UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO (EMC)
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

GUILHERME FERNANDES DOS SANTOS

Algoritmos de Inteligência Computacional Aplicados à Otimização de Sistemas de Controle em Acionamentos Elétricos

GOIÂNIA
2023

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[X] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Guilherme Fernandes dos Santos

3. Título do trabalho

“Algoritmos de Inteligência Computacional Aplicados à Otimização de Sistemas de Controle em Acionamentos Elétricos”

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Guilherme Fernandes Dos Santos**, **Usuário Externo**, em 06/04/2023, às 10:53, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 10/04/2023, às 15:59, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3652761** e o código CRC **7F0C388A**.

Referência: Processo nº 23070.009585/2023-08 SEI nº 3652761

Guilherme Fernandes dos Santos

Algoritmos de Inteligência Computacional Aplicados à Otimização de Sistemas de Controle em Acionamentos Elétricos

Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Engenharia Elétrica e de Computação, da Escola de Engenharia Elétrica, Mecânica e de Computação, da Universidade Federal de Goiás (UFG), como requisito para obtenção do Título de Mestre em Engenharia Elétrica e de Computação.

Área de Concentração: Engenharia de Computação.

Orientador: Prof. Dr. Gelson da Cruz Junior

Coorientador: Prof. Dr. Wander Gonçalves da Silva

GOIÂNIA

2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Santos, Guilherme Fernandes dos
Algoritmos de Inteligência Computacional Aplicados à Otimização de Sistemas de Controle em Acionamentos Elétricos [manuscrito] / Guilherme Fernandes dos Santos. - 2023.
CXXIX, 129 f.

Orientador: Prof. Gelson da Cruz Junior; co-orientador Wander Gonçalves da Silva.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2023.
Bibliografia.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Otimização. 2. Inteligência computacional . 3. Meta-heurísticas. 4. Configuração de algoritmos. 5. Acionamentos elétricos. I. Junior, Gelson da Cruz, orient. II. Título.

CDU 621.3

UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº 05 da sessão de Defesa de Dissertação de **Guilherme Fernandes dos Santos**, que confere o título de Mestre em **Engenharia Elétrica e de Computação**, na área de concentração em **Engenharia de Computação**.

Aos **vinte e nove dias do mês de março de dois mil e vinte e três**, a partir das **14h00min**. na sala Caryocar Brasiliensis da Escola de Engenharia Elétrica, Mecânica e de Computação da UFG, realizou-se a sessão pública de Defesa de Dissertação intitulada **"Algoritmos de Inteligência Computacional Aplicados à Otimização de Sistemas de Controle em Acionamentos Elétricos"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **Gelson da Cruz Junior - (EMC/UFG),** com a participação dos demais membros da Banca Examinadora: Professor Doutor **Wander Gonçalves da Silva - (EMC-UFG)** Coorientador, membro titular externo; Professor Doutor **Marco Antonio Assfalk de Oliveira - (EMC/UFG)** Membro Titular Externo; Professor Doutor **Alisson Assis Cardoso - (EMC/UFG)** Membro Titular Interno e Professor Doutor **VOLKER PICKERT - (UNIVERSIDADE DE NEWCASTLE - REINO UNIDO)** Membro Titular Externo. Durante a arguição os membros da banca **não fizeram** sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor **Gelson da Cruz Junior**, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos **vinte e nove dias do mês de março de dois mil e vinte e três**.

TÍTULO SUGERIDO PELA BANCA

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Volker Pickert**, **Usuário Externo**, em 30/03/2023, às 11:50, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Wander Gonçalves Da Silva**, **Professor do Magistério Superior**, em 30/03/2023, às 21:08, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marco Antonio Assfalk De Oliveira**, **Professor do Magistério Superior**, em 30/03/2023, às 22:47, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 31/03/2023, às 06:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Guilherme Fernandes Dos Santos**, **Usuário Externo**, em 31/03/2023, às 13:17, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 31/03/2023, às 18:47, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3634263** e o código CRC **DAB89C84**.

Referência: Processo nº 23070.009585/2023-08 SEI nº 3634263

Acknowledgements

Firstly, I would like to express my heartfelt gratitude to my mother, a true example of strength and courage. From the very beginning, she has been a fearless champion for our family, overcoming countless obstacles that have crossed our path. I can never truly express the depth of my appreciation for all that she has done for me, but I hope to make her proud in all that I do. I would also like to extend my gratitude to my entire family, with special mention to my father Valdir, my sister Giulia and my brother Henrique, who have been unwavering companions.

To Sâmya, for all the love and understanding throughout this journey. Your patience and encouragement were invaluable and motivated me to be a better person and to pursue my goals with confidence. Thank you, Sam, for being the guiding constellation that leads me to new horizons and for being the best partner I could ever ask for.

I am also extremely grateful for the wonderful friends that life has brought into my path. Despite the challenges of distance and time, our hearts have remained close. Thank you for your unwavering support, encouragement, comfort, and happiness. I would like to give special thanks to André, CJ, Henrique, Isaque, João, Josephy, Lulu, Madu, Najulia, Paulo Vitor, and Vitim.

I would like to express my deepest gratitude to Professors Gélson and Wander for their competent supervision of my master's work. I am grateful for the opportunity to work with you, for your company, time, patience, and the valuable teachings I received. Undoubtedly, you are excellent examples of researchers, professionals, and human beings that one day I hope to become. Thank you for the inspiration and encouragement to move forward.

Finally, I would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) for the financial support.

Power is nothing without control.
(Pirelli)

Resumo

Este trabalho apresenta o uso de diferentes métodos de inteligência computacional aplicados ao ajuste de um conjunto de controladores PI para acionamento de um motor CC com controle de velocidade e posição. Para controle de posição, são utilizadas três malhas de controle fechadas: corrente de armadura, velocidade e posição. Para controle de velocidade são utilizadas somente as malhas de corrente de armadura e velocidade. Em ambos os casos, as saídas dos controladores de corrente e velocidade são limitadas à tensão e a corrente nominal de armadura, respectivamente. Através disto, é possível a utilização de ganhos mais elevados para os controladores, o que faz com que o sistema responda mais rápido. No entanto, o fenômeno de *windup* pode surgir e, para evitá-lo, circuitos *anti-windup* também são utilizados e, portanto, o sistema torna-se não-linear. Devido a isto, um ajuste ótimo dos controladores se torna uma difícil tarefa. No intuito de explorar diferentes possibilidades, primeiramente, os problemas de controle de velocidade e posição são formulados de modo com que somente um objetivo seja minimizado. Dentro deste contexto de otimização mono-objetivo, os algoritmos PSO e SA são utilizados para realizar o ajuste dos parâmetros dos controladores e, a partir disto, é investigado a capacidade de cada um quando comparados entre si. Formulações multiobjetivo também são exploradas visando atender três objetivos simultaneamente. Nesta parte do trabalho, os algoritmos evolucionários multiobjetivo NSGA-II e SPEA2 são utilizados. Todos os algoritmos foram implementados no MATLAB e os modelos de acionamentos elétricos foram desenvolvidos no ambiente SIMULINK. Os resultados das simulações são apresentados mostrando que para a formulação mono-objetivo, tanto para o problema de controle de velocidade quanto para controle de posição, o algoritmo PSO possuiu um desempenho superior ao SA. Para a formulação multiobjetivo o algoritmo SPEA2 apresentou melhores características com respeito ao indicador de qualidade *Spread* no problema de controle de velocidade. Além disso, demonstrou superar o NSGA-II em relação ao indicador *Hypervolume* no problema de controle de posição. Foram realizadas séries de testes variando os valores dos principais parâmetros de cada algoritmo, sendo observado que não houve vantagem estatisticamente significativa na maioria dos casos. De uma forma geral, os resultados apresentados evidenciam a capacidade dos algoritmos de encontrar ajustes ótimos para os controladores, seja para o problema mono-objetivo ou com objetivos múltiplos e conflitantes.

Palavras-chave: Otimização, inteligência computacional, meta-heurísticas, algoritmos evolucionários multiobjetivo, configuração de algoritmos, acionamentos elétricos, controle de velocidade, controle de posição, motor CC.

Abstract

This work presents the use of different computational intelligence methods applied to the tuning of a set of PI controllers for a DC motor drive with speed and position control. For position control, three closed control loops are used: armature current, speed and position. For speed control, only the armature current and speed loops are considered. In both cases, the outputs of the PI armature current and speed regulators are limited to the rated armature voltage and current, respectively. Then, it is possible to use higher gains for the controllers, what makes the system to respond faster. However, the windup phenomenon can arise. To avoid it, anti-windup circuits are also used and therefore, the system becomes non-linear. Because of this, an optimum tuning of the controllers may become a difficult task. In order to explore different possibilities, firstly, the speed and position control problems are formulated so that only one objective is minimised. Within this single-objective optimisation context, the PSO and SA algorithms are used to tune the controller parameters, them, the capability of each one is investigated when compared to each other. Multi-objective formulations are also explored to address three objectives simultaneously. In this part of the work, the multi-objective evolutionary algorithms NSGA-II and SPEA2 are used. All algorithms were implemented in MATLAB and the electric drive models were developed in the SIMULINK environment. Simulation results are presented showing that for the single-objective formulation, for both, the for speed and position control problems, the PSO algorithm outperformed the SA. For the multi-objective formulation, the SPEA2 algorithm presented better characteristics with respect to the Spread quality indicator in the only, when compared to NSGA-II. Furthermore, it's shown to outperform the NSGA-II with respect to the Hypervolume indicator within the position control problem. A series of tests were carried out by varying the values of the main parameters setting for each algorithm. However, in most cases, no statistically significant advantage was observed. In general, the results presented demonstrate the ability of the algorithms to find optimal tuning for the controllers, either for the single-objective or the multiple and conflicting objectives problem.

Keywords: Optimisation, computational intelligence, metaheuristics, multi-objective evolutionary algorithms, algorithm configuration, electric drives, speed control, position control, DC motor.

List of Figures

List of Tables

List of symbols

I_a	*Armature current*
V_a	*Armature voltage*
$\mathbf{x}$	*Candidate solution*
c_1, c_2	*Cognitive and social coefficients*
α	*Cooling coefficient*
A_t	*External population*
$w(t)$	*Inertia weight*
T_0	*Initial temperature*
n_t	*Maximum number of iterations*
n	*Number of decision variables*
m	*Number of objectives*
Q	*Offspring population*
OS	*Overshoot*
PF	*Pareto Front*
PS	*Pareto Set*
P	*Population*
N_P	*Population size*

θ	*Position*
T_r	*Rise time*
T_s	*Settling time*
ω_m	*Speed*
E_{ss}	*Steady-state error*

List of abbreviations and acronyms

ANOVA	*Analysis of Variance*
HV	*Hypervolume*
IAE	*Integral Absolute Error*
MOEA	*Multi-Objective Evolutionary Algorithm*
MOP	*Multi-objective Problem*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
PSO	*Particle Swarm Optimisation*
PI	*Proportional-Integral Controller*
RNI	*Ratio of Non-dominated Individuals*
SA	*Simulated Annealing*
S	*Spread*
SPEA2	*Strength Pareto Evolutionary Algorithm 2*
WSM	*Weighted Sum Model*

Contents

CHAPTER 1

Introduction

In most areas of engineering, problems involving dynamic systems are very common. Most of them must present a desirable behaviour to produce useful application, thus demanding some action by using correct control strategies. Such systems are usually found in industry and are considered in many different applications such as printers, plotters, escalators, conveyor systems, manufacturing and film making processes, among many others. Depending on the control strategy, the performance of the controlled system may not meet the requirements which are necessary for a specific problem. Should non-linearities and/or parameters variations be present, the performance may be reduced, or the control performance may not no longer meet the requirements. Within this context, several control structures have been studied and improved over the decades.

Within the classical control theory, the traditional Proportional-Integral (PI) controllers are very common and largely used in linear control systems. In electric drive systems, they are commonly used for torque, speed, and position control for industrial application and, in most of the cases, must present a good speed holding capability against disturbance and/or parameter variation, for instance.

PI controllers are well known and used worldwide. However, there are applications where the system presents non-linearities such as saturation and/or parameter variation, for instance. In such case, the desired performance of the controlled system does not perform optimally in terms of disturbance rejection, noise attenuation and/or trajectory tracking. However, depending on the degree of the system to be controlled, tuning the controller may become a difficult task.

Some different classical strategies to tune the PI controller have already been proposed and discussed in the literature. One of the most commonly used methods, which is still widespread, was presented by Ziegler and Nichols (ZIEGLER; NICHOLS et al., 1942). Other well-known techniques include Chien, Hrones, and Reswick (CHIEN; HRONES; RESWICK, 1952), Cohen and Coon (COHEN; COON, 1953), and Internal Model Control (GARCIA; MORARI, 1982). Even though, the study of new strategies are still investigated in constant relevance in the academic and industrial areas (DUTTA; KUMAR; PANKAJ, 2014; SEN et al., 2015; CHIDAMBARAM; SAXENA, 2018). However, should the system presents high order transfer function or non-linearities, reaching an optimal tuning may become a difficult task. In such cases, an alternative tuning method must be considered such as optimisation methods supported by metaheuristics.

From the point of view of accuracy, a purely mathematical approach to the study of automatic control systems may be preferred by many engineers. However, depending on the system, the mathematics involved can be excessively difficult and not practical. On the other hand, metaheuristic optimisation algorithms, despite not always guaranteeing a global optimum, provide feasible solutions in a satisfactory length of time. In this way, such computational intelligence techniques are useful and have been consolidating in recent decades, becoming an area of growing interest (RODRÍGUEZ-MOLINA et al., 2020).

Within the field of computational intelligence, metaheuristics have been widely used to solve real-world optimisation problems. Such algorithms orchestrate an interaction between local improvement procedures (heuristics) and high-level strategies that aims to create processes capable of avoiding local minima and perform a robust search within the searching space (GENDREAU; POTVIN et al., 2010). Thus, heuristics are specialised in solving problems with domains while metaheuristics have a more general characteristics, which makes them commonly used to solve different problems where classical methods may not be feasible.

To apply a metaheuristic to a problem, is necessary to define a function from which it is possible to evaluate the quality of the generated solutions, known as *fitness* or *objective* function. Thus, within the context of optimisation, a problem can be formulated and classified according to their number of objectives; it can be *single-objective* for one objective or *multi-objective* (MOPs) for more than one. Regarding to applications, it is increasingly demanded to meet several specifications simultaneously, which tend to present conflicting necessities with each other (FASIH et al., 2018). In the particular scope of this work, different configurations of controller parameters imply different levels of satisfaction for each demand. Fortunately, tuning controllers with multiple objectives can be treated as a multi-objective mathematical programming problem, which allows the use of several multi-objective optimisation techniques to find a set of solutions that can be used to solve

a problem. Then, based on preferences, the designer can choose the solution that best fits the demand (RODRÍGUEZ-MOLINA et al., 2020).

The use of computational intelligence techniques in control engineering is defined by Ruano et al. (2014) as *intelligent control*. In general, some examples of single-objective algorithms can be cited: Genetic Algorithm (GA), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), Simulated Annealing (SA) and Memetic Algorithm (MA) (AGRAWAL et al., 2021). In the context of multi-objective algorithms Non-dominated Sorting Genetic Algorithm II (NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA2), Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) and Multi-Objective Particle Swarm Optimisation (MOPSO) have been studied by different researchers worldwide (TIAN et al., 2021).

Within the context of electric drives, applications with speed and/or position control find uncountable industry applications. However, due to the complexity of the control system, number of parameters to be adjusted and inherently present non-linearities, an optimal tuning of the controllers may become a difficult task. Furthermore, many different system restrictions applies and/or goals must be achieved, what may add more difficulties to the designer to find the best tuning for the controllers. A couple of publications on the speed and position control, with alternative control approach together with the use of stochastic search and optimisation techniques, can be found (SILVA; ACARNLEY; FINCH, 2001; SANTOS et al., 2021a; VILLARREAL-CERVANTES et al., 2017). Some problems have a single objective to be met (WONG; HOO; MOHYI, 2018; MANIKANDAN; ARULMOZHIYAL, 2014) while others, target a multi-objective one (SARDAHI; BOKER, 2018; SANPRASIT; ARTRIT, 2019).

This dissertation presents a comparative study of performance between a set of algorithms when used to find the best tuning of a set of PI controllers, for speed and position control of a separately excited DC motor drive. The problems are formulated in two ways, single-objective and multi-objective problem. The general and specific objectives are described in the following section.

1.1 Objectives

The main objective of this work is to use and compare different optimisation algorithms applied to a DC motor drive system. Two different scenarios for speed and position control of a separately excited DC motor drive system are explored. The first consists of a single-objective whereas the other one, a multi-objective formulation problem, which admits more objectives to be explored and satisfied at the same time.

It is important to highlight that, although the speed and position control problems are explored separately, they are similar. Nevertheless, they are treated as two distinct

problems. In the first one, the algorithms must tune the speed controller and, in the second one, the position controller.

More specifically, the objectives are as follows:

- Investigate the impact of different parameter settings for the PSO, SA, NSGA-II, and SPEA2 algorithms when applied to controller tuning problems in electric drives;
- Apply and compare the performance of two different metaheuristics (PSO and SA), for the single-objective formulation of the speed and position control of a DC motor drive system problems;
- Evaluate the performance of the implemented multi-objective metaheuristics (NSGA-II and SPEA2) for the speed and position control problems;
- Compare the multi-objective algorithms based on a set of three quality indicators;
- Validate the results through statistical tests;
- Evaluate how the performance of each metaheuristic varies when applied to different problems;
- Perform the analysis of the control parameters obtained for the controllers in the different scenarios.

1.2 Text Organisation

The dissertation is divided into six chapters where the first one is the Introduction. The second one presents a literature review on metaheuristics; the third one presents the problem characterisation; in the fourth and fifth chapters, results are presented. The sixth chapter presents a general conclusion of this work.

The chapters are described next:

- Chapter 2 - covers basic concepts of optimisation, metaheuristics, multi-objective problems, multi-objective algorithms and quality indicators;
- Chapter 3 - presents a description of the system, in which the DC motor is modelled as a block diagram, and the speed and position control problems are formulated. The need and use of anti-windup circuits are also briefly explored. The objective functions are presented and the statistical tests used to compare the performance of the algorithms are described;
- Chapter 4 - presents the results obtained using the single-objective PSO and SA algorithms;

- Chapter 5 - shows the results referred to the multi-objective formulation, obtained by the NSGA-II and SPEA2 algorithms;
- Chapter 6 - presents the final remarks and future work.

CHAPTER 2

Optimisation and Computational Intelligence

This chapter presents theoretical concepts that gives support to the studies presented in this work, on the optimisation and methods by using computational intelligence. In Section 2.1 general concepts associated with optimisation problems are introduced. Sections 2.2 and 2.3 presents the descriptions of heuristic and metaheuristic techniques as well as the single-objective PSO and SA algorithms. In Section 2.4 the basic concepts of multi-objective optimisation are presented and, in the Section 2.5, the general aspects of evolutionary algorithms are discussed along with details on the multi-objective ones (NSGA-II and SPEA2). Finally, in Section 2.6, some quality indicators used for performance evaluation within the context of multi-objective optimisation, are presented.

2.1 Optimisation Problems

An optimisation problem consists of finding extreme values (minimum or maximum) for a fitness function. For this purpose, a set of decision variables $\mathbf{x} = (x_1, x_2, ..., x_n)$ have to be considered and represent a possible solution for a problem. In this context, the fitness function can be: *i*)*continuous* when the variables assume real values; *ii*) *discrete* when the variables assume integer or discrete values; and *iii*) *mixed* when it contains continuous and integer variables (BARBOSA, 2017).

A problem is said to be an optimisation one because the solutions found are the best possible among all the solutions in the search space, being then called *optimal solutions* (BECCENERI, 2008). A mathematical definition for optimisation problems is as follows.

Given the function $f : \Omega \rightarrow \mathbb{R}$, where Ω is a set of real numbers, a vector $\mathbf{X}_0 \in \Omega$ is optimal if:

- $f(\mathbf{X}_0) \leqslant f(\mathbf{X}), \forall \mathbf{X} \in \Omega$ (minimisation);
- $f(\mathbf{X}_0) \geqslant f(\mathbf{X}), \forall \mathbf{X} \in \Omega$ (maximisation).

Most optimisation problems in the real world are very difficult to solve, requiring long time to reach to an end. Because of this, the use of computational resources is essential. The algorithms to solve this type of problem can be deterministic or stochastic. The deterministic ones are those that, for a given input, the output always will be the same. Some examples are the simplex method, conjugate gradient and Newton's method. Stochastic algorithms, as described in the next sections, make decisions based on probabilities and used random-valued parameters subject to statistical perturbations, and this means that the final result may vary for the same input (which is why are also known as *non-deterministic*) (BARBOSA, 2017).

2.2 Heuristics

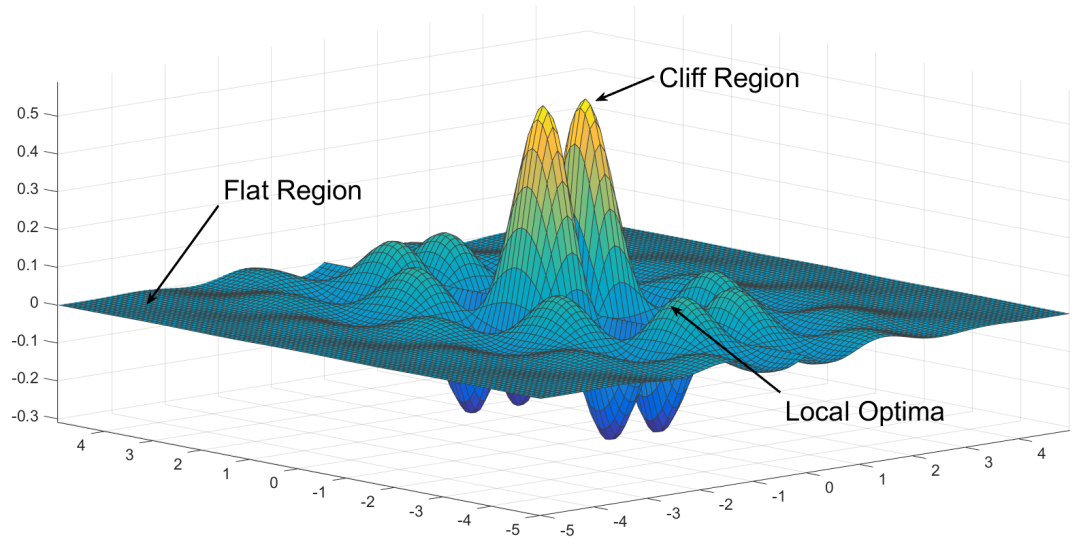
Heuristic methods for optimisation are developed to quickly solve complex and large-scale problems, for which exact methods would require impractical computational time. Such methods use some knowledge about the problem (or about some set of solutions) to take shortcuts and fast find high-quality solutions, which are not necessarily the global optimum. Thus, heuristics produce a balance between computational effort and quality solutions. To achieve this balance, it is necessary to have some prior knowledge about the characteristics of the problem, and thus explore the search space more efficiently. And it is this knowledge that guides the algorithm, reducing the investigated search space (MELO, 2009).

Figure 2.1 exemplifies a search surface of a continuous problem. Assuming it is a maximisation problem, three important characteristics can be identified:

- Flat region: regions in which an algorithm may face difficulties in identifying the best direction to follow. In this type of location, the algorithm performs a blind search;
- Local optima: when the algorithm identifies one or more points that have higher quality results, the search starts to be carried out in the neighbourhood of these points. There are practical problems in which local optima are satisfactory;
- Cliff region: steep local optima. In contrast to the typical local optima, where minor adjustments to surface coordinates lead to commensurate changes in fitness

function values, cliff regions demonstrate abrupt shifts in function values with even slight changes in the coordinates. Consequently, navigating such regions efficiently is essential, as algorithms may become trapped in peaks, which may be unsuitable local optima.

Figure 2.1 – Search space of a continuous problem: *i*) Flat region; *ii*) Local optima and *iii*) Cliff region.



Reference: author

Within the context of flat, local optima and cliff regions, it is evident that a heuristic restricting the search space, even in a probabilistic way, can prevent the algorithm from reaching the cliffs and find the global optimum. Furthermore, algorithms that perform a local analysis of the surface behaviour can drive solutions towards the sub-optimal regions. This effect becomes more evident if the search space is large and/or the number of flat, local optima and cliff is greater. As heuristics and other local search techniques are often applied to find the optimum of a neighbourhood, they become unsuitable for use in global optimisation problems (MELO, 2009).

Over the past decades, a wide variety of research has been carried out in the development of simple and generic algorithms (AGRAWAL et al., 2021), seeking to solve complex problems for which there are no adequate heuristics and even local search algorithms are unable to find satisfactory solutions. Among the main techniques that have been developed to overcome this challenge are the metaheuristics, described in Section 2.3.

2.3 Metaheuristics

The term *metaheuristic* was coined by Glover (1986). It refers to a general algorithmic search framework applied to optimisation problems for which there is no exact or satisfactory heuristic method capable of solving them. Metaheuristics have the ability to find feasible solutions for problems of realistic size in reasonable computation time (MAASHI, 2014).

The advantages of employing metaheuristic algorithms are derived from their property of being derivation-free, in addition to their simplicity, flexibility, and ability to avoid local optima (MIRJALILI; MIRJALILI; LEWIS, 2014). Metaheuristics exhibit stochastic behaviour, in contrast to deterministic algorithms, and commence the optimisation process by generating random solutions. A crucial characteristic of these algorithms is their *problem-independent* classification, allowing for easy customisation according to the problem domain, and demonstrating remarkable aptitude in circumventing premature convergence. They are successfully applied to several engineering and science problems, for example in electrical engineering (to find optimal solutions for power generation), industrial fields (scheduling jobs, transportation, vehicle routing problem and facility location problem), civil engineering (to design the bridges and buildings), communication (radar design and networking) and data mining (classification, prediction, clustering and system modeling) (AGRAWAL et al., 2021).

To make the understanding on the difference between heuristics and metaheuristics clearer, one can say that heuristic methods are usually efficient to find local optima. Then, cannot be applied globally due to dependence on information from the problem domain to restrict the search space. On the other hand, metaheuristics are characterised as high-level techniques, applied to global optimisation problems with multi-modal objective functions, for which there is a large number of high-quality solutions around the neighbour ones but only a few or a single one is the global optimum (BARA'A et al., 2021). To exemplify, it can be said that a genetic algorithm is a metaheuristic, while the Ziegler-Nichols method is a heuristic.

2.3.1 Classification of Metaheuristic Algorithms

In the literature, there are different perspectives on the classification of metaheuristics. Agrawal et al. (2021) classifies into the following two main categories:

- Single-solution based algorithms - These techniques start the optimisation process with a single solution and update it throughout the iterations. Such algorithms may suffer from two limitations. Firstly, the algorithm may become trapped in local optima, resulting in suboptimal solutions. Secondly, the search space may not be

explored extensively, limiting the algorithm's ability to identify optimal solutions in complex problems;

- Population-based algorithms - Initially, these algorithms generate a population of solutions before starting the optimisation process. The population of possible solutions is updated along the generations/iterations. The algorithms are capable to avoid local minima since multiple solutions assist each other, providing a greater exploration capability of the search space. Furthermore, they have the capability to jump towards the promising area of the search space.

Researchers have given special attention to metaheuristic algorithms because of their characteristics. Several algorithms have been emerged and solved different types of problems. Based on their behaviour, the metaheuristic algorithms can be divided into three main categories (AGRAWAL et al., 2021):

- Evolution-based algorithms - Are inspired by natural evolution and starts the process with a randomly generated population of possible solutions to the problem. In this sort of algorithms, the best solutions are put together to create new individuals. The new individuals are formed by using a crossover function and mutation is applied. The most popular algorithm in this category is the genetic algorithm. There are other algorithms such as evolution strategy, genetic programming and differential evolution, for instance;
- Swarm intelligence-based algorithms - These algorithms are inspired by the social behaviours of insects, animals, fishes or birds. Like evolution-based algorithms, they also use the concept of population. Among the most popular ones are the Particle Swarm Optimisation and Ant Colony Optimisation;
- Physics-based algorithms - These are inspired by the rules of physics in the universe. Simulated Annealing and Harmony Search are known examples.

