Aplicações didáticas de algoritmos bio-inspirados para o projeto ótimo de filtros analógicos[1]

Rayann Pablo de Alencar Azevedo[2], Eliel Poggi dos Santos[3], Paulo Henrique da Fonseca Silva[4]

[1]Parte da pesquisa de iniciação científica do primeiro autor, financiada pelo CNPq
[2]Graduando do Curso de Engenharia Elétrica – IFPB. Bolsista do PIBITI/CNPq. e-mail: rayannazevedo@gmail.com
[3]Graduando do Curso de Engenharia Elétrica – IFPB. e-mail: elielpoggi@gmail.com
[4]Professor do Curso de Engenharia Elétrica – IFPB. e-mail: henrique@ifpb.edu.br

Resumo: Este artigo descreve aplicações didáticas de algoritmos bio-inspirados para o projeto ótimo de filtros analógicos, que são analisados no domínio da frequência através de funções de transferência. Os algoritmos bio-inspirados (algoritmo genético e a otimização por enxame de partículas) são aplicados a filtros passivos passa-faixa (RLC série e RLC paralelo) e a um filtro ativo passa-baixa Sallen-key. O processo de otimização é realizado a partir de uma especificação de projeto inicial e de uma região de interesse ou espaço de busca das variáveis de entrada. A convergência e a eficiência computacional dos algoritmos bio-inspirados implementados em Matlab® são analisadas. Estes algoritmos possibilitaram o projeto de filtros analógicos de forma bastante rápida e precisa. Suas respostas em frequência são apresentadas e discutidas. Os resultados obtidos mostram que os algoritmos bio-inspirados são ferramentas computacionais eficientes, que podem ser aplicadas com sucesso para o projeto ótimo de filtros analógicos ativos e passivos.

Palavras–chave: algoritmos bio-inspirados, filtros analógicos, aplicações didáticas, otimização

Introdução

A primeira definição de inteligência artificial (AI) foi estabelecida por Alan Turing na década de 1950. Em seus trabalhos Turing estudou como as máquinas podem ser usadas para imitar processos do cérebro humano e porque os organismos desenvolvem formas particulares, talvez um dos primeiros estudos sobre vida artificial (ENGELBRECHT, 2007). As tentativas de definição de inteligência ainda provocam extensos debates, mas o teste de Turing geralmente é aceito para identificação de um computador inteligente. A inteligência artificial pode ser definida como o estudo de como tornar os computadores capazes de fazer coisas que as pessoas fazem melhor.

A inteligência computacional (CI) é um ramo da inteligência artificial, que estuda mecanismos adaptativos para possibilitar ou facilitar o comportamento inteligente em ambientes complexos e dinâmicos. Estes mecanismos incluem os paradigmas de AI que exibem uma habilidade para aprender ou se adaptar a novas situações, generalizar, abstrair, descobrir e associar (ENGELBRECHT, 2007). São exemplos de paradigmas de CI: redes neurais artificiais, computação evolucionária, inteligência de enxame, etc.

Os algoritmos computacionais de otimização bio-inspirada executam métodos de busca global inspirados na natureza, tais como: o algoritmo genético (GA), a otimização por enxame de partículas (PSO), a otimização por colônia de formigas (ACO), o algoritmo das abelhas (BA), entre outros. Estes são exemplos de algoritmos heurísticos baseados numa população de indivíduos, que são efetivos para a solução de problemas de otimização complexos, com várias variáveis de projeto, nos quais o cálculo da função *fitness* (ou função custo) represente um baixo custo computacional (HAUPT, 1995; HAUPT; WERNER, 2007; KENNEDY; EBERHART, 1995).

Este artigo descreve aplicações didáticas de algoritmos genéticos e otimização por enxame de partículas voltadas ao projeto ótimo de filtros analógicos. São consideradas três aplicações: um filtro passa-faixa tipo RLC série para aplicação em circuitos equalizadores de áudio; um filtro passa-faixa RLC paralelo para a frequência intermediária de rádios AM (455 KHz); e um filtro ativo passa-baixa para sinais de voz. Na seção 2 descreve-se o projeto de filtros e os modelos matemáticos utilizados para análise dos filtros analógicos abordados no domínio da frequência. Os algoritmos GA e PSO são descritos resumidamente na seção 3. Os resultados obtidos são apresentados e discutidos na seção 4. Ao final do artigo são feitas algumas considerações finais e agradecimentos.

