A VNS Algorithm for PID Controller: Hardware-In-The-Loop Approach

Guilherme Silva, Pedro Silva, Valéria Santos, Alan Kardek Rêgo Segundo, Eduardo Luz and Gladston Moreira, *Member, IEEE,*

***Resumo*—Tuning the Proportional Integral Derivative, or PID, controller in cyber-physical systems is a major challenge as it requires advanced mathematical skills. Several authors in the literature have shown that optimization algorithms are efficient for auto-adjust PID controller constants, especially when there is no mathematical modeling. However, the literature lacks works that show the efficiency of the Variable Neighborhood Search (VNS) algorithm to auto-adjust the PID. In this work, we investigate the efficiency of the Variable Neighborhood Algorithm to fine-tune a PID controller of a real cyber-physical-system: a birotor flying drone. The approach consists of applying a numerical neighborhood structure to optimize the three constants of the PID, according to a proposed fitness function. Experiments reveal the feasibility of fine-tuning the PID controller and the birotor balancing with the Variable Neighborhood Algorithm with reduced time. We compared the VNS-approach against one based on genetic algorithms, and on average, the VNS-approach achieves better results with lower computational and memory costs. Results suggest that the approach may be used in real or commercial systems, helping to fine-tune the controller to new environment changes or even last-minute project modifications**

***Index Terms*—VNS, Cyber-physical systems, Proportional Integral Derivative, Self-tuning, Hardware-in-the-loop.**

I. Introdução

Nos últimos anos, o acesso à informação tornou-se mais fácil para pessoas de todo o mundo. Esse cenário permitiu que entusiastas em tecnologia não só compartilhassem seus projetos, mas também procurassem aprender outros projetos similares. Com base nisso, os projetos de sistemas ciber-físicos (em inglês, *Cyber Physical Systems* ou CPS) foram amplamente compartilhados e adaptados. Em um CPS, um sistema de computador é dedicado a monitorar e controlar uma planta física [1]. Nesse sentido, é necessário que se tenha não só conhecimento da modelagem matemática do sistema a ser construído, mas também de sua integração com um sistema embarcado.

Tendo isso em mente, a replicação de um CPS por parte de pessoas que não sejam especialistas pode ser uma tarefa complexa, visto que, geralmente, tem-se um modelo físico diferente daquele proposto em manuais, seja por customizações ou erro de projeto, e, por isso, faz-se necessário uma nova modelagem matemática. Além disso, mesmo que a modelagem física seja idêntica, a mudança no ambiente em que o CPS está inserido pode afetar em seu comportamento. Esse cenário pode ser um obstáculo para quem não é especialista e até mesmo para um especialista, conforme descrito em [2], em que os autores mostram a dificuldade de se ajustar manualmente um Veículo Operado Remotamente (em inglês, *Remote Operated Vehicles* ou ROV), para aplicações em tarefas submarinas. Nesse cenário, as condições ambientais podem alterar as características da planta, como, por exemplo, vazamentos de óleo. No ROV proposto, os sensores da planta devem fornecer informações para o sistema embarcado, que, a partir delas, gera uma resposta de controle para os atuadores. Essa abordagem é uma simulação de uma planta real em tempo real, sendo conhecida como *hardware-in-the-loop*.

Avaliando um cenário semelhante, uma abordagem *hardware-in-the-loop* é apresentada em [3], em que um controlador *fuzzy* é projetado e direcionado para controlar o hardware. Neste caso, um otimizador adapta as configurações da lógica *fuzzy* do controlador.

A abordagem mais usada na indústria para a calibração de uma planta física que usa um controlador Proporcional Integral Derivativo (PID) é o método de Ziegler-Nichols [4], o qual é baseado na curva de reação do sistema. No entanto, essa abordagem consome um tempo considerável para realizar a sintonia do controlador, visto que se trata de um procedimento matemático complexo. Por esse motivo, o processo de calibração do PID seria trabalhoso para sistemas que precisem de calibração contínua devido a uma mudança sistemática em seu ambiente. Além disso, o método Ziegler-Nichols não garante a calibração ideal para todos os casos e, ocasionalmente, requer uma ferramenta auxiliar para ajustar o controlador [5]–[7]. Visto a complexidade desse processo de calibração, diversas abordagens heurísticas foram propostas para o ajuste de um controlador PID.

Ang *et al.* [8] apresentam uma visão geral das tecnologias computacionais que envolvem a calibração de um controlador PID. De acordo com os autores, diversos estudos foram feitos aplicando-se diferentes técnicas de otimização com o objetivo de facilitar e/ou otimizar o ajuste dos ganhos de um controlador PID. Dentre as técnicas, destacam-se: Redes Neurais [9]–[11], Lógica Fuzzy [12]–[14] e Algoritmo Genético (em inglês, *Genetic Algorithm* ou GA) [15]–[17].

Dentre os métodos explorados na literatura para otimização de valores reais, aqueles baseados em Pesquisa de Vizinhança Variável (em inglês, *Variable Neighborhood Search* ou VNS) apresentaram soluções muito próximas ao valor ótimo com um custo computacional moderado em diversos contextos [18]–[20]. Fortes *et al.* [21] propõem o uso do VNS para sintonizar os

G. Silva and A. K. Rêgo Segundo are with the Control and Automation Engeneering, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil 35400-000.

P. Silva, V. Santos, E. Luz and G. Moreira are with the Computing Department, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil 35400-000.

parâmetros de dois controladores de amortecimento suplementares: um estabilizador de sistema de energia e um controlador de fluxo de potência entre malhas, com o objetivo de amortecer sinais locais e os modos de oscilação entre áreas presentes nos dois sistemas de potência estudados. O VNS foi comparado com um método multi-início. Os resultados mostraram que não só o VNS foi mais eficiente, mas também gerou soluções com níveis mais altos de amortecimento. Dessa forma, com base nesses resultados, propõe-se neste artigo a investigação da viabilidade de se aplicar o VNS para a calibração de um controlador PID aplicado a sistemas reais instáveis, como drones, em um cenário de um CPS.

