Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 8, n. 1, 2021.

Trabalho apresentado no XL CNMAC, Evento Virtual - Co-organizado pela Universidade do Mato Grosso do Sul (UFMS).

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Melhoria do desempenho da Hibridização GA-PSO através do Peso Inercial Adaptativo *Fuzzy*

Rodrigo Possidônio Noronha[1]
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, Imperatriz, MA

Resumo. A hibridização de algoritmos de otimização pertencentes à teoria da computação evolucionária é realizada com o objetivo de combinar características desejáveis e atenuar características indesejáveis dos métodos envolvidos. De forma específica para uma hibridização envolvendo o *Genetic Algorithm* (GA) e *Particle Swarm Optimization* (PSO), embora o GA insira bastante diversidade de posições no espaço de busca permitindo contornar com uma maior facilidade a convergência prematura, o processo de busca converge lentamente. Além disso, somente inserir diversidade de posições não é o suficiente para contornar a convergência prematura em problemas com maior complexidade. Para obter um processo de busca com uma convergência rápida e não prematura, é necessário realizar um bom *trade-off* entre a busca global e local. Nessa hibridização, embora não seja trivial, o *trade-off* pode ser realizado, por exemplo, através de uma boa seleção de valores para o peso inercial do PSO. Para isso, nesse artigo, é proposta a utilização de um sistema de inferência *fuzzy* Mamdani para adaptação paramétrica, a cada iteração, do peso inercial. Sendo assim, é possível melhorar o desempenho da hibridização GA-PSO através de um bom *trade-off* entre a busca global e local.

Palavras-chave. Computação Evolucionária, *Genetic Algorithm*, Hibridização, *Particle Swarm Optimization*, Sistema de Inferência *Fuzzy* Mamdani.

1 Introdução

Algoritmos de otimização pertencentes à teoria de computação evolucionária têm sido utilizados para resolver problemas complexos em várias áreas do conhecimento, de tal forma que, além de não necessitar que a função de custo seja contínua no espaço de busca, solucionam com uma maior facilidade problemas multimodais, não lineares, multidimensionais, e entre outros. Dentre a variedade de algoritmos, pode-se citar o GA e PSO, que têm sido amplamente utilizados devido ao robusto e eficiente desempenho na otimização de problemas complexos do mundo real [1]. O PSO, proposto por Kennedy J. e Eberhart R., é um método de otimização da teoria de inteligência de enxames bio-inspirando no comportamento individual e social de animais (pássaros, e entre outros) em bando, por exemplo, em busca de alimentos [2]. O processo de busca realizado pelo PSO converge rapidamente para a solução ótima em problemas unimodais. Porém, em problemas multimodais, devido a ausência de um mecanismo que insira diversidade de posições que torna o processo de busca unidirecional conforme descrito em [2], a convergência prematura pode vir a ocorrer. O GA, proposto por Holland J. H., é um método de otimização bio-inspirado na evolução genética de seres vivos [4]. Através da utilização dos operadores genéticos, a cada geração ou iteração, uma grande diversidade de posições é inserida no espaço de busca. Devido a isso, naturalmente, o GA apresenta uma maior facilidade em contornar a convergência prematura em

[1]rodrigo.noronha@ifma.edu.br

problemas multimodais, quando comparado ao PSO; porém, devido a grande diversidade inserida que torna o processo de busca multidirecional, a convergência do GA é lenta [3]. A convergência prematura do processo de busca realizado pelo PSO e a lenta convergência do processo de busca realizado pelo GA são resultados de um ineficiente *trade-off* entre a busca global e local.

