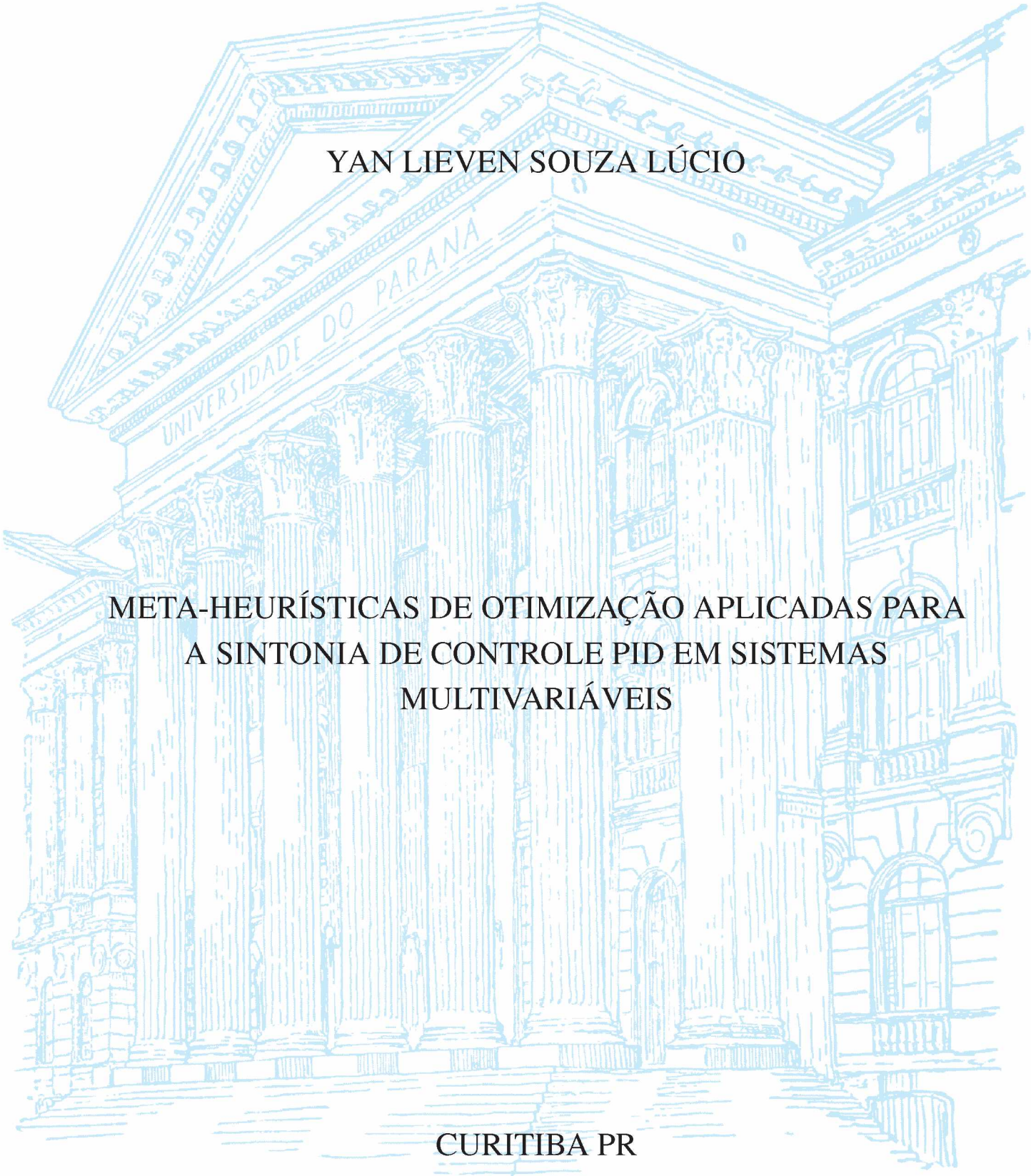
UNIVERSIDADE FEDERAL DO PARANÁ

YAN LIEVEN SOUZA LÚCIO



META-HEURÍSTICAS DE OTIMIZAÇÃO APLICADAS PARA A SINTONIA DE CONTROLE PID EM SISTEMAS MULTIVARIÁVEIS

CURITIBA PR
2022

YAN LIEVEN SOUZA LÚCIO

META-HEURÍSTICAS DE OTIMIZAÇÃO APLICADAS PARA A SINTONIA DE CONTROLE PID EM SISTEMAS MULTIVARIÁVEIS

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Engenharia Elétrica, setor de tecnologia, no Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), da Universidade Federal do Paraná.

Área de concentração: *Sistemas Eletrônicos*.

Orientador: Prof. Dr. Leandro dos Santos Coelho.

CURITIBA PR
2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Lúcio, Yan Lieven Souza
Meta-heurísticas de otimização aplicadas para a sintonia de controle PID em sistemas multivariáveis / Yan Lieven Souza Lúcio. – Curitiba, 2022.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica.

Orientador: Leandro dos Santos Coelho

1. Controladores PID. 2. Otimização matemática. 3. Heurística. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Engenharia Elétrica. III. Coelho, Leandro dos Santos. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **YAN LIEVEN SOUZA LÚCIO** intitulada: ***Meta-heurísticas de otimização aplicadas para a sintonia de controle PID em sistemas multivariáveis*** , sob orientação do Prof. Dr. LEANDRO DOS SANTOS COELHO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 04 de Agosto de 2022.

Assinatura Eletrônica
09/08/2022 23:31:59.0
LEANDRO DOS SANTOS COELHO
Presidente da Banca Examinadora

Assinatura Eletrônica
05/08/2022 12:45:49.0
GIDEON VILLAR LEANDRO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
05/08/2022 13:22:34.0
MARLIO BONFIM
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
05/08/2022 10:53:31.0
GILBERTO REYNOSO MEZA
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ)

À minha família

Agradecimentos

Quero agradecer às pessoas que me apoiaram e foram fonte de força durante esses anos difíceis de estudo e pandemia: Minha mãe Delvais, por ter lutado por muito tempo para que eu pudesse estudar, meu irmão Yuri por ser meu melhor amigo, e minha namorada Steffi por estar comigo nos melhores e piores momentos. Também devo agradecer ao meu orientador Leandro, que ajudou a desenvolver meu pensamento acadêmico e que me ensinou detalhes de pesquisa científica dos quais eu nem sabia. Agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por me auxiliar financeiramente no período que estive trabalhando nessa dissertação.

RESUMO

Na indústria busca-se obter o melhor desempenho para as mais variadas aplicações, utilizando-se na maior parte dos casos sintonia fina para controladores PID (Proporcional-Integral-Derivativo) clássico. Entretanto, o uso de métodos de sintonia fina não extraem de fato o melhor comportamento do sistema em casos mais complexos, como em sistemas não-lineares, ou com múltiplas entradas e múltiplas saídas (do inglês *Multiple Input and Multiple Output*, MIMO), sendo possível a melhora desse desempenho por meio de técnicas mais avançadas. Além disso, o PID possui ainda variantes com diferentes estruturas e capazes de realizar um controle com mais capacidade de adaptação a depender das condições de cada aplicação. Algumas variantes do PID são o PID fracionário (do inglês *Fractional Order PID*, FOPID), baseado em cálculo fracionário, e o PID adaptativo (do inglês *Adaptive PID*, APID), com capacidade de adaptação em tempo real. O problema de se otimizar uma configuração de controlador PID pode ser abordado por meta-heurísticas, técnicas de otimização baseadas em múltiplos agentes evoluindo ao longo de várias iterações. Nesse contexto diversos trabalhos já foram feitos, e predominam o uso do algoritmo genético (do inglês *Genetic Algorithm*, GA) e da otimização por enxame de partículas (do inglês *Particle Swarm Optimization*, PSO). Há uma lacuna referente ao uso de meta-heurísticas mais recentemente desenvolvidas para a sintonia de controladores PID dos três tipos mencionados. O objetivo desse trabalho é o de se avaliar o uso de 3 tipos de controladores PID quando sintonizados por 5 meta-heurísticas: GA, PSO, otimização de gavião asa-de-telha (do inglês *Harris Hawks Optimization*, HHO), algoritmo de predadores marítimos (do inglês *Marine Predators Algorithm*, MPA), e otimização da arraia (do inglês *Manta Ray Optimization*, MRO). Os três sistemas (processos) utilizados nos testes para avaliar o desempenho das metaheurísticas de otimização e sistemas de controle em malha fechada foram: a coluna de destilação de *Wood* e *Berry*, a coluna de destilação de *Ogunnaike* e *Ray*, e o sistema de pulverização de moinho de bolas (do inglês *Ball Mill Pulverizing System*, BMPS). Para cada uma das configurações feitas, foram executadas 50 rodadas dos algoritmos tendo como função objetivo a integral do tempo multiplicado ao erro ao quadrado (do inglês *Integral of Time Multiply squared Error*, ITSE), com o objetivo de se analisar estatisticamente os resultados obtidos. Ao final desse processo, ficou evidente que as técnicas são eficientes no geral, e ao se avaliar os desempenhos e fazer uma classificação (baseada em menor valor de ITSE) em cada estudo de caso, gerou-se ao final uma classificação geral somando as classificações anteriores: MPA em primeiro, GA e HHO em segundo, PSO em terceiro e MRO em quarto. Com relação aos tipos de controlador, o FOPID foi o mais eficiente, a não ser na aplicação com maior número de parâmetros, enquanto que o APID teve bom desempenho, mas foi mais inconsistente, devido à sua natureza de tempo-real. O PID clássico foi inferior às outras variantes em todas as aplicações, com exceção da coluna de Ogunnaike e Ray (OR).

Palavras-chave: Controlador proporcional-integral-derivativo, Otimização, Múltiplas entradas e múltiplas saídas, meta-heurísticas de otimização.

ABSTRACT

The industry seeks to obtain the best performance for the most varied applications, using in most cases fine tuning for classical Proportional-Integral-Derivative (PID) controllers. However, the use of fine-tuning methods do not actually extract the best behavior from the system in more complex cases, such as in non-linear systems, or with multiple inputs and multiple outputs (MIMO), being possible to improve this performance through more advanced techniques. In addition, the PID also has variants with different structures and capable of performing a control with more adaptability depending on the conditions of each application. Some variants of PID are Fractional Order PID (FOPID), which is based on fractional calculation, and Adaptive PID (APID), with real-time adaptability. The problem of optimizing a PID controller configuration can be addressed by metaheuristics, optimization techniques based on multiple agents evolving over several iterations. In this context, several works have already been done, and the use of the genetic algorithm (GA) and the Particle Swarm Optimization (PSO) predominates. There is a gap regarding the use of more recently developed metaheuristics for tuning PID controllers of the 3 types mentioned. The objective of this work is to evaluate the use of 3 types of PID controllers when tuned by 5 metaheuristics: GA, PSO, Harris Hawks Optimization (HHO), Marine Predators Algorithm (MPA), and Manta Ray Foraging Optimization (MRFO). The 3 systems (processes) used in the tests to evaluate the performance of the optimization metaheuristics and closed-loop control systems were: the Wood and Berry distillation column, the Ogunnaike and Ray distillation column, and the Ball Mill Pulverizing System (BMPS). For each of the configurations, 50 rounds of the algorithms were performed by having the time integral multiplied by the squared error (ITSE) as the objective function, in order to statistically analyze the results obtained. At the end of this process, it became evident that the techniques are efficient in general, and after evaluating their performances and making a classification (based on lower ITSE values) in each case study, a final general classification was generated by adding the previous classifications: MPA first, GA and HHO second, PSO third, and MRO fourth. Regarding controller types, FOPID was the most efficient, except in the application with the highest number of parameters, while APID performed well, but was more inconsistent, due to its real-time nature. The classical PID was inferior to the other variants in all applications with the excpetion of the column of Ogunnaike and Ray (OR).

Keywords: Proportional-Integral-Derivative Controller, Optimization, Multiple Input and Multiple Output, Optimization Metaheuristics.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ABC	*Artificial Bee Colony*
ACO	*Ant Colony Optimization*
APID	*Adaptive PID*
BA	*Bat Algorithm*
BBBCA	*Big Bang Big Crunch Algorithm*
BFA	*Bacteria Foraging Algorithm*
BLT	*Big Log Modulus Tuning*
BMPS	*Ball Mill Pulverizing System*
CBIC	Congresso Brasileiro de Inteligência Computacional
CBO	*Cloud-Based Optimization*
CFO	*Central Force Optimization*
CLP	Controlador Lógico Programável
CMAES	*Covariance Matrix Adaptation Evolution Strategy*
COBEM	Congresso Internacional de Engenharia Mecânica
CS	*Cuckoo Search*
CSA	*Crow Search Algorithm*
DE	*Differential Evolution*
DMC	*Dynamic Matrix Control*
DRGA	*Dynamic Relative Gain Array*
DSA	*Differential Search Algorithm*
EHO	*Elephant Herding Optimization*
ELA	*Electromagnetism-Like Algorithm*
EP	*Evolutionary Programming*
ES	*Evolution Strategy*
FA	*Firefly Algorithm*
FFRD	*Finite Frequency Response Data*
FGA	*Football Game Algorithm*
FLC	*Fuzzy Logic Controller*
FOPID	*Fractional Order PID*
FOFPID	*Fractional Order Fuzzy PID*
FPID	*Fuzzy PID*
FSO	*Fish Swarm Optimization*
GA	*Genetic Algorithm*
GNPSO	*Grey NLJ Particle Swarm Optimization*
GO	*Grasshopper Optimization*
GP	*Genetic Programming*
GSA	*Gravitational Search Algorithm*
GWO	*Grey Wolf Optimizer*
HHO	*Harris Hawks Optimization*

HS	*Harmony Search*
HTS	*Heat Transfer Search*
IAE	*Integral of Absolute Error*
ICA	*Imperialist Competitive Algorithm*
IMC	*Internal Mode Control*
IMCPID	*Internal Model Control PID*
ISE	*Integral of Squared Error*
ITAE	*Integral of Time Multiply Absolute Error*
ITSE	*Integral of Time Multiply Squared Error*
INVLAP	*Numerical Inverse Laplace Transform Algorithm*
IWDA	*Intelligent Water Drops Algorithm*
KGMO	*Kinetic Gas Molecule Optimization*
LMI	*Linear Matrix Inequality*
LQR	*Linear Quadratic Regulator*
MCIWO	*Modified Chaotic Invasive Weed optimization*
MIMO	*Multiple-Input Multiple-Output*
MOA	*Magnetic Optimization Algorithm*
MPA	*Marine Predators Algorithm*
MPC	*Model Predictive Control*
MPSO	*Modified Particle Swarm Optimization*
MRO	*Manta Ray Optimization*
NLJ	*New Luus-Jaakola*
PID	Controlador Proporcional-Integral-Derivativo
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
PBSO	*Probability-Based Binary Particle Swarm Optimization*
PSO	*Particle Swarm Optimization*
SA	*Simulated Annealing*
SBX	*Simulated Binary Crossover*
SLP	*Swarm Learning Process*
SMC	*Sliding Mode Control*
SML	*Simultaneous Multi-Loop*
SPM	*Singularly Perturbation Method*
SPSA	*Simultaneous Perturbation Stochastic Approximation*
SSA	*Squirrel Search Algorithm*
SSO	*Social Spider Optimization*
SSPFC	*State-Space Predictive Functional Control*
TLBO	*Teaching-Learning-Based Optimization*
TPO	*Tree Physiology Optimization*
TWO	*Tug-Of-War Optimization*
UFPR	Universidade Federal do Paraná
WCA	*Water Cycle Algorithm*
XCS	*Extended Classifier System*

Sumário

Capítulo 1

Introdução

No desenvolvimento de sistemas de controle, há uma busca natural por projetos de controlador que satisfaçam as necessidades de operação. Segundo An et al. (2018), devido à sua simplicidade, funcionalidade, e ampla aplicabilidade, o controlador PID (Proporcional-Integral-Derivativo) é o mais utilizado na indústria. Os controladores PID possuem variantes que podem se adequar melhor a determinadas funções, como adaptação do controlador em tempo real, estabilidade e capacidade de se antecipar às mudanças no sistema sendo controlado, simplicidade de manuseio, entre outros. Logo há diferentes tipos de procedimentos com complexidades diferentes. Entre as variantes do controlador PID há o controlador PID adaptativo (do inglês *Adaptive PID*, APID), o qual possui algum componente de adaptação de parâmetros de controle em tempo-real, o controlador PID de ordem fracionária (do inglês *Fractional Order PID*, FOPID), o qual possui componentes integral e derivativa de ordem não inteira, e também o controlador PID preditivo, que busca adaptar a arquitetura de um controlador preditivo baseado em modelo (do inglês *Model Predictive Controller*, MPC) para tornar o controle mais eficiente. Em todas essas possibilidades, é necessário definir parâmetros para o controle, ainda que esses parâmetros sejam apenas iniciais em alguns *designs*. Logo é essencial encontrar parâmetros de controle que gerem respostas do sistema com erro nulo em regime permanente, rápido tempo de subida, rápida estabilização, mínima variância do sinal de controle, a depender de qual é a necessidade da aplicação. O resposta do processo pode ainda ser avaliada por meio de métricas de desempenho como por exemplo a integral do erro absoluto (do inglês *Integral of Absolute Error*, IAE), integral do erro ao quadrado (do inglês *Integral of Squared Error*, ISE), integral do tempo multiplicado ao erro absoluto (do inglês *Integral of Time Multiply Absolute Error*, ITAE), e a integral do tempo multiplicado ao erro ao quadrado (do inglês *Integral of Time Multiply Squared Error*, ITSE).

A sintonia de parâmetros de controle é uma área com intensa pesquisa focada em diferentes aplicações, tais como em sistemas robóticos (Mandava e Vundavilli (2020)), hidráulicos (Wang et al. (2017)), sistemas de nível de tanque (Lakshmanaprabu et al. (2017)), hidroelétricos (Altinoz et al. (2020)). Algumas técnicas clássicas são a sintonia de Ziegler e Nichols (1942), a sintonia por relé, de Hang e Åström (1988), e a sintonia de Cohen (1953). Essas técnicas clássicas por vezes nos retornam resultados satisfatórios em aplicações mais simples, mas falham quando há uma necessidade de maior eficiência em sistemas mais complexos, com não-linearidades e/ou com múltiplas entradas e múltiplas saídas (do inglês *Multiple Input and Multiple Output*, MIMO). Logo é necessário buscar técnicas que sejam capazes de encontrar soluções suficientemente capazes de manter processos mais complexos nesse comportamento desejado. Para isso é possível buscar procedimentos no campo da otimização.

Otimizar significa encontrar a melhor solução para um determinado problema, a partir de um conjunto de soluções possíveis. O campo da otimização busca agrupar os melhores

mecanismos de busca que possibilitem alcançar esse objetivo. Um desses mecanismos de busca é a busca por heurísticas. Heurísticas são técnicas baseadas em experiências. Elas buscam solucionar o problema por meio de senso comum, intuição, "chutes" que sigam um padrão aceitável de busca de soluções, sem que necessariamente se encontre uma solução suficientemente em tempo aceitável. Utilizando-se dessa questão, é possível encontrar heurísticas de nível mais elevado, as quais buscam se utilizar de heurísticas de nível menor para encontrar soluções superiores para um mesmo problema. Essas heurísticas de maior nível são chamadas meta-heurísticas, e utilizam múltiplos agentes para formar um sistema de soluções que evoluem durante várias iterações por meio de equações matemáticas e um conjunto de regras (Hussain et al. (2019)). Esse tipo de técnica pode ser usada então na sintonia de controladores PID, devido à sua capacidade de prover soluções matematicamente capazes em aceitável tempo computacional.

1.1 Motivação

O PID é ainda o tipo de controlador mais utilizado atualmente na indústria (Borase et al. (2021)), sendo de fácil implementação e entendimento. Na realidade atual os controladores PID usados nesse contexto possuem sintonias de má qualidade, ou até mesmo não possuem sintonia (Hägglund (2019)), acarretando em perda ou lentidão de produção. Essas razões trazem motivação de se trabalhar com esse tipo de controlador devido à sua relevância ainda hoje.

No contexto de indústria 4.0, no qual há uma transformação da lógica da engenharia rumo ao próximo passo do desenvolvimento tecnológico (Muhuri et al. (2019)), é então necessário buscar alternativas de sintonia que se encaixem no problema e que visem alcançar o padrão desejado. Com isso, o campo da otimização acaba por ser uma área de profundo interesse para contribuição e aprendizado. O estudo das técnicas meta-heurísticas particularmente chama a atenção por serem fáceis de entender e aplicar.

No contexto acadêmico de uso de meta-heurísticas para sintonia de PID em processos multivariáveis a maior parte dos trabalhos se utilizam de algoritmos meta-heurísticos mais antigos. Há uma lacuna no que diz respeito ao uso de técnicas mais novas para a realização dessa sintonia. A motivação desse trabalho deriva então do desejo de se contribuir com o estudo da sintonia de controladores PID em aplicações multivariáveis, com foco no uso de meta-heurísticas mais novas.

1.2 Objetivos

O objetivo geral desta dissertação é comparar numericamente o desempenho de 5 meta-heurísticas na sintonia de 3 tipos de controladores PID para controle de 3 processos MIMO diferentes, sendo 2 meta-heurísticas mais antigas e outras 3 mais novas.

Os objetivos específicos são:

- Avaliar o desempenho dos controladores PID em cada processo MIMO controlado e definir qual controlador PID foi melhor em cada processo.
- Fazer uma classificação geral de desempenho de cada meta-heurística levando em conta todos os casos estudados.
- Verificar qual meta-heurística obteve as menores medidas de função objetivo no geral.
- Concluir qual tipo de PID alcançou o maior número de melhores sintonias (isto é, menor valor de função objetivo).

- Comparar as sintonias das 2 meta-heurísticas mais antigas com as 3 mais novas, e concluir se há diferença significativa de desempenho entre elas.

1.3 Estrutura da dissertação

A estrutura do restante dessa dissertação é organizada da seguinte forma: o capítulo 2 apresenta definições de otimização e de estrutura de problema de otimização, assim como as meta-heurísticas trabalhadas nesta dissertação. O capítulo 3 descreve o controlador PID e as variantes de ordem fracionária e adaptativa. O capítulo 4 expõe trabalhos relacionados à sintonia de controladores PID para o controle de processos MIMO, assim como destaca alguns trabalhos para cada tipo de controlador. O capítulo 5 exibe os processos estudados e os resultados obtidos pelas sintonias feitas por cada meta-heurística e tipo de controlador. O capítulo 6 faz as conclusões a que se pode chegar sobre os resultados obtidos e sua ligação com os objetivos.

Capítulo 2

Otimização

Du et al. (2016) definem otimização como "o processo de procurar pela solução ótima", enquanto que Rao (2019) define como "o ato de obter o melhor resultado sob dadas circunstâncias". Otimização num processo prático é então o ato de se encontrar a configuração que retorne o melhor desempenho para esse processo. Na engenharia durante vários processos da produção é necessário decidir quais decisões administrativas, tecnológicas e organizacionais serão utilizadas, visando minimizar o custo, o tempo, o esforço e consumo de recursos dos agentes que participam do processo, sejam eles humanos ou máquinas, ou ainda maximizar os ganhos da operação. É então notório que a área de otimização é importante para a engenharia, e a subárea de sistemas de controle se beneficia do uso de técnicas de otimização que adéquem a resposta do sistema ao comportamento desejado.

2.1 Definição de problema de otimização

Um problema de otimização é definido matematicamente como:

$$\text{Encontrar} \quad X = x_1, x_2, ..., x_n \quad \text{tal} \quad \text{que} \quad f(X) = \min(f(X)) \tag{2.1}$$

onde X é denominado vetor solução ou vetor de *design*, e f é a função objetivo, responsável por avaliar o quão boa é a solução representada por X. As variáveis x_i, com $i = 1,2,...,n$ são chamadas de variáveis de decisão, e representam os parâmetros variáveis que caracterizam uma solução. O vetor solução é sujeito a limitações de igualdade e desigualdade, definidas respectivamente como

$$\begin{aligned} l_j(X) &\leq 0, j = 1, 2, ..., m \\ g_j(X) &= 0, j = 1, 2, ..., p \end{aligned} \tag{2.2}$$

A Figura 2.1 apresenta um exemplo simples de função objetivo para um problema de vetor solução X com apenas um parâmetro.

2.2 Meta-heurísticas

Heurística é uma palavra de origem grega que significa "encontrar" ou "descobrir" (Simon (2013)), além de ser o termo usado para se referir a técnicas de descoberta baseadas em "chutes", ou senso comum. "Meta" significa "além" (Rao (2019)). Partindo disso, meta-heurísticas são heurísticas mais bem desenvolvidas, com um poder maior, de um nível superior.

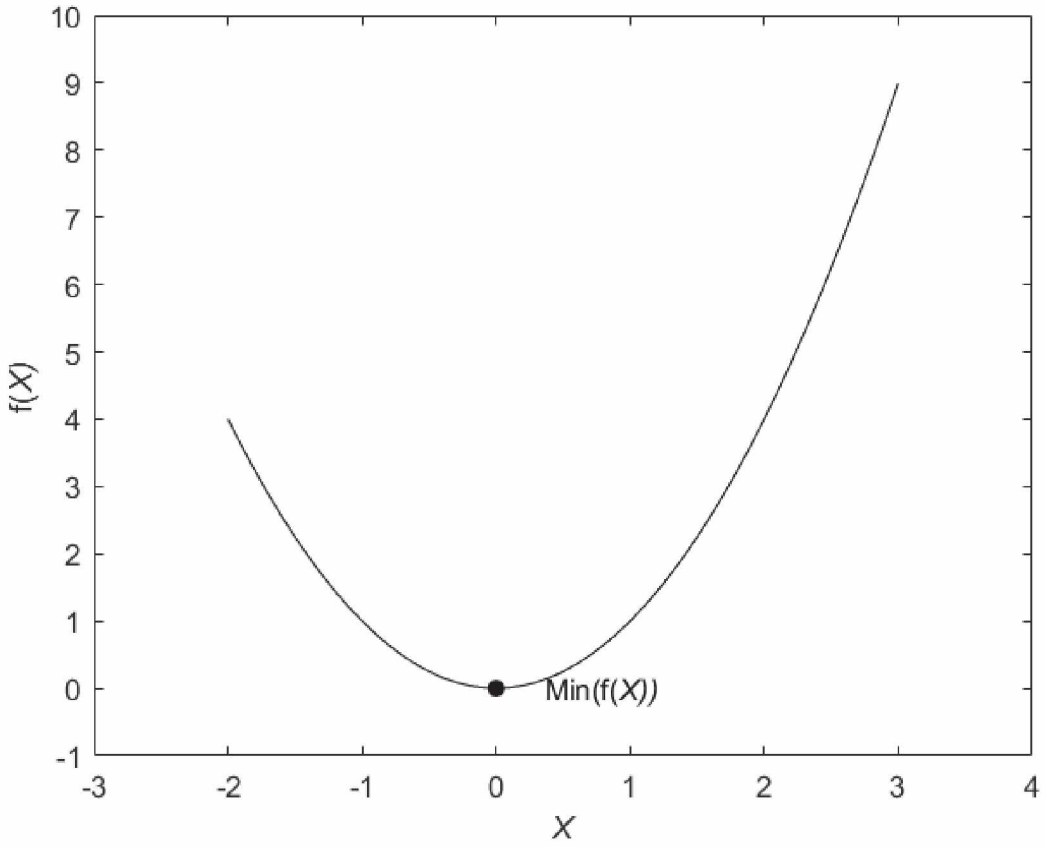


Figura 2.1: Função objetivo para um problema com valor ótimo em (0,0).

A complexidade dos problemas do mundo real tornam extremamente difícil para que métodos tradicionais de otimização os abordem, (Wang et al. (2015)). Meta-heurísticas são uma alternativa que pode dar soluções satisfatórias para problemas de otimização num tempo razoável. Elas podem ter várias inspirações mas se baseiam em processos evolutivos ou inteligência de enxame, apesar de alguns autores considerarem inteligência de enxame um subconjunto de algoritmos evolutivos (Simon (2013)).

Desde os anos 80 meta-heurísticas têm sido objeto de grandes estudos na área de otimização, tendo vários métodos desenvolvidos durante esse tempo. Segundo Boussaïd et al. (2013) o considerável desenvolvimento de meta-heurísticas pode ser explicado pelo incremento significante no poder de processamento de computadores, e o desenvolvimento de arquiteturas massivamente paralelas. O desenvolvimento acelerado de meta-heurísticas nos últimos anos faz então necessário avaliar o desempenho de métodos mais novos em comparação com métodos mais comumente utilizados, como a otimização por enxame de partículas (do inglês *Particle Swarm Optimization*, PSO) e o algoritmo genético (do inglês *Genetic Algorithm*, GA). A seguir são apresentados os princípios evolutivo e inteligência de enxame, bastante presentes em meta-heurísticas, em seguida são mostradas alguns exemplos de meta-heurísticas e suas respectivas inspirações. As meta-heurísticas utilizadas nos experimentos realizados por esse trabalho vêm em seguida, e a última seção apresenta o teorema *No Free Lunch*.

2.2.1 Algoritmos evolutivos (ou evolucionários)

Na natureza, entidades biológicas se transformam por meio de processos evolutivos. A noção de evolução é baseada na ideia darwiniana da seleção natural (Wald (2002)), na qual se desenvolve o pensamento de que indivíduos mais adaptados sobrevivem aos processos evolutivos, enquanto que os menos adaptados perecem. Aprendizagem seria nesse contexto a capacidade de um indivíduo de melhorar sua adaptação ao ambiente no qual está inserido por meio da experiência que adquire.

Algoritmos evolutivos se utilizam dessa noção de evolução darwiniana e aprendizagem presente na natureza por meio de um processo bem estruturado de busca estocástica composto por gerações e população de indivíduos. As interações dos indivíduos entre si e com o próprio processo evolutivo simulado permite então que os indivíduos progridam e melhorem a si mesmos, ao combinar aprendizagem e evolução numa estratégia lamarckiana (Ross (2019)) ou no efeito de Baldwin (Whitley et al. (1994)). A aprendizagem consiste então em sintonizar o arranjo genético de cada indivíduo, enquanto que a evolução consiste em performar um processo de seleção na população visando "filtrar" aqueles indivíduos menos adaptados. O exemplo mais conhecido de algoritmo evolutivo é o GA (Holland (1962)).