2.3.2 Criteria for Algorithm Selection

In terms of the wide variety of philosophies and characteristics of these algorithms, the criteria established for choosing those that will be used in this work were the following: the first algorithm is the Particle Swarm Optimisation (PSO), which according to Hussain et al. (2019), between 1983 and 2016, it was the most prevalent algorithm, appearing in 23.16% of all metaheuristic publications. The distinction between the PSO and the other methods is significant, being the most adopted due to its relative simplicity and effectiveness in several scientific fields. The second algorithm chosen was Simulated Annealing (SA), which appeared in only 2.13% of metaheuristic publications (HUSSAIN et al., 2019),

placing it as the tenth most commonly used algorithm. Nevertheless, SA was selected due to its distinctive characteristics, as it is the most frequently used algorithm featuring a single-solution mechanism and based on physical principles.

Therefore, the PSO algorithm represents the group of metaheuristics based on population, as well as swarm intelligence, and the SA algorithm, the group that uses a single-solution and is physically inspired. The class of genetic algorithms, for reasons that will be described in Section 2.5, are used in the multi-objective formulation. It is evident that there is a significant number of metaheuristics, and the application of all of them is not feasible, therefore, the algorithms used in this work were chosen due to their popularity and because they present distinct aspects in terms of behaviour.

2.3.3 Particle Swarm optimisation (PSO)

The Particle Swarm optimisation algorithm was developed by Kennedy and Eberhart (1995) for solving non-linear problems. It has been thought to mirror the social behaviour of a bird flock aiming to discover patterns that guide the ability of birds to fly synchronously and, suddenly, change direction with regrouping in an optimal formation. With this aim, the concept has evolved to a simple and efficient optimisation algorithm (SANTOS et al., 2021b).

Within the PSO algorithm, each individual is referred to as a particle that flies within a multi-dimensional search space. The particle's position changes according to the tendency of the individuals to emulate the success of each other. Then, the changing of a particle's location within the swarm is influenced by the knowledge and experience of its neighbours. Therefore, the global or collective success of the swarm lies on finding an optimal region within a searching space where each particle represents a potential solution to the problem.

Be $\mathbf{x}_i(t)$ the position vector of the particle i within the search space at a discrete time step t. The position of each particle is adjusted by adding a velocity vector $\mathbf{v}_i(t)$ to the particle's current position, i.e.,

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \tag{2.1}$$

with $\mathbf{x}_i(0) \sim U(\mathbf{x}_{min}, \mathbf{x}_{max})$.

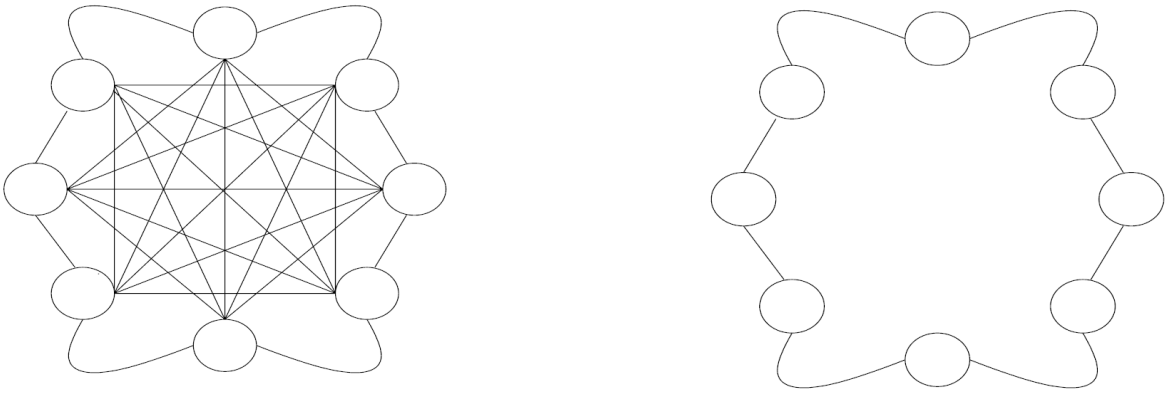
The concept of going after an optimum position is given by the changing of the velocity vector of each particle in time. It means that each particle changes its position according to the position of a better located one. The velocity vector drives the optimisation process.

There are several ways to select which individual is going to influence others at each iteration or time step. However, there are two most used ones, which topologies are

shown in Figure 2.2. Within the ring shape topology, as shown in Figure 2.2b, each particle has information only on its two closest neighbours. Different from the ring one, within the star topology, as shown in Figure 2.2a, every single individual can correspond to one another.

Figure 2.2 – Example social network structures for PSO.

(a) Star. (b) Ring.



Reference: (ENGELBRECHT, 2007)

While many studies have been done using different topologies, there is no outright best topology for all problems. In general, the fully connected structures perform best for multi-modal problems, while the less connected structures perform better on uni-modal (PEER; BERGH; ENGELBRECHT, 2003). Therefore, due to the nature of the problem, the star topology is the most suitable for this work, in which there is no prior knowledge of the characteristics of the evaluation criteria.

The velocity of particle i is computed as:

$$v_{ij}(t+1) = w(t)v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{2.2}$$

Where:

- $w(t)$ - Inertia weight;
- $v_{ij}(t)$ - Velocity of particle i in dimension j $(j = 1, 2, ..., n)$ at time step t;
- $x_{ij}(t)$ - Position of particle i in dimension j at time step t;
- $y_{ij}(t)$ - The personal best position in dimension j, associated with particle i is the best position the particle has visited since the first time step;
- $\hat{y}_j(t)$ - The global best position in dimension j at time step t;
- c_1 - Cognitive component;

- c_2 - Social component;
- $r_{1j}(t), r_{2j}(t)$ - Random values within the range [0, 1], sampled from a uniform distribution.

The velocity calculation consists of three terms (ENGELBRECHT, 2007):

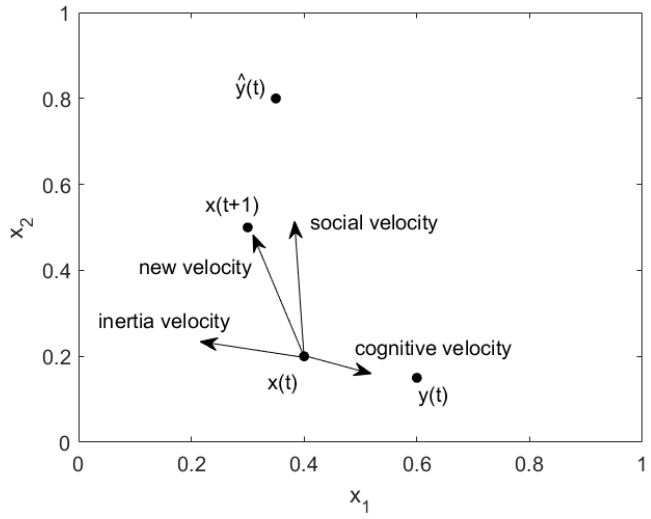
- The previous velocity, $w(t)\mathbf{v}_i(t)$, which serves as a memory of the previous flight direction. This memory term can be seen as a momentum, which prevents the particle from drastically changing direction, and to bias towards the current direction;
- The cognitive component, $c_1\mathbf{r}_1(\mathbf{y}_i - \mathbf{x}_i)$, which quantifies the performance of particle i relative to its past performances. In other words, the cognitive component resembles the individual's memory of its own best position within the swarm. The effect of this term is that the particles are drawn back to their own best positions, resembling the tendency of individuals to return to situations or places that most satisfied them in the past;
- The social component, $c_2\mathbf{r}_2(\hat{\mathbf{y}} - \mathbf{x}_i)$, which quantifies the performance of particle i relative to a group of particles, or neighbours. Conceptually, the social component resembles a group norm or standard that individuals seek to attain. The effect of the social component is to drive each particle is towards the best position found by the particle's neighbours.

The effect of the velocity equation can easily be illustrated in a two-dimensional vector space. Figure 2.3a illustrates the state of the swarm at time t. The new position, $\mathbf{x}(t+1)$, moves closer towards the global best $\hat{\mathbf{y}}(t)$. For time step $t+1$, as illustrated in Figure 2.3b, the particle's own best position does not change. The figure shows how the three components contribute to the particle's own moving towards the global best one. The cumulative effect of all the position updates of a particle is that each particle converges to a point on the line that connects the global best position and the self-best best position of the particle (ENGELBRECHT, 2007).

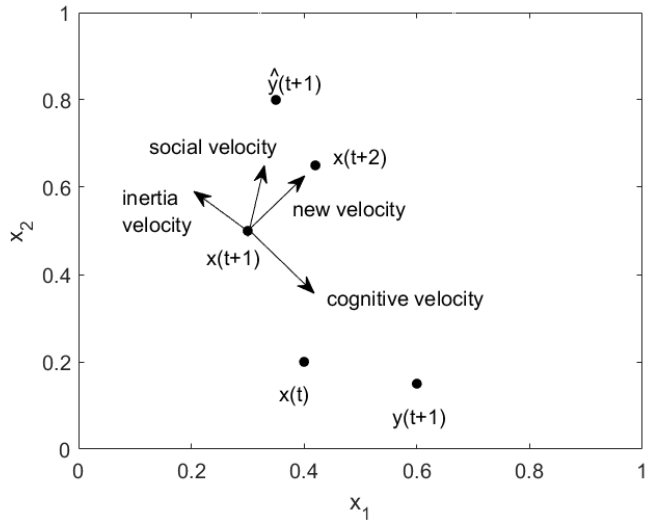
The constants c_1 and c_2 are also referred to as trust parameters, where c_1 expresses how much confidence a particle has on itself, while c_2 expresses how much confidence a particle has on its neighbours (ENGELBRECHT, 2007). For the case $c_1 >> c_2$, particles may find difficulties to converging to close positions, while otherwise, whereas otherwise, $c_1 << c_2$, particles may quickly fall into a local minimum and fail to move out. The usual approach is to make both coefficients to have the same values, thus giving equal importance to one's own knowledge and to collective one (EBERHART; SHI; KENNEDY, 2001).

Figure 2.3 – Geometrical illustration of velocity and position updates for a single two-dimensional particle.

(a) Time step t.



(b) Time step $t + 1$.



Reference: author

The inertia weight, $w(t)$, was introduced by Shi and Eberhart (1998) as a mechanism to control the exploration and exploitation abilities of the swarm, by controlling the contribution of the particle's previous velocity into the current one, weighing the impact of $v_i(t)$ over $v_i(t+1)$. Higher values of $w(t)$ make the exploitation easier by enhancing diversity. On the other hand, lower values gives room to a more refined local exploration. However, it is important to highlight the important of relationship between the values of $w(t)$ and the acceleration coefficients (c_1 and c_2), since Bergh and Engelbrecht (2006) showed that:

$$w(t) > \frac{1}{2}(c_1 + c_2) - 1 \tag{2.3}$$

guarantees convergent particle trajectories. If this condition is not satisfied, divergent or cyclic behaviour may occur.

Two main approaches to dynamically vary the inertia weight can be grouped into the following categories:

- Random adjustments: From Peng, Chen and Eberhart (2000) and defined as:

$$w(t) = c_1 r_1(t) + c_2 r_2(t) \tag{2.4}$$

- Linear decreasing: as suggested in Ratnaweera (2002), the linear decreasing starts with a high inertia weight (usually 0.9) and is linearly reduced up to 0.4, according to Equation 2.5.

$$w(t) = (w(0) - w(n_t))\frac{(n_t - t)}{n_t} + w(n_t) \tag{2.5}$$

  Where:

  - n_t - The maximum number of iterations;
  - $w(0)$ - Initial inertia weight value;
  - $w(n_t)$ - Final inertia weight value.

The pseudo code of the PSO algorithm is presented as follows:

Algorithm 1 PSO - Minimisation Problem

```
function PSO()
    Initialise population
    Evaluate population
    for t=1: Maximum number of iterations do
        for i=1: Population size do
            Update the velocity, v_i(t)
            Update the position, x_i(t)
            Evaluate solution
            if x_ij(t) < y_ij(t) then
                y_ij(t) ← x_ij(t)
            end if
            if x_ij(t) < ŷ_j(t) then
                ŷ_j(t) ← x_ij(t)
            end if
            Update inertia weight
        end for
        t=t+1
    end for
    return ŷ
end function
```

2.3.4 Simulated Annealing (SA)

Simulated annealing (SA) is a search algorithm proposed by Kirkpatrick, Jr and Vecchi (1983), and it is considered the first metaheuristic approach to use an explicit method that accepts worse solutions to escape from local optima. Initially, SA was used to tackle combinatorial optimisation problems (often within the discrete problem domain). It was later expanded to include continuous time problems (MAASHI, 2014). The concept of SA is based on the Metropolis algorithm for statistical mechanics developed by Metropolis et al. (1953). The Metropolis algorithm is a model for simulating the physical annealing process with solid materials like metals and glass. These materials are placed in a heat bath under a high temperature and then gradually cooled down according to an appropriate cooling schedule until they reach a thermal equilibrium state (HENDERSON; JACOBSON; JOHNSON, 2003).

The mathematical basics underlying annealing are derived from Boltzmann's distribution, which is defined as:

$$P(i) = \frac{1}{N_0} \exp\left(-\frac{E(i)}{kT}\right) \tag{2.6}$$

Initially, a sufficiently large thermodynamic system is considered. Within this context, the system can admit i possible states where each one has an associated energy level, $E(i)$. Then, $P(i)$ represents the probability of the system reach an i energy state. Based on such theory, it is assumed that, when the arrangement of the atoms is stable, the probability of the system's energy to be E is proportional to $\exp(-E/kT)$. Consequently, the probability of a system's energy to be $(E + dE)$ can be determined according to Equation 2.7 (SANTOS et al., 2021a).

$$prob(E + dE) = prob(E) \exp\left(-\frac{dE}{kT}\right) \tag{2.7}$$

Where:

$$dE = E(i + 1) - E(i) \tag{2.8}$$

It means that as the system's temperature cools down according to a cooling coefficient, α, the probability of its energy state to change is reduced to the point which represents the minimum energy state.

The SA algorithm starts with high-temperature values and an initial solution, $\mathbf{x}$, randomly generated. During the optimisation process, the temperature is slowly decreased and at each iteration, a new solution, $\mathbf{x}'$, is generated in the neighbourhood of the previous one. Next, the difference between the two solutions is computed as $\Delta = f(\mathbf{x}') - f(\mathbf{x})$. If Δ is less than zero, means that the solution $\mathbf{x}'$ is better than $\mathbf{x}$ (in a minimisation problem),

so the current solution is changed. However, if $\Delta > 0$, a real number $r \in [0, 1]$ is randomly generated, and if $r < \exp(-\Delta/T)$, then the current solution $\mathbf{x}$ is also replaced by $\mathbf{x}'$. After this process, an iteration is passed, the system's temperature is reduced, and the search continues until the stop criteria are satisfied.

It should be noted that the larger the distance between the new solution compared to the current one in the objective space, the smaller the acceptance probability. It is also applied to the temperature, i.e., the lower the temperature the smaller the probability of the new solution to be accepted when compared to the current one. Together, these characteristics, assure that local minima are avoided.

$$\text{Acceptance Probability} = \begin{cases} \exp\left(-\dfrac{\Delta}{T}\right) & \text{, if } f(\mathbf{x}') > f(\mathbf{x}) \\ 1 & \text{, if } f(\mathbf{x}') < f(\mathbf{x}) \end{cases} \tag{2.9}$$

Another important parameter, N, commonly used, is related to the number of tested solutions within each temperature level. That is, given a fixed temperature T, there are several energetic possibilities within, i.e, different states that need to be explored. Normally, in literature problems, this value is adjusted according to the complexity of the problem. In this work, different values for N will be considered to verify their impact on the convergence of the algorithm.

A pseudo code of SA algorithm is shown as follows:

Algorithm 2 SA - Minimisation Problem

```
1: function SA()
2:     Initialize parameters
3:     x ← Initial Solution
4:     while stop criteria is not reached do
5:         for i=1: N do
6:             Generate a neighbour x' within x neighbourhood
7:             Δ ← f(x') − f(x)
8:             if Δ < 0 then
9:                 x ← x'
10:            else if random[0, 1] < exp(−Δ/t) then
11:                x ← x'
12:            end if
13:        end for
14:        T ← αT
15:    end while
16:    return x
17: end function
```

2.4 Basic Concepts of Multi-Objective Optimisation

A multi-objective optimisation problem (MOP) can be defined as:

$$\begin{aligned} &\min \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x})) \\ &\text{subject to: } \mathbf{x} \in \Omega \end{aligned} \tag{2.10}$$

where $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$ is a candidate solution, $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ comprises all m conflicting objectives, and $\Omega \subset \mathbb{R}^n$ is the search space, such as $\Omega = \{\mathbf{x} \in \mathbb{R}^n | h_i(\mathbf{x}) \leq 0; \forall i = 1, ..., n_h\}$, where its image $\mathbf{F}(\Omega)$ defines the feasible set, and $\mathbf{h}$ represents a set of n_h constraints (LOPEZ, 2017).

Within multi-objective optimisation, two solutions can be evaluated using the Pareto Dominance concept. A solution $\mathbf{x}$ is said to dominate another solution $\mathbf{y}$ (denoted by $\mathbf{F}(\mathbf{x}) \prec \mathbf{F}(\mathbf{y})$) if and only if, $\mathbf{x}$ is better or equal than $\mathbf{y}$ in all objectives and has at least one of the objectives where $\mathbf{x}$ is better than $\mathbf{y}$. In mathematical terms can be described as: $f_k(\mathbf{x}) \leqslant f_k(\mathbf{x}') \forall k \in \{1, ..., m\} \wedge \exists k | f_k(\mathbf{x}) \neq f_k(\mathbf{x}')$ (COELLO et al., 2007a).

Therefore, a solution $\mathbf{x}^*$ is *Pareto Optimal* for 2.11 if there is no other solution $\mathbf{x}$ such that $\mathbf{F}(\mathbf{x}) \prec \mathbf{F}(x^*)$, and the set of all Pareto Optimal solutions is called Pareto Optimal Set (LOPEZ, 2017), defined as:

$$PS = \{\mathbf{x}^* \in \Omega | \mathbf{x} \in \Omega : \mathbf{F}(\mathbf{x}) \prec \mathbf{F}(\mathbf{x}^*)\} \tag{2.11}$$

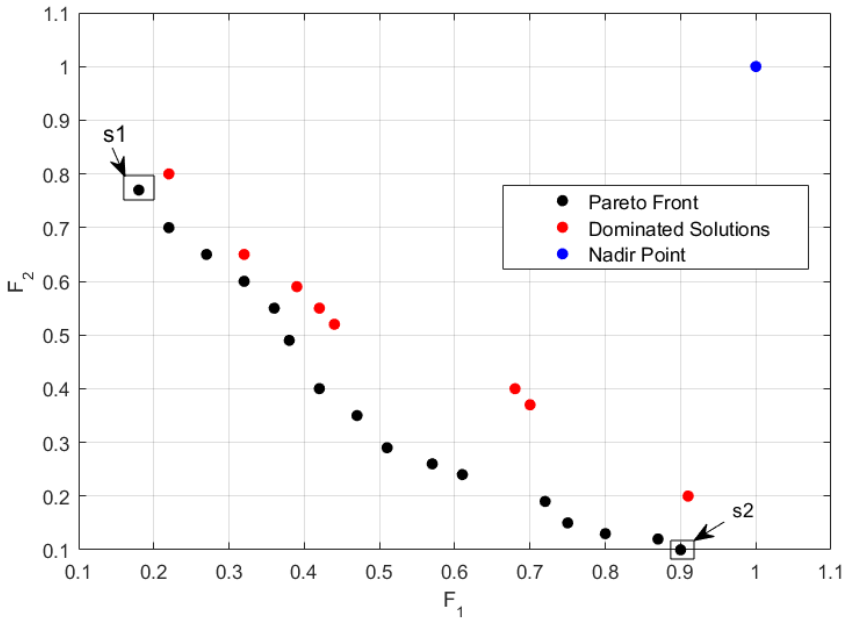
Another important concept is the *Pareto Front*, which is defined as the image of the Pareto Optimal Set, is:

$$PF = \{\mathbf{F}(\mathbf{x}^*) | \mathbf{x}^* \in PS\} \tag{2.12}$$

Figure 2.4 presents a two-objective minimisation problem, in which the Pareto Front (in black) and a set of dominated solutions (in red) are shown. Two non-dominated solutions (s_1 and s_2) are highlighted, and both belong to the Pareto Front. Solution s_1 prioritises objective 2 while s_2 objective 1. There are 14 other solutions that are also non-dominated. Thus, if only one solution has to be selected, it is necessary that a decision maker perform a trade-off between the 16 solutions found.

Represented in blue in Figure 2.4, the *nadir point* represents the worse possible solution for the problem. The nadir point is important since it is used as the reference point by some quality indicators, enabling to compare different algorithms performance on solving MOPs (CARVALHO, 2022).

Figure 2.4 – Two objective Pareto Front minimisation example.



Reference: author

2.5 Multi-Objective Evolutionary Algorithms (MOEAs)

Evolutionary Algorithms (EAs), within the context of multi-objective optimisation, are widely disseminated and used successfully in real-world applications (VACHHANI; DABHI; PRAJAPATI, 2015). This is a group of bio-inspired metaheuristics, which refers to systems that use computational models of evolutionary processes, such as natural selection, survival of the fittest and reproduction, as the fundamental components to perform optimisation procedures.

The different ways in which the EA components are implemented result in different evolutionary computation paradigms: genetic algorithms (GAs), genetic programming (GP), evolutionary programming (EP), differential evolution (DE) and co-evolution (CoE).

These algorithms are constituted by a population of candidates (called individuals or chromosomes) for solutions to an optimisation problem. To each individual a *fitness value* is assigned, which numerically represents the quality of the solution. Analogous to the concept of iteration in other algorithms, in EAs the term *generations* also can be used, in which at each pass, the population is modified through some operators, which aim to obtain better solutions.

The main operators in genetic algorithms are: *selection*, *crossover* and *mutation*. The selection operator selects solutions from the population while the crossover operator performs the combination of two individuals, generating new possible set of solutions. These new solutions can suffer the influence of the mutation operator, which performs changes in the codified structure of the decision variables, also known as *genes*. From the

application of these operators, there are new solutions and those with the best evaluation are maintained for the next generation (SILVA et al., 2014).

The Multi-Objective Evolutionary Algorithms (MOEAs) are extensions of EAs for multi-objective problems. MOEAs can be classified into three main categories that which define the type of adopted approach, based on Pareto dominance, preferences and methods that use aggregation functions (LI et al., 2015). This work is restricted to methods based on Pareto dominance, which, in a simplified way, use the concept of *fronts* to select and order the best solutions.

2.5.1 Criteria for Algorithm Selection

The multi-objective algorithms chosen to be implemented and studied in this work were NSGA-II and SPEA2. The first reason for choosing these algorithms is the fact that both are based on evolutionary principles. The evolutionary behaviour is desirable because the other algorithms considered in this work already encompassed groups of metaheuristics based on swarm intelligence and physics, as discussed in Section 2.3. The second motivation is associated with the literature review, in which according to Rodríguez-Molina et al. (2020), evolutionary algorithms are the mostly used in controller's tuning problems, being present in 51% of related works. The most commonly used is the NSGA-II algorithm, followed by SPEA2 (RODRÍGUEZ-MOLINA et al., 2020). In the following subsections, the particular characteristics of each selected algorithm are described.

2.5.2 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

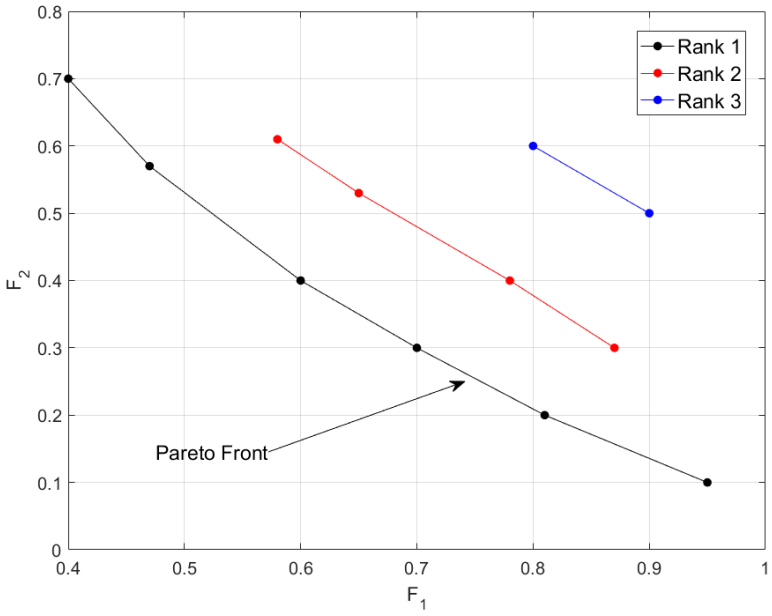
NSGA-II is an improved version of the non-dominated sorting genetic algorithm (NSGA) (SRINIVAS; DEB, 1994), which has been criticised by researchers due to its limitations such as the absence of elitism, the need to define sharing parameters for diversity preservation, and its high computational complexity. On the other hand, the NSGA-II, proposed by Deb et al. (2002), is considered one of the most efficient MOEAs and exhibits the property of elitism and does not need any sharing parameter (VERMA; PANT; SNASEL, 2021).

The philosophy of NSGA-II is based on four main principles: Non-Dominated Sorting, Elite Preserving Operator, Crowding Distance and Truncation Operator (VERMA; PANT; SNASEL, 2021). These are briefly described below:

- *Non-dominated sorting* - In this procedure, the population members are sorted using the concept of Pareto dominance. The process begins by performing a classification in order to identify which are the non-dominated members of P. These first-ranked members are then placed in the first front and removed from the population P. Next, the non-dominating sorting procedure is performed on the remaining population

members. Further, to the non-dominated members of the remaining population are assigned the second rank and placed in the second front. This process continues until the whole population members are put on different fronts according to their ranks (VERMA; PANT; SNASEL, 2021). This concept is illustrated in Figure 2.5.

Figure 2.5 – Non-dominated sorting.



Reference: author

- *Elitism* - Elite preserving strategy retains the elite solutions of a population by directly transferring them to the next generation. In other words, the non-dominated solutions found in each generation move on to the next generations until some solutions dominate them.

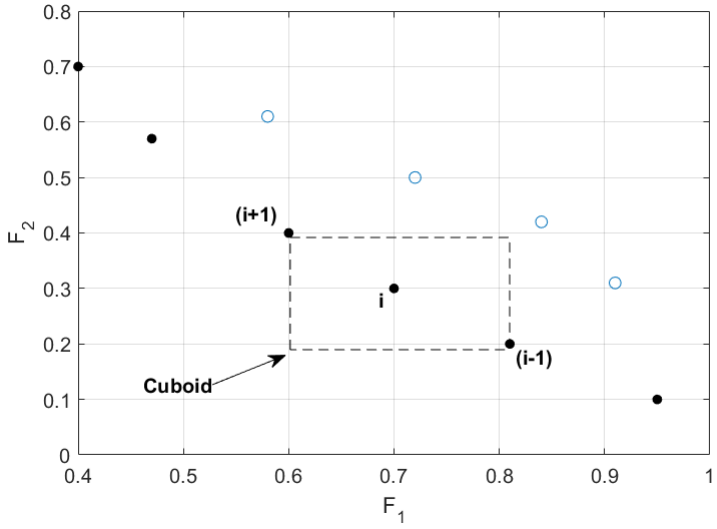
- *Crowding distance* - The crowding distance is computed to estimate the density of solutions surrounding a particular solution. This operator evaluates the distance of a solution i to its nearest neighbours $(i-1)$ and $(i+1)$ within the objective space. This distance is calculated as a normalised semiperimeter, according to Equation 2.13 and represented in Figure 2.6.

$$CD(i) = \sum_{j=1}^{m} \frac{f_j^{i+1} - f_j^{i-1}}{f_j^{max} - f_j^{min}} \tag{2.13}$$

  Solutions with a higher value of *Crowding distance* are privileged because they represent greater diversity. It should be noted that the solutions that are in extremities receive a value of $CD = \infty$.

- *Truncation Operator* - The population for the next generation is selected using a crowded tournament selection operator, which uses the rank of the population

Figure 2.6 – Crowding distance.



Reference: author

members and their crowding distances for the selection. The rule for selecting one out of two population members for the next generation is *i*) Should both the population members have different ranks, then the one with the better rank is selected for the next generation, *ii*) If both the population members are of the same ranks, then the one with the higher crowding distance is selected for the next generation (VERMA; PANT; SNASEL, 2021).

