Computação Natural: Trabalho Prático 2

Otimização por Colônia de Formigas

Jônatas Renan
{*jonatas@dcc.ufmg.br*}

30 de Novembro de 2015

1 Introdução

O algoritmo da otimização da colônia de formigas (ACO, do inglês ant colony optimization algorithm), introduzido por Marco Dorigo em sua tese de PhD é uma heurística baseada em probabilidade, criada para solução de problemas computacionais que envolvem procura de caminhos em grafos. Este algoritmo foi inspirado na observação do comportamento das formigas ao saírem de sua colônia para encontrar comida.

Na natureza, as formigas andam inicialmente sem rumo até que, encontrada comida, elas retornam à colônia deixando um rastro de feromônio. Se outras formigas encontram um desses rastros, elas tendem a não seguir mais caminhos aleatórios. Em vez disso, seguem a trilha encontrada, retornando e inclusive liberando mais feromônio se acharem mais alimento.

Com o transcorrer do tempo, entretanto, as trilhas de feromônio começam a evaporar, reduzindo, assim, sua força atrativa. Quanto mais formigas passarem por um caminho predeterminado, mais tempo será necessário para que o feromônio associado àquela trilha evapore. Analogamente, elas marcharão mais rapidamente por sobre um caminho curto, o que implica aumento da densidade de feromônio depositado antes que ele comece a evaporar. A evaporação do feromônio também possui a vantagem de evitar a convergência para uma solução local ótima: se a evaporação não procedesse, todas as trilhas escolhidas pelas primeiras formigas se tornariam excessivamente atrativas para as outras e, neste caso, a exploração do espaço da solução se delimitaria consideravelmente.

Todavia, quando uma formiga encontra um bom caminho entre a colônia e a fonte de alimento, outras formigas tenderão a seguir este caminho, gerando assim feedback positivo, o que eventualmente torna um determinado caminho mais interessante. A idéia do algoritmo da colônia de formigas é imitar este comportamento através de "formigas virtuais"que caminham por um grafo que

por sua vez representa o problema a ser resolvido.

Apesar de ter sido inicialmente abstraído para produzir soluções quase ótimas para o problema do caixeiro viajante, neste trabalho aplicou-se o ACO como uma metaheurística para o problema do Maior Caminho *Longest Path* ou *LP*.O problema do maior caminho é NP Completo, enquanto a metaheurística que usa o ACO opera em tempo polinomial.

2 Implementação

2.1 Ambiente de Desenvolvimento

O trabalho foi desenvolvido em ambiente de desenvolvimento *Ruby 2.0.0p645 - 2015-04-13 revision 50299* com a inclusão da biblioteca Parallel de Michael Grosser (Obtida através do comando *gem install parallel*).

2.2 Parâmetros de Entrada

Para que possam ser feitas todas as análises de sensibilidade, foram expostos todos os parâmetros do problema. Segue a lista abaixo com uma breve explicação:

1. −−min_runs value : valor mínimo de runs
2. −−max_runs value : valor máximo de runs
3. −−step_runs value : tamanho do passo de runs
4. −−min_ants value : valor mínimo de ants
5. −−max_ants value : valor máximo de ants
6. −−step_ants value : tamanho do passo de ants
7. −−iterations value : número de iterações
8. −−initial_pheromone_amount value : valor de feromônio inicial
9. −−ant_pheromone_deposit_amount value : valor do padrão do depósito de feromônio quando a formiga caminha
10. −−pheromone_evaporation_rate value : taxa de evaporação do feromônio
11. −−daemon_pheromone_deposit_amount value : valor de depósito da melhor formiga
12. −−max_pheromone_threshold value : valor máximo de feromônio na aresta
13. −−min_pheromone_threshold value : valor mínimo de feromônio na aresta
14. −−alpha value : Parâmetro da função de seleção de aresta pela formiga relacionado ao feromônio

15. −−beta value : Parâmetro da função de seleção de aresta pela formiga relacionado ao peso da aresta
16. −−exploitation_factor value
17. −−input_file_path file
18. −−optimum_cost value
19. −−output_header
20. −−output_file value
21. −−description description

2.3 Representação do Problema: Longest Path

A entrada do problema consiste em um grafo ponderado direcionado, assim como os nós de início e fim. Deseja-se encontrar o maior caminho entre os dois nós. Para isto apresentaremos parâmetros de entrada que serão analisados para que a saída consista em um caminho válido que parta do nó início e caminhe até o nó fim e seu devido comprimento.

