Programa de Pós-Graduação em Engenharia Elétrica
Mestrado - Doutorado (Conceito 4 CAPES)

Tópicos Especiais de Otimização - Técnicas Inteligentes – 210115-IC

Programa de Pós-Graduação em Engenharia Elétrica / UFJF

Prof. Leonardo Willer de Oliveira/ Prof. Ivo Chaves da Silva Junior

Relatório de Técnicas Inteligentes: Otimização por colônia de fomigas

Maximização da corrente de um circuito elétrico

Doutorando: Thainan Santos Theodoro

Matrícula: 103080112

thainan.theodoro@engenharia.ufjf.br

Juiz de Fora, 17 de abril de 2016

Sumário

1. Introdução

Estes relatório tem a finalidade descrever a construção de um programa computacional baseado no algoritmo de otimização por colônia de formigas (ACO – "*Ant Colony Optmization*") que maximiza uma determinada corrente em um circuito de corrente contínua. Tanto a modelagem do problema, quanto o desenvolvimento do programa são descritos detalhadamente. No final, baterias de testes são feitos para avaliar o desempenho do algoritmo na obtenção da resposta quanto ao número de formigas, número de iterações e evaporação.

2. Dados do problema

A Figura 1 e a Tabela 1 mostram o circuito elétrico para otimização e respectivos dados do problema [1].

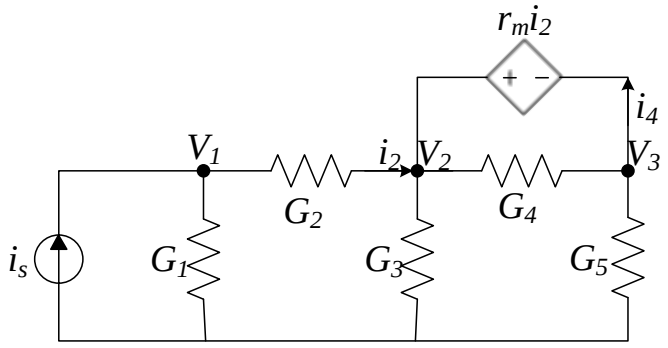

Figura 1 – Circuito elétrico para otimização.

Tabela 1 – Dados do problema.

Grandeza	Valor
i_s	110 A
V_1	10,6 V
V_2	10,2 V
V_3	-9,8 V
G_1	10,0 S
G_2	10,0 S

O propósito do algoritmo ACO é obter os valores ótimos de G_3, G_4 e G_5 de forma a maximizar a corrente i_4 na fonte controlada de tensão, dados os possíveis valores das condutâncias organizados na Tabela 2.

Tabela 2 – Possíveis valores de G_3, G_4 e G_5.

Possibilidades	1	2	3	4	5	6	7	8	9	10	11	12
Valores (S)	1	3	5	10	12	15	17	20	22	25	27	30

O valor de r_m pode ser encontrado a partir dos valores da Tabela 1 e da Equação (2):

$$r_m = \frac{(V_2 - V_3)}{G_2(V_1 - V_2)} = \frac{\big(10{,}2 - (-9{,}8)\big)}{10(10{,}6 - 10{,}2)} = 5\ \Omega \tag{1}$$

3. Formulação natural

A seguir é desenvolvida a formulação natural de resolução de circuitos (isto é, quando se conhece as condutâncias e se quer saber as correntes e tensões) com intuito de fornecer meios para verificação da resposta com ACO. Aplicando a lei dos nós em (1), (2) e (3), tem-se:

Nó 1:

$$i_s = G_2(V_1 - V_2) + G_1 V_1$$

$$V_1(G_1 + G_2) + V_2(-G_2) = i_s \tag{2}$$

Nó 2:

$$G_2(V_1 - V_2) = -i_4 + G_4(V_2 - V_3) + G_3 V_2$$

$$V_1(-G_2) + V_2(G_2 + G_3 + G_4) + V_3(-G_4) - i_4 = 0 \tag{3}$$

Nó 3:

$$G_4(V_2 - V_3) = i_4 + G_5 V_3$$

$$V_2(-G_4) + V_3(G_4 + G_5) + i_4 = 0 \quad (4)$$

Aplicando a lei de malhas na fonte de tensão, pode-se dizer:

$$V_2 - V_3 = r_m i_2$$

$$V_2 - V_3 - r_m G_2(V_1 - V_2) = 0$$

$$V_1(-r_m G_2) + V_2(1 + r_m G_2) = 0 \quad (5)$$

Dessa forma, é pode-se construir um sistema de 4 equações: (2), (3), (4), e (5) e 4 incógnitas: V_1, V_2, V_3 e i_4, que matricialmente pode ser escrito como:

$$[G][V] = [I]$$

$$\begin{bmatrix} G_1 + G_2 & -G_2 & 0 & 0 \\ -G_2 & G_2 + G_3 + G_4 & -G_4 & -1 \\ 0 & -G_4 & G_4 + G_5 & 1 \\ -r_m G_2 & 1 + r_m G_2 & -1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} i_s \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (6)$$

4. Teste de todas as possibilidades

Foram testadas todas as possibilidades de soluções do circuito considerando os valores de G_3, G_4 e G_5 apresentados na Tabela 2, ou seja, foram testadas 1.728 combinações ($12^3 = 1.728$). A Tabela 3 mostra as soluções que geram os valores de $V_1 = 10{,}6\ V$, $V_2 = 10{,}2\ V$, $V_3 = -9{,}8\ V$ dados, dentro de uma faixa de $\pm 0{,}1\ V$.

Tabela 3 – Soluções da Equação (6) considerando os valores de V_1, V_2, V_3 para faixa de $\pm 0{,}1\ V$.

Solução	$G_3\ (S)$	$G_4(S)$	$G_5(S)$	V_1	V_2	V_3	i_4
01	**10**	**1**	**10**	**10,6000**	**10,2000**	**-9,8000**	**118,0000**
02	12	1	12	10,5988	10,1976	-9,8632	138,4195
03	**10**	**3**	**10**	**10,6000**	**10,2000**	**-9,8000**	**158,0000**
04	12	3	12	10,5988	10,1976	-9,8632	178,5410
05	**10**	**5**	**10**	**10,6000**	**10,2000**	**-9,8000**	**198,0000**
06	12	5	12	10,5988	10,1976	-9,8632	218,6626
07	**10**	**10**	**10**	**10,6000**	**10,2000**	**-9,8000**	**298,0000**
08	12	10	12	10,5988	10,1976	-9,8632	318,9666
09	**10**	**12**	**10**	**10,6000**	**10,2000**	**-9,8000**	**338,0000**
10	12	12	12	10,5988	10,1976	-9,8632	359,0881
11	**10**	**15**	**10**	**10,6000**	**10,2000**	**-9,8000**	**398,0000**
12	12	15	12	10,5988	10,1976	-9,8632	419,2705
13	**10**	**17**	**10**	**10,6000**	**10,2000**	**-9,8000**	**438,0000**
14	12	17	12	10,5988	10,1976	-9,8632	459,3921
15	**10**	**20**	**10**	**10,6000**	**10,2000**	**-9,8000**	**498,0000**
16	12	20	12	10,5988	10,1976	-9,8632	519,5745
17	**10**	**22**	**10**	**10,6000**	**10,2000**	**-9,8000**	**538,0000**
18	12	22	12	10,5988	10,1976	-9,8632	559,6960
19	**10**	**25**	**10**	**10,6000**	**10,2000**	**-9,8000**	**598,0000**
20	12	25	12	10,5988	10,1976	-9,8632	619,8784
21	**10**	**27**	**10**	**10,6000**	**10,2000**	**-9,8000**	**638,0000**
22	12	27	12	10,5988	10,1976	-9,8632	660,0000
23	**10**	**30**	**10**	**10,6000**	**10,2000**	**-9,8000**	**698,0000**
24	12	30	12	10,5988	10,1976	-9,8632	720,1824

Foram encontradas 24 soluções dentro da faixa escolhida e foram dispostas em ordem crescente. As soluções ímpares mostram as soluções exatas do problema da Equação (6) e as pares as soluções que não são exatas, porém estão dentro da faixa proposta. A solução (1) dá o menor valor de $i_4 = 118{,}0000\ [A]$ e a solução (23) o maior exato de $i_4 = 698{,}0000\ [A]$. A solução (24) dá o maior valor de $i_4 = 720{,}1824\ [A]$, no entanto fere os valores exatos propostos no problema.