As contribuições deste trabalho podem ser sumarizadas como:

- Proposta de um método, baseado em VNS, para ajustar o PDI de uma planta real (bi-rotor);
- Análise da proposta em um sistema *hardware-in-the-loop*;
- Proposta de uma função objetivo (ou aptidão) eficiente para o problema em questão;
- Comparação da proposta baseada em VNS contra uma alternativa baseada em algoritmos genéticos;

Além desta seção, o trabalho está organizado da seguinte forma: Seção II, trabalhos relacionados; Seção III, problema e visão geral sobre controlador PID; Seção IV, algoritmo VNS e metodologia referente aos algoritmos usados para ajustar o controlador PID; Seção V, algoritmo GA; Seção VI, experimentos da abordagem VNS-PID e GA-PID; Seção VII, apresentação e discussão dos resultados obtidos por meio dos experimentos realizados; e Seção VIII, conclusão e trabalhos futuros.

II. Trabalhos Relacionados

Em relação às técnicas de otimização e calibração de um controlador PID, vários autores contribuíram efetivamente para esse problema.

Angel *et al.* [22] apresentaram uma abordagem *hardware-in-the-loop* para o controle da velocidade de um sistema motor-gerador. Para isso, foram usados um controlador PID e um Algoritmo Genético para sintonizá-lo (PID-GA), com o objetivo de se obter um tempo de ultrapassagem e acomodação específicos. Os autores compararam o modelo obtido no trabalho com um controlador PID ajustado pela técnica *Internal Model Control* (PID-IMC). Os resultados mostraram que o PID-GA apresentou melhor desempenho de rastreamento do que o PID-IMC.

Um ajuste automático das constantes PID usando o *Adaptive Simulated Annealing* (ASA) é aplicado em [23]. Os autores compararam alguns métodos diferentes de ASA com algumas outras técnicas, destacando-se o método de Ziegler-Nichols. Para isso, foram usados 20 sistemas de *benchmark* comumente encontrados no setor industrial para testar cada abordagem. Os testes realizados em quatro cenários do *benchmark* demonstraram a viabilidade do ASA no ajuste de controladores PID para plantas com distúrbios constantes. O algoritmo proposto relacionado ao ASA apresentou melhores resultados em relação ao método Ziegler-Nichols.

Zhang *et al.* [24] empregaram um GA autoajustado para otimizar os parâmetros PID, o que melhora a eficiência da pesquisa global. Os resultados mostraram o potencial do GA para uso colaborativo com os controladores existentes para ajuste do PID.

Um algoritmo de lógica *fuzzy* para otimizar os ganhos de um controlador PID de um sistema elétrico de potência é apresentado em [25]. Os autores utilizaram um modelo matemático baseado em informações estruturais de uma usina hidrelétrica brasileira como base de testes. O algoritmo apresentado foi comparado com um controlador PID convencional implementado na usina hidrelétrica em estudo. Os resultados mostraram que o controlador *fuzzy* apresentou respostas não só mais suaves, mas também menos oscilantes.

Em [26], é apresentado um controle PID neural adaptativo projetado para sistemas configurados em tempo discreto. Esse controlador proporciona uma implementação mais eficiente, ao passo que evita o problema de implementação de controladores projetados para tempo contínuo em sistemas digitais. O controlador ajusta os ganhos PID de forma automática. Dessa forma, qualquer pessoa sem conhecimento prévio da planta é capaz de ajustar tais constantes. Os experimentos foram feitos em um quadrotor e mostraram a eficácia desse método em relação a um PID de ganho fixo clássico, e que o desempenho do primeiro é melhor em relação ao segundo.

III. Descrição do problema

Um sistema instável é uma planta física na qual a posição esperada é diferente da posição natural [27]. Logo, deve-se estabilizá-lo para se atingir a condição esperada.

Como exemplos de sistemas instáveis, tem-se: pêndulo invertido, quadrotores ou qualquer variação com a mesma premissa. Para este trabalho, o foco será um birotor.
A posição natural de um birotor apresenta um motor suspenso e outro encostado no chão. Em contrapartida, a posição esperada se dá quando os dois motores estão alinhados e o sistema permanece paralelo ao solo.

A estabilização da planta requer um sensor para orientar o processo, como um giroscópio, para o caso específico de um birotor. O giroscópio fornece informações que permitem ao microcontrolador medir seu ângulo do sistema em relação ao chão.

A cada período de amostragem do sensor, o microcontrolador calcula o erro em relação ao valor de referência. A partir desse erro, uma saída é gerada levando-se em conta as constantes do controlador PID. Essa saída, ou sinal de controle, visa corrigir o comportamento dos atuadores para que a planta retorne à posição desejada.

Existem várias técnicas de controle para sistemas instáveis presentes na literatura, dentre elas destacam-se o controle de estabilidade de Lyapunov [28] e o controle PID [29], [30]. No entanto, o mais popular e amplamente utilizado é o controlador PID, o qual é aplicado neste trabalho.

Neste trabalho, o birotor é representado por uma estrutura suspensa com dois motores sem escova. Os motores sem escova funcionam como atuadores e são colocados nas bordas junto as hélices, com o objetivo de empurrar o ar para baixo e elevar o eixo. A Fig. 1 mostra um esquema do sistema birotor guiado por um controlador PID, em que $x(t)$ é a referência

em graus, que neste trabalho é 180 (posição horizontal) e $y(t)$ é o ângulo atual do birotor. A diferença entre $x(t)$ e $y(t)$ fornece o erro $e(t)$. Este último é enviado ao controlador PID, que transforma a entrada $x(t)$ em um ganho, controlando a modulação por largura de pulso necessária para definir a velocidade dos motores. O resultado desse ganho altera ou mantém a inclinação do sistema $y(t)$.