The procedure of NSGA-II begins with an initial population P_t of size N_P. Then, a new population of descendants, Q_t, is generated from the genetic operators applied on P_t. Next, the population P_t and Q_t are combined to form a new population R_t, and the non-dominated sorting procedure is performed. Then, the members of R_t are ranked into different fronts and the crowding distance for each individual is calculated.

The next procedure is to apply the Truncation Operator in R_t, to create the next population P_{t+1}, of size N_P. This process is repeated until a stop criteria is satisfied, which is usually a maximum number of generations or a number of evaluated solutions.

The pseudo-code of the NSGA-II algorithm is shown in Algorithm 3.

2.5.3 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) was proposed by Zitzler, Laumanns and Thiele (2001) and presents two main differences when compared to NSGA-II: the way of ranking the solutions and the use of elitism through an external population named *archive* (CARVALHO, 2022). SPEA2 introduced the concept of *strength value* to select new solutions to compose the next generations. The algorithm also incorporates a

Algorithm 3 NSGA-II - Minimisation Problem

```
1: function NSGA-II()
2:     Initialize parameters
3:     Initialize population P_t
4:     Evaluate population
5:     Generate Child Population Q_t
6:     R_t ← P_t ∪ Q_t
7:     while stop criteria is not reached do
8:         for i=1:length(R_t) do
9:             Assign Rank based on Pareto Dominance
10:            Generate sets of non-dominated solution
11:            Calculate Crowding Distance
12:            Add the solutions for the next generation starting from the first front until
    reaching N_P individuals
13:        end for
14:        Select points on the lower front with high crowding distance
15:        Apply crossover and mutation operators
16:        Create next generation
17:    end while
18:    return PF
19: end function
```

fine-grained fitness assignment strategy, which considers for each solution the number of solutions that it dominates and that it is dominated by.

In SPEA2, an initial population P_t is created with N_P individuals and an archive A_t, of size N_A is used to store the best solutions. Then, the individuals are evaluated the objective functions.

For all individuals, a *strength value* is computed, indicated by $S(i)$, and represents the number of solutions that a certain individual dominates.

$$S(i) = |\{j | j \in P_t \cup A_t \wedge i \prec j\}| \tag{2.14}$$

To each individual a value called Raw Fitness is assigned, which is equivalent to the sum of the strength values of the individuals that dominate the one which is being analysed, both in the population and in the archive.

$$R(i) = \sum_{j \in P_t \cup A_t | i \succ j} S(j) \tag{2.15}$$

Thus, the greater the strength value of an individual, the more individuals are dominated by it. The smaller the raw fitness value, the fewer individuals dominated it.

To provide information about neighbour density (similar to what *crowding distance* represents for NSGA-II) and guide the search effectively, SPEA2 uses a technique adapted

from the k-nearest neighbours algorithm (KNN) (FIX; HODGES, 1989), which considers the inverse of the distance to the k^{th} nearest neighbour as a density estimate. One way to do this is to consider the term $k = \sqrt{N_P + N_A}$ as a common point and list the results obtained for all individuals. After sorting in ascending order, the k^{th} element will be the one that gives the smallest searched distance, denoted by σ_i^k. Therefore, the density $D(i)$, corresponding to the individual i, is given by the equation 2.16.

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{2.16}$$

Finally, based on the raw fitness value and the neighbour density, it is possible to assign a fitness $F(i)$ to each individual, defined by the equation 2.17.

$$F(i) = R(i) + D(i) \tag{2.17}$$

Therefore, the smaller the value of $F(i)$ of an individual, the fitter it is, and the more chances it will have to propagate its characteristics. After this step of computing the fitness of all solutions, the non-dominated individuals are copied to the next generation, and if the number exceeds the maximum population size, a truncation operator is necessary to remove the unwanted individuals (CHOŁODOWICZ; ORŁOWSKI, 2017).

The evolutionary algorithm SPEA2 also uses the genetic operators of crossover and mutation, with their respective defined probabilities. The SPEA2 pseudo-code is represented in the algorithm 4.

Algorithm 4 SPEA2 - Minimisation Problem

```
1:  function SPEA2()
2:      Initialise parameters
3:      Initialise population P_t
4:      Create empty archive A_t
5:      while stop criteria not satisfied do
6:          Evaluate P_t ∪ A_t
7:          Compute F(i) of each individual in P_t ∪ A_t
8:          A_t ← Non-dominated solutions
9:          if size of A_t is bigger than N_A then
10:             Use truncation operator to remove elements from A_t
11:         end if
12:         Apply crossover and mutation operators
13:     end while
14:     return A
15: end function
```

2.5.4 Genetic Algorithm Operations

In Section 2.5 a set of evolutionary algorithms was described. All of them makes use of genetic operators in their respective optimisation processes. Such operators play fundamental roles for the success of each of them. Because of this, below is described which and how they were used in this work.

In Genetic Algorithms, only two kinds of operations are used, the genetic operation and the evolution operation. The genetic operations are crossover and mutation whereas the evolution operation is the selection (SILVA, 1999). It is worth noting that the MOEAs usually uses a binary representation of the individuals or chromosomes. However, real value individuals can, and were used in this work since they provide easier understanding and more freedom to use different genetic operators (KORA; YADLAPALLI, 2017).

2.5.4.1 Crossover Operation

Due to the choice of using a real representation for the solutions, the *arithmetic crossover* operator was adopted. In an arithmetic crossover, two chromosomes or individuals are selected for crossover $\mathbf{x}(t)$ and $\mathbf{y}(t)$, and by means of a linear combination of these chromosomes, two offspring are produced, $O_1(t)$ e $O_2(t)$. This linear combination is made as follows (CHAUHAN; SINGH; AGGARWAL, 2021):

$$O_1(t) = r\mathbf{x}(t) + (1-r)\mathbf{y}(t) \tag{2.18}$$

$$O_2(t) = r\mathbf{y}(t) + (1-r)\mathbf{x}(t) \tag{2.19}$$

where r is a random parameter in the range [0,1].

The performance or fitness of the offspring depends largely on the performance of the crossover operator used. A crossover rate is defined as the ratio between the number of offspring produced in each generation and the population size. Then, in Genetic Algorithms, this ratio, also known as crossover probability p_c, controls the number of chromosomes selected to undergo the crossover operation (SILVA, 1999).

2.5.4.2 Mutation Operation

The non-uniform mutation operator (NEUBAUER, 1997) was adopted and is used to change the individuals of a population at a probability p_m. When applied, an individual $\mathbf{x}'(t) = (x'_1(t), ..., x'_n(t))$ is modified according to the Equation 2.20.

$$\mathbf{x}'_i(t) = \mathbf{x}_i(t) + \delta_i \tag{2.20}$$

where δ_i is a sensitivity parameter associated with the limit of the search space for each variable. This operator is also very important for providing diversity to the solutions, helping the algorithm explore new regions in the search space.

2.5.4.3 Selection Operation

The selection operation is the process of selecting individuals for generating offspring. In this work, for both EAs (NSGA-II and SPEA2), the usual binary tournament selection operator was used and involves running two tournaments among four individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover (COELLO et al., 2007b).

For the NSGA-II algorithm, the evaluation is carried out by firstly considering the front to which the solution belongs, and then its crowding distance value. For SPEA2, the best individual is the one with the highest value of $F(i)$.

2.6 Quality Indicators

Because multi-objective algorithms have a set of solutions instead of a single one, direct comparisons between algorithms results cannot be directly performed.

Figure 2.7 illustrates this impasse by comparing the hypothetical results of two algorithms, A and B. For a given minimisation problem, in the first situation, Figure 2.7a, the solutions of algorithm A are better than those of B, since the frontier defined by A dominates all solutions of B, but it is not possible to say how much A is better than B. In the second case, Figure 2.7b, the results of the two algorithms are complementary, because some solutions of A dominate solutions of B, and some solutions of B dominate solutions of A. Determining which is the best algorithm requires understanding the characteristics of the solutions that can best be applied to the evaluated problem (SILVA et al., 2014).

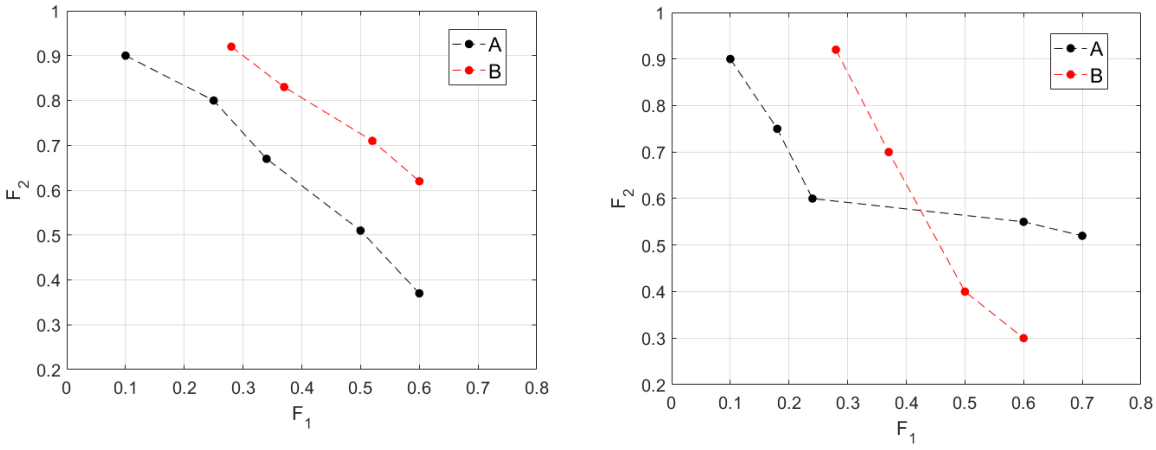
To allow a better comparison between multi-objective algorithms, several quality indicators have been proposed in the literature. In this section, some of the main metrics are presented and will be used later in Chapter 5.

2.6.1 Hypervolume (HV)

The Hypervolume was proposed by Zitzler and Thiele (1999) and it is the mostly used indicator to evaluate sets of non-dominated solutions (FEI et al., 2016). This indicator computes the area (or volume when more than two objectives are employed) of the region covered by the estimated Pareto Front, PF, and a reference point (usually, the nadir point). Formally, this indicator is described as the Lebesgue measure Λ of the union of

Figure 2.7 – Comparison of two Pareto Fronts for a two-objective function minimisation problem.

(a) A dominates B completely. (b) A and B are complementary.



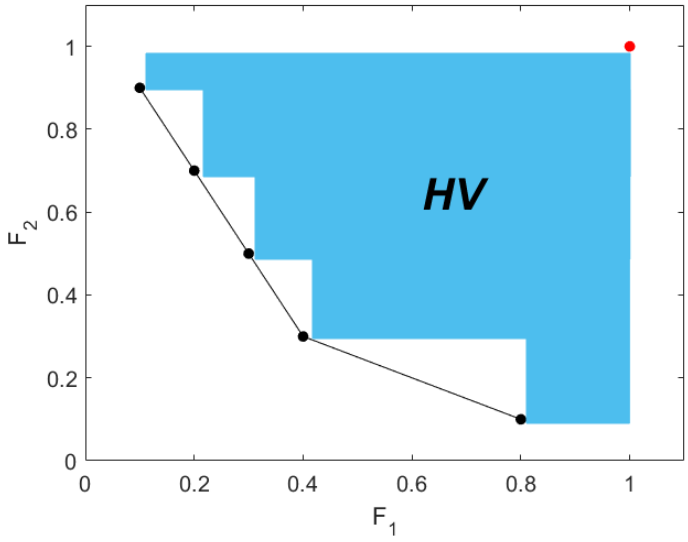
Reference: author

rectangles (or cubes, in $m > 2$) formed by consecutive points of the Pareto Front and can be generalized by the Equation 2.21.

$$HV(PF) = \Lambda\left(\left\{\bigcup_i a_i | \mathbf{x}_i \in PF_P\right\}\right) \tag{2.21}$$

where a_i is the area occupied by the solution $\mathbf{x}_i$ and PF_P represents the Pareto Front of the population P.

Figure 2.8 – Hypervolume.



Reference: author

Therefore, this metric reflects Pareto dominance, so that when one set dominates another, this will be reflected in the hypervolume. Assuming the minimisation of a MOP, higher hypervolumes are preferred to lower ones when the reference point is the nadir

point. In Figure 2.8, the blue region represents the hypervolume measure for a set of arbitrary solutions.

2.6.2 Spread (S)

The Spread (ZITZLER, 1999), or Δ metric, evaluates the distribution of non-dominated solutions over the Pareto Front (MAASHI, 2014). This indicator is calculated through the relative distance between consecutive solutions and is given by the equation 2.22.

$$S = \frac{1}{N_{PF} - 1} \sum_{i=1}^{N_{PF}} (\overline{L} - L_i)^2 \tag{2.22}$$

where $L_i = min_j(\sum_{k=1}^{m} |f_k^i(x) - f_k^j(x)|)$, for $i, j = 1, ..., N_{PF}$; $\overline{L}$ represents the distance between adjacent solutions and N_{PF} is the number of non-dominated solutions on the evaluated front.

This metric allows calculating the spacing uniformity of the solutions in the object space. The closer to zero, the better distributed the solutions on the Pareto Front will be. At $S = 0$, equidistant solutions are obtained.

For problems in which the Pareto Front has discontinuities, this metric can be misinterpreted since there is an increase in its value. Therefore, it is important to combine results with other indicators to perform a better analysis (SILVA et al., 2014).

2.6.3 Ratio of Non-dominated Individuals (RNI)

The Ratio of Non-dominated Individuals (TAN; LEE; KHOR, 2002) evaluates the percentage of non-dominated solutions $ND(P)$ in the population P, according to Equation 2.23. Higher RNI values are better than lower ones.

$$RNI = \frac{ND(P)}{N_P} \tag{2.23}$$

where N_P represents the population size.

2.6.4 Discussion

When evaluating the performance of multi-objective algorithms, there are two main goals to pursue (ELARBI et al., 2017):

- Convergence: closeness of the provided non-dominated solution set to the Pareto optimal front;

- Divergence: diversity of the obtained solution set (with a good distribution) along the Pareto optimal front.

Table 2.1 summarise the quality indicators presented. The mentioned indicators were chosen because they have different characteristics and, when use together can provide enough information for a good performance comparison of multi-objective algorithms.

Table 2.1 – Quality indicators comparison.

Quality Indicator	**Convergence**	**Divergence**	**Maximising/Minimising**
Hypervolume	X		maximising
Spread		X	minimising
RNI	X		maximising

Reference: author

2.7 Conclusion

Computational intelligence methods have proven to be effective in finding optimal solutions for a wide range of problems in various application areas. In this chapter, the fundamental concepts related to optimisation, metaheuristics, single and multi-objective algorithms, evolutionary algorithms, and quality indicators were presented. All of them will be used later to the tuning of the PI controllers used in the position and speed-controlled electric drive system.

CHAPTER 3

The Problem Characterisation

In this chapter, explanations are given on the speed and position controller tuning for a separately excited DC motor drive system by using the algorithms previously presented in Chapter 2.

Section 3.1 gives a brief introduction to the controller tuning problem, citing some related works, as well as the classical methodology used to solve this kind of problem. Sections 3.2 and 3.3 give a description of the system, where the DC motor is modelled in terms of block diagram, and the speed and position control problems are formulated. In Section 3.4, a set of objective functions, necessary to obtain a desired behaviour of the system, are presented. Section 3.5 explores the concept of the decision-making process, necessary for choosing solutions for multi-objective problems. Section 3.6 presents how the algorithms are applied to the controller tuning problem. At last, in Section 3.7, the statistical analysis used to compare the algorithm's performance is explained.

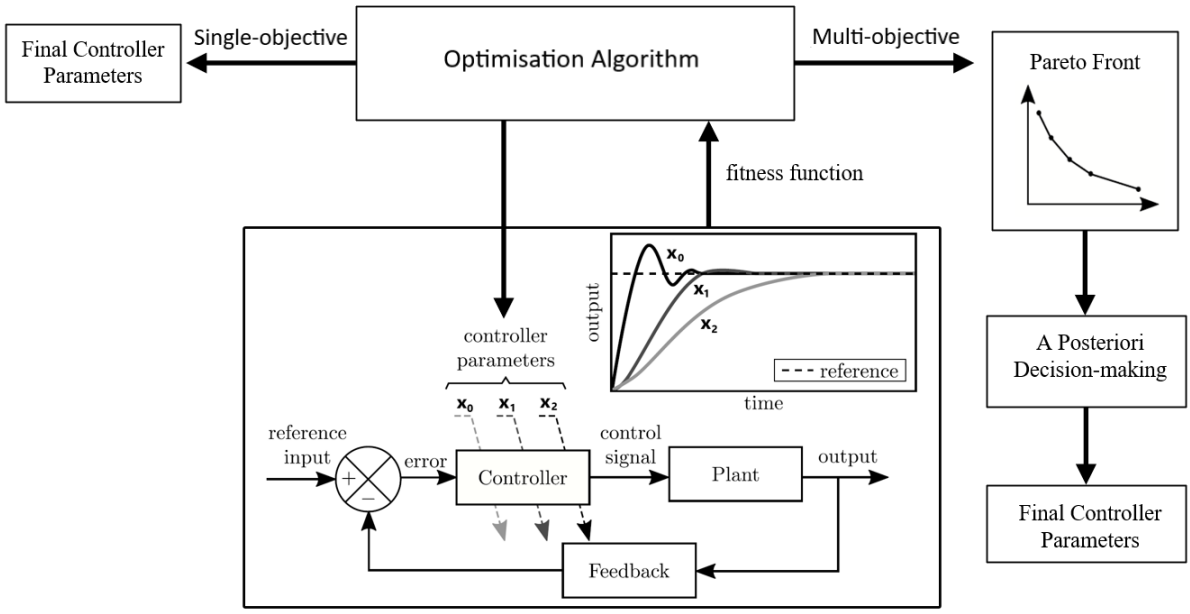
3.1 The General Controller Tuning Problem

Among the intelligent techniques, metaheuristics have been widely used to solve many real-world optimisation problems, as observed in Jovic et al. (2018), Deveci and Demirel (2018). The interest is due to their relatively simple operation, capability to deal with very complex problems at a reasonable computational cost, and their applicability within different contexts (TALBI, 2009). These characteristics make metaheuristics good alternatives to be applied in problems involving controller tuning in closed loop control systems.

The main objective of the controllers is to guarantee the stability of a dynamic system response and, to achieve it, a set of parameters must be considered for the system's stabilisation. These parameters can be adjusted to comply with different performance conditions such as acceleration, settling time (LEQUIN; GEVERS; TRIEST, 1999) and load disturbances (SILVA, 1999). Smoothing the transient is usually an issue to be targeted in dynamic systems stability (RIBEIRO; REYNOSO-MEZA, 2018).

Figure 3.1 illustrates how different parameter sets, denoted by $\mathbf{x}_0$, $\mathbf{x}_1$, and $\mathbf{x}_2$, used in the same controller structure, affect the stabilisation towards a reference value of a dynamic system. The same figure shows a general process of tuning controllers by using an optimisation algorithm. The algorithm is responsible for defining the controllers' setting of a feedback system, to accomplish system behaviour that minimises or maximises one or more objective functions. In a single-objective scenario, the algorithm's output consists of the final control parameters. However, in the case of a Multi-Objective Problem, the algorithm's output is a Pareto Front. A decision-making process is then carried out using the Pareto Front, leading to the final controller parameters. Each step will be covered in detail in the next sections.

Figure 3.1 – General multi-objective metaheuristic optimisation controller tuning process.



Reference: adapted from Rodríguez-Molina et al. (2020)

Known as the *controller tuning*, the correct setting of the controller parameters is usually a difficult task. The general controller tuning problem refers to the search for a set of controller parameters that stabilizes the dynamic system response under a given set of performance criteria and, at the same time, guarantees a desired behaviour (CARRASCO et al., 2020).

3.2 DC Motor Modelling

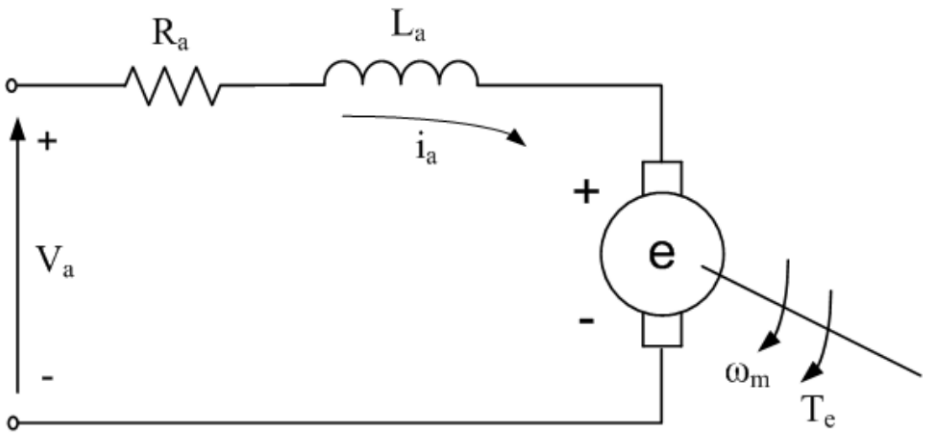
The DC motors present excellent functioning characteristics, i.e., large torque per unit of current for low speeds and great speed controllability (REIS et al., 2014).

Between the end of the 19^{th} century and the beginning of the 20^{th} century, the earliest power systems were in Direct Current – DC. However, gradually the AC power systems were gradually taking over the DC one. Despite this fact, DC motors continued to be a significant fraction of the machinery purchased for industrial application (CHAPMAN, 2013).

Today DC motors are still in use. For example, DC motors are used to power ancillaries in vehicles and can be found in household applications and electric tools. Another application for DC motors was a situation in which variable speed is needed. Before the widespread use of power electronic rectifier-inverters, DC motors were unbeatable in speed-controlled motor drive systems. Even if no DC power source were available, solid-state rectifier and chopper circuits were used to create the necessary DC power, and DC motors were used to provide the desired speed control (CHAPMAN, 2013). Currently, inverter fed induction motor drive represent a good choice over DC motors for most speed control applications. However, there are still applications where DC motors are preferred, such as: *i)* Textile machines; *ii)* Piston pumps; *iii)* Presses and *iv)* Vehicle applications (RONCHI, 2015).

The equivalent circuit of a separately excited DC motor is shown in Figure 3.2. In this configuration, the field is supplied from a separate constant-voltage power supply.

Figure 3.2 – Equivalent circuit of a separately excited DC motor.



Reference: author

Where:

- V_a - Armature voltage (V);

- I_a - Armature current (A);
- R_a - Armature resistance (Ω);
- L_a - Armature inductance (H);
- e - Electromotive force - emf (V);
- ω_m - Angular speed (rad/s);
- T_e - Electromagnetic torque (N·m).

By applying Kirchoff's voltage law to the circuit shown in Figure 3.2, is obtained Equation 3.1.

$$V_a = e + R_a I_a + L_a \frac{dI_a}{dt} \tag{3.1}$$

The electromotive force is a function of the field flux, ϕ_f, the motor angular speed, ω_m, and a constant, K, which is related to the machine design:

$$e = K\phi_f \omega_m \tag{3.2}$$

For the DC motor with constant field flux, Equation 3.2 can be written as:

$$e = K_b \omega_m \tag{3.3}$$

where K_b is the emf constant in (V/rad/s).

The power balance equation states that:

$$V_a I_a = eI_a + R_a I_a^2 \tag{3.4}$$

From Equation 3.4, one can clearly see that, $V_a I_a$, represents the input power and, $R_a I_a^2$, the armature loss. Then, the remaining term, eI_a, is the electrical air gap power which is transformed into mechanical one.

In terms of angular speed, ω_m, and electromagnetic torque, T_e, the mechanical power can be given as:

$$P_a = \omega_m T_e = eI_a \tag{3.5}$$

By substituting Equation 3.3 into 3.5, the electromagnetic torque can be found as:

$$T_e = K_b I_a \tag{3.6}$$

One can recognise that the *emf* constant, K_b, is the same one for the torque. However, in Equation 3.6 it is measured in (N·m).

The load torque is usually modelled as a moment of inertia, J, and a viscous friction coefficient, B. Then, from the electromechanical modelling, the acceleration torque can be given as:

$$J\frac{d\omega_m}{dt} + B\omega_m = T_e - T_L = T_a \tag{3.7}$$

Where:

- J - Inertia (Kg·m^2);
- B - Viscous friction coefficient (N·m/rad/s);
- T_L - Load torque (N·m);
- T_a - Acceleration torque (N·m);

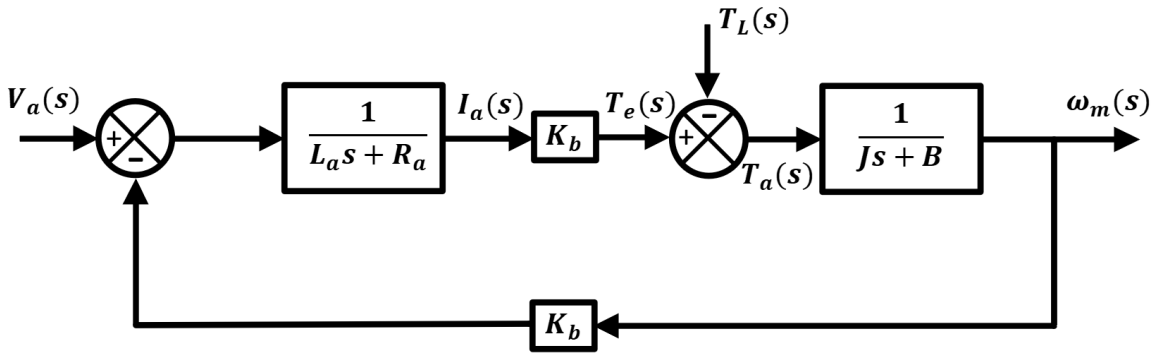
By applying Laplace Transform to 3.1 and 3.7, one obtains:

$$I_a(s) = \frac{V_a - K_b\omega_m(s)}{L_a s + R_a} \tag{3.8}$$

$$\omega_m(s) = \frac{K_b I_a(s) - T_L}{Js + B} \tag{3.9}$$

Equations 3.8 and 3.9 can be represented as the block diagrams given in Figure 3.3.

Figure 3.3 – Block diagram of the separately excited DC motor.



Reference: author

Two transfer functions are obtained from the block diagram: *i*) taking the armature voltage as input and angular speed as the output and *ii*) taking load torque as input and angular speed as output:

$$\frac{\omega_m(s)}{V_a(s)} = \frac{K_b}{JL_a s^2 + (BL_a + JR_a)s + (BR_a + K_b^2)} = G_{\omega V}(s) \tag{3.10}$$

$$\frac{\omega_m(s)}{T_L(s)} = \frac{-(L_a s + R_a)}{JL_a s^2 + (BL_a + JR_a)s + (BR_a + K_b^2)} = G_{\omega L}(s) \tag{3.11}$$

The speed response, taking the two simultaneous inputs, armature voltage and load torque, is:

$$\omega_m(s) = G_{\omega V}(s)V_a(s) + G_{\omega L}(s)T_L(s) \tag{3.12}$$

The position response is obtained from the integration of the speed as:

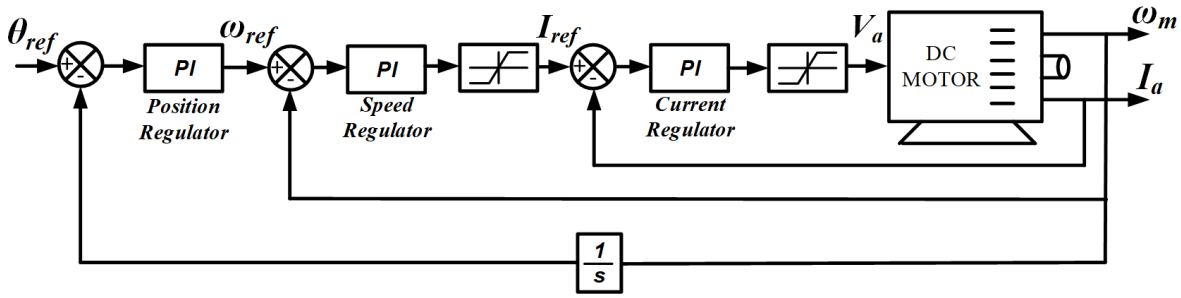
$$\theta(s) = \frac{\omega(s)}{s} \tag{3.13}$$

The inverse Laplace transform of 3.12 yields to the time speed response of the DC motor due to the armature voltage and load torque inputs, which is seen as a load disturbance. From 3.10 and 3.11, one can see that the separately excited DC motor represents a second order linear system.