Para a representação do grafo foi utilizado uma matriz de adjacências implementada como um **Hash de Arrays** configurados como [**Fonte** [**Destino, Peso da Aresta**].

Abaixo, pseudo código do Problema:

2.3.1 Pseudo Código: Longest Path

```
Obtém lista de parâmetros de entrada
for iterações do
  for formigas do
    for execuções map paralelo do
      inicializa formiga no nó inicial
      executa formiga
    end for
    Obtém resultados e guarda melhor resultado global: reduce
  end for
end for
retorna melhor resultado
```

2.4 Representação do ACO

O ACO em LP opera em conformidade com o método já aplicado no caixeiro viajante. A maior diferença reside na função de avaliação de uma solução. No ACO, a solução avaliada é o caminho percorrido pela formiga. Para o problema do Maior Caminho, a formiga que percorre o maior caminho liberará mais feromônio que as outras, para atrair as demais formigas para convergência daquela solução.

2.4.1 Pseudo Código: ACO

Dessa forma, o ACO segue o seguinte pseudo-algoritmo para o problema do maior caminho:

```
for iterações do
  for formigas do
    inicializa formiga
    formiga procura uma solução recursiva
    if é uma solução válida then
      deposita feromônio
      atualiza a melhor solução local
      atualiza o feromônio globalmente
    end if
  end for
  atualiza a melhor solução global
end for
```

2.4.2 Pseudo código: Atualização global do feromônio

A cada iteração, a formiga que percorreu o maior caminho em relação as outras formigas, deposita mais feromônio também representado pro uma matriz de adjacências. A quantidade inicial, máxima e mínima de feromônio que uma formiga pode soltar são parâmetros de entrada do modelo.

Antes de atualizar os feromônios, a matriz de feromônios é evaporada de acordo com o valor de entrada **taxa de evaporação** . Esta taxa controla a convergência. Uma taxa muito baixa faz o algoritmo convergir rapidamente, pois as formigas vão escolher caminhos com alto teor de feromônio. Uma taxa muito alta deixa o algoritmo aleatório e não há progressão na solução.

O pseudo código da atualização global de feromônio é:

```
for i vértices do
  for j vértices do
```

if i != j **then**
remove o valor de evaporação para a aresta entre i e j
end if
end for
end for
deposita mais feromônio na *melhor* solução

2.5 Formiga: Escolhendo um novo vértice

A decisão de escolher um novo vértice para construir um caminho válido é um dos principais fatores para a convergência do ACO. Para que a solução seja boa, será preciso levar em consideração o peso da aresta (distância) e a quantidade de feromônio depositada. Desta forma ela cumpre sua função local que é a maior peso da aresta e sua função global, representada pelo feromônio depositado. A escolha do próximo nó a ser incorporado na solução é baseada em uma função de distribuíção de probabilidade:

$$P_{ij} = \frac{F_{ij}^{\alpha} + W_{ij}^{\beta}}{\sum_{j=0}^{N} F_{ij}^{\alpha} + W_{ij}^{\beta}} \tag{1}$$

Em que [i,j] é a aresta que sai de i e chega em j, F_{ij} é a quantidade de feromônio na aresta, W_{ij} é o peso da aresta, α e β são parâmetros de entrada relacionados ao Feromônio e ao Peso, respectivamente.

Segue o pseudo código partindo de i:

if Somente um vizinho j **then**
retorna j
end if
for j sendo cada vizinhos **do**
Calcula o valor de P_{ij}
Guarda o melhor vizinho
end for
if fator de exploitation **then**
retorna o melhor vizinho
else
retorna um vizinho de acordo com a distribuição de probabilidade da vizinhança
end if

3 Experimentos

3.1 Dados de entrada

Para os experimentos foram utilizadas 3 bases de dados diferentes, disponibilizadas em arquivos de texto simples contendo o número de nós e os pesos das arestas direcionadas do grafo. Para duas entradas sabe-se a solução ótima, enquanto que para uma das entradas não se sabe a solução ótima.