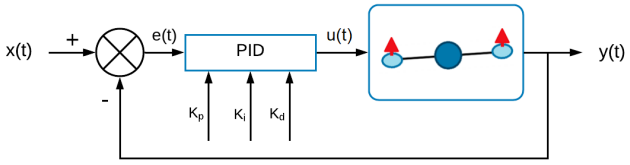

Figura 1. PID aplicado para estabilizar o birotor.

A. Controlador PID

O controlador PID (Fig. 1) é um sistema de controle em malha fechada que fornece uma correção da resposta de uma determinada entrada. Essa correção é baseada em três termos: o proporcional (K_p), o integral (K_i) e o derivado (K_d). O ajuste do PID é a ação de equilibrar essas três constantes com a intenção de melhorar a resposta do sistema. O modelo matemático calculado em cada iteração de um controlador PID é dado por:

$$u(t) = K_p e(t) + K_i \sum_{k=0}^{X} e(t) \quad t + K_d \frac{e(t)}{t}; \qquad (1)$$

em que $u(t)$ é a saída da função PID, $e(t)$ é a variação do erro no intervalo de amostragem e t é o tempo de amostragem do sistema.

A Equação 1 é apresentada esquematicamente na Fig. 2. O controlador PID fornece uma resposta ajustada, $u(t)$, para a planta, a fim de minimizar o erro entre a referência desejada.

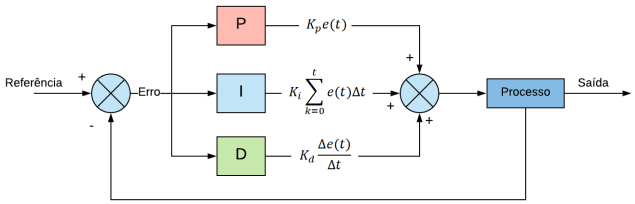

Figura 2. Representação do controle PID digital.

Alguns sistemas apresentam algumas características específicas, como: sobressinal e erro em regime permanente. O sobressinal pode ser definido como a maior diferença entre a posição do birotor e sua resposta em regime permanente e o erro em regime permanente é o erro entre o sistema e sua referência após a estabilização.

Tendo isso em mente, o impacto da alteração de uma das constantes do controlador PID pode alterar uma dessas características específicas, alterando o comportamento do sistema [8]. Por exemplo, o aumento do ganho proporcional (K_p) diminui o tempo de subida, pois a resposta do sistema se torna mais forte e mais rápida a um estímulo externo, uma vez que há um aumento da potência dos atuadores. No entanto, o sobressinal aumenta e o sistema tende a ficar mais instável. Portanto, em um sistema do tipo 0 (sem integrador na função de transferência em malha aberta), o ganho proporcional não poderá levar o erro em regime permanente para zero por si só. Por esse motivo, geralmente, ele é utilizado junto a um termo Integral.

O ganho integral tem uma influência maior na resposta do sistema quando o erro é pequeno e o sistema está próximo da referência. O termo K_i soma o erro ao longo do tempo, para levá-lo a zero. Esse cenário força uma resposta mais abrupta ao longo do tempo, uma vez que o erro é diferente de zero. Por esse motivo, à medida que K_i aumenta, uma pequena diminuição no tempo de subida é observada, o erro em regime permanente diminui e o sobressinal aumenta.

Uma vantagem do controle integral é que ele introduz um termo s no denominador, aumentando o tipo do sistema em uma unidade. O tipo do sistema é dado pelo número de integradores puros presentes no denominador. Caso o sistema seja do tipo 0, o controlador integral elimina o erro estacionário que deveria ocorrer para uma entrada em degrau, se fosse utilizado unicamente o tipo de controlador proporcional. Porém, uma desvantagem deste controlador é a introdução de um polo na origem. Se nenhum zero for introduzido, a diferença entre o número de polos e zeros é aumentada, diminuindo os ângulos das assíntotas dos lugares das raízes. Desse modo, os ângulos das assíntotas apontam mais em direção ao semiplano direito do plano S, reduzindo a estabilidade relativa do sistema. Por isso, a ação integral geralmente não é utilizada de maneira isolada.

Por fim, o ganho derivativo (K_d) exerce grande influência quando o erro apresenta uma variação rápida. Seu aumento pode causar uma diminuição no tempo de subida e no sobressinal. Entretanto, a influência do K_d no erro em regime permanente é quase inexistente quando o sistema está estabilizado, uma vez que a variação do sinal de erro ($e(t)/$) é próxima de zero e, portanto, a contribuição da parcela derivativa é muito pequena.

B. Controlador PID para Sistemas Reais

Um grande desafio na aplicação do PID em sistemas reais é a calibração das constantes, visto que a combinação das mesmas pode levar a um espaço de busca muito grande. Encontrar as melhores constantes, manualmente, pode ser uma tarefa inviável.

Uma alternativa a essa calibração manual é a implementação de técnicas de otimização para automatizar o processo. Essas técnicas promovem um ajuste de forma mais inteligente e rápida, em especial quando comparadas à soluções convencionais. Tendo isso em mente, vários trabalhos na literatura têm mostrado sucesso nessa forma de calibração, como indicam os estudos que utilizam GA e PSO [15], [24], [31], [32]. Contudo, poucos autores exploram técnicas de VNS para a tarefa. Nas próximas seções, uma abordagem baseada em VNS é apresentada para o contexto *hardware-in-the-loop*.

IV. Variable Neighborhood Search para a Calibração de um PID (VNS-PID)

Nesta seção, é apresentada a abordagem proposta, chamada VNS-PID, que aplica o algoritmo VNS ao problema de ajuste dos ganhos PID. O principal objetivo do VNS-PID é sintonizar os parâmetros do controlador PID do birotor.

O VNS é uma meta-heurística que se baseia em mudanças sistemáticas em uma vizinhança [33]. Diferentes vizinhanças são exploradas assumindo-se que um ótimo local em uma delas não é, necessariamente, um ótimo local em outra.