ISBN 978-85-62830-10-5

Filtros Analógicos

O projeto de um filtro analógico (causal, estável e localizado) pode ser descrito como um problema de otimização, que satisfaz a um conjunto de especificações, algumas das quais contraditórias. O objetivo é encontrar uma realização do filtro que atenda às especificações de projeto de forma satisfatória. As especificações para o projeto de filtros analógicos incluem: a resposta em frequência *H(s)* (magnitude e fase), que descreve o funcionamento em regime permanente senoidal; a resposta ao impulso *h(t)*, que descreve a resposta transitória no domínio do tempo. As etapas do processo de projeto de um filtro analógico auxiliadas por computador podem ser automatizadas, mas normalmente a experiência de um engenheiro eletricista é fundamental para obtenção de um bom resultado.

Em função de sua resposta em frequência, um filtro analógico pode ser classificado em: passa-baixa, passa-alta, passa-faixa ou rejeita-faixa. Neste artigo são abordados dois filtros passa-faixa passivos dos tipos RLC série e paralelo, bem como um filtro passa-baixa ativo tipo Sallen-key (ver Figura 1). Estes filtros são considerados simples e são apropriados para fins didáticos. A análise destes filtros foi feita no domínio da frequência, com base na magnitude (dB) da resposta em frequência 20log(|H(ω)|). A função de transferência de um filtro é definida por $H(\omega) = Vo(\omega)/Vi(\omega)$, em que: $Vi(\omega)$ *é* o sinal de tensão aplicado na entrada do filtro e $Vo(\omega)$ é o sinal obtido na saída do filtro (MUSSOI, 2004).

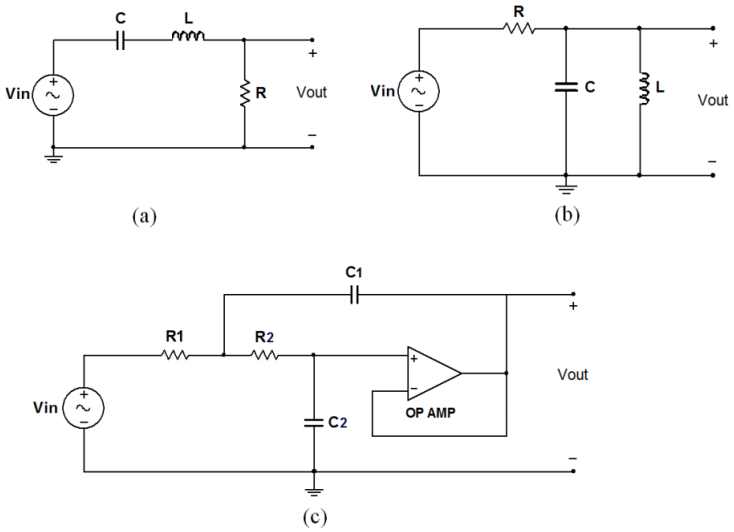


Figura 1 - Exemplos de circuitos de filtros analógicos: (a) RLC série; (b) RLC paralelo; (c) Sallen-key.

A boa seletividade, a baixa perda de inserção e a possibilidade de operação em alto nível de potência, são características que tornam os filtros RLC candidatos a muitas aplicações práticas. As topologias dos filtros RLC, são consideradas simples em relação a outros tipos de filtros passivos: mecânicos, de cavidade, cerâmicos, etc. A facilidade de análise, projeto e realização de baixo custo, são outras vantagens dos filtros RLC, que são constituídos apenas por resistor, indutor e capacitor.

Os filtros passa-faixa RLC são muito usados nos aparelhos de rádio e TV como filtros passa-faixa e filtros de harmônicos. Um filtro passa-faixa é um dispositivo que permite a passagem das frequências de certa faixa e rejeita (atenua) as frequências fora dessa faixa. Todo circuito que tenha a função de transferência de acordo com a expressão geral (1) é um filtro passa-faixa com frequência central em ω_0 e largura de banda β. As larguras de banda dos filtros RLC série e paralelo valem $\beta = R/L$ e $\beta = 1/RC$, respectivamente, e $\omega_0^2 = 1/LC$. Sua função de transferência é definida em função da frequência de operação e dos valores *R*, *L* e *C*. As funções de transferência dos filtros RLC série e paralelo são dadas em (2) e (3) respectivamente (MUSSOI, 2004).