3.3 The Closed Loop Speed and Position Control

The position control of a DC motor drive is carried out by using three control loops, as shown in Figure 3.4. The inner one represents the armature current control loop, the one in the middle, the speed, and the outer one, the position control loop. To control the armature current, the actual value, I_a, is compared to a reference one, I_{ref}, generating an error, which is taken into the PI controller. Therefore, the output of the PI current controller represents armature voltage, V_a, to the DC motor. The speed control is carried out by comparing the actual speed, ω_m, to a reference value, ω_{ref}, generating a speed error which is taken into the PI speed controller. The output of the speed controller represents reference armature current to the current control loop. Within the position control loop, the current position, θ, is compared to the reference value, θ_{ref}, generating a position error which is taken into the PI position controller. The output of the position controller represents reference speed value to the speed control loop.

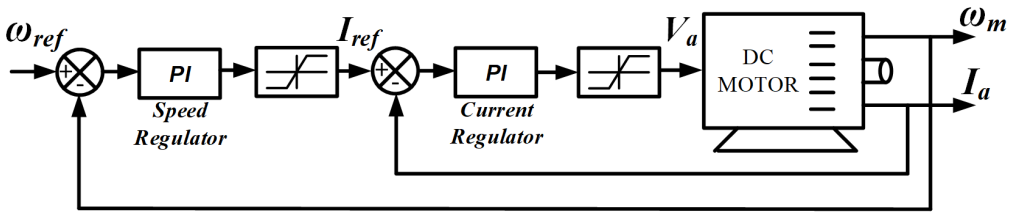
Figure 3.4 – Position control DC motor drive block diagram.



Reference: author

For speed control, the strategy is very similar to that required for position control, with the difference that the position control loop is eliminated and the system input is a reference speed, as illustrated in Figure 3.5. It is important to highlight that in this case, there is one less controller to be adjusted, resulting in a problem with a smaller number of variables.

Figure 3.5 – Speed control DC motor drive block diagram.



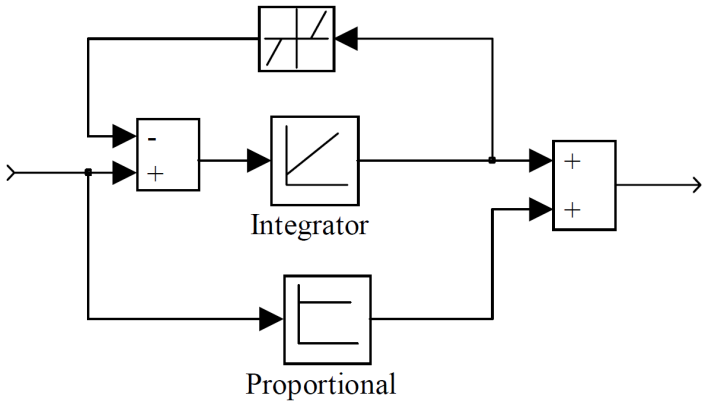
Reference: author

In this work, the armature current and voltage are limited by using a saturation block placed at the output of PI controllers. It means that the rated armature current demand and voltage will not exceed the rated values at any time. Then, it is possible to assign high gains values to the PI controllers, allowing the algorithm to look for the best tuning within a larger searching space. It is also important to highlight that high gain values can lead to large armature current demand to the DC motor and, therefore, large armature voltage applied to the motor. However, due to the limitations imposed to the controller's output, when the value of a variable reaches the maximum (or minimum) limit, signal saturation occurs. This causes the feedback loop to be ruptured, as the actuator will remain at its maximum (or minimum) limit regardless of the process output. Then, if the integral action of the controller is active, as in the case of a PI controller, the error will continue to be integrated and the integral term tends to become very large. Because of this, a problem known as *integrator windup* appears in almost every real system (SILVA; FLESCH; NORMEY-RICO, 2018), resulting in large overshoot and long settling times. To overcome the windup problem, many researchers have suggested anti-windup circuits to be used together with the PI controller. Although some authors have given

guidelines for anti-windup design (VRANCIC; PENG, 1996), tuning the controller for optimal performance in a non-linear system together with the anti-windup circuit may become a difficult task (SILVA; ACARNLEY; FINCH, 2001).

One of the basic anti-windup circuits as proposed in Silva, Acarnley and Finch (2001) uses a dead zone block to limit the integrator's action, as shown in Figure 3.6.

Figure 3.6 – PI controller with an anti-windup circuit based on the use of a dead zone block.



Reference: (SILVA; ACARNLEY; FINCH, 2001)

With such configuration, the functioning of the circuit is as follows:

1. When the integrator's output, which is the *input* of the dead zone block, is greater than the *positive limit*, the output of the dead-zone block becomes *output* = *input* – *positive limit*.

2. When the output of the integrator is smaller than *negative limit*, the output of the dead-zone becomes *output* = *input* – *negative limit*.

3. For any *input* of the dead zone in the interval [*negative limit*, *positive limit*], the output of the dead-zone block is zero.

Then, within the context of this work, the optimisation algorithms must find the best tuning of the PI controllers together with the dead zone window.

3.4 Objective Functions

Low cost, fast response, robustness, low sensitivity, small error rates, high accuracy, and efficient energy usage are examples of common requirements for closed-loop control systems, which may conflict with each other. These features strongly depend on the settling of the controller parameters (RODRÍGUEZ-MOLINA et al., 2020).

Depending on the closed-loop control system representation, different performance indexes are chosen in the reviewed works (RODRÍGUEZ-MOLINA et al., 2020; ZOLFAGHARIAN et al., 2013; EUZÉBIO et al., 2015). For a time-domain representation, some indices have been chosen to be used in this work, and are listed below:

- Integral Absolute Error (IAE) - Determines how far is the system response from the reference signal. It's calculated according to Equation 3.14.

$$IAE = \int_0^T |e(t)|dt \tag{3.14}$$

  where T represents the total simulation time and $e(t)$ is the error between the system response signal and its reference value at the time t. This indicator calculates the area formed between the two curves;

- Steady-state error (E_{ss}) - Is the deviation of the output of the controlled variable from its desired response at steady state;

- Rise Time (T_r) - Defined as the time required for the response to rise from 10% to 90% of its final value;

- Settling Time (T_s) - It is the time required for the response curve to reach and stay within a range of certain percentage (defined in this work at 2%) of the final value;

- Overshoot (OS) - It is a relation between the maximum value of the system response and the steady state response.

3.5 Decision Making for Multi-Objective Problems

Within the context of multi-objectives controller tuning problems, there is usually a moment when decision must be made, and they are related to the selection of the best trade-off towards finding the best solutions among the universe of possible ones.

There are two principal ways to deal with these preferences (COELLO et al., 2007b):

- *A priori*: The decision making is performed before searching for solutions. Using well-established preferences about the objectives is useful to define an aggregate objective function that indicates the pertinence degree of solutions. In some cases, the MOP can be transformed into a single-objective problem to which global solution includes the desired trade-off;

- *A posteriori*: The optimisation process is executed first and, afterwards, a decision method is applied. Once the Pareto Front is obtained, the decision-maker can select

the best trade-off based on preferences. Currently, there is a vast diversity of *a posteriori* decision-makers among which those based on the position of solutions stands out, on the closeness to a well-established reference point, or on the value of a utility function that ranks the preference of each objective.

A priori techniques are rarely used in the literature since they are not suitable for general use, requiring the decision maker to specify his preferences in advance (PURSHOUSE et al., 2014). The associated difficulty resides in the fact that the decision-maker does not always know in advance what is possible to achieve in the problem, how realistic his expectations are or the consequences arising from the relationship between the objectives (MIETTINEN; HAKANEN; PODKOPAEV, 2016). On the other hand, when carried out *a posteriori*, it is possible to visualise the Pareto Front, making the decision-making process more flexible as it is possible to clearly know what are the non-dominated solutions.

In this work, *a posteriori* decision-making process was adopted, which enables the generation of multiple possible solutions that highlight the conflict of interest. This approach allows for a comprehensive analysis of the problem and consideration of various factors in the decision-making process.

Since the algorithm gives a set of non-dominated solutions, the Weighted Sum Model (WSM), from decision theory, is used to evaluate the number of possible solutions in terms of a decision-making criteria. It means that, for a problem with N_P possible solutions, a vector, w, paring a different degree of relevance to each objective can be defined and, for each possible solution, a score can be computed such as:

$$A_i^{WSM-SCORE} = \sum_{j=1}^{m} w_j a_{ij}, i = 1, ..., n \tag{3.15}$$

where w_j is the weight associated to the objective f_j and a_{ij} is the performance of the solution A_i when evaluated according to the objective f_j. For a minimisation problem, the best solution is the one with the lowest score.

Through WSM, it is possible to establish a degree of importance for each of the objectives of the optimisation problem. Therefore, different non-dominated solutions for the speed or position-controlled DC motor drive can be obtained.

3.6 Computational Intelligence Applied to Controller Tuning

In this work, several optimisation algorithms are used to find the optimal tuning of PI controllers for the speed and position control of the DC motor drive as shown in Figures 3.4 and 3.5, together with the anti-windup circuits of the armature current and speed closed control loop. Each one of the dead-zone blocks in the anti-windup circuits

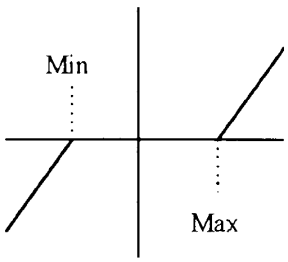
has an upper and a lower limit window. The absolute value was chosen to be the same then, a single value is necessary for each dead-zone block. Therefore, for position control, each individual, $\mathbf{x}_i^p$, is an eight-column size vector. The first six columns represent the proportional and integral gains of the PI controllers whereas the two remaining ones, the 7^{th} and 8^{th}, represent the dead-zone block window limit:

$$\mathbf{x}_i^p = (k_{pp}, k_{ip}, k_{p\omega}, k_{i\omega}, k_{pi}, k_{ii}, dz_\omega, dz_i) \tag{3.16}$$

Where:

- k_{pp} - Proportional gain of the PI position controller (1/s);
- k_{ip} - Integral gain of the PI position controller ($1/s^2$);
- $k_{p\omega}$ - Proportional gain of the PI speed controller (A·s/rad);
- $k_{i\omega}$ - Integral gain of the PI speed controller (A/rad);
- k_{pi} - Proportional gain of the PI current controller (V/A);
- k_{ii} - Integral gain of the PI current controller (V/A·s);
- dz_ω - Limit of the dead-zone window of the PI speed controller (A);
- dz_i - Limit of the dead-zone window of the PI current controller (V).

Figure 3.7 – Dead-zone block showing the dead-zone window (Min - Max).



Reference: (SILVA, 1999)

For speed control, the difference is that there is no position control loop, so the individual $\mathbf{x}_i^\omega$ is represented as follows:

$$\mathbf{x}_i^\omega = (k_{p\omega}, k_{i\omega}, k_{pi}, k_{ii}, dz_\omega, dz_i) \tag{3.17}$$

For a more comprehensive study, encompassing different types of configurations, algorithms and objectives, it was decided to formulate the problems in two different ways:

- Single-objective - This approach considers the minimisation of only one objective which uses the Integral of the Absolute Error (IAE) as the evaluation function. For the case of speed control, the algorithm must find the set of controller's gains that result in the lowest IAE value. The same goes for position control.

  Therefore, the PSO and SA algorithms were used to find a unique optimal solution that satisfied Equation 3.18. Through the value of IAE it is also possible to establish a direct comparison between the performance of the both algorithms.

$$\mathbf{x}^* = \min\{IAE(\mathbf{x}_1), IAE(\mathbf{x}_2), ..., IAE(\mathbf{x}_{N_P})\} \tag{3.18}$$

- Multi-objective - For this approach, four important classical control theory parameters should be considered: E_{ss}, T_r, T_s and OS. However, for such number of objectives to be targeted, it would not be possible to visualise the Pareto Front. Because of this, it was more convenient to form a vector of objective functions with only three of these objectives, as generally formulated in Equation 3.19. Then, a series of test results can be found for different combinations of objective functions to be targeted. The combination that presents characteristics with a higher level of conflict and diversity of solutions will be the chosen for simulations later, in Chapter 5.

$$\min \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})) \tag{3.19}$$

  Once the objectives have been defined, NSGA-II and SPEA2 algorithms can be used to find the set of non-dominated solutions that best satisfy the required specifications. To evaluate and compare the performance of multi-objective algorithms, the set of quality indicators already mentioned is used: Hypervolume (HV), Spread (S) and Ratio of non-dominated individuals (RNI). To visualise the possible different types of speed and position responses resulted from the setting of the PI controllers with the objectives shown at each Pareto Front, the WSM *a posteriori* will be used.

3.7 Statistical Analysis

A very important aspect in the study of computational intelligence algorithms is to evaluate their performance to determine if there is a significant difference between the investigated approaches (CARRASCO et al., 2020).

The results of a metaheuristic, even when applied to the same problem, may vary each time the algorithm is run. This behaviour is due to the stochastic elements inherently present in the operation. For this reason, it is common for the results to be provided taking into account the mean and standard deviation. Consequently, appropriate statistical tests are required to compare these results.

Firstly, it is necessary to define some concepts such as the *Hypothesis Test*, which is a procedure used to accept or reject a statistical hypothesis based on samples, classified as:

- Parametric test: have a normal sample data distribution;
- Non-parametric test: independent of sample data distribution, thus being a more generic and robust test.

For these procedures, some other concepts can be highlighted (MELO, 2009):

- Types of Hypothesis: H_0, called *null hypothesis*, designates the statistical hypothesis to be tested (e.g.: the means of two samples are equal), and by H_1 the *alternative hypothesis* (e.g.: the means of two samples are different). The null hypothesis expresses an equality, while the alternative hypothesis is given by an inequality;
- Types of errors in hypothesis tests:
  1. Type I error: rejection of a hypothesis when it is true;
  2. Type II error: acceptance of a hypothesis when it is false.

The probabilities of type I and II errors are primarily designated by the probability α, also known as the *significance level* of the test. As was said initially, the purpose of hypothesis testing is to determine whether the hypothesis null may or may not be accepted. This decision is taken considering a rejection region. If the observed value of the statistic belongs to the rejection region, H_0 must be rejected and it is said that there is not enough information to accept the null hypothesis; otherwise H_0 should not be rejected and it is said that there is not enough information to reject the null hypothesis (MELO, 2009).

In general, small values are assigned to α, the most common being $\alpha = 5\%$. The hypothesis H_0 is formulated with the aim of rejecting it. Hence, the null hypothesis name. The significance test does not eliminate the probability of error. However, it provides the probability value, called *p-value* (SCHERVISH, 1996) that allows you to decide, with respect to α, whether there is enough evidence to reject H_0 (if *p-value*$< \alpha$) (MELO, 2009).

There are multiple types of statistical tests, and the choice of the most suitable test for a given situation depends on some factors. To make an accurate selection, the following steps should be followed: *i)* determine whether there are two or more groups being compared, and *ii)* assess the normality of the sample data.

The first class of frequency-based tests that can be used to compare algorithm performance are parametric tests; which assume that the sample data comes from a normal

distribution, and that it can be described only by mean and standard deviation values (CARRASCO et al., 2020). The two main parametric tests are:

- t-test: This classic test is used to compare two samples. To perform this test, the required input is a set of observations obtained from different runs of the algorithms for a given problem;
- Analysis of Variance (ANOVA): Used to compare k different samples (k>2). This test is necessary since repeating the t-test for every pair of algorithms would increment the type I error (CASTILLO-VALDIVIESO et al., 2002).

To determine whether a sample data follows a normal distribution, the Shapiro-Wilk test can be employed (SHAPIRO; WILK, 1965). The test yields a *p-value*, and if this value is greater than α, it can be concluded that the sample follows a normal distribution, and parametric tests t and ANOVA can be employed. However, if the *p-value* is less than α, a non-parametric test should be utilised:

- Mann-Whitney U test: is a non-parametric alternative to the two-sample t-test and considers the magnitude of the difference between each compared pair. It is valid for data from any distribution and is less sensitive to outliers than the t-test (WILCOXON, 1992);
- Friedman test: non-parametric test equivalent to the ANOVA test, which seeks to compare more than two groups (FRIEDMAN, 1937).

It is important to note that when comparing multiple groups, the result of a statistical test only indicates the presence of a significant difference between at least two of them, without identifying which ones. Therefore, it is essential to conduct a *post-hoc* test for multiple comparisons. For ANOVA, the Tukey test can be used, while for the Friedman test, the Durbin-Conover test is recommended (CARRASCO et al., 2020).

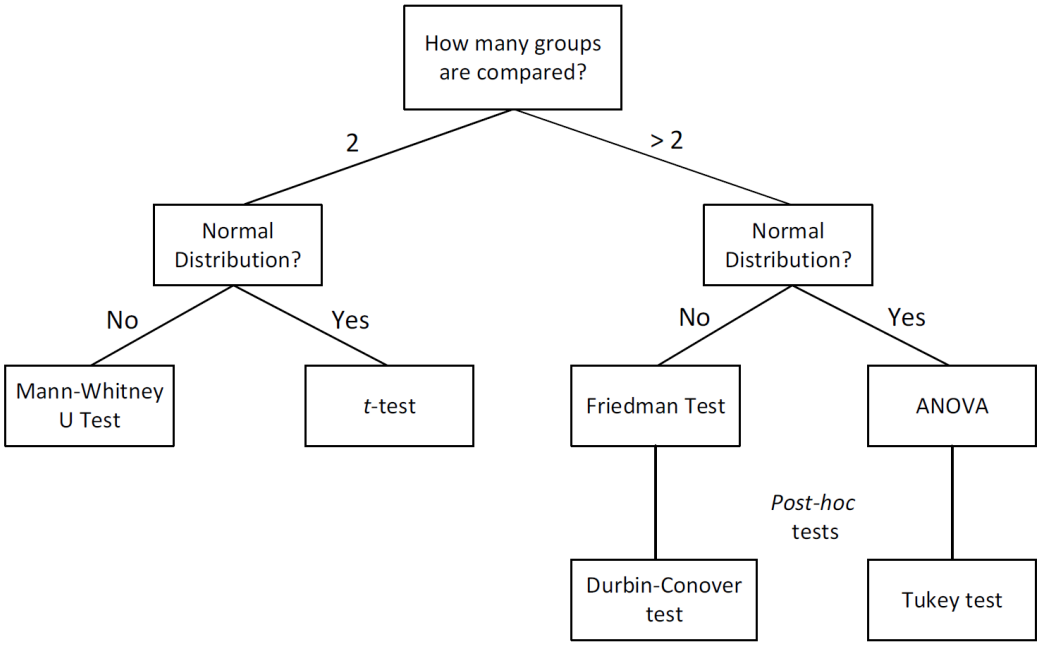
For all tests, parametric and non-parametric, the hypotheses are as follows:

- H_0: Null hypothesis that posits that the performances of the algorithms are equivalent;
- H_1: Alternative hypothesis that suggests that the algorithms do not have equivalent performance.

This work involves two types of comparisons: (i) evaluating the performance of an algorithm by varying some of its significant parameters, and (ii) comparing different algorithms applied to the same problem. These comparisons will be conducted in the

following chapters using the statistical tests discussed earlier. Figure 3.8 depicts a step-by-step process for selecting the most appropriate test for each situation that arises in this study.

Figure 3.8 – Frequentist tests



Reference: author

All the necessary tests will be performed using MATLAB R2022a.

3.8 Conclusion

Throughout this chapter, the controller tuning problem for speed and position control of a DC motor drive was characterised. Since it is inherently a non-linear problem with several parameters to be adjusted at the same time, within a very large search space, it represents a difficult task. Due to these characteristics, it is very challenging from the point of view of an optimisation problem. Because of this, the controller tuning problem was chosen to be addressed where the optimisation algorithms will be used with different parameter settings, which plays an important role on their performance. Furthermore, comparisons will be made to find out their relative strengths and/or weakness.

CHAPTER 4

Results: Single-objective Formulation

This chapter describes the investigations carried out concerning the single-objective formulation of the controller tuning problem. The description of the parameters used in the configuration of the test problems is present in Section 4.1. Section 4.2 presents the results regarding the speed control of a DC motor drive, while Section 4.3 is associated with position control.

4.1 Parameters Set up

To start the optimisation process, some parameters must be set up. The DC motor parameters, as obtained from KRİSHNAN (2001), are displayed in Table 4.1.

Table 4.1 – DC Motor Parameters.

Symbol	**Parameter**	**Value**
V_a	Armature Voltage	220 (V)
R_a	Armature Resistance	0.5 (Ω)
L_a	Armature inductance	3 (mH)
J	Inertia	0.0167 (Kg·m²)
B	Viscous coefficient	0.01 (N ·m/rad/s)
K_b	Emf constant	0.8 (V/rad/s)
K_t	Torque constant	0.8 (N·m/A)

Reference: adapted from KRİSHNAN (2001)

In this work, the maximum I_{ref} and V_{ref} were limited to 20 (A) and 220 (V) respectively through a saturator block. The maximum step input reference speed and

position was set up as 100 units with 5 (N·m) constant load torque. The simulation time was set to 5 (s).

The range of the proportional and integral gains of all the PI controllers were set to vary from 0 to 20 units. The dead-zone window for the anti-windup circuit of the PI speed regulator was set to vary from 0 to 20 (A). It means that, the dead-zone window ranges from -20 to 20 (A). The dead-zone window of the PI current controller was set to range from 0 to 220 (V), what means that the dead-zone window ranges from -220 to 220 (V). The limits imposed are large enough to cause big overshoots and long settling times. This makes the search space very large and the task for the algorithms more difficult. Table 4.2 presents the limits of the search space.

Table 4.2 – Search space.

Variable	**Range**	**Unit**
k_{pp}	0-20	(1/s)
k_{ip}	0-20	($1/s^2$)
$k_{p\omega}$	0-20	(A·s/rad)
$k_{i\omega}$	0-20	(A/rad)
k_{pi}	0-20	(V/A)
k_{ii}	0-20	(V/A·s)
dz_{ω}	0-20	(A)
dz_i	0-220	(V)

Reference: author

The DC motor drive was developed in the SIMULINK environment whereas the PSO and SA algorithms were run in MATLAB. The computer used to run the algorithm was an Intel Core i7-7500U 7^{th} Gen with 256 (GB) SSD, 8 (GB) RAM Memory and GeForce 920MX Dedicated Graphics Card.

4.2 Case Study I: Speed Control

4.2.1 PSO

As already mentioned throughout this work, most algorithms have some parameters that may need to be carefully defined for the algorithm to achieve better performance. Maximising the performance of these algorithms may not be a simple task since the algorithm's parameter settings can have several different possible configurations. (HUTTER et al., 2014). Despite there being a couple of different parameter suggestions available in the literature, they are generally determined for specific problems or application contexts. Therefore, it may happen that, when facing a particular problem, an unusual configuration of algorithm parameters presents a superior performance than the general and typical one as suggested in the literature (LÓPEZ-IBÁÑEZ et al., 2016).

There are several publications related to this area known as *automatic algorithm configuration* (SOUZA, 2022), in which methods are proposed for algorithm set up. It is not the focus of this work to carry out a detailed study about the parameters of the algorithms used here, but neglecting any type of adjustment could affect their performance and also the comparisons made between the algorithms. In this way, several variations of the main parameters of each algorithm were tested, aiming to identify the impact of these parameters on the convergence of the algorithms when applied to the tuning of the PI speed and position controllers of a DC motor drive.

For the PSO algorithm, there are three main parameters associated with its performance, they are: *i*) population size (N_P); *ii*) acceleration coefficients (c_1 and c_2) and *iii*) inertia weight ($w(t)$). The importance of each parameter was discussed in Chapter 2. Table 4.3 presents the different values and settings assigned to each of these parameters.

Table 4.3 – PSO parameters used.

Parameters	**Values**
Population size	{20, 50, 100}
Acceleration coefficients	$\{c_1 = c_2,\ c_1 > c_2,\ c_1 < c_2\}$
Inertia weight	{Linear decreasing, Random adjustments}

Reference: author

Three different population sizes (20, 50 and 100 individuals) were selected in order to identify how much this quantity influences the convergence of the algorithm. The values were chosen based on Piotrowski, Napiorkowski and Piotrowska (2020), Santos et al. (2021b) combined with the designer's experience about the need for each problem.

Cognitive and social coefficients, also known as acceleration coefficients, were tested in three different configurations. In the first one, the coefficients had the same value, $c_1 = c_2 = 1.49445$. In the second configuration, the cognitive component had a higher value than the social one, with $c_1 = 2.05$ and $c_2 = 0.5$. Finally, a more relevant degree of importance was given to the social component than to the cognitive one, $c_2 = 2.05$ and $c_1 = 0.5$. The settings, as well as the values used, were obtained from Silva et al. (2014).

Regarding the inertia weight, two popular approaches present in the literature were considered: *i*) random adjustment and *ii*) linear decreasing, as presented in Equations 2.4 and 2.5. It was suggested by Ratnaweera (2002), in the case of linear decreasing method, that the inertia weight starts with a high value (usually 0.9) and reduces to 0.4. However, the value of $w(t) = 0.4$ contradicts Equation 2.3, given the values of c_1 and c_2 chosen above. In this way, it was chosen in this work so that the linear variation of the inertia weight occurs between 0.9 and 0.5.

In summary, three population sizes, three relations for the acceleration coefficients and two approaches for variation for the inertia weight have been chosen. As the mentioned

parameters are simultaneously present in the algorithm, there are 18 possible different combinations to configure the PSO. All these possibilities are investigated to evaluate the impact of the parameters on the convergence of the algorithm and also on how they influence each other. Due to the nature of metaheuristics, all configurations were repeated 20 times, so that the random aspect cannot directly influence the results of performance comparisons, thus, providing greater reliability in the use of statistical tests.

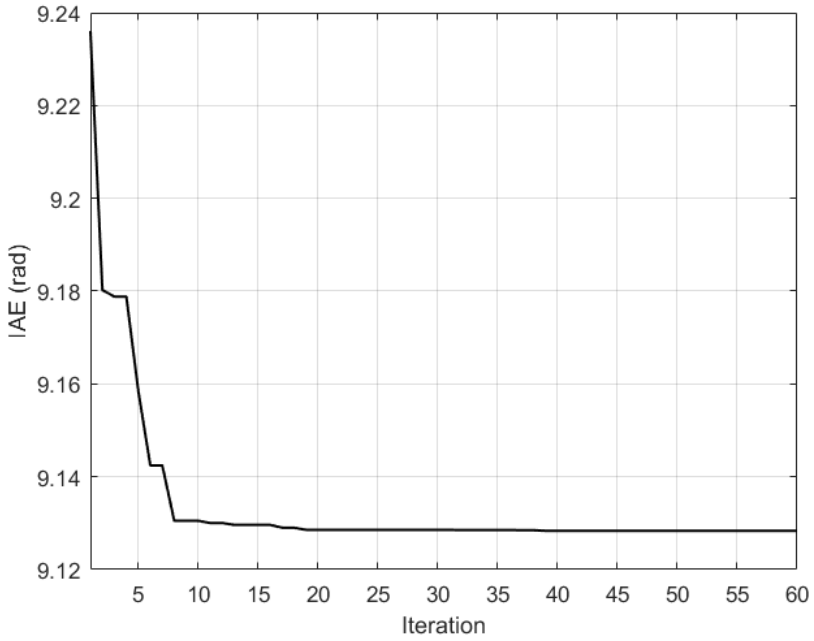
The utilisation of optimisation algorithms has a well-defined goal, which is to find the global minimum. However, it is often unclear whether this objective has been achieved, especially when addressing real-world problems for which the global optimum is not known. Therefore, it is not an easy task to decide when the execution of an algorithm should be terminated. The scientific literature on metaheuristics lacks a definitive consensus regarding stop criteria. Nevertheless, the most common approach involves setting a maximum number of fitness evaluations, which is often tailored to the problem and frequently determined through the experience of the researcher (LIMA, 2017; CHEN et al., 2022; HU et al., 2022).