2.2.2 Inteligência de enxame

Inteligência de enxame consiste no conjunto de comportamentos coletivos de enxames existentes na natureza, como em formigas, abelhas ou peixes, e que governam a dinâmica desses seres (Holland et al. (1992)). Essa capacidade faz com que simples interações individuais entre agentes nessas populações levem à emergência de comportamento global, ainda que não haja uma estrutura de controle centralizada (Du et al. (2016)), de forma que muitas atividades desses insetos sejam auto-organizadas.

Esses comportamentos coletivos podem ser (Banks et al. (2009)):

- Cooperação natural e comportamento de grupo;
- Estratégias de busca;
- Estratégias de comunicação.

Seguindo esses princípios os algoritmos baseados em inteligência de enxame possuem inicialmente uma população (enxame) de soluções espalhada pelo universo de busca, com cada solução possuindo uma posição e uma velocidade. As soluções se baseiam na comunicação com as outras para atualizar suas posições a cada iteração, seguindo então uma combinação de 3 fatores simples (Rao (2019)):

- Coesão - Permanecer juntas;
- Separação - Não se aproximarem demais;
- Alinhamento - Seguir a direção geral do grupo.

Métodos baseados em inteligência de enxame consistem então em "traduzir" o comportamento coletivo presente em diversas espécies na natureza em algoritmos de otimização para variadas aplicações, e se diferenciam de algoritmos evolutivos pois tendem a ser mais cooperativos enquanto que algoritmos evolutivos são mais competitivos (Du et al. (2016)). Alguns exemplos de algoritmos baseados em inteligência de enxame são a otimização por colônia de formigas (do

inglês *Ant Colony Optimization*, ACO) (Dorigo et al. (1996)), a colônia de abelhas artificial (do inglês *Artificial Bee Colony*, ABC) (Karaboga e Basturk (2007)) e o PSO (Kennedy e Eberhart (1995)).

2.2.3 Exemplos de meta-heurísticas

Meta-heurísticas podem ter diversas origens, sendo as mais comuns advindas da inteligência de enxame e de processos evolutivos. A seguir são apresentados exemplos com diferentes fontes de inspiração, com um diagrama ilustrando suas inspirações na Figura 2.2.

Meta-heurísticas inspiradas em processos evolutivos

Alguns exemplos de meta-heurísticas inspiradas em processos evolutivos como o GA são os algoritmos baseados em estratégias evolutivas (do inglês *Evolution Strategy*, ES) (Rechenberg (1973)), programação evolucionária (do inglês *Evolutionary Programming*, EP) (Fogel (1998)), programação genética (do inglês *Genetic Programming*, GP) (Koza e Koza (1992)), e DE (Storn e Price (1997)).

Meta-heurísticas inspiradas em inteligência de enxame

Entre as meta-heurísticas baseadas em inteligência de enxame além do ACO, ABC e PSO estão:

- O algoritmo de forrageamento de bactérias (do inglês *Bacteria Foraging Algorithm*, BFA) (Liu e Passino (2002));
- O algoritmo do vaga-lume (do inglês *Firefly Algorithm*, FA) (Yang (2009));
- O algoritmo do morcego (do inglês *Bat Algorithm*, BA) (Yang (2010));
- A otimização de aranha social (do inglês *Social Spider Optimization*, SSO) (Cuevas et al. (2013));
- A otimização de enxame de peixes (do inglês *Fish Swarm Optimization*, FSO) (Li et al. (2004));
- A busca do cuco (do inglês *Cuckoo Search*, CS) (Yang e Deb (2009));
- O algoritmo de busca do corvo (do inglês *Crow Search ALgorithm*, CSA) (Askarzadeh (2016));
- O otimizador do lobo cinza (do inglês *Grey Wolf Optimizer*, GWO) (Mirjalili et al. (2014));
- A otimização de rebanho de elefantes (do inglês *Elephant Herding Optimization*, EHO) (Wang et al. (2015));
- O algoritmo de busca de esquilo (do inglês *Squirrel Search Algorithm*, SSA) (Jain et al. (2019));
- A otimização do gafanhoto (do inglês *Grasshopper Optimization*, GO).

Meta-heurísticas inspiradas em fenômenos físicos ou químicos

Meta-heurísticas também podem ser fundamentadas em ciências como química e física. Algumas técnicas são:

- O algoritmo de busca gravitacional (do inglês *Gravitational Search algorithm*, GSA) (Rashedi et al. (2009)) e a otimização de força central (do inglês *Central Force Optimization*, CFO) (Formato (2007)), baseados nas leis de Newton;
- O algoritmo semelhante ao eletromagnetismo (do inglês *Electromagnetism-Like Algorithm*, ELA) (Birbil e Fang (2003)) e o algoritmo de otimização magnética (do inglês *Magnetic Optimization Algorithm*, MOA) (Akbarzadeh Totonchi et al. (2008)), de inspiração em leis eletromagnéticas;
- A busca de transferência de calor (do inglês *Heat Transfer Search*, HTS) (Cuevas et al. (2014)) e a otimização de molécula de gás cinético (do inglês *Kinetic Gas Molecule Optimization*, KGMO) (Patel e Savsani (2015)), ambos de inspiração em princípios termo-energéticos;
- O algoritmo inteligente de queda d'água (do inglês *Intelligent Water Drops ALgorithm*, IWDA) (Shah-Hosseini (2009)) e o algoritmo do ciclo da água (do inglês *Water Cycle Algorithm*, WCA) (Eskandar et al. (2012)), inspirados em fenômenos naturais;
- O algoritmo de *Big Bang* e *Big Crunch* (do inglês *Big Bang Big Crunch Algorithm*, BBBCA) (Erol e Eksin (2006)) e a otimização baseada em nuvem (do inglês *Cloud-Based Optimization*, CBO) (Yan e Hao (2013)), de inspiração cosmológica.

Meta-heurísticas inspiradas por comportamento social humano

Algumas meta-heurísticas inspiradas por comportamento social humano são:

- A busca de harmonia (do inglês *Harmony Search*, HS) (Geem et al. (2001));
- A otimização de cabo-de-guerra (do inglês *Tug-of-War Optimization*, TWO) (Kaveh e Zolghadr (2016));
- O TLBO (Rao et al. (2011)).
- O algoritmo competitivo imperialista (do inglês *Imperialist Competitive Algorithm*, ICA) (Atashpaz-Gargari e Lucas (2007));
- O algoritmo de jogo de futebol (do inglês *Football Game Algorithm*, FGA).

2.3 Algoritmo Genético

Pensado originalmente por Holland (1962), o GA é um algoritmo evolucionário inspirado pela teoria da evolução natural (Wald (2002)). O algoritmo genético inicia com uma população de soluções (ou cromossomos) iniciais, e no *loop* geral do algoritmo as submete em 3 fases: seleção, *crossover*, e mutação.

Na seleção são escolhidos 2 pares de cromossomos para exercerem a reprodução, baseado em seu valor de *fitness*. A fase de *crossover* é responsável por gerar novos cromossomos

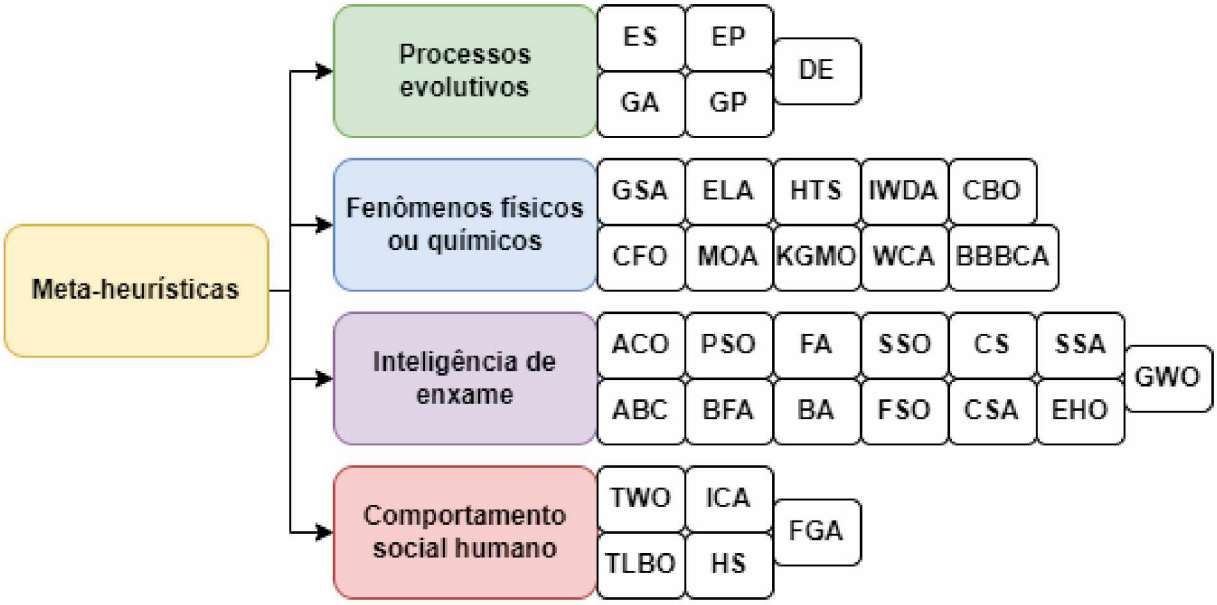


Figura 2.2: Alguns exemplos de meta-heurísticas e suas inspirações, segundo Molina et al. (2020).

a partir dos "pais" por meio de alguma estratégia. Por exemplo, os novos cromossomos podem ter a primeira metade de seus genes igual a do primeiro "pai" e a segunda metade igual à do segundo. Nesse trabalho os "filhos" foram gerados cruzando a metade superior da população com a metade inferior, de forma que para cada gene, uma fração aleatória do gene do primeiro pai some com seu complemento no segundo pai. Para manter a diversidade na população e evitar convergência rápida demais, a fase de mutação modifica 1 ou mais genes das soluções geradas na fase de *crossover*. O algoritmo se repete por um número de gerações até o critério de terminação ser alcançado. O algoritmo genético utilizado neste trabalho está descrito no Algoritmo 1.

Algoritmo 1 Pseudocódigo do GA

Entradas: O tamanho de população N e o número máximo de iterações T
Saídas: A posição da melhor solução e seu valor de *fitness*
Inicializar a posição atual $X_i (i = 1, 2, ..., N)$ de cada cromossomo
Avaliar seu valor *fitness*
enquanto (condição de parada não alcançada) **faça**
  Seleção: Escolher as $N/2$ melhores soluções e definí-las como os "pais"
  Crossover: Sejam X_i e X_j 2 cromossomos "pais". Seus 2 cromossomos "filhos" X_{k1} e X_{k2} são gerados por: $X_{k1} = r \cdot X_i + (1 - r) \cdot X_j$ e $X_{k2} = r \cdot X_j + (1 - r) \cdot X_i$
  Mutação:
  para Cada cromossomo X gerado **faça**
    Alterar um gene aleatório seu com um novo valor, também aleatório
  fim para
  Substituir os $N/2$ piores indivíduos da população pelos novos $N/2$ gerados
  Avaliar as soluções da população e encontrar seu valor *fitness*
fim enquanto
retornar X com o menor *fitness*

2.4 Otimização por enxame de partículas

O PSO foi originalmente desenvolvido por Kennedy e Eberhart (1995) motivado pelo comportamento coletivo de aglomerações de peixes e pássaros, sendo um algoritmo de inteligência de enxame baseado em população. A versão do PSO utilizada neste trabalho foi tirada do livro de Sun et al. (2016) e está ilustrada no algoritmo 2, onde c_1 e c_2 têm valor 2, e r_i e R_i são números aleatórios numa distribuição uniforme gerados no intervalo [0,1].

Algoritmo 2 Pseudocódigo do PSO

Entradas: O tamanho de população N e o número máximo de iterações T
Saídas: A posição da melhor solução e seu valor de *fitness*
Inicializar a posição atual $X_i(i = 1, 2, ..., N)$, velocidade V_i, e melhor posição P_i de cada partícula
Avaliar seu valor *fitness* e encontrar a melhor posição global G
enquanto (condição de parada não alcançada) **faça**
  para (cada partícula X_i) **faça**
    $V_i = V_i + c_1 \cdot r_i \cdot (P_i - X_i) + c_2 \cdot R_i \cdot (G - X_i)$
    se $V_i > V_{max}$ **então**
      $V_i = V_{max}$
    fim se
    se $V_i < -V_{max}$ **então**
      $V_i = -V_{max}$
    fim se
    $X_i = X_i + V_i$
  fim para
  Avaliar o valor *fitness* de X_i, isso é, $f(X_i)$
  se $f(X_i) < f(P_i)$ **então**
    $P_i = X_i$
    $f(P_i) = f(X_i)$
  fim se
  Atualizar G
fim enquanto
retornar G

2.5 Otimização do Gavião-Asa-de-Telha

A Otimização do Gavião-Asa-de-Telha (do inglês *Harris Hawks Optimization*, HHO) se baseia como o próprio nome diz no comportamento de caça que gaviões asa-de-telha (Heidari et al. (2019)), presentes na fauna estadunidense e brasileira.

Esses gaviões possuem uma estratégia chamada "pulo surpresa" na qual os gaviões unidos cercam sua presa por diferentes direções e ângulos com o intuito de cansar, encurralar e finalmente capturá-la (Bednarz (1988)), quando a energia da presa está baixa.

O algoritmo é dividido nas fases de exploração global e local. Além disso cada gavião pode transicionar entre as duas fases, a depender da energia da presa, modelada como

$$E = 2E_0 \left(1 - \frac{t}{T}\right), \tag{2.3}$$

Figura 2.3: Gavião asa-de-telha (Heidari et al. (2019)).

onde E_0 é a energia inicial da energia, definida como

$$E_0 = 2rand - 1, \tag{2.4}$$

onde *rand* retorna um número aleatório em uma distribuição uniforme gerado no intervalo [0,1].

2.5.1 Fase de exploração global

Na primeira fase, de exploração global, considera-se que os gaviões têm de se empoleirar em árvores para procurar sua presa. Com isso, dois casos são considerados, o primeiro quando o gavião baseia sua posição baseado nas posições de outros gaviões do grupo e de uma possível presa, chamada de coelho, e modelada como o gavião de menor valor de função *fitness*, e um segundo caso quando o gavião se empoleira em árvores aleatórias. Logo na fase de exploração global a posição do gavião é atualizada como

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases}, \tag{2.5}$$

onde X_{rand} é um gavião aleatório na população, X_{rabbit} é a posição do coelho, LB (do inglês *Lower Bound*) e UB (do inglês *Upper bound*) são os limites inferior e superior dos valores das variáveis, r_1, r_2, r_3 e r_4 são números aleatórios na distribuição uniforme gerados no intervalo [0,1], e X_m é a posição média da população.

2.5.2 Fase de exploração local

Na fase de exploração local a estratégia de "pulo surpresa" é empregada, sendo dividida em 4 possíveis ações dependendo de dois fatores: a chance r da presa escapar e a energia E da presa. As 4 ações são: Sitiar suavemente, sitiar agressivamente, sitiar suavemente com mergulhos rápidos e sitiar agressivamente com mergulhos rápidos.

Sitiar suavemente

Quando $r \geq 0.5$ e $|E| \geq 0.5$ a presa ainda tem energia e tenta escapar, mas falha. Os gaviões então suavemente a cercam para exaurí-la de energia e realizar o "pulo surpresa". A movimentação do gavião é modelada como

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)|, \tag{2.6}$$

onde

$$\Delta X(t) = X_{rabbit}(t) - X(t), \tag{2.7}$$

$$J = 2(1 - r_5), \tag{2.8}$$

e r_5 é um número aleatório em uma distribuição uniforme gerado no intervalo [0,1].

Sitiar agressivamente

Quando $r \geq 0.5$ e $|E| < 0.5$ a presa está exausta demais para escapar, então os gaviões performam movimentos agressivos em torno dela, como na Figura 2.4. A posição do gavião é modificada então como

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)|. \tag{2.9}$$

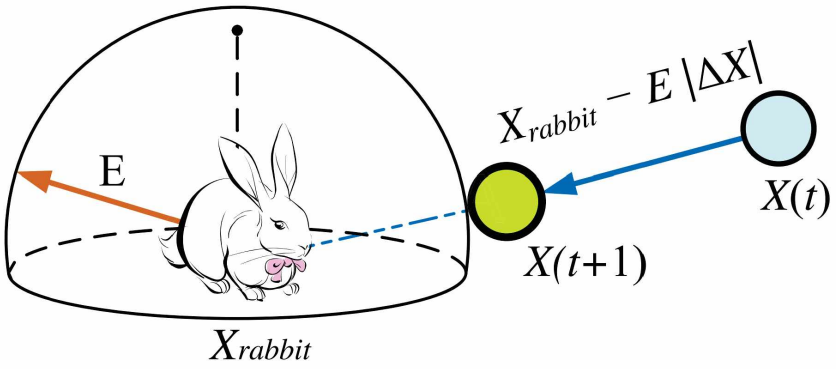


Figura 2.4: O processo de sitiar agressivamente (Heidari et al. (2019)).

Sitiar suavemente com mergulhos rápidos

Quando $|E| > 0.5$ mas $r < 0.5$ a presa tem energia suficiente para escapar então os gaviões executam uma série de mergulhos suaves com o intuito de fazer a presa cansar. O HHO considera que os gaviões conseguem avaliar qual o melhor movimento possível a se fazer a cada instante, e nesse caso o gavião avalia dois possíveis movimentos. O primeiro é

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)|, \tag{2.10}$$

e caso Y possua valor de *fitness* menor que a da posição anterior o gavião se movimenta segundo Y.

Em seguida o gavião checa se o segundo movimento é melhor que seu último movimento, e em caso afirmativo o executa. O segundo movimento é modelado como

$$Z = Y + S \times LF(D), \tag{2.11}$$

no qual S é um vetor aleatório de dimensões $1 \times D$, D é o número de variáveis de decisão, e LF é a função de voo de Lévy, usada para modelar os movimentos em zigue-zague da presa, então definida como

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \tag{2.12}$$

com

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}}, \tag{2.13}$$

onde u e v são valores aleatórios em uma distribuição uniforme gerados no intervalo [0,1], e β é uma constante especificada em 1.5. Portanto a posição do gavião é atualizada como

$$X(t+1) = \begin{cases} Y, & se \quad F(Y) < F(X(t)) \\ Z, & se \quad F(Z) < F(X(t)) \end{cases}. \tag{2.14}$$

Na imagem 2.5 a ação de sitiar suavemente com mergulhos rápidos está ilustrada.

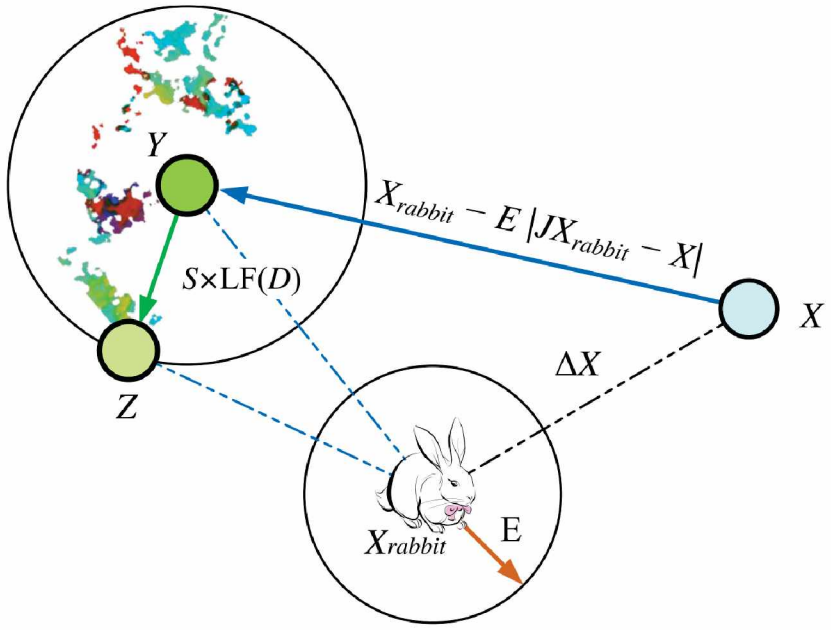


Figura 2.5: O processo de sitiar suavemente com mergulhos rápidos (Heidari et al. (2019)).

Sitiar agressivamente com mergulhos rápidos

Os gaviões sitiam agressivamente com mergulhos rápidos quando a presa não possui energia suficiente para escapar, com o intenção de cercá-la e finalmente apanhá-la. Para tanto,

o procedimento é semelhante à ação de sitiar suavemente com mergulhos rápidos. Os dois movimentos Y e Z são modificados para

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \tag{2.15}$$

e

$$Z = Y + S \times LF(D), \tag{2.16}$$

e a nova posição do gavião é gerada como na Equação 2.14. Esse tipo de ação está exposta na Figura 2.6.

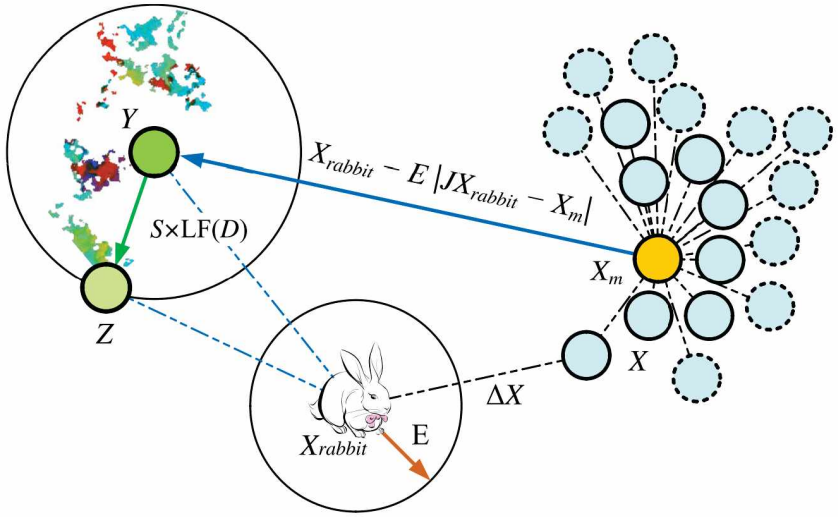


Figura 2.6: O processo de sitiar agressivamente com mergulhos rápidos (Heidari et al. (2019)).

O pseudocódigo do HHO está ilustrado no Algoritmo 3.

Algoritmo 3 Pseudocódigo do HHO

Entradas: O tamanho de população N e o número máximo de iterações T
Saídas: A posição da presa e seu valor de *fitness*
Inicializar a população $X_i (i = 1, 2, ..., N)$
enquanto (condição de parada não alcançada) **faça**
  Calcular os valores de *fitness* dos gaviões
  Salvar X_{rabbit} como a posição da presa (melhor posição)
  para (cada gavião X_i) **faça**
    Atualizar a energia inicial E_0 e J
    Atualizar E usando a Equação 2.3
    se ($|E| \geq 1$) **então**
      Atualizar posição de gavião por meio da Equação2.5
    else se ($|E| < 1$) **então**
      se ($r \geq 0.5$ e $E \geq 0.5$) **então**
        Atualizar gavião por meio da Equação 2.6
      else se ($r \geq 0.5$ e $E < 0.5$) **então**
        Atualizar gavião por meio da Equação 2.9
      else se ($r < 0.5$ e $E \geq 0.5$) **então**
        Atualizar gavião por meio das Equações 2.10, 2.11, e 2.14
      else se ($r < 0.5$ e $E < 0.5$) **então**
        Atualizar gavião por meio das Equações 2.15, 2.16, e 2.14
      fim se
    fim se
  fim para
fim enquanto
retornar X_{rabbit}

2.6 Algoritmo de Predadores Marítimos

O Algoritmo de Predadores Marítimos (do inglês *Marine Predators Algorithm*, MPA) é inspirado pelo movimento de criaturas marítimas, baseado em voo de Lévy e movimento Browniano (Faramarzi et al. (2020)).

2.6.1 Divisão de fases

O algoritmo é dividido em 3 fases, nas quais o movimento dos predadores variam, e cada fase corresponde a $1/3$ do número de iterações do algoritmo. A primeira fase é o equivalente a uma fase de exploração global, a segunda de transição para exploração local, e a última uma fase de exploração local. As fases estão expostas na Figura 2.7.

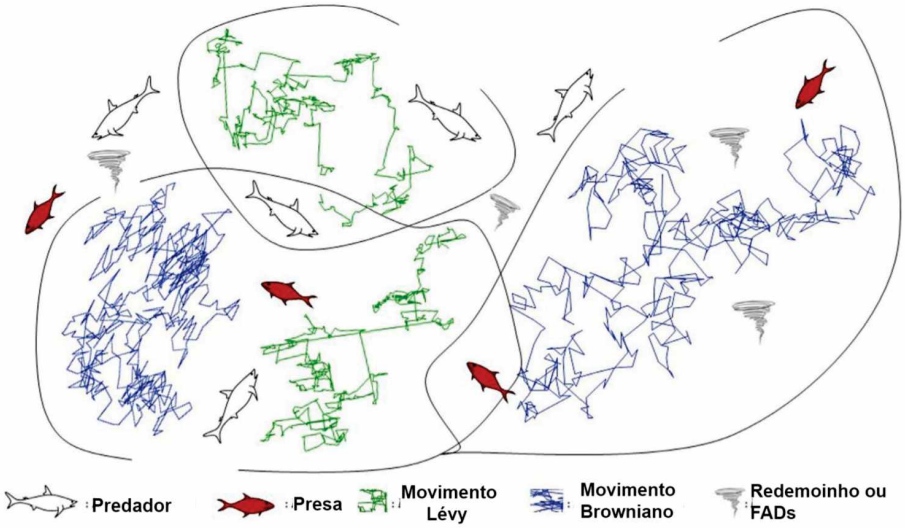


Figura 2.7: As fases do MPA, adaptado do trabalho de Faramarzi et al. (2020).

2.6.2 Caracterização do algoritmo

A população inicial do algoritmo é gerada aleatoriamente por meio de

$$X_i = X_{min} + rand(X_{max} - X_{min}), \tag{2.17}$$

onde $rand$ é um número gerado no intervalo [0,1], X_{min} é o vetor de valores mínimos possíveis e X_{max} é o vetor de valores máximos possíveis.

Após gerar a população inicial e avaliá-la, é necessário definir o vetor *Elite*, correspondente à melhor solução, aquela que representa o melhor predador. Para tal, replica-se a melhor solução n vezes, sendo n o tamanho da população:

$$Elite = \begin{bmatrix} X^I_{1,1} & X^I_{1,2} & \dots & X^I_{1,d} \\ X^I_{2,1} & X^I_{2,2} & \dots & X^I_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X^I_{n,1} & X^I_{n,2} & \dots & X^I_{n,d} \end{bmatrix}. \tag{2.18}$$

Após definir a *Elite*, Considera-se então a população como um vetor chamado *Prey*, sobre o qual os predadores atualizam suas posições:

$$Prey = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,d} \\ X_{2,1} & X_{2,2} & \dots & X_{2,d} \\ X_{3,1} & X_{3,2} & \dots & X_{3,d} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & \dots & X_{n,d} \end{bmatrix}. \tag{2.19}$$

Posteriormente, o algoritmo entra em seu *loop* principal, o qual é dividido nas 3 fases comentadas anteriormente.

Fase 1

Na fase 1 as presas se movimentam de forma Browniana. Nessa etapa predadores se movem na mesma velocidade que presas. Como inicialmente a distribuição dos agentes é uniforme e a distância entre presa e predador é grande, o movimento Browniano serve como artifício para explorar o espaço de busca. Se o *fitness* de uma presa for melhor que a do predador (solução de melhor *fitness*), essa se torna predador, levando ao processo de forrageamento de predador e presas, iniciando assim a fase 2.

Nessa fase, cada solução no vetor *Prey* se movimenta por meio da seguinte equação:

$$Prey_i = Prey_i + P \cdot R \otimes stepsize_i, \tag{2.20}$$

onde

$$stepsize_i = R_B \otimes (Elite_i - R_B \otimes Prey_i), \tag{2.21}$$

R_B é um vetor de números aleatórios numa distribuição uniforme, representando o movimento Browniano, P tem o valor fixo de 0.5, e R é um vetor de números aleatórios uniformes gerados no intervalo [0,1].

Fase 2

A fase 2 é a transição entre exploração global e local, na qual ambos predador e presa procuram por comida. Predadores procuram por comida num movimento Browniano, enquanto que presas se movimentam por meio de voo de Lévy. Dessa maneira presas podem procurar localmente de forma eficiente ao mesmo tempo que podem buscar em outra vizinhança caso não obtenham sucesso na vizinhança atual. Nesse estágio presa e predador se aproximam um do outro devido ao deslocamento com tamanho reduzido. Os efeitos de dispositivos de agglomeração de peixes (do inglês *fish aggregating devices*, FADs), assim como o movimento por voo de Lévy ajudam o algoritmo a não cair em mínimos locais.

Por ser uma etapa de transição, a fase 2 separa a população em duas, uma que se movimenta segundo o voo de Lévy (presas) e outra que se movimenta segundo o movimento Browniano (predadores). Com isso, para a primeira metade da população, correspondente às presas, seu deslocamento é definido por

$$Prey_i = Prey_i + P \cdot R \otimes stepsize_i, \tag{2.22}$$

onde

$$stepsize_i = R_L \otimes (Elite_i - R_L \otimes Prey_i), \tag{2.23}$$

R_L é um vetor de números aleatórios baseados na distribuição de Lévy, representando o movimento por voo de Lévy.

O movimento da segunda metade da população, referente aos predadores, é descrito por

$$Prey_i = Elite_i + P \cdot CF \otimes stepsize_i, \tag{2.24}$$

onde

$$stepsize_i = R_B \otimes (R_B \otimes Elite_i - Prey_i), \tag{2.25}$$

$$CF = \left(1 - \frac{iter}{Max\_iter}\right)^{\left(2\frac{iter}{Max\_iter}\right)}, \tag{2.26}$$

$iter$ é a iteração atual e Max_iter é o número máximo de iterações.