Entrada	Nº Nós	Solução Ótima
Entrada 1	100	990
Entrada 2	20	168
Entrada 3	1000	*Sem resultado conhecido*

3.2 Análise de sensibilidade dos Parâmetros

3.2.1 Metodologia dos testes de sensibilidade

Primeiramente iremos executar testes para analisar alguns parâmetros de entrada do problema. Os valores serão executados separadamente para cada entrada do problema.

1. α e β

4/1	3/1	2/1	3/2	1/1	2/3	1/2	1/3	1/4

α e β

2. Exploitation Factor

0.02	0.05	0.1	0.15	0.2

Exploitation Factor

3. Número de Formigas

20	40	60	80	100	120	140	160	180	200

Ant

Para isto, foi analisado cada um dos parâmetros em separado mantendo fixo todos os outros parâmetros. Então, utilizaremos os melhores valores descobertos para executar um teste completo parametrizado para encontrar a melhor solução.

A intenção é encontrar valores razoáveis que executarão em tempo hábil no ambiente de testes e que aproximavam melhor dos valores ótimos. O número de iterações para a solução ótima foi fixado em 100. Como métrica de análise dos parâmetros isolados, foi considerado o melhor valor de custo do resultado para o problema Longest Path - ACO.

3.3 Análise de sensibilidade: α e β

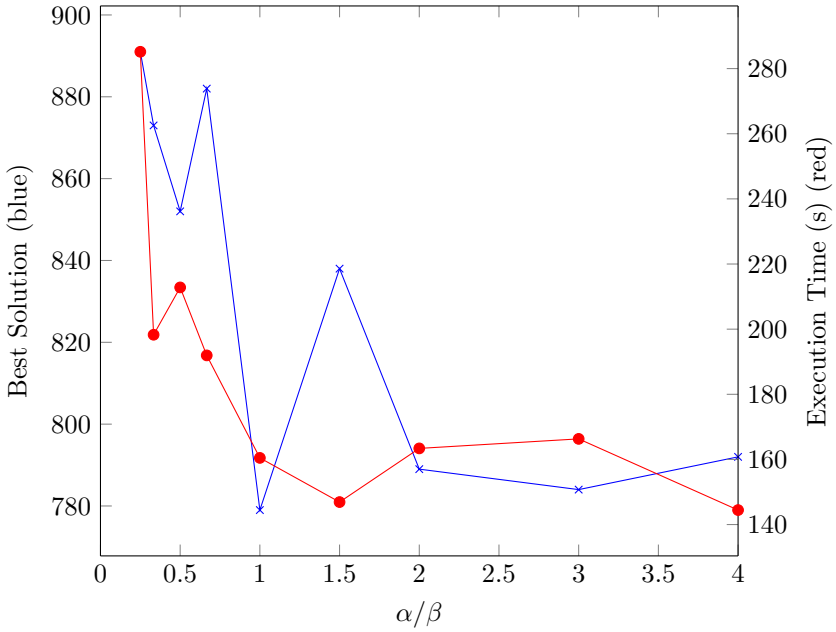
Iremos utilizar o melhor valor de α e β baseando no valor do resultados para diferentes valores para α e α conforme metodologia acima.

3.3.1 α/β Entrada 1

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	160
Iterations	100
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	0.1
Alpha/Beta	Variáveis