O VNS-PID concentra-se em maximizar a pontuação atribuída a uma solução. Essa pontuação é dada por meio da soma dos valores retornados por uma função de aptidão. Uma solução do VNS-PID consiste em uma combinação dos valores das três constantes PID: K_p, K_i e K_d.

A pontuação de uma solução, em um determinado momento t, é calculada da seguinte forma:

$$g(t) = \frac{x(t) \quad y(t)}{A \quad x(t)} \; ; \tag{2}$$

em que $x(t)$ é a referência, $y(t)$ é o ângulo atual do birotor (conforme descrito na Seção III) e A, o ângulo máximo que o birotor pode alcançar. Já a pontuação final é dada pela seguinte equação:

$$f(s) = 100 \quad \frac{\sum_{t=0}^{T} g(t)}{T}; \tag{3}$$

em que $f(s)$ é a pontuação de uma solução s, $g(t)$ é a pontuação no momento t e T é o tempo de duração do experimento. A função de pontuação de uma solução (ou função de aptidão de uma solução) é uma das contribuições deste trabalho.

O intervalo da função de pontuação final é de zero a 100. A pontuação máxima indica que o birotor está estabilizado desde o início, enquanto a pontuação mínima indica que o birotor não saiu de sua posição de origem. Neste trabalho, o cálculo da função de pontuação final é iniciado desde o início da avaliação da solução, portanto, a solução candidata é penalizada durante o período em que o sistema está iniciando. Essa é uma maneira de permitir que o sistema priorize soluções com menor sobressinal e tempo de estabilização. Com isso, é impossível atingir a pontuação máxima, tendo em vista que o sistema inicia com um dos motores na posição inferior (posição natural) e precisa de um período para atingir a referência, que é de 180 graus. Decidiu-se começar nessa posição para garantir que todas as soluções comecem nas mesmas condições.

O Algoritmo 1 apresenta o pseudo-algoritmo VNS-PID, que se assemelha a uma implementação básica do VNS, todavia, o processo de busca local utiliza o conceito de *first improvement*, ou seja, quando é encontrada uma solução melhor que a atual: (i) o algoritmo interrompe a busca local atual, (ii) o valor de k no Algoritmo 1 é definido como 1, (iii) o intervalo da busca local é dividido por dois e (iv) a próxima iteração é iniciada. Decidiu-se por essa abordagem devido ao alto custo relacionado à avaliação de uma única solução e ao estreitamento da vizinhança avaliada, o que a aproxima de um máximo local. Além disso, a variável i foi adicionada, a qual é responsável por restringir o número máximo de iterações. Dessa forma, os critérios de parada para a abordagem proposta são o número máximo de pesquisas locais sem aprimoramento da solução (k) e o número máximo de pesquisas locais a serem executadas (i). Se qualquer um dos dois critérios for atendido, o algoritmo encerra sua execução.

A estrutura de vizinhança usada neste trabalho consiste na adição de um valor aleatório diferente (negativo ou positivo) para cada combinação das constantes PID. Uma nova solução é considerada uma solução viável se todas as constantes respeitarem as restrições. Se uma restrição for violada, a constante é arredondada para o valor mais próximo no intervalo (máximo ou mínimo).

Algoritmo 1: Algoritmo VNS-PID.

```
1:  Procedure: Neighborhoods N^1; : : : ; N^m.
2:  s    generateSolution()
3:  k    1
4:  range    [ 0:4 0:4]
5:  while k    m do
6:     s'    randomNeighbor(s; N^k; range)
7:     s     localSearch(s'; range' = range=2)
8:     if f(s ) > f(s) then
9:        s    s
10:       k    1
11:       range    range'
12:    else
13:       k    k + 1
14:       range    [ 0:4 0:4]
15: return: s
```

V. Algoritmo Genético para a Calibração de um PID (GA-PID)

Nesta seção, é apresentada a abordagem proposta, chamada GA-PID, que aplica o algoritmo genético ao problema de ajuste dos ganhos PID e que utiliza da mesma função de pontuação aplicada no VNS (Seção IV).

O algoritmo genético [34], [35] foi desenvolvido com base na teoria de seleção natural de Darwin. Portanto, ele simula a sobrevivência do indivíduo mais apto (uma solução possível) entre uma população (um grupo de soluções possíveis) ao longo das gerações. Nesse contexto, os indivíduos mais aptos possuem uma tendência maior de se reproduzir e propagar suas características. Esse algoritmo possui quatro operações principais: (i) seleção, (ii) cruzamento, (iii) mutação e (iv) elitismo.

Neste trabalho, a seleção ocorre por meio de um torneio binário. O torneio binário consiste na seleção do melhor indivíduo (o que tiver maior aptidão) entre dois indivíduos diferentes, escolhidos de forma aleatória. Essa abordagem aumenta a probabilidade de se propagar as melhores características para as gerações futuras, mas mantendo a diversidade da população.

O cruzamento é um método para produzir descendentes combinando dois indivíduos diferentes, os quais foram selecionados em dois torneios binários diferentes. Para o contexto deste trabalho, em que as características do indivíduo são números reais, o cruzamento (S_g) ocorre de acordo com a seguinte equação:

$$S_g = \quad S_1 + (1 \qquad)S_2; \tag{4}$$

em que S_1 é uma das constantes PID do primeiro indivíduo selecionado, é um número aleatório entre 0,0 e 1,0 e S_2 é uma das constantes PID do outro indivíduo.

A mutação é um mecanismo de variabilidade genética, que permite alterar uma característica específica de um indivíduo de forma aleatória e ocasional. Esse mecanismo evita que o algoritmo fique estacionado em um máximo local, visto que é possível explorar outras regiões do espaço de busca. O operador de mutação pode ser definido de acordo com a seguinte equação:

$$S_m = S + \quad ; \tag{5}$$

em que S é uma das constantes PID do indivíduo selecionado e é um número aleatório entre 0,0 e 1,0.