Fase 3

Na terceira e última fase, o método muda para exploração local, então os predadores modificam seu movimento de Browniano para voo de Lévy, com o intuito de procurar uma vizinhança de forma mais eficiente.

Como agora predadores se movem mais rápido que presas, esses se movem segundo a estratégia de voo de Lévy. Portanto seu deslocamento passa a ser

$$Prey_i = Elite_i + P \cdot CF \otimes stepsize_i, \tag{2.27}$$

onde

$$stepsize_i = R_L \otimes (R_L \otimes Elite_i - Prey_i), \tag{2.28}$$

e logo a multiplicação de R_L pela $Elite$ simula o movimento do predador em Lévy enquanto ao adicionar o passo ao vetor $Prey$ ajuda no movimento das presas.

2.6.3 Dispositivos de aglomeração de peixes

O método se baseia não apenas nos movimentos de Lévy e Browniano, mas também nos efeitos de "formação de redemoinho" e de FADs, os quais estão, segundo Filmalter et al. (2011), em proximidade imediata de tubarões em mais de 80% do tempo desses predadores. Nos outros 20% do tempo, os predadores tomam pulos de distância maior par encontrar um ambiente com outra distribuição de presas. FADs são considerados então mínimos locais que prendem pontos do espaço de busca. O efeito dos FADs na população de soluções é feito como a seguir:

$$Prey_i = \begin{cases} Prey_i + CF[X_{min} + R \otimes (X_{max} - X_{min})] \otimes U & se \quad r \leq FADs \\ Prey_i + [FADs(1-r) + r](Prey_{r1} - Prey_{r2}) & se \quad r > FADs \end{cases}, \tag{2.29}$$

onde FADs tem o valor de 0.2, U é um vetor binário construído a partir da geração de um vetor aleatório de distribuição uniforme gerado no intervalo [0,1] e mudando o valor para 0 se esse valor for menor que 0.2 e para 1 se o valor for maior que 0.2. r é um número aleatório com distribuição uniforme gerado no intervalo [0,1], e $r1$ e $r2$ são índices aleatórios de soluções na população.

O pseudocódigo para o MPA está exposto no algoritmo 4.

Algoritmo 4 Pseudocódigo do MPA

Entradas: O tamanho de população N e o número máximo de iterações Max_iter
Saídas: A melhor solução e seu valor de *fitness*
Inicializar a população $X_i (i = 1, 2, ..., N)$
enquanto (condição de parada não alcançada) **faça**
  Calcular o *fitness* da nova população e construir a matrix *Elite*
  se $(iter < Max\_iter)$ (**Fase 1**) **então**
    Atualizar o vetor *Prey* a partir da Equação 2.20.
  else se $(Max\_iter/3 < iter < 2 \cdot Max\_iter/3)$ (**Fase 2**) **então**
    para (Primeira metade da população) **faça**
      Atualizar primeira metade de *Prey* a partir da Equação 2.22
    fim para
    para (Segunda metade da população) **faça**
      Atualizar segunda metade de *Prey* a partir da Equação 2.24
    fim para
  else se $(iter > 2 \cdot Max\_iter/3)$ (**Fase 3**) **então**
    Atualizar o vetor *Prey* a partir da Equação 2.27
  fim se
  Salvar a nova população com os indivíduos mais eficientes
  Aplicar o efeito dos FADs segundo a Equação 2.29.
fim enquanto
retornar X com menor *fitness*

2.7 Otimização da Arraia

A Otimização da Arraia (do inglês *Manta Ray Optimization*, MRO) é uma técnica de otimização bio-inspirada em arraias, animais representados na Figura 2.8. Mais especificamente, a técnica é inspirada em 3 estratégias únicas de forrageamento de arraias: forrageamento em cadeia, forrageamento em ciclone, e forrageamento por cambalhota (Zhao et al. (2020)).

2.7.1 Procedimento do algoritmo

Após definir os parâmetros iniciais (tamanho de população e número de gerações), gera-se a população inicial dentro dos limites inferior e superior de valores para as variáveis de decisão. O *loop* principal do algoritmo, dividido em 3 fases (forrageamento em cadeia, em ciclone, e por cambalhota), começa com o forrageamento em ciclone e finaliza com o forrageamento por cambalhota.

Figura 2.8: Uma arraia (*Getty Images*, 2022).

Forrageamento em cadeia

A primeira estratégia de forrageamento do algoritmo é a estratégia em cadeia. Essa estratégia acontece quando um grupo de arraias formam uma linha, e as arraias machos se aderem nas costas das arraias fêmeas, sincronizando as batidas de suas barbatanas peitorais com a de sua respectiva fêmea. Dessa maneira, o plâncton não apanhado por arraias anteriormente é recolhido pelas arraias seguintes. Esse comportamento cooperativo, ilustrado na Figura 2.9, permite a esses seres coletar a maior quantidade de de plâncton em suas brânquias e melhorar suas recompensas de comida (Zhao et al. (2020)).

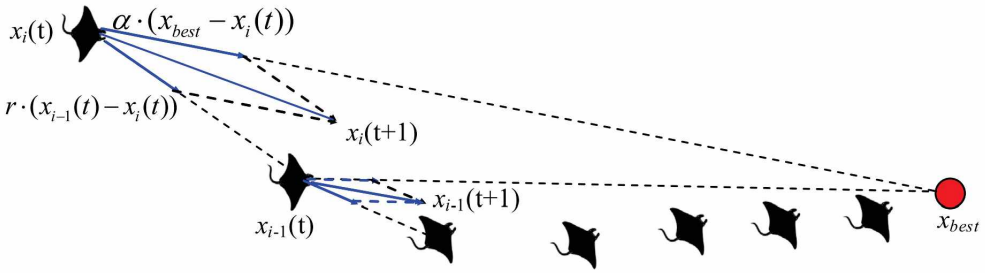


Figura 2.9: Forrageamento em cadeia, segundo Zhao et al. (2020).

Arraias buscam posições as quais possuam a maior quantidade de plâncton possível. No algoritmo, esse comportamento é simulado por meio da Equação 2.30.

$$X_i^d(t+1) = \begin{cases} X_i^d(t) + r \cdot (X_{best}^d(t) - X_i^d(t)) + \alpha \cdot (X_{best}^d(t) - X_i^d(t)) & i = 1 \\ X_i^d(t) + r \cdot (X_{i-1}^d(t) - X_i^d(t)) + \alpha \cdot (X_{best}^d(t) - X_i^d(t)) & i = 2, \ldots, N \end{cases}. \quad (2.30)$$

Onde

$$alpha = 2 \cdot r \cdot \sqrt{|log(r)|}, \quad (2.31)$$

$X_i^d(t)$ é a posição da variável de decisão d, da arraia de número i, no tempo t. A variável r é um vetor aleatório uniforme gerado no intervalo [0,1], a é um coeficiente de peso, $X_{best}^d(t)$ é a

posição com maior concentração de plâncton, ou seja, o indivíduo na população de arraias que retorna o melhor valor de *fitness*, e N é o número de variáveis de decisão.

Forrageamento em ciclone

Quando um enxame de arraias encontra um amontoado de plâncton, elas formam uma corrente e nadam em direção ao alvo por uma espiral, ao mesmo tempo que cada arraia segue a que está em sua frente na corrente. A Figura 2.10 demonstra esse movimento realizado por arraias num espaço bidimensional.

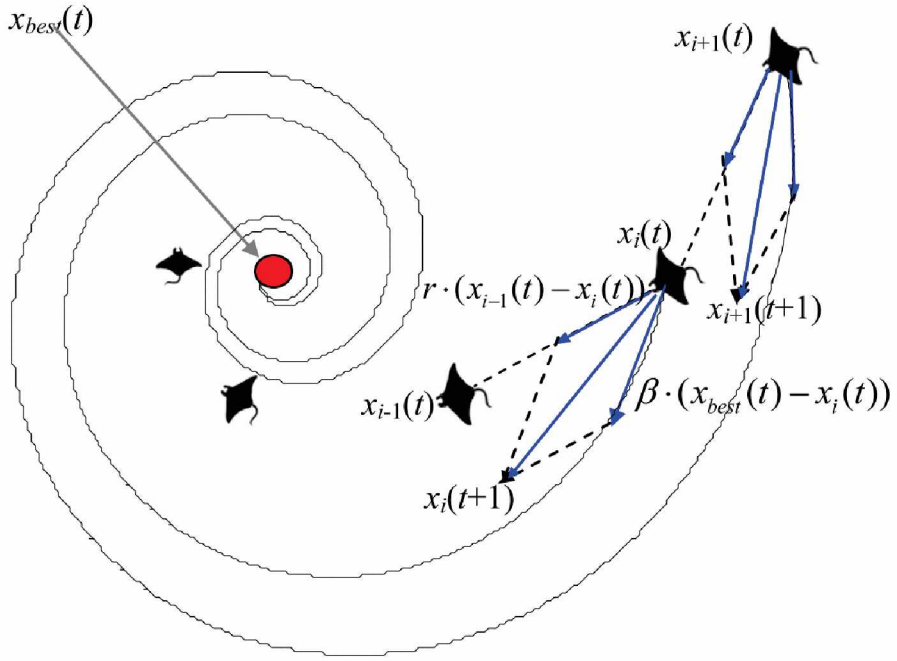


Figura 2.10: Forrageamento em ciclone, segundo Zhao et al. (2020).

O modelo matemático de tal comportamento pode ser estendido para n dimensões, e com isso a posição de cada arraia é calculada no algoritmo como

$$X_i^d(t+1) = \begin{cases} X_{best}^d(t) + r \cdot (X_{best}^d(t) - X_i^d(t)) + \beta \cdot (X_{best}^d(t) - X_i^d(t)) & i = 1 \\ X_{best}^d(t) + r \cdot (X_{i-1}^d(t) - X_i^d(t)) + \beta \cdot (X_{best}^d(t) - X_i^d(t)) & i = 2, \ldots, N \end{cases}, \quad (2.32)$$

onde

$$\beta = 2e^{r_1 \frac{T-t+1}{T}} \sin 2\pi r_1, \quad (2.33)$$

onde β é um coeficiente de peso, T é o número máximo de iterações e r_1 é um número aleatório uniforme gerado no intervalo [0,1].

O comportamento em ciclone das arraias com a melhor posição como referência permite ao algoritmo uma boa exploração local na área local de maior concentração de plâncton. Esse comportamento pode ainda ser usado para melhorar o fator de exploração global por meio da atribuição de uma posição aleatória no espaço de busca como referência para um indivíduo na

população. Dessa forma é garantido à técnica a capacidade de de exercer uma busca global eficiente, e isso é feito por meio da Equação 2.34.

$$X_i^d(t+1) = \begin{cases} X_{rand}^d(t) + r \cdot (X_{rand}^d(t) - X_i^d(t)) + \beta \cdot (X_{rand}^d(t) - X_i^d(t)) & i = 1 \\ X_{rand}^d(t) + r \cdot (X_{i-1}^d(t) - X_i^d(t)) + \beta \cdot (X_{rand}^d(t) - X_i^d(t)) & i = 2, \ldots, N \end{cases}, \tag{2.34}$$

onde

$$X_{rand}^d = LB^d + r \cdot (UB^d - LB^d) \tag{2.35}$$

é a posição aleatória no espaço de busca selecionada a fim de globalizar a busca da população de arraias, e LB^d e UB^d são os limites de valores inferior e superior para a dimensão d das variáveis de decisão.

Forrageamento por cambalhota

Nessa estratégia a posição da comida é como um pivô, para o qual as arraias se aproximam e se afastam fazendo uma cambalhota para uma nova posição. Logo toda atualização de posição é feita em torno da melhor posição até então encontrada. O modelo matemático para tal movimento é

$$X_i^d(t+1) = X_i^d(t) + S \cdot (r_2 \cdot X_{best}^d - r_3 \cdot X_i^d(t)), \tag{2.36}$$

onde $S = 2$ é o fator de cambalhota que decide o intervalo de cambalhota das arraias e r_2 e r_3 são dois números aleatórios de distribuição uniforme gerado no intervalo [0,1].

Com esse movimento definido, é possível para cada indivíduo na população se mover para qualquer posição entre a posição atual e a posição simétrica com relação à melhor solução encontrada até então. Conforme a distância entre os indivíduos e a melhor posição diminui, a perturbação na posição atual também se reduz. Todos os indivíduos se aproximam gradualmente da solução ótima, e portanto o intervalo de cambalhota se reduz à medida que o número de gerações se aproxima de seu máximo. Isso é apresentado na Figura 2.11.

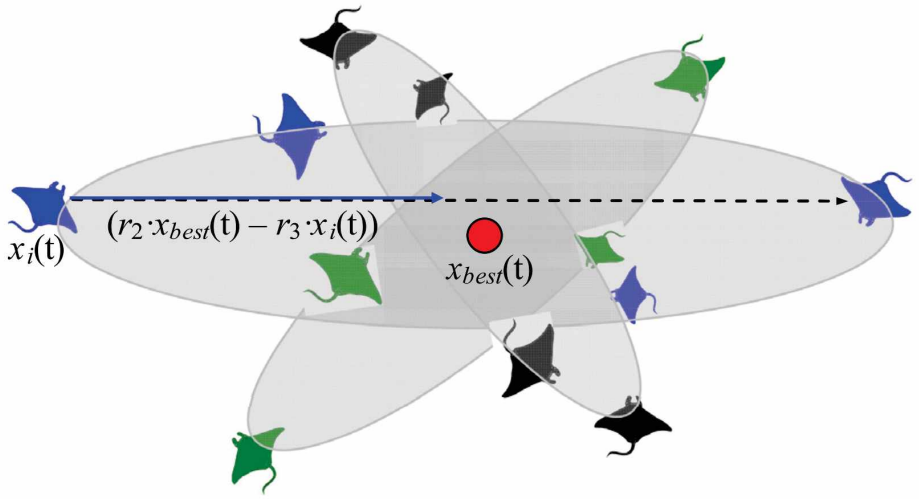


Figura 2.11: Forrageamento por cambalhota, segundo Zhao et al. (2020).

O procedimento para implementação do MRO pode ser visto no algoritmo 5.

Algoritmo 5 Pseudocódigo do MRO

Entradas: O tamanho de população N e o número máximo de iterações T
Saídas: A melhor solução e seu valor de *fitness*
Inicializar a população $X_i (i = 1, 2, ..., N)$ e calcular seu valor de *fitness*
enquanto (condição de parada não alcançada) **faça**
  para $(i = 1...N)$ **faça**
    se $(rand < 0.5)$ **então**
      // Forrageamento em ciclone
      se $(t/T < rand)$ **então**
        Modificar a posição da arraia segundo a Equação 2.34
      else
        Modificar a posição da arraia segundo a Equação 2.32
      fim se
    else
      // Forrageamento em cadeia
      Modificar a posição da arraia segundo a Equação 2.30
    fim se
    Computar o *fitness* do indivíduo X_i
    se $(f(X_i(t+1)) < f(X_{best}))$ **então**
      $X_{best} = X_i(t+1)$
    fim se
  fim para
  para $(i = 1...N)$ **faça**
    // Forrageamento por cambalhota
    Modificar a posição da arraia segundo a Equação 2.36
    Computar o *fitness* do indivíduo X_i
    se $(f(X_i(t+1)) < f(X_{best}))$ **então**
      $X_{best} = X_i(t+1)$
    fim se
  fim para
fim enquanto
retornar X_{best}

2.8 Teorema *No Free Lunch*

O teorema *No Free Lunch* (Wolpert et al. (1995)) afirma que não há método ou algoritmo de busca que seja universalmente superior em encontrar soluções ótimas para todo o conjunto de funções possíveis. Ou seja, todos os algoritmos de busca alcançam a mesma performance numa enumeração aleatória, quando avaliados no conjunto de todas as funções de custo possíveis.

Contudo esse teorema se baseia na performance média para resolver todos os problemas existentes, não levando em consideração que ao se desenvolver uma solução para determinado problema costuma-se ter algum conhecimento sobre ele, e a avaliação de função de custo do quão boa é a solução se baseia nesse conhecimento a priori sobre o problema. Logo soluções específicas são geradas para aplicações específicas, as quais são portanto superiores a soluções não trabalhadas dentro desse contexto.

2.9 Comentários sobre o capítulo

Nesse capítulo foram apresentadas a definição de otimização, a estrutura de um problema de otimização, o que são meta-heurísticas e como elas podem resolver problemas, suas principais aspirações em processos evolutivos e inteligência de enxame, assim como exemplos de meta-heurísticas com outras inspirações como fenômenos físicos e comportamento social humano. As 5 meta-heurísticas usadas para as sintonias dos controladores foram detalhadas. Por fim, o teorema *No Free Lunch*, o qual determina não existir método superior que retorne a melhor solução em todos os casos foi apresentado.

Capítulo 3

Controle PID

Segundo Åström et al. (2006), o controlador PID é uma simples implementação da realimentação numa malha de controle. Ele tem a capacidade de eliminar erros de regime permanente por meio de sua componente integral e de antecipar o futuro por meio da componente derivativa. Além disso, em controle de processos, mais de 95% das malhas de controle são do tipo PID, em sua maioria do tipo PI (Proporcional-Integrativo). Essa dominância do controlador PID se deve à sua simplicidade, clara funcionalidade, aplicabilidade e facilidade de uso, segundo Iruthayarajan e Baskar (2009).

Os controladores PID possuem variantes que podem se adequar melhor a determinadas funções, como adaptação do controlador em tempo real, estabilidade e capacidade de se antecipar às mudanças no sistema sendo controlado, simplicidade de manuseio, entre outros. Logo há diferentes tipos de procedimentos com complexidades diferentes. Entre as variantes do controlador PID há o controlador APID, o qual possui algum componente de adaptação de parâmetros de controle em tempo-real, o controlador FOPID, o qual possui componentes integral e derivativa de ordem menor que um, e também o controlador PID preditivo, que busca adaptar a arquitetura de um controlador preditivo baseado em modelo MPC para tornar o controle mais eficiente.

Neste capítulo são apresentadas as variantes do PID trabalhadas nessa dissertação. A estrutura do PID clássico e suas ações de controle são explicadas, e em seguida são apresentadas as variantes fracionária e adaptativa.

3.1 Controlador PID clássico

O controlador PID do tipo clássico possui 3 ações de controle. A proporcional, a integral e a derivativa. Na malha de controle, o controlador se utiliza do erro proveniente da realimentação e por meio de ganhos proporcional, integral e derivativo o controlador gera uma ação de controle que é a soma das 3 ações. As ações de controle são apresentadas à frente.

3.1.1 Ação de controle proporcional

A ação de controle proporcional do controlador PID tem a forma da equação

$$u(t) = K_P \cdot e(t), \tag{3.1}$$

onde K_P é o ganho proporcional e *e(t)* é o erro no instante *t*. Essa ação permite que a resposta do sistema seja mais rápida e diminua o erro em regime permanente, mas não o anule. Assim, sua função de transferência é então

$$K(s) = \frac{U(s)}{E(s)} = K_P, \tag{3.2}$$

com *U(s)* e *E(s)* sendo o sinal de controle e o erro no domínio da frequência, e *s* sendo a frequência.

3.1.2 Ação de controle Integral

A ação de controle integral possui como equação

$$u(t) = K_I \cdot \int_0^t e(t)dt, \tag{3.3}$$

onde K_I é o ganho integral. Essa ação se baseia em utilizar o "acúmulo" de erro passado para fazer com que a resposta ao sistema se aproxime do *setpoint*, e possui a utilidade de reduzir a 0 o erro em regime permanente. Entretanto a componente integral também possui a desvantagem de aumentar o *overshoot* da resposta do sistema.

A função de transferência dessa componente é portanto

$$K(s) = \frac{U(s)}{E(s)} = \frac{K_I}{s}. \tag{3.4}$$

3.1.3 Ação de controle derivativa

A ação de controle derivativa utiliza a derivada do erro no tempo e sua equação é

$$u(t) = K_D \cdot \frac{de(t)}{dt}, \tag{3.5}$$

Onde K_D é o ganho derivativo. A ação de controle derivativa visa melhorar a resposta transitória do sistema, isso é, diminuir seu tempo de subida, tempo de acomodação, e o *overshoot* da resposta. Contudo a ação derivativa pode gerar ruídos indesejáveis na resposta em regime permanente.

Sua representação na frequência é

$$K(s) = \frac{U(s)}{E(s)} = K_D \cdot s. \tag{3.6}$$

3.1.4 Estrutura do PID

A ação de controle do PID clássico é então a soma das ações de controle de suas componentes proporcional, integral e derivativa. Sua estrutura é como na equação a seguir:

$$u(t) = K_P + K_I \cdot \int_0^t e(t)dt + K_D \cdot \frac{de(t)}{dt}. \tag{3.7}$$

Para manipulação no domínio da frequência, a equação acima leva à função de transferência

$$K(s) = K_P + \frac{K_I}{s} + K_D \cdot s, \tag{3.8}$$

a qual representa as dinâmicas do PID na malha com realimentação. Isso pode ser bem observado na figura 3.1, a qual demonstra o papel do PID em uma malha fechada. Ref é a referência do processo.

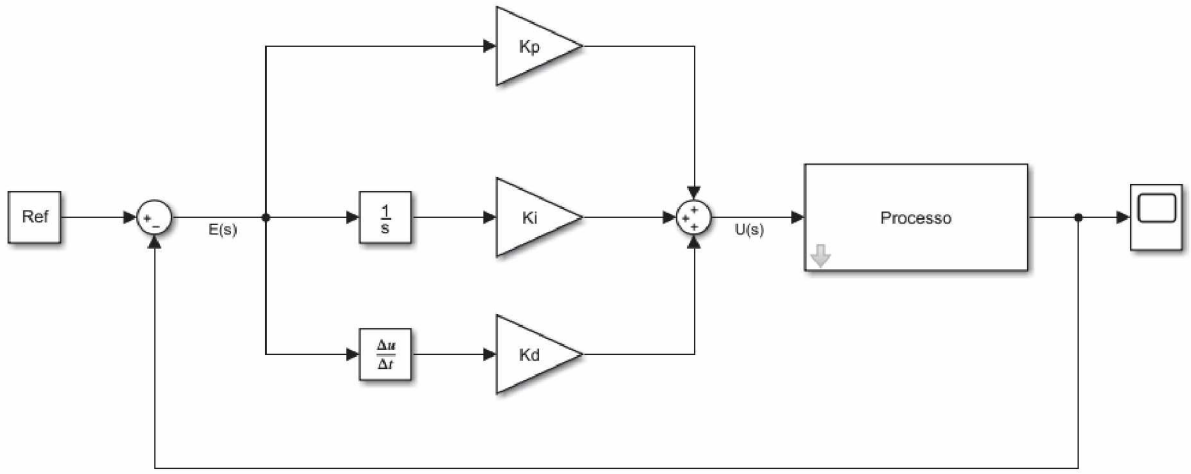


Figura 3.1: Estrutura do controlador PID.

3.1.5 Forma multivariável do controlador PID

Para sistemas MIMO com dimensões n por n, é necessário desenvolver uma matriz de controle com mais de um controlador. A depender se a estrutura é centralizada ou não, a matriz n por n do controlador pode possuir todos os elementos não-nulos, no caso centralizado, ou apenas a diagonal principal com elementos não-nulos, no caso descentralizado. Neste trabalho foi utilizada a estrutura descentralizada do controlador PID MIMO. Para chegar a ela, considera-se primeiramente um sistema MIMO n por n com função de transferência

$$G(s) = \begin{bmatrix} G_{11}(s) & \dots & G_{1n}(s) \\ \vdots & \ddots & \vdots \\ G_{n1}(s) & \dots & G_{nn}(s) \end{bmatrix}. \tag{3.9}$$

Para esse sistema generalizado, o controlador PID MIMO descentralizado deve ter como matriz de controle

$$K(s) = \begin{bmatrix} K_{11}(s) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & K_{nn}(s) \end{bmatrix}, \tag{3.10}$$

onde cada K_{ii} é como na equação 3.8, ou seja

$$K_{ii}(s) = K_{P_i} + \frac{K_{I_i}}{s} + K_{D_i} \cdot s. \tag{3.11}$$

3.2 Controlador PID de ordem Fracionária

O conceito de controladores FOPID remete ao trabalho de Podlubny (1994), no qual a estrutura de um controlador PID é alcançada por meio de definição de *Caputo* de derivada

fracionária. O FOPID é uma generalização do PID clássico, por meio da aplicação de cálculo de ordem fracionária. Desde sua origem, controladores FOPID têm sido usados em aplicações variadas, como no controle de manipuladores robóticos (Silva et al. (2004), Sharma et al. (2015)), reguladores automáticos de voltagem (Karimi-Ghartemani et al. (2007)), sistemas de tanque (Lakshmanaprabu et al. (2017)), sistemas de carregamento hidráulico (Wang et al. (2018)), entre outros (Liu et al. (2015), Djari et al. (2013), Bhookya e Jatoth (2020a)).

Segundo Kumar et al. (2018) o advento do uso do FOPID se deve à sua grande adaptabilidade, conquistas de robustez, operação realizável e depuração conveniente. Contudo, devido ao seu maior número de parâmetros, o FOPID é um controlador com sintonia mais complexa quando comparado ao PID clássico (Shah e Agashe (2016)).

A seguir a derivação do controlador FOPID por meio do cálculo de ordem fracionária é demonstrada segundo a definição de derivada fracionária de *Caputo*.

3.2.1 Cálculo de ordem fracionária

Segundo Cafagna (2007), *Gottfried Wilhelm von Leibniz* é considerado o pai do cálculo de ordem fracionária, devido a uma troca de mensagens por cartas de 1695, na qual ao ser perguntado por *Guillaume de l'Hôpital* acerca da possibilidade de derivadas de ordem não inteira, particularmente de ordem igual a 1/2, *Leibniz* respondeu: "Segue que ela será igual a $x\sqrt{dx} : x$, um paradoxo aparente, do qual um dia consequências úteis serão tiradas".

Cálculo de ordem fracionária ganhou ímpeto nas últimas décadas e se tornou frequente em sistemas de controle preciso (Kumar et al. (2018)). Consiste na generalização das ordens de diferenciação e integração no operador de ordem fracionária ${}_aD_t^\alpha$, onde α é a ordem fracionária e a e t são os terminais superior e inferior da operação.

A definição do operador geral de derivação de Caputo para a ordem α de uma função *f(t)* é (Valério e Da Costa (2012))

$$ {}_cD_t^\alpha f(t) = \begin{cases} \int_c^t \frac{(t-\tau)^{-\alpha-1}}{\Gamma(-\alpha)} f(\tau)d\tau, & se \quad \alpha \in \mathbb{R}^- \\ f(t), & se \quad \alpha = 0 \\ {}_cD_t^{\alpha-\lceil\alpha\rceil} \frac{d^{\lceil\alpha\rceil}}{dt^{\lceil\alpha\rceil}} f(t), & se \quad \alpha \in \mathbb{R}^+ \end{cases}, \tag{3.12} $$

onde Γ é a função gama.

A transformada de *Laplace* do operador derivativo seguindo essa definição é

$$ \mathcal{L}[{}_0D_t^\alpha f(t)] = \begin{cases} s^\alpha F(s), & se \quad \alpha \in \mathbb{R}^- \\ F(s), & se \quad \alpha = 0 \\ s^\alpha F(s) - \sum_{k=0}^{\lceil\alpha\rceil-1} s^{\alpha-k-1} D^k f(0), & se \quad \alpha \in \mathbb{R}^- \end{cases}. \tag{3.13} $$

3.2.2 Estrutura do PID de ordem fracionária

A partir das definições passadas é possível então generalizar a equação 3.8 para o caso onde as ordens derivativa e integrativa do controlador sejam não-inteiras. A equação de transferência que representa a ação de controle de um FOPID é definida por

$$ C(s) = K_P + \frac{K_I}{s^\lambda} + K_D \cdot s^\mu, \tag{3.14} $$

onde λ é a ordem de integração e μ é a ordem de derivação.

Essa generalização provê um universo de possibilidades matematicamente maior do que aquele do PID clássico, isso torna o controlador mais adaptável e poderoso. Isso pode ser visto na figura 3.2, a qual apresenta representa a diferença de região de atuação entre o PID clássico e o FOPID.

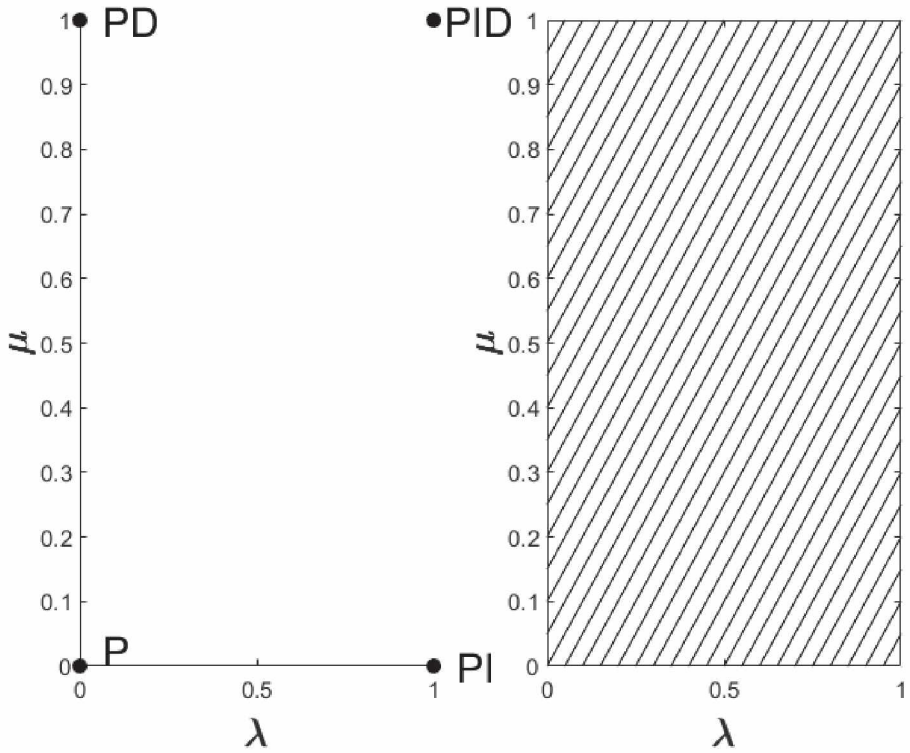


Figura 3.2: Região de um PID clássico (esquerda) e de um FOPID (direita). Os pontos pretos na esquerda indicam a região de atuação do PID clássico, e a hachura na direita indica que toda a região é atuada pelo FOPID.