Soluções e tempos para variados α/β



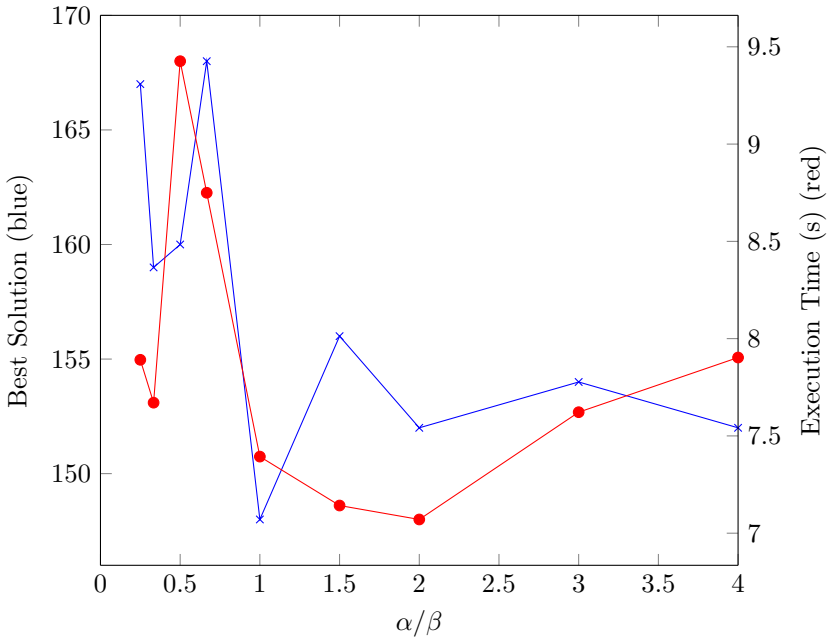
Podemos observar que o melhor resultado de α e β para a Entrada 1 é 0,25 ou α 1 e β 4 com resultado 285,16.

3.3.2 α/β Entrada 2

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	160
Iterations	100
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	0.1
Alpha/Beta	Variáveis

Soluções e tempos para variados α/β



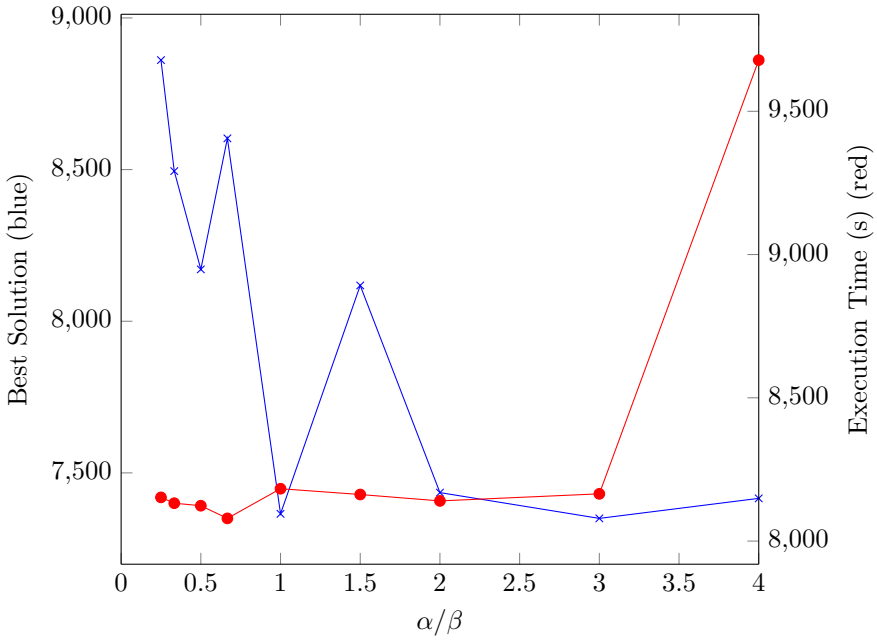
Podemos observar que o melhor resultado de α e β para a Entrada 2 é 0,66667 ou α 2 e β 3 com resultado 168.0.

3.3.3 α/β Entrada 3

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**100**
Iterations	**10**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	0.1
Alpha/Beta	Variáveis

Soluções e tempos para variados α/β



Podemos observar que o melhor resultado de α e β para a Entrada 3 é 0,25 ou α 4 e β 1 com resultado 285,16.

