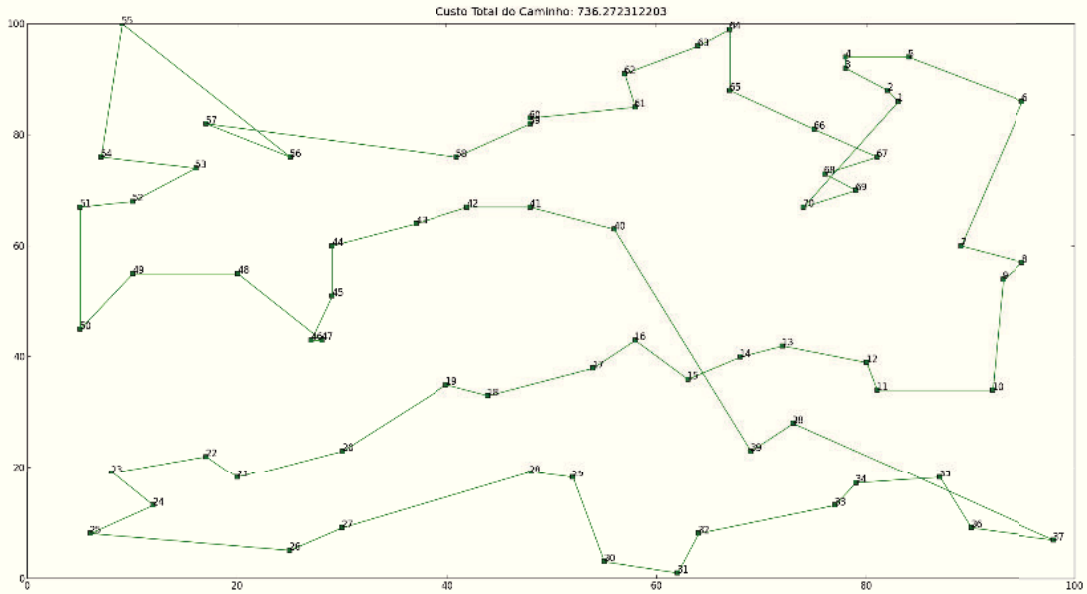



Figura : Solução da instância st70 com *Ant System*.

# Resultados e Discussão

## Visualizando os Resultados: st70

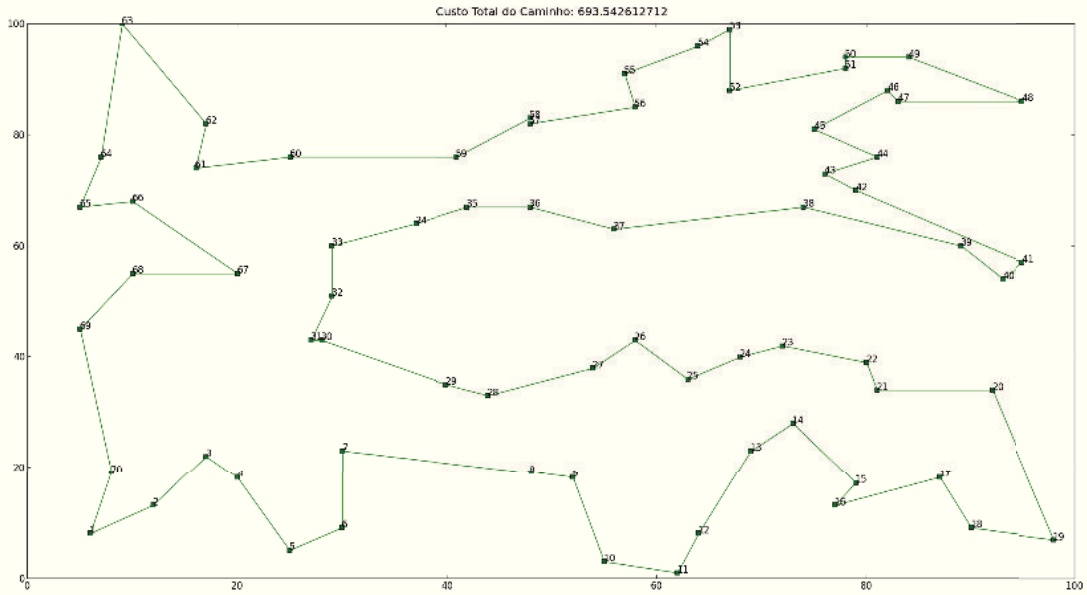



Figura : Solução da instância st70 com *Elistism Ant System*.