Por possuir 5 parâmetros de controle (K_P,K_I,K_D,λ,μ) e não 3, como no PID clássico, a sintonia do FOPID se torna mais complexa, exigindo que o processo de otimização seja capaz de lidar com esse aumento, especialmente considerando a aplicação de FOPIDs em sistemas MIMO, que demandam mais de um controlador. Neste trabalho, os FOPIDs encontrados possuem ordens de integração e derivação no intervalo entre 0 e 1.

3.2.3 FOPID no Simulink

Neste trabalho, o FOPID foi implementado no Simulink com um intervalo de frequência $[\omega_b, \omega_h] = [0.001, 1000]$ e ordem de aproximação 5, por meio da *toolbox* desenvolvida por Tepljakov et al. (2011).

3.3 Controlador PID adaptativo

Os tipos de controle PID tratados nas seções anteriores se baseiam na ideia de que há um certo conhecimento das dinâmicas do processo a ser controlado, ainda que esse conhecimento não seja traduzido em um modelo matemático bem definido. Entretanto, há instâncias de problemas a dinâmica do processo é complexa, ou não há modelo matemático algum para ela, ou mesmo a

dinâmica do sistemas pode ser variante. O controle para esses tipos de problema demandam o uso de técnicas de identificação visando um entendimento melhor do processo controlado (Sastry et al. (1990)). Logo o controle adaptativo é um tipo de controle que se utiliza de um processo de identificação do sistema por meio dos sinais de entrada e saída para então ajustar os parâmetros do controle em tempo-real.

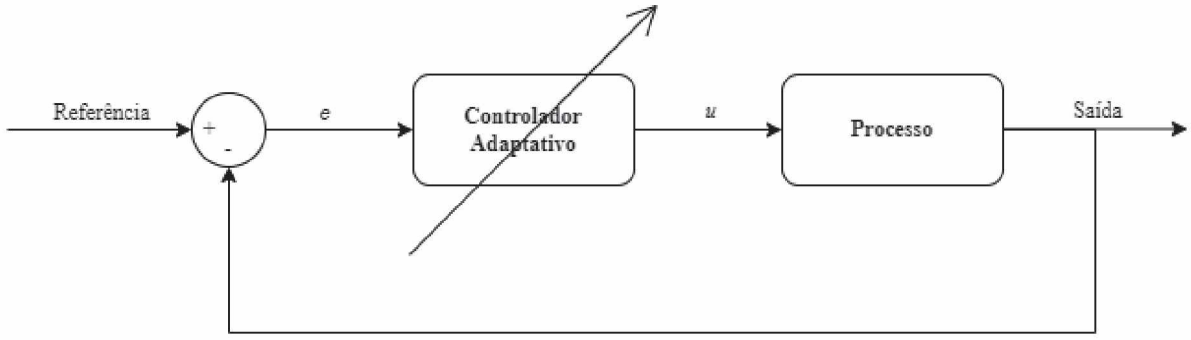


Figura 3.3: Esquema de um controlador adaptativo.

Entre as técnicas para controladores APID mais conhecidas estão o controlador APID indireto de Ziegler e Nichols (1942), o controlador APID indireto de Camacho et al. (1992) e o controlador APID direto de Yuan (1985).

3.3.1 PID adaptativo por meta-heurística

Os controladores APID ajustados por meta-heurísticas tem por base gerar inicialmente uma população aleatória de soluções, e a cada instante de tempo avaliar qual a melhor solução naquele instante e aplicá-la ao sistema. Costuma-se utilizar como função de *fitness* no algoritmo alguma combinação entre os valores atuais de erro e sua derivada, como no trabalho de Goud e Swarnkar (2019), o qual consiste numa comparação entre o algoritmo genético, a colônia de abelhas artificial e a otimização por enxame de partículas para a sintonia de um APID para o controle de um reator de tanque agitado contínuo.

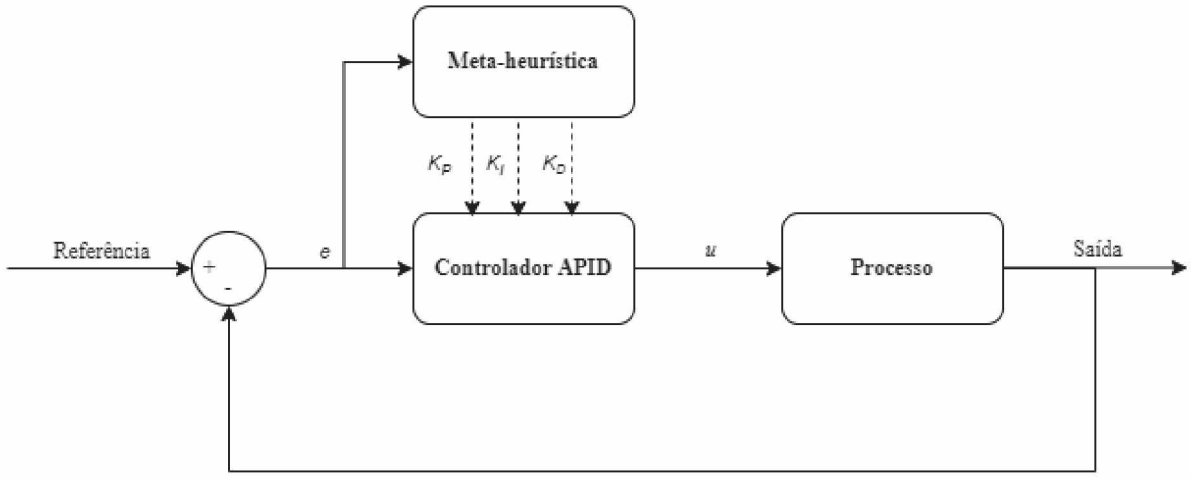


Figura 3.4: Estrutura de APID por meta-heurística.

Na figura 3.4, R e Y são a referência e a saída, respectivamente, e e u são o erro e o sinal de controle. K_P, K_I e K_D são os ganhos proporcional, integral e derivativo do controlador APID a serem sintonizados pela meta-heurística de otimização.

3.4 Comentários sobre o capítulo

Os 3 tipos de PID usados no controle dos processos nessa dissertação foram apresentados. O PID clássico, tendo apenas as 3 ações de controle (proporcional, integral, e derivativa), o de ordem fracionária, tendo 2 parâmetros a mais (ordens de derivação e integração), assim como o adaptativo, com seus ganhos sendo atualizados a cada instante de tempo.

Capítulo 4

Trabalhos relacionados

Uma revisão da literatura nos temas da dissertação foi realizada por meio da consulta aos periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior). A busca foi feita tendo como termo chave: "MIMO" ***AND*** "PID" ***AND*** "*Optimization*". A pesquisa retornou 1255 resultados, em sua ampla maioria artigos, e a distribuição de documentos está na Tabela 4.1.

Tabela 4.1: Distribuição de tipos de documentos

Tipo de documento	Contagem	Porcentagem
Artigos	1202	98,28
Atas de congresso	17	1,39
Resenhas	3	0,24
Recursos textuais	1	0,08

Quanto às bases dos resultados encontrados, os resultados foram divididos em 20 bases, com sua distribuição do número de artigos na Tabela 4.2. A *Directory of Open Journals* retornou o maior número de resultados, responsável por quase 30% dos documentos.

Tabela 4.2: Distribuição de publicações por base de dados

Base de dados	Contagem	Porcentagem (%)
Directory of Open Access Journals (DOAJ)	624	29,74
ROAD: Directory of Open Access Scholarly Resources	447	21,31
Computers & Applied Sciences Complete	205	9,77
Academic Search Premier	174	8,29
Wiley-Blackwell Full Collection 2013	157	7,48
Freedom Collection Journals	113	5,39
Sage premier Journal collection	95	4,53
SAGE Premier 2007	93	4,43
IEEE Electronic Library (IEL) Journals	54	2,57
Medline Complete	32	1,53
Emerald Journals	19	0,91
Single Journals	16	0,76
arXiv.org	16	0,76
Latindex	13	0,62
Project Euclid Prime	12	0,57
Taylor & Francis Open Access	8	0,38
SciELO Brazil	7	0,33
American Chemical Society Legacy Archives	5	0,24
ACS Publications	5	0,24
IRDB:Institutional Repositories DataBase	3	0,14

O número de publicações por ano está exposto na Figura 4.1, na qual é possível averiguar que o número de publicações nesse tema teve um aumento considerável na última década, com 1118 artigos desde 2011, um número superior aos 137 trabalhos publicados até 2010. O ano de maior número de publicações é 2020, com 164 artigos.

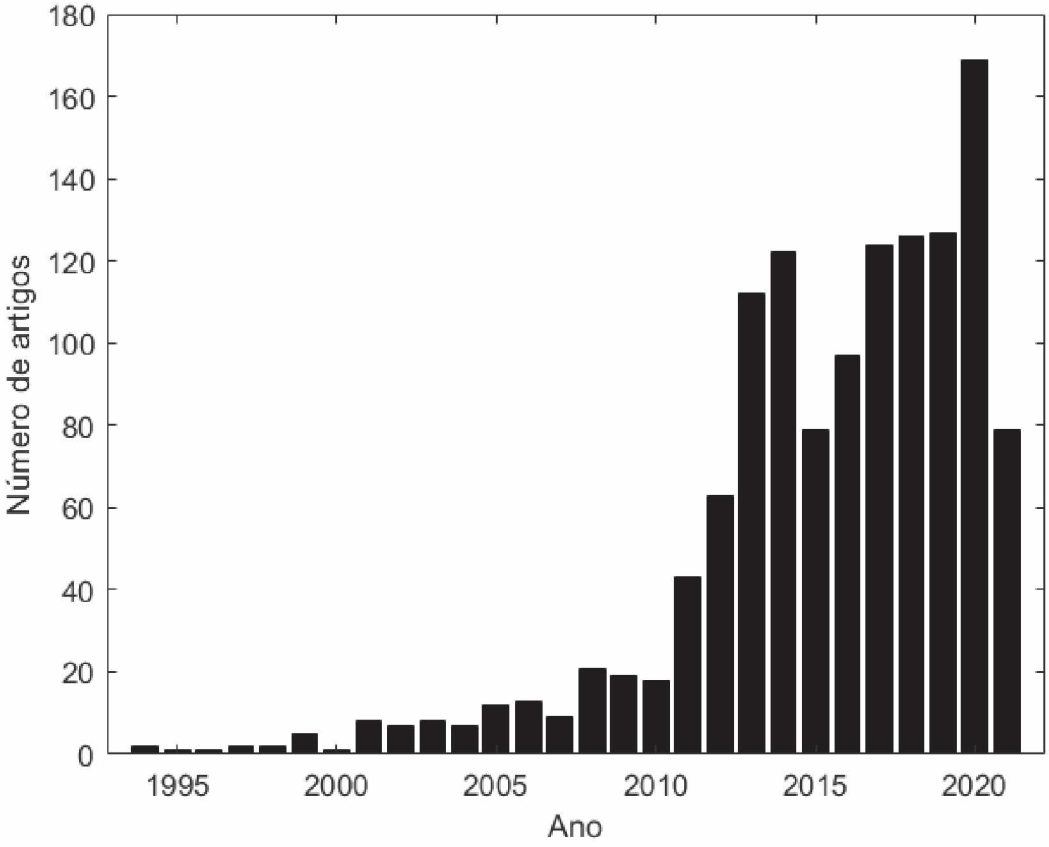


Figura 4.1: Número de publicações por ano

A fim de se catalogar trabalhos que inspirassem mais esta dissertação, chegou-se a 93 artigos depois de um processo de seleção. Dos artigos encontrados 55 envolveram o PID clássico e suas variantes, 16 envolveram APIDs, 11 FOPIDs, 5 PIDs *fuzzy* (do inglês *Fuzzy* PID, FPID), 1 PID *fuzzy* de ordem fracionária (do inglês *Fractional Order Fuzzy PID*, FOFPID), 3 sobre PIDs H∞, 1 PID com controle por modelo interno (do inglês *Internal Model Control* PID, IMCPID) e 1 sobre uma implementação de PID utilizando MPC, como ilustrado na Figura 4.2. Esses trabalhos estão detalhados nas Tabelas A.1, A.2, e A.3, presentes no Apêndice A.

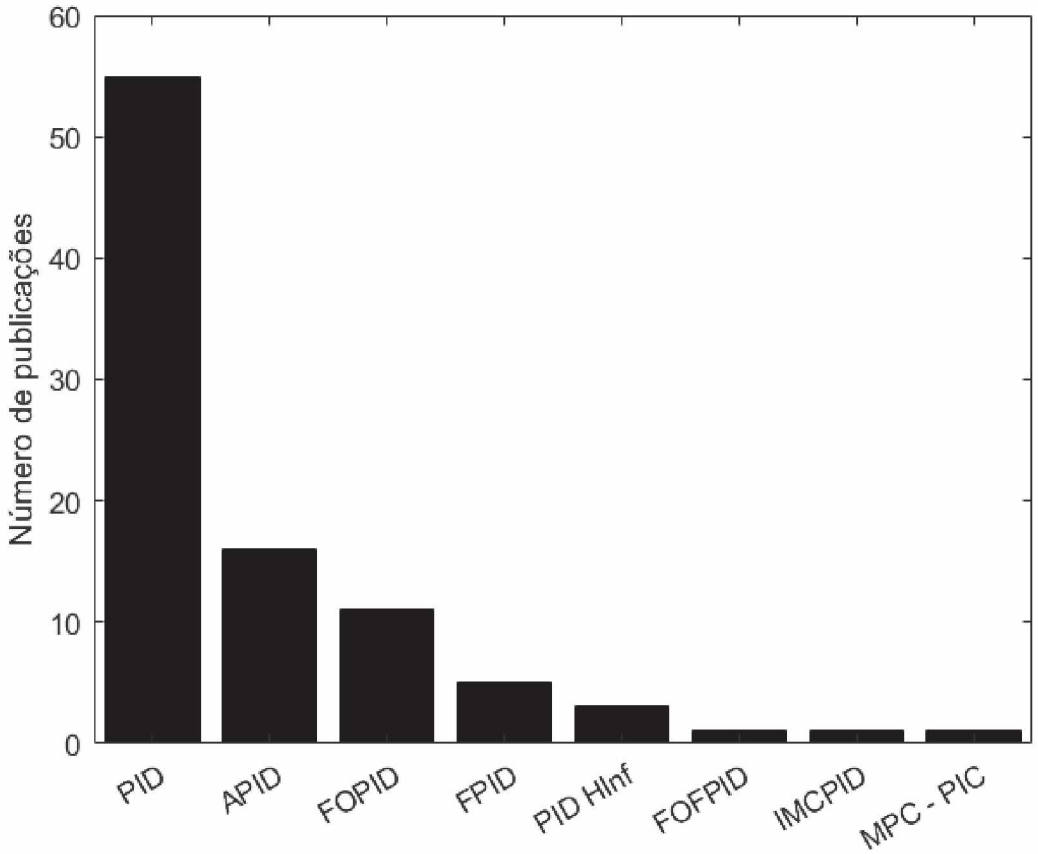


Figura 4.2: Número de publicações por tipo de PID

Na Tabela 4.3 estão expostas as técnicas de sintonia usadas nos 93 artigos encontrados, assim como o número de vezes que determinada técnica apareceu nos artigos. É possível ver que GA e PSO são as técnicas que mais aparecem. Nas seções seguintes serão mencionados trabalhos com algumas dessas sintonias feitas para os 3 tipos de controlador trabalhados nesta dissertação.

Tabela 4.3: Técnicas de otimização encontradas em trabalhos relacionados

Técnica de Otimização	Ocorrências
PSO	19
GA	18
Redes neurais	6
Desigualdade de matriz linear (do inglês *Linear Matrix Inequality*, LMI)	6
Método próprio	6
CS	4
Regulador Quadrático Linear (do inglês *Linear Quadratic Regulator*, LQR)	4
Descida de gradiente	2
Aproximação Estocástica de Perturbação simultânea (do inglês *Simultaneous Perturbation Stochastic Approximation*, SPSA)	2
Vetor de ganhos relativo dinâmico (do inglês *Dynamic Relative Gain Array*, DRGA)	2
TLBO	2
Levenberg-Marquardt	2
Dados da resposta em frequência finita (do inglês *Finite Frequency Response Data*, FFRD)	2
ES	2
DE	2
Método de Ziegler e Nichols (1942)	2
Análise de resposta ao degrau monotônica e subamortecida	1
Controle por modelo deslizante (do inglês *Sliding Mode Control*, SMC)	1
Sistema classificador extendido (do inglês *Extended Classifier System*, XCS)	1
Otimização de erva invasiva caótica modificada (do inglês *Modified Chaotic Invasive Weed optimization*, MCIWO)	1
Método de Nelder e Mead (1965)	1
Método de Åström e Hägglund (1995)	1
GO	1
MPC	1
Índice quadrático de Lyapunov	1
Controle de matriz dinâmica (do inglês *Dynamic Matrix Control*, DMC)	1
Sintonia de módulo de log grande (do inglês *Big Log Modulus Tuning*, BLT)	1
Soma de quadrados	1
Recozimento simulado (do inglês *Simulated Annealing*, SA)	1
IMC	1
Multi-loop simultâneo (do inglês *Simultaneous Multi-Loop*, SML)	1
Método singular de perturbação (do inglês *Singularly Perturbation Method*, SPM)	1
Controle funcional preditivo em espaço de estados (do inglês *State-Space Predictive Functional Control*, SSPFC)	1
Otimização da fisiologia da árvore (do inglês *Tree Physiology Optimization*, TPO)	1
Cruzamento binário simulado (do inglês *Simulated Binary Crossover*, SBX)	1
GSA	1
FA	1
ACO	1
Lógica *fuzzy*	1
Processo de aprendizagem por enxame (do inglês *Swarm Learning Process*, SLP)	1

4.1 Trabalhos envolvendo PID clássico

Foram encontrados em maior número PIDs clássicos com técnicas de sintonia para casos com processos MIMO na última década (2011 a 2020), com 45 artigos. 8 artigos foram encontrados na década de 2000 que se encaixassem na pesquisa, e apenas 2 artigos na década de 90. A técnica mais utilizada foi o uso de meta-heurísticas, com 27 artigos. Também foram usadas técnicas como redes neurais artificiais, LMI, LQR, otimização convexa, SLP, o método de Ziegler e Nichols (1942), SA, SML, SPSA e FFRD. As aplicações mais frequentes entre os artigos são aplicações em colunas de destilação, manipuladores robóticos, controle de velocidade, consumo e conversão de energia, sistemas de tanques, aeronaves, reatores de polimerização, e turbinas de caldeiras. Entre os artigos publicados que mais se destacam estão uma comparação da estratégia de evolução com adaptação de matriz de covariância (do inglês *Covariance Matrix Adaptation Evolution Strategy*, CMAES) e de uma otimização por enxame de partículas modificada (do inglês *Modified Particle Swarm Optimization*, MPSO) com outros algoritmos evolucionários na sintonia de um PID para controle da coluna de destilação de Wood e Berry (1973) (do inglês *Wood and Berry destillation column*, WB), por Iruthayarajan e Baskar (2009), e o uso da TPO para a mesma coluna por Halim e Ismail (2017). Além de uma comparação entre a evolução diferencial (do inglês *Differential Evolution*, DE), o GA, e o PSO no controle de suspensões de veículos eletro-hidráulicos, por Dangor et al. (2014).

A estrutura de um controlador PID está exposta na Figura 4.3, onde $E(s)$ é o erro, K_p é o ganho proporcional, T_i o ganho integrativo, T_d o ganho derivativo, e s representa o domínio na frequência.

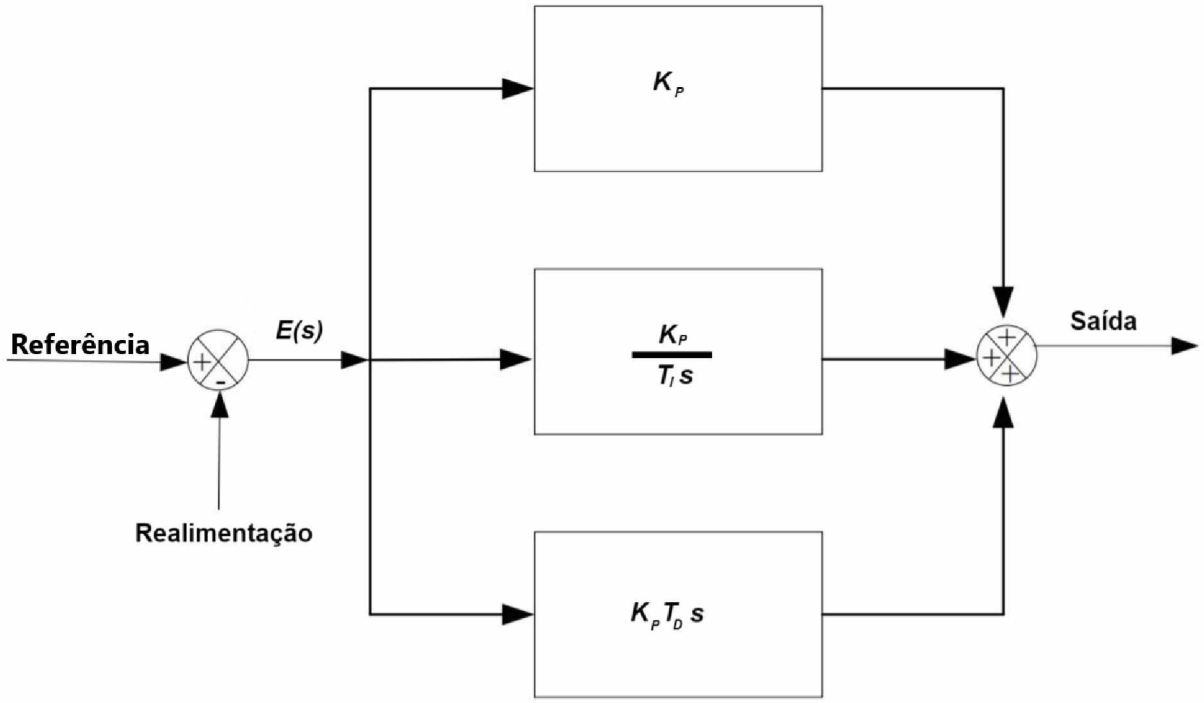


Figura 4.3: Estrutura de um controlador PID clássico. Figura adaptada. Original por Dwi et al. (2017).

4.2 Trabalhos envolvendo PID adaptativo

A sintonia de controladores PID adaptativos começou no final dos anos 2000, com a combinação de estruturas de controle adaptadas para as aplicações (como o SMC) e meta-heurísticas, assim como técnicas como o método de Nelder e Mead (1965), redes neurais artificiais, DRGA, XCS. Aplicações em robótica são as mais frequentes, com algumas aplicações em suspensão de carros, cilindros hidráulicos, transmissão mecânica automática, sistemas de nível líquido, helicóptero, pêndulo invertido, um sistema de levitação magnética, e um oscilador Duffing-Holmes. Lengare et al. (2008) desenvolveram dois algoritmos para a sintonia de controladores PID em sistemas MIMO, um baseado na resposta ao degrau monotônica e outro na subamortecida, se utilizando também de um agendamento de ganho para variar o valor do ganho proporcional do PID, ajustando assim a resposta quando há uma mudança nos parâmetros do processo. Nas aplicações com meta-heurísticas 3 trabalhos se destacaram: O primeiro, por Mahmoodabadi et al. (2017b), envolveu o uso de um GA multiobjetivo para a sintonia de um PID supervisionado por um SMC num sistema de nível de tanque. O segundo trabalho consiste no uso de um algoritmo de otimização híbrido (uma combinação entre o PSO e o GA desenvolvido por Wang et al. (2017) para um cilindro hidráulico, enquanto que o terceiro trabalho utilizou a MCIWO para o treinamento de uma rede neural que modifica os ganhos do PID para a dinâmica de um robô bípede.

Na Figura 4.4, θ_{di} é a trajetória desejada, e_i é o erro, u_{PIDi} é a ação de controle do APID, u_{SMCi} é a ação de controle do modo deslizante, u_i é a soma das duas ações de controle, e θ_i é a trajetória de saída.

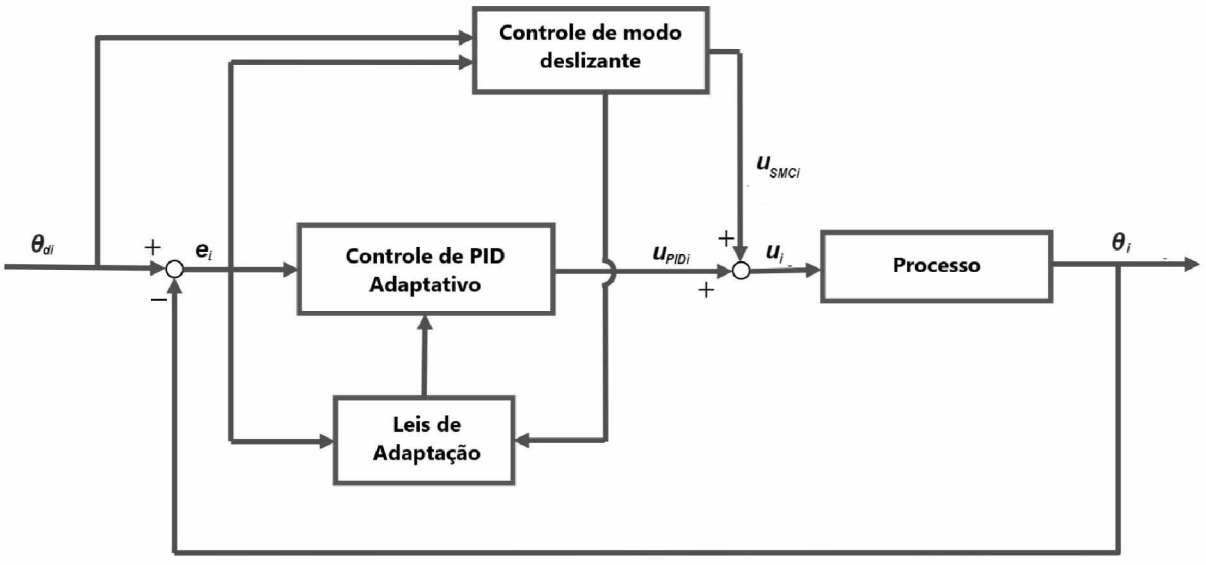


Figura 4.4: Controle de robô por PID com modo deslizante, modelado por Taherkhorsandi et al. (2015). Figura adaptada.

4.3 Trabalhos envolvendo PID de ordem fracionária

Trabalhos com PID de ordem fracionária para sistemas com múltiplas entradas e múltiplas saídas tiveram seu primeiro trabalho proposto por Silva et al. (2004), numa aplicação para o controle de um robô de 6 pernas. Naquela ocasião, a sintonia foi feita de forma experimental, por tentativa e erro. Na última década aplicações em sistemas com atraso de ordem fracionária,

manipuladores robóticos, sistemas de carregamento hidráulico, modelo de estação hidroelétrica, e um Sistema de alimentação de ar para célula de combustível de membrana de troca de prótons foram desenvolvidos. Assim como nos trabalhos para PIDs de estrutura clássica e adaptativos, sintonia por meta-heurísticas foi a técnica mais utilizada. Entretanto também houveram usos de LQR, ALgoritmo de transformada inversa de Laplace numérica (do inglês *Numeric Inverse Laplace Transform Optimization*, INVLAP), redes neurais artificiais, e algoritmo de Åström e Hägglund (1995). O trabalho que mais se mostrou próximo ao escopo deste trabalho consistiu no uso do TLBO para a sintonia de um FOPID para a WB, e foi realizado por Bhookya e Jatoth (2020a).

A estrutura de um FOPID se define pelos ganhos proporcional (K_p), integral (K_i) e derivativo (K_d), bem como as ordens de integração (λ) e derivação (μ). Isso pode ser observado na Figura 4.5, onde *e(t)* é o erro, e *t* é o tempo.

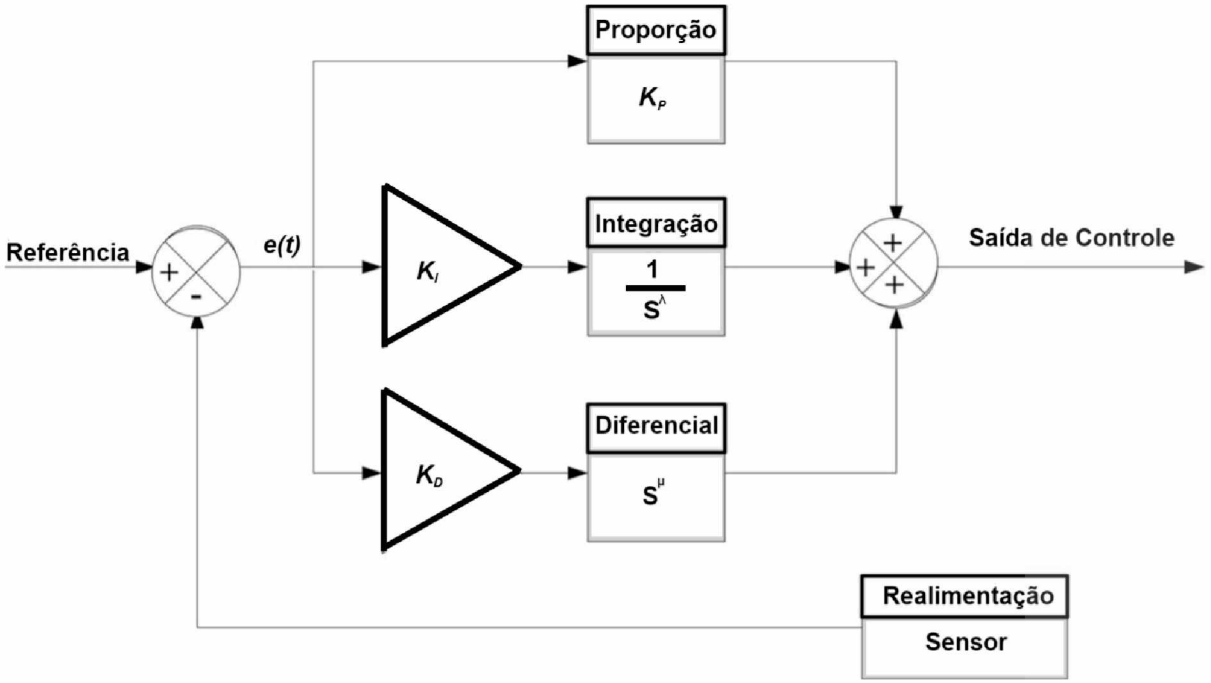


Figura 4.5: Estrutura de FOPID, adaptada da descrição feita por Bhookya e Jatoth (2020a).

