Revista de Engenharia e Pesquisa Aplicada, Volume 2, Número 1, 2016

Sintonia de Controlador PID baseado em Busca por Cardumes

PID controller tuning on Fish School Search

Janderson S. de Freitas
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
sfreitas.janderson@gmail.com

Carmelo J. A. Bastos-Filho
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
carmelo.filho@upe.br

Roberto F. Dias Filho
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil

Luiz Felipe V. Verçosa
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 – Recife, Brasil

Eril V. P.
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 – Recife, Brasil2

Resumo *A atenção no desenvolvimento e na eficácia de algoritmos de otimização têm crescido nos últimos anos devido à sua aplicabilidade em muitos problemas de otimização em engenharia. Seguindo esta tendência, este artigo tem como objetivo avaliar o desempenho de um algoritmo de otimização, chamado algoritmo de otimização por busca em cardume, o FSS (Fish School Search), na tarefa de otimizar os ganhos de um controlador proporcional integral e derivativo (PID). O FSS é um dos algoritmos que compõem o gênero de otimizadores inteligentes por enxame. O FSS é inspirado no comportamento de um cardume, no qual os peixes se deslocam em um espaço de busca estabelecido à procura de comida, os peixes com melhores resultados engordam e influenciam os demais a segui-lo, concentrando depois de um tempo, o cardume em soluções ótimas. O controlador proporcional Integral e derivativo, controlador PID, é uma ferramenta da engenharia de análise e controle de porcessos, com o objetivo de verificar o erro entre o valor de saída de um processo e o valor desejado, ele interage sobre o erro com o objetivo de minimiza-lo, zera-lo ou preveni-lo. O controlador PID possui três parâmetros (Kp, Ki e Kd) que precisam ser sintonizados para execução do controle, Este processo, muitas vezes, vem sendo realizado de forma manual ou através de métodos heurísticos clássicos. Este artigo apresenta uma ferramenta de simulação computacional (FSS) para sintonizar, de modo rápido e eficiente, o controlador PID. O algoritmo foi implementado em interface de desenvolvimento JAVA, o Eclipse, e integrado à plataforma MatLab. O MatLab foi usado como ferramenta essencial para a simulação do controlador PID, do processo, das funções de teste e do método clássico Ziegler-Nichols, garantindo, dessa forma, maior confiabilidade nos testes.*

Palavras-Chave: *Controlador PID, Controle de processos, Sintonia PID, Inteligência Computacional, Inteligência de enxames, Busca por Cardumes.*

1 Introdução

Problemas de otimização se caracterizam pela busca da melhor solução entre todas as soluções possíveis para um determinado problema. Para problemas simples, com duas variáveis de entrada por exemplo, pode-se encontrar a melhor solução utilizando os conceitos matemáticos de diferenciação e integração. Entretanto, existem problemas mais complexos que podem possuir dezenas de variáveis de entrada, combinações não lineares entre elas e ainda espaços de busca amplos. Considere um problema hipotético em que seja necessário avaliar variáveis como: temperatura, umidade, pressão, velocidade do vento, luminosidade, altura e acidez do solo para realizar um obra de engenharia, por exemplo. Para tal problema pode ser inviável utilizar métodos matemáticos tradicionais. Felizmente, pode-se fazer uso de algoritmos de otimização que apesar de não garantirem sempre a melhor solução, ao menos podem garantir soluções aceitáveis [2].

Algoritmos de otimização são técnicas de busca por uma solução ótima para um certo problema que pode ser modelado por funções objetivo. Inspirações oriundas da natureza permitiram o desenvolvimento de modelos capazes de resolver diversos problemas complexos de otimização. Esses modelos fazem parte do campo de estudo chamado Inteligência Computacional [2].

O Algoritmo de Otimização por Cardume (FSS, *Fish School Search*) foi proposto por Bastos Filho e Lima Neto em 2008 [1] em sua versão original. O FSS inspirasse na busca de alimento por parte de uma população de peixes. Ele é um algoritmo recente quando comparado ao ano de surgimento da maioria dos outros. Por exemplo, o algoritmo de otimização por enxame de partículas (PSO, *Particle Swarm Optimization*) surgiu em 1995, enquanto que a primeira versão do FSS foi lançada em 2008. Naturalmente, algoritmos recentes precisam de reajustes e melhorias. Esse é um dos principais motivos de estudo do desempenho do FSS por meio desse trabalho.