A partir da década de 1980, a hibridização de algoritmos de otimização pertencentes à teoria de computação evolucionária tem sido recorrente, uma vez que é possível combinar as características desejáveis e atenuar as características indesejáveis dos algoritmos envolvidos. Originalmente, a hibridização GA-PSO é realizada com o objetivo de obter um algoritmo de otimização que tenha a capacidade de otimizar problemas multimodais com uma convergência rápida e com grande diversidade. Existem algumas contribuições à hibridização GA-PSO publicadas na literatura, como em [5] que a proposta é baseada em codificação paralela bidimensional com o objetivo de otimizar os recursos para bloqueio de *Jammers*. Em [8], a hibridização GA-PSO foi proposta para com o objetivo de diagnosticar falhas em rolamentos mecânicos. Em [6], os parâmetros de controladores *fuzzy* tipo 1 e tipo 2 foram sintonizados pela hibridização GA-PSO. É importante notar que devido a grande diversidade de posições no espaço de busca inserida pelo GA, o processo de busca realizado pela hibridização converge lentamente. Além disso, embora o processo de busca realizado hibridização tenha uma maior facilidade em contornar a convergência prematura, apenas inserir diversidade no espaço de busca pode não ser o suficiente. Em função da complexidade do problema (por exemplo, quantidade de modos, dimensão do espaço de busca, não linearidade, e entre outros), a convergência prematura pode ocorrer devido ao excesso ou insuficiência de diversidade. Uma vez que a diversidade é uma medida de distância entre as partículas no espaço de busca, através de um bom *trade-off* entre a busca global e local é possível evitar a convergência prematura e, além disso, realizar um processo de busca que convirja rapidamente. Na hibridização GA-PSO, embora não seja trivial, um bom *trade-off* entre a busca global e local pode ser realizado, por exemplo, através de uma boa seleção de valores para o peso inercial [7].

Neste artigo, é proposto a utilização de um sistema de inferência *fuzzy* Mamdani (do inglês, MFIS) visando, a cada iteração, realizar a adaptação paramétrica do peso inercial. A justificativa para descrever o comportamento dinâmico do peso inercial em um contexto *fuzzy*, é devido a capacidade de um MFIS representar o conhecimento subjetivo em uma base de regras; dessa forma, o comportamento dinâmico do peso inercial é descrito em função do conhecimento do especialista sobre como deve ser realizado o *trade-off*. Portanto, essa é a motivação para essa proposta, que é clara e justificada. As entradas do MFIS são a diversidade e a iteração normalizadas pelo método Min-Max; a saída é o peso inercial. Dessa forma, o peso inercial é adaptado parametricamente em função de informações sobre a distância entre as partículas e sobre o avanço temporal do processo de busca. Esse artigo é organizado da seguinte forma: na Seção 2 são apresentadas as declarações do problema que a metodologia de otimização proposta vista solucionar; na Seção 3, é apresentada a metodologia de otimização proposta, que é organizada em cinco subseções; na Seção 4, são apresentados os resultados computacionais obtidos através da otimização de *benchmarks*.

2 Declarações do Problema

O objetivo da metodologia de otimização proposta é realizar a busca por soluções que maximize ou minimize uma função de custo, formulado como:

$$\begin{cases} J = J(x_{i,1}[k], x_{i,2}[k], \ldots, x_{i,n}[k]) \\ sujeito\ a: \\ min(\mathbf{x}_i) \leq \mathbf{x}_i[k] \leq max(\mathbf{x}_i) \text{ e } min(\mathbf{v}_i) \leq \mathbf{v}_i[k] \leq max(\mathbf{v}_i) \\ k \in [1, K] \end{cases} \tag{1}$$

DOI: 10.5540/03.2021.008.01.0432

onde, $J : \mathbb{R}^n \to \mathbb{R}$ é a função de custo, $\mathbf{v}_i[k] = [v_{i,1}[k], v_{i,2}[k], \ldots, v_{i,n}[k]] \in \mathbb{R}^{1\times n}$ e $\mathbf{x}_i[k] = [x_{i,1}[k], x_{i,2}[k], \ldots, x_{i,n}[k]] \in \mathbb{R}^{1\times n}$ são, respectivamente, os vetores de velocidade e posição da i-ésima partícula pertencente a uma população de N partículas. Para um problema de minimização, a atualização do vetor de melhor posição individual $\mathbf{p}_i[k]$ da i-ésima partícula, é dado por:

$$\mathbf{p}_i[k+1] = \begin{cases} \mathbf{p}_i[k] \text{ se } J(\mathbf{x}_i[k+1]) \geq J(\mathbf{p}_i[k]) \\ \mathbf{x}_i[k+1] \text{ se } J(\mathbf{x}_i[k+1]) < J(\mathbf{p}_i[k]) \end{cases} \tag{2}$$

para um problema de maximização, a atualização é dada por:

$$\mathbf{p}_i[k+1] = \begin{cases} \mathbf{p}_i[k] \text{ se } J(\mathbf{x}_i[k+1] \leq J(\mathbf{p}_i[k]) \\ \mathbf{x}_i[k+1] \text{ se } J(\mathbf{x}_i[k+1]) > J(\mathbf{p}_i[k]) \end{cases} \tag{3}$$

