X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

ABORDAGENS HÍBRIDAS BASEADAS EM BUSCA POR CARDUMES E OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS PARA OTIMIZAÇÃO EM AMBIENTES DINÂMICOS

George M. Cavalcanti Júnior, Carmelo J. A. Bastos Filho, Fernando B. Lima Neto
Escola Politécnica de Pernambuco - Universidade de Pernambuco
{gmcj, carmelofilho, fbln}@ecomp.poli.br

Resumo – Algoritmos de inteligência de enxames vêm sendo bastante empregados para solução de problemas complexos de otimização. No entanto, alguns deles, como Otimização por Enxame de Partículas (PSO, *Particle Swarm Optimization*), não possuem a capacidade de gerar diversidade após mudanças no ambiente. Isto pode ser percebido principalmente quando esses algoritmos são aplicados em problemas dinâmicos. O algoritmo de Busca por Cardumes (FSS, *Fish School Search*) tem um operador que auto-regula a granularidade do processo de busca e pode gerar diversidade de forma auto-adaptativa. Este artigo apresenta uma abordagem híbrida para inserir o operador volitivo do FSS para gerar diversidade em variações do PSO e mostra que a hibridização melhora o desempenho do PSO e do PSO heterogêneo para problemas dinâmicos. Foram realizadas comparações com abordagens previamente propostas para minimizar este problema do PSO.

Palavras-chave – Algoritmos de otimização, Algoritmos híbridos, ambientes dinâmicos, Otimização por enxame de partículas, Otimização por enxame de partículas heterogênea, Busca por cardumes.

Abstract – Swarm intelligence algorithms have been widely applied to solve optimization problems. However, some of them, such as Particle Swarm Optimization (PSO), may not present the capacity to generate diversity after environmental changes. It is a common situation in dynamic problems. We demonstrate that we can hybridize the Fish School Search (FSS) and the PSO. By doing this, we can obtain an effective solution for dynamic problems. We show that the FSS volitive operator applied to generate diversity in the PSO and the Heterogeneous PSO approaches outperforms the original approaches with reinitialization mechanisms.

Keywords – Optimization algorithms, hybrid algorithms, dynamic environments, Particle Swarm Optimization, Heterogeneous Particle Swarm Optimization, Fish School Search.

1. INTRODUÇÃO

Em problemas de otimização do mundo real, as soluções ótimas podem variar ao longo do tempo. Algoritmos de otimização para aplicados a tais problemas devem ser capazes de se adaptar a ambientes dinâmicos, nos quais os ótimos podem mudar ao longo do tempo.

Vários algoritmos de otimização bioinspirados foram propostos nas duas últimas décadas. Dentre eles, podem ser citados os algoritmos de inteligência de enxames, que são inspirados por comportamentos coletivos. Em geral, algoritmos de enxames são baseados em grupos de animais, como bando de pássaros, cardumes, colônia de formigas, etc. Embora vários desses algoritmos tenham sido propostos, poucos deles foram desenvolvidos para tratar problemas dinâmicos.

Um dos algoritmos de inteligência de enxames mais utilizados é o de Otimização por Enxame de Partículas (PSO, *Particle Swarm Optimization*). Apesar da rápida velocidade de convergência, a versão original do PSO não apresenta bom desempenho em ambientes dinâmicos. Isso ocorre porque todo o enxame tende a se concentrar em uma boa região do espaço de busca, reduzindo a diversidade global da população. Dado que este problema foi diagnosticado, algumas alternativas foram propostas para aumentar a diversidade do enxame e a capacidade de escapar de sub-regiões do espaço de busca, nas quais o ótimo não está mais localizado [1, 2].

Por outro lado, outro algoritmo de inteligência de enxames proposto em 2008 [3], Busca por cardumes (FSS, *Fish School Search*), apresenta uma característica interessante que se mostrou bastante útil para ambientes dinâmicos [4]. O FSS possui um operador chamado coletivo-volitivo, que pode auto-regular a habilidade de busca em amplitude e busca em profundidade.

Com o intuito de aliar a rápida convergência do PSO com a capacidade de auto-regulação da granularidade de busca do FSS, Cavalcanti-Júnior e colaboradores propuseram um algoritmo híbrido para otimização em ambientes dinâmicos, o PSO Volitivo [4].

Engelbrecht propôs o PSO heterogêneo (HPSO, *Heterogeneous PSO*) [5], no qual as partículas podem mudar de comportamento durante o processo de busca. Em seguida, Leonard *et al.* [6] propuseram aplicar o HPSO em ambientes dinâmicos. No entanto, a alternativa empregada nesse caso para gerar diversidade após a mudança no ambiente foi a de simplesmente reini-

cializar a metade das partículas. Reinicializar as partículas tem como consequência a perda de informações obtidas durante o processo de busca.