A contribuição deste artigo está vinculada à aplicação e análise de desempenho de um algoritmo FSS utilizando otimização monobjetivo dos ganhos de um controlador proporcional, integral e derivativo (PID). Em Latha, K. [4] e Jau-Woei Perng [5] também foi abordado o mesmo estudo de caso, mas estes foram analisados para um procedimento de otimização monobjetivo baseado no algoritmo de otimização por enxame de partículas (PSO).

O restante do artigo está organizado da seguinte forma. Na seção 2 são descritos os fundamentos que regem o algoritmo FSS. Na seção 3 é realizada a descrição do controlador PID utilizado. Resultados obtidos na simulação são apresentados na seção 4. A conclusão é apresentada na seção 5.

2 Otimização por busca em cardume (FSS)

O algoritmo de otimização baseado em cardumes foi proposto em 2008 [1] e, como já mencionado, baseia-se na busca de alimentos por parte de um cardume. Foram observados os seguintes aspectos presentes em cardumes reais:

- Os peixes se organizam em cardumes de maneira a evitar o ataque de predadores e facilitar a obtenção de comida;
- Cada peixe possui certa independência para buscar alimento por conta própria sem por isso abandonar o cardume;
- O cardume nada com uma componente em uma mesma direção no oceano;
- Quando a comida torna-se abundante em um certo ponto do cardume, ele contrai-se em torno daquela região para alimentar-se. Em contrapartida, quando a comida torna-se esparsa, ocorre uma dilatação no cardume com o intuito de aumentar a região de busca.

2.1 Conceitos importantes

Antes de descrever o comportamento dos peixes no algoritmo, faz-se necessário definir alguns conceitos intrínsecos à busca por cardume. O primeiro deles é o peso individual dos peixes. Esse peso é a medida de quão gordo (e bem-sucedido) é o peixe. Peixes gordos representam um histórico de sucesso em encontrar comida (*fitness*) e exercem uma maior influência no cardume que os magros. O segundo conceito é o de passo. O passo, como o nome sugere, representa quanto o peixe irá nadar em certa dimensão. Sendo assim, ele representa um deslocamento no espaço de busca em certa dimensão e controla a busca por profundidade e por largura do algoritmo. O terceiro e último conceito é a divisão do movimento do cardume em duas categorias, que são: movimento individual e movimento coletivo. Cada movimento possui uma correspondência direta com as características observadas no cardume real.

Resumindo, o comportamento de cada peixe é definido por um ciclo de operações considerando o movimento

individual, coletivo instintivo e coletivo volitivo. O movimento Individual é caracterizado por uma busca local gulosa com passo pré-definido *stepind*. Então, os pesos são atualizados usando (1):

$$W_i(t+1) = W_i(t) + \frac{f[x_i(t+1)] - f[x_i(t)]}{\max\{|f[x_i(t+1)] - f[x_i(t)]|\}}, \quad (1)$$

em que a diferença de aptidão é normalizada para cada iteração. O movimento Coletivo Instintivo é calculado usando (2).

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \frac{\sum_{i=1}^{N} \Delta\mathbf{x}_{indi}\{f[x_i(t+1)] - f[x_i(t)]\}}{\sum_{i=1}^{N}\{f[x_i(t+1)] - f[x_i(t)]\}} \quad (2)$$

Então o Baricentro do Cardume é calculado usando (3). E o movimento coletivo Volitivo é executado de acordo com (4) ou (5), dependendo se o processo é de contração ou expansão.

$$Bari(t) = \frac{\sum_{i=1}^{N} \mathbf{x}_i(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)} \quad (3)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) - step_{vol}.rand.[\mathbf{x}_i(t) - Bari(t)], \quad (4)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + step_{vol}.rand.[\mathbf{x}_i(t) - Bari(t)], \quad (5)$$

A Figura 1 mostra as equações apresentadas anteriormente implementadas na rotina padrão do algoritmo de otimização FSS.

```
INICIO {

  Inicialize todos os peixes em posições xI(0) aleatórias e distantes do
  mínimo global;

  Inicialize aleatoriamente o peso wi(0) de todos os peixes;

  ENQUANTO critério de parada não for alcançado FAÇA {

     PARA CADA peixe FAÇA {

         Encontre uma posição vizinha;

         Avalie a variação no fitness de acordo com e mova o peixe
         apenas se o fitness melhorou;

         Alimente os peixes utilizando (1);

     FIM-PARA

     PARA CADA peixe FAÇA

         Execute o movimento instintivo utilizando (2);

     FIM-PARA

     Calcule o baricentro utilizando (3);

     PARA CADA peixe FAÇA

         Execute o movimento volitivo usando (4);

     FIM-PARA

     Atualize  step_ind  e  step_vol;

  FIM-ENQUANTO

FIM
```

Figura 1 : Pseudo-Codigo referente a rotina do algoritmo FSS (*Fish School Search*).