4.4 Trabalhos envolvendo outras variantes do PID

As 5 aplicações de FPID encontradas se iniciam no trabalho de Ko e Wu (2008), que utilizaram o PSO para a sintonia das funções de pertinência, o número de regras *Fuzzy* e os ganhos do PID para o controle de um sistema MIMO de um experimento de gangorra. Savran (2013) utilizou o algoritmo de otimização denominado Levenberg-Marquardt para a sintonia do FPID para dois modelos matemáticos MIMO. O algoritmo de TLBO foi usado por Khooban et al. (2013) na sintonia de controladores FPID tipo-2 para o rastreamento da trajetória de robôs não-holonômicos. Gil et al. (2014) sintonizaram os ganhos de um FPID para o controle de um sistema de três tanques, e Wu et al. (2020) utilizaram o PSO para a sintonia de dois controladores FPID em tempo real, para o controle de fluxo e temperatura de um sistema de controle de ambiente de cabine de uma aeronave. Uma estrutura de FPID está descrita na Figura 4.6, onde *d/dt* é a derivação no tempo e *u* é a ação de controle.

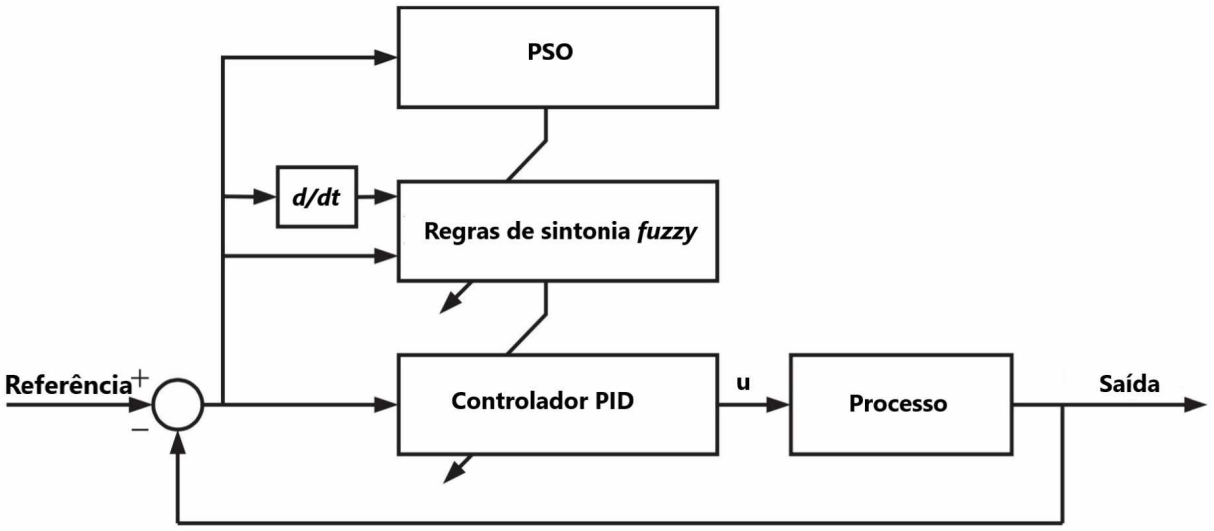


Figura 4.6: Estrutura de FPID, baseada no trabalho de Ko e Wu (2008).

Algumas outras aplicações de PIDs incluem: O PSO na sintonia de um controlador PID H∞ no controle de um sistema de aeronave de combate F18/Harv, por Zamani et al. (2009). O trabalho de Ou et al. (2014), na sintonia dos parâmetros de um controlador PID H∞ por meio do teorema de Hermite-Biehler, encontrando o conjunto tridimensional dos ganhos pela intersecção das regiões de PID H∞ nos subsistemas decompostos. O uso do CS para a sintonia dos ganhos de controladores FOFPID no controle de um manipulador robótico rígido de dois links planares, proposto por Sharma et al. (2014). Um IMCPID para controle de sistemas de vibração para asas, sintonizado por uma nova variante do PSO, a nova otimização por enxame de partículas de Luus-Jaakola cinza (do inglês *Grey New Luus-Jaakola PSO*, GNPSO). E por fim o projeto de um controlador PID baseado em MPC, por meio da fusão dos dois controladores, o PID implementado num controlador lógico programável (CLP) e o MPC em um sistema SCADA como o controlador de nível mais alto, proposto por Aboelhassan et al. (2020).

4.5 Comentários sobre o capítulo

Esse capítulo descreveu a revisão feita sobre as técnicas de otimização encontradas para a sintonia de controladores PID para processos MIMO. A busca retornou que nessa área, entre as diversas técnicas de otimização utilizadas, as meta-heurísticas são as que ocorrem em maior incidência, com uma maior prevalência de GA e PSO utilizadas para esse tipo de sintonia. A incidência de meta-heurísticas na área de otimização de controladores PID cresceu na última década e tem cada vez mais produzido trabalhos em novas aplicações. Com relação aos tipos de controlador PID, o PID clássico é o tipo de controlador mais frequentemente sintonizado, seguido por APID e FOPID.

Capítulo 5

Análise de Resultados

Os resultados são apresentados nas seções a seguir, onde em cada seção o processo MIMO em estudo é explicado, e em seguida são apresentados os resultados para cada tipo de PID na sequência: clássico, FOPID, e APID. Para cada caso são apresentadas tabela de valores de função objetivo e tabela de valores de melhores PID, assim como resposta do sistema e ações de controle.

Os algoritmos foram implementados no Matlab R2019a, enquanto que os sistemas foram implementados no Simulink 9.3 com o *solver* Ode45. As simulações foram executadas em 3 computadores diferentes, com as especificações da Tabela 5.1, onde a unidade central de processamento (do inglês *Central Processing Unit*, CPU) e a memória RAM (do inglês *Random Acess Memory*) são detalhados.

Tabela 5.1: Especificações de máquina.

Computador	CPU	Memória RAM
1	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz	8 GB
2	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz	8 GB
3	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz	8 GB

Cada algoritmo foi executado tendo tamanho de população 50 e número de gerações 100. O tempo de simulação para avaliação de função foi de 100 segundos, com 0.1 segundo de tempo de amostragem. A função objetivo utilizada foi a soma dos valores de ITSE nas respostas do processo, definida como

$$F_{obj}(X) = \sum_{i=1}^{n} \int_0^T (Y_r(t) - Y(t))^2 dt, \tag{5.1}$$

onde X é a solução avaliada, n o número de saídas no processo, t o tempo atual, T o tempo final (100 segundos), Y_r o valor desejado para a resposta em regime permanente, e Y a resposta ao degrau.

Para Haver uma melhor avaliação do desempenho de cada técnica, os algoritmos foram executados 50 vezes, e ao fim disso foram calculados os valores mínimo, máximo, médio, mediana, e o desvio padrão da função objetivo ao longo das 50 rodadas.

Os resultados para o PID clássico foram encontrados levando em consideração os seguintes espaços de busca para os ganhos: [-1,1] para a WB e [-5,5] para a OR, seguindo o

trabalho de Iruthayarajan e Baskar (2010), e [0,5] para o BMPS, seguindo o trabalho de Menhas et al. (2012).

Os resultados para o FOPID encontrados a seguir foram feitos tendo como espaço de busca das ordens de integração e derivação [0,1], os intervalos de valores para ganhos dos PIDs das duas colunas de destilação foram os mesmos do PID clássico, enquanto que o do BMPS foi [0,50], esse último seguindo o trabalho de Lúcio et al. (2021b).

Os resultados encontrados para o APID se basearam no mesmo espaço de busca de parâmetros de controle que o PID clássico.

5.1 Coluna de *Wood* e *Berry*

A WB é um modelo de coluna de destilação binária utilizado em processos químicos para controle a separação de componentes presentes numa mistura líquida. A coluna possui a estrutura mostrada na Figura 5.1.

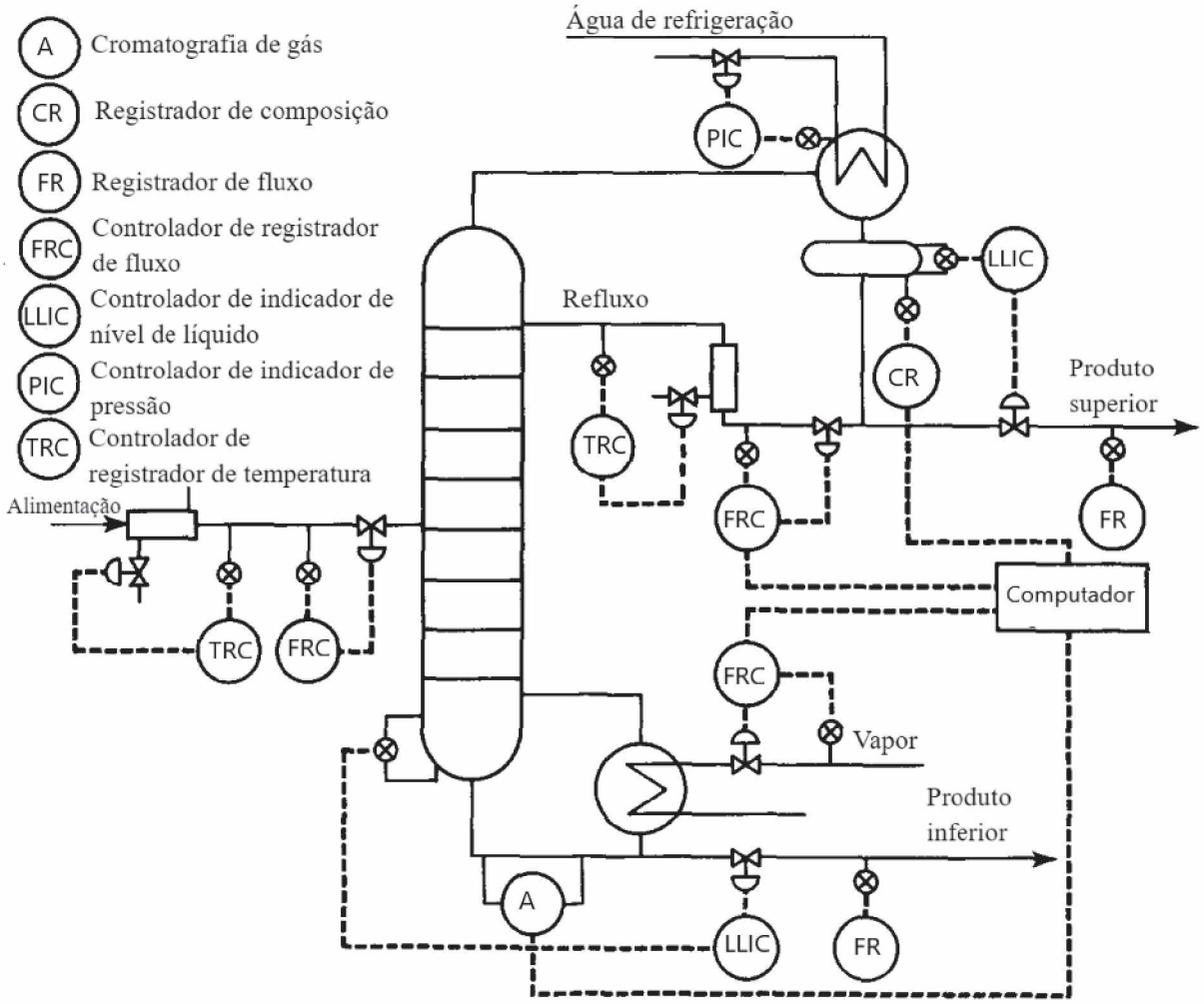


Figura 5.1: Diagrama esquemático da WB, como descrito por Wood e Berry (1973).

O sistema é um sistema com 2 entradas e saídas, sendo a primeira saída o fluxo de líquido não vaporizado no processo de fervura (ou produto inferior), e a segunda saída é o líquido condensado que é removido do sistema, conhecido como produto destilado (ou produto

superior). No sistema em malha fechada as duas entradas são o fluxo de produtos superior e inferior desejados. Seu modelo matemático é descrito em transformada de Laplace por

$$G(s) = \begin{bmatrix} \frac{12{,}8e^{-s}}{1+16{,}7s} & \frac{-18{,}9e^{-3s}}{1+21s} \\ \frac{6{,}6e^{-7s}}{1+10{,}9s} & \frac{-19{,}4e^{-3s}}{1+14{,}4s} \end{bmatrix}. \tag{5.2}$$

5.1.1 PID clássico

O desempenho de cada meta-heurística medido pelo ITSE pode ser verificado na tabela 5.2. Com relação ao valor mínimo de ITSE encontrado ao longo das 50 rodadas, o MPA obteve o melhor resultado, com o MRO tendo o pior. A classificação das técnicas ficou então, da ordem do melhor para o pior: MPA, GA, HHO, PSO, e MRO.

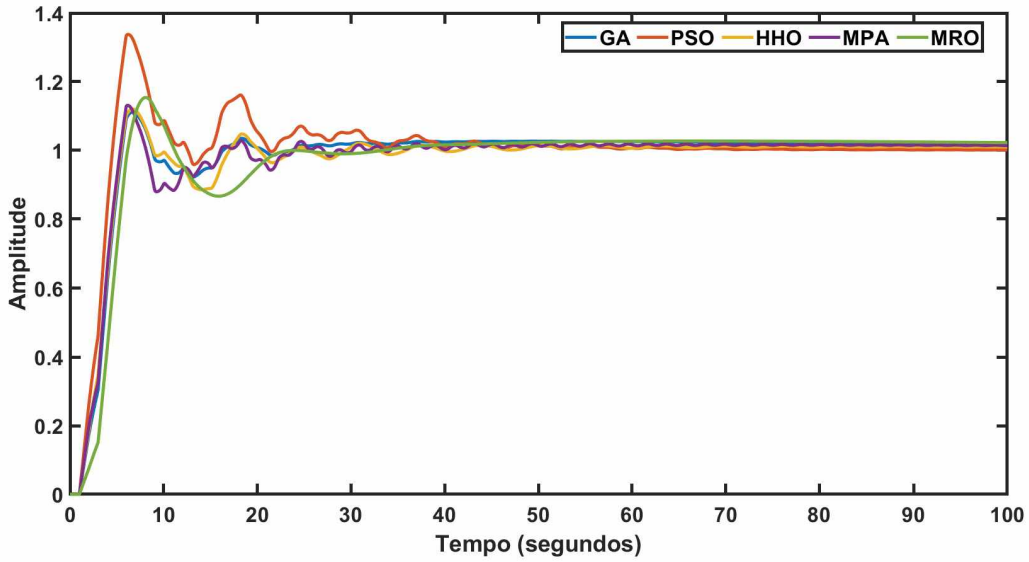
Tabela 5.2: Desempenho do controle para o PID clássico (WB).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	13,70	16,41	14,08	13,72	0,66
GA	2º	14,53	21,33	17,92	18,62	2,05
HHO	3º	16,23	201,36	33,85	28,32	27,29
PSO	4º	17,90	39,76	25,54	24,50	5,02
MRO	5º	19,32	37,07	101,08	47,90	99,45

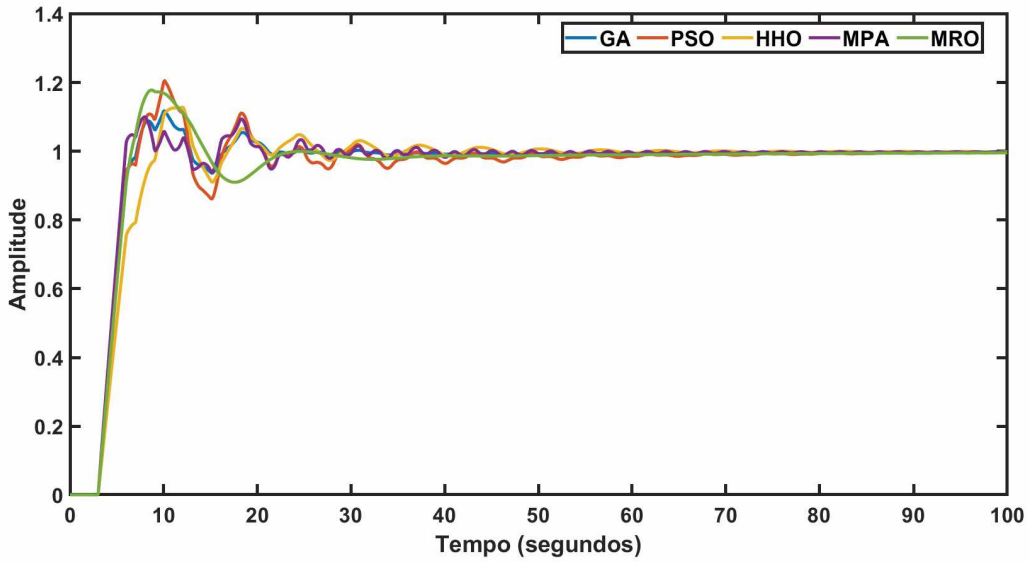
Os valores dos ganhos dos melhores PID encontrados pelas técnicas de otimização nas 50 rodadas podem ser vistos na Tabela 5.3, enquanto que as respostas do sistema a eles estão na Figura 5.2. Por elas é possível inferir que o desempenho superior de cada meta-heurística veio com uma curva mais ruidosa quando comparado com o desempenho inferior de outra. Particularmente o MRO, mesmo tendo o pior desempenho, apresentou curvas mais suaves.

Tabela 5.3: Parâmetros dos melhores PID encontrados (WB).

Meta-heurística	K_{P1}	K_{I1}	K_{D1}	K_{P2}	K_{I2}	K_{D2}
MPA	0,26	0,003	0,25	-0,26	-0,01	-0,55
GA	0,23	0,005	0,16	-0,24	-0,01	-0,46
HHO	0,24	0,002	-0,06	-0,19	-0,01	-0,53
PSO	0,33	0,03	0,16	-0,24	-0,001	-0,54
MRO	0,09	0,003	-0,13	-0,23	-0,009	-0,25



(a) Primeira saída.



(b) Segunda saída.

Figura 5.2: Comparação do controlador PID para as saídas do sistema de WB.

5.1.2 FOPID

Os valores de ITSE encontrados para os controladores FOPID estão na Tabela 5.4. Os valores mínimos encontrados em todas as meta-heurísticas foram menores que os seus correspondentes no controlador PID da subseção 5.1.1, sendo que MPA e GA foram as técnicas que mais melhoraram com relação ao seus desempenhos no PID clássico.

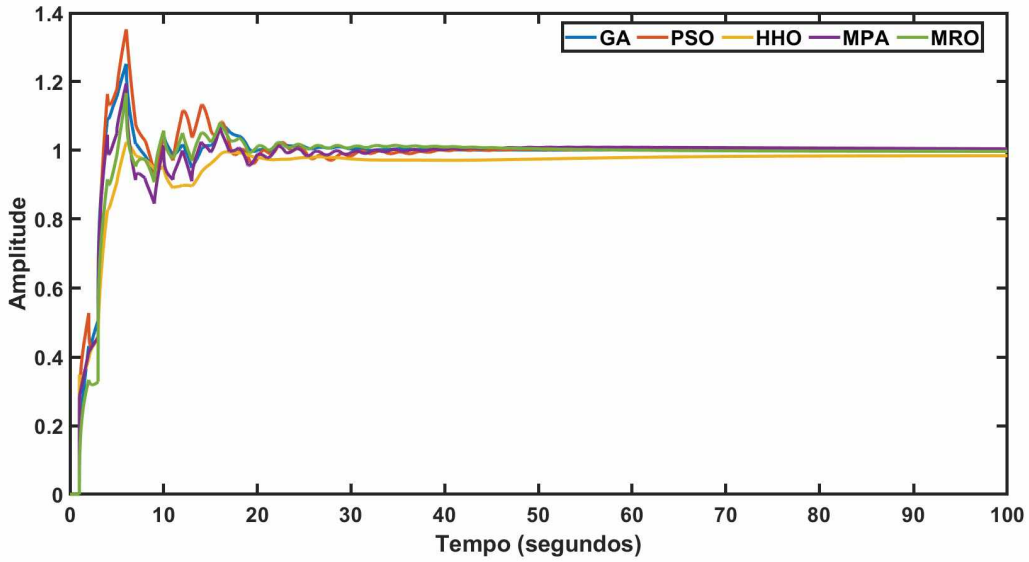
Tabela 5.4: Desempenho do controle para o FOPID (WB).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	8,82	23,28	13,30	12,79	3,13
GA	2º	9,43	31,81	13,38	12,71	3,26
PSO	3º	14,29	110,76	29,90	20,08	25,83
HHO	4º	15.09	149,28	60,59	47,26	37,94
MRO	5º	18,96	158,16	105,37	99,66	42,14

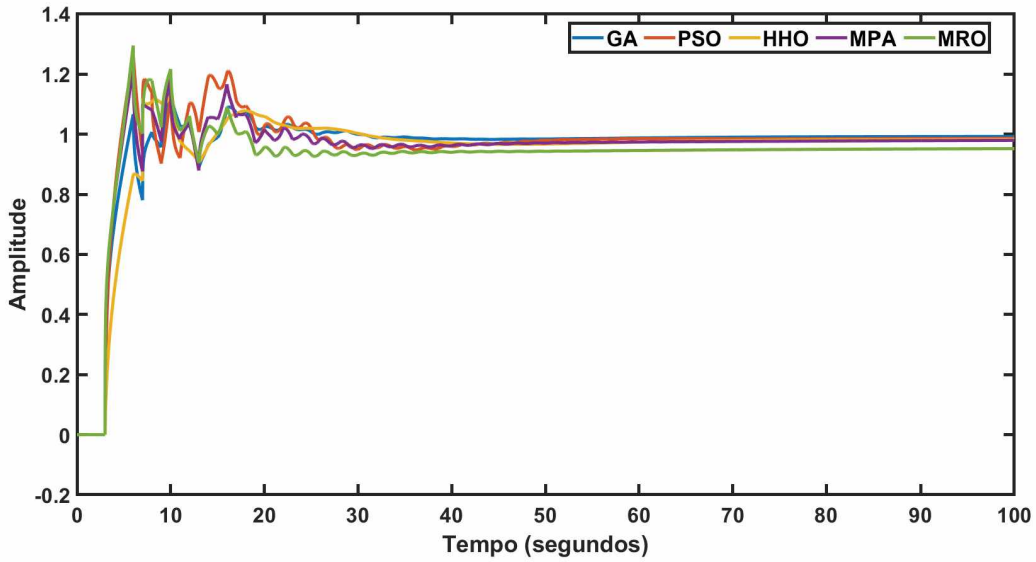
Os parâmetros dos FOPID gerados pelas 50 rodadas estão na Tabela 5.5, enquanto que as 2 saídas estão presentes na Figura 5.3. O comportamento das curvas para cada técnica de otimização foi bem semelhante. Na segunda saída, o MPA conseguiu subir rápido sem que houvesse um *overshoot* grande, o que fez diferença no cálculo do ITSE para essa configuração de controlador. Algo a ser observado é que apesar de os resultados numéricos terem sido melhores no FOPID, o comportamento da saída foi menos suave, com as curvas de saída oscilando mais que no PID clássico.

Tabela 5.5: Parâmetros dos melhores FOPID encontrados (WB).

Meta-heurística	K_{P1}	K_{I1}	K_{D1}	λ_1	μ_1	K_{P2}	K_{I2}	K_{D2}	λ_2	μ_2
MPA	-1	0,89	0,80	0,22	0,63	-0,09	-0,01	-0,43	1	0,76
GA	-0,02	0,22	0,38	0,32	0,97	-0,0004	-0,02	-0,45	0,89	0,56
PSO	0,20	0,06	0,46	0	1	-0,09	-0,03	-0,38	1	0,84
HHO	-0,25	0,25	0,15	0,19	0,47	0,03	-0,08	-0,46	0,64	0.53
MRO	-0,99	0,22	1	0,35	0,24	1	-0,41	-1,61	0,35	0,20



(a) Primeira saída.



(b) Segunda saída.

Figura 5.3: Comparação do controlador FOPID para as saídas do sistema de WB.

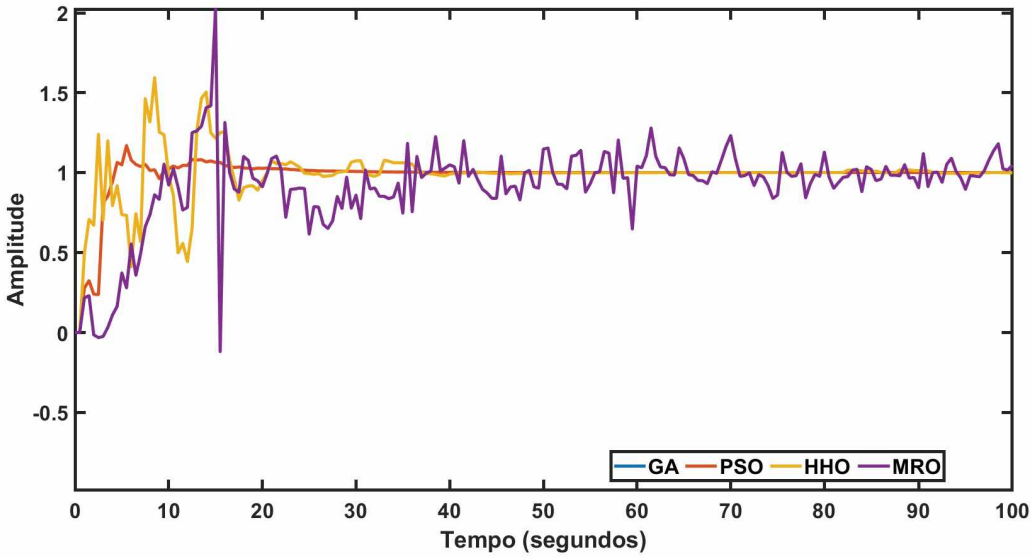
5.1.3 APID

A sintonia do APID para a WB está na tabela 5.6. O melhor algoritmo para essa sintonia foi o PSO, seguido de perto pelo HHO. No geral a sintonia feita pelas meta-heurísticas foi boa, com a exceção do MPA, que apesar de seu ótimo desempenho nos outros tipos de controladores não foi bem no APID. Se analisar também as medições estatísticas dos valores de erro, verifica-se que as meta-heurísticas mais utilizadas nessa área (GA e PSO) obtiveram os valores mais baixos, sendo mais robustas.

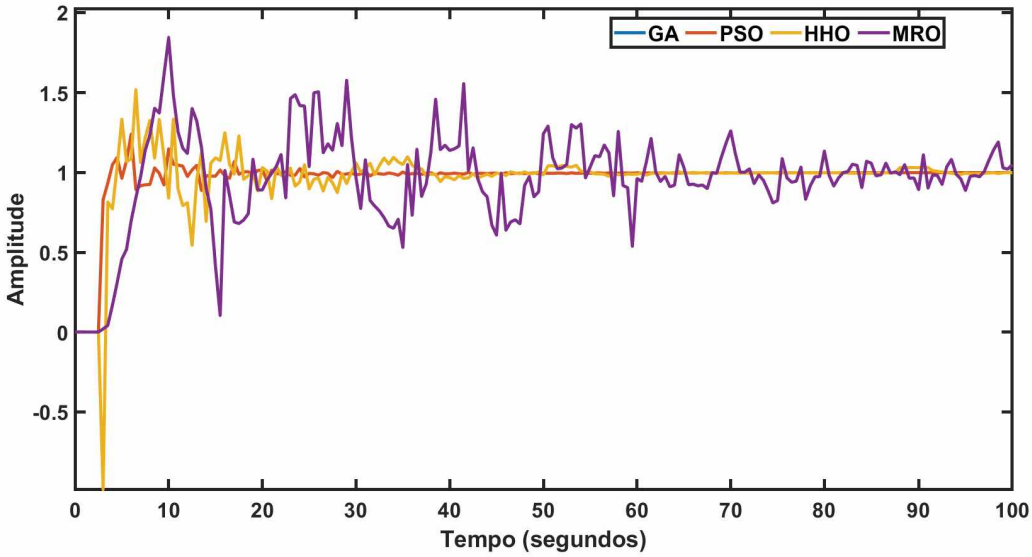
Tabela 5.6: Desempenho do controle para o APID (WB).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
PSO	1º	7,46	456,42	45,57	21,25	87,69
HHO	2º	12,43	1,9e+05	4,9e+03	134,32	2,7e+04
GA	3º	38,94	979,74	187,69	143,35	160,05
MRO	4º	243,56	1,1e+03	563,71	610,51	287,27
MPA	5º	2,8e+24	1,8e+42	6e+40	5,7e+31	3e+41

Na Figura 5.4 as 2 saídas para 4 das 5 meta-heurísticas usadas estão expostas. Devido ao seu desempenho inferior às restantes, o qual não convergiu, o MPA não foi colocado na ilustração para não reduzir a qualidade da imagem (procedimento realizado em todas as imagens do APID). O comportamento ruidoso já era esperado por parte do controlador APID, afinal seus valores de controle são gerados iterativamente. Contudo o PSO e o HHO, as técnicas de sintonia mais bem avaliadas, conseguiram obter curvas menos ruidosas, o que é melhor para o funcionamento do controlador. O PSO foi superior aos outros devido à sua capacidade de subir rapidamente a resposta do sistema sem que isso levasse a um grande *overshoot* e sem que ela oscilasse demais.