Neste artigo, propõe-se usar o operador volitivo no HPSO como mecanismo para gerar diversidade. O algoritmo foi chamado de HPSO Volitivo. Além disso, é objetivo deste artigo comparar a utilização do operador coletivo-volitivo para gerar diversidade com a técnica comum de reinicialização das partículas.

Este artigo está estruturado da seguinte forma. Na Seção 2 está apresentada a fundamentação teórica, onde são apresentados conceitos dos algoritmos PSO, HPSO e FSS. O detalhamento de como o operador coletivo-volitivo é incorporado no PSO ou no HPSO está mostrado na seção 3. A seção 4 mostra o arranjo de simulaçáo, que contém as configurações dos algoritmos, a descrição dos ambientes de testes e as métricas empregadas para avaliação. A seção 5 apresenta os resultados, divididos em duas subseções: análise paramétrica dos algoritmos com operador volitivo e comparação dessas abordagens com as demais técnicas. Por fim, a seção 6 apresenta as conclusões e os trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Otimização por Enxame de Partículas (PSO)

PSO é uma técnica baseada em populações amplamente aplicada a problemas de otimização, proposta em 1995. A abordagem padrão é composta por um conjunto de partículas, as quais possuem individualmente suas posições no espaço de busca $\vec{x}_i$ e cada posição representa uma solução para o problema. As partículas se movem no espaço de busca procurando pela melhor solucão, de acordo com a sua velocidade atual $\vec{v}_i$. A melhor posicão encontrada por determinada partícula é armazenada no vetor $\vec{P}_{best_i}$ e a melhor posicão encontrada por todo o enxame até o presente instante de tempo é armazenada no vetor $\vec{G}_{best}$. De acordo com a abordagem proposta por Shi e Eberhart [7], conhecida como *inertia* PSO, a velocidade de uma partícula i pode ser avaliada a cada iteração do algoritmo utilizando a seguinte equação:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + r_1c_1[\vec{P}_{best_i} - \vec{x}_i(t)] + r_2c_2[\vec{G}_{best} - \vec{x}_i(t)], \tag{1}$$

onde r_1 e r_2 são números gerados aleatoriamente no intervalo $[0, 1]$. O fator de inércia (w) controla a influência da velocidade anterior e equilibra os comportamentos de busca em amplitude e profundidade ao longo do processo. c_1 e c_2 são denominados coeficientes de aceleração cognitivo e social, respectivamente, e ponderam a influência da memória da partícula e a informação adquirida pela vizinhança. A posição de cada partícula é atualizada baseada em sua velocidade de acordo com a seguinte equação:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1). \tag{2}$$

A topologia de comunicação define a vizinhança da partícula e, consequentemente, o fluxo de informação entre elas. Existem duas topologias básicas: global e local. Na primeira, cada partícula compartilha e recebe informações diretamente de todas as outras, ou seja, todas as partículas compartilham a mesma memória social, chamada $\vec{G}_{best}$. Na segunda topologia, cada partícula compartilha informações somente com duas vizinhas e a memória social não é a mesma para todo o enxame. Essa segunda abordagem, chamada de $\vec{L}_{best}$, ajuda a evitar a atração prematura de todas as partículas para uma única região do espaço de busca.

2.2. PSO heterogêneo dinâmico (dHPSO)

O dHPSO é uma variação de PSO proposta por Engelbrecht [5], na qual as partículas podem ter comportamentos diferentes umas das outras. Em outras palavras, as partículas podem usar regras diferentes para atualização de posição e velocidade. Leonard e colaboradores propuseram a utilização do dHPSO para problemas dinâmicos em [6]. Existem diversas formas de PSO heterogêneo, como por exemplo: *Charged* PSO [8], PSO predador-presa [9], PSO por divisão de trabalho [10], entre outras. Nessas abordagens, as partículas possuem comportamentos bem definidos durante todo o tempo [1, 8] ou em momentos específicos do processo de busca [10].

O dHPSO não segue a noção de comportamentos pré-definidos. Ao invés disso, qualquer partícula é capaz de mudar seu comportamento, caso ela entre no estado de estagnação. Esse estado é alcançado quando a partícula não consegue melhorar sua solução durante um determinado número pré-estabelecido de iterações. Assim, essa partícula tenta escapar do estado de estagnação escolhendo aleatoriamente um novo comportamento dentre um conjunto de comportamentos pré-selecionados. Na abordagem de Engelbrecht, esse conjunto é composto por cinco comportamentos distintos:

- **Comportamento PSO padrão**: as partículas usam as regras de atualização de velocidade e posição do PSO, apresentadas nas equações (1) e (2), respectivamente. A topologia global foi adotada para esse tipo de comportamento.
- **Comportamento apenas social**: a componente cognitiva é retirada da equação (1), então as partículas são atraídas diretamente para a posição do $\vec{G}_{best}$. Esse comportamento facilita a busca em profundidade em torno de uma mesma região.
- **Comportamento apenas cognitivo**: a componente social é retirada da equação (1), então cada partícula é atraída apenas para seu $\vec{P}_{best}$. Esse comportamento favorece a busca em amplitude.