In this work, to establish the stop criteria, an exhaustive trial using the Particle Swarm Optimisation (PSO) algorithm was performed. For the speed control problem, a population of 100 individuals was employed, and a total of 6000 solutions were evaluated. The IAE objective function evaluation graph was plotted over the iterations, as shown in Figure 4.1. This figure reveals that only minor improvements occurred after the eighth iteration (which corresponds to 800 fitness evaluations), which does not justify letting the algorithm run for a longer time beyond that. Then, a slightly higher stop criteria was established for the remaining simulations, limiting the number of particle evaluations per algorithm's run to 900. Although this value may be considered low when compared to the search space, it has proven to be satisfactory, enabling the evaluation of the algorithm's capacity to find a good solution within a shorter time.

To ensure a fair comparison between different algorithm configurations, the stop criteria must always be the same. However, it should be noted that when running the algorithm with varying population sizes but a fixed number of fitness evaluations, each configuration will require a different number of iterations. A smaller population size, for instance, will require more iterations, whereas a larger population size will need fewer iterations to reach the stop criteria. Specifically, a population size of 20 individuals demands 45 iterations to meet the stop criteria, while populations with 50 and 100 individuals require 18 and 9 iterations, respectively.

Table 4.4 presents the minimisation results obtained by the PSO, with mean and standard deviation of the IAE for speed control of a separately excited DC motor drive, considering the different scenarios of mentioned parameters. The average simulation time

Figure 4.1 – Definition of the stop criteria - Speed control problem.



Reference: author

for each run of the algorithm was 470.5 (s). The value in bold represents the configuration with the lowest average obtained.

The configuration that presented the lowest average was the one with the smallest population, cognitive coefficient bigger than the social one, and the linear decreasing method of the inertia weight (configuration number 4). It is also evident the low standard deviation and the small difference between the means of all configurations, leads to conclusion that the stop criteria was appropriate to the problem.

From the results obtained using different PSO configurations, it is possible to identify direct interactions between the algorithm's parameters. This helps to establish relationships between specific settings that lead to greater synergy and more effective minimisation of the IAE function. In Figure 4.2a the relationship between population size and acceleration coefficients is shown. Populations with 20 and 50 individuals obtained better averages when in collaboration with $c_1 > c_2$, while the larger population achieved better results when combined with a higher social coefficient. This behaviour can be understood as follows: smaller populations occupy smaller regions within the search space, which implies the need for individuals to assume a greater role, distancing themselves from the collective to carry out a better exploration. On the contrary, large populations need less of this individual exploration, as they naturally occupy a larger region in the search space, meaning that the social coefficient is more important for these groups when compared to the population of less individuals.

Table 4.4 – IAE value results for the PSO algorithm to the speed control problem.

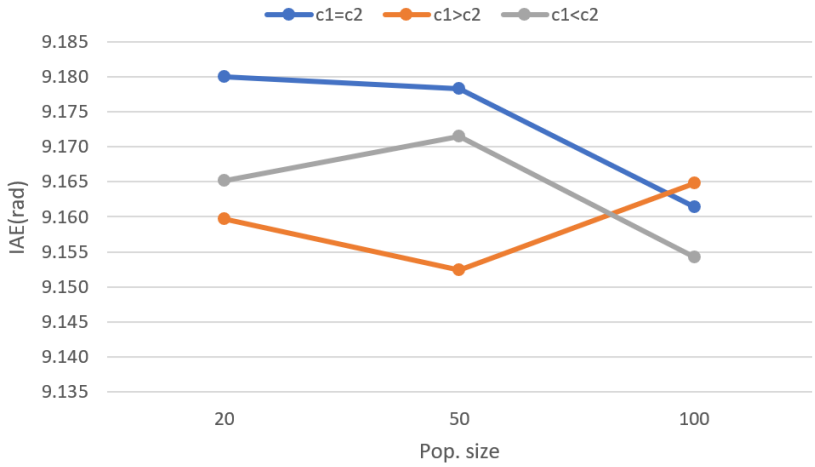
	Configuration			**Average (rad)**	**Standard Deviation (rad)**
	Pop. size	**A. coefficients**	**Inertia**		
1	20	$c_1 = c_2$	Linear	9.1708	0.0472
2	50	$c_1 = c_2$	Linear	9.1768	0.0470
3	100	$c_1 = c_2$	Linear	9.1565	0.039
4	**20**	$\mathbf{c_1 > c_2}$	**Linear**	**9.1468**	**0.0342**
5	50	$c_1 > c_2$	Linear	9.1551	0.0385
6	100	$c_1 > c_2$	Linear	9.1709	0.0334
7	20	$c_1 < c_2$	Linear	9.1512	0.0485
8	50	$c_1 < c_2$	Linear	9.1667	0.046
9	100	$c_1 < c_2$	Linear	9.1593	0.0394
10	20	$c_1 = c_2$	Random	9.1887	0.0307
11	50	$c_1 = c_2$	Random	9.1797	0.0334
12	100	$c_1 = c_2$	Random	9.1664	0.0299
13	20	$c_1 > c_2$	Random	9.1725	0.0378
14	50	$c_1 > c_2$	Random	9.1498	0.0224
15	100	$c_1 > c_2$	Random	9.1587	0.0357
16	20	$c_1 < c_2$	Random	9.1790	0.0446
17	50	$c_1 < c_2$	Random	9.1762	0.0383
18	100	$c_1 < c_2$	Random	9.1492	0.0164

Reference: author

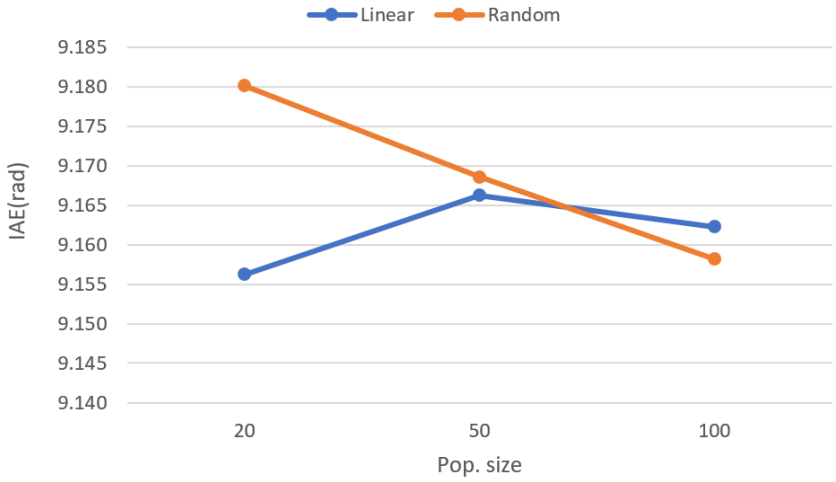
Figure 4.2b, relates the size of the population with the type of variation in the inertia weight. An opposite behaviour of the curves can be observed, with linear variation being preferable for smaller populations, while a larger population presents advantage when used together with random adjustment. Especially regarding populations of 50 and 100 individuals, the average difference is minimal, with no real advantage for either side. Finally, in Figure 4.2c, the interaction between the acceleration coefficients and the types of inertia weight variation is presented, showing the superiority of linear decreasing in all cases.

Figure 4.2 – Interaction between PSO parameters.

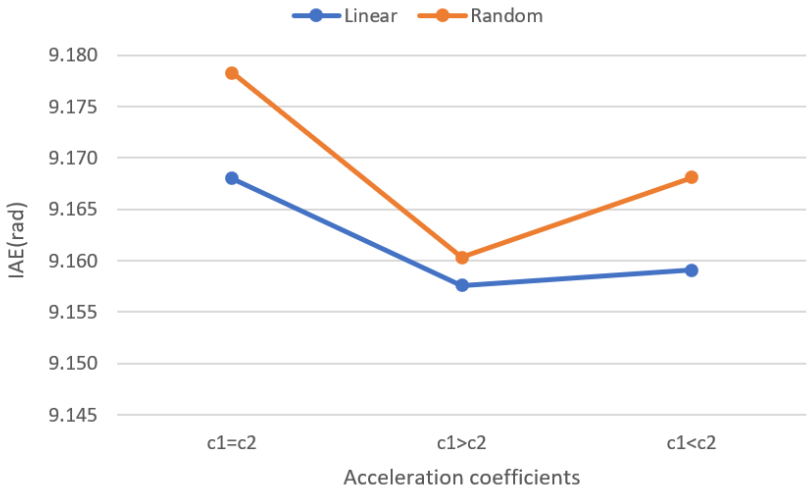
(a) Population and acceleration coefficients.



(b) Population and variation of inertia weight.



(c) Acceleration coefficients and variation of inertia weight.

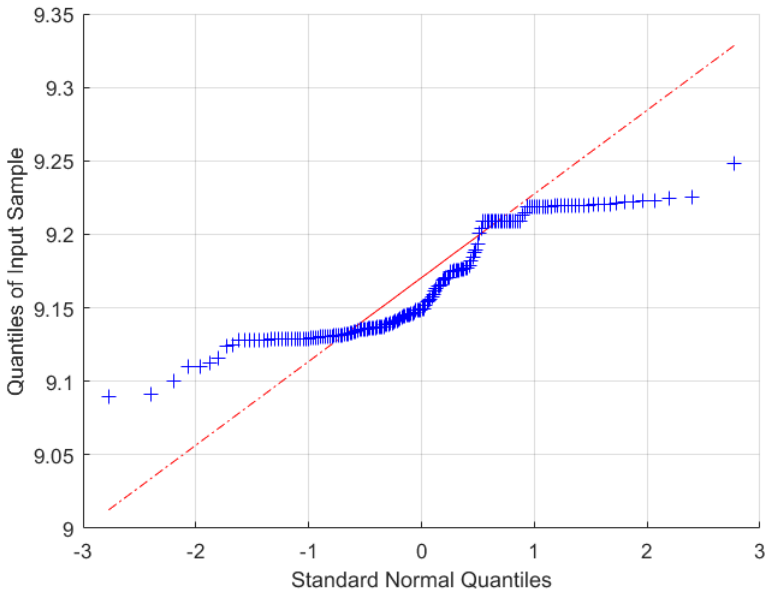


Reference: author

Regarding the statistical approach, the Shapiro-Wilk test was first performed to identify whether the results have a normal distribution, making it possible to define the most appropriate test to compare the results. Therefore, the test was carried out considering all the results obtained by the PSO algorithm, i.e., referring to the 18 configurations and considering the 20 repetitions of each one. From these data, a *p-value*$<$ 0.0001 was obtained, which means that the null hypothesis must be rejected (H_0: data sample has normal distribution).

The previous result is also evident from the Q-Q plot, Figure 4.3, in which the normal distribution is compared with the distribution of the data obtained by the algorithm. As the points follow a non-linear pattern, displaced from the red line, it is suggested that the data are not distributed with a normal pattern, and therefore, the non-parametric Friedman test is the most suitable for comparing results.

Figure 4.3 – Q-Q Plot of PSO sample data *versus* standard normal - Speed control problem.

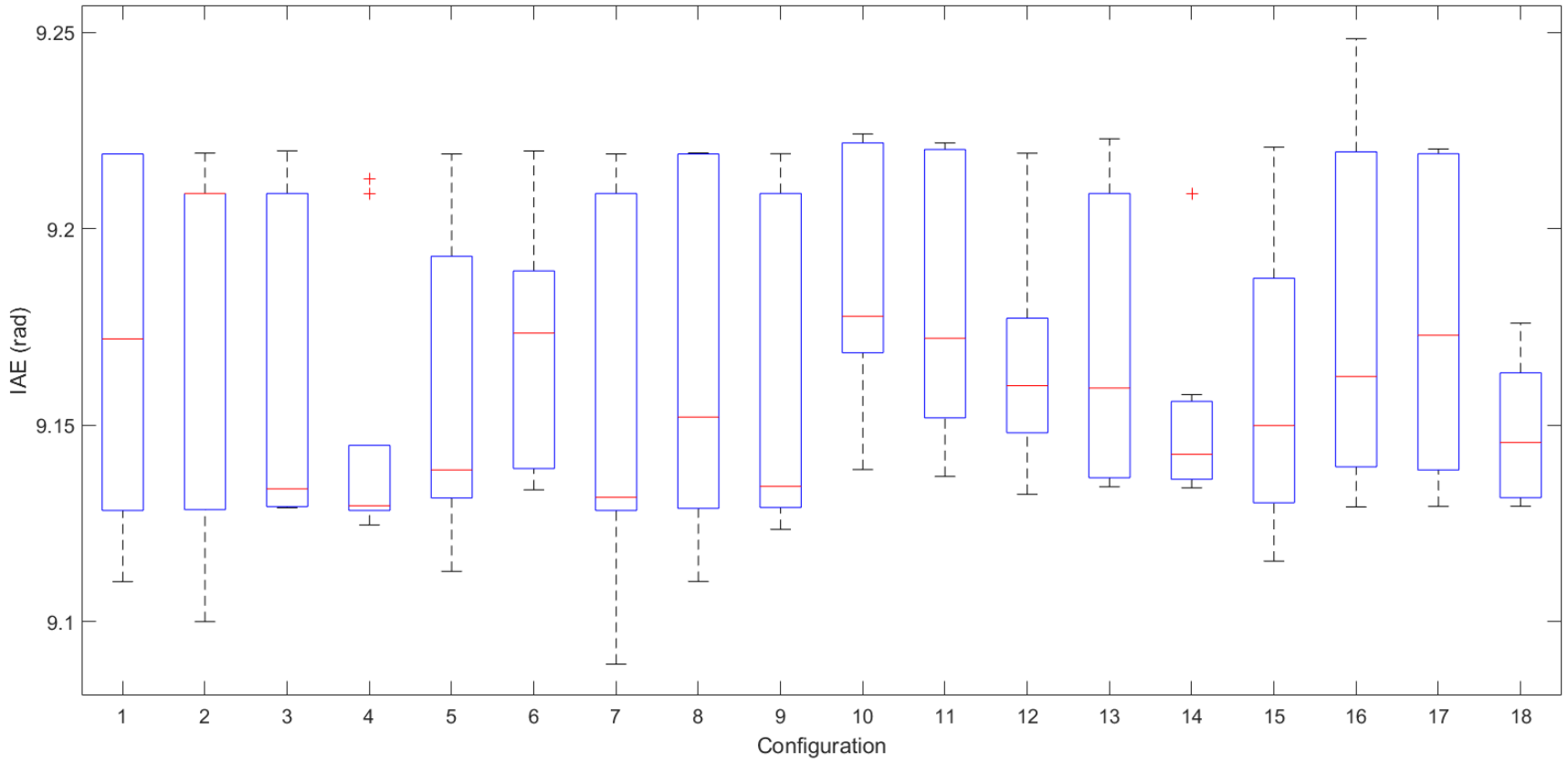


Reference: author

Through the Friedman test, a *p-value*=0.3796 was obtained, which means that there was no statistically meaningful difference between the configurations, thus, it can be said that the algorithm had the same performance, regardless of the adopted configuration.

Figure 4.4 shows the box plot for the PSO algorithm. Through this diagram it is possible to observe the numerical variation by means of quartiles. The box plot has lines that extend vertically, indicating the variability of the upper quartile and the lower one. Outliers are plotted with a "+" sign, and the red horizontal line represents the median for each configuration.

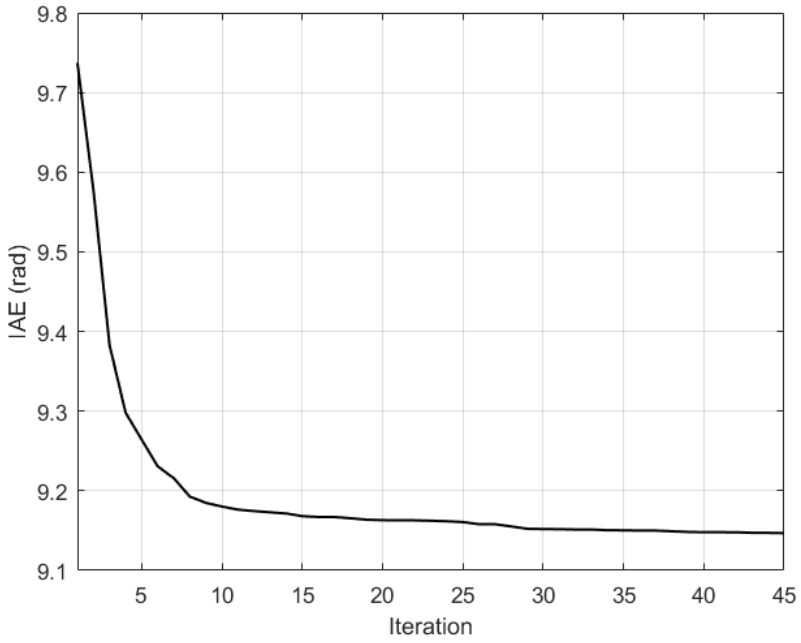
Figure 4.4 – Box plot of PSO algorithm for the tuning of the controllers for the speed-controlled DC motor drive.



Reference: author

To illustrate part of the behaviour of the PSO, configuration number 4 was selected and the minimisation process of the IAE function is shown in Figure 4.5.

Figure 4.5 – Average of integral of the absolute speed error along the iterations - PSO Algorithm.



Reference: author

4.2.2 SA

Like what was done for the PSO, different values for important parameters of the SA algorithm were also explored, in order to analyse its performance.

The main parameters of SA are the initial temperature T_0, cooling coefficient α and number of sub-iterations N. Aiming to obtain the same number of configurations tested for the PSO, 18, three different values were chosen for the initial temperature, three values of cooling coefficients and finally, two amounts of sub-iterations. The values that were adopted are shown in Table 4.5.

Table 4.5 – SA Parameters.

Parameters	**Values**
Initial Temperature (T_0)	{10, 50, 100}
Cooling coefficient (α)	{0.8, 0.9, 0.99}
Number of sub-iterations (N)	{1, 4}

Reference: author

The initial temperature in the SA algorithm plays an important role in the search process since it is associated with the probability that the initial transitions are accepted.

To find an appropriate value for this parameter, three different ones were considered in this work: 10, 50 and 100. Regarding the cooling coefficient, it is widely disseminated in the literature that the value of α should be between 0.8 and 0.99 (DELAHAYE; CHAIMATANAN; MONGEAU, 2019). Finally, for the number of sub-iterations, which is associated with the number of solutions evaluated at the same temperature, values of 1 and 4 are proposed in Silva, Santos and Pickert (2020).

The Table 4.6 shows the mean and standard deviation results obtained by SA. 20 repetitions of each configuration were also performed. The average simulation time was 440.3 (s). It is important to note that the stop criteria was again 900 fitness evaluations, the same established for the PSO.

Table 4.6 – IAE value results for the SA algorithm to the speed control problem.

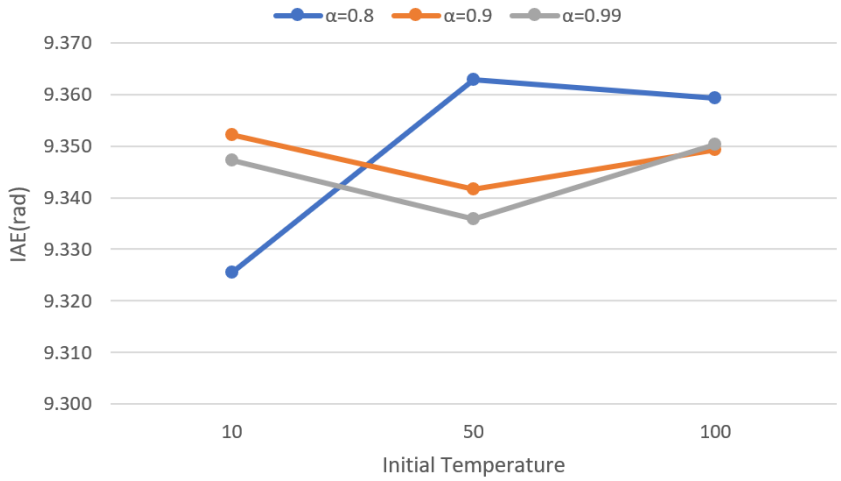
	Configuration			**Average (rad)**	**Standard Deviation (rad)**
	T_0	α	**N**		
1	10	0.8	1	9.3337	0.0705
2	50	0.8	1	9.3570	0.0436
3	100	0.8	1	9.3632	0.0375
4	10	0.9	1	9.3495	0.0601
5	50	0.9	1	9.3361	0.0655
6	100	0.9	1	9.3532	0.0484
7	10	0.99	1	9.3584	0.0628
8	50	0.99	1	9.3269	0.0622
9	100	0.99	1	9.3734	0.0402
10	**10**	**0.8**	**4**	**9.3172**	**0.0294**
11	50	0.8	4	9.3687	0.0296
12	100	0.8	4	9.3553	0.0460
13	10	0.9	4	9.3548	0.0712
14	50	0.9	4	9.3472	0.0336
15	100	0.9	4	9.3454	0.0481
16	10	0.99	4	9.3360	0.0603
17	50	0.99	4	9.3447	0.0806
18	100	0.99	4	9.3272	0.0475

Reference: author

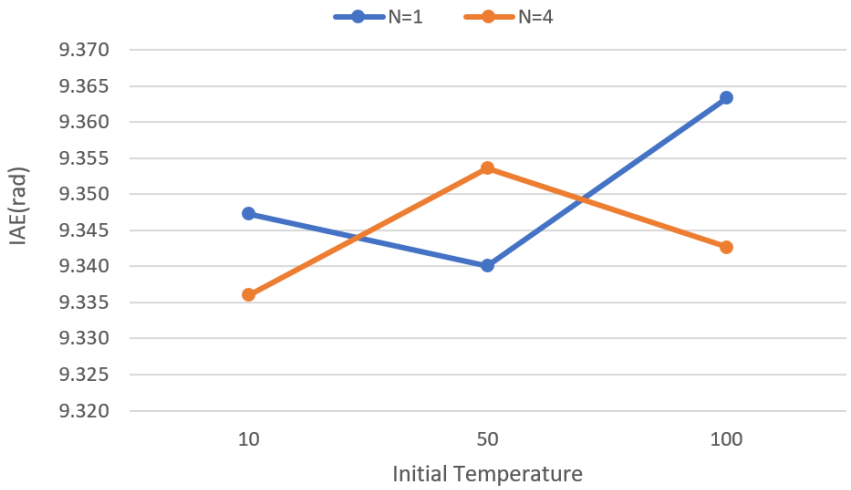
The configuration that presented the lowest IAE value was number 10, i.e, the lowest value of T_0, as well as the lowest cooling coefficient and $N = 4$. Similarly to the PSO results, the SA algorithm converged to near final values of IAE, regardless of the parameter configurations used. Additionally, low standard deviations were observed in all tests, indicating the algorithm's robustness as well as its ability to find optimal solutions for the problem.

Figure 4.6 – Interaction between SA parameters.

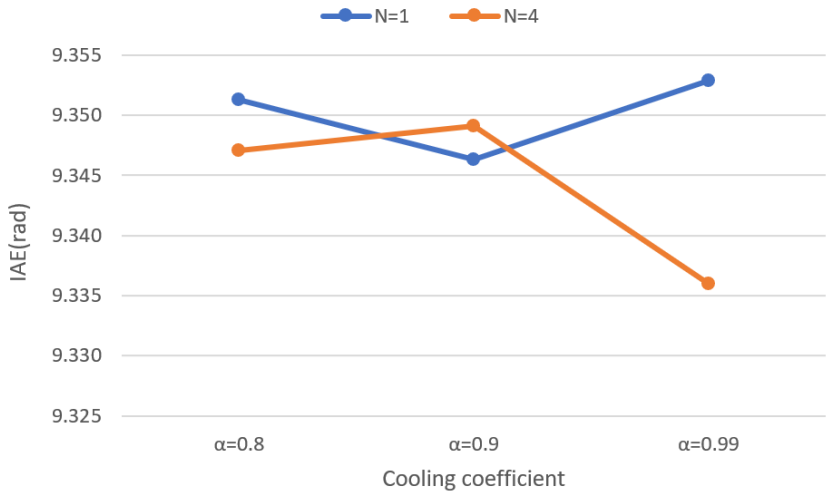
(a) Initial temperature and cooling coefficient.



(b) Initial temperature and number of sub-iterations.



(c) Cooling coefficient and number of sub-iterations.

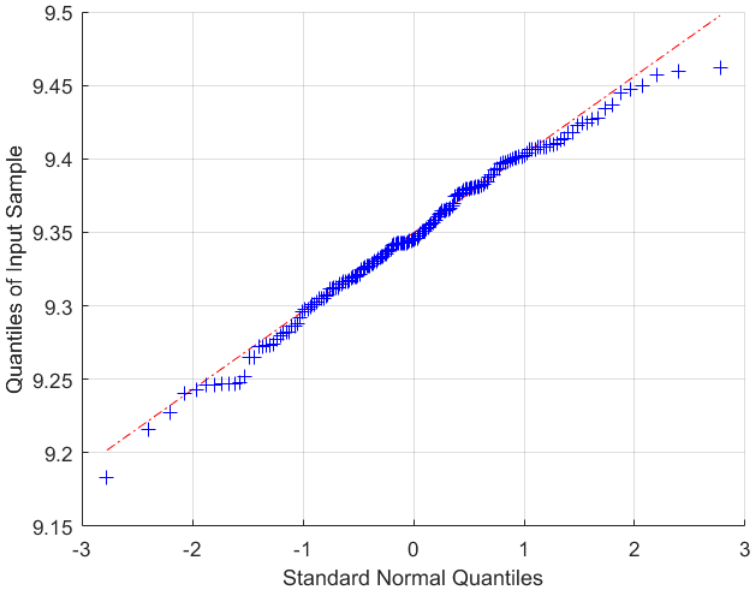


Reference: author

Figure 4.6 presents the results of interactions between the SA parameters setting. In Figure 4.6a the relationship that occurred between T_0 and α is established. For the lowest initial temperature, the coefficient that promoted the fastest cooling proved to be more efficient, while for high temperatures, the slower cooling, obtained lower averages. Figures 4.6b and 4.6c show the other relationships, showing that higher values of N were preferable.

The statistical comparison for the SA algorithm starts with the Shapiro-Wilk test, which provided a *p-value* = 0.2465, that is, as *p-value*>α, the null hypothesis H_0 must be accepted, meaning that the results have an approximately normal distribution. This conclusion is graphically reinforced through Figure 4.7, where the linearity of the points suggests that the sample data are normally distributed. Thus, the parametric ANOVA test can be used to continue the comparison of results.

Figure 4.7 – Q-Q Plot of SA sample data *versus* standard normal - Speed control problem.