3.4 Análise de sensibilidade: Exploitation Factor

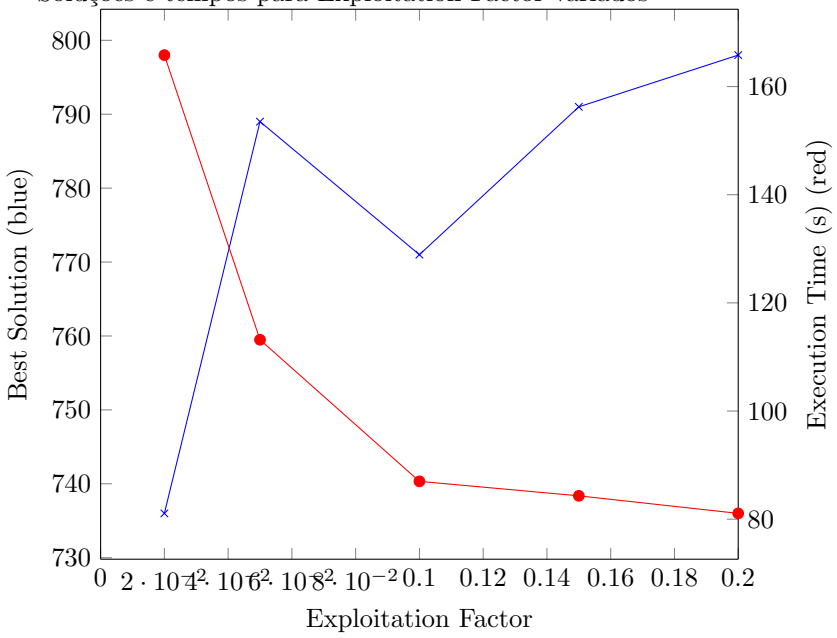
Iremos utilizar o melhor valor de Exploitation Factor baseando no valor do resultados para diferentes valores, conforme metodologia.

3.4.1 Exploitation Factor: Entrada 1

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**100**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	Variável
Alpha/Beta	1/1

Soluções e tempos para Exploitation Factor variados



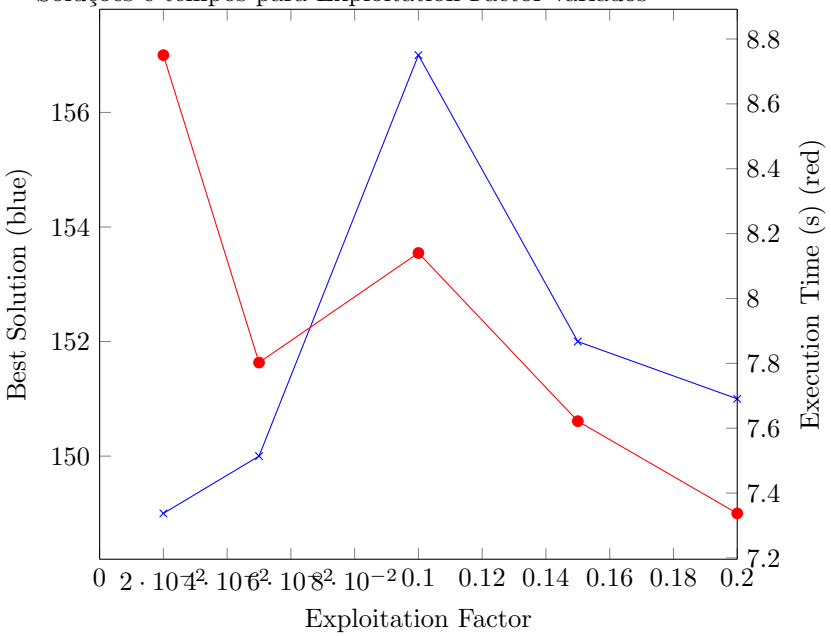
Podemos observar que o melhor resultado de Exploitation Factor para a Entrada 1 é 0,2 com resultado 798.0.

3.4.2 Exploitation Factor: Entrada 2

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**100**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	Variável
Alpha/Beta	1/1