3 Descrição do Controlador PID

O controle Proporcional-Integral-Derivativo (PID) é uma da ferramentas de controle mais usadas na indústria e tem sido utilizado em todo o mundo para sistemas de controle industrial. A popularidade de controladores PID pode ser atribuída em parte ao seu desempenho robusto em uma ampla gama de condições de funcionamento e em parte à sua simplicidade funcional, que permite aos engenheiros operá-los de uma forma simples e direta. Com o objetivo de verificar o erro entre o valor de saída de um processo e o valor desejado, ele interagi sobre o erro visando minimiza-lo, zera-lo ou preveni-lo. O controlador PID possui três parâmetros (*Kp*, *Ki* e *Kd*) que precisam ser sintonizados para execução do controle, Este processo, muitas vezes, vem sendo realizado de forma manual através de métodos heurísticos clássicos, como exemplo, o método de sintonia de Ziegler-Nichols.

A Figura 2 mostra o diagrama em malha fechada contendo um controlador PID aplicado a um porcesso qualquer:

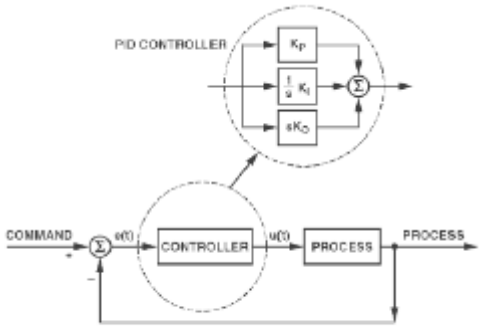

Figura 2: Planta de processo com controle PID.

A equação padrão de sintonia de um controlador PID em função do tempo é dado por:

$$MV(t) = Kp.e(t) + Ki.Integral[e(\tau)] + Kd.Derivada[e(t)] \quad (5)$$

Ou, no dominio de laplace por:

$$MV(s) = Kp + Ki/s + Kd \cdot s \quad (6)$$

A seguir, são mostrados as influências dos ganhos, proporcional (*Kp*), integral (*Ki*) e derivativo (*Kd*) sobre o erro de resposta do processo.

Ganho Proporcional (*Kp*): O ganho proporcional (*Kp*) determina a taxa de resposta de saída para o sinal de erro. Por exemplo, se o termo de erro tem uma magnitude de 10, um ganho proporcional de 5 produziria uma resposta proporcional de 50. Em geral, o aumento do ganho proporcional irá aumentar a velocidade da resposta do sistema de controle. No entanto, se o ganho proporcional é muito grande, a variável de processo começará a oscilar. Se *Kp* é aumentado ainda mais, as oscilações ficarão maiores e o sistema ficará instável e poderá oscilar.

Ganho Integral (*Ki*): A componente integral soma o termo de erro ao longo do tempo. O resultado é que mesmo um pequeno erro fará com que a componente integral aumente lentamente. A resposta integral irá aumentando ao longo do tempo a menos que o erro seja zero, portanto, o efeito é o de conduzir o erro de estado estacionário para zero. O *Steady-State* de erro é a diferença final entre as variáveis do processo e do ponto de referência.

Ganho Derivativo (*Kd*): A componente derivada faz com que a saída diminua se a variável de processo aumenta rapidamente. A derivada de resposta é proporcional à taxa de variação da variável de processo. Aumentar o ganho derivativo (*Kd*) faz com que o sistema de controle reaja mais fortemente a mudanças no parâmetro de erro aumentando a velocidade da resposta global de controle do sistema. Na prática, a maioria dos sistemas de controle utilizam o ganho derivativo (*Kd*) muito pequeno, pois a derivada de resposta é muito sensível ao ruído no sinal da variável de processo. Se o sinal de *feedback* do sensor é ruidoso ou se a taxa de malha de controle é muito lenta, a derivada de resposta pode tornar o sistema de controle instável.

A Figura 3 mostra uma curva que representa a resposta a um sinal de degrau unitário, onde é possivel obsevar todos os parâmetros de resposta nescessários pra a sintonia de um controlador PID. De forma geral, procura-se minimizar o *overshooting* e os tempos de subida (*Tr*) e estabelecimento (*TS*).