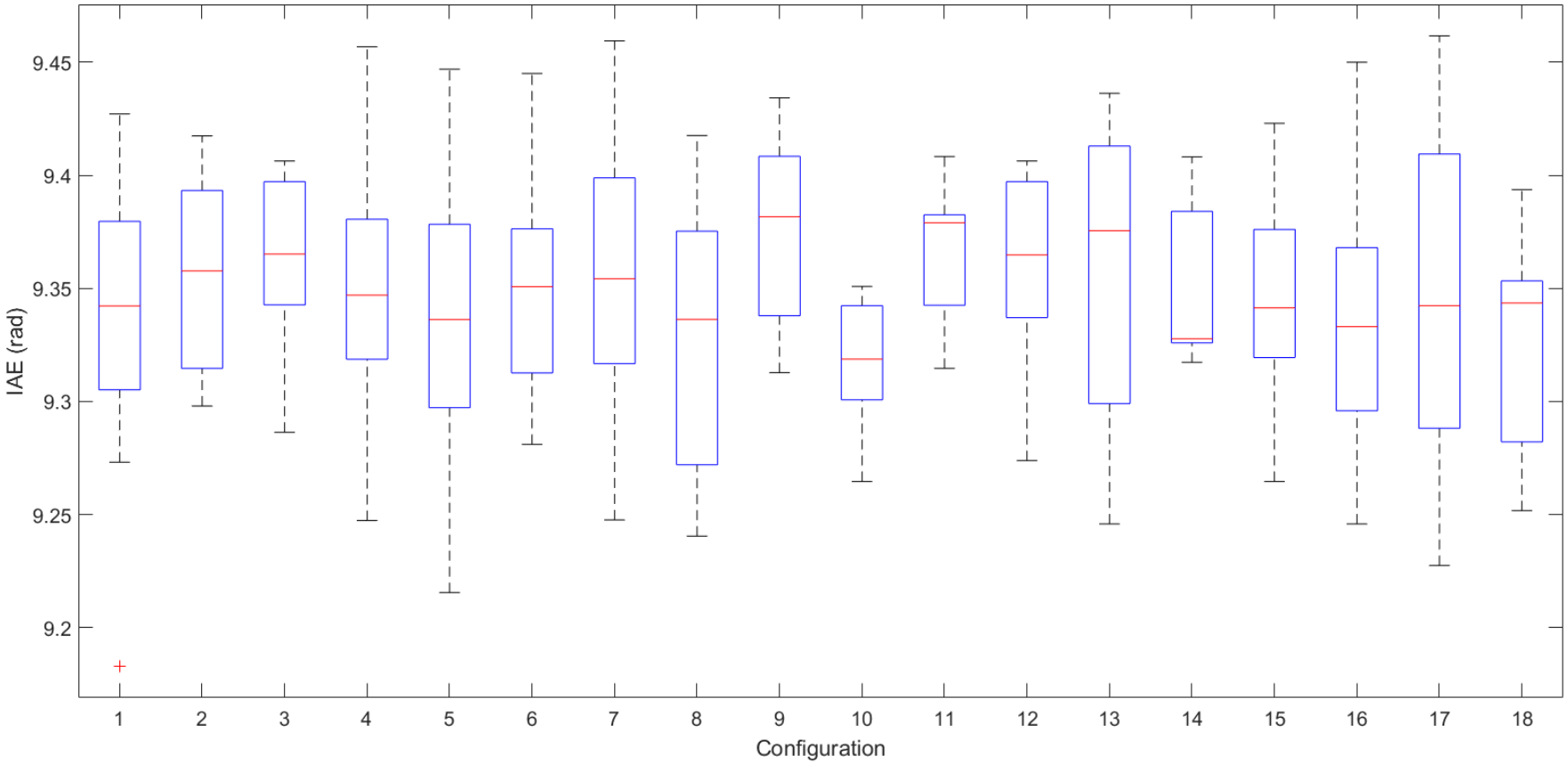


Reference: author

It is worth mentioning briefly that the normality (or not) of a sample does not indicate an advantage or disadvantage in performance for an algorithm. This is just a characteristic presented by the data, which helps one to chose the most appropriate statistical test.

ANOVA's test indicated a *p-value* = 0.7016, greater than 0.05, which means that the null hypothesis H_0 can be accepted with a confidence level of 95% and that the performance of all SA configurations was statistically equal. Figure 4.8 contains the box plot diagram referring to the SA algorithm for speed control.

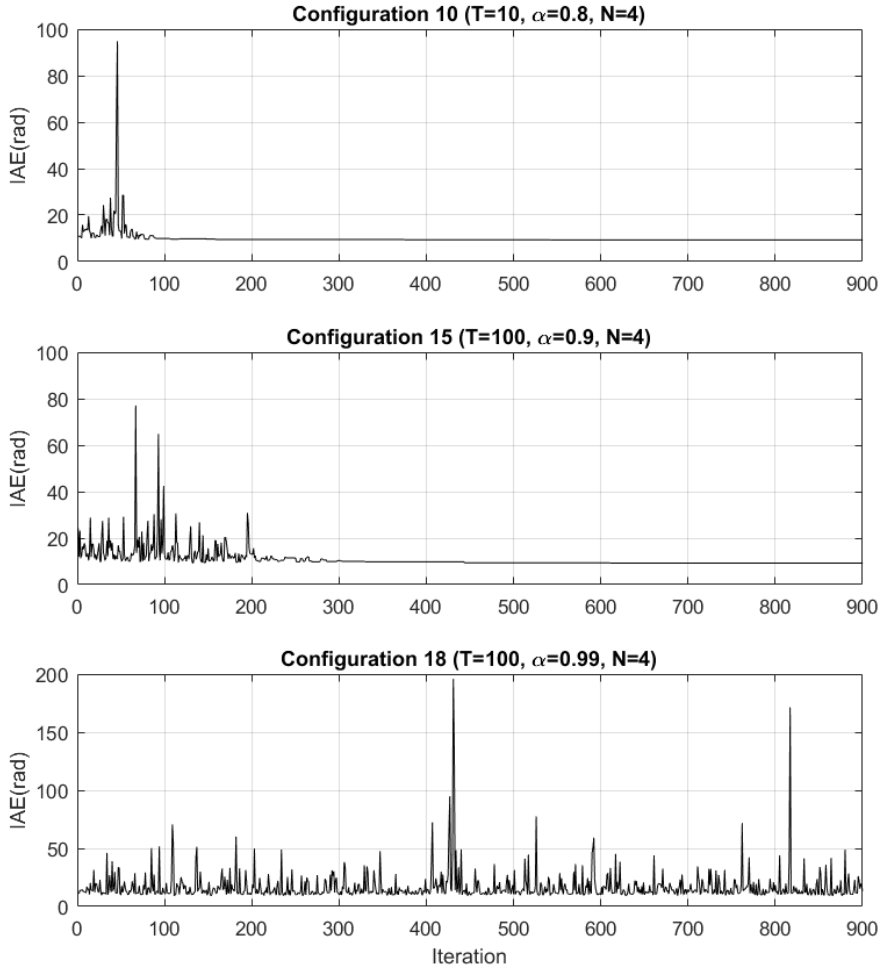
Figure 4.8 – Box plot of SA algorithm for the tuning of the controllers for the speed-controlled DC motor drive.



Reference: author

An interesting aspect about the operation of the SA algorithm is its probabilistic mechanism that allows the acceptance of worse solutions, which is directly related to the configuration of the algorithm. Figure 4.9 shows the evolution of the IAE value along the iterations for three different configurations. During the early stages of the optimisation process in the three presented examples, the value of the temperature parameter remains high. This allows the current solution $\mathbf{x}$ to accept multiple solutions with IAE evaluations that are worse than the best one previously found. This is done with the aim of exploring diverse locations within the search space. As the temperature was reduced, the algorithm became more rigorous, accepting only better solutions. The difference that can be observed between the subplots is associated with the temperature of the system, in this way, in the case of Configuration 10, the system reached a low temperature quickly, while the other configurations took longer to cool down or even reached the stop criteria with high values for T.

Figure 4.9 – Comparison of IAE value over the iterations for speed control using three SA algorithm configurations.



Reference: author

It should be noted that the SA is not a population-based algorithm and therefore, at each iteration there is only one fitness evaluation, and thus, to reach the stop criteria, 900 iterations are necessary.

4.2.3 Algorithm Comparisons

Throughout this section, 18 configurations for each algorithm were tested, each of which was repeated 20 times. Thus, for the single-objective formulation of the speed control of a DC motor drive system, a total of 720 simulations were performed. It is an amount large enough for an accurate performance comparison of the algorithms.

To carry out this comparison among the two algorithms, PSO and SA, the statistical procedures must be done again. Firstly, the normality of the data obtained was verified, however, the Shapiro-Wilk test has already been performed for both algorithms in the previous subsections, revealing the absence of normality for PSO. Therefore, the non-parametric Mann–Whitney U test will be performed directly, since the aim is to compare only two groups. As in the study of both algorithms there was no statistical difference between the configurations, to carry out the general comparison between PSO and SA, all the results obtained were put together.

Table 4.7 presents the result of the test, revealing through a *p-value* lower than 0.05, that the algorithms had statistically different performances. This difference is illustrated through the box plot diagram in Figure 4.10, where it is possible to see the advantage of PSO over SA.

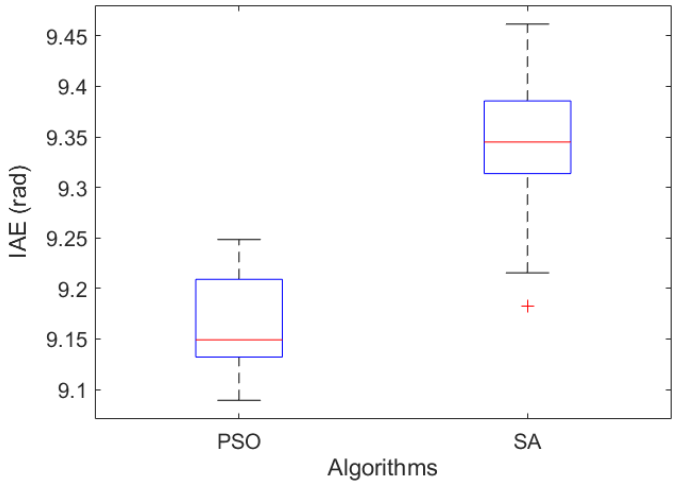
Table 4.7 – Mann–Whitney U test result - Speed control problem.

Comparison		
Algorithm 1	Algorithm 2	*p-value*
PSO	SA	<0.0001

Reference: author

Therefore, it can be concluded that the different parameter settings, in none of the cases, led to performance increase, however, in terms of results, the PSO algorithm converged to better solutions faster than the SA algorithm. It should be noted that both algorithms were able to find optimal solutions within the large search space defined for the problem.

Figure 4.10 – Box plot comparison of algorithm performance for the speed control problem.



Reference: author

4.2.4 Considerations about the Tuning Results

Regarding to the tuned gains of the controllers, the Tables 4.8 and 4.9 show the average and standard deviation of the different gain values, together with the windows of anti-windup circuits found by the algorithms considering the configurations that presented the lowest average for the integral of the absolute speed error.

Table 4.8 – Results of the tuning of the PI controllers given by the PSO.

	Variables	Mean	Std. Deviation
PI Speed Controller	$k_{p\omega}$	19.5270	1.4375
	$k_{i\omega}$	16.8027	4.6235
	dz_{ω}	10.9328	3.5829
PI Currrent Controller	k_{pi}	18.7591	2.5641
	k_{ii}	16.8429	0.5259
	dz_i	87.7725	42.1280

Reference: author

Table 4.9 – Results of the tuning of the PI controllers given by the SA.

	Variables	Mean	Std. Deviation
PI Speed Controller	$k_{p\omega}$	17.8839	1.7884
	$k_{i\omega}$	15.8954	2.9370
	dz_{ω}	8.5508	1.0246
PI Currrent Controller	k_{pi}	19.1205	0.4579
	k_{ii}	15.4997	1.0817
	dz_i	107.1778	52.3762

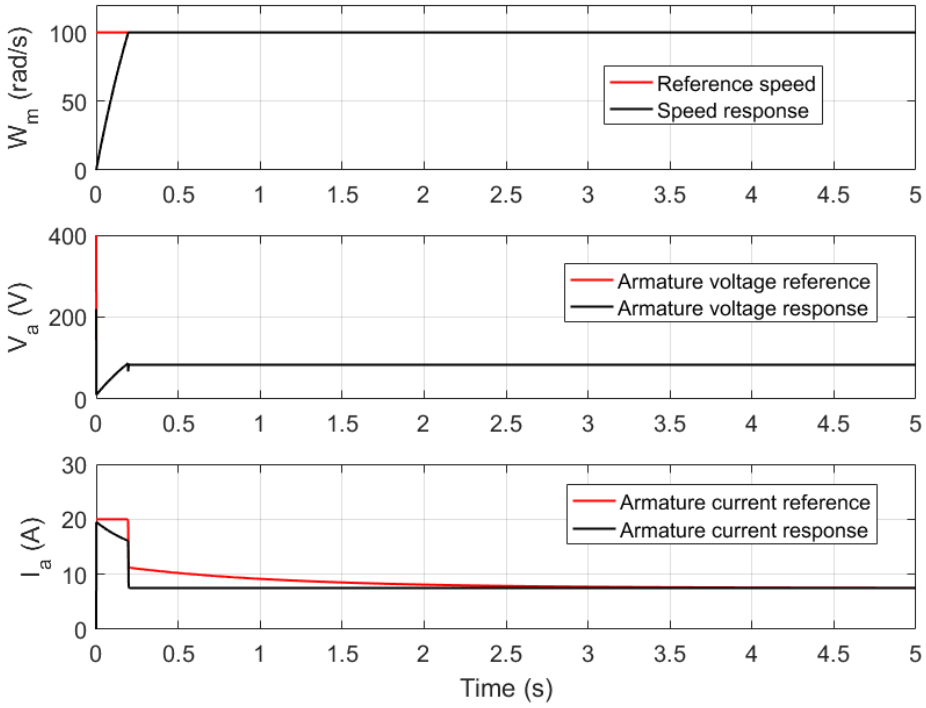
Reference: author

The results obtained by the algorithms were similar and showed that:

- From the control theory point of view, the proportional gains of the two PI regulators reached high values, what makes the dynamic responses towards the reference values faster;
- The integral gains also reached high values, since the outputs of the controllers are limited and the integrator's windup is avoided. This was also expected since high values of the integral gains are important to make the response to reach steady-state faster;
- The window of the anti-windup circuit of the PI speed controller was set up to values close of the motor current necessary to drive the load torque;
- The window of the anti-windup circuit of the PI current controller was set up to values well above the rated armature current and presented a large standard deviation. It means that the anti-windup circuit to limit the action of the PI current controller was not necessary since it was not active. It is explained since the electrical time constant of the DC motor is very small, what makes the speed and position response to be very slow when compared to the current response. Then, the limitation imposed by the saturation of the controller's output did not cause integrator's windup in the current control loop, therefore, any value for the dead-zone window could be admitted.

Since the tuning given by both algorithms, PSO and SA was similar, the behaviour of the responses of speed, current and armature voltage of the DC motor drive is also very similar, and therefore, the responses obtained through the best tuning are presented. As can be seen in Figure **??**, the speed response of the DC motor goes quickly from zero to the imposed reference speed, presenting an overshoot of only 0.87% and with low rise and settling times, respectively 0.16 (s) and 0.20 (s). The armature voltage started right at the rated value, 220 (V) but was immediately reduced to a 10 (V) level, the right one to ensure the rated armature current, 20 (A). In steady state, the armature voltage assumed the value of 83 (V), which was necessary to equal the voltage drop of the armature resistance. The current response reached the rated value at start and remained high during the acceleration. At steady-stated, settled right to the value capable of neutralising the load torque, 6.25 (A).

Figure 4.11 – DC motor speed (a), armature voltage (b) and armature current (c) with speed controller gains found by using the PSO algorithm.



4.3 Case Study II: Position Control

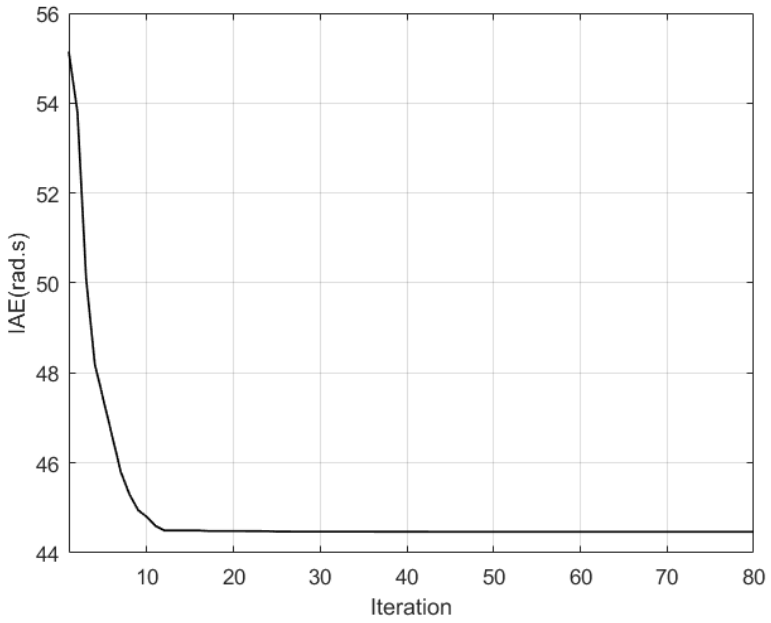
4.3.1 PSO

The methodology to find the best tuning of a set of PI controllers for the position-controlled DC motor drive is analogous to what was done for the speed. The main difference is that now there are two more variables to be tuned.

To establish the stop criteria, the PSO algorithm was exhaustively executed to determine the point of no significant improvements in the IAE objective function value. The IAE value over iterations is illustrated in Figure 4.12. A population of 100 individuals was considered, evaluating a total of 8000 solutions, resulting in 80 iterations. The figure shows that no significant improvements occur after the twelfth iteration. Therefore, for the position control problem, it was defined that the stop criteria is 1200 solution evaluations per algorithm execution. Consequently, an additional 300 fitness evaluations will be performed compared to the previous problem due to the inclusion of two tuning variables.

Table 4.10 displays the average and standard deviation results for the integral of the absolute position error of the DC motor drive, in which each algorithm configuration was repeated 20 times. The average simulation time was 747.23 (s).

Figure 4.12 – Definition of the stop criteria - Position control problem.



Reference: author

Table 4.10 – IAE value results for the PSO algorithm to the position control problem.

	Configuration			**Average (rad·s)**	**Standard Deviation (rad·s)**
	Pop. size	**A. coefficients**	**Inertia**		
1	20	$c_1 = c_2$	Linear	44.5265	0.0344
2	50	$c_1 = c_2$	Linear	44.5014	0.0249
3	100	$c_1 = c_2$	Linear	44.4838	0.0144
4	20	$c_1 > c_2$	Linear	44.5071	0.0297
5	50	$c_1 > c_2$	Linear	44.4921	0.0269
6	**100**	$\mathbf{c_1 > c_2}$	**Linear**	**44.2018**	**0.034**
7	20	$c_1 < c_2$	Linear	44.7810	0.4188
8	50	$c_1 < c_2$	Linear	44.5872	0.2212
9	100	$c_1 < c_2$	Linear	44.5144	0.0392
10	20	$c_1 = c_2$	Random	44.9781	0.3762
11	50	$c_1 = c_2$	Random	44.5331	0.0251
12	100	$c_1 = c_2$	Random	44.5344	0.0246
13	20	$c_1 > c_2$	Random	44.6126	0.2273
14	50	$c_1 > c_2$	Random	44.6215	0.2675
15	100	$c_1 > c_2$	Random	44.5246	0.0211
16	20	$c_1 < c_2$	Random	44.7061	0.3499
17	50	$c_1 < c_2$	Random	44.6096	0.2121
18	100	$c_1 < c_2$	Random	44.5184	0.0299

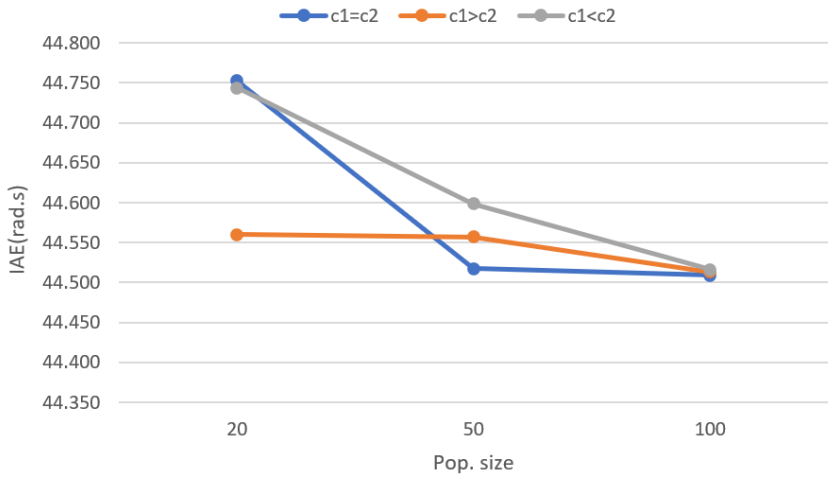
Reference: author

Configuration number 6, in bold, was the one that presented the lowest IAE mean, having the largest population size, a higher cognitive coefficient than the social one, and a linear decrease in the inertia weight. The result obtained has points in common with the configuration that presented the lowest average in the speed control problem, which was more interesting $c_1 > c_2$ and linear decreasing method, but with the difference in the population size. It is also important to highlight the low standard deviation presented by most configurations, ensuring that the stop criteria was correctly defined to the algorithm.

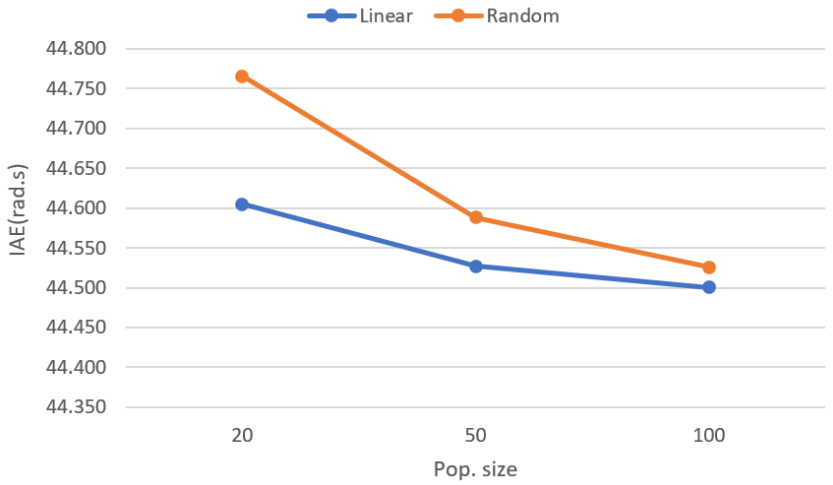
Figure 4.13 presents a comparison of the degrees of synergy among the parameters of the PSO algorithm when applied to the position control problem. Figure 4.13a shows that the population with 20 individuals performed better when combined with a higher value for c_1, while for the population with 100, the average was practically the same, regardless of the acceleration coefficients. Figure 4.13b shows the superiority of linear decreasing method over random adjustment with all population sizes. Finally, equal values of c_1 and c_2 had better results with the linear decrease of the inertia weight, however, as the population size increases, the performance difference becomes poorer.

Figure 4.13 – Interaction between PSO parameters.

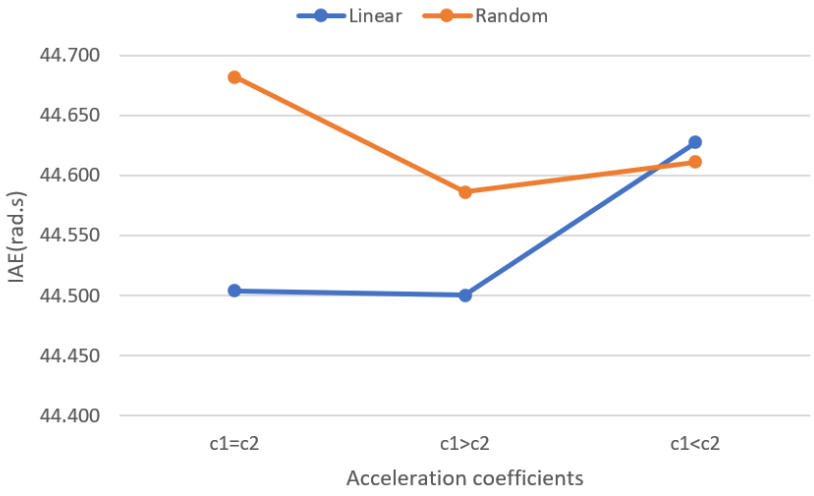
(a) Population and acceleration coefficients.



(b) Population and variation of inertia weight.



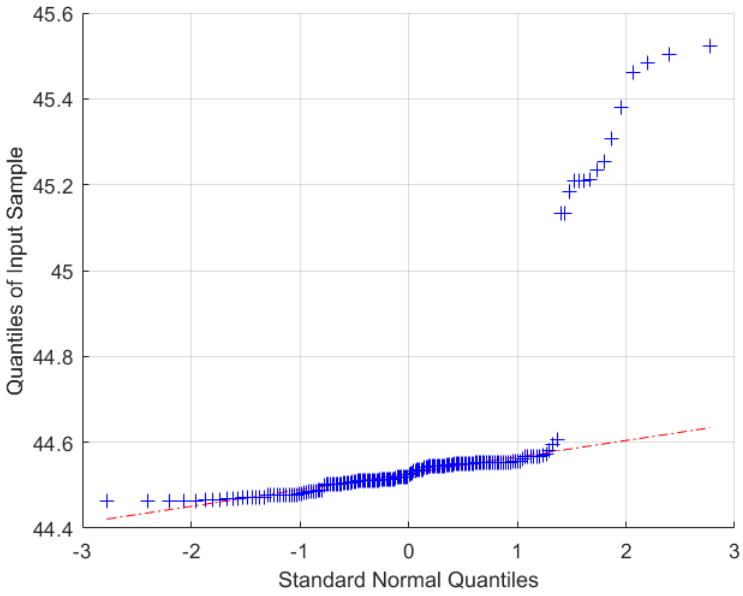
(c) Acceleration coefficients and variation of inertia weight.



Reference: author

For statistical comparison, the normality test was performed indicating a *p-value* $<$ 0.0001, which implies the non-normality of the sample data, and the need for the non-parametric Friedman test to compare the results.

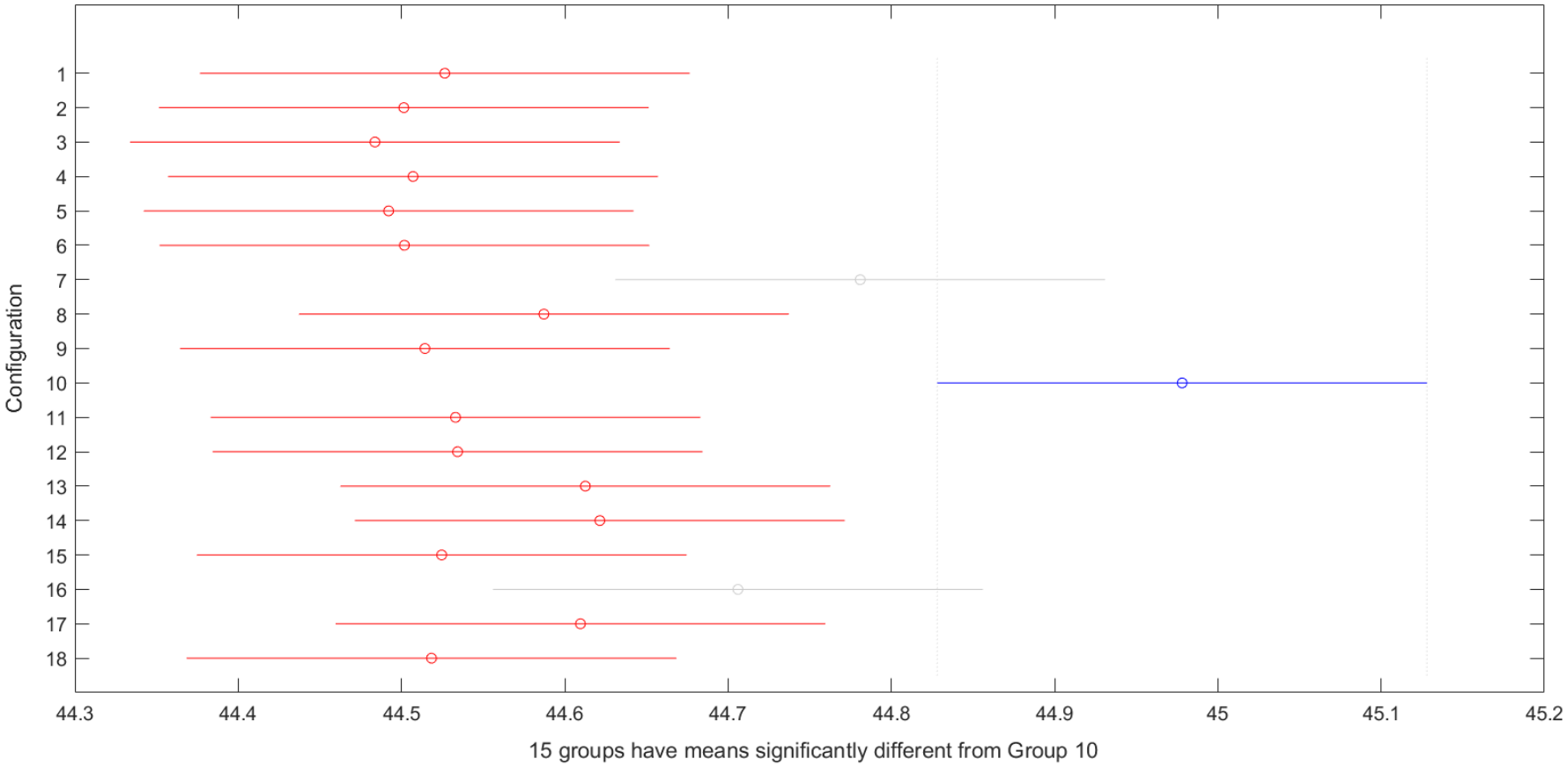
Figure 4.14 – Q-Q plot of PSO sample data *versus* standard normal- Position control problem.