$$H(\omega) = \frac{j\beta\omega}{\omega_0^2 - \omega^2 + j\beta\omega} \tag{1}$$

$$H(\omega) = \frac{1}{1 - j\left(\frac{1 - \omega^2 LC}{\omega RC}\right)} \tag{2}$$

$$H(\omega) = \frac{1}{1 - j\left(\omega RC - \frac{R}{\omega L}\right)} \tag{3}$$

Algoritmos Bio-Inspirados

A simulação dos algoritmos bio-inspirados parte de uma população inicial de indivíduos, que são candidatos a uma solução ótima. Considerando-se um problema de otimização com N_{var} variáveis de projeto (variáveis de entrada) e *Npop* indivíduos, a população na *i*-ésima iteração é representada por uma matriz $\boldsymbol{P}(i)_{Npop \times Nvar}$ de valores reais $p^i_{m,n}$, na qual, a cada linha *m* está associado um indivíduo e a cada coluna *n*, uma variável de projeto. No jargão dos algoritmos GA e PSO, os indivíduos são chamados de cromossomos e partículas (ou agentes), respectivamente. A simulação dos algoritmos é um processo iterativo, interrompido quando são satisfeitos critérios de parada, tais como: número máximo de iterações, valor mínimo da função *fitness*, entre outros.

O algoritmo genético com valores reais é bastante similar ao algoritmo genético com valores binários, mas faz uso de valores de ponto flutuante. Os cromossomos de um GA com valores reais é definido na *i*-ésima geração como em (4), em que, a matriz $\boldsymbol{P}(i)_{m,n}$ possui uma população N_{pop} cromossomos, cada um com N_{var} variáveis reais de otimização. Cada cromossomo é avaliado por meio de uma função custo associada, que é computada através da função *fitness* dada em (5).

$$\boldsymbol{P}(i)_{m,n} = \left[p^i_{m,1},\quad p^i_{m,2},\quad \ldots,\quad p^i_{m,N\,var}\right],\quad m = 1,2,\ldots,Npop \qquad n = 1,2,\ldots,N\,var \tag{4}$$

$$fitness(i,m) = \boldsymbol{E}\left(cromossomo\,(i,m)\right) \tag{5}$$

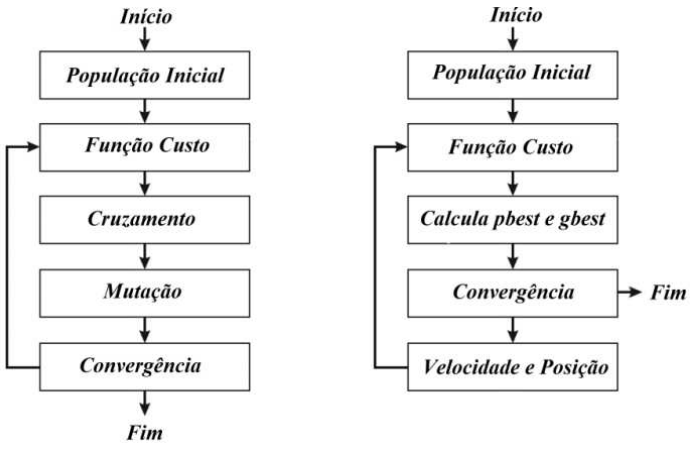
A evolução da população ocorre em função do custo associado, pela realização do cruzamento e da mutação, durante as iterações. O diagrama da Figura 2(a) resume o procedimento de otimização com o algoritmo genético. O cruzamento inclui o método de seleção *roulette wheel* apresentado por Haupt & Werner, 2007. A seleção da população é realizada após os N_{pop} cromossomos serem ordenados do menor para o maior custo. Então, os melhores cromossomos (N_{keep} mais aptos) são selecionados para cruzamento enquanto os demais são descartados. Os pais cruzam aleatoriamente pelo método *blending crossover* (HAUPT; HAUPT, 2004). Cada par produz dois filhos que contêm traços dos pais. Além disso, os pais sobrevivem para formar a próxima geração. Após o cruzamento, uma fração da população sofrerá mutação. Geralmente os usuários adicionam um número aleatório à variável do cromossomo selecionada para mutação com valores reais. Este valor aleatório possui distribuição normal de média zero e desvio padrão unitário, que é obtido pela função *randn*() do Matlab. (HAUPT; HAUPT, 2004)