(a) Primeira saída.



(b) Segunda saída.

Figura 5.4: Comparação do controlador APID para as saídas do sistema de WB.

5.2 Coluna de *Ogunnaike* e *Ray*

A coluna de Ogunnaike et al. (1983) (do inglês *Ogunnaike and Ray distillation column*, OR) é uma coluna de destilação binária utilizada num sistema de etanol-água, exposta na Figura 5.5. Nela, A é a alimentação, B é o pré-aquecimento, C e E são controles de concentração, D é o controle de nível, F é o controle de temperatura, G indica o controle de saída de produto inferior, P é um sensor de pressão, d_1 é a taxa de fluxo de alimentação (m^3/s), d_2 é a temperatura de alimentação ($°C$), u_1 é a taxa de refluxo (m^3/s), u_2 é a taxa de fluxo de produto lateral (m^3/s), e u_3 é a pressão de fluxo da caldeira (kPa).

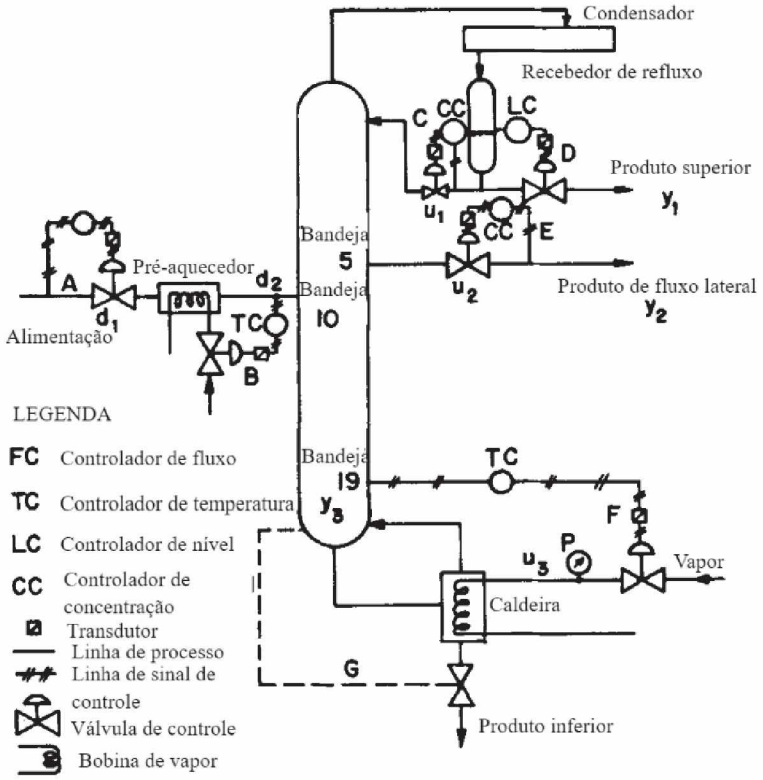


Figura 5.5: Diagrama esquemático da OR, como descrito por Ogunnaike et al. (1983).

O modelo possui 3 entradas e saídas, sendo que a primeira saída (y_1) corresponde à fração molar de etanol superior, a segunda (y_2) à fração molar de etanol de fluxo lateral, e a terceira saída (y_3) correspondente à temperatura em graus Celsius da bandeja número 19 da coluna, que coincide com a composição inferior. Assim, o modelo possui forma matemática retratada por

$$G(s) = \begin{bmatrix} \frac{0{,}66e^{-2{,}6s}}{6{,}7s+1} & \frac{-0{,}61e^{-3{,}5s}}{8{,}64s+1} & \frac{-0{,}0049e^{-s}}{9{,}06s+1} \\ \frac{1{,}11e^{-6{,}5s}}{3{,}25s+1} & \frac{-2{,}36e^{-3s}}{5s+1} & \frac{-0{,}012e^{-1{,}2s}}{7{,}09s+1} \\ \frac{-34{,}68e^{-9{,}2s}}{8{,}15s+1} & \frac{46{,}2e^{-9{,}4s}}{10{,}9s+1} & \frac{0{,}87(11{,}61s+1)e^{-s}}{(3{,}89s+1)(18{,}8s+1)} \end{bmatrix}. \tag{5.3}$$

5.2.1 PID clássico

Na tabela 5.7 os valores estatísticos para o sistema de OR são apresentados, mostrando que a tendência de desempenho superior por parte do MPA e inferior do MRO se mantiveram na sintonia do controlador, e a classificação entre meta-heurísticas ficou como na sintonia da WB.

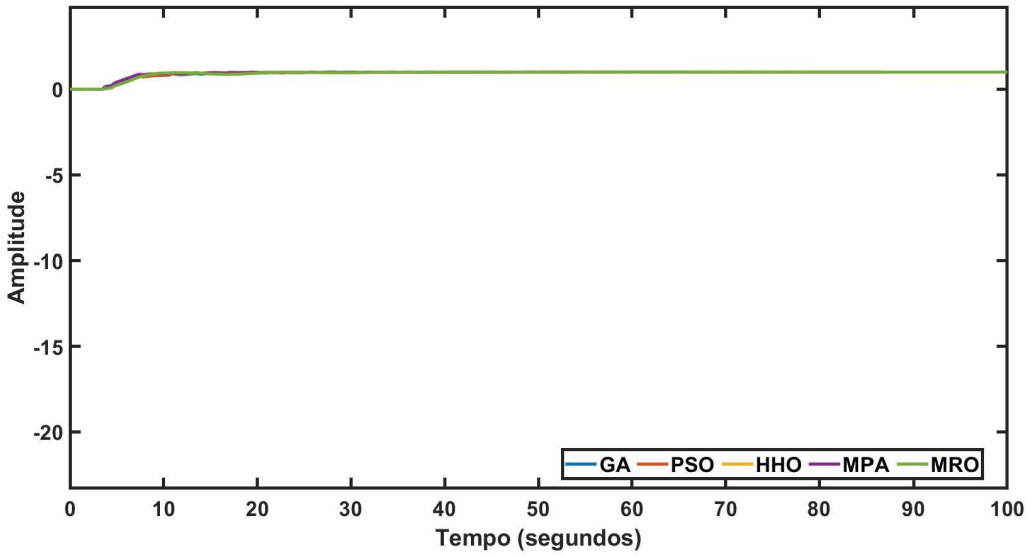
Tabela 5.7: Desempenho do controle para o PID clássico (OR).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	40,38	50,80	41,77	40,84	2,05
GA	2º	41,64	85,77	51,34	47,82	9,14
HHO	3º	42,26	322,25	73,21	60,89	44,89
PSO	4º	44,76	570,83	114,11	73,49	102,95
MRO	5º	53,11	1582,7	472,43	332,03	486,12

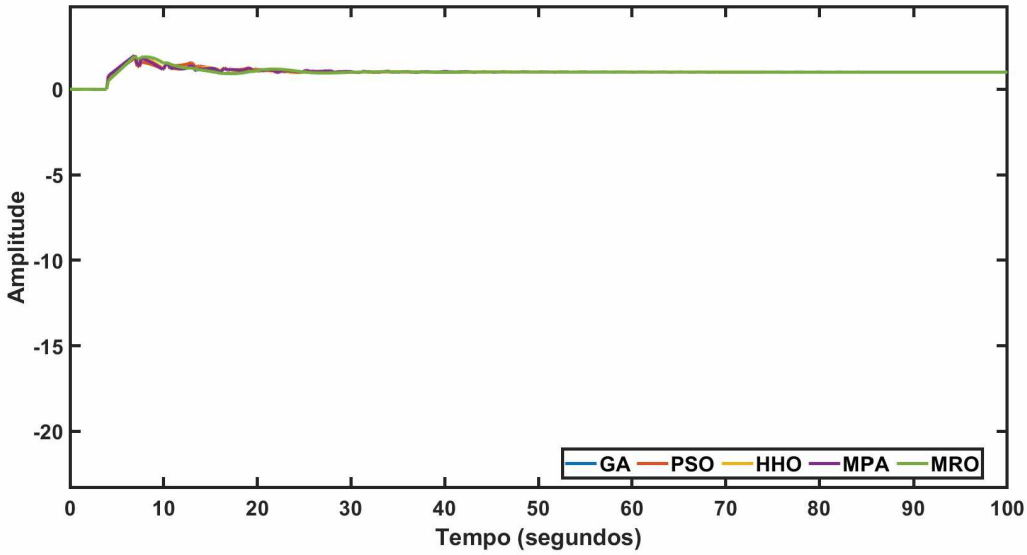
Os melhores PIDs encontrados na sintonia feita pelas diferentes técnicas têm seus valores de ganho mostrados na Tabela 5.8. As respostas ao degrau das 3 saídas do sistema estão presentes na Figura 5.6. Na primeira e segunda saídas a resposta ao degrau oscila bastante, com o MPA sendo capaz de chegar mais rapidamente no *setpoint*, mesmo na segunda saída, quando seu *overshoot* é maior. Na terceira e última saída, que possui uma tendência de decair abaixo de 0 com amplitude elevada, o MPA conseguiu decair menos em comparação com as outras técnicas, ao mesmo tempo que subiu mais rapidamente (ainda que com *overshoot* e oscilação maiores).

Tabela 5.8: Parâmetros dos melhores PID encontrados (OR).

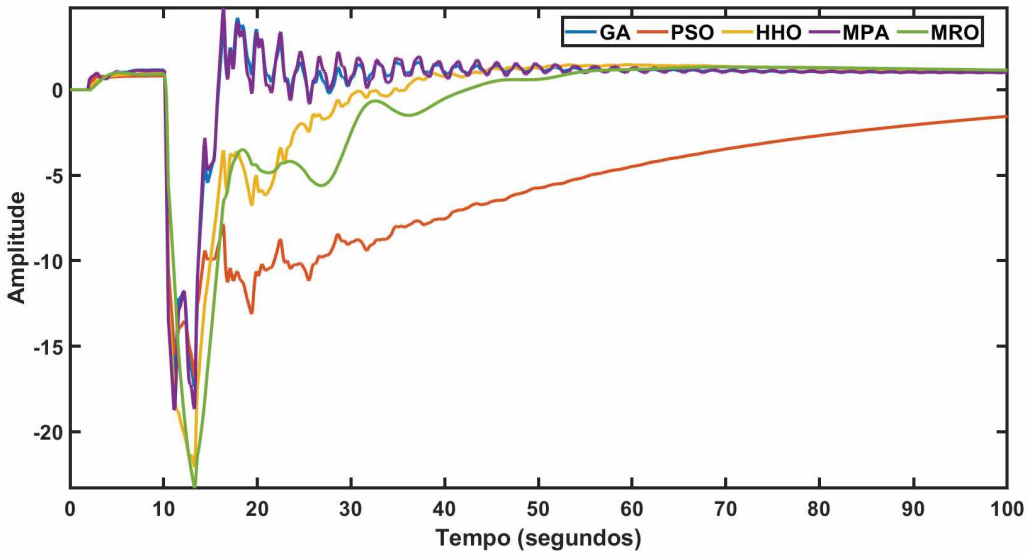
Meta-heurística	K_{P1}	K_{I1}	K_{D1}	K_{P2}	K_{I2}	K_{D2}	K_{P3}	K_{I3}	K_{D3}
MPA	0,92	0,45	1,43	-1,25	-0,15	-1,54	3,64	1,47	4,35
GA	0,72	0,39	1,21	-1,17	-0,19	-1,47	3,26	1,51	3,98
HHO	0,79	0,44	1,33	-1,25	-0,18	-1,42	3,40	0,52	0,80
PSO	0,69	0,36	0,85	-1,10	-0,17	-1,44	4,73	0,10	2,06
MRO	0,64	0,37	0,21	-1,26	-0,21	-0,92	3,75	0,40	-0,99



(a) Primeira saída.



(b) Segunda saída.



(c) Terceira saída.

Figura 5.6: Comparação do controlador PID para as saídas do sistema de OR.

5.2.2 FOPID

No controle da OR, o controlador FOPID não obteve resultados satisfatórios, com todos os melhores controladores tendo sua resposta indo para infinito. Esse péssimo desempenho nesse processo se deveu ao fato de o algoritmo ter de lidar com muitas variáveis de decisão, o que dificultou tremendamente sua convergência para um bom resultado. Com isso, o FOPID foi inferior ao PID clássico para o controle desse processo, como pode ser visto pelos altíssimos valores de ITSE da Tabela 5.9, onde 'Inf' remete a um valor infinito.

Tabela 5.9: Desempenho do controle para o FOPID (OR).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	2249,5	2,0e+09	2,0e+08	2,6e+04	6,0e+08
GA	2º	22180	1,8e+156	3,6e+154	3,0e+10	Inf
HHO	3º	46234	2,3e+23	4,6e+21	3,1e+05	3,2e+22
MRO	4º	55357	2,2e+11	4,5e+09	1,2e+06	3,2e+10
PSO	5º	576220	Inf	Inf	3,4e+155	Inf

Os parâmetros encontrados para os FOPID na OR estão na Tabela 5.8. OS algoritmos não conseguiram convergir em nenhum momento diante da enorme quantidade de parâmetros, o que acabou fazendo com que o processo ficasse instável.

Tabela 5.10: Parâmetros dos melhores FOPID encontrados (OR).

Meta-heurística	K_{P1}	K_{I1}	K_{D1}	λ_1	μ_1	K_{P2}	K_{I2}	K_{D2}	λ_2	μ_2	K_{P3}	K_{I3}	K_{D3}	λ_3	μ_3
MPA	-1.37	0.47	0.62	0.67	0.63	2.06	-2.35	-1.28	0.21	0.88	4.60	4.99	2.42	1	0.86
GA	0.08	0.78	-0.98	0.39	0.27	3.72	-2.50	-2.41	0.18	0.49	-13.58	14.18	16.95	0.64	0.36
HHO	-0.78	0.88	0.24	0.42	0.98	-0.07	0.35	-0.56	0.001	0.29	1.82	4.86	6.89	1	0.20
MRO	-0.11	0.03	0.15	0.89	0.89	0.03	-0.004	-0.007	0.89	0.89	-0.18	2.46	0.09	0.89	0.89
PSO	-24.02	-5.96	29.69	0	0	18.13	-19.17	-0.81	0	1	-0.12	1.27	2.58	0.79	1

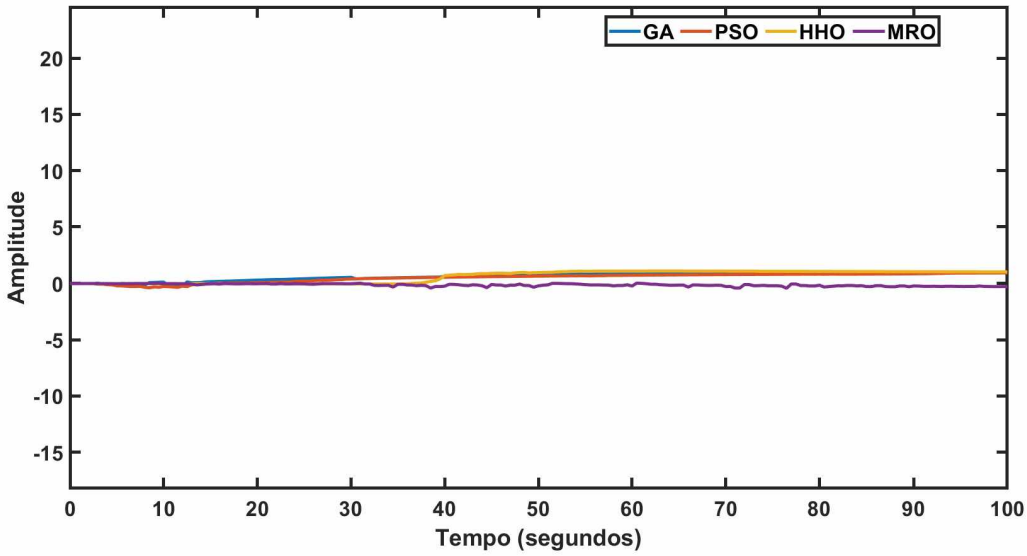
5.2.3 APID

Pela tabela 5.11 pode-se ver que o GA foi a meta-heurística que teve o melhor resultado, enquanto que MRO e principalmente MPA foram inferiores. As medidas estatísticas apontam que apesar de ter ficado em 3º lugar, o HHO foi o que obteve o melhor desempenho se levar em consideração as medidas gerais como a média e a mediana, e não apenas o melhor resultado ao longo das 50 rodadas. Isso quer dizer que o HHO tem mais chances de retornar um valor de ITSE menor que técnicas como o GA e o PSO, que apresentaram um mínimo menor.

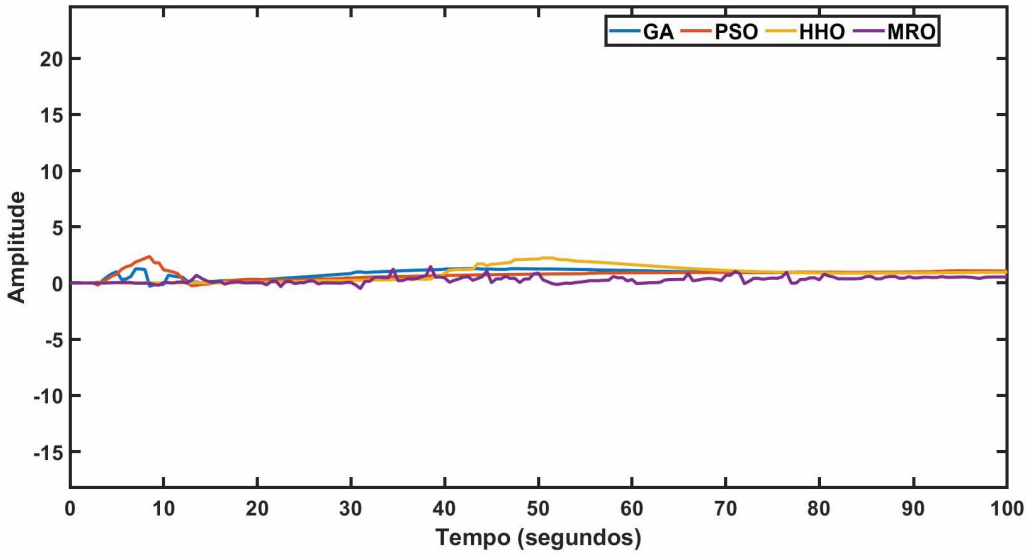
Tabela 5.11: Desempenho do controle para o APID (OR).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
GA	1º	623,57	2,7e+14	5,8e+12	5,9e+07	3,8e+13
PSO	2º	1,05e+03	3,95e+20	9,9e+18	1,7e+11	5,6e+19
HHO	3º	2,2e+03	9,4e+03	7,5e+03	8,8e+03	2,2e+03
MRO	4º	1,1e+04	1,6e+04	1,3e+04	1,3e+04	854
MPA	5º	5,9e+25	1,4e+43	2,8e+41	2,1e+35	1,9e+42

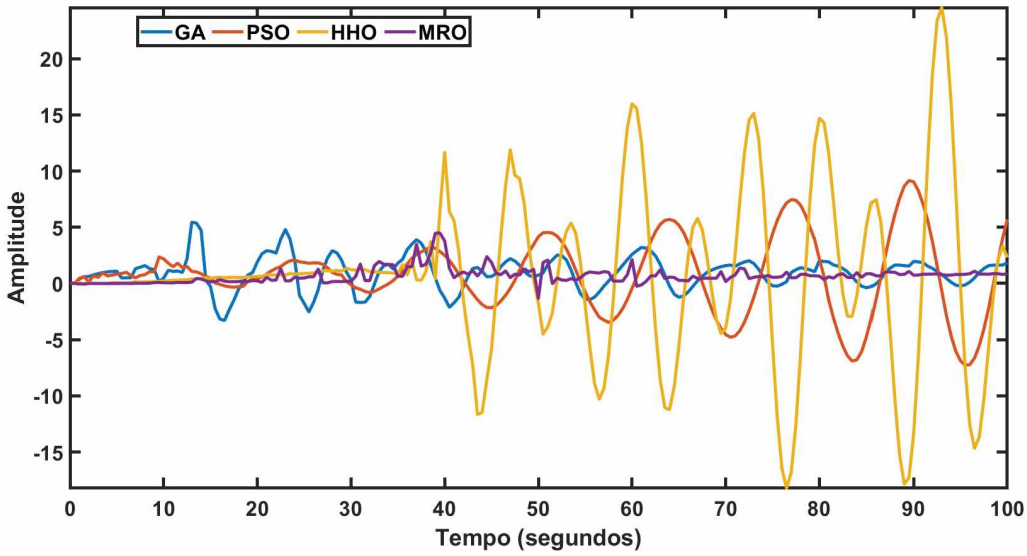
A Figura 5.7 mostra como o GA obteve um resultado superior às outras meta-heurísticas. O HHO foi mais rápido mas teve um *overshoot* alto na segunda saída e demasiada oscilação na terceira, enquanto que o PSO não subiu tão rápido e teve *overshoot* um *overshoot* maior que o GA, e portanto o GA foi o mais balanceado, obtendo assim o menor ITSE.



(a) Primeira saída.



(b) Segunda saída.



(c) Terceira saída.

Figura 5.7: Comparação do controlador APID para as saídas do sistema de OR.

5.3 Sistema de Pulverização por Moinho de Bolas

O Sistema de Pulverização por Moinho de Bolas (do inglês *Ball Mill Pulverizing System*, BMPS) foi exposto por Menhas et al. (2012), sendo um sistema utilizado para pulverizar carvão em usinas termelétricas a carvão. Outras utilidades desse tipo de processo com outros modelos são na pulverização de alumínio (Ramezani e Neitzert (2012)) ou ferro (Muñoz et al. (2007)), por exemplo.

O sistema é MIMO 2 x 2, e as saídas correspondem à temperatura de saída T e à pressão negativa de entrada P do processo. Dessa forma as duas entradas do sistema em malha aberta são a abertura de ar quente para controlar T, e a abertura de ar reciclado para controlar P. O modelo matemático é descrito por

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{3,5}{(80s+1)^3} & \frac{-0,14}{(60s+1)^2} \\ \frac{-2}{(8s+1)^2} & \frac{-0,18}{10s+1} \end{bmatrix}, \tag{5.4}$$

que não é um processo dominante diagonalmente, e portanto para reduzir o impacto da interação entre os processos é colocado um desacoplador de regime permanente (Menhas et al. (2012)), dado por

$$D(s) = G^{-1}(0) = \begin{bmatrix} D11 & D12 \\ D21 & D22 \end{bmatrix} = \begin{bmatrix} 0,1978 & -0,1538 \\ -2,1978 & -3,8462 \end{bmatrix}. \tag{5.5}$$

O esquema feito no Simulink para executar as simulações foi o da Figura 5.8, onde $T0$ e $P0$ são os *setpoints* para temperatura e pressão respectivamente, eT e eP são os erros de temperatura e pressão, uT e uP são os sinais de controle dos PIDs 1 e 2, T e P são as saídas do sistema (temperatura e pressão), e ref e Y são os valores de referência e saída agrupados para cálculo de função *fitness*.

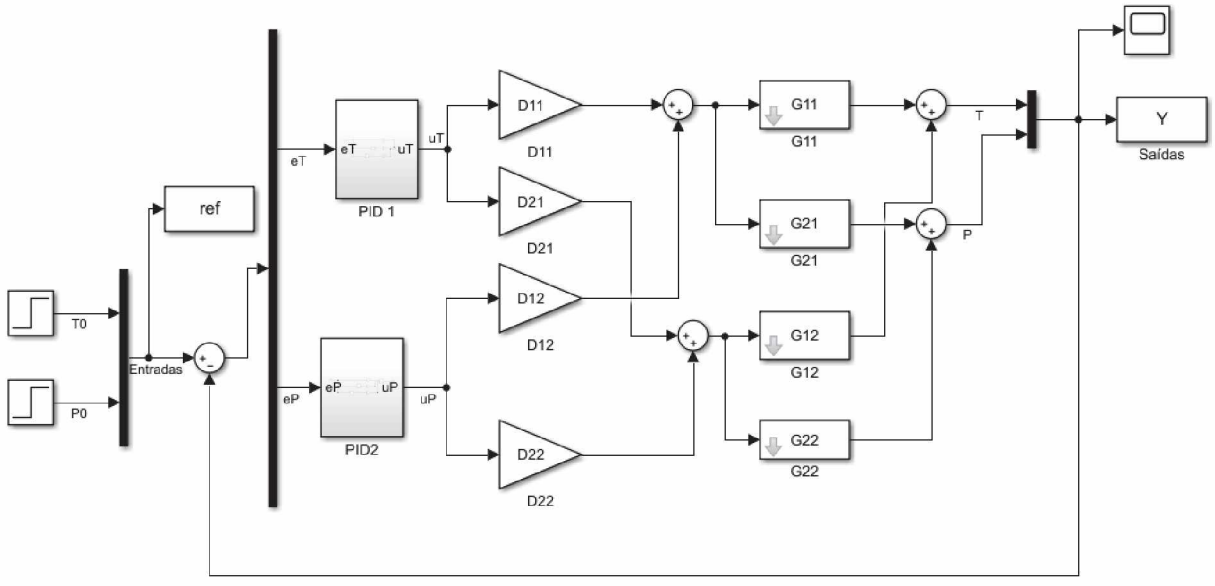


Figura 5.8: Esquema no Simulink do BMPS.

5.3.1 PID clássico

Os resultados para o BMPS está na Tabela 5.12. O desempenho foi bem semelhante, e os algoritmos alcançaram o mínimo global dentro das limitações de valores impostas, com o MPA obtendo o melhor resultado e o GA o pior. GA e PSO ficaram em 4º e 5º lugares, mas nessa configuração de controlador e processo os resultados foram próximos, o que não torna essas técnicas quantitativamente menos eficientes. A colocação final foi escolhida com base não apenas no mínimo, mas levando em consideração as medidas estatísticas.

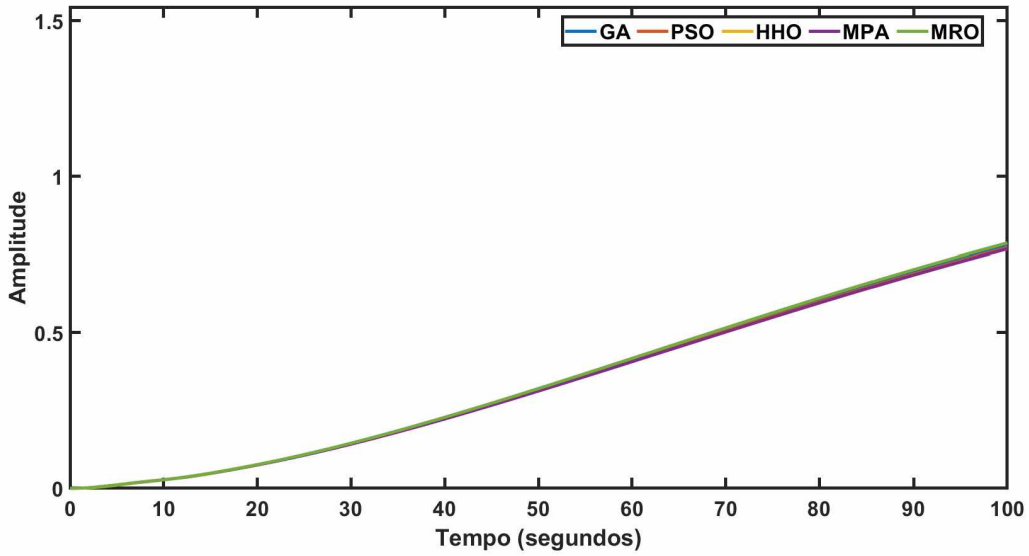
Tabela 5.12: Desempenho do controle para o PID clássico (BMPS).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	871,10	871,19	871,11	871,10	0,01
HHO	2º	871,10	873,97	872,36	871,98	0,81
MRO	3º	871,10	896,60	873,46	873,24	3,50
PSO	4º	871,10	956,14	874,58	871,12	16,82
GA	5º	871,73	1052,7	890,29	877,87	34,43

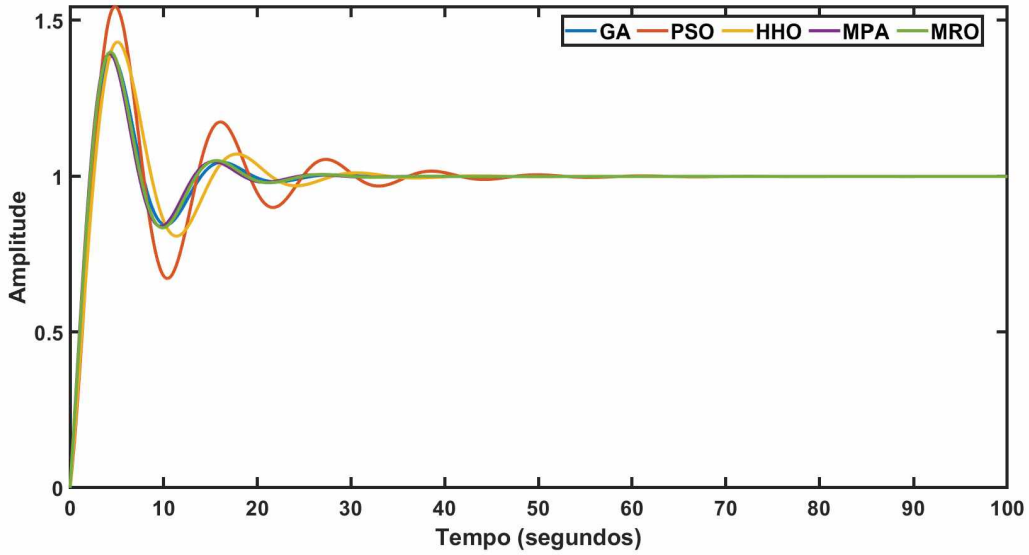
Os valores de parâmetro dos melhores controladores encontrados pelas meta-heurísticas estão na Tabela 5.13. Como o apresentado na tabela é o melhor controlador encontrado, os resultados acabaram sendo iguais para 4 das 5 técnicas de otimização que foram capazes de encontrar o mínimo local (dentro das limitações de valores impostas). A Figura 5.9 explicita bem isso, onde é possível ver que as curvas do GA foram separadas das curvas das outras 4 técnicas, que apresentaram o mesmo comportamento.