- **Comportamento *Bare bones* PSO**: abordagem introduzida por Kennedy [11]. A atualização de velocidade partícula i na dimensão j é substituída por uma distribuição Gaussiana dada por:

$$v_{ij}(t+1) \sim N\left(\frac{x_{ij}(t)+G_{best_j}}{2}, |x_{ij}(t)+G_{best_j}|^2\right). \tag{3}$$

A equação de atualização de posição é modificada para:

$$x_i(t+1) = v_i(t+1). \tag{4}$$

- **Comportamento *Modified Bare bones* PSO**: Também proposto por Kennedy [11]. A equação de atualização de posição é alterada em relação ao *Bare bones* PSO, assumindo a seguinte forma:

$$v_{ij}(t+1) = \begin{cases} x_{ij}(t), & \text{se } U(0,1) < 0,5\ ; \\ N\left(\frac{x_{ij}(t)+G_{best_j}}{2}, |x_{ij}(t)+G_{best_j}|^2\right), & \text{caso contrário .} \end{cases} \tag{5}$$

2.3. Busca por cardumes/FSS (*Fish School Search*)

O FSS é um algoritmo de otimização baseado no comportamento gregário de cardumes. Ele foi proposto por Bastos-Filho e colaboradores [3, 12]. No FSS, cada peixe representa uma solução para o problema. O sucesso de um peixe durante o processo de busca é indicado pelo seu peso. Esse algoritmo possui quatro operadores, que são executados para cada peixe do cardume em toda iteração: (i) movimento individual, que é responsável pela busca local e tem como parâmetro $step_{ind}$; (ii) alimentação, que atualiza o peso do peixe indicando o grau de sucesso ou insucesso durante o processo de busca; (iii) movimento coletivo-instintivo, que desloca os peixes de acordo com uma direção resultante; e (iv) movimento coletivo-volitivo, que controla a granularidade da busca. Como apenas os operadores (ii) e (iv) serão utilizados para a hibridização, somente estes serão detalhados neste artigo.

Operador de alimentação Esse operador determina a variação do peso do peixe a cada iteração. Sendo assim, um peixe pode aumentar ou diminuir seu peso dependendo, respectivamente, do seu sucesso ou insucesso durante o processo de busca. O peso do peixe é atualizado de acordo com a seguinte equação:

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{max(|\Delta f|)}, \tag{6}$$

onde $W_i(t)$ é o peso do peixe i, Δf_i é a variação da função objetivo entre a nova posição e a posição atual do peixe, $max(|\Delta f|)$ é o valor absoluto da maior variação de aptidão dentre todos os peixes na mesma iteração. w_{scale} limita o peso máximo do peixe. O peso de cada peixe pode variar entre 1 e w_{scale}. $W_i(t)$ é inicializado com valor $\frac{w_{scale}}{2}$ para todos os peixes.

Operador de movimento coletivo-volitivo Esse operador controla a granularidade da busca realizada pelo cardume. Quando o cardume atinge resultados melhores, o operador aproxima os peixes visando acelerar a convergência em uma dada região promissora. Caso contrário, o operador afasta os peixes do baricentro do cardume e os peixes terão maior capacidade de escapar de um ótimo local. A expansão e a contração do cardume é aplicado como um deslocamento na posição de cada peixe considerando o baricentro do cardume. O baricentro é calculado de acordo com a seguinte equação:

$$\vec{B}(t) = \frac{\sum_{i=1}^{N} \vec{x_i}(t) W_i(t)}{\sum_{i=1}^{N} \vec{x_i}(t)}. \tag{7}$$

A equação (8) é utilizada para realizar a expansão (sinal $+$) ou a contração (sinal $-$) do cardume.

$$\vec{x_i}(t+1) = \vec{x_i}(t) \pm step_{vol} r_1 \frac{\vec{x_i}(t) - \vec{B}(t)}{d(\vec{x_i}(t), \vec{B}(t))}, \tag{8}$$

onde r_1 é um número gerado aleatoriamente no intervalo $[0, 1]$. $d(\vec{x_i}(t), \vec{B}(t))$ corresponde à distância euclidiana entre o peixe i e o baricentro. $step_{vol}$ é o passo volitivo e controla o tamanho passo do peixe. O $step_{vol}$ é limitado por dois parâmetros ($step_{vol\_min}$ e $step_{vol\_max}$) e decresce linearmente de $step_{vol\_max}$ até $step_{vol\_min}$ ao longo das iterações do algoritmo. Isso provoca no algoritmo um comportamento inicial de busca em amplitude e muda dinamicamente para um comportamento de busca em profundidade.