Para um problema de minimização, a atualização do vetor de melhor posição global $\mathbf{p}_g[k]$, é dado por:

$$\mathbf{p}_g[k+1] = \arg\min\{\mathbf{p}_1[k+1], \mathbf{p}_2[k+1], \ldots, \mathbf{p}_N[k+1]\} \tag{4}$$

para um problema de maximização, a atualização é dada por:

$$\mathbf{p}_g[k+1] = \arg\max\{\mathbf{p}_1[k+1], \mathbf{p}_2[k+1], \ldots, \mathbf{p}_N[k+1]\} \tag{5}$$

3 Metodologia de Otimização Proposta

Nesta seção, é realizada a descrição da metodologia de otimização proposta. A cada iteração k, o vetor posição $\mathbf{x}_i[k]$ é atualizado pelos operadores genéticos de seleção de pais, *crossover* e mutação. Logo após, os novos vetores posição são avaliados pela função de custo e são atualizadas os vetores das melhores posições individual $\mathbf{p}_i[k]$ e o vetor da melhor posição global $\mathbf{p}_g[k]$. Logo em seguida, são calculadas a diversidade normalizada e a normalização da iteração pelo método Min-Max. Após feito isso, é executado o MFIS, no qual a saída é o peso inercial adaptado. Após realizada a adaptação do peso inercial, são atualizados os vetores de velocidade e posição, $\mathbf{v}_i[k]$ e $\mathbf{x}_i[k]$. Por fim, são atualizados os vetores de melhor posição individual $\mathbf{p}_i[k]$ e de melhor posição global $\mathbf{p}_g[k]$. Se a condição de parada for satisfeita, então é obtida a solução para o problema; caso contrário, a busca pela solução ótima é continuada.

3.1 Operador Genético de Seleção de Pais

Através do operador genético de seleção de pais, é possível selecionar as partículas com melhor custo, para serem reproduzidas pelo operador genético *crossover*, de acordo com a taxa de *crossover* μ_c. O método de seleção de pais utilizado nessa metodologia é do tipo torneio, que seleciona, aleatoriamente, m partículas da população para participarem do torneio, no qual a partícula com melhor custo será a partícula pai da próxima geração, conforme a pseudocódigo a seguir:

Algorithm 1 Pseudocódigo do Método Torneio

```
Definir a taxa de crossover μ_c;
for g = 1 até Nμ_c do
    for l = 1 até 2 do
        Escolher aleatoriamente m partículas;
        pai_l = a partícula com melhor custo;
    end for
end for
```


3.2 Operador Genético de *Crossover*

Após realizada a seleção das partículas com melhores custos, é realizado o *crossover*. O operador genético de *crossover* é utilizado para combinar as características dos pais selecionados, visando obter partículas mais aptas a serem possíveis soluções para o problema. Valores típicos de μ_c estão no intervalo $[0,5\ 1]$. O método de *crossover* utilizado nessa metodologia é do tipo uniforme, conforme o pseudocódigo a seguir:

Algorithm 2 Pseudocódigo do Método de *Crossover* Uniforme

1: **for** $g = 1$ até $N\mu_c/2$ **do**
2: Escolher duas partículas aleatoriamente;
3: Obter aleatoriamente $\alpha \in [0, 1]$;
4: $filho_1 = \alpha pai_1 + (1-\alpha)pai_2$ e $filho_2 = \alpha pai_2 + (1-\alpha)pai_1$;
5: **end for**

3.3 Operador Genético de Mutação

O próximo passo após a realização do *crossover* é realizar a mutação dos genes das partículas de uma população. Através do operador genético de mutação, é possível explorar novas regiões no espaço de busca. A mutação modifica, com uma taxa μ_m, alguma característica da partícula. Sendo assim, o uso do operador genético de mutação é importante para inserir diversidade de posições no espaço de busca. Valores típicos de μ_m estão no intervalo $[0,005\ 0,05]$. O método de mutação utilizado nessa metodologia é do tipo aleatório, conforme pseudocódigo a seguir:

Algorithm 3 Pseudocódigo do Método de Mutação Aleatória

1: **for** $i = 1$ até N **do**
2: Obter aleatoriamente $\beta \in [0, 1]$;
3: **if** $\beta < \mu_m$ **then**
4: Obter aleatoriamente $\gamma \in \{1, 2, \ldots, n\}$;
5: Realizar a mutação do gene: $x_{i,\gamma}[k] = max(x_{i,\gamma}) + \beta(max(x_{i,\gamma}) - min(x_{i,\gamma}))$;
6: **end if**
7: **end for**

3.4 Peso Inercial Adaptativo *Fuzzy*

Após executados os operadores genéticos,, o próximo passo é a atualização do peso inercial através de um MFIS. A adaptação do peso inercial é descrita a seguir:

$$\begin{gathered}\omega[k] = MFIS(\mathcal{D}[k], \mathcal{K}[k]) \\ \mathcal{D}[k] = \frac{D[k] - d_{min}[k]}{d_{max}[k] - d_{min}[k]} \text{ e } \mathcal{K}[k] = \frac{k-1}{K-1}\end{gathered} \tag{6}$$

em que, $D[k]$ é a medida de diversidade dada pela distância Euclidiana média entre o vetor de posição $x_i[k]$ e o vetor de melhor posição global $p_g[k]$. As variáveis $d_{min}[k]$ e $d_{max}[k]$ são, respectivamente, o menor e maior valor obtidos para diversidade até a iteração k. As variáveis $\mathcal{D}[k]$ e $\mathcal{K}[k]$ são, respectivamente, a diversidade e iteração normalizadas pelo método Min-Max. As entradas do MFIS, que são variáveis linguísticas, são a diversidade normalizada e a iteração normalizada. A cada iteração k, através da fuzzificação, a j-ésima função de pertinência (do inglês, MBF) realiza o mapeamento dos universos de discurso das variáveis linguísticas de entrada do MFIS para o intervalo $[0, 1]$ de grau de pertinência, ou seja, $m_j(\mathcal{K}[k]) : \mathbb{R} \to [0, 1]$ e $m_j(\mathcal{D}[k]) : \mathbb{R} \to [0, 1]$, para

$j = 1, 2, 3$. As MBFs são definidas em função do conhecimento do especialista sobre como deve ser realizado o *trade-off* entre a busca global e local. Para cada variável linguística de entrada do MFIS, foram definidas três MBFs do tipo triangular, com os valores linguísticos "pequeno"(P), "médio"(M) e "grande"(G), cujo os parâmetros podem ser vistos na Tabela 1. As variáveis linguísticas são associadas aos valores linguísticos por meio de proposições *fuzzy*, que são conectadas através de conectivos lógicos no antecedente e consequente, formando regras *fuzzy* conforme o seguinte exemplo $\boldsymbol{\mathcal{R}}^i$: **Se** $\mathcal{K}$ é P **e** $\mathcal{D}$ é P **então** ω^i é G. Na base de regras *fuzzy* desenvolvida, conforme pode ser visto em (7), as proposições *fuzzy* do antecedente são conectadas por conectivos lógicos "**e**", que realiza a seguinte operação: $t[m_j(\mathcal{K}[k]), m_j(\mathcal{D}[k])] = min(m_j(\mathcal{K}[k]), m_j(\mathcal{D}[k]))$, em que $t[\bullet]$ representa a norma-t. Após o cálculo da norma-t, é obtido o grau de pertinência de ativação da i-ésima regra *fuzzy*, que seleciona o valor mínimo entre dois graus de pertinência.