Soluções e tempos para Exploitation Factor variados



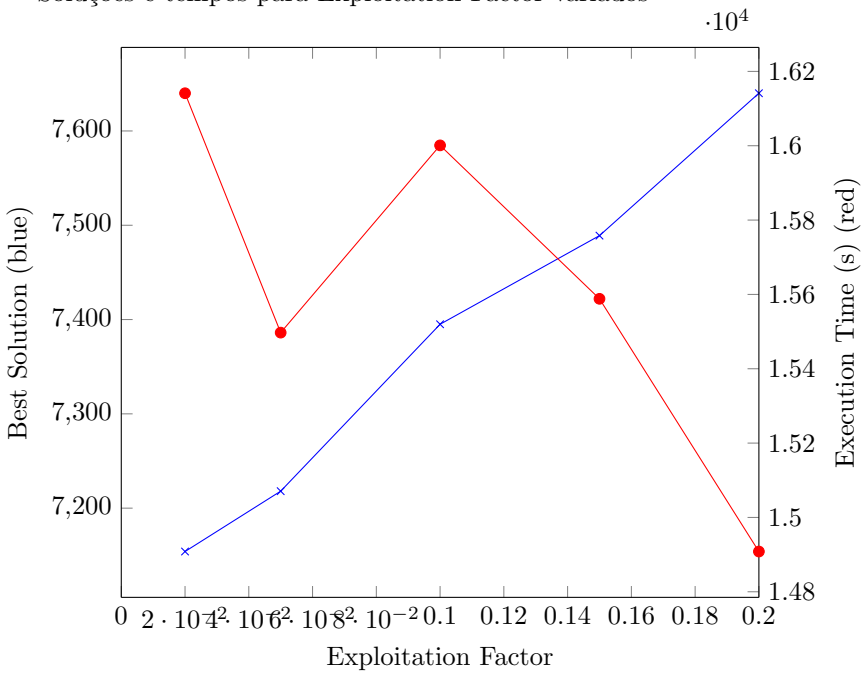
Podemos observar que o melhor resultado de Exploitation Factor para a Entrada 2 é 0,1 com resultado 157.0.

3.4.3 Exploitation Factor: Entrada 3

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**60**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	Variável
Alpha/Beta	1/1

Soluções e tempos para Exploitation Factor variados



Podemos observar que o melhor resultado de Exploitation Factor para a Entrada 2 é 0,1 com resultado 157.0.

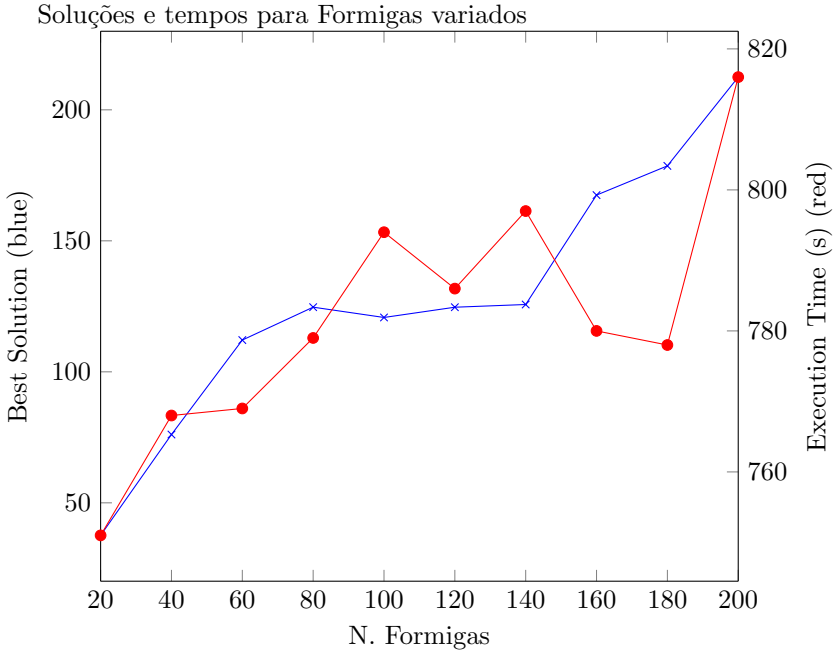
3.5 Análise de sensibilidade: Formigas

Iremos utilizar o melhor valor de número de formigas, baseando no valor do resultados para diferentes valores, conforme metodologia.