A otimização por enxame de partículas foi formulada por Kennedy & Eberhart, 1995. O algoritmo do PSO foi inspirado no comportamento social de animais (bando de pássaros, enxame de peixes, etc.) dos quais emerge uma inteligência coletiva. A simulação do algoritmo PSO é semelhante ao GA desde que começa com uma população inicial aleatória. Contudo, o PSO não possui operadores típicos de algoritmos evolucionários, por exemplo, cruzamento e mutação. Nele, cada partícula move-se em torno da função custo com uma velocidade individual, buscando ajustar as velocidades e posições das partículas com base nas melhores soluções locais e global, de acordo com (6) e (7) (KENNEDY; EBERHART, 1995).

$$v_{m,n}^{i+1} = C\left[r_0 v_{m,n}^{i} + \Gamma_1 \cdot r_1 \cdot \left(p_{m,n}^{local\ bes(i)} - p_{m,n}^{i}\right) + \Gamma_2 \cdot r_2 \cdot \left(p_{m,n}^{global\ best(i)} - p_{m,n}^{i}\right) \right] \quad (6)$$

$$p_{m,n}^{i+1} = p_{m,n}^{i} + v_{m,n}^{i+1} \quad (7)$$

Em que, $v_{m,n}$ é a velocidade da partícula; $p_{m,n}$ é a posição da partícula nos espaço de busca das variáveis de otimização; r_0, r_1 e r_2 são números aleatórios independentes; Γ_1 é o parâmetro cognitivo e Γ_2 é o parâmetro social; $p_{m,n}^{localbest(i)}$ é a melhor solução local e $p_{m,n}^{globalbest(i)}$ é a solução global; C é o parâmetro de constrição (KENNEDY; EBERHART, 1995). Se a melhor solução local tem um custo menor que o custo da solução global, então a melhor solução local substitui a solução global. O PSO é um algoritmo bio-inspirado muito simples e, com poucos parâmetros, é de fácil implementação computacional. O diagrama mostrado na Figura 2(b) resume o procedimento do PSO.



(a) (b)

Figura 2 – Diagramas em blocos: (a) GA e (b) PSO.

Resultados e Discussão

Nesta seção são apresentados os resultados obtidos das aplicações dos algoritmos GA e PSO para projeto ótimo de filtros analógicos. Eles foram implementados no Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB) e funcionam para uma ampla região de interesse das variáveis de entrada. Nas aplicações didáticas descritas nesta seção, a convergência dos algoritmos GA e PSO depende do tamanho do espaço de busca e do número de indivíduos da população.

O primeiro exemplo consiste no projeto ótimo de um filtro passa-faixa tipo RLC série para aplicação num circuito equalizador gráfico de áudio de 12 faixas. A faixa de frequências de áudio até 24 KHz é dividida em sub-faixas de 2 KHz. Cada LED do equalizador acende conforme a potência do sinal em cada sub-faixa. As especificações do filtro passa-faixa para acionar o segundo LED são as frequências de corte inferior (2 KHz) e superior (4 KHz), que são obtidas de acordo com (8).

$$\omega_c = \frac{-RC \pm \sqrt{(RC)^2 + 4LC}}{2LC} \quad (8)$$

Nesse exemplo, os algoritmos GA e PSO são simulados com uma população de vinte indivíduos, N_{pop}=20. Considerou-se o resistor R=8 Ω; as variáveis otimizadas foram o capacitor e o

indutor, assim temos N_{var}=2 e $\boldsymbol{p}_m^i = [\,L_m^i, C_m^i\,]^T$. A função *fitness* escolhida foi a função quadrática dada em (9); f_{ci} e f_{cs} são dadas por (8), $f_{CI} = 2$ KHz e $f_{CS} = 4$ KHz são as especificações de projeto.