Dessa forma, espera-se que o algoritmo ACO apresente o resultado da linha (23) da Tabela 3. A Figura 2 e a Tabela 4 mostram os valores das tensões e correntes para esta solução.

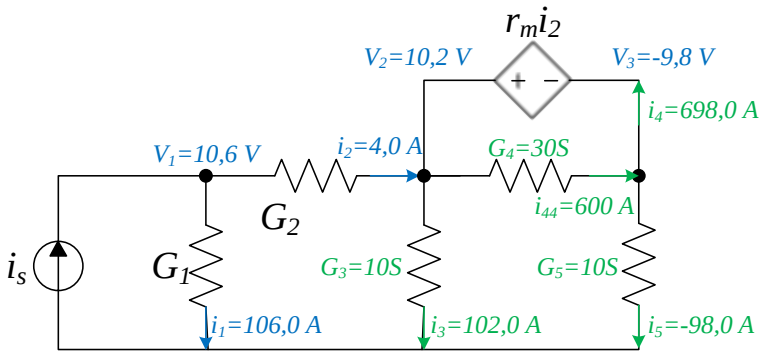

Figura 2 – Circuito, valores das grandezas para a solução exata (23).

Tabela 4 – Valores das grandezas para a solução exata (23).

Grandeza	Valor
G_3	10 S
G_4	30 S
G_5	10 S
i_4	698,0 V
i_3	102,0 A
i_{44}	600,0 A
i_5	-98,0 A

5. Formulação da FOB e a restrição

A seguir é desenvolvida a formulação da FOB e a restrição que serão aplicadas no programa computacional baseado na otimização por colônia de formigas. O objetivo é encontrar uma expressão para i_4 em função de G_3, G_4 e G_5, ou seja, $FOB = i_4(G_3, G_4, G_5)$.

Somando as Equações (3) e (4), resultado da aplicação da lei dos nós nos Nós (2) e (3), pode-se encontrar a expressão para $i_4(G_3, G_4, G_5)$:

$$\mathrm{i_4} = \mathrm{G_4(V_2 - V_3) + G_3(V_2) - i_2} \qquad \text{(7) da Equação (3)}$$

$$\mathrm{i_4} = \mathrm{G_4(V_2 - V_3) - G_5(V_3)} \qquad \text{(8) da Equação (4)}$$

$$\mathrm{2i_4 = G_3V_2 + 2G_4(V_2 - V_3) - G_5V_3 - i_2} \qquad \text{(7) + (8)}$$

$$\mathrm{i_4 = FOB(G_3, G_4, G_5) = G_3\left(\frac{V_2}{2}\right) + G_4(V_2 - V_3) + G_5\left(\frac{-V_3}{2}\right) - \frac{i_2}{2}} \qquad (9)$$

Para encontrar a restrição é aplicado o conceito de supernó entre os nós (2) e (3), conforme:

$$i_2 = i_3 + i_5$$

$$\mathrm{G_3V_2} + G_5V_3 = i_2 \qquad (10)$$

Com as Equações (9) e (10) pode-se modelar o sistema a ser otimizado.

$$\text{Maximize:} \quad FOB(G_3, G_4, G_5) = G_3\left(\frac{V_2}{2}\right) + G_4(V_2 - V_3) + G_5\left(\frac{-V_3}{2}\right) - \frac{i_2}{2}$$

$$\text{Sujeito a:} \quad \mathrm{G_3V_2} + G_5V_3 = i_2 \qquad (11)$$

6. Programa computacional ACO desenvolvido

O fluxograma de aplicação do algoritmo de otimização por colônia de formigas é mostrado na Figura 3 [1].