3. ABORDAGENS HÍBRIDAS COMBINANDO FSS E VARIAÇÕES DE PSO

A primeira abordagem híbrida entre FSS e PSO foi proposta por Cavalcanti-Júnior e colaboradores em [4], e foi denominada de PSO Volitivo. Nesta abordagem foram incorporados os operadores de alimentação e movimento coletivo-volitivo do FSS ao PSO. Cada partícula passa a ter peso, o qual é utilizado para o disparar o movimento coletivo-volitivo. O parâmetro $step_{vol}$ é atualizado de acordo com a equação (9) e deve estar no intervalo [0,100].

$$step_{vol}(t+1) = step_{vol}(t)\frac{100 - decay_{vol}}{100}. \tag{9}$$

O $step_{vol}$ é reinicializado para $step_{vol\_max}$ quando for detectada uma mudança no ambiente. Para essa detecção é usada uma partícula sentinela [13], que avalia o *fitness* dessa partícula no fim de cada iteração e no começo da próxima.

Neste artigo, é proposta a hibridização do FSS com o HPSO, visando unir vantagens desses algoritmos. Pretende-se testar esta proposta, denominada HPSO Volitivo, em ambientes dinâmicos com mudanças de baixa e alta severidade. O pseudocódigo do HPSO Volitivo está mostrado no Algoritmo 1.

Algorithm 1: Pseudocódigo do HPSO Volitivo.

```
Inicialize as partículas;
enquanto a condição de parada não for atingida faça
    se detectar mudança no ambiente então
        Reinicialize step_vol e p% de partículas;
    fim
    Escolha uma partícula para ser sentinela;
    para cada partícula do enxame faça
        se partícula estiver em estado de estagnação então
            Escolha aleatoriamente novo comportamento para a partícula;
        fim
    fim
    Atualize velocidade, posição e avalie o fitness das partículas;
    Execute o operador de alimentação o e de movimento coletivo-volitivo em todo o enxame;
    para cada partícula do enxame faça
        Atualize P⃗_best, se necessário;
    fim
    Atualize G⃗_best, c1, c2, w e step_vol usando a equação (9);
fim
```

4. ARRANJO EXPERIMENTAL

Esta seção fornece os detalhes das configurações dos algoritmos usados, das métricas empregadas e dos ambientes de testes. Os primeiros experimentos conduzidos objetivaram realizar uma análise paramétrica do algoritmo proposto. Depois foi realizada uma comparação de seu desempenho entre algoritmos existentes. Por se tratar de ambientes dinâmicos, foram realizados experimentos em dois cenários de mudança no ambiente diferentes: de baixa e de alta severidade.

Cada experimento foi realizado 30 vezes, com 10.000 iterações cada um. Todos os algoritmos foram configurados com topologia local (exceto no HPSO), fator de inércia igual a 0,729844 e população de 54 partículas. Todos os algoritmos foram testados em dois casos: com reinicialização de 50% das partículas a cada mudança detectada no ambiente e sem reinicialização das partículas.

A métrica utilizada foi o *fitness* médio, que foi introduzida por Morrison [14]. Tal métrica é capaz de avaliar o desempenho de um algoritmo ao longo do processo de otimização em ambientes dinâmicos, pois leva em conta todas as iterações. Ela é dada pela média de todos os melhores valores encontrados nas iterações anteriores. O cálculo é realizado usando a seguinte equação:

$$F_{medio}(T) = \frac{\sum_{t=1}^{T} F_{melhor}(t)}{T}, \tag{10}$$

onde T é o total de iterações até o instante de tempo atual e F_{melhor} é o *fitness* da melhor partícula depois da iteração t.

4.1. Ambiente dinâmico de testes

O ambiente de testes utilizado foi o gerador de função DF1, proposto por Morrison e Jong [15]. Esse gerador é capaz de simular ambientes dinâmicos com qualquer quantidade de máximos. Cada máximo possui suas componentes que mudam dinamicamente (posição, altura e inclinação), sendo associado a uma função logística. Esses parâmetros podem variar dentro de uma faixa de valores pré-definidos e com grau de severidade ajustável.

Nas simulações foram utilizadas 10 dimensões, 10 pontos de máximo e domínio da função $[-50, 50]^d$ em todas as dimensões. Todos os pontos de máximo possuem inclinação fixa determinada no início de cada simulação aleatoriamente no intervalo $[1, 7]$. Os picos se movem de forma independentemente, respeitando o domínio da função e variam de altura no intervalo $[10, 50]$ a cada 100 iterações. Para o caso de mudança de baixa severidade, o parâmetro de escala das componentes dinâmicas usado assumiu valor 0,1 e para mudança de alta severidade assumiu valor 0,5. O coeficiente A da função logística foi configurado com valor 2,1 em ambos os casos. O resultado da função logística, multiplicado pelo parâmetro de escala e tamanho do intervalo de cada componente dinâmica, dita o tamanho do passo de mudança na posição e os valores dos pontos de máximo. A cada uma das execuções, os ambientes são gerados usando a mesma semente para todos os algoritmos. No entanto, a dinâmica do ambiente é diferente a cada simulação.