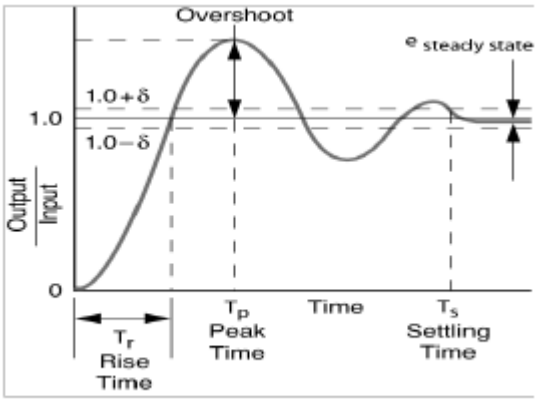

Figura 3: Curva de processo em resposta a um degrau unitário.

4 Resultados e Simulação

Toda o desenvolvimento deste trabalho foi realizado em Java, usando IDE Eclipse, em conjunto com o MatLab. No ambiente Eclipse foi implementado o algoritmo de otimização FSS e validado usando funções de teste bem conhecidas, como: *Sphere, Rastrigin, Griewank* e *Ackley*. No ambiente Matlab foi implementado um processo teste e o controlador PID, bem como sua sintonia com o método clássico proposto por Ziegler-Nichols [3] para fins de comparação.

A função a ser otimizada pelo Algoritmo, foi uma ponderação proposta por Latha e Raijinikanth [4]. Neste trabalho, foram considerados dois parâmetros baseados em função objetivo, considerando as restrições de erro e no domínio do tempo como importante, o *overshooting* (percentagem de ultrapassagem - *Ms*) e o *settling time* (tempo de estabilização - *TS*). A função de aptidão foi definida por (7).

$$fitness(\theta) = w_1.M_p + w_2.T_s \qquad (7)$$

em que θ é um vetor com as três variáveis de otimização $[Kp, Ki, Kd]$. $w1$ e $w2$ são pesos que ponderam dois subobjetivos.

A Figura 4 mostra a caracterização da curva de resposta do processo estudado sem controlador. Pode-se perceber que esta pode ser simulada juntamente com um Controlador proporcional puro (P), com ganho Kp = 1, Ki= 0 e Kd = 0. As Figuras 5, 6 e 7, mostram os resultados finais dos parâmetros da curva resposta, pra os melhores valores encontrados de *Kp*, *Ki* e *Kd*, utilizando o metodo de ZN, FSS e PSO, respectivamente.

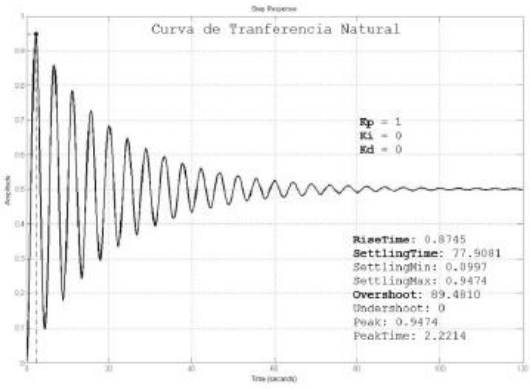

Figura 4: Resposta ao degrau unitário com PID sem otimização.

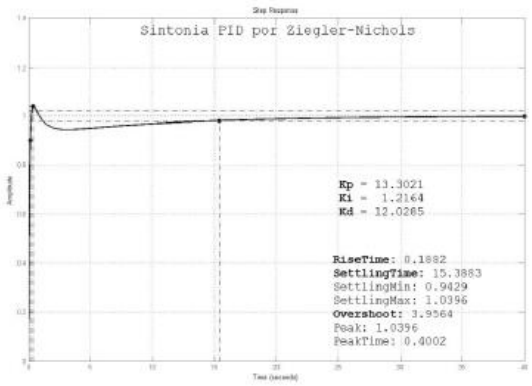

Figura 5: Resposta ao degrau unitário por sintonia PID Ziegler-Nichols.

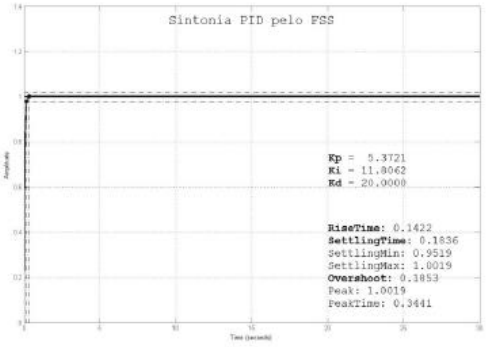

Figura 6: Resposta ao degrau unitário com PID sintonizado pelo FSS.