O Algorítimo 2 apresenta o pseudocódigo do GA utilizado neste trabalho. O algoritmo gera um número pré-definido de indivíduos, por meio da função *generatePopulation* e então os avalia por meio da função *evaluateIndividual*. Se a probabilidade for menor que a definida para cruzamento, é feito o torneio binário duas vezes: um para selecionar o primeiro pai e outro para selecionar o segundo pai. O torneio binário é feito pela função *tournament*, que pega dois indivíduos de forma aleatória e retorna o melhor entre eles. O torneio é feito até que os pais sejam indivíduos diferentes. Feito isso, a função *crossover* gera um novo indivíduo resultado da mistura genética dos pais. Se um valor aleatório gerado for menor que a probabilidade de mutação, o filho sofre uma mutação por meio da função *mutation*. Essa função adiciona de forma aleatória um valor entre 0 e 1 em cada uma das características genéticas do filho (constantes PID). Após a etapa de cruzamento, o filho é avaliado. Com isso, ao final da geração, tem-se o dobro da população inicial. Porém, apenas metade dessa população fará parte da população da próxima geração. Essa geração é composta pelos melhores indivíduos da geração anterior (pais e filhos). Esse processo é chamado de elitismo e é feito pela função *elitism*. Após a conclusão de todas as gerações, o algoritmo retorna o melhor indivíduo.

Algoritmo 2: Algoritmo GA-PID.

```
s[]  generatePopulation();
evaluateIndividual();
for i = 0; i < M; i++ do
   while j < N do
      probability  random(0,1);
      if probability < 0.95 then
         j++;
         s'  tournament(s[]);
         while s' != s'' do
            s''  tournament(s[]);
         s  crossover(s', s'');
         probability  random(0,1);
         if probability < 0.05 then
            s  mutation(s);
         evaluateIndividual();
   s[]  elitism();
return: s[0]
```

VI. Experimentos

Nesta seção, discute-se a configuração dos experimentos propostos, que é separada em duas partes: hardware e software.

A. Hardware do Birotor

O sistema do Birotor usado neste trabalho consiste em uma estrutura de madeira maciça, em formato de "U" usada como base do sistema, mantendo-o no chão enquanto os experimentos são executados. Uma barra de metal cilíndrica passa através das extremidades da estrutura de madeira e é fixada por porcas. Essa barra conecta a estrutura ao birotor. O esquema do protótipo é apresentado na Fig. 3 e a calibração do PID em execução no modelo real pode ser visto em https://drive.google.com/file/d/1DqXWo-F6qARl65lYRxBYxZtGl1Fhrh21/view?usp=sharing.

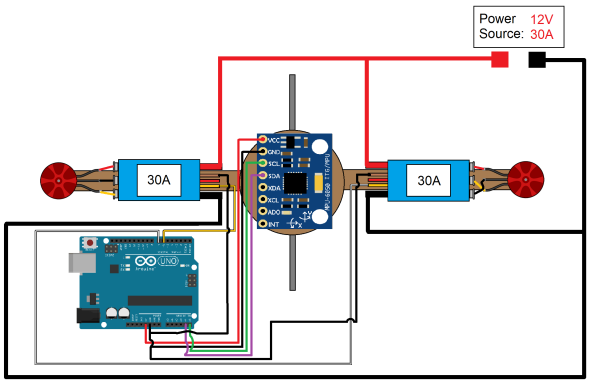

Figura 3. Esquema proposto do birotor.

Um motor de corrente contínua (CC) sem escova de 5 Volts com uma hélice de nylon e carbono 8045 é fixado em cada uma das extremidades da estrutura. Para alimentá-los, utiliza-se uma fonte de alimentação CC que fornece os 12 Volts e 30 Amperes exigidos pelo sistema. As hélices produzem um impulso ascendente que desequilibra todo o sistema devido ao seu torque. Portanto, os dois motores sem escova têm diferentes direções de rotação para compensar a reação de torque um do outro.

O microcontrolador usado é o ATmega 328p, que trabalha com a frequência de 16 MHz, juntamente com os módulos necessários para ler os sensores e enviar os sinais dos motores. O sensor utilizado neste trabalho é o giroscópio[1], que fornece as informações necessárias para que o microcontrolador possa medir a inclinação do birotor em relação ao solo.

A superfície tem uma grande importância para o ajuste das constantes PID, visto que a avaliação de todos os indivíduos é baseada no ângulo em que o sistema está em relação à superfície. Por esse motivo, os testes são feitos no chão em um lugar plano.

[1]As especificações do giroscópio pode ser encontrado em: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3\%204.pdf.

B. Algoritmo do VNS-PID

Conforme descrito na Seção IV, a solução do sistema é um conjunto de três constantes PID: K_p, K_i e K_d. Neste trabalho, as constantes PID são limitadas entre 0,0 e 7,0. O limite superior é definido como 7,0 por meio de experimentos empíricos.

O ângulo mais alto alcançado pelo birotor é de 220 graus e a referência é ajustada para 180 graus. A função de aptidão para cada solução possível é a soma da pontuação obtida em 6000 ciclos—duração suficiente para permitir que o sistema se estabilize.

No VNS-PID apresentado na Seção IV, o vetor da solução armazena quatro informações diferentes: K_p, K_i, K_d e a aptidão da solução. O número máximo de pesquisas locais sem aprimoramento (k) e o número máximo de pesquisas locais a serem executadas (i) são definidas para 20 e 50, respectivamente.

Além disso, a estrutura de vizinhança é descrita como a adição de um valor aleatório diferente, inicialmente, no intervalo $[\ 0{,}4\ ;\ 0{,}4]$ para cada constante PID. Além disso, há sete estruturas de vizinhança diferentes: (i) adição em K_p; (ii) adição em K_i; (iii) adição em K_d; (iv) adição em K_p e K_i; (v) adição em K_p e K_d; (vi) adição em K_i e K_d; e (vii) adição em K_p, K_i e K_d.