4.2. Configuração dos algoritmos

O valor do coeficiente de aceleração cognitivo do PSO c_1 é inicialmente 2,5 e decresce linearmente até 0,5 a cada 100 iterações (intervalo de mudança no ambiente). Já o valor inicial do coeficiente de aceleração social c_2é iniciado em 0,5 e cresce linearmente até 2,5 ao longo do mesmo intervalo de mudança. Logo, a cada 100 iterações os valores de c_1 e c_2 são reiniciados. Dessa forma, o algoritmo tem maior capacidade de gerar diversidade após mudanças no ambiente e assim explorar o espaço de busca visando o novo ótimo. O algoritmo muda o comportamento gradativamente para busca em profundidade, até que ocorra outra nudança no ambiente, retornando à primeira etapa desse processo.

No Charged PSO (CPSO) os valores de c_1 e c_2 foram fixados em 1,49618. 50% das partículas foram carregadas com intensidade de carga 16, de acordo com as especificaçoes apresentadas em [1]. Os parâmetros p e p_{core} utilizados foram 1 e 30, respectivamente.

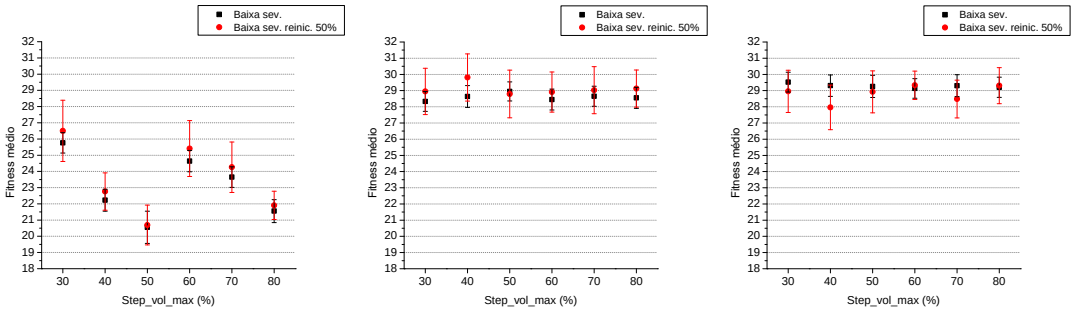
No HPSO, as partículas com comportamento de PSO padrão seguiram a mesma configuração do PSO. As partículas com comportamento apenas social tiveram o valor c_2 fixo em 2,5 e para comportamento apenas cognitivo tiveram o valor de c_1 fixo em 2,5. Qualquer partícula que ficasse 10 iterações sem melhorar seu P_{best} foi considerada em estado de estagnação, e tinha seu comportamento alterado aleatoriamente.

No PSO Volitivo e HPSO Volitivo, os parâmetros relativos ao PSO e HPSO foram os mesmos citados anteriormente. $w_{scale} = 5000$ e $step_{vol\_min} = 0,01\%$. Foram testados os seguintes valores para o parâmetro $decay_{vol}$ 0%, 5% e 10% combinados com os valores de $step_{vol\_max}$ iguais a 30%, 40%, 50%, 60%, 70% e 80%.

5. RESULTADOS

5.1. Análise paramétrica

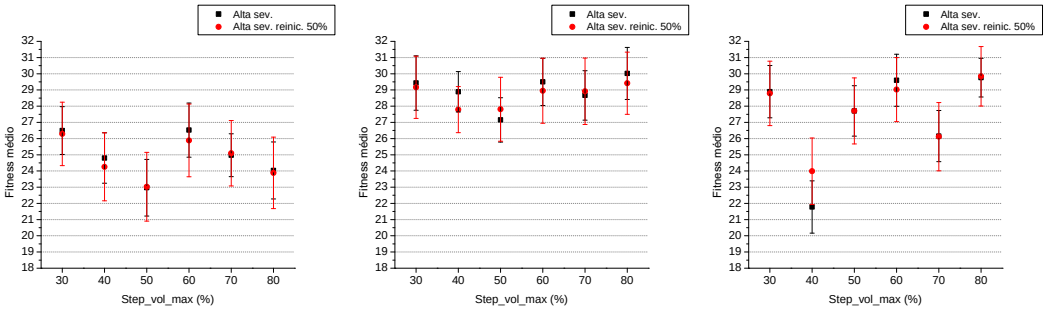
As Figuras 1, 2, 3 e 4 mostram o gráfico com as médias e desvio padrão (entre parênteses) dos parâmetros avaliados para o PSO Volitivo e HPSO Volitivo, considerando e desconsiderando a reinicialização das partículas. As Figuras 1(b) e 1(c) mostram que em ambiente com baixa severidade, os resultados para o Volitive PSO são similares para $decay_{vol}$ 5% e 10%. No entanto, se não houver decaimento do passo volitivo (ver Figura 1(a)), ocorre degradação do desempenho do algoritmo, pois sem decaimento o algoritmo gera diversidade, mas reduz a capacidade de convergência. Para o ambiente com alta severidade, o PSO Volitivo obteve resultados melhores e mais consistentes com $decay_{vol}$ = 5% (Figura 2(b)), do que com os outros valores testados (ver Figuras 2(a) e 2(c)). Além disso, de acordo com a Figura 2(b), $decay_{vol}$ = 5% torna o algoritmo menos sensível ao parâmetro $step_{vol\_max}$. Em ambos os tipos de ambientes, não houve diferença significativa em reinicializar 50% das partículas.