É possível perceber uma grande semelhança na curva resposta controlada pelo FSS e pelo PSO. Apesar disso, pode-se observar nos resultados apresentados na Figura 6, que o FSS se aproximam mais do degrau ideal, em relação aos outros métodos.

A Figura 8 mostra a distribuição de dados, em *boxplot*, das melhores soluções encontradas pelos algoritmos de otimização ao término de cada simulação. Cada distribuição representa um teste de 30 simulações com 100 iterações cada.

Apesar de resultados finais semelhantes, como mostram as Figuras 6 e 7, o desempenho do FSS, para esse problema estudado, foi mais estável, como mostrado na Figura 8.

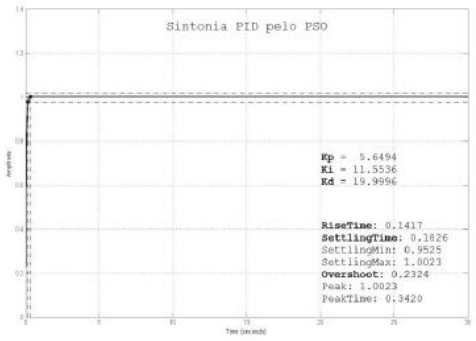

Figura 7: Resposta ao degrau unitário por sintonia PID-PSO.

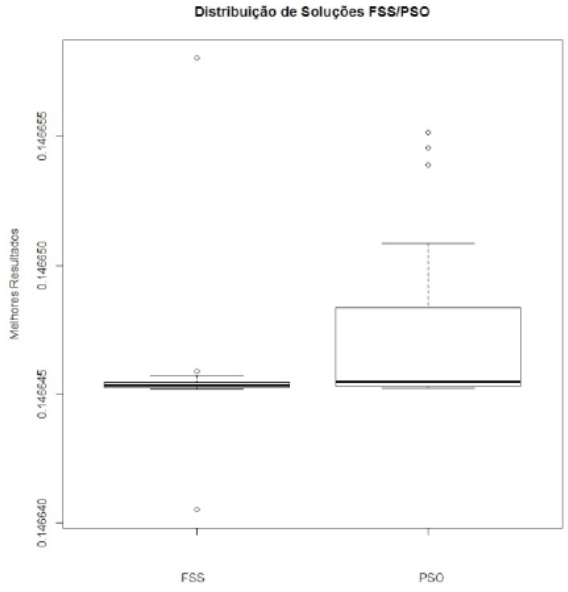

Figura 8: Distribuição de dados em boxplots dos melhores resultados para função de aptidão usando FSS e PSO.

5 Conclusões

Este artigo procurou mostrar a viabilidade da utilização do algoritmo de busca FSS para sintonia de controladores PID. O algoritmo FSS apresentou bom desempenho na otimização da função objetivo estudada em relação aos métodos heurístico clássico e algoritmo de otmização por enxame de particulas (PSO).

Foi observado durante os experimentos a relação conflitante entre si, dos parâmetros a serem otimizados da função objetivo, fazendo com que, nem sempre seja possível ponderá-los, o que nos leva, em pesquisas futuras, à uma abordagem multiobjetiva das funções objetivo, utilizando-se de um algoritmo mais robusto, com soluções multiobjetivas.

6 Agradecimento

Os autores agradecem ao programa de iniciação científica PIBIC/POLI-PE.

Referências

[1] BASTOS-FILHO, C. J. A. et al. A novel search algorithm based on fish school behavior. IEEE International Conference on Systems, Man, and Cybernetics, p. 2646–2651, 2008.

[2] ENGELBRECHT, A. Computational Intelligence An Introduction. [S.l.]: Wiley & Sons, 2007.

[3] J. G. Ziegler and N. B. Nichols, “Optimum settlings for automatic controllers,” Transactions of the ASME, vol. 64, pp. 759–768, 1942.

[4] LATHA, K.; RAIJINIKANTH, V.; P. M. SUREKHA, P. M.: “PSO-Based PID Controller Design for a Class of Stable and Unstable Systems”, Hindawi Publishing Corporation, ISRN Artificial Intelligence, Volume 2013, Article ID 543607, 11 pages.

[5] Jau-Woei Perng ; Guan-Yan Chen ; Shan-Chang Hsieh: “Optimal PID Controller Design Based on PSO-RBFNN for Wind Turbine Systems”, Energies 2014, 7, 191-209; doi:10.3390/en7010191.