3.5.1 Formigas: Entrada 1

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**Variável**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	**0.1**
Alpha/Beta	1/1



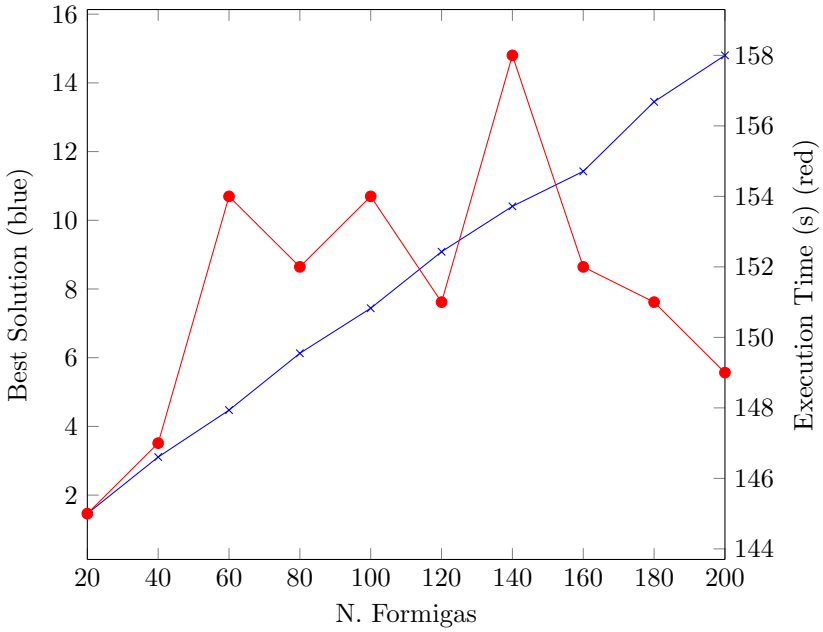
Podemos observar que o melhor resultado de Formigas para a Entrada 1 é com resultado 200 .

3.5.2 Formigas: Entrada 2

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**Variável**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	**0.1**
Alpha/Beta	1/1

Soluções e tempos para Formigas variados



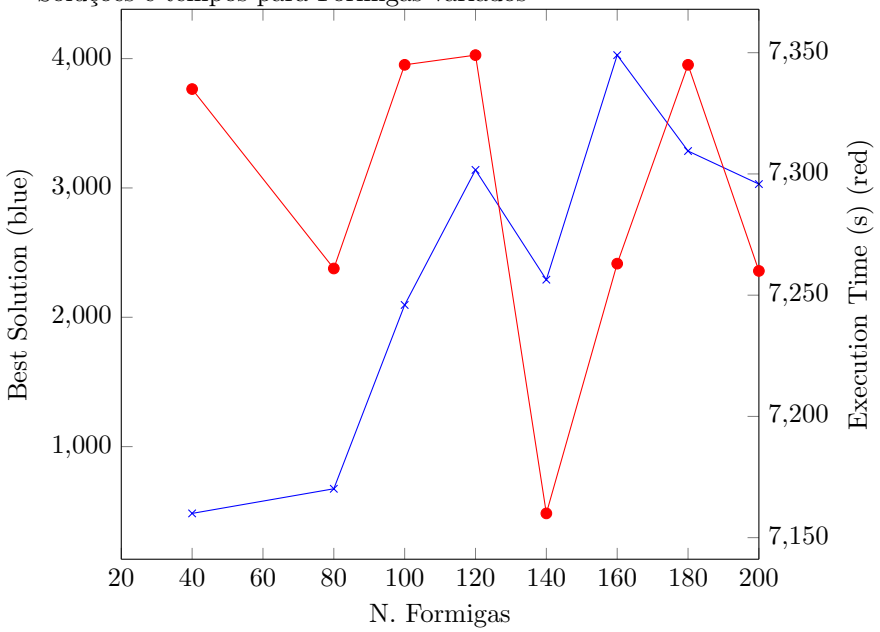
Podemos observar que o melhor resultado de Formigas para a Entrada 2 é com resultado 200.

3.5.3 Formigas: Entrada 3

Parâmetros:

Parâmetro	Valor
Runs	1
Ants	**Variável**
Iterations	**100**
Initial pheromone amount	20.0
Ant pheromone deposit amount	1.0
Pheromone evaporation rate	0.25
Daemon pheromone deposit amount	2.0
Max pheromone threshold	100.0
Min pheromone threshold	0.00
Exploitation factor	**0.1**
Alpha/Beta	1/1

Soluções e tempos para Formigas variados



Podemos observar que o melhor resultado de Formigas para a Entrada 3 é com resultado 160.