(a) $decay_{vol} = 0\%$ (b) $decay_{vol} = 5\%$ (c) $decay_{vol} = 10\%$

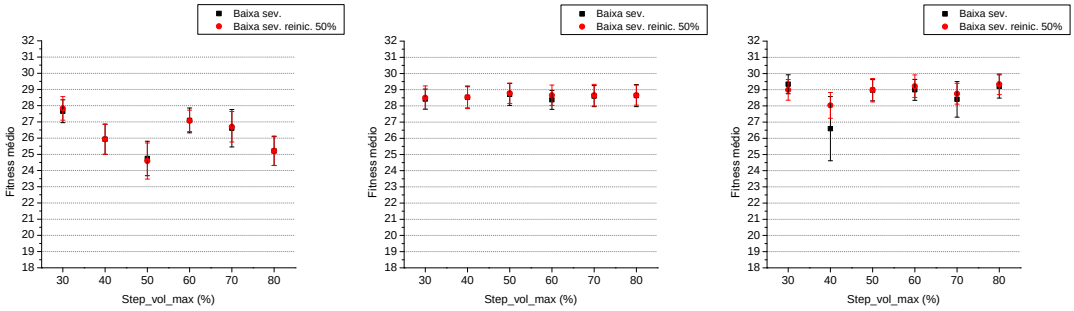
Figura 1: *Fitness* médio da última iteração do PSO Volitivo em ambiente com baixa severidade.

A Figura 3 mostra que o HPSO Volitivo, em ambiente de baixa severidade, se torna menos sensível a $step_{vol\_max}$ para $decay_{vol}$ = 5% (Figura 3(b)), pois a variação nos resultados para os diferentes valores de $step_{vol\_max}$ é menor e o erro também é menor. Para alta severidade, o HPSO Volitivo obteve resultados similares para $decay_{vol}$ 0% e 5% (Figuras 4(a) e 4(b)). Entretanto, para $decay_{vol}$ = 10% (ver Figura 4(c)) ocorreu alta degradação nos resultados quando não foi utilizada a reinicialização das partículas após mudanças no ambiente. Isso indica que o operador volitivo foi comprometido pela alta taxa de decaimento do passo volitivo, ou seja, isso reduziu a capacidade do operador gerar diversidade (dilatação do enxame) para explorar o ambiente modificado e encontrar o novo ótimo.



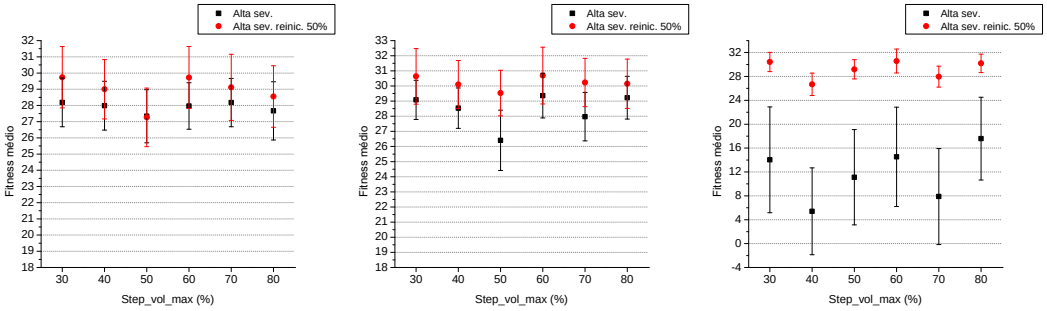
(a) $decay_{vol} = 0\%$ (b) $decay_{vol} = 5\%$ (c) $decay_{vol} = 10\%$

Figura 2: *Fitness* médio da última iteração do PSO Volitivo em ambiente com alta severidade.



(a) $decay_{vol} = 0\%$ (b) $decay_{vol} = 5\%$ (c) $decay_{vol} = 10\%$

Figura 3: *Fitness* médio da última iteração do HPSO Volitivo em ambiente com baixa severidade.



(a) $decay_{vol} = 0\%$ (b) $decay_{vol} = 5\%$ (c) $decay_{vol} = 10\%$

Figura 4: *Fitness* médio da última iteração do HPSO Volitivo em ambiente com alta severidade.