# Conclusões

## Observações Relevantes

- Como afirmado em [2], métodos de otimização por colônia de formigas são uma boa opção para solução de problemas combinatoriais, tais como o PCV;
- O AS fornece bons resultados para grafos pequenos (30 cidades ou menos), mas para grafos maiores os resultados tendem a piorar, e exigem uma boa combinação entre os parâmetros ou variações de sua implementação para se obter uma boa solução;
- Para problemas grandes, uma grande quantidade de memória é utilizada, comprometendo seu desempenho;
- O EAS é uma boa opção para problemas grandes, pois apresenta uma rápida convergência.

# Conclusões

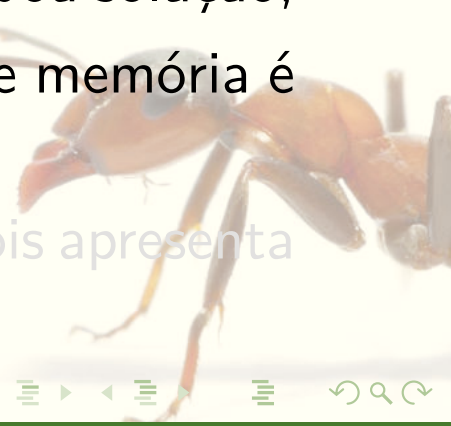
## Observações Relevantes

- Como afirmado em [2], métodos de otimização por colônia de formigas são uma boa opção para solução de problemas combinatoriais, tais como o PCV;
- O AS fornece bons resultados para grafos pequenos (30 cidades ou menos), mas para grafos maiores os resultados tendem a piorar, e exigem uma boa combinação entre os parâmetros ou variações de sua implementação para se obter uma boa solução;
- Para problemas grandes, uma grande quantidade de memória é utilizada, comprometendo seu desempenho;
- O EAS é uma boa opção para problemas grandes, pois apresenta uma rápida convergência.

# Conclusões

## Observações Relevantes

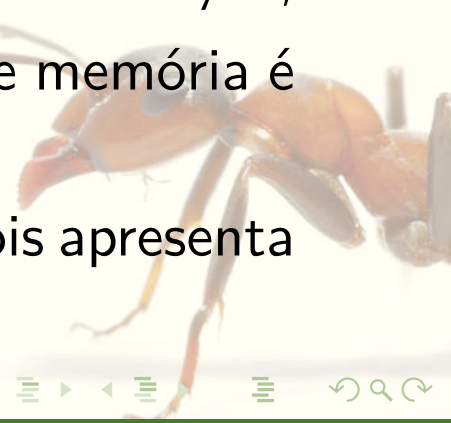
- Como afirmado em [2], métodos de otimização por colônia de formigas são uma boa opção para solução de problemas combinatoriais, tais como o PCV;
- O AS fornece bons resultados para grafos pequenos (30 cidades ou menos), mas para grafos maiores os resultados tendem a piorar, e exigem uma boa combinação entre os parâmetros ou variações de sua implementação para se obter uma boa solução;
- Para problemas grandes, uma grande quantidade de memória é utilizada, comprometendo seu desempenho;
- O EAS é uma boa opção para problemas grandes, pois apresenta uma rápida convergência.



# Conclusões

## Observações Relevantes

- Como afirmado em [2], métodos de otimização por colônia de formigas são uma boa opção para solução de problemas combinatoriais, tais como o PCV;
- O AS fornece bons resultados para grafos pequenos (30 cidades ou menos), mas para grafos maiores os resultados tendem a piorar, e exigem uma boa combinação entre os parâmetros ou variações de sua implementação para se obter uma boa solução;
- Para problemas grandes, uma grande quantidade de memória é utilizada, comprometendo seu desempenho;
- O EAS é uma boa opção para problemas grandes, pois apresenta uma rápida convergência.



# Referências

[1] A. M. Monteiro and J. L. Soares, "Resolução do problema do caixeiro viajante assimétrico (e uma variante) através da relaxação lagrangeana," 2006.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," tech. rep., Technical Report No. 91-016, Politecnico di Milano, Italy, 1991.

[3] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*.
Lulu Enterprises Incorporated, 2011.

[4] M. Dorigo and T. Stützle, *Ant Colony Optimization*.
A Bradford Book, Bradford Book, 2004.