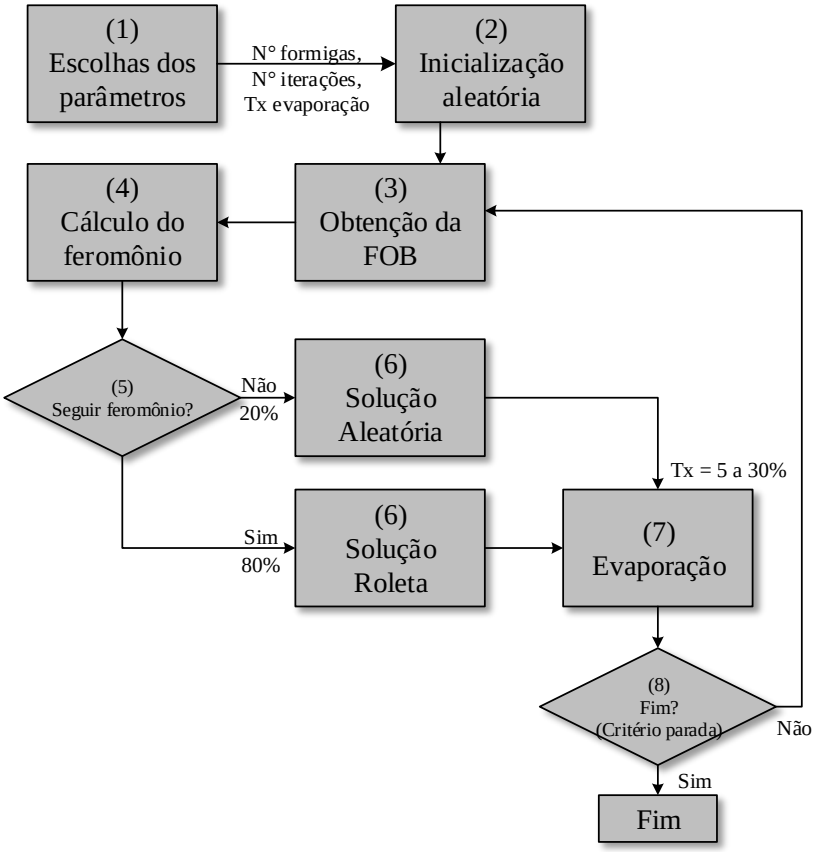

Figura 3 – Fluxograma de funcionamento da otimização por colônia de formigas ([1] adaptado).

Nas secções seguintes serão discutidos cada passo e como foram implementados pelo autor.

1.1. Passo 1 – Escolhas dos parâmetros

A tabela a seguir resume os parâmetros escolhidos inicialmente. Na fase de testes estes parâmetros serão variados a fim de comparação da performance.

Tabela 5 – Parâmetros usados no algoritmo.

Parâmetro	Valor
N° de formigas	30
N° de iterações (critério de parada)	400
Taxa de evaporação	0,15

1.2. Passo 2 – Inicialização aleatória

Para a inicialização das possíveis soluções (formigas) foi usada a função "`randsample`" que pode escolher um valor aleatoriamente (mesma probabilidade) dentre vários em um vetor. O trecho a seguir tirado do código mostra o funcionamento.

```
xA = [1 3 5 10 12 15 17 20 22 25 27 30]; % Possibilidades de G3
xB = xA; % Possibilidades de G4
XC = xA; % Possibilidades de G5
F = 30; % N° de formigas
for k = 1:F
    FORMIGAS(k,:) = [randsample(xA,1) randsample(xB,1) randsample(xC,1)];
end, clear k
```

1.3. Passo 3 – Obtenção da FOB

Neste passo é obtida a FOB e as restrições (Equação (11)) a partir das soluções (formigas - G_3, G_4 e G_5). Como a restrição é de igualdade, optou-se por dividir a igualdade em duas inequações com a inclusão de um erro, escolhido arbitrariamente, pois pode acontecer uma "falsa diferença" em uma comparação devido à forma de armazenamento valor (lógico ou double). Dessa forma:

$$\begin{aligned} equação &= restrição \\ equação &< restrição(1 + erro) \\ equação &> restrição(1 - erro) \end{aligned} \tag{12}$$

No caso do problema de maximização as penalidades devem ser no sentido de diminuir a FOB (inversamente ao problema de minimização) [1], [2]. Assim, se uma das soluções não atende a restrição, sua FOB correspondente deve ser "diminuída", escolheu-se o multiplicador $penalidade = 0{,}001$. O trecho do programa ilustra as operações.

```
FOB = FORMIGAS*[V2/2 V2-V3 V3/2] - I2/2;
RESTRICOES = [V2 0 V3];
erro = 1e-6;
PENALIDADES = 1e-3;

rest1 = FORMIGAS*RESTRICOES';
teste = (rest1<RESTRICOES*(1+erro) & rest1>RESTRICOES*(1-erro));
teste = teste + 0; % logical2double
teste(teste==0) = PENALIDADES;
FOB = FOB.*teste;
```