Tabela 5.13: Parâmetros dos melhores PID encontrados (BMPS).

Meta-heurística	K_{P1}	K_{I1}	K_{D1}	K_{P2}	K_{I2}	K_{D2}
MPA	5	0,04	0	5	5	0
HHO	5	0,04	0	5	5	0
MRO	5	0,04	0	5	5	0
PSO	5	0,04	0	5	5	0
GA	4,99	0,04	0,19	4,99	4,76	0,48



(a) Primeira saída.



(b) Segunda saída.

Figura 5.9: Comparação do controlador PID para as saídas do sistema de BMPS.

5.3.2 FOPID

O desempenho da sintonia do controlador FOPID foi bom por parte de todos os 5 algoritmos testados, e o ITSE das melhores soluções foi semelhante em cada algoritmo, com o MPA obtendo a posição de melhor meta-heurística. O MPA foi não só a técnica de otimização que retornou o menor ITSE, como também teve os menores valores estatísticos, e suas soluções ao longo das 50 rodadas foram bem próximas, sendo evidenciado pelo seu desvio padrão abaixo de 1.

Tabela 5.14: Desempenho do controle para o FOPID (BMPS).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
MPA	1º	304,56	304,67	304,59	304,59	0,03
MRO	2º	304,64	705,79	377,22	306,48	150,78
HHO	3º	304,65	525,58	323,95	305,54	60,13
GA	4º	319,96	341,48	328,13	328,32	6,13
PSO	5º	322,45	424,70	362,02	355,82	24,45

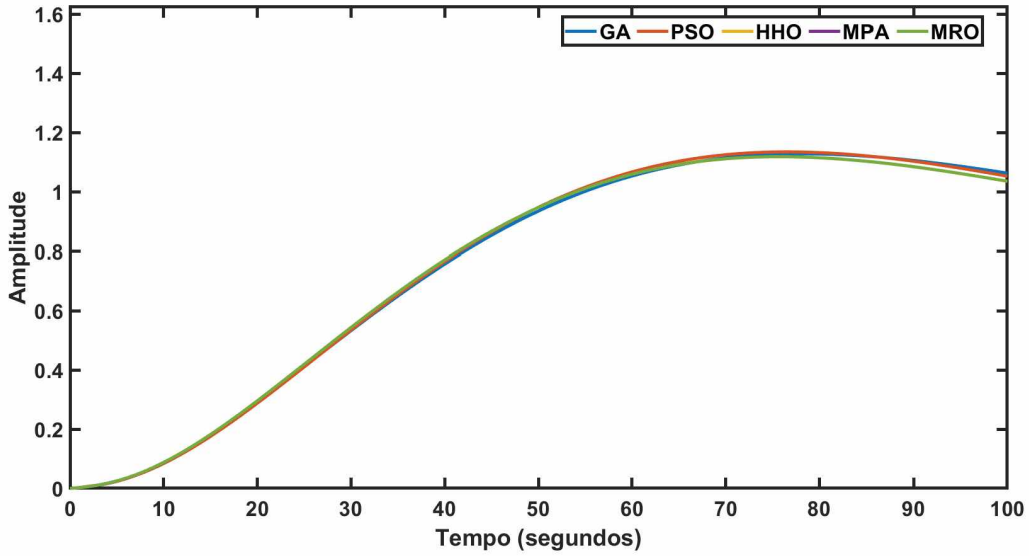
Os valores de controlador FOPID que retornaram os valores de ITSE da tabela 5.14 estão na tabela 5.15. Os valores evidenciam que para essa aplicação, valores nulos de K_{P1} e K_{I1} com valores máximos de K_{D1}, K_{P2}, K_{I2}, K_{D2} retornam o melhor desempenho, quando combinados com λ_1 e λ_2 máximos e baixos μ_1 e μ_2.

Tabela 5.15: Parâmetros dos melhores FOPID encontrados (BMPS).

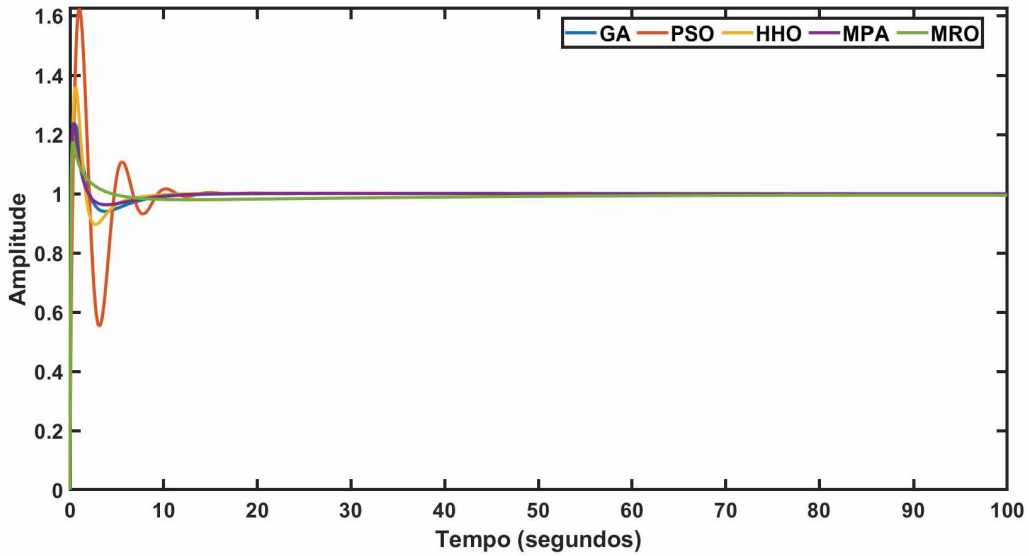
Meta-heurística	K_{P1}	K_{I1}	K_{D1}	λ_1	μ_1	K_{P2}	K_{I2}	K_{D2}	λ_2	μ_2
MPA	3e-05	0	50	1	0,37	49,99	50	49,99	0,99	0,003
MRO	0.001	0	50	0	0,37	50	50	0	1	0
HHO	0,0005	0	50	1	0,37	29,07	49,78	16,18	1	0,12
GA	1,13	1,37	49,93	0,02	0,43	42,92	44,33	26,03	0,89	0,45
PSO	0.001	0	46,22	0	0,35	0.001	28,56	11,25	1	0

Na Figura 5.10 está exposta as respostas do sistema a degraus unitários. Na primeira saída as curvas têm o comportamento parecido, possuindo *overshoot* próximo aos 75 segundos, mas sem conseguir alcançar o *setpoint* desejado de 1. Já na segunda saída as curvas alcançam o *setpoint* em pouco tempo, antes até mesmo de 1 segundo. É possível verificar como o MPA de fato obteve o melhor resultado quando comparado às outras técnicas de otimização.

Quando comparamos as curvas da Figura 5.10 com aquelas da Figura 5.9, pode-se perceber o quanto o FOPID acelerou a resposta do BMPS em relação ao PID clássico, além de diminuir o *overshoot* na segunda saída. Esses fatores evidenciam um desempenho superior do FOPID em relação ao PID clássico.



(a) Primeira saída.



(b) Segunda saída.

Figura 5.10: Comparação do controlador FOPID para as saídas do sistema de BMPS.

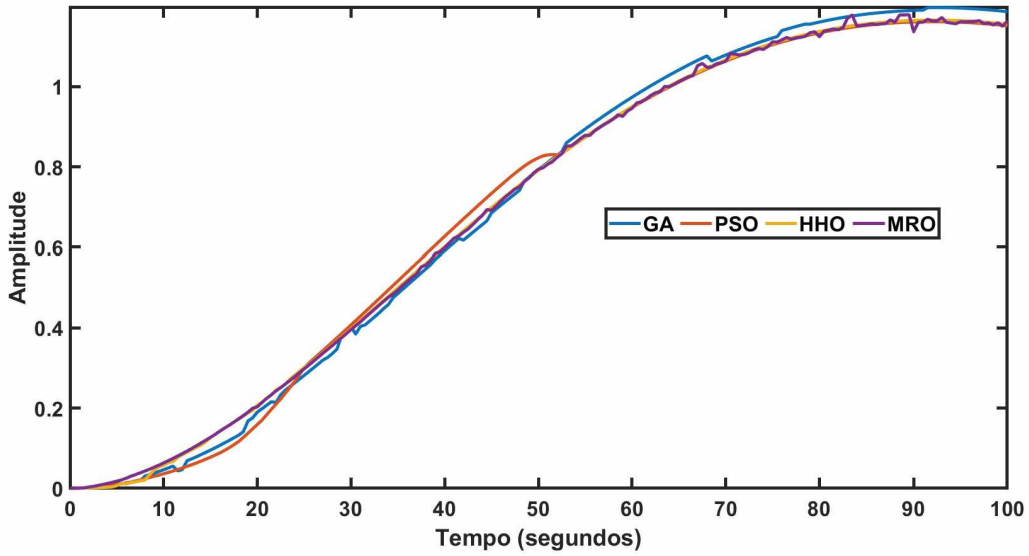
5.3.3 APID

Para esse sistema, PSO, HHO, e MRO obtiveram resultados próximos numericamente, com o PSO obtendo o menor ITSE, além de medidas estatísticas melhores, juntamente ao HHO. Os baixos valores de ITSE para as meta-heurísticas (com exceção do MPA) demonstram uma facilidade maior de se encontrar uma sintonia satisfatória para o APID no BMPS, ao comparar seus resultados aos dos outros sistemas. Isso pode ser observado na tabela 5.16.

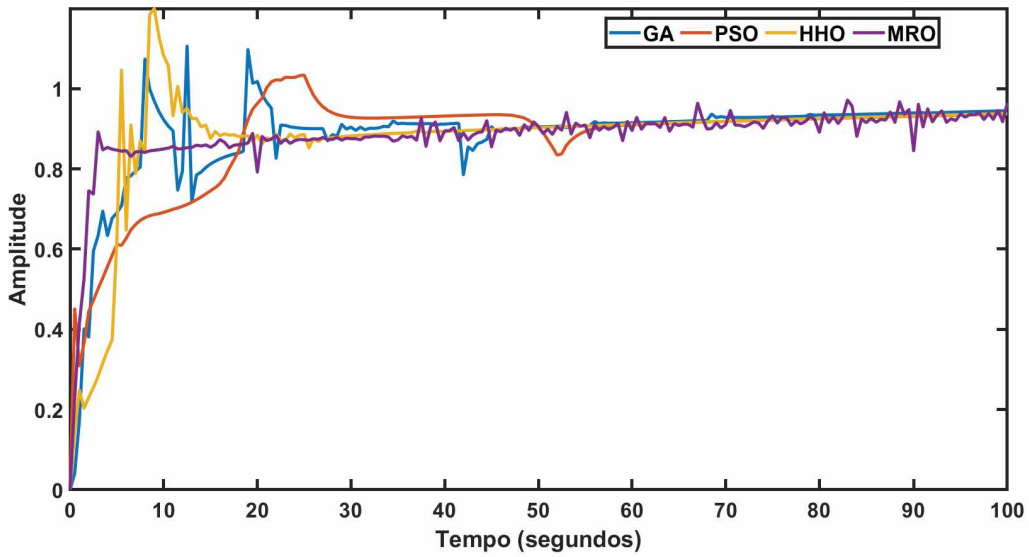
Tabela 5.16: Desempenho do controle para o APID (BMPS).

Meta-heurística	Colocação	Mínimo	Máximo	Média	Mediana	Desvio padrão
PSO	1º	517,36	744,67	571,98	538,69	61,27
HHO	2º	518,32	768,98	562,75	539,41	60,72
MRO	3º	519,89	1036.7	623,05	665,94	110,82
GA	4º	555,04	777,15	626,15	619,64	48,85
MPA	5º	3,1e+04	2,6e+06	7,1e+05	4,8e+05	6,3e+05

A boa sintonia das meta-heurísticas está na Figura 5.11, a qual expõe que o comportamento das curvas foi semelhante, com o HHO tendo o maior *overshoot*, o MRO a curva menos oscilatória, e o PSO obtendo uma resposta mais próxima do *setpoint* mais rápido que os outros (por volta dos 20 segundos), o que lhe conferiu um ITSE mais baixo.



(a) Primeira saída.



(b) Segunda saída.

Figura 5.11: Comparação do controlador APID para as saídas do sistema de BMPS.

5.4 Comparação

5.4.1 Entre meta-heurísticas

Para comparar o desempenho das técnicas de otimização utilizadas no trabalho, adotou-se o seguinte critério: as meta-heurísticas receberam pontos de acordo com sua colocação em cada par de sintonia sistema-controlador. Com isso se recebe 1 ponto para um 1º lugar e 5 para um 5º lugar. Assim a técnica de otimização melhor no geral é aquela com menos pontos.

Tabela 5.17: Classificação das meta-heurísticas em cada estudo de caso

Processo	Controlador	GA	PSO	HHO	MPA	MRO
WB	PID	2º	4º	3º	1º	5º
WB	FOPID	2º	3º	4º	1º	5º
WB	APID	3º	1º	2º	5º	4º
OR	PID	2º	4º	3º	1º	5º
OR	FOPID	2º	5º	3º	1º	4º
OR	APID	1º	2º	3º	5º	4º
BMPS	PID	5º	4º	2º	1º	3º
BMPS	FOPID	4º	5º	3º	1º	2º
BMPS	APID	4º	1º	2º	5º	3º

Tabela 5.18: Desempenho geral de meta-heurísticas.

Meta-heurística	Colocação geral	Nº de pontos
MPA	1º	21
GA	2º	25
HHO	2º	25
PSO	3º	29
MRO	4º	35

Na tabela 5.17 vê-se que o MPA possuiu os melhores desempenhos gerais, mesmo levando em consideração que ele teve o pior desempenho no APID, tendo obtido a 5º colocação nos 3 sistemas. Isso demonstra que o MPA teve dificuldades em convergir para um resultado coerente quando precisou fazer a sintonia em tempo-real, mas no cenário mais simples (PID e FOPID) obteve a maior consistência, sendo o melhor otimizador nos 6 cenários de sintonia.

O GA e o PSO, sendo as técnicas mais usadas na área de sintonia de controladores PID, obtiveram resultados bons, tendo terminado na 2º e 4º colocações. O fato de serem técnicas mais usuais não implicou que seriam menos eficientes na sintonia do que técnicas mais recentemente desenvolvidas.

Com relação ao HHO e ao MRO, pode-se inferir que o MRO é uma meta-heurística que no cenário estruturado nesse trabalho não é capaz de obter os melhores resultados em sintonia de controladores PID dos 3 tipos usados, enquanto que o HHO teve os resultados mais medianos, não obtendo o 1º nem o 5º lugar em nenhum dos 15 cenários de sintonia estudados, mesmo mantendo um desempenho que o coloque no 2º lugar.

5.4.2 Entre controladores

Nessa subseção é feita uma comparação rápida entre o desempenho dos controladores PID dos 3 tipos trabalhados com relação aos menores valores de ITSE encontrados para cada um dos 3 processos estudados. Na tabela 5.19 pode-se observar que cada tipo de PID foi melhor para 1 processo diferente. O PID otimizou melhor o sistema de OR, o FOPID foi melhor para o BMPS, enquanto que o APID foi o melhor para a WB.

Tabela 5.19: Mínimo ITSE dos tipos de controlador.

Controlador	WB	OR	BMPS
PID	13,7	**40,38** (MPA)	871,10
FOPID	8,82	2249,5	**304,56** (MPA)
APID	**7,46** (PSO)	623,57	517,36

Para fazer uma comparação entre qual tipo de PID retorna o melhor desempenho para cada meta-heurística, a tabela 5.20 foi gerada. Para a WB o FOPID foi o melhor controlador em 3 das 5 meta-heurísticas, com o APID vindo logo depois com 2. Na OR o PID clássico foi o melhor controlador para todas as meta-heurísticas, isso se deve ao fato de que esse processo possui o maior número de parâmetros, o que torna mais fácil a sintonia a partir de um controlador mais estável como o PID clássico. Para o BMPS o FOPID foi o melhor em todas, e dessa forma o FOPID foi o melhor controlador em 8 dos 15 casos, mostrando a força do cálculo fracionário.

Tabela 5.20: Melhor controlador para cada sintonia de meta-heurística.

Meta-heurística	WB	OR	BMPS
MPA	8,82 (FOPID)	40,38 (PID)	304,56 (FOPID)
GA	9,43 (FOPID)	41,64 (PID)	319,96 (FOPID)
HHO	12,43 (APID)	42,26 (PID)	304,65 (FOPID)
PSO	7,46 (APID)	44,76 (PID)	322,44 (FOPID)
MRO	13,18 (FOPID)	53,11 (PID)	304,63 (FOPID)

5.5 Comentários sobre o capítulo

Nesse capítulo os 3 processos a serem controlados foram apresentados, e em seguida as sintonias feitas por cada meta-heurística para cada tipo de controlador (PID clássico, FOPID, e APID) foram demonstradas numericamente através dos valores de função objetivo (ITSE) alcançados ao longo das 50 rodadas de simulação.

A partir disso, foi possível verificar quais técnicas de otimização se saíram melhor, formando uma classificação geral implicando o MPA como a técnica de melhor sintonia. Ainda foi possível saber qual controlador foi melhor em cada processo, assim como saber que o FOPID obteve o maior número de sintonias ótimas levando em conta todos os controladores e meta-heurísticas.

Capítulo 6

Conclusão

Nesse trabalho realizou-se um estudo de aplicação de meta-heurísticas na sintonia de controladores PID de 3 variantes diferentes: a variante clássica, a variante de ordem fracionária (FOPID), e a variante adaptativa (APID). Para esse processo de otimização utilizou-se 5 meta-heurísticas diferentes: GA, PSO, HHO, MPA, e MRO. As 2 primeiras sendo as 2 técnicas mais usadas na área de sintonia de controlador PID por meta-heurística, enquanto que as outras 3 são mais novas. O objetivo do trabalho então foi o de avaliar se o uso das meta-heurísticas estudadas seria satisfatório na sintonia ao se usar de função objetivo o ITSE, e além disso se as 3 meta-heurísticas mais recentemente desenvolvidas seriam mais eficientes do que as 2 mais usadas.

Foram usados 3 processos para testar a eficiência dos 5 algoritmos na sintonia dos tipos de controlador PID: a WB, a OR, e o BMPS. Para cada uso de algoritmo em cada processo (totalizando 15 cenários para cada tipo de controlador) foram comparados os valores de ITSE das sintonias entre as meta-heurísticas usadas e atribuídas posições sobre quais foram melhores.

Ao final desse processo chegou-se às seguintes conclusões quanto aos tipos de controladores: no PID clássico, todas as sintonias foram satisfatórias, obtendo valores razoáveis. No FOPID as sintonias para a WB e para o BMPS foram boas, mas a sintonia para a OR se mostrou numericamente complicada devido ao alto número de parâmetros a serem otimizados (15 parâmetros). No APID o MPA, que havia sido o melhor otimizador nos outros tipos de controlador, se mostrou completamente ineficiente, não conseguindo obter resultados que convergissem para um comportamento adequado por parte do controlador. Já os outros 4 otimizadores foram bem sucedidos na sintonia ao se obterem valores de ITSE e curvas de resposta coerentes. Além disso, observou-se que o APID otimizou melhor a WB, o PID clássico otimizou melhor a OR, e o FOPID otimizou melhor o BMPS.

Considerou-se que o FOPID foi o melhor controlador no geral, já que ele retornou a maior frequência de sintonias boas em cada meta-heurística, enquanto que o PID clássico se mostrou mais efetivo na sintonia com alto número de parâmetros, ao se atingir um melhor desempenho na resposta da OR. O APID obteve resultados satisfatórios, contudo devido à sua natureza iterativa, ele acaba sendo inconsistente a depender do sistema a ser sintonizado.

Com relação à comparação entre meta-heurísticas, averiguou-se que o MPA foi a meta-heurística mais bem avaliada por esse trabalho, e GA e HHO ficaram tecnicamente empatados no 2º lugar pelo critério especificado na subseção 5.4.1. Elas são seguidas por PSO e MRO, nessa ordem. O excepcional desempenho do MPA no PID clássico e no FOPID demonstra que essa é uma técnica eficiente nesses tipos de sintonia, em detrimento de sua fraca capacidade na sintonia do APID. O MRO não obteve bons resultados e não conseguiu ser o melhor sintonizador em nenhum cenário do estudo, enquanto que o HHO se manteve mediano, sendo mais constante às

mudanças de cenário na aplicações usadas. Quanto ao GA e ao PSO, o fato de serem técnicas mais usadas na área de sintonia para PID não influenciou positiva nem negativamente na sintonia feita nesse trabalho, e seus resultados foram satisfatórios no geral.

Esse trabalho gerou publicações importantes como uma comparação entre 4 variantes do HHO: Cultural, com matriz de covariância, com mecânica quântica, e oposicional, na sintonia de controladores PID clássicos para o controle do BMPS (Lúcio et al. (2021a)), trabalho enviado para o Congresso Internacional de Engenharia Mecânica (COBEM). Bem como uma publicação para o Congresso Brasileiro de Inteligência Computacional (CBIC) sobre uma comparação entre variantes quântica e oposicional do MPA na sintonia do BMPS (Lúcio et al. (2021b)).

A pesquisa sobre meta-heurísticas na sintonia de controladores pode seguir algumas possíveis direções. É possível expandir o número de meta-heurísticas usadas na análise comparativa, por meio também de modificações como a adição de aprendizagem por oposição (Xu et al. (2014a)) e de mecânicas quânticas (Sun et al. (2016)). Outra possibilidade é fazer a sintonia de outros tipos de controlador como MPC ou LQR, e ver como o desempenho deles varia para alguma determinada aplicação. Processos com outras dinâmicas podem ser usados para se entender como as técnicas de otimização reagem a diferentes situações.

Referências

Abbasi, E. e Naghavi, N. (2017). Offline auto-tuning of a pid controller using extended classifier system (xcs) algorithm. *Journal of Advances in Computer Engineering and Technology*, 3(1):39–44.

Aboelhassan, A., Abdelgeliel, M., Zakzouk, E. E. e Galea, M. (2020). Design and implementation of model predictive control based pid controller for industrial applications. *Energies*, 13(24). 6594.

AbouOmar, M. S., Zhang, H.-J. e Su, Y.-X. (2019). Fractional order fuzzy pid control of automotive pem fuel cell air feed system using neural network optimization algorithm. *Energies*, 12(8). 1435.

Ahmad, M. A., Azuma, S.-i. e Sugie, T. (2014). Performance analysis of model-free pid tuning of mimo systems based on simultaneous perturbation stochastic approximation. *Expert Systems with Applications*, 41(14):6361–6370.

Akbarzadeh Totonchi, M. R. et al. (2008). Magnetic optimization algorithms, a new synthesis. Em *IEEE international conference on evolutionary computation*, páginas 2659–2664, Honk Kong, China. IEEE.

Altinoz, O. T., Kosalay, I. e Gezer, D. (2020). Optimal controller design for speed governors of hydroelectric power plant. *Advances in Electrical and Electronic Engineering*, 18(2):72–84.

An, W., Wang, H., Sun, Q., Xu, J., Dai, Q. e Zhang, L. (2018). A pid controller approach for stochastic optimization of deep networks. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 8522–8531, Salt Lake City, United States. IEEE.

Arruda, L. V., Swiech, M., Delgado, M. e Neves-Jr, F. (2008). Pid control of mimo process based on rank niching genetic algorithm. *Applied Intelligence*, 29(3):290–305.

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, 169:1–12.

Åström, K. J. e Hägglund, T. (1995). *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, North Carolina, United States.

Åström, K. J., Hägglund, T. e Åström, K. J. (2006). *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park, Pittsburgh, United States.

Atacak, I. e Kucuk, B. (2017). Pso-based pid controller design for an energy conversion system using compressed air. *Tehnicki Vjesnik-Technical Gazette*, 24(3):671–679.

Atashpaz-Gargari, E. e Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. Em *2007 IEEE congress on evolutionary computation*, páginas 4661–4667. IEEE.

Augusti Lindiya, S., Subashini, N. e Vijayarekha, K. (2019). Cross regulation reduced optimal multivariable controller design for single inductor dc-dc converters. *Energies*, 12(3). 477.

Banks, A., Vincent, J. e Phalp, K. (2009). Natural strategies for search. *Natural Computing*, 8(3). 547.

Bednarz, J. C. (1988). Cooperative hunting harris' hawks (parabuteo unicinctus). *Science*, 239(4847):1525–1527.

Belhaj, W. e Boubaker, O. (2015). Multivariable pid control via ilmis: Preformances assessment. *International Journal on Smart Sensing & Intelligent Systems*, 8(4):1896–1916.

Bhookya, J. e Jatoth, R. K. (2020a). Fractional order pid controller design for multivariable systems using tlbo. *Chemical Product and Process Modeling*, 1(15). 20190061.

Bhookya, J. e Jatoth, R. K. (2020b). Fractional order pid controller design for multivariable systems using tlbo. *Chemical Product and Process Modeling*, 15(2). 0061.

Birbil, Ş. İ. e Fang, S.-C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, 25(3):263–282.

Borase, R. P., Maghade, D., Sondkar, S. e Pawar, S. (2021). A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818–827.

Boubakir, A., Labiod, S. e Boudjema, F. (2012). A stable self-tuning proportional-integral-derivative controller for a class of multi-input multi-output nonlinear systems. *Journal of Vibration and Control*, 18(2):228–239.

Boussaïd, I., Lepagnot, J. e Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117.

Boyd, S., Hast, M. e Åström, K. J. (2016). Mimo pid tuning via iterated lmi restriction. *International Journal of Robust and Nonlinear Control*, 26(8):1718–1731.

Cafagna, D. (2007). Fractional calculus: A mathematical tool from the past for present engineers [past and present]. *IEEE Industrial Electronics Magazine*, 1(2):35–40.

Camacho, E., Rubio, F. e Hughes, F. (1992). Self-tuning control of a solar power plant with a distributed collector field. *IEEE Control Systems Magazine*, 12(2):72–78.

Chang, W.-D. (2006). A multi-crossover genetic approach to multivariable pid controllers tuning. *Expert Systems with Applications*, 33:620–626.

Chang, W.-D. e Chen, C.-Y. (2014). Pid controller design for mimo processes using improved particle swarm optimization. *Circuits, Systems, and Signal Processing*, 33(5):1473–1490.

Chang, Y., Shieh, L.-S., Liu, C. R. e Cofie, P. (2009). Digital modeling and pid controller design for mimo analog systems with multiple delays in states, inputs and outputs. *Circuits, Systems & Signal Processing*, 28(1):111–145.

Che Razali, M., Abdul Wahab, N., Balaguer, P., Rahmat, M. e Samsudin, S. I. (2015). Singularly perturbation method applied to multivariable pid controller design. *Mathematical Problems in Engineering*, 2015. 818353.

Chen, B.-S. e Cheng, Y.-M. (1998). A structure-specified h/sup/spl infin//optimal control design for practical applications: a genetic approach. *IEEE Transactions on Control Systems Technology*, 6(6):707–718.

Chen, Q., Luan, X. e Liu, F. (2013). Analytical design of centralized pi controller for high dimensional multivariable systems. *IFAC Proceedings Volumes*, 46(32):643–648.

Cohen, G. (1953). Theoretical consideration of retarded control. *Transactions of the ASME*, 75:827–834.

Cuevas, E., Cienfuegos, M., Zaldívar, D. e Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications*, 40(16):6374–6384.

Cuevas, E., Echavarría, A. e Ramírez-Ortegón, M. A. (2014). An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. *Applied Intelligence*, 40(2):256–272.

Dachang, Z., Baolin, D., Puchen, Z. e Shouyan, C. (2020). Constant force pid control for robotic manipulator based on fuzzy neural network algorithm. *Complexity*, 2020. 3491845.

Dangor, M., Dahunsi, O. A., Pedro, J. O. e Ali, M. M. (2014). Evolutionary algorithm-based pid controller tuning for nonlinear quarter-car electrohydraulic vehicle suspensions. *Nonlinear Dynamics*, 78(4):2795–2810.

Djari, A., Bouden, T. e Boulkroune, A. (2013). Design of fractional-order pid controller (fopid) for a class of fractional-order mimo systems using a particle swarm optimization (pso) approach. Em *3rd International Conference on Systems and Control*, páginas 1055–1060, Algiers, Algeria. IEEE.

Dorigo, M., Maniezzo, V. e Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

Du, K.-L., Swamy, M. et al. (2016). *Search and optimization by metaheuristics*. Springer, Basel, Switzerland.

Dwi, L., Herlambang, S. e Muhammad, R. D. (2017). Optimization pitch angle controller of rocket system using improved differential evolution algorithm. *International Journal of Advances in Intelligent Informatics*, 3(1):27–34.

Erol, B. e Delibaşı, A. (2018). Proportional–integral–derivative type h∞ controller for quarter car active suspension system. *Journal of Vibration and Control*, 24(10):1951–1966.

Erol, O. K. e Eksin, I. (2006). A new optimization method: big bang–big crunch. *Advances in Engineering Software*, 37(2):106–111.

Eskandar, H., Sadollah, A., Bahreininejad, A. e Hamdi, M. (2012). Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, 110:151–166.

Faramarzi, A., Heidarinejad, M., Mirjalili, S. e Gandomi, A. H. (2020). Marine predators algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications*, 152. 113377.

Filmalter, J. D., Dagorn, L., Cowley, P. D. e Taquet, M. (2011). First descriptions of the behavior of silky sharks, carcharhinus falciformis, around drifting fish aggregating devices in the indian ocean. *Bulletin of Marine Science*, 87(3):325–337.