Reference: author

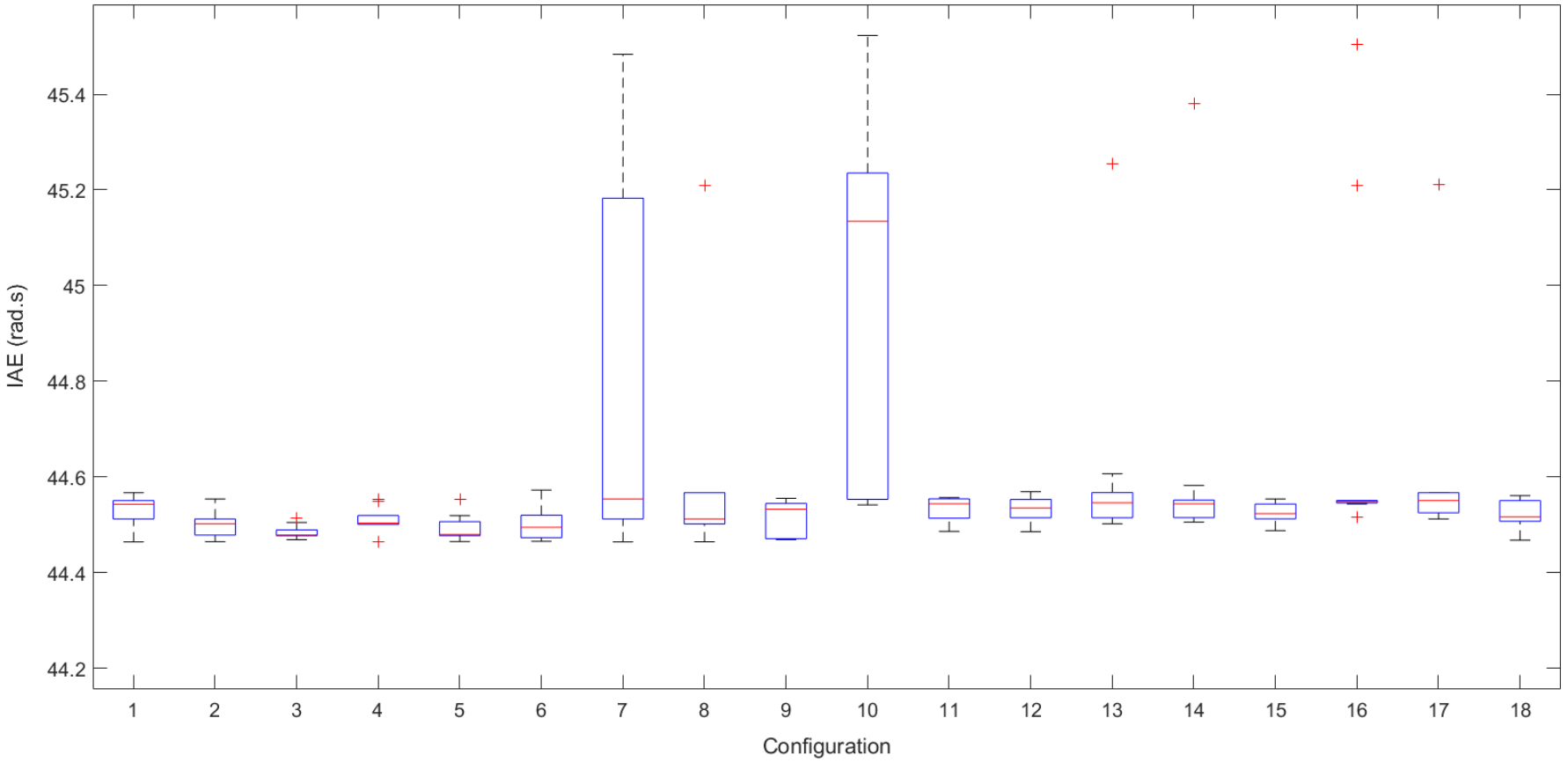
Through Friedman's test, a *p-value* $<$ 0.0001 was obtained. From this result, a difference between the configurations is observed. As the Friedman test only tells if there is a difference between the results, not specifying which one, it was necessary to perform the *post-hoc* Durbin-Conover test for multiple comparison. Through this test, illustrated in Figure 4.15, the poorer performance of configuration number 10 (population of 20 individuals, $c_1 = c_2$, and random variation for the inertia weight) in relation to all settings (in red) except numbers 7 and 16 (in grey), is evident. As it can be seen, apart from configuration number 10, all the others had performances statistically similar. Furthermore, configurations 7 and 16, even with worse averages when compared to others, are still in the region of statistical equality. Finally, the box plot of all configurations is also presented in Figure 4.16.

Figure 4.15 – *Post-hoc* analysis of the PSO algorithm for the position-controlled DC motor drive.



Reference: author

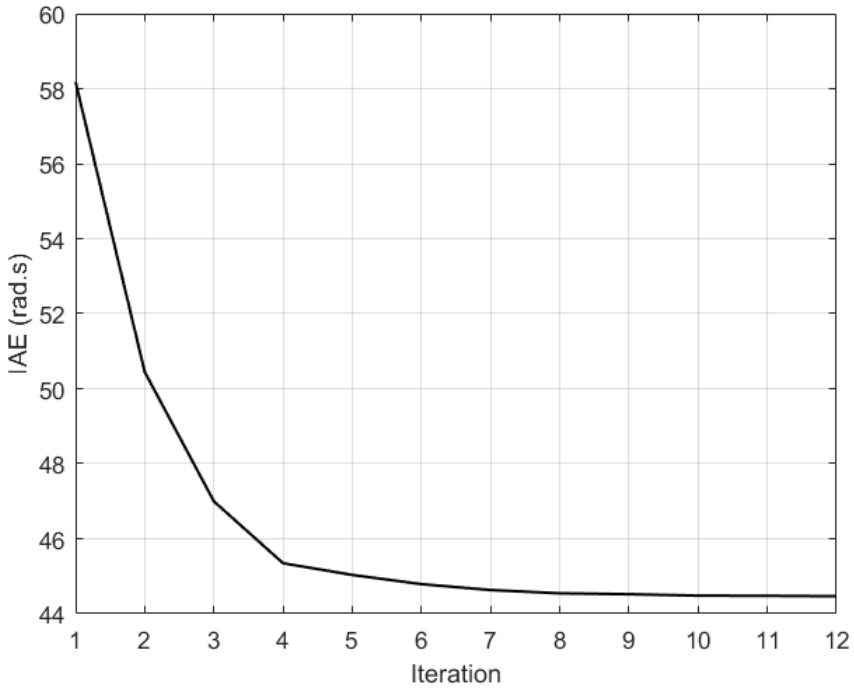
Figure 4.16 – Box plot of PSO algorithm for the tuning of the controllers for the position-controlled DC motor drive.



Reference: author

The evolution of the optimisation process is shown in Figure 4.17. This result are based on simulations performed with configuration number 6 (population size of 100 individuals, $c_1 > c_2$, and linear variation of inertia weight). It is important to note that with a population size of 100 individuals, the total number of iterations is equal to 12, resulting in 1200 fitness evaluations.

Figure 4.17 – Average of integral of the absolute position error along the iterations - PSO Algorithm.



Reference: author

4.3.2 SA

To finalise the tests involving the single-objective algorithms, simulations of the SA algorithm was used with the same parameters as proposed Table 4.5. The average simulation time per run was 718.01 (s).

The mean and standard deviation results of the IAE objective for the position-controlled DC motor are shown in Table 4.11. The configuration that presented the lowest mean IAE is shown in bold, i.e., configuration number 10 ($T_0 = 10$, $\alpha = 0.8$ and $N = 4$), which is the same one that presented the lowest mean IAE for speed control.

Table 4.11 – IAE value results for the SA algorithm to the position control problem.

	Configuration			**Average (rad·s)**	**Standard Deviation (rad·s)**
	Temperature	α	**N**		
1	10	0.8	1	47.4822	0.9235
2	50	0.8	1	47.5506	0.8878
3	100	0.8	1	47.8065	1.1886
4	10	0.9	1	47.7981	1.6175
5	50	0.9	1	47.4221	0.9935
6	100	0.9	1	47.6437	1.3046
7	10	0.99	1	48.0006	2.0467
8	50	0.99	1	47.7735	1.3162
9	100	0.99	1	47.4496	0.8468
10	**10**	**0.8**	**4**	**47.2410**	**1.3174**
11	50	0.8	4	47.6095	1.7248
12	100	0.8	4	47.2903	0.0854
13	10	0.9	4	47.7467	1.2809
14	50	0.9	4	47.9152	0.7050
15	100	0.9	4	47.5310	1.0910
16	10	0.99	4	47.5858	0.6690
17	50	0.99	4	47.6107	1.1816
18	100	0.99	4	47.8128	0.7046

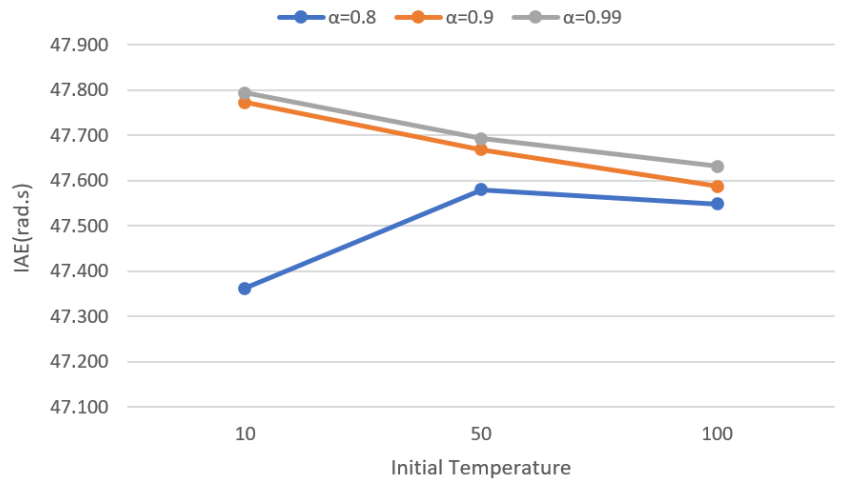
Reference: author

The synergy between the SA parameters, when applied to the position control problem, are shown in Figure 4.18. As it can be seen in Figure 4.18a, the cooling coefficient equal to 0.8 showed the lowest average IAE for all temperature values.

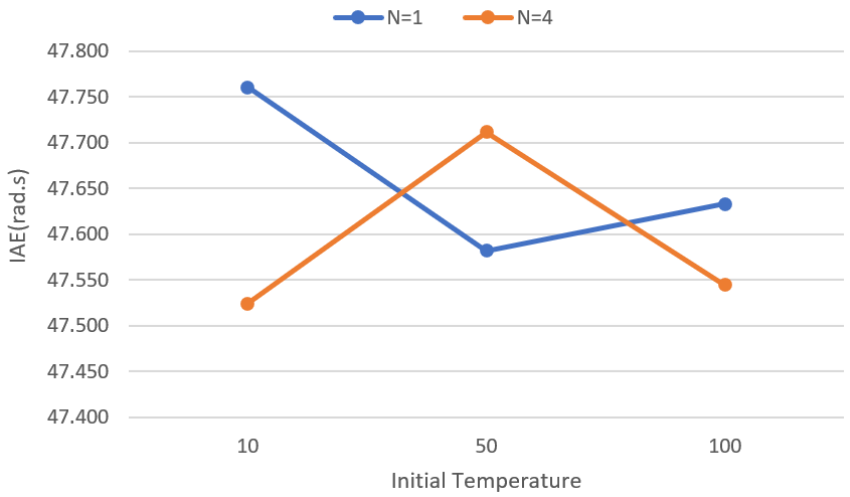
The relationship between the parameters T_0 and N is depicted in Figure 4.18b. The figure shows that the best results were achieved with $N = 4$ at both the lowest and highest temperatures. Furthermore, Figure 4.18c demonstrates that higher values of N were more beneficial for the algorithm when used in combination with $\alpha = 0.8$ and $\alpha = 0.99$. It is noteworthy that the behaviour observed in Figures 4.18b and 4.18c closely resembles that observed when applying the SA algorithm to the speed control problem. This indicates that the algorithm's dynamics remain consistent when applied to different problems.

Figure 4.18 – Interaction between SA parameters.

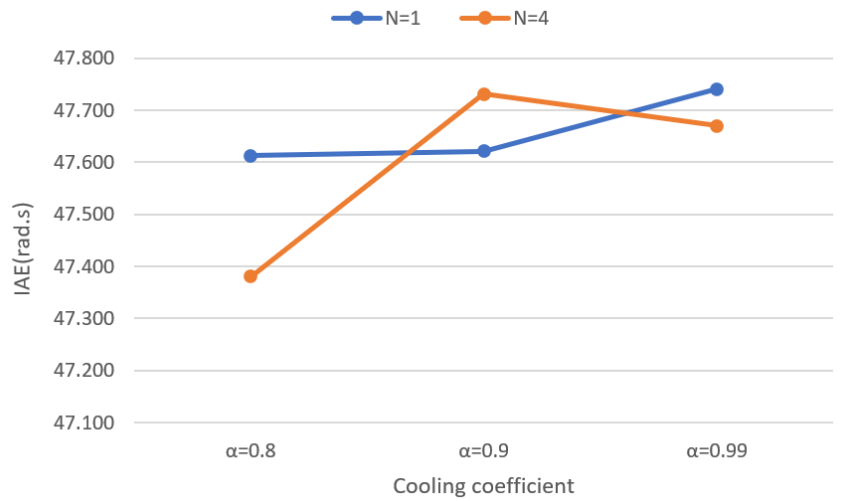
(a) Initial temperature and cooling coefficient.



(b) Initial temperature and number of sub-iterations.



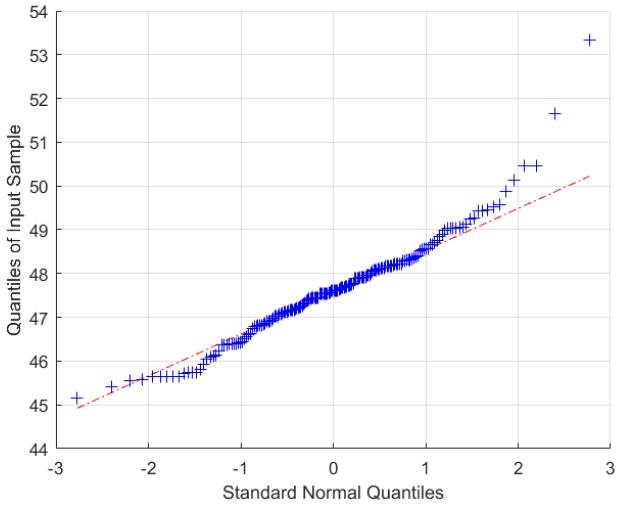
(c) Cooling coefficient and number of sub-iterations.



Reference: author

Although the points in Figure 4.19 approaches a straight line, it was not straight enough to consider that the data sample has a normal distribution, since the extremity points do not follow a linear pattern. In addition to this, through the Shapiro-Wilk test, was obtained a $p\text{-}value < 0.0001$, indicating that the non-normality assumption was valid. Thus, the non-parametric Friedman test was selected to compare the performance of the 18 configurations. The test yielded a $p\text{-}value = 0.9972$, which confirms that the different SA configurations had statistically equivalent results. This conclusion is supported by the box plot in Figure 4.21.

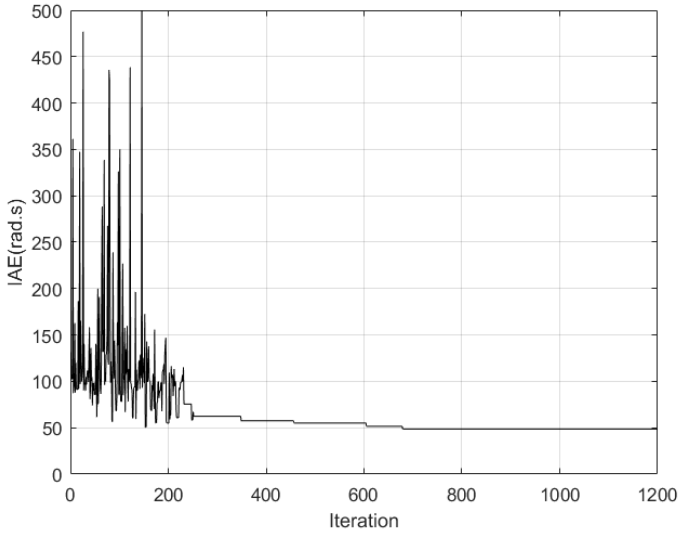
Figure 4.19 – Q-Q plot of SA sample data *versus* standard normal- Position control problem.



Reference: author

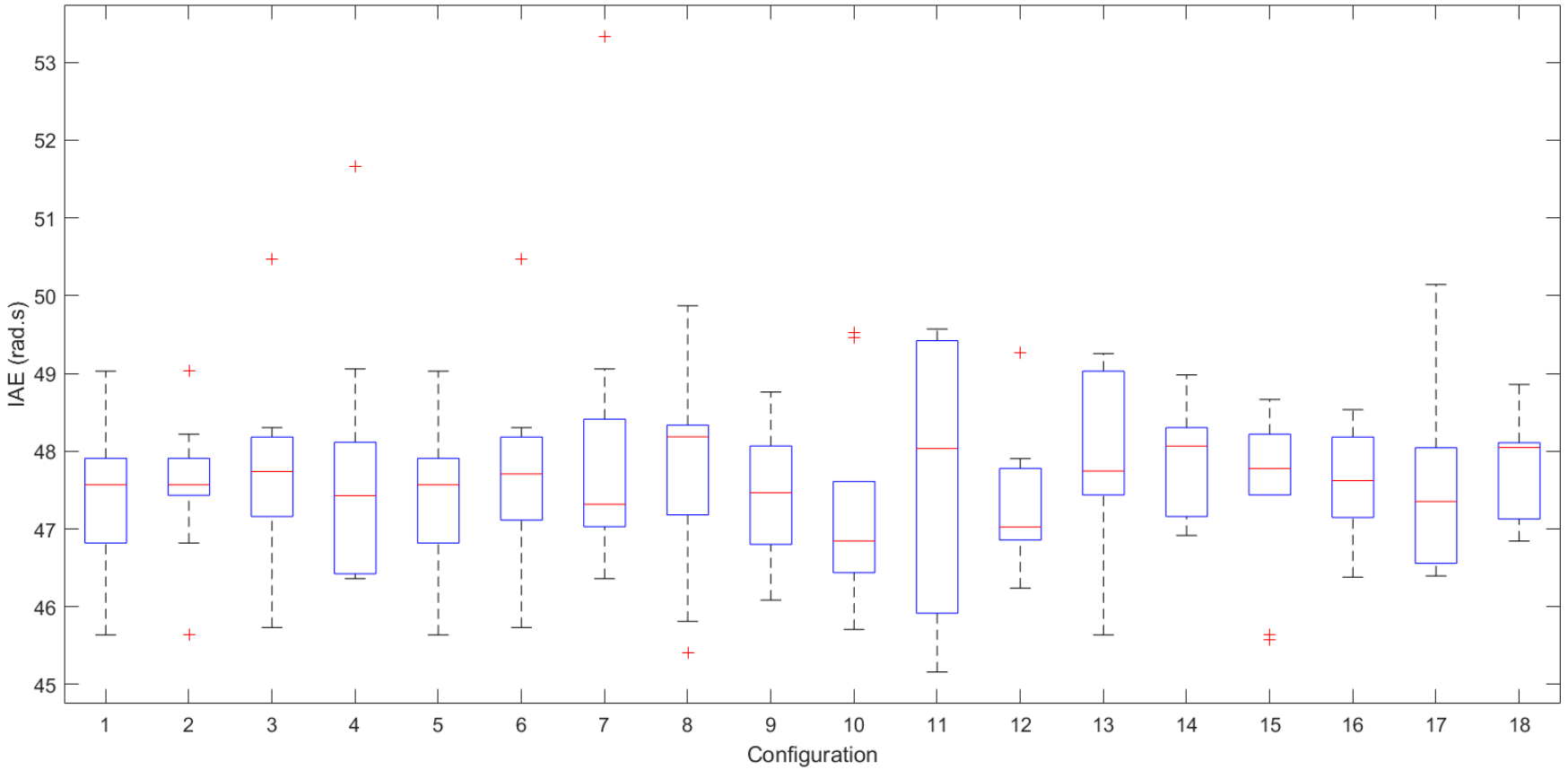
Figure 4.20 shows the value of the current solution over the iterations to configuration number 10 ($T_0 = 10$, $\alpha = 0.8$ and $N = 4$), which presented the lowest mean IAE value.

Figure 4.20 – IAE over the iterations for the position control problem - SA Algorithm.



Reference: author

Figure 4.21 – Box plot of SA algorithm for the tuning of the controllers for the position-controlled DC motor drive.



Reference: author

4.3.3 Algorithm Comparisons

It could be seen that for the position control problem, among the different configurations tested for the PSO algorithm, one had a statistically lower performance than the others.

A final algorithms comparison for a single-objective formulation was evaluated. Table 4.12 presents the result of the Mann-Whitney U test for all IAE results obtained for tuning the controllers for the position-controlled DC motor drive using the PSO and SA algorithm.

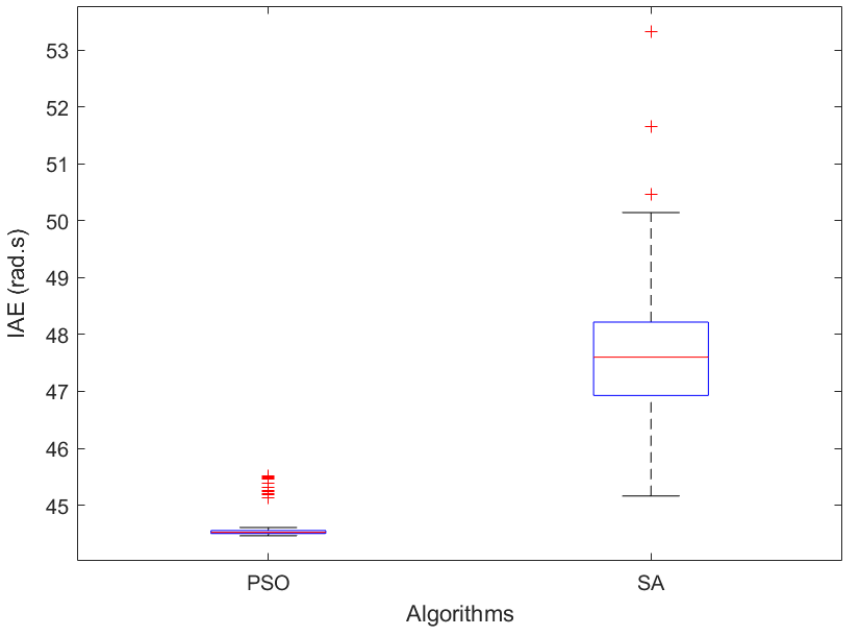
Table 4.12 – Mann–Whitney U test result - Position control problem.

Comparison		
Algorithm 1	Algorithm 2	*p-value*
PSO	SA	<0.0001

Reference: author

As with the speed control problem, the results again illustrate the superiority of the PSO algorithm, which is even more evident through the box plot diagram in Figure 4.22. Another point that draws attention is that the PSO algorithm had a very low standard deviation, equal to 0.132 (rad·s), when compared to SA (1.105 (rad·s)). The results indicate that the first algorithm had a faster convergence and greater robustness, while the second presented oscillations for the chosen stop criteria.

Figure 4.22 – Box plot comparison of algorithm performance for the position control problem.



Reference: author

4.3.4 Considerations about the Tuning Results

The mean and standard deviation of the controller's gains were computed for the configurations of the algorithms that produced the lowest mean IAE values in the position-controlled DC motor drive. The results are presented in Tables 4.13 and 4.14.

Table 4.13 – Results of the tuning of the PI controllers given by the PSO.

	Variables	Mean	Std. Deviation
PI Position Controller	k_{pp}	12.3098	0.4568
	k_{ip}	0.0000	0.0000
PI Speed Controller	$k_{p\omega}$	14.4510	2.4099
	$k_{i\omega}$	12.9662	2.7367
	dz_{ω}	9.2498	2.9597
PI Current Controller	k_{pi}	20.0000	0.0000
	k_{ii}	19.9594	0.1285
	dz_i	125.1407	54.2912

Reference: author

Table 4.14 – Results of the tuning of the PI controllers given by the SA.

	Variables	Mean	Std. Deviation
PI Position Controller	k_{pp}	14.7954	1.6766
	k_{ip}	0.0775	0.1045
PI Speed Controller	$k_{p\omega}$	15.0005	2.7847
	$k_{i\omega}$	9.4664	2.6655
	dz_{ω}	9.5830	3.0019
PI Current Controller	k_{pi}	18.3500	1.8714
	k_{ii}	18.2317	2.6338
	dz_i	123.7595	46.8445

Reference: author

The results showed that the proportional gain of the position controller reached high values, however, not high enough to cause an overshoot in the position response, which would be undesirable for the objective function. The integral gain of the position controller assumed very low values (nearly zero), since there is no limitation on the controller's output or anti-windup circuit associated with the PI position controller. Therefore, high values for the integral gain could cause overshoot and long settling times.

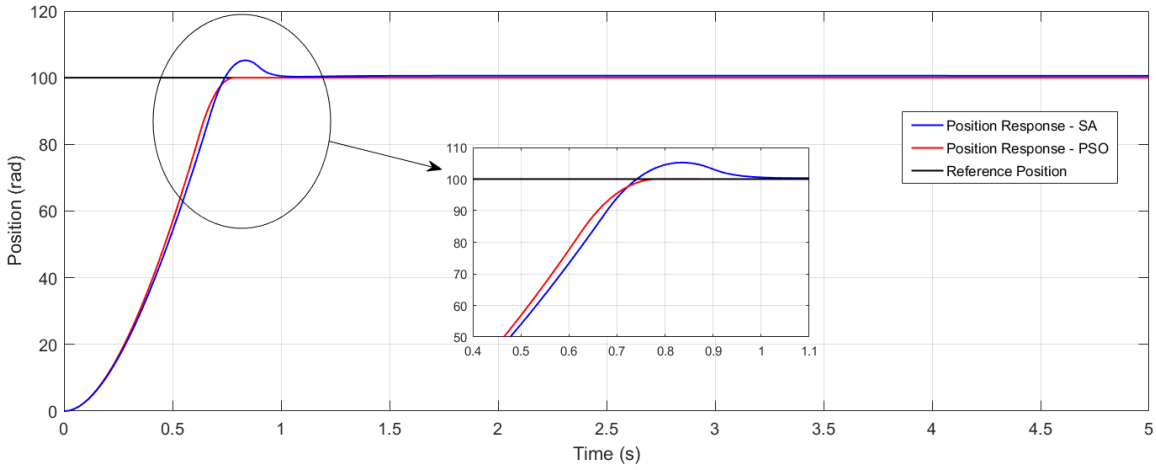
Regarding the speed and current controllers, it can be observed that:

- Proportional gains were set to high values;
- The integral gains reached high values since the outputs of the controllers are limited and the integrator's windup is avoided. This was also expected since high values of the integral gains are important to make the response to reach steady-state faster;

- The window of the anti-windup circuit of the PI current controller showed a high standard deviation, again indicating the absence of integrator's windup in the current control loop.

Figure 4.23 presents the position responses from the tuned gains provided by Tables 4.13 and 4.14, and it is possible to observe that the response associated with PSO, IAE = 44.5216 (rad·s), showed a lower value for the IAE objective than the response obtained by SA, IAE = 45.3782 (rad·s).

Figure 4.23 – Comparison of DC motor position responses with gains tuned by PSO and SA algorithms.