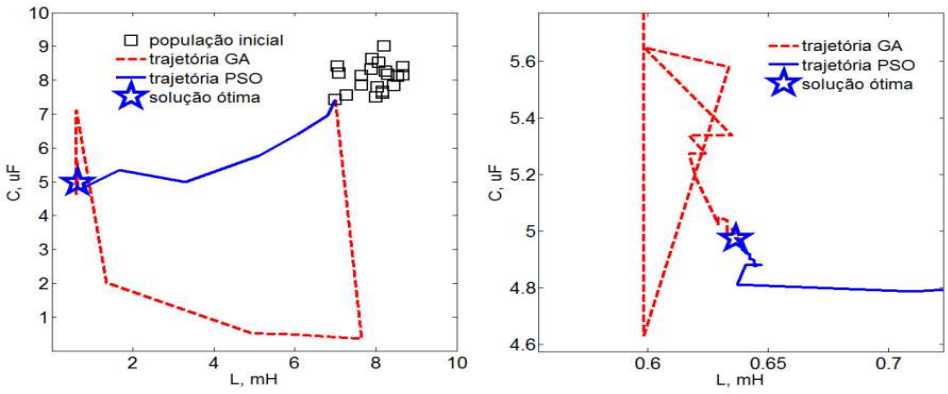
$$fitness = \frac{(f_{CI} - f_{ci})^2 + (f_{CS} - f_{cs})^2}{(f_{CS} - f_{CI})^2} \tag{9}$$

Os algoritmos foram simulados na região de busca: 1 µF≤ C ≤ 10 µF e 1 nH ≤ L ≤ 10 mH. As trajetórias GA e PSO podem ser vistas na Figura 3(a), a partir de uma mesma população inicial, dada por (10) com centro $(L_m^0, C_m^0) = (8mH, 8\mu F)$. Neste exemplo, o GA teve uma taxa de cruzamento e de mutação igual a 50% e o PSO teve os parâmetros $\Gamma_1 = \Gamma_2 = 2$ e $C = 1.3$. O critério de parada foi o número máximo de iterações. A trajetória GA em ziguezague, melhor observada nas vizinhanças do mínimo global (ver Figura 3(b)), resulta numa pior convergência conforme indicam os resultados da Figura 3(c). Por outro lado, destaca-se a robustez do algoritmo PSO em relação à convergência, cuja precisão é $\approx 10^{-32}$. Em termos de custo computacional, devido a sua simplicidade, o algoritmo PSO é mais eficiente que o GA. A Figura 4(b) apresenta a resposta em frequência obtida (magnitude, dB). Os pontos em -3 dB indicam as frequência de corte do filtro passa-faixa. Os valores ótimos obtidos, L= 0,637 mH e C=0,497 µF, atendem as especificações de projeto do filtro passa-faixa RLC série.

$$\boldsymbol{p}_m^0 = [randn()/5 + L_m^0 \quad randn()/5 + C_m^0] \tag{10}$$

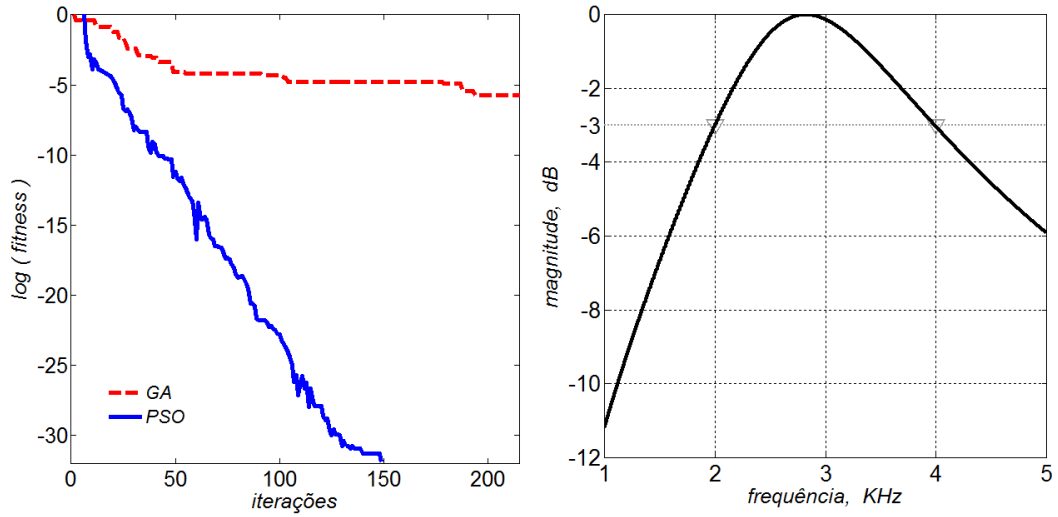
A segunda aplicação consiste no projeto de um filtro passa-faixa RLC paralelo para a frequência intermediária (455 KHz) de rádios AM. Considerando-se sinais de voz de até 5 KHz, resulta a seguinte especificação para este filtro: $f_{CI} = 450$ KHz e $f_{CS} = 460$ KHz. Os algoritmos foram simulados considerando-se para região de busca: 1 µF≤ C ≤ 1 µF e 1 nH ≤ L ≤ 1 mH. As frequências de corte foram obtidas através de (11). Já a população inicial é dada pela (10), com centro em $(L_m^0, C_m^0) = (0{,}8mH, 0{,}8\mu F)$. E tanto a função *fitness* como os demais parâmetros foram os mesmos do exemplo 1.