$$
\begin{array}{ll}
\boldsymbol{\mathcal{R}}^1: \textbf{ Se } \mathcal{K} \text{ é P } \textbf{e } \mathcal{D} \text{ é P } \textbf{então } \omega^1 \text{ é G} & \boldsymbol{\mathcal{R}}^2: \textbf{ Se } \mathcal{K} \text{ é P } \textbf{e } \mathcal{D} \text{ é M } \textbf{então } \omega^2 \text{ é M} \\
\boldsymbol{\mathcal{R}}^3: \textbf{ Se } \mathcal{K} \text{ é P } \textbf{e } \mathcal{D} \text{ é G } \textbf{então } \omega^5 \text{ é P} & \boldsymbol{\mathcal{R}}^4: \textbf{ Se } \mathcal{K} \text{ é M } \textbf{e } \mathcal{D} \text{ é P } \textbf{então } \omega^4 \text{ é M} \\
\boldsymbol{\mathcal{R}}^5: \textbf{ Se } \mathcal{K} \text{ é M } \textbf{e } \mathcal{D} \text{ é M } \textbf{então } \omega^5 \text{ é M} & \boldsymbol{\mathcal{R}}^6: \textbf{ Se } \mathcal{K} \text{ é M } \textbf{e } \mathcal{D} \text{ é G } \textbf{então } \omega^6 \text{ é M} \\
\boldsymbol{\mathcal{R}}^7: \textbf{ Se } \mathcal{K} \text{ é G } \textbf{e } \mathcal{D} \text{ é P } \textbf{então } \omega^7 \text{ é M} & \boldsymbol{\mathcal{R}}^8: \textbf{ Se } \mathcal{K} \text{ é G } \textbf{e } \mathcal{D} \text{ é M } \textbf{então } \omega^8 \text{ é P} \\
\boldsymbol{\mathcal{R}}^9: \textbf{ Se } \mathcal{K} \text{ é G } \textbf{e } \mathcal{D} \text{ é G } \textbf{então } \omega^9 \text{ é P} &
\end{array}
\tag{7}
$$

Após obtido o grau de pertinência de ativação da i-ésima regra *fuzzy*, em seguida é calculada a implicação *fuzzy* do consequente. A entrada da implicação *fuzzy* é o grau de pertinência de ativação da i-ésima regra *fuzzy*, e a saída é uma MBF obtido através do operador mínimo (min). Cada regra *fuzzy* é ativada com um determinado grau de pertinência, sendo assim, as MBFs de saída precisam ser combinados para que o MFIS possa ter uma resposta total. Através da agregação, todas as MBFs de saída são combinados para se obter uma única MBF. Após realizada a agregação, é necessário obter um valor numérico (crisp) de saída. Para isso, é realizada a defuzzificação do MBF de saída obtido pela agregação. Nessa metodologia, o método de defuzzificação utilizado é do tipo centroide, conforme a seguir: $\omega[k] = \sum_{i=1}^{9} \omega^i[k] m_j(\omega^i[k]) (\sum_{i=1}^{9} m_j(\omega^i[k]))^{-1}$ no qual, para a variável linguística de saída do MFIS, foram definidas três MBFs do tipo triangular, com os valores linguísticos "pequeno"(P), "médio (M) e "grande"(G), cujo os parâmetros podem ser vistos na Tabela 1.

Tabela 1: Intervalos paramétricos das MBFs triangulares do antecedente e consequente.

$\mathcal{K}[k]$		$\mathcal{D}[k]$		$\omega^i[k]$	
Valor Linguístico	Intervalo	Valor Linguístico	Intervalo	Valor Linguístico	Intervalo
Pequeno	[0 0 0,1]	Pequeno	[0 0 0,5]	Pequeno	[0,001 0,09 0,15]
Médio	[0 0,1 0,3]	Médio	[0 0,5 1]	Médio	[0,09 0,5 0,35]
Grande	[0,1 0,3 1,0]	Grande	[0,5 1,0 1,0]	Grande	[0,15 0,35 0,9]

3.5 Atualização dos Vetores de Velocidade e Posição das Partículas

Após a adaptação *fuzzy* do peso inercial, o próximo passo é a atualização dos vetores de velocidade e posição. O vetor velocidade $\mathbf{v}_i[k]$ tem a função de controlar a atualização do vetor de posição $\mathbf{x}_i[k]$. A equação de atualização do vetor velocidade é composta por dois termos que introduzem a natureza estocástica ao processo de busca $r_1, r_2 \in [0,\ 1]$, dos coeficientes de aceleração $C_1, C_2 \in [0,\ 2]$ e do peso inercial $\omega \in [0,\ 1,2]$. As equações de atualização do vetor velocidade e posição da i-ésima partícula, são dadas, respectivamente, por:

$$
\begin{aligned}
v_{i,j}[k+1] &= v_{i,j}[k]\omega + C_1 r_1 (p_{i,j}[k] - x_{i,j}[k]) + C_2 r_2 (p_{g,j}[k] - x_{i,j}[k]) \\
\mathbf{x}_i[k+1] &= \mathbf{x}_i[k] + \mathbf{v}_i[k+1]
\end{aligned}
\tag{8}
$$