A função *generateSolution* gera 50 soluções aleatórias e retorna a melhor. A função *randomNeighbor* retorna um vizinho (s') de uma solução (s) adicionando um valor aleatório no intervalo de vizinhança atual para cada constante PID. Por fim, a função *localSearch* busca 20 novas soluções aleatórias na estrutura de vizinhança s' por meio da adição de um valor aleatório em seu intervalo atual para cada constante PID (semelhante ao *randomNeighbor*). A estrutura de vizinhança será uma das sete descritas anteriormente, dependendo do valor de k. Além disso, aplica-se o *first improvement* na função *localSearch* para reduzir o número de avaliações.

C. Algoritmo do GA-PID

No GA-PID apresentado na Seção V, o vetor da solução armazena quatro informações diferentes: K_p, K_i, K_d e a aptidão da solução. Porém, ao contrário do VNS-PID, em que se trabalha com uma solução apenas, no GA, trabalha-se com um conjunto de soluções que evoluem com o passar das gerações. O tamanho da população definiu-se como 20 indivíduos e o número de gerações máximas como 15. Esses valores foram calculados de acordo como número de soluções média que os testes do VNS-PID avaliaram. A média das avaliações de soluções foi de 300. Dessa forma, a comparação entre os dois algoritmos é mais justa.

VII. Resultados e Discussão

Nesta seção, os resultados alcançados com a abordagem proposta são apresentados e discutidos.

Na Tabela I, são apresentados os resultados de dez execuções diferentes dos experimentos realizados com o VNS-PID, enquanto que na Tabela II, os experimentos com o GA-PID. A "Primeira solução" é a solução de saída da função *generateSolution* e a "Melhor solução" é a encontrada pelo método após o fim do seu processamento.

Nos testes apresentados nas Tabelas I e II, nota-se que os resultados apresentam valores abaixo de 95%. Esse fato é aceitável, uma vez que o birotor inicia com um motor na parte superior e outro na parte inferior. Com isso, o sistema precisa de um tempo para estabilizar até chegar na referência, conforme descrito na Seção IV. Com isso, o algoritmo é penalizado desde o início.

Ressalta-se que, como pode ser visto na Tabela I, a "Melhor solução" de cada execução é diferente entre si, o que significa que o algoritmo pesquisou vizinhanças distintas e, consequentemente, encontrou soluções diferentes. Além disso, é possível verificar que o VNS-PID melhora a solução inicial, na maioria dos casos, exceto na quarta execução, na qual a solução inicial e a final são iguais. Isso também acontece nas execuções do GA-PID, uma vez que, conforme pode ser observado na Tabela II, o teste 10 não melhora. Vale ressaltar que os valores de aptidão na Tabela I apresentam uma média igual a 87,73 e um desvio padrão igual a 1,63, enquanto os valores referentes à Tabela II apresentam uma média de 85,65 e desvio padrão de 2,97. Observa-se que a média das execuções do GA é menor, ou seja, as soluções que o GA-PID encontra geralmente são piores que aquelas geradas pelo VNS-PID. Além disso, pelo fato de o VNS-PID apresentar um desvio padrão menor do que o GA-PID, o primeiro demonstra ter uma variabilidade menor em seus resultados, isto é, as soluções finais possuem valores de aptidão mais próximos que os do GA-PID e a uma menor tendência em gerar soluções com uma aptidão muito inferior as encontradas.

No que se refere às soluções encontradas pelos métodos, verifica-se que seus valores de K_p estão abaixo de 0,7. Isso se deve ao fato de que a estrutura que contém os motores é leve, logo, não há necessidade de uma resposta muito forte dos atuadores para mover a estrutura do birotor. Por esse motivo, um ganho proporcional alto levaria o sistema a um comportamento instável, uma vez que o sobressinal também seria alto e os ganhos derivativos e integrais não seriam capazes de estabilizar o sistema.

Além disso, o VNS-PID encontrou diferentes tipos de controle: PI e PD. Um evento semelhante ocorreu com o GA-PID: encontrou-se uma solução com tipo de controle PD. Um controle PI ocorre quando o valor de K_d é zero, ou seja, o sistema tem uma resposta mais lenta a variações do que um controle PID normal, pois o ganho derivado é mais sensível a grandes variações na entrada do sistema. Um controle PD, em contra partida, ocorre quando o K_i é zero, o que significa que o sistema apresenta um erro maior no estado estacionário do que um controle PID devido à falta de ação integral. Na aplicação real, o controlador PI apresentou uma resposta subamortecida. Além disso, o sistema ficou mais lento, se comparado com as demais soluções encontradas nos testes feitos com o VNS-PID. Já para os controladores PD sintonizados pelo VNS-PID e GA-PID, a resposta do sistema foi mais rápida se comparada com ao do controlador PI. Por outro lado, eles apresentaram maior erro em regime permanente se comparado com as demais soluções encontradas. Com isso, observa-se que a utilização dos

Tabela I
RESULTADOS DAS DEZ EXECUÇÕES DO VNS-PID NO BIROTOR.

Execução	Primeira solução				Melhor solução				
	K_p	K_i	K_d	Aptidão	K_p	K_i	K_d	Aptidão	Tempo (min)
1	0,46	2,92	3,95	74,54	0,50	1,67	3,95	83,70	82
2	0,52	0,04	0,64	84,97	0,61	0,00	0,64	87,06	81
3	0,57	1,65	2,06	83,81	0,64	1,03	2,49	89,60	118
4	0,65	1,82	0,77	89,50	0,65	1,82	0,77	89,50	51
5	0,55	0,43	0,36	86,28	0,45	0,45	0,00	88,33	95
6	0,48	0,99	1,19	82,99	0,48	0,99	1,19	87,67	64
7	0,53	2,55	3,20	83,86	0,54	2,14	2,27	86,48	90
8	0,56	1,26	4,53	87,02	0,67	1,16	4,30	87,94	96
9	0,46	0,15	0,07	83,35	0,49	0,31	0,71	88,59	117
10	0,59	1,22	4,65	87,86	0,62	1,17	4,31	88,46	88

Tabela II
RESULTADOS DAS DEZ EXECUÇÕES DO GA-PID NO BIROTOR.