$$f_c = \frac{1}{2\pi}\left(\pm\frac{1}{2RC} + \sqrt{\left(\frac{1}{2RC}\right)^2 + \frac{1}{LC}}\right) \tag{11}$$



(a) (b)

Figura 3 – (a) População inicial e trajetórias GA e PSO; (b) vizinhança do mínimo global.



(a) (b)

Figura 4 – (a) Convergência dos algoritmos. (b) Resposta em frequência do filtro otimizado.

Os resultados obtidos para o segundo exemplo são apresentados na Figura 5. A Figura 5(a) mostra a convergência dos dois algoritmos. Já a Figura 5(b) mostra a população inicial e intermediária, trajetória e a solução ótima encontrada através do PSO. A população intermediária neste caso é típica da inteligência coletiva. A Figura 5(c) apresenta a magnitude (em dB) da resposta em frequência. Os pontos em -3 dB indicam as frequências de corte do filtro. Os valores ótimos obtidos, L= 1,538 µH e C=79,577 nF, atendendo as especificações de projeto do filtro RLC paralelo.

A terceira aplicação consiste no projeto de um filtro passa-baixa tipo Butterworth com uma topologia Sallen-key (veja Figura 1(c)). O filtro é projetado para a faixa de voz (0 a 20 KHz) com uma atenuação mínima de 40 dB para a frequências acima de 200 KHz. Nas simulações foi considerado para região de busca: 1 pF≤ $C1$, $C2 \leq 100$ nF. A magnitude da resposta em frequência (dB) do filtro é obtida através de (12). Nessa simulação foi utilizada uma população com 30 indivíduos centrada em $(C1_m^0, C2_m^0) = (80\text{ nF}, 80\text{ nF})$. A escolha da função *fitness* foi feita com base na comparação da resposta em frequência do filtro com as seguintes especificações de projeto: $|H(0)| = 0$ dB; $|H(20000)| = -3$ dB; e $|H(200000)| = -40$ dB. A função *fitness* utilizada é definida em (13), em que $|H(f)|$ é dada em (12). E os demais parâmetros permaneceram iguais ao exemplo 1.

$$|H(f)|_{(\text{dB})} = 20\log\left(\left|\frac{1}{(-4\pi^2 \cdot R1 \cdot C1 \cdot R2 \cdot C2 \cdot f^2 + 1 + j2\pi C1(R1+R2)f)}\right|\right) \quad (12)$$

$$fitness = (H(0))^2 + (H(20000) + 3)^2 + (H(200000) + 40)^2 \quad (13)$$

Os resultados obtidos para o filtro Sallen-key estão apresentados na Figura 6. A Figura 6(a) mostra a convergência dos algoritmos GA e PSO. A Figura 6(b) mostra a população inicial, trajetória, população intermediária e a solução ótima encontra através do algoritmo PSO. A Figura 6(c) apresenta a magnitude (em dB) da resposta em frequência obtida. As marcas no gráfico ilustrado na Figura 6(c) indicam as especificações de projeto do filtro passa-baixa. Os valores ótimos obtidos, C1= 5,627 nF e C2=11,254 nF, atendem as especificações de projeto do filtro passa-baixa Sallen-key.