4 Resultados Computacionais

Para validar a metodologia de otimização proposta, nessa seção é realizada a minimização de dois *benchmarks*, que são a função *Rastrigin* ($f_1[\mathbf{x}]$) e função *Griewank* ($f_2[\mathbf{x}]$), descritas a seguir:

$$f_1(\mathbf{x}[k]) = 10K + \sum_{k=1}^{K}\left(\mathbf{x}[k]^2 - 10cos2\pi\mathbf{x}[k]\right) \quad f_2(\mathbf{x}[k]) = 1 + \frac{1}{4000}\sum_{k=1}^{K}\mathbf{x}[k]^2 - \prod_{k=1}^{K} cos\frac{\mathbf{x}[k]}{\sqrt{k}} \tag{9}$$

Os resultados apresentados na Tabela 2, foram obtidos pela definição dos seguintes parâmetros: taxa de *crossover* $\mu_c = 0,9$, taxa de mutação $\mu_m = 0,1$, coeficiente de aceleração $C_1 = C_2 = 2$ e o peso inercial $\omega = 1$ (para o PSO tradicional). O parâmetro m utilizado para selecionar as partículas pais pelo método do torneio foi definido como $m = 3$. Os vetores de posição inicial $\mathbf{x}_i[1]$ foram inicializados de forma aleatória e sendo iguais para todos os algoritmos. Os vetores de velocidade e de melhores posições individuais foram inicializados como $\mathbf{p}_i[1] = \mathbf{x}_i[1]$ e $\mathbf{v}_i[1] = \mathbf{0}$. O tamanho da população foi definida como $N = 100$ partículas e foram avaliadas durante $K = 300$ iterações. Uma vez que todos os algoritmos avaliados na Tabela 2 são estocásticos, então, o processo de busca, para cada algoritmo, foi realizado 100 vezes com o objetivo de obter uma maior coerência estatística dos resultados obtidos. A dimensão do espaço de busca foi definida como $n = 20$. Devido aos algoritmos terem desenvolvido os piores resultados na otimização do *benchmark* $f_1(\mathbf{x}[k])$, assim a análise do desempenho será realizada apenas nesse *benchmark*; é importante notar que o mínimo global do *benchmark* $f_1(\mathbf{x}[k])$ é igual a zero e o espaço de busca é limitado ao intervalo $[-5,\ 10]$.

Tabela 2: Resultados obtidos para 100 realizações do processo de busca.

Algoritmo	**Resultados**	$f_1(\mathbf{x}[k])$	$f_2(\mathbf{x}[k])$
PSO	**Custo Médio**	$163,0306$	$0,0017$
	Desvio Padrão	$162,1489$	$0,0028$
GA	**Custo Médio**	$8,0351$	$0,0069$
	Desvio Padrão	$8,1492$	$0,0032$
GA-PSO	**Custo Médio**	$3,9898$	0
	Desvio Padrão	$4,9871$	0
Proposto	**Custo Médio**	$0,5912$	0
	Desvio Padrão	$0,5601$	0

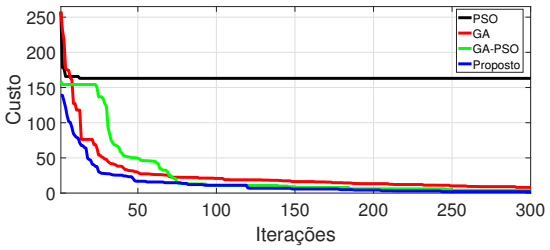

Figura 1: Custo para o *benchmark* $f_1(\mathbf{x}[k])$.

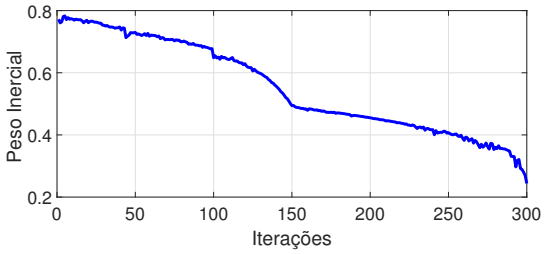

Figura 2: Peso inercial do *benchmark* $f_1(\mathbf{x}[k])$.