Execução	Primeira solução				Melhor solução			
	K_p	K_i	K_d	Aptidão	K_p	K_i	K_d	Aptidão
1	0,95	0,11	0,66	82,76	0,30	0,44	1,28	84,77
2	0,80	0,07	2,65	80,79	0,49	0,08	2,14	82,58
3	0,31	0,42	0,60	79,85	0,31	0,18	0,60	84,45
4	0,30	0,10	2,98	82,12	0,42	0,12	2,84	88,91
5	0,30	0,11	0,34	71,13	0,41	0,00	0,60	84,30
6	0,75	0,47	0,25	84,61	0,41	0,13	0,11	90,85
7	0,48	0,80	0,27	76,31	0,30	0,12	3,26	80,10
8	0,33	0,17	1,10	80,03	0,33	0,15	1,04	86,08
9	0,67	0,07	3,51	83,63	0,34	0,04	2,37	87,99
10	0,51	0,09	4,50	86,44	0,51	0,09	4,50	86,44

controladores PI e PD sintonizados pelos algoritmos propostos não são vantajosas quando comparadas aos controladores PID sintonizados pelos algoritmos.

No que tange o tempo de execução de cada método, na Tabela I, apresenta-se o tempo em minutos que cada teste do VNS-PID demorou para encontrar a solução. Verifica-se que o tempo varia de acordo com o teste. Esse evento ocorre porque utiliza-se o conceito de *first improvement* e, por isso, a busca local nem sempre encontra uma solução melhor nos mesmos instantes (conforme descrito na Seção VI). Há uma diferença considerável entre o tempo de execução dos testes dois e três, todavia, há uma proximidade muito alta entre os valores de aptidão.

Além disso, ressalta-se que o teste três, que possui o menor tempo, não apresentou melhoria entre a primeira solução e a encontrada pelo VNS-PID. Por esse motivo, seu tempo de execução foi o menor, uma vez que as buscas locais feitas em cada uma das vizinhanças não encontraram soluções melhores e ficaram presas em um ótimo local. O tempo de execução do GA-PID, ao contrário do VNS-PID, é fixo, demorando 90 minutos para encontrar a solução final. Para o contexto de um sistema embarcado, caso fosse necessário recalcular os ganhos, no caso do VNS-PID, levaria em média, 88 minutos, enquanto que o GA-PID levaria 90.

Ao avaliar o consumo de memória, o GA-PID apresenta uma grande desvantagem em um cenário embarcado com memória restrita (2 Mb). Para o contexto deste trabalho, ele consome 640 bytes de memória para lidar com a população e suas operações. O VNS-PID, por outro lado, consome apenas 4 bytes, uma vez que o algoritmo trabalha com apenas uma única solução. Desta forma, o VNS-PID possui uma vantagem para a sintonia do PID em sistemas embarcados de baixo poder computacional.

Na Figura 4 apresenta-se o comportamento da primeira solução da terceira execução, a qual apresentou a melhor solução gerada, e a solução otimizada pela abordagem proposta VNS-PID (consulte Tabela I). É possível verificar que o termo K_i da primeira solução diminuiu em relação à solução gerada. Os termos K_p e K_d, por outro lado, aumentaram. Dessa forma, conforme discutido na Seção III, quando o ganho proporcional (K_p) e o ganho derivativo (K_d) aumentam, o sobressinal também aumenta. Por esse motivo, o sobressinal da solução aumentou, como mostra a Fig. 4. Além disso, é possível verificar que a solução encontrada pela abordagem proposta VNS-PID está mais próxima da referência que a primeira solução, ou seja, seu erro em regime permanente é muito inferior, visto que a primeira solução se encontra a sete graus da referência.

A Figura 5 apresenta o comportamento da primeira solução do sexto teste do GA-PID, em relação à solução encontrada por ele. Por meio da Tabela II, verifica-se que o valor das constantes proporcional e integral da primeira solução são muito elevadas em relação às constantes proporcional e integral da solução encontrada. Por esse motivo, a primeira solução apresenta um sobressinal elevado.

VIII. CONCLUSÃO

Neste artigo, é proposto o uso da meta-heurística VNS para a calibração dos parâmetros de um controlador PID

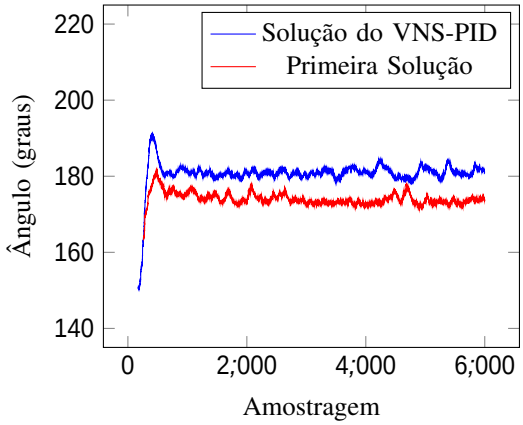

Figura 4. Comparando a primeira solução com a solução gerada pelo VNS-PID.