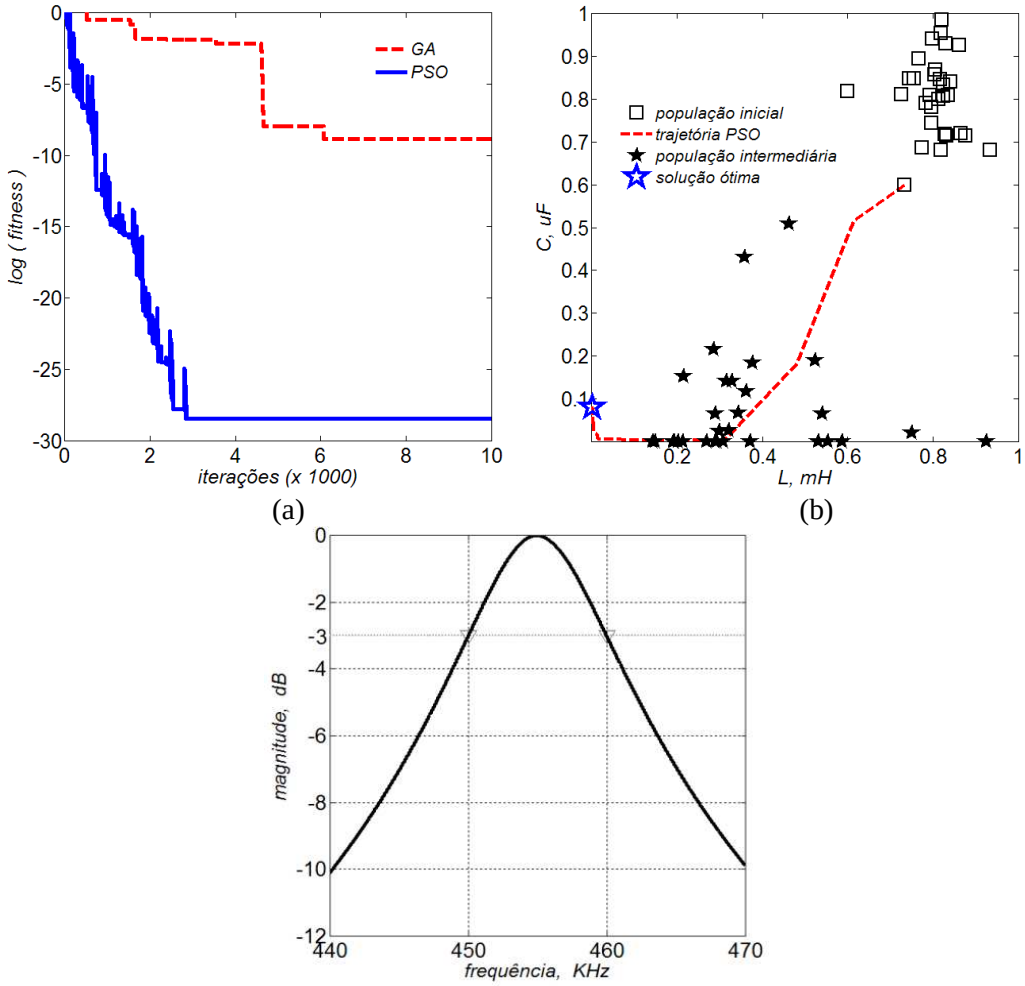
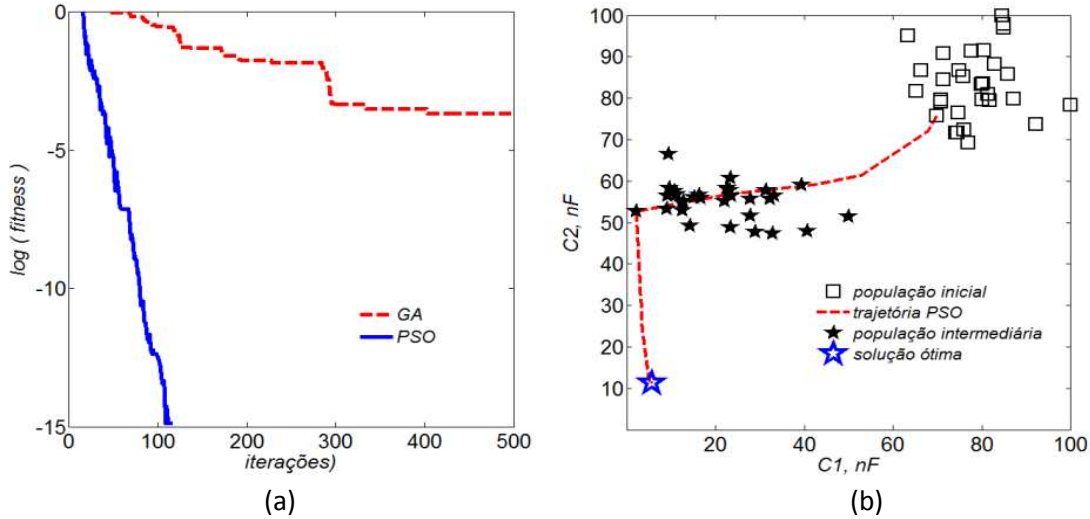
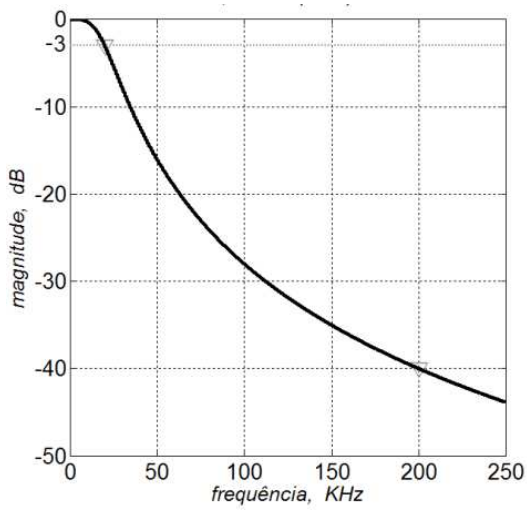