1.4. Passo 4 – Cálculo do feromônio artificial

No caso de um problema de maximização os feromônios artificiais devem ser proporcionais à FOB [1], [2] e [3]. Optou aqui por apenas normalizar a FOB calculando assim o ferômonio. Dessa forma:

$$\tau_k = \frac{FOB_k}{\sum_{k=1}^{n} FOB_k} \tag{13}$$

1.5. Passo 5 – Seguir rastro de ferômonios

Um dos pontos chaves da estratégia do algoritmo é seguir ou não o rastro do feromônio, afinal isso determina se as formigas poderão encontrar novas soluções. Essa implementação foi feita de forma simples também com a função "`randsample`". Escolhe-se as opções de seguir o ferômonio ou não com probabilidades distintas de 80 e 20% respectivamente. O trecho do programa mostra a implementação.

```
F = 3;
decisao = randsample([0 1], F, true, [0.2 0.8]);
% decisão == 0 :. não segue o feromonio
% decisão == 1 :. segue o feromonio
```

1.6. Passo 6 – Solução: Aleatória x Roleta

Com a decisão a solução pode-se ou não seguir o caminho do feromônio. Se a decisão for seguir aleatoriamente a implementação é idêntica a da Seção 1.2 e será suprimida aqui. Se a decisão for seguir o rastro do feromônio, é necessário a passagem pela roleta.

Essa técnica consiste em avaliar a probabilidade de uma formiga seguir um caminho em função da quantidade de feromônio presente na solução. Uma roleta sorteia qual solução seguir em função da probabilidade calculada. A expressão usada para a probabilidade é dada por:

$$P_j = \frac{\tau_j}{\sum_{k=1}^{F} \tau_k} \tag{14}$$

Onde P_j é a probabilidade da solução j, τ_j é o feromônio da solução j e $\sum_{k=1}^{F} \tau_k$ é o somatório de todos os feromônios.

Na implementação do algoritmo foi feito o armazenamento dinâmico para: (*i*) as soluções, (*ii*) o feromônio e (*iii*) as probabilidades, isto é, de acordo que as soluções são visitadas os dados são armazenados. Sendo assim, a roleta consiste em escolher uma solução dentre as visitadas em função da probabilidade aferida. Para isso também usou-se a função "`randsample`" só que agora com probabilidades distintas para as soluções, conforme mostra o trecho do código a seguir com alocação dinâmica:

```
F = 3;
for k = 1:F
  % Escolha de uma das soluções visitadas a partir das probabilidades
      % datasample(universo,n°amostras,'Weights',probabilidades)
  solucao = datasample(1:size(FEROMONIO,1),1,'Weights',FEROMONIO(:,end)');
  % Atualização da solução sorteada
  FORMIGAS(k,1:3) = FEROMONIO(solucao,1:3);
end
```

1.7. Passo 7 – Evaporação

A evaporação é importante pois no decorrer do tempo penaliza as soluções "ruins" em detrimento das boas, mas a escolha do seu valor é igualmente importante pois soluções boas podem ser penalizadas erroneamente [1]. Valores usuais da taxa de evaporação vão de 5 a 30% [1]. A expressão usada para aplicação do conceito de evaporação é dada por:

$$\tau_i^{h+1} = (1 - \rho)\tau_i^{h+1} \tag{15}$$

A aplicação no matlab é simples:

```
FEROMONIO(:,4) = (1 - ro)*FEROMONIO(:,4);
```

1.8. Passo 8 – Critério de parada

Como critério de parada, foi escolhido o número de iterações, em detrimento da estagnação, pela simplicidade conforme mostra a Tabela 5.

7. Testes no algoritmo de otimização por colônia de formigas

1.1. Apresentação dos dados

A fim de comparar o desempenho do algoritmo variou-se: (*i*) o n° de formigas, (*ii*) o n° de iterações e (*iii*) a taxa de evaporação em 18 topologias conforme mostra a Tabela 6. Cada topologia foi simulada 10 vezes.

Tabela 6 – Testes feitos com o algoritmo.

N°	Topologia	N° de Formigas	N° iterações	Taxa de evaporação
01	15-300-05	15	300	05%
02	15-300-15			15%
03	15-500-05		500	05%
04	15-500-15			15%
05	15-700-05		700	05%
06	15-700-15			15%
07	30-300-05	30	300	05%
08	30-300-15			15%
09	30-500-05		500	05%
10	30-500-15			15%
11	30-700-05		700	05%
12	30-700-15			15%
13	45-300-05	45	300	05%
14	45-300-15			15%
15	45-500-05		500	05%
16	45-500-15			15%
17	45-700-05		700	05%
18	45-700-15			15%

Os resultados dos testes são mostrados na Tabela 7. Mostrando as taxas de acertos das condutâncias e corrente e o tempo de processamento medido.