3.6 Melhores Resultados: Aplicando a análise de sensibilidade no Problema

Após determinarmos os melhores valores para os parâmetros estudos, dentre os valores analisados, iremos aplicá-los ao problema para determinar a melhor solução para o problema Longest Path utilizando ACO.

Abaixo vemos uma tabela com detalhes de cada configuração a ser testada.

Entrada	Execuções	Formigas	Iterações	Exploitation Factor	α/β
Entrada 1	100	200	100	0.2	1/4
Entrada 2	100	200	100	0.1	2/3
Entrada 3	10	160	100	0.2	1/4

3.6.1 Resultados: Entrada 1

Melhor Resultado Encontrado: **926.0**
Número de execuções: **100 execuções**
Número de iterações: **100 iterações**

Resultado Ótimo: 990
Resultado Encontrado / Ótimo : 926/990 **93.5%**

Tempo Total: **5.7 hrs.** Tempo médio: **207 segs.**

Caminho: [0, 26, 42, 24, 78, 91, 8, 3, 37, 85, 38, 66, 58, 79, 15, 1, 11, 98, 2, 30, 5, 62, 57, 28, 19, 45, 86, 32, 22, 96, 53, 20, 83, 16, 75, 88, 14, 51, 73, 93, 69, 90, 31, 6, 76, 63, 4, 18, 55, 64, 52, 13, 40, 39, 94, 9, 97, 17, 67, 50, 35, 71, 70, 34, 77, 82, 61, 12, 36, 65, 44, 23, 25, 48, 49, 72, 27, 54, 81, 92, 68, 29, 21, 84, 95, 10, 43, 56, 89, 60, 80, 59, 7, 74, 87, 47, 33, 41, 46, 99]

3.6.2 Resultados: Entrada 2

Melhor Resultado: **168.0**
Número de execuções: **100 execuções**
Número de iterações: **100 iterações**

Resultado Ótimo: 168
Resultado Encontrado / Ótimo : 168/168 **100%**

Tempo Total: **10 min.** Tempo Médio: **6 segs.**

Caminho: [0, 18, 14, 1, 9, 10, 3, 6, 12, 8, 7, 11, 16, 17, 15, 5, 4, 2, 13, 19]

3.6.3 Resultados: Entrada 3

Melhor Resultado: **9,063**
Número de execuções: **10 execuções**
Número de iterações: **100 iterações**

Tempo Total de Execução **7,57 hrs.** Tempo Médio: **46min.**

Caminho: Segue juntamente do código fonte.

3.7 Conclusão e Próximos passos

Foi implementado um ACO buscando resolver o problema do Longest Path com resultados de 93,5% da solução ótima no primeiro problema, 100% no segundo e um resultado de 9,063 para o terceiro problema.

Como foi verificado, alguns parâmetros poderiam ser melhorados ainda mais, visto que ficaram na borda do domínio que foram estudados, como o número de formigas e o exploitation factor no exemplo 3.

O grande problema na classe de problemas metaheurísticas é selecionar os melhores valores para os parâmetros possibilita minimizar o processamento e equilibrar a taxa de convergência do problema.

Mesmo aplicando conceitos de paralelização, ainda assim o tempo computacional atrapalhou a execução dos testes. Uma alternativa seria aplicar o problema em uma linguagem mais rápida como java ou c.

3.8 Ambiente de Testes

OS X El Capitan 10.11.1 ($15B42$)
MacBook Pro (Retina, 13-inch, Early 2015)
2.9 GHz Intel Core i5
8 GB 1867 MHz DDR3

Referências

[1] M. Dorigo, V. Maniezzo e A. Colorni, *The Ant System: Optimization by a colony of cooperating agents*, 1996.

[2] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Politecnico di Milano, Italy. 1992.

[3] M. Dorigo, *Ant Colony Optimization*, Scholarpedia, 2007.

[4] Wan-Shiou Yang e Shi-Xin Weng, *Application of the Ant Colony Optimization Algorithm to the Influence-Maximization Problem*, 2012.