A partir dos resultados mostrados nas Figuras 1, 2, 3 e 4, foi possível identificar que o HPSO Volitivo foi mais sensível ao parâmetro $decay_{vol}$. Apesar disso, o valor de $decay_{vol}$ = 5% gerou melhor desempenho em ambos os algoritmos e em ambos os ambientes. então, para esse valor de $decay_{vol}$ fixado, a maioria dos valores de $step_{vol\_max}$ não provocaram diferenças significativas nos resultados. Sendo assim, $step_{vol\_max}$ foi fixado em 80%. Esses valores de $decay_{vol}$ e $step_{vol\_max}$ foram usados nos experimentos de comparação entre os algoritmos, cujos resultados estão na seção 5.2.

5.2. COMPARAÇÃO ENTRE ABORDAGENS

A Figura 5 mostra a evolução do *fitness* médio ao longo das 10.000 iterações com reinicialização de partículas no ambiente de alta severidade. Os algoritmos com operador volitivo obtiveram resultados bastante próximos e superaram os demais. O HPSO obteve desepenho um pouco inferior às abordagens com operador volitivo. Por fim, O PSO e o *Charged* PSO obtiveram resultados inferiores aos demais algoritmos.

A Tabela 1 mostra os valores do *fitness* médio na última iteração. Em todos os casos testados, os algoritmos com operador volitivo foram superiores, obtendo resultados similares entre si. Por outro lado, os demais algoritmos apresentaram maior diferença nos resultados entre os tipos de ambientes.

Comparando os casos com e sem reinicialização de partículas (Tabelas 1(a) e 1(c) com 1(b) e 1(d)), os algoritmos sem operador volitivo tiveram maior variação nos resultados, principalmente em ambiente com alta severidade. Isso indica que esses algoritmos são mais dependentes da reinicialização de partículas para gerar diversidade.

A Figura 6 mostra os gráficos *box plot* comparando o *fitness* médio da última iteração dos algoritmos em ambiente com alta severidade. Comparando as Figuras 6(a) e 6(b), os algoritmos sem o operador volitivo melhoraram o desempenho quando

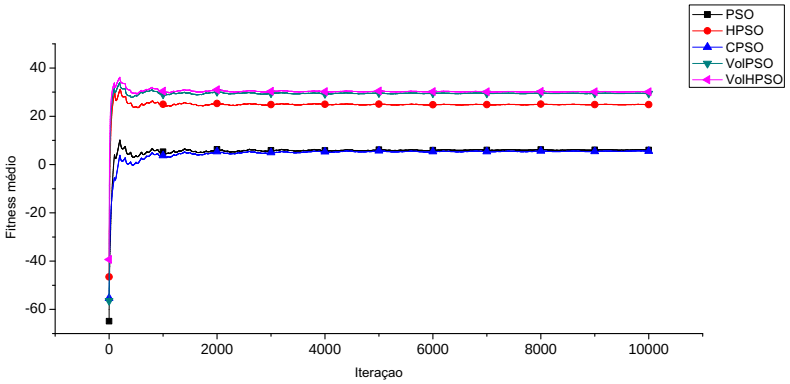


Figura 5: Evolução do *fitness* médio no ambiente com alta severidade e reinicialização de 50% das partículas.

Tabela 1: *Fitness* médio na última iteração - média e desvio padrão após 10.000 iterações.

(a) Baixa severidade.

Algoritmo	média	Desvio padrão
PSO	16,91	1,00
HPSO	25,92	0,90
CPSO	6,34	4,20
VolPSO	**28,55**	0,65
VolHPSO	**28,64**	0,68

(b) Baixa severidade com reinicialização.

Algoritmo	média	Desvio padrão
PSO	19,59	1,30
HPSO	26,60	0,91
CPSO	17,49	2,90
VolPSO	**29,13**	1,15
VolHPSO	**28,66**	0,62

(c) Alta severidade.

Algoritmo	média	Desvio padrão
PSO	-7,89	1,59
HPSO	16,96	1,66
CPSO	-4,17	2,86
VolPSO	**30,02**	1,60
VolHPSO	**29,23**	1,41

(d) Alta severidade com reinicialização.

Algoritmo	média	Desvio padrão
PSO	6,00	2,02
HPSO	24,82	1,57
CPSO	5,57	1,67
VolPSO	**29,42**	1,92
VolHPSO	**30,15**	1,63

realizada a reinicialização de partículas. No entanto, não superaram os algoritmos com operador volitivo. Isso mostra que o operador volitivo é um mecanismo bastante eficiente para aumentar a capacidade dos algoritmos baseados em PSO para gerar diversidade e acelerar a convergência.

6. CONCLUSÃO

O presente artigo apresentou um algoritmo híbrido usando FSS e HPSO, o HPSO Volitivo, que por sua vez apresentou resultados similares aos obtidos pelo PSO Volitivo. A contribuição mais relevante deste artigo foi mostrar que, quando incorporado o operador volitivo, os algoritmos melhoraram seus desempenhos de forma significativa. Tornando desnecessários outros mecanismos, como a reinicialização das patículas.