Tabela 7 – Resultados dos testes para variação de parâmetros.

N°	Topologia	N° Formigas	N° Iterações	Taxa de evaporação	Taxa de acertos				Tempo médio (s)
					G3 [%]	G4 [%]	G5 [%]	I4 [%]	
1	15-300-05	15	300	0,05	100	20	100	20	2,78
2	15-300-15	15	300	0,15	100	50	100	50	2,72
3	15-500-05	15	500	0,05	100	10	100	10	5,12
4	15-500-15	15	500	0,15	100	30	100	30	5,50
5	15-700-05	15	700	0,05	100	30	100	30	9,00
6	15-700-15	15	700	0,15	100	40	100	40	8,88
7	30-300-05	30	300	0,05	100	30	100	30	6,64
8	30-300-15	30	300	0,15	100	30	100	30	6,08
9	30-500-05	30	500	0,05	100	60	100	60	12,70
10	30-500-15	30	500	0,15	100	60	100	60	12,52
11	30-700-05	30	700	0,05	100	70	100	70	19,23
12	30-700-15	30	700	0,15	100	30	100	30	19,46
13	45-300-05	45	300	0,05	100	30	100	30	10,36
14	45-300-15	45	300	0,15	100	50	100	50	10,79
15	45-500-05	45	500	0,05	100	60	100	60	26,18
16	45-500-15	45	500	0,15	100	60	100	60	25,01
17	45-700-05	45	700	0,05	100	80	100	80	30,71
18	45-700-15	45	700	0,15	100	50	100	50	38,71

O algoritmo de otimização por colônia de formigas não tem garantia de atingir o ótimo global em função de sua característica probabilística, dessa forma ele pode atingir falsos máximos, isto é, resultados errados.

As Tabela 8 mostra a classificação das topologias em função do desempenho de acertos da FOB.

Tabela 8 – Organização das topologias em função do desempenho.

N°	Topologia	N° Formigas	N° Iterações	Taxa de evaporação	Taxa de acertos				Tempo médio (s)
					G3 [%]	G4 [%]	G5 [%]	I4 [%]	
17	45-700-05	45	700	0,05	100	80	100	80	30,71
11	30-700-05	30	700	0,05	100	70	100	70	19,23
9	30-500-05	30	500	0,05	100	60	100	60	12,70
10	30-500-15	30	500	0,15	100	60	100	60	12,52
15	45-500-05	45	500	0,05	100	60	100	60	26,18
16	45-500-15	45	500	0,15	100	60	100	60	25,01
2	15-300-15	15	300	0,15	100	50	100	50	2,72
14	45-300-15	45	300	0,15	100	50	100	50	10,79
18	45-700-15	45	700	0,15	100	50	100	50	38,71
6	15-700-15	15	700	0,15	100	40	100	40	8,88
4	15-500-15	15	500	0,15	100	30	100	30	5,50
5	15-700-05	15	700	0,05	100	30	100	30	9,00
7	30-300-05	30	300	0,05	100	30	100	30	6,64
8	30-300-15	30	300	0,15	100	30	100	30	6,08
12	30-700-15	30	700	0,15	100	30	100	30	19,46
13	45-300-05	45	300	0,05	100	30	100	30	10,36
1	15-300-05	15	300	0,05	100	20	100	20	2,78
3	15-500-05	15	500	0,05	100	10	100	10	5,12

1.2. Conclusões parciais

A melhor topologia, segundo o desempenho, é a (17) com 45 formigas, 700 iterações e taxa de evaporação de 0,05 com 80% de chances de acerto. É interessante notar que na topologia (18) também com 45 formigas e 700 iterações o desempenho é de 50%, fato que só pode estar ligado à taxa de evaporação. Pode-se concluir que para muitas formigas (45) a combinação de um grande número de iterações e uma alta taxa de evaporação pode deturpar o resultado, neste caso diminuiu a taxa de acerto em 30%.