Fogel, D. B. (1998). *Artificial intelligence through simulated evolution*, páginas 227–296. Wiley-IEEE Press.

Formato, R. (2007). Central force optimization: a new metaheuristic with applications in applied electromagnetics. prog electromagn res 77: 425–491.

Geem, Z. W., Kim, J. H. e Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68.

Gil, P., Lucena, C., Cardoso, A. e Palma, L. B. (2014). Gain tuning of fuzzy pid controllers for mimo systems: a performance-driven approach. *IEEE Transactions on Fuzzy Systems*, 23(4):757–768.

Gilbert, A., Yousef, A., Natarajan, K. e Deighton, S. (2003). Tuning of pi controllers with one-way decoupling in 2× 2 mimo systems based on finite frequency response data. *Journal of Process Control*, 13(6):553–567.

Goud, H. e Swarnkar, P. (2019). Investigations on metaheuristic algorithm for designing adaptive pid controller for continuous stirred tank reactor. *MAPAN*, 34(1):113–119.

Guha, D., Roy, P. K. e Banerjee, S. (2020). Grasshopper optimization algorithm scaled fractional order pi-d controller applied to reduced order model of load frequency control system. *International Journal of Modelling and Simulation*, 40(3):217–242.

Hägglund, T. (2019). The one-third rule for pi controller tuning. *Computers & Chemical Engineering*, 127:25–30.

Halim, A. H. e Ismail, I. (2017). Single and multiple variables control using tree physiology optimization. Em *UTP-UMP Symposium on Energy Systems 2017 (SES 2017)*, volume 131, Perak, Malaysia. EDP Sciences. 03017.

Halim, A. H. e Ismail, I. (2019). Tree physiology optimization on siso and mimo pid control tuning. *Neural Computing and Applications*, 31(11):7571–7581.

Hang, C. C. e Åström, K. (1988). Practical aspects of pid auto-tuners based on relay feedback. *IFAC Proceedings Volumes*, 21(7):153–158.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. e Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, 97:849–872.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314.

Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, Cambridge, United States.

Hussain, K., Salleh, M. N. M., Cheng, S. e Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191–2233.

Iruthayarajan, M. W. e Baskar, S. (2009). Evolutionary algorithms based design of multivariable pid controller. *Expert Systems with Applications*, 36(5):9159–9167.

Iruthayarajan, M. W. e Baskar, S. (2010). Covariance matrix adaptation evolution strategy based design of centralized pid controller. *Expert Systems with Applications*, 37(8):5775–5781.

Ishizuka, S. e Kajiwara, I. (2015). Online adaptive pid control for mimo systems using simultaneous perturbation stochastic approximation. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 9(2). 14-00390.

Jain, M., Singh, V. e Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*, 44:148–175.

Jin, Q. e Liu, Q. (2014). Multi-loop pi/pid controllers design for disturbance rejection based on non-parametric effective model and non-convex optimisation. *IET Control Theory & Applications*, 8(15):1499–1512.

Jing, X. e Cheng, L. (2012). An optimal pid control algorithm for training feedforward neural networks. *IEEE Transactions on Industrial Electronics*, 60(6):2273–2283.

Kang, J., Meng, W., Abraham, A. e Liu, H. (2014). An adaptive pid neural network for complex nonlinear system control. *Neurocomputing*, 135:79–85.

Karaboga, D. e Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

Karami, M., Tavakolpour-Saleh, A. R. e Norouzi, A. (2020). Optimal nonlinear pid control of a micro-robot equipped with vibratory actuator using ant colony algorithm: Simulation and experiment. *Journal of Intelligent & Robotic Systems*, 99(3):773–796.

Karimi-Ghartemani, M., Zamani, M., Sadati, N. e Parniani, M. (2007). An optimal fractional order controller for an avr system using particle swarm optimization algorithm. Em *2007 Large Engineering Systems Conference on Power Engineering*, páginas 244–249, Quebec, Canada. IEEE.

Kassai, M., Kajtar, L. e Nyers, J. (2019). Experimental optimization of energy consumption for direct current refrigerator by pid controller tuning and comparison with on/off refrigerator. *Thermal Science*, 23(2 Part B):941–952.

Kaveh, A. e Zolghadr, A. (2016). A novel meta-heuristic algorithm: tug of war optimization. *Iran University of Science & Technology*, 6(4):469–492.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. Em *Proceedings of ICNN'95-international conference on neural networks*, volume 4, páginas 1942–1948, Perth, Australia. IEEE.

Khan, L., Qamar, S. e Khan, U. (2016). Adaptive pid control scheme for full car suspension control. *Journal of the Chinese Institute of Engineers*, 39(2):169–185.

Khooban, M. H., Alfi, A. e Abadi, D. N. M. (2013). Teaching-learning-based optimal interval type-2 fuzzy pid controller design: A nonholonomic wheeled mobile robots. *Robotica*, 31(7). 1059.

Kiyak, E. (2016). Tuning of controller for an aircraft flight control system based on particle swarm optimization. *Aircraft Engineering and Aerospace Technology*, 88(6):799–809.

Ko, C.-N. e Wu, C.-J. (2008). A pso-tuning method for design of fuzzy pid controllers. *Journal of Vibration and Control*, 14(3):375–395.

Kondo, H. e Ochi, Y. (2011). Mimo pid controller design based on integral-type optimal servo-mechanism and its extension to model-following-type. *SICE Journal of Control, Measurement, and System Integration*, 4(6):439–444.

Koza, J. R. e Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, Cambridge, United States.

Kumar, L., Narang, D. et al. (2018). Tuning of fractional order piλ dμ controllers using evolutionary optimization for pid tuned synchronous generator excitation system. *IFAC-PapersOnLine*, 51(4):859–864.

Lakshmanaprabu, S., Nasir, A. W. e Banu, U. S. (2017). Design of centralized fractional order pi controller for two interacting conical frustum tank level process. *Journal of Applied Fluid Mechanics*, 10:23–32.

Lengare, M. et al. (2008). Auto tuning of pid controller for mimo processes. *World Academy of Science, Engineering and Technology*, 45:306–309.

Li, N., Yang, H. e Mu, A. (2019). Improved grey particle swarm optimization and new luus-jaakola hybrid algorithm optimized imc-pid controller for diverse wing vibration systems. *Complexity*, 2019. 8283178.

Li, X.-l., Lu, F., Tian, G.-h. e Qian, J.-x. (2004). Applications of artificial fish school algorithm in combinatorial optimization problems. *Journal of Shandong University (Engineering Science)*, 5. 015.

Liao, G. e Zhao, J. (2019). Auto-tuning for cascade pid height position controller of rotorcraft. Em *2018 International Joint Conference on Metallurgical and Materials Engineering (JCMME 2018)*, volume 277, Wellington, New Zealand. EDP Sciences. 01008.

Liu, L., Pan, F. e Xue, D. (2015). Fractional-order controller design for oscillatory fractional time-delay systems based on the numerical inverse laplace transform algorithms. *Mathematical Problems in Engineering*, 2015. 917382.

Liu, X.-h., Chen, X.-h., Zheng, X.-h., Li, S.-p. e Wang, Z.-b. (2014). Development of a ga-fuzzy-immune pid controller with incomplete derivation for robot dexterous hand. *The Scientific World Journal*, 2014. 564137.

Liu, Y. e Passino, K. (2002). Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors. *Journal of Optimization Theory and Applications*, 115(3):603–628.

Luan, X., Chen, Q. e Liu, F. (2015). Equivalent transfer function based multi-loop pi control for high dimensional multivariable systems. *International Journal of Control, Automation and Systems*, 13(2):346–352.

Lúcio, Y. L. S., Aquino, L. S. e Coelho, L. d. S. (2021a). Harris hawks optimization approaches on the multivariable controller tuning. Em *Anais do 26 Congresso Internacional de Engenharia Mecânica*, páginas 1–10. ABCM.

Lúcio, Y. L. S., Aquino, L. S. e Coelho, L. d. S. (2021b). Marine predators algorithm approaches on a multivariable fractional pid controller tuning. Em Filho, C. J. A. B., Siqueira, H. V., Ferreira, D. D., Bertol, D. W. e ao de Oliveira, R. C. L., editores, *Anais do 15 Congresso Brasileiro de Inteligência Computacional*, páginas 1–7, Joinville, SC. SBIC.

Mahmoodabadi, M. J., Maafi, R. A. e Taherkhorsandi, M. (2017a). An optimal adaptive robust pid controller subject to fuzzy rules and sliding modes for mimo uncertain chaotic systems. *Applied Soft Computing*, 52:1191–1199.

Mahmoodabadi, M. J., Taherkhorsandi, M. e Talebipour, M. (2017b). Adaptive robust pid sliding control of a liquid level system based on multi-objective genetic algorithm optimization. *Control and Cybernetics*, 46(3):227–246.

Malouche, I. e Bouani, F. (2018). A new adaptive partially decentralized pid controller for non-square discrete-time linear parameter varying systems. *International Journal of Control, Automation and Systems*, 16(4):1670–1680.

Mandava, R. K. e Vundavilli, P. R. (2020). An adaptive pid control algorithm for the two-legged robot walking on a slope. *Neural Computing and Applications*, 32(8):3407–3421.

Menhas, M. I., Fei, M., Wang, L. e Qian, L. (2012). Real/binary co-operative and co-evolving swarms based multivariable pid controller design of ball mill pulverizing system. *Energy Conversion and Management*, 54(1):67–80.

Menhas, M. I., Wang, L., Fei, M.-R. e Ma, C.-X. (2011). Coordinated controller tuning of a boiler turbine unit with new binary particle swarm optimization algorithm. *International Journal of Automation and Computing*, 8(2):185–192.

Mirjalili, S., Mirjalili, S. M. e Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61.

Mobayen, S. (2015). An adaptive chattering-free pid sliding mode control based on dynamic sliding manifolds for a class of uncertain nonlinear systems. *Nonlinear Dynamics*, 82(1):53–60.

Mohamed, Z., Khairudin, M., Husain, A. e Subudhi, B. (2016). Linear matrix inequality-based robust proportional derivative control of a two-link flexible manipulator. *Journal of Vibration and Control*, 22(5):1244–1256.

Mohd Basri, M. A., Husain, A. R. e Danapalasingam, K. A. (2015). Enhanced backstepping controller design with application to autonomous quadrotor unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems*, 79(2):295–321.

Molina, D., Poyatos, J., Ser, J. D., García, S., Hussain, A. e Herrera, F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. *Cognitive Computation*, 12(5):897–939.

Moness, M. e Moustafa, A. M. (2015). Tuning a digital multivariable controller for a lab-scale helicopter system via simulated annealing and evolutionary algorithms. *Transactions of the Institute of Measurement and Control*, 37(10):1254–1273.

Muhuri, P. K., Shukla, A. K. e Abraham, A. (2019). Industry 4.0: A bibliometric analysis and detailed overview. *Engineering Applications of Artificial Intelligence*, 78:218–235.

Muñoz, J. E., Cervantes, J., Esparza, R. e Rosas, G. (2007). Iron nanoparticles produced by high-energy ball milling. *Journal of Nanoparticle Research*, 9(5):945–950.

Nandong, J. (2015). Heuristic-based multi-scale control procedure of simultaneous multi-loop pid tuning for multivariable processes. *Journal of Process Control*, 35:101–112.

Nandy, S., Prabhu J, J., Nagarajan, V., Sha, O. P. et al. (2020). Pid-type controller for marine cycloidal propeller: a simulation study. *Journal of Marine Science and Technology*, 25(1):111–137.

Naranjo, J. E., Serradilla, F. e Nashashibi, F. (2020). Speed control optimization for autonomous vehicles with metaheuristics. *Electronics*, 9(4). 551.

Nelder, J. A. e Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Ogunnaike, B., Lemaire, J., Morari, M. e Ray, W. (1983). Advanced multivariable control of a pilot-plant distillation column. *AIChE Journal*, 29(4):632–640.

Ou, L.-L., Chen, J.-J., Zhang, D.-M., Zhang, L. e Zhang, W.-D. (2014). Distributed h∞ pid feedback for improving consensus performance of arbitrary-delayed multi-agent system. *International Journal of Automation and Computing*, 11(2):189–196.

Panda, R. C. e Subramaniam, S. (2009). Design of a pid controller from a predictive control algorithm. *Chemical Product and Process Modeling*, 4(1). 29.

Patel, V. K. e Savsani, V. J. (2015). Heat transfer search (hts): a novel optimization algorithm. *Information Sciences*, 324:217–246.

Peretz, Y. (2018). A randomized algorithm for optimal pid controllers. *Algorithms*, 11(6). 81.

Piccagli, S. e Visioli, A. (2009). An optimal feedforward control design for the set-point following of mimo processes. *Journal of Process Control*, 19(6):978–984.

Podlubny, I. (1994). Fractional-order systems and fractional-order controllers. *Institute of Experimental Physics, Slovak Academy of Sciences, Kosice*, 12(3):1–18.

Pongfai, J., Angeli, C., Shi, P., Su, X. e Assawinchaichote, W. (2021). Optimal pid controller autotuning design for mimo nonlinear systems based on the adaptive slp algorithm. *International Journal of Control, Automation and Systems*, 19(1):392–403.

Pradhan, J. K. e Ghosh, A. (2015). Multi-input and multi-output proportional-integral-derivative controller design via linear quadratic regulator-linear matrix inequality approach. *IET Control Theory & Applications*, 9(14):2140–2145.

Rajarathinam, K., Gomm, J. B., Yu, D.-L. e Abdelhadi, A. S. (2016). Pid controller tuning for a multivariable glass furnace process by genetic algorithm. *International Journal of Automation and Computing*, 13(1):64–72.

Ramezani, M. e Neitzert, T. (2012). Mechanical milling of aluminum powder using planetary ball milling process. *Journal of Achievements in Materials and Manufacturing Engineering*, 55(2):790–798.

Rao, R. V., Savsani, V. J. e Vakharia, D. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315.

Rao, S. S. (2019). *Engineering optimization: theory and practice*. John Wiley & Sons, New Jersey, United States.

Rashedi, E., Nezamabadi-Pour, H. e Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

Rechenberg, I. (1973). Evolutionsstrategie—optimierung technischer systeme nach prinzipien der biologischen information. Em *Simulationsmethoden in der Medizin und Biologie*, páginas 83—-114. Fromman Verlag, Freiburg, Germany.

Ross, B. J. (2019). *A Lamarckian evolution strategy for genetic algorithms*, páginas 1–16. CRC Press.

Ruiz-López, I., Rodríguez-Jimenes, G. e García-Alvarado, M. (2006). Robust mimo pid controllers tuning based on complex/real ratio of the characteristic matrix eigenvalues. *Chemical Engineering Science*, 61(13):4332–4340.

Sahin, M. E., Cam Taskiran, Z. G., Guler, H. e Hamamci, S. E. (2020). Application and modeling of a novel 4d memristive chaotic system for communication systems. *Circuits, Systems, and Signal Processing*, 39(7):3320–3349.

Sastry, S., Bodson, M. e Bartram, J. F. (1990). Adaptive control: Stability, convergence, and robustness.

Savran, A. (2013). A multivariable predictive fuzzy pid control system. *Applied Soft Computing*, 13(5):2658–2667.

Shah, P. e Agashe, S. (2016). Review of fractional pid controller. *Mechatronics*, 38:29–41.

Shah-Hosseini, H. (2009). The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-inspired computation*, 1(1-2):71–79.

Sharma, R., Gaur, P. e Mittal, A. (2015). Performance analysis of two-degree of freedom fractional order pid controllers for robotic manipulator with payload. *ISA Transactions*, 58:279–291.

Sharma, R., Rana, K. e Kumar, V. (2014). Performance analysis of fractional order fuzzy pid controllers applied to a robotic manipulator. *Expert Systems with Applications*, 41(9):4274–4289.

Shashidhar, V. e Padhan, D. G. (2019). A rubust fractional order pid controller for mimo power systems. Em *1st International Conference on Sustainable Energy and Future Electric Transportation (SeFet 2019)*, volume 87, Hyderabad, India. EDP Sciences. 01024.

Shinoda, S., Yubai, K., Yashiro, D. e Hirai, J. (2017). Multivariable controller design achieving diagonal dominance using frequency response data. *Electronics and Communications in Japan*, 100(10):12–23.

Silva, M. F., Machado, J. T. e Lopes, A. (2004). Fractional order control of a hexapod robot. *Nonlinear Dynamics*, 38(1):417–433.

Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons, New Jersey, United States.

Singh, B. e Mulvaney, S. J. (1994). Modeling and process control of twin-screw cooking food extruders. *Journal of Food engineering*, 23(4):403–428.

Sivadasan, J. e Willjuice Iruthayarajan, M. (2018). Tuning of nonlinear pid controller for trms using evolutionary computation methods. *Tehnički vjesnik*, 25(Supplement 1):105–111.

Slama, S., Errachdi, A. e Benrejeb, M. (2019). Neural adaptive pid and neural indirect adaptive control switch controller for nonlinear mimo systems. *Mathematical Problems in Engineering*, 2019. 7340392.

Soukkou, A., Belhour, M. e Leulmi, S. (2016). Review, design, optimization and stability analysis of fractional-order pid controller. *International Journal of Intelligent Systems and Applications*, 8(7). 73.

Štimac, G., Braut, S. e Žigulić, R. (2014). Comparative analysis of pso algorithms for pid controller tuning. *Chinese Journal of Mechanical Engineering*, 27(5):928–936.

Storn, R. e Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Sun, J., Lai, C.-H. e Wu, X.-J. (2016). *Particle swarm optimisation: classical and quantum perspectives*. CRC press, Boca Raton, United States.

Sun, Z., Sanada, K., Gao, B., Jin, J., Fu, J., Huang, L. e Wu, X. (2020). Improved decoupling control for a powershift automatic mechanical transmission employing a model-based pid parameter autotuning method. *Actuators*, 9. 54.

Taherkhorsandi, M., Mahmoodabadi, M., Talebipour, M. e Castillo-Villar, K. (2015). Pareto design of an adaptive robust hybrid of pid and sliding control for a biped robot via genetic algorithm optimization. *Nonlinear Dynamics*, 79(1):251–263.

Tao, C., Wan, J. e Ai, J. (2017). A nonlinear control approach for a hypersonic vehicle. *Aircraft Engineering and Aerospace Technology*, 89(2):320–329.

Tepljakov, A., Petlenkov, E. e Belikov, J. (2011). Fomcom: a matlab toolbox for fractional-order system identification and control. *International Journal of Microelectronics and computer science*, 2(2):51–62.

Valério, D. e Da Costa, J. S. (2012). *An introduction to fractional control*, volume 91. IET, Stevenage, United Kingdom.

Wald, S. E. (2002). The history of science and religion in the western tradition: An encyclopedia. *Technology and Culture*, 43(3):654–656.

Wang, G.-G., Deb, S. e Coelho, L. d. S. (2015). Elephant herding optimization. Em *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, páginas 1–5, Bali, Indonesia. IEEE.

Wang, N., Wang, J., Li, Z., Tang, X. e Hou, D. (2018). Fractional-order pid control strategy on hydraulic-loading system of typical electromechanical platform. *Sensors*, 18(9). 3024.

Wang, R., Tan, C., Xu, J., Wang, Z., Jin, J. e Man, Y. (2017). Pressure control for a hydraulic cylinder based on a self-tuning pid controller optimized by a hybrid optimization algorithm. *Algorithms*, 10(1). 19.

Whitley, D., Gordon, V. S. e Mathias, K. (1994). Lamarckian evolution, the baldwin effect and function optimization. Em *International Conference on Parallel Problem Solving from Nature*, páginas 5–15, Jerusalem, Israel. Springer.

Wolpert, D. H., Macready, W. G. et al. (1995). No free lunch theorems for search. Relatório técnico, Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Wood, R. e Berry, M. (1973). Terminal composition control of a binary distillation column. *Chemical Engineering Science*, 28(9):1707–1717.

Wu, S. (2015). State space predictive functional control optimization based new pid design for multivariable processes. *Chemometrics and Intelligent Laboratory Systems*, 143:16–27.

Wu, T.-Y., Jiang, Y.-Z., Su, Y.-Z. e Yeh, W.-C. (2020). Using simplified swarm optimization on multiloop fuzzy pid controller tuning design for flow and temperature control system. *Applied Sciences*, 10(23). 8472.

Xu, Q., Wang, L., Wang, N., Hei, X. e Zhao, L. (2014a). A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*, 29:1–12.

Xu, Q., Zhang, C., Zhang, L. e Wang, C. (2014b). Multiobjective optimization of pid controller of pmsm. *Journal of Control Science and Engineering*, 2014. 471609.

Yan, G.-W. e Hao, Z.-J. (2013). A novel optimization algorithm based on atmosphere clouds model. *International Journal of Computational Intelligence and Applications*, 12(01). 1350002.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. Em *International symposium on stochastic algorithms*, páginas 169–178, Sapporo, Japan. Springer.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. Em *Nature inspired cooperative strategies for optimization (NICSO 2010)*, páginas 65–74. Springer.

Yang, X.-S. e Deb, S. (2009). Cuckoo search via lévy flights. Em *2009 World congress on nature & biologically inspired computing (NaBIC)*, páginas 210–214, Coimbatore, India. IEEE.

Yu, W., Li, X. e Carmona, R. (2013). A novel pid tuning method for robot control. *Industrial Robot: An International Journal*, 40(6):574–582.

Yuan, Z.-z. (1985). On pid self-tuning regulators and its practical applications. *IFAC Proceedings Volumes*, 18(5):335–339.

Zakeri, H. e Ozgoli, S. (2014). A sum of squares approach to robust pi controller synthesis for a class of polynomial multi-input multi-output nonlinear systems. *Nonlinear Dynamics*, 76(2):1485–1495.

Zamani, M., Sadati, N. e Ghartemani, M. K. (2009). Design of an h∞ pid controller using particle swarm optimization. *International Journal of Control, Automation and Systems*, 7(2):273–280.

Zhao, S.-Z., Iruthayarajan, M. W., Baskar, S. e Suganthan, P. N. (2011). Multi-objective robust pid controller tuning using two lbests multi-objective particle swarm optimization. *Information Sciences*, 181(16):3323–3335.

Zhao, W., Zhang, Z. e Wang, L. (2020). Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence*, 87. 103300.

Ziegler, J. G. e Nichols (1942). Optimum settings for automatic controllers. *Transactions of the ASME*, 64(11):759–765.

Apêndice A

Trabalhos relacionados

Tabela A.1: Trabalhos com otimização para PID - Parte 1

Referência	Tipo de PID	Processo	Método de sintonia
Lengare et al. (2008)	APID	MIMO	Análise de resposta ao degrau monotônica e subamortecida
Boubakir et al. (2012)	APID	Robô com 2 graus de liberdade	Descida de gradiente
Taherkhorsandi et al. (2015)	APID	Robô bípede	GA
Kang et al. (2014)	APID	Sistema MIMO arbitrário	PSO e rede neural
Ishizuka e Kajiwara (2015)	APID	Modelo de vibração	SPSA
Mobayen (2015)	APID	Pêndulo invertido	SMC
Khan et al. (2016)	APID	Suspensão de carro	Descida de gradiente
Mahmoodabadi et al. (2017a)	APID	Sistema oscilador e robô	GA
Wang et al. (2017)	APID	Cilindro hidráulico	GA e PSO
Mahmoodabadi et al. (2017b)	APID	Controle de nível de líquido	GA
Abbasi e Naghavi (2017)	APID	Sistema de levitação magnética	XCS
Malouche e Bouani (2018)	APID	Robô móvel	DRGA
Slama et al. (2019)	APID	Sistema MIMO arbitrário	Rede neural
Liao e Zhao (2019)	APID	Helicóptero	Rede neural
Mandava e Vundavilli (2020)	APID	Robô bípede	MCIWO
Sun et al. (2020)	APID	Transmissão mecânica	Método de Nelder e Mead (1965)
Sharma et al. (2014)	FOFPID	Robô	CS
Silva et al. (2004)	FOPID	Robô hexápode	Método próprio
Djari et al. (2013)	FOPID	Sistema MIMO de ordem fracionária	PSO
Liu et al. (2015)	FOPID	Sistema fracionário oscilatório	INVLAP
Sharma et al. (2015)	FOPID	Robô	CS
Soukkou et al. (2016)	FOPID	Helicóptero	GA
Lakshmanaprabu et al. (2017)	FOPID	Sistema com 2 tanques	GA
Wang et al. (2018)	FOPID	Sistema hidráulico	Método de Åström e Hägglund (1995)
Shashidhar e Padhan (2019)	FOPID	Sistema de energia	CS
AbouOmar et al. (2019)	FOPID	Célula de combustível	Rede neural
Bhookya e Jatoth (2020b)	FOPID	Coluna de Wood e Berry (1973)	TLBO
Guha et al. (2020)	FOPID	Sistema de energia	GO
Ko e Wu (2008)	FPID	Gangorra	PSO
Savran (2013)	FPID	Sistema MIMO arbitrário	Levenberg-Marquardt
Khooban et al. (2013)	FPID	Robô móvel	TLBO
Gil et al. (2014)	FPID	Sistema com 3 tanques	Levenberg-Marquardt e rede neural
Wu et al. (2020)	FPID	Controle de fluxo e temperatura	PSO

Tabela A.2: Trabalhos com otimização para PID - Parte 2

Referência	**Tipo de PID**	**Processo**	**Método de sintonia**
Zamani et al. (2009)	PID H∞	Avião de combate	PSO
Ou et al. (2014)	PID H∞	Sistema MIMO arbitrário	LMI
Erol e Delibaşı (2018)	PID H∞	Sistema de suspensão	LMI
Li et al. (2019)	IMCPID	Sistema de vibração de asa diversa	PSO
Aboelhassan et al. (2020)	PID com MPC	Sistema com 3 tanques	MPC
Singh e Mulvaney (1994)	PID	Rosca de transporte de comida	DRGA
Chen e Cheng (1998)	PID	Avião	GA
Gilbert et al. (2003)	PID	Máquina de papel	FFRD
Ruiz-López et al. (2006)	PID	Coluna de destilação	Índice quadrático de Lyapunov
Chang (2006)	PID	Coluna de Wood e Berry (1973)	GA
Arruda et al. (2008)	PID	Craqueamento catalítico de fluido	GA
Panda e Subramaniam (2009)	PID	Sistema MIMO arbitrário	DMC
Piccagli e Visioli (2009)	PID	Fracionador de óleo pesado	BLT
Iruthayarajan e Baskar (2009)	PID	Coluna de Wood e Berry (1973)	CMAES e PSO
Chang et al. (2009)	PID	Reator químico	LQR
Iruthayarajan e Baskar (2010)	PID	Colunas de Wood e Berry (1973) e Ogunnaike et al. (1983)	CMAES
Menhas et al. (2011)	PID	Turbina de caldeira	PSO
Zhao et al. (2011)	PID	Caça a jato	PSO
Kondo e Ochi (2011)	PID	Caça a jato	LQR
Menhas et al. (2012)	PID	BMPS	PSO
Chen et al. (2013)	PID	Reator de polimerização	Método próprio
Jing e Cheng (2012)	PID	Rede neural	LMI
Yu et al. (2013)	PID	Robô	Método próprio
Chang e Chen (2014)	PID	Coluna de Wood e Berry (1973)	PSO
Liu et al. (2014)	PID	Robô	GA
Xu et al. (2014b)	PID	Motor síncrono	GA
Ahmad et al. (2014)	PID	Diversos sistemas MIMO	SPSA
Mohamed et al. (2016)	PID	Robô	LMI
Zakeri e Ozgoli (2014)	PID	Sistema MIMO não-linear	Soma de quadrados
Štimac et al. (2014)	PID	Disco magnético ativo	PSO
Dangor et al. (2014)	PID	Suspensão de veículo	DE, GA, e PSO
Jin e Liu (2014)	PID	Colunas de Wood e Berry (1973) e Ogunnaike et al. (1983)	Método próprio
Kiyak (2016)	PID	Caça a jato	PSO

Tabela A.3: Trabalhos com otimização para PID - Parte 3

Referência	Tipo de PID	Processo	Método de sintonia
Mohd Basri et al. (2015)	PID	Avião autônomo	PSO
Moness e Moustafa (2015)	PID	Helcóptero	SA
Luan et al. (2015)	PID	Coluna de destilação	IMC
Nandong (2015)	PID	Coluna de destilação	SML
Belhaj e Boubaker (2015)	PID	Sistema com 4 tanques	LMI
Pradhan e Ghosh (2015)	PID	Sistema com 4 tanques e caça a jato	LQR
Che Razali et al. (2015)	PID	Tratamento de água	SPM
Wu (2015)	PID	Coluna de destilação	SSPFC
Boyd et al. (2016)	PID	Coluna de Wood e Berry (1973)	LMI
Tao et al. (2017)	PID	Veículo hipersônico	GA
Rajarathinam et al. (2016)	PID	Forno	GA
Atacak e Kucuk (2017)	PID	Conversão de energia	PSO
Dwi et al. (2017)	PID	Foguete	DE
Shinoda et al. (2017)	PID	Aparato de tensão e velocidade	FFRD
Halim e Ismail (2017)	PID	Coluna de Wood e Berry (1973)	TPO
Peretz (2018)	PID	Veículo aéreo	Método próprio
Halim e Ismail (2019)	PID	Forno	TPO
Sivadasan e Willjuice Iruthayarajan (2018)	PID	Rotor duplo	DSA, GA, SBX, GSA, e PSO
Kassai et al. (2019)	PID	Refrigerador	Método de Ziegler e Nichols (1942)
Nandy et al. (2020)	PID	Hélice marinha cicloidal	Método próprio
Augusti Lindiya et al. (2019)	PID	Conversor de energia	LQR e GA
Altinoz et al. (2020)	PID	Hidrelétrica	Método de Ziegler e Nichols (1942), DE, e PSO
Sahin et al. (2020)	PID	Memristor	GA e FA
Karami et al. (2020)	PID	Robô	ACO
Dachang et al. (2020)	PID	Robô	Lógica Fuzzy e rede neural
Pongfai et al. (2021)	PID	Pêndulo invertido	SLP
Naranjo et al. (2020)	PID	Controle de velocidade	GA