Figura 5 – (a) Convergência dos algoritmos. (b) Trajetória PSO. (c) Resposta do filtro RLC paralelo.





(c)

Figura 6 – (a) Convergência dos algoritmos. (b) Trajetória PSO. (c) Resposta do filtro Sallen-key.

Conclusões

As etapas de projeto de filtros analógicos foram automatizadas através do uso de algoritmos bio-inspirados: GA e PSO. O PSO é um algoritmo simples, com poucos parâmetros e de fácil implementação computacional. Nas aplicações didáticas abordadas, a convergência dos algoritmos GA e PSO depende, além dos seus parâmetros, do tamanho do espaço de busca e do número de indivíduos da população. A trajetória GA em ziguezague, melhor observada nas vizinhanças do mínimo global, resulta numa convergência mais lenta. Por outro lado, destaca-se a robustez do algoritmo PSO. Em termos de custo computacional, devido a sua simplicidade, o algoritmo PSO é mais eficiente que o GA. Limitado apenas pela precisão do Matlab ($\approx 10^{-32}$), o algoritmo PSO foi capaz de convergir para a solução ao encontrar o mínimo global do problema de otimização. A metodologia de projeto e otimização de filtros analógicos descrita neste artigo pode ser adaptada para outros dispositivos, circuitos e aplicações na área da Engenharia Elétrica.

Agradecimentos

Os autores agradecem ao Instituto Federal de Educação, Ciência e Tecnologia da Paraíba- IFPB. Este trabalho foi financiado pelo CNPq (IFPB/PIBITI 2011) sob o processo no. 23326.007586/2011-27.

Referências

ENGELBRECHT, A. P. **Coputacional Intelligence:** An Introcuction. 2. ed. New York: Wiley, 2007. 628 p.

HAUPT, R. L. An introduction to Genetic Algorithms for Electromagnetics. **IEEE Antennas propagation Magazine**, 37, n. 2, Abril 1995. 7-15.

HAUPT, R. L.; HAUPT, S. E. **Practical Genetic Algorithms**. 2. ed. New Jersey: Wiley, 2004.

HAUPT, R. L.; WERNER, D. H. **Genetic Algorithms in Electromagnetics**. 1. ed. Hoboken: Willey, 2007. 299 p.

KENNEDY, L.; EBERHART, R. C. **Particle Swarm Optimization**. Proceedings of the IEEE International Conference on Neural Network. Perth: [s.n.]. 1995. p. 1942-1948.

MUSSOI, F. L. R. **Resposta em Frequência:** Filtros Passivos. 2°. ed. Florianópolis: [s.n.], 2004. 85 p.