As piores topologias (3) e (1) tiveram desempenho de 10 e 20% com 15 formigas e 500 e 300 iterações e taxa de evaporação de 0,05. Índices muito provavelmente ligados a baixa quantidade de formigas e baixa taxa de evaporação. As configurações com 15 formigas, (6) e (4), tiveram resultados melhores (40 e 30%) com 700 e 500 iterações e taxas de evaporação de 0,15. Assim, pode-se dizer que para poucas formigas (15) a combinação de muitas iterações e alta taxa de evaporação pode melhorar o desempenho. A fim de confirmar as proposições, a Tabela 8 mostra as melhores configurações para cada quantidade de formigas.

Tabela 9 – Melhores topologias em função da quantidade de formigas.

N°	Topologia	N° Formigas	N° Iterações	Taxa de evaporação	Taxa de acertos				Tempo médio (s)
					G3 [%]	G4 [%]	G5 [%]	I4 [%]	
2	15-300-15	15	300	0,15	100	50	100	50	2,72
11	30-700-05	30	700	0,05	100	70	100	70	19,23
17	45-700-05	45	700	0,05	100	80	100	80	30,71

A relação desempenho-taxa de evaporação-número de formigas não é tão simples e varia com o número de iterações. Quando se tem poucas formigas (15) a melhor solução tem menos iterações e uma maior taxa de evaporação. Pode-se dizer que para menos iterações a velocidade de evaporação pode ser maior. Com maiores colônias de formigas (30) e (45) uma taxa de evaporação maior é mais interessante, ou seja, com mais formigas é mais interessante uma menor velocidade de evaporação. Essa questão é de fácil visualização no gráfico da Figura 4.

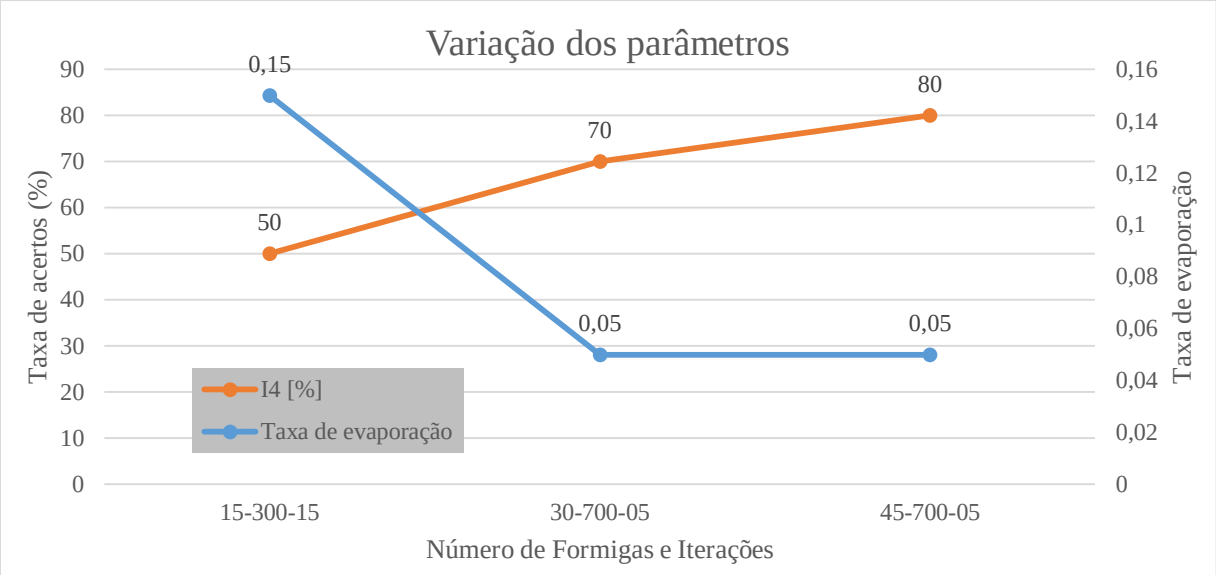

Figura 4 – Variação dos parâmetros de solução em função do número de formigas.

O gráfico da Figura 5 mostra a relação entre o desempenho e o tempo de processamento, é possível aferir, como esperado, que quanto maior o desempenho, maior o número de iterações e tempo de processamento.