Da análise paramétrica dos algoritmos com operador volitivo, concluiu-se que os resultados foram mais consistentes para $decay_{vol} = 5\%$. Esse valor para este parâmetro é suficientemente baixo para deixar o operador volitivo gerar diversidade e alto o suficiente para acelerar a convergência do enxame.

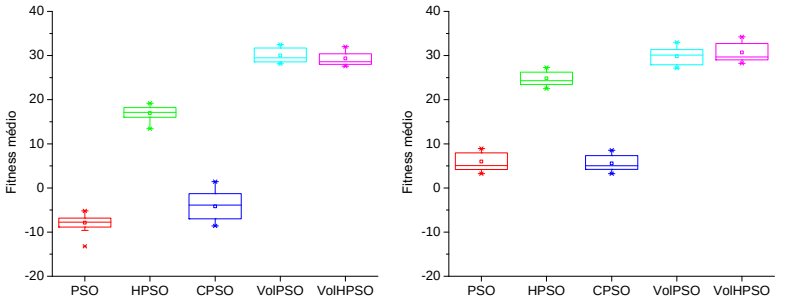
Como trabalho futuro será realizado um estudo de escalabilidade dos algoritmos com operador volitivo. Como o HPSO possui maior escalabilidade, espera-se que o HPSO Volitivo supere o PSO Volitivo em ambientes dinâmicos de mais alta dimensionalidade. Outro ponto de melhoria é que atualmente as partículas do HPSO escolhem o novo comportamento aleatoriamente. Um novo mecanismo pode se criado para seleção mais eficiente do comportamento dinâmico.

7. AGRADECIMENTOS

Os autores deste artigo agradecem pelo apoio da Universidade de Pernambuco, POLI, FACEPE, CAPES e CNPq.

REFERÊNCIAS

[1] T. Blackwell and P. Bentley. "Dynamic Search with Charged Swarms". *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 19–26, 2002.



(a) Sem reinicialização de partículas. (b) Com reinicialização de 50% das partículas.

Figura 6: Boxplot do *Fitness médio* da última iteração dos algoritmos testados no ambiente com alta severidade.

[2] A. Nickabadi, M. M. Ebadzadeh and R. Safabakhsh. "Evaluating the performance of DNPSO in dynamic environments". *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2640–2645, October 2008.

[3] C. J. A. Bastos Filho, F. B. de Lima Neto, A. J. C. C. Lins, A. I. S. Nascimento and M. P. Lima. "A novel search algorithm based on fish school behavior". In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2646–2651. IEEE, October 2008.

[4] G. M. Cavalcanti-Júnior, C. J. A. Bastos-Filho, F. B. Lima-Neto and R. M. C. S. Castro. "A Hybrid Algorithm Based on Fish School Search and Particle Swarm Optimzation for Dynamic Problems". In *Proceedings of the International Conference on Swarm intelligence, ICSI 2011*, edited by Y. Tan, Lecture Notes in Computer Science, pp. 543–552, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] A. P. Engelbrecht. "Heterogeneous particle swarm optimization". In *Proceedings of the 7th international conference on Swarm intelligence*, pp. 191–202, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] B. J. Leonard, A. P. Engelbrecht and A. B. van Wyk. "Heterogeneous Particle Swarms in Dynamic Environments". *IEEE Symposium on Swarm Intelligence, SIS 2011*, pp. 9–16, 2011.

[7] Y. Shi and R. Eberhart. "A modified particle swarm optimizer". *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73, 1998.

[8] A. Silva, A. Neves and E. Costa. "An Empirical Comparison of Particle Swarm and Predator Prey Optimisation." In *AICS'02*, pp. 103–110, 2002.

[9] M. A. M. de Oca, J. Pena, T. Stutzle, C. Pinciroli and M. Dorigo. "Heterogeneous particle swarm optimizers." In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009*, pp. 698–705, 2009.

[10] J. S. Vesterstrom, J. Riget and T. Krink. "Division of labor in particle swarm optimisation". In *Proceedings of the Congress on Evolutionary Computation on 2002. CEC '02. Volume 02*, pp. 1570–1575, Washington, DC, USA, 2002. IEEE Computer Society.

[11] J. Kennedy. "Bare bones particle swarms". In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03*, pp. 80–87, april 2003.

[12] C. J. A. Bastos-Filho, F. B. de Lima-Neto, A. J. C. C. Lins, A. I. S. Nascimento and M. P. Lima. "Fish School Search". In *Nature-Inspired Algorithms for Optimisation*, edited by R. Chiong, volume 193 of *Studies in Computational Intelligence*, pp. 261–277. Springer, 2009.

[13] A. Carlisle and G. Dozier. "Applying the particle swarm optimizer to non-stationary environments". Phd thesis, Auburn University, Auburn, AL, 2002.

[14] R. W. Morrison. "Performance Measurement in Dynamic Environments". *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pp. 5–8, 2003.

[15] R. Morrison and K. De Jong. "A test problem generator for non-stationary environments". *Proceedings of the Congress on Evolutionary Computation*, pp. 2047–2053, 1999.