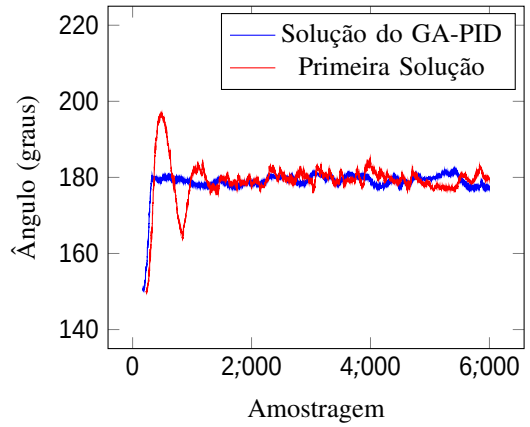

Figura 5. Comparando a primeira solução com a solução gerada pelo GA-PID.

com o objetivo de estabilizar um sistema birotor em uma abordagem *hardware-in-the-loop*. Com relação às contribuições, destacamos a função de aptidão proposta. Ela se mostrou viável, juntamente com uma abordagem baseada em VNS, para estabilizar um sistema ciber-físico real, permitindo um ajuste fino, *in loco*, e barato.

Experimentos mostram a viabilidade da aplicação do VNS para otimizar as constantes PID da planta de um birotor. A abordagem proposta é capaz de gerar soluções com alto valor de aptidão (superior a 87% na média), mesmo usando pouca memória (4 bytes).

A abordagem proposta tem potencial para ser usada em sistemas reais ou comerciais, de forma a ajustar o controlador PID a novos ambientes ou mesmo modificações de projeto de última hora.

AGRADECIMENTOS

Os autores gostariam de agradecer a UFOP, CAPES, CNPq e FAPEMIG.

REFERÊNCIAS

[1] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.

[2] R. Hernández-Alvarado, L. García-Valdovinos, T. Salgado-Jiménez, A. Gómez-Espinosa, and F. Fonseca-Navarro, "Neural network-based self-tuning PID control for underwater vehicles," *Sensors*, vol. 16, no. 9, p. 1429, 2016.

[3] P. Stewart, D. Stone, and P. J. Fleming, "Design of robust fuzzy-logic control systems by multi-objective evolutionary methods with hardware in the loop," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 3, pp. 275–284, 2004.

[4] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[5] K. Ogata and Y. Yang, *Modern control engineering*. Prentice-Hall, 2002, vol. 4.

[6] L. Castro, G. Amorim, A. Silveira *et al.*, "Design of PID Type Local Controller Network with Fuzzy Supervision," *IEEE Latin America Transactions*, vol. 17, no. 05, pp. 759–765, 2019.

[7] G. V. Lima, R. M. J. A. de Souza, A. S. de Morais, L. C. Oliveira-Lopes, and G. M. V. Ladeira, "Stabilization and path tracking of a mini quadrotor helicopter: experimental results," *IEEE Latin America Transactions*, vol. 17, no. 03, pp. 485–492, 2019.

[8] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[9] H. Shu and Y. Pi, "PID neural networks for time-delay systems," *Computers & Chemical Engineering*, vol. 24, no. 2-7, pp. 859–862, 2000.

[10] Y. Zhao, X. Du, G. Xia, and L. Wu, "A novel algorithm for wavelet neural networks with application to enhanced PID controller design," *Neurocomputing*, vol. 158, pp. 257–267, 2015.

[11] M. Fang, Y. Zhuo, and Z. Lee, "The application of the self-tuning neural network PID controller on the ship roll reduction in random waves," *Ocean Engineering*, vol. 37, no. 7, pp. 529–538, 2010.

[12] A. Visioli, "Tuning of PID controllers with fuzzy logic," *IEE Proceedings-Control Theory and Applications*, vol. 148, no. 1, pp. 1–8, 2001.

[13] A. Fereidouni, M. A. Masoum, and M. Moghbel, "A new adaptive configuration of PID type fuzzy logic controller," *ISA Transactions*, vol. 56, pp. 222–240, 2015.

[14] J. E. R. Castellanos and J. E. C. Ballesteros, "Implementation of a Direct Fuzzy Controller Applied to a Helicopter with one Degree of Freedom," *IEEE Latin America Transactions*, vol. 17, no. 11, pp. 1808–1814, 2019.

[15] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 78–82, 2001.

[16] J. Amaral, R. Tanscheit, and M. Pacheco, "Tuning PID controllers through genetic algorithms," *Complex Systems*, vol. 2, p. 3, 2018.

[17] S. Kumari, P. Prince, V. K. Verma, B. Appasani, and R. K. Ranjan, "GA Based Design of Current Conveyor PLD Controller for the Speed Control of BLDC Motor," in *2018 4th International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE, 2018, pp. 1–3.

[18] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[19] J. Brimberg, N. Mladenović, R. Todosijević, and D. Urošević, "Solving the capacitated clustering problem with variable neighborhood search," *Annals of Operations Research*, vol. 272, no. 1-2, pp. 289–321, 2019.

[20] J. Pei, Z. Dražić, M. Dražić, N. Mladenović, and P. M. Pardalos, "Continuous variable neighborhood search (C-VNS) for solving systems of nonlinear equations," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 235–250, 2019.

[21] E. de Vargas Fortes, L. H. Macedo, P. B. de Araujo, and R. Romero, "A VNS algorithm for the design of supplementary damping controllers for small-signal stability analysis," *International Journal of Electrical Power & Energy Systems*, vol. 94, pp. 41–56, 2018.

[22] L. Angel, J. Viola, and M. Vega, "Hardware in the loop experimental validation of PID controllers tuned by genetic algorithms," in *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*. IEEE, 2019, pp. 1–6.

[23] L. F. Fraga-Gonzalez, R. Q. Fuentes-Aguilar, A. Garcia-Gonzalez, and G. Sanchez-Ante, "Adaptive simulated annealing for tuning PID controllers," *AI Communications*, vol. 30, no. 5, pp. 347–362, 2017.

[24] J. Zhang, J. Zhuang, H. Du *et al.*, "Self-organizing genetic algorithm based tuning of PID controllers," *Information Sciences*, vol. 179, no. 7, pp. 1007–1018, 2009.

[25] C. Osinski, G. V. Leandro, and G. H. da Costa Oliveira, "Fuzzy PID controller design for LFC in electric power systems," *IEEE Latin America Transactions*, vol. 17, no. 01, pp. 147–154, 2019.