Através da Figura 1, nota-se que o PSO convergiu prematuramente, uma vez que o *benchmark* $f_1(\mathbf{x}[k])$ é multimodal; esse desempenho insatisfatório pode ser confirmado através da Tabela 2. Além do PSO, na Figura 1 são mostrados os resultados obtidos pelo algoritmo GA-PSO (sem a adaptação *fuzzy* do peso inercial) e pelo GA. Nota-se que o GA convergiu mais lentamente do que o GA-PSO, que é devido ao processo de busca ser realizado unicamente de forma multidirecional; o

GA-PSO obteve uma convergência mais rápida do que o GA, que é devido ao processo de busca ser realizado tanto de forma multidirecional como unidirecional. Já em relação ao algoritmo proposto, que é o GA-PSO com o peso inercial adaptativo *fuzzy*, nota-se que, além de evitar a convergência prematura, o algoritmo obteve a convergência mais rápida e os melhores resultados de otimização, de acordo com a Tabela 2. A dinâmica de adaptação descrita por um MFIS, permitiu que o peso inercial, durante as primeiras iterações, visando uma busca global, tivesse um valor elevado; ao fim das iterações, visando uma busca local e a convergência, o seu valor foi decrescido, de acordo com a Figura 2. Dessa forma, foi possível obter um bom *trade-off* entre a busca global e local.

5 Conclusão

Nesse artigo, foi proposta um nova versão da hibridização GA-PSO com o peso inercial adaptativo *fuzzy*, visando uma melhoria no desempenho do algoritmo através de um bom *trade-off* entre a busca global e local. Nota-se que a adaptação paramétrica realizada pelo MFIS realizou o decrescimento do valor do peso inercial, indo desde uma busca global e, consequentemente realizando uma exploração do espaço de busca, até uma busca local e, consequentemente realizando a convergência. Porém, o decrescimento não foi realizado de forma linear, que é um resultado coerente, uma vez que o processo de busca, que é estocástico, não permite que as partículas desenvolvam uma comportamento linear. O comportamento não linear do valor do peso inercial permitiu que, em função de informações sobre as distâncias entre as partículas e do avanço temporal do processo de busca, quando presas em ótimos locais, as partículas saltassem para melhores posições; assim evitando uma convergência prematura e desenvolvendo uma rápida convergência.

Referências

[1] Ding, Y., Zhang, W., Yu, L., Lu, K. The accuracy and efficiency of GA and PSO optimization schemes on estimating reaction kinetic parameters of biomass pyrolysis, *Energy*, 176:582–588, 2019. DOI: 10.1016/j.energy.2019.04.030.

[2] Eberhart, R. C., Kennedy, J. Particle swarm optimization, *Proceedings of the IEEE international conference on neural networks*, 1995. DOI: 10.1109/MHS.1995.494215.

[3] Gao, W. An improved fast-convergent genetic algorithm, *EEE International Conference on Robotics, Intelligent Systems and Signal Processing*, 2003. DOI: 10.1109/RISSP.2003.1285761.

[4] Holland, J. H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[5] Liu, F., Wang, Y., Chen, J., Wang, Q., Yuan, N. Research on Jamming Resource Allocation Technology Based on Improved GAPSO Algorithm, *Journal of Physics: Conference Series*, 1738:012075, 2021.

[6] Martínez-Soto, R., Castillo, O., Aguilar, L. T., Rodriguez, A. A hybrid optimization method with PSO and GA to automatically design Type-1 and Type-2 fuzzy logic controllers, *International Journal of Machine Learning and Cybernetics*, 6:175–196, 2013.

[7] Shi, Y., Eberhart, R. C. Parameter selection in particle swarm optimization, *International Conference on Evolutionary Programming*, 591 – 600, 1998. DOI: 10.1007/BFb0040810.

[8] Zhang, X., Zhang, W., Guo, Q., Lei, W. Optimization of HMM Based on Adaptive GAPSO and Its Application in Fault Diagnosis of Rolling Bearing, *5th International Conference on Control and Robotics Engineering (ICCRE)*, 53–57, 2020.