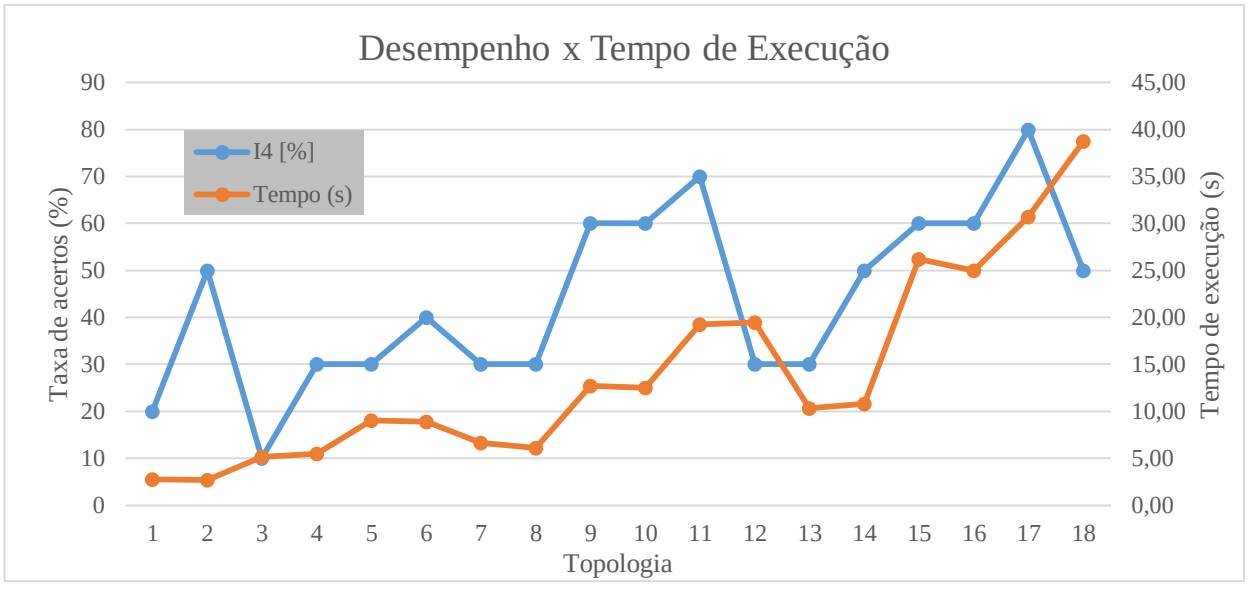

Figura 5 – Evolução do desempenho e tempo de processamento.

8. Conclusão

Neste relatório foi descrita a construção de um algoritmo baseado na otimização por colônia de formigas para solução de um problema de maximização de uma corrente em um circuito. Foi desenvolvida a FOB e a restrição do problema. Todas as combinações possíveis do problema foram avaliadas a fim de construir um ponto para validação da resposta do algoritmo.

Uma bateria de testes (18 topologias) foi conduzida para a avaliação do desempenho do algoritmo varando: (*i*) o número de formigas, (*ii*) o número de iterações e (*iii*) a taxa de evaporação. Foram avaliados dois pontos: o tempo de processamento e a taxa de acertos. A taxa de acertos é importante devido ao cunho probabilístico do algoritmo ACO, ou seja, falsos máximos podem aparecer como resultado.

Como esperado, o tempo de processamento foi proporcional ao número de iterações. A taxa de acertos variou em função do número de formigas, iterações e taxa de evaporação. Em resumo, **para os testes conduzidos**, pode-se concluir que:

I. Para um número mais baixo de formigas (15), menos iterações são mais interessantes (300), assim como uma alta taxa de evaporação (0,15), ou seja, com menos iterações uma maior velocidade de evaporação é mais eficiente;

II. Para números mais altos de formigas (30 e 45), mais iterações são mais interessantes (500 a 700), assim como baixa taxa de evaporação (0,05), isto é, com mais iterações uma menor velocidade de evaporação é mais eficiente.

9. Bibliografia

[1] I. C. d. S. Junior, *Técnicas Inteligentes: Otimização por Colônia de Formigas*, Tópicos Especiais de Otimização: 210115-IC - PPEE/UFJF, 2017.

[2] S. Katiyar, "Ant Colony Optimization: A Tutorial Review," *National Conference on Advances in Power and Control*, nº Manav Rachna International University, 2015.

[3] M. Dorigo, Ant colony optimization, The MIT Press, 2004.