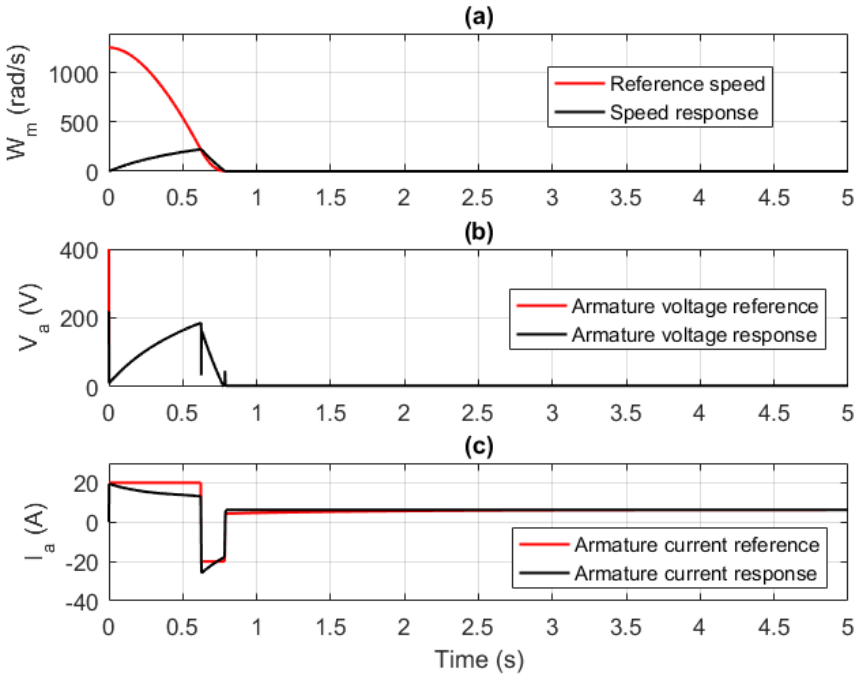


Reference: author

Finally, Figure 4.24 shows the responses of speed, voltage and armature current associated with the position responses present in Figure 4.23. As the two position responses were similar and the other system characteristics were near, it was decided to illustrate the behaviour only related to the adjustment of gains made by the PSO.

As expected, the DC motor is rapidly accelerated towards the reference position and, when reaching it, it has its speed reduced to zero. The armature voltage started right at the rated value but was immediately reduced to a 10 (V) level, the right one to ensure the rated armature current, 20 (A). As the motor speeded up towards the reference position, the armature voltage increased and, at steady state, settled right to the value high enough to equal the armature resistance voltage drop. The current response reached the rated value at start and remained high during the acceleration. At steady-stated, settled right to the value capable of neutralising the load torque.

Figure 4.24 – DC motor speed (a), armature voltage (b) and armature current (c) with controller gains found by using the PSO algorithm.



Reference: author

4.4 Conclusion

This chapter presented the effectiveness of PSO and SA algorithms in achieving optimal tuning of controllers for electrical drive problems. Both algorithms were used to optimise the gains of a set of PI controllers for a DC motor drive with speed and position control, aiming to minimise the Integral of the Absolute speed and position Error.

In terms of algorithm performance, both PSO and SA were considered with different parameter configurations, and each one repeated 20 times. From the results, it was observed that there was no statistically significant advantage of one over the other in most cases. Therefore, it was demonstrated that both algorithms were able to find optimal solutions, regardless of their configuration. However, it was observed that PSO algorithm outperformed SA, indicating that it is better suited for the problems addressed in this work.

CHAPTER 5

Results: Multi-objective Formulation

This chapter presents the results of the multi-objective formulation for optimisation problems. In Section 5.1, the selection of the objective functions that will be used is presented. Section 5.2 presents the quality indicators used for multi-objective algorithms. Section 5.3 show the results given by the NSGA-II and SPEA2 algorithms with different parameters configuration. Additional results, including the Pareto Fronts, performance comparison between algorithms, and decision-making process for speed and position control problems, are presented in sections 5.4 and 5.5.

5.1 Choice of Objective Functions

In classical control theory, four key specifications are associated with a system's response, including three that associated to the transient regime: *i*) rise time, *ii*) settling time, and *iii*) overshoot; and one related to the steady-state, which is the steady-state error. While several methods can be used to evaluate a system's response, such as the IAE used in the previous chapter, these four specifications were selected due to their ability to represent distinct features of a response. This is crucial for a multi-objective algorithm approach as these algorithms are only applicable for problems where objectives are in conflict to one another.

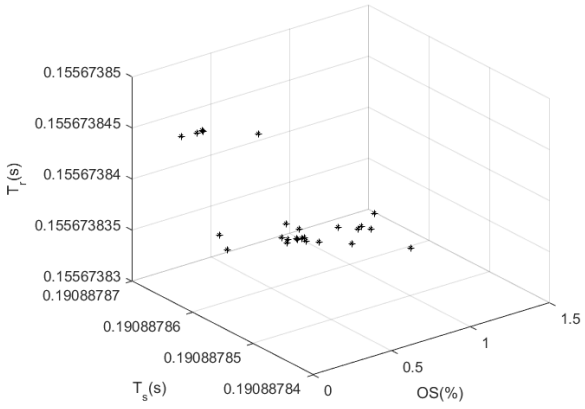
As described in Chapters 2 and 3, a multi-objective algorithm returns a set of non-dominated solutions, known as the Pareto Front, at the end of the optimisation process. This front can be visualised in the objective space. In this work, it was decided that the number of objectives for a problem would not exceed three, giving room to a

clear visualisation of the results. Then, only three objectives have been chosen to build the formulation of the speed and position control problems. Then, four classic objectives (E_{ss}, T_r, T_s and OS) were taken 3 by 3, resulting in 4 different combinations to outline the characteristics of the Pareto Fronts generated from each objective vector. By doing so, it is possible to select the set of objectives that present a higher degree of conflict.

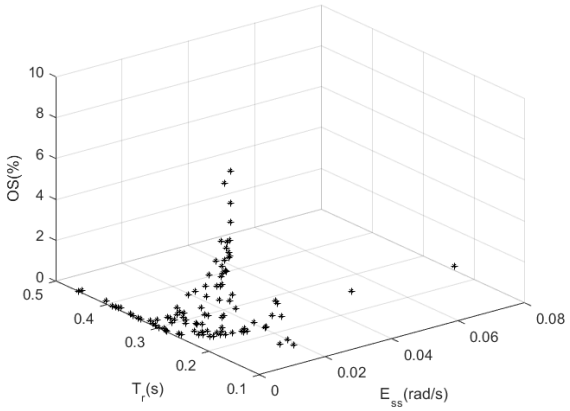
Figure 5.1 presents the four different Pareto Fronts obtained for the speed control problem of the Speed-Controlled DC Motor Drive.

Figure 5.1 – Pareto Fronts considering different combination of objectives for the speed-controlled DC Motor Drive.

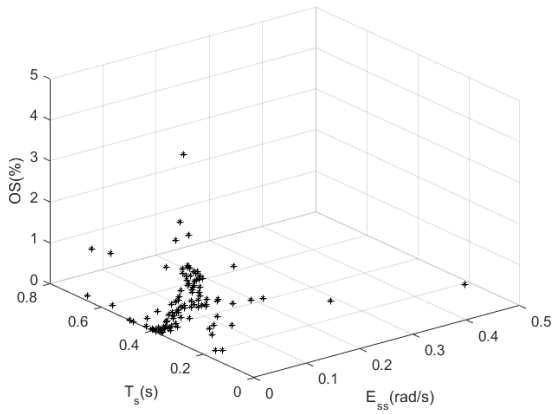
(a) Overshoot, Settling Time and Rise Time.



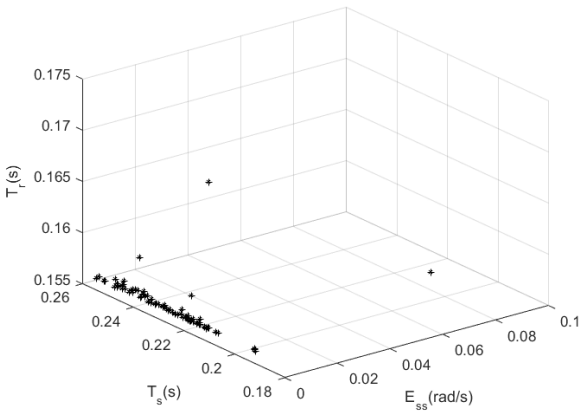
(b) Error, Rise Time and Overshoot.



(c) Error, Settling Time and Overshoot.



(d) Error, Settling Time and Rise Time.



Reference: author

The first result, shown in Figure 5.1a, took into account the minimisation of overshoot, settling time and rise time. It is evident that this combination of objectives results in a set of solutions with low diversity, located in a restricted region of the objective space, implying a low degree of conflict and that the speed responses have similar characteristics, not justifying a multi-objective approach. A similar characteristic occurred

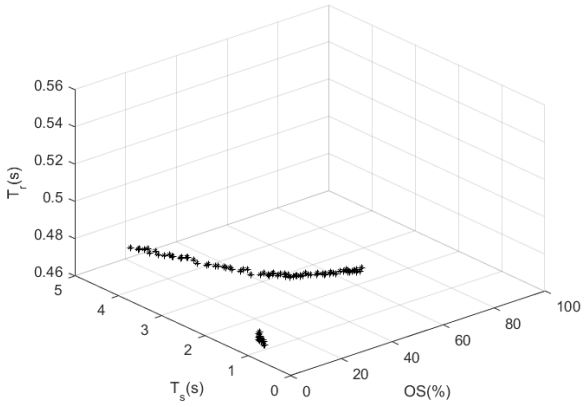
in Figure 5.1d, where although the solutions occupy a larger region, they practically do not show any variation with respect to the E_{ss} and T_r axes.

In contrast, the Pareto Fronts presented in Figures 5.1b and 5.1c displays more conflicting characteristics for the speed control problem. The vectors containing the most interesting combinations of objectives from the optimisation standpoint were those that combined the steady-state error and overshoot together with the rise and settling times, respectively.

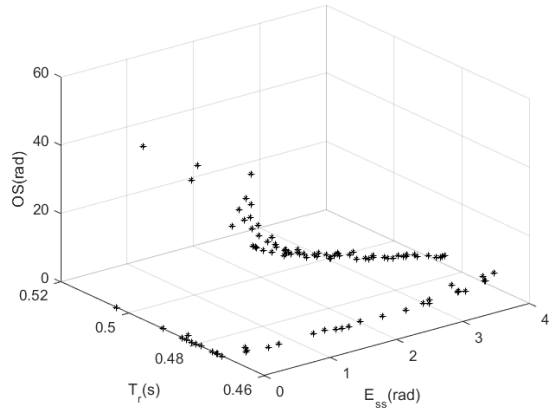
Similarly, Figure 5.2 presents the Pareto Front related to the position control of the DC motor drive. As it can be seen, despite the similarities between the position control problem and the speed control problem, the characteristics of the Pareto Fronts are distinctive.

Figure 5.2 – Pareto Fronts considering different combination of objectives for the position-controlled DC motor drive.

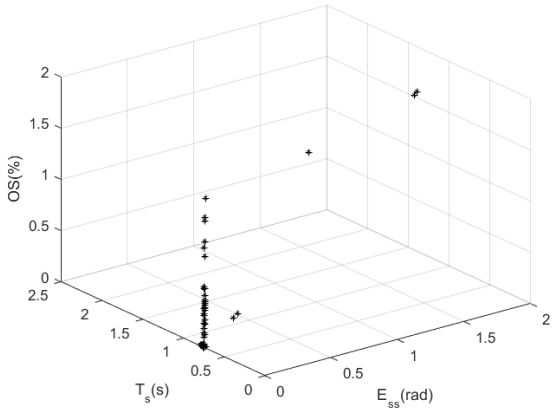
(a) Overshoot, Settling Time and Rise Time.



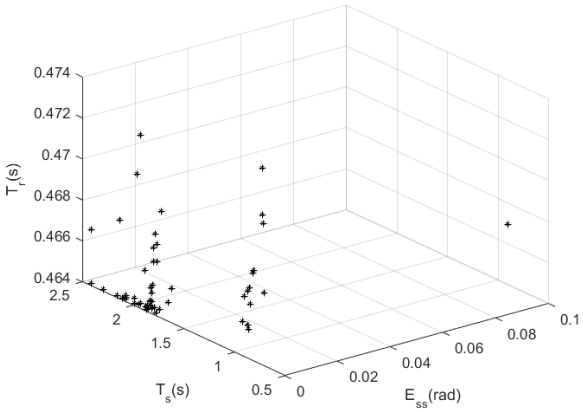
(b) Error, Rise Time and Overshoot.



(c) Error, Settling Time and Overshoot.



(d) Error, Settling Time and Rise Time.



Reference: author

Regarding the Pareto Fronts obtained for position control, two objective combinations proved to be the most interesting, as shown in Figures 5.2a and 5.2b. However, the objective set formed by E_{ss}, T_r, and OS has better characteristics because it provides a greater variety of possibilities with respect to the values of the objectives.

Based on these preliminary tests, aiming to use all four classic objectives and to choose those which provide conflicting characteristics, it was determined that for the speed control problem, the following objectives will be considered: Overshoot, Steady-state error, and Settling Time. Relatively to position control, the chosen objectives were Overshoot, Steady-state error, and Rise Time. Thus, in the next sections, the NSGA-II and SPEA2 algorithms will be applied and compared against each other, based on their respective abilities to provide good solutions for the speed and position control problems, both defined as follows:

$$\min \mathbf{F}(\mathbf{x}^{\omega}) = (E_{ss}(\mathbf{x}^{\omega}), T_s(\mathbf{x}^{\omega}), OS(\mathbf{x}^{\omega})) \tag{5.1}$$

$$\min \mathbf{F}(\mathbf{x}^{\mathbf{p}}) = (E_{ss}(\mathbf{x}^{\mathbf{p}}), T_r(\mathbf{x}^{\mathbf{p}}), OS(\mathbf{x}^{\mathbf{p}})) \tag{5.2}$$

5.2 Considerations about the Quality Indicators

The quality indicators, or metrics, described in Section 2.6 were computed from the Pareto Fronts obtained from each algorithm's run. To obtain results that enable a better comparison between the algorithms used, the normalisation of the solutions was carried out in advance, followed by the calculation of the metrics. This measure is important to standardizing the solutions represented in a three-dimensional space, allowing them to be fairly compared. Furthermore, normalisation also enables solutions to be evaluated independently without the magnitude order of each axis influencing the final evaluation.

To perform the normalisation, it is necessary to define boundary values for all objectives, for both speed control and position control problems. Thus, it was defined that the normalisation would be performed using the minimum and maximum values found for each objective, considering all the runs of the algorithms. From the result is possible to normalise the data using Equation 5.3.

$$f_j^k = \frac{f_j - f_{jmin}}{f_{jmax} - f_{jmin}} \tag{5.3}$$

where f_j^k represents the normalised value of the objective function f_j of the non-dominated solution k. The terms f_{jmin} and f_{jmax} are the minimum and maximum values found for the objective f_j.

Tables 5.1 and 5.2 display the boundary values found for each objective in the speed control and position control problems, respectively.

Table 5.1 – Limit values obtained in simulations for normalisation - Speed control problem.

OS (%)		E_{ss} (rad/s)		T_s (s)	
Min.	Max.	Min.	Max.	Min.	Max.
0.0000	15.2732	0.0000	1.6489	0.1909	2.1274

Reference: author

Table 5.2 – Limit values obtained in simulations for normalisation - Position control problem.

OS (%)		E_{ss} (rad)		T_r (s)	
Min.	Max.	Min.	Max.	Min.	Max.
0.0000	88.7548	0.0000	3.9127	0.4641	1.0612

Reference: author

The calculation of the Hypervolume (HV) was performed using the nadir point as a reference, with each coordinate of this point indicating the worst value achieved for each objective. Since this study deals with three-dimensional normalised minimisation problems, the nadir point is defined as $z_{ref} = (1, 1, 1)$. The HV indicator measures the volume between the obtained Pareto Front and the reference point, thus higher values of HV are preferable over lower ones.

In addition to the HV, two other quality indicators are used, Spread and RNI. It is important mention again that for MOPs, there is no direct way to compare the performance of the algorithms, which must be done through the combination of different quality indicators. Each of these indicators is responsible for providing different information about the analysed Pareto Front. The RNI is associated with the number of non-dominated solutions that the algorithm was able to find, but alone, this metric means nothing since the solutions found may still be poor. On the other hand, the HV says how far the whole set is from an anti-ideal point, and finally, the S indicator quantifies how well the solutions are distributed within the frontier, with $S = 0$ meaning that the solutions are equidistant. In this way, these quality indicators provide a set of information that gives room for a better evaluation and comparison of algorithms.

5.3 Algorithms Configuration

The algorithms used here have some parameters which adjustment can result in gain or loss of performance during the optimisation process. Unlike the previous chapter where algorithms with different characteristics were used, i.e., the PSO based on swarm intelligence and the SA based on physical principles, in this multi-objective formulation for the controller tuning problem, two evolutionary algorithms were selected, which implies that the set of parameters to be configured is similar for both.

The effectiveness of evolutionary algorithms is associated with three main parameters: *i*) population size; *ii*) crossover probability and *iii*) mutation probability. The mentioned parameters vary according to each problem, with no rules in the literature to be followed, although some guidelines can be found. Hassanat et al. (2019) suggests that the population size should be in the range of [50, 100] individuals. The crossover probability is commonly set to 0.9, as used in the original works of the NSGA-II and SPEA2 algorithms (DEB et al., 2002; ZITZLER; LAUMANNS; THIELE, 2001), but several values have been proposed and studied over the years, generally ranging from 0.5 to 1.0 (YE et al., 2020; DIABAT; DESKOORES, 2016). Finally, it has been shown that a mutation probability of $1/n$ (where n is the number of variables) works efficiently for problems with real-valued representation (HASSANAT et al., 2019). However, several suggestions can be found in the literature, commonly in the range [0.01, 0.5] (SILVA, 1999; CAPRARO et al., 2008).

Given the large number of possibilities for setting the parameters of these algorithms, various values were considered to verify their influence on the algorithm's performance. Based on the works cited above, Table 5.3 presents the different values and configurations assigned to each parameter. It should be noted that these values are associated with both NSGA-II and SPEA2.

Table 5.3 – Parameters of multi-objective algorithms.

Parameters	**Values**
Population size	{50, 100}
Crossover probability	{0.5, 0.9}
Mutation probability	{$1/n$, 0.5}

Reference: author

It was established that two values will be tested for each of the parameters, resulting in 8 distinct configurations. To statistically treat the obtained results, each of these configurations was repeated 20 times, with a total of 160 simulations for each algorithm applied to both, the speed control problem and the position control one. To each simulation, the associated algorithm returns a Pareto Front, which is used to compute the HV, S, and RNI metrics.

In the previous chapter, a strategy was employed to define the stop criteria for single-objective algorithms. This strategy involved performing a simulation with excessive fitness evaluations, and through the graph of the objective function value over iterations, it was possible to determine at what point there were no significant improvements. This test served as a reference for the other simulations carried out. However, for a multi-objective approach, this strategy cannot be repeated, as there are three quality indicators, and two of them are calculated after all simulations are performed, since the data is first normalised. Therefore, in this part of the work, the experience on PI controller optimisation problems was used to define the stop criteria. For the speed control problem, the number of fitness

evaluations should not exceed 3800, while for the position control problem, which has a larger number of variables to be adjusted, this number should not exceed 5000 within each run (SANTOS; SILVA; JÚNIOR, 2022). These values have proven to be satisfactory, although considered small to the Evolutionary Algorithm's standard (HUANG; LI; YAO, 2019).

Table 5.4 displays the results obtained from the NSGA-II and SPEA2 algorithms for the speed-controlled DC motor drive. For each configuration, the mean value is presented along with its standard deviation in parentheses. The values that are highlighted in bold indicate that the algorithm performed better than the other one in terms of HV, Spread, or RNI values, although not necessarily with a statistical difference. It can be seen from the results that both algorithms exhibit similar performance. After examining the average value of the HV where higher values are preferable, it can be observed that the NSGA-II and SPEA2 performed in a similar way. However, when considering the Spread quality indicator, which reflects the distribution of solutions and lower values are preferred, the situation is reversed, with SPEA2 demonstrating an advantage in 6 out of 8 configurations tested. Nevertheless, it is important to highlight that both algorithms were able to find the maximum RNI value in most cases, except for configurations 2 and 4 in the SPEA2 algorithm as shown in Table 5.4. For the speed control problem, the average simulation time was 1948.28 (s) for the NSGA-II algorithm and 1809.40 (s) for the SPEA2 algorithm.

Although the results presented in Table 5.4 may suggest some conclusions, it is important to carry out the necessary statistical tests. Six normality tests were performed, each one associated with an algorithm and a quality indicator. Since results of *p-values* below 0.0001 were obtained in all of them, it means that normal distributions were absent for the data samples showing that Friedman test must be used.

The results of the Friedman tests considering the three quality indicators (HV, Spread and RNI), are presented in Table 5.5. The *p-values* greater than 0.05 indicate that there was no statistically significant difference between the configurations of the same algorithm. The test revealed that a significant difference in performance was observed only for the SPEA2 algorithm and was associated with the S metric. In order to identify the specific configurations with different performances, a Durbin-Conover *post-hoc* test was conducted. The results of this test are presented graphically in Figure 5.3, which shows that the only significant difference in performance was observed between configurations 4 and 5. Configuration 4, which had a population size of 100, crossover probability of 0.9, and mutation probability of $1/n$, performed significantly better than configuration 5, which had a population size of 100, crossover probability of 0.5, and mutation probability of 0.5. Specifically, configuration 4 provided a Pareto Front with a better distribution within the objective space.

Table 5.4 – Results of quality indicators - Speed control problem.

	Configuration			Hypervolume		Spread		RNI	
	Population Size	**Crossover Probability**	**Mutation Probability**	**NSGA-II**	**SPEA2**	**NSGA-II**	**SPEA2**	**NSGA-II**	**SPEA2**
1	50	0.5	$1/n$	**0.9998 (0.0002)**	0.9990 (0.0021)	0.0289 (0.0131)	**0.0213 (0.0108)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
2	100	0.5	$1/n$	**0.9997 (0.0002)**	**0.9997 (0.0002)**	0.0210 (0.0150)	**0.0167 (0.0053)**	**1.0000 (0.0000)**	0.994 (0.0013)
3	50	0.9	$1/n$	**0.9998 (0.0001)**	0.9986 (0.0028)	0.0270 (0.0223)	**0.0199 (0.0115)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
4	100	0.9	$1/n$	**0.9999 (0.0001)**	0.9998 (0.0003)	0.0187 (0.0171)	**0.0111 (0.0043)**	**1.0000 (0.0000)**	0.982 (0.0057)
5	50	0.5	0.5	**0.9993 (0.008)**	0.9995 (0.0012)	**0.0191 (0.0104)**	0.0238 (0.0137)	**1.0000 (0.0000)**	**1.0000 (0.0000)**
6	100	0.5	0.5	**0.9999 (0.0001)**	**0.9999 (0.0001)**	0.0249 (0.0186)	**0.0144 (0.0029)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
7	50	0.9	0.5	**0.9994 (0.0009)**	0.9992 (0.0004)	**0.0194 (0.0072)**	0.0225 (0.0101)	**1.0000 (0.0000)**	**1.0000 (0.0000)**
8	100	0.9	0.5	**0.9999 (0.0001)**	**0.9999 (0.0001)**	0.0235 (0.0083)	**0.0138 (0.0071)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**

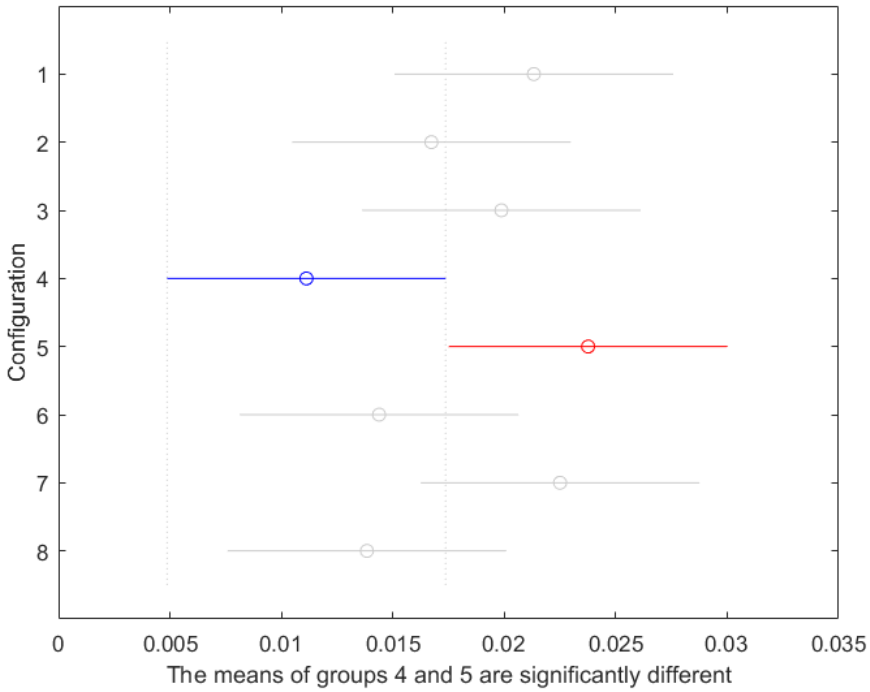
Reference: author

Table 5.5 – Results of *p-values* from the Friedman tests - Speed control problem.

	NSGA-II	**SPEA2**
HV	0.4343	0.4109
S	0.6893	0.0179
RNI	0.9999	0.9540

Reference: author

Figure 5.3 – *Post-hoc* analysis of the SPEA2 algorithm associated with Spread quality indicator for speed-controlled DC motor drive.



Reference: author

Regarding the position control problem, the algorithms were run with the same configurations previously used. The results of the three quality indicators, mean and standard deviation, are presented in Table 5.6. Unlike the results obtained for speed control, only observing the mean value of the HV indicator, there was a superiority in all configurations towards the SPEA2 algorithm. A similar behaviour is also observed for the Spread metric, where 6 out the 8 configurations also expressed an advantage of the SPEA2 algorithm. With regards to the RNI metric, there was a tie between the algorithms, with the maximum value for the metric obtained in almost all configurations, with only two exceptions that are meaningless given the number of repetitions performed. For the position control problem, the average simulation time was 3051.25 (s) for the NSGA-II algorithm and 2903.48 (s) for the SPEA2 algorithm.

Table 5.6 – Results of quality indicators - Position control problem.

	Configuration			Hypervolume		Spread		RNI	
	Pop. Size	**Crossover Probability**	**Mutation Probability**	**NSGA-II**	**SPEA2**	**NSGA-II**	**SPEA2**	**NSGA-II**	**SPEA2**
1	50	0.5	$1/n$	0.9975 (0.0021)	**0.9987 (0.0007)**	0.0318 (0.0080)	**0.0313 (0.0107)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
2	100	0.5	$1/n$	0.9977 (0.0018)	**0.9992 (0.0004)**	0.0256 (0.0083)	**0.0193 (0.0054)**	**1.0000 (0.0000)**	0.9990 (0.0032)
3	50	0.9	$1/n$	0.9973 (0.0013)	**0.9988 (0.0011)**	0.0307 (0.0054)	**0.0280 (0.0119)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
4	100	0.9	$1/n$	0.9984 (0.0008)	**0.9995 (0.0001)**	0.0495 (0.0455)	**0.0188 (0.0045)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
5	50	0.5	0.5	0.9979 (0.0017)	**0.9982 (0.0031)**	0.0366 (0.0307)	**0.0351 (0.0095)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
6	100	0.5	0.5	0.9992 (0.0003)	**0.9996 (0.0001)**	**0.0208 (0.0054)**	0.0252 (0.0113)	0.9920 (0.0253)	**1.0000 (0.0000)**
7	50	0.9	0.5	0.9976 (0.0016)	**0.9990 (0.0007)**	0.0360 (0.0080)	**0.0300 (0.0094)**	**1.0000 (0.0000)**	**1.0000 (0.0000)**
8	100	0.9	0.5	0.9989 (0.0002)	**0.9991 (0.0016)**	**0.0185 (0.0034)**	0.0334 (0.0277)	**1,0000 (0.0000)**	**1.0000 (0.0000)**

Reference: author

For the statistical treatment of the results, the Shapiro-Wilk normality test was used and a *p-value* less than 0.0001 was obtained, indicating the non-normality of all data samples. As a result, the Friedman test was again selected to compare the performance of the 8 tested configurations for each algorithm. The results are presented in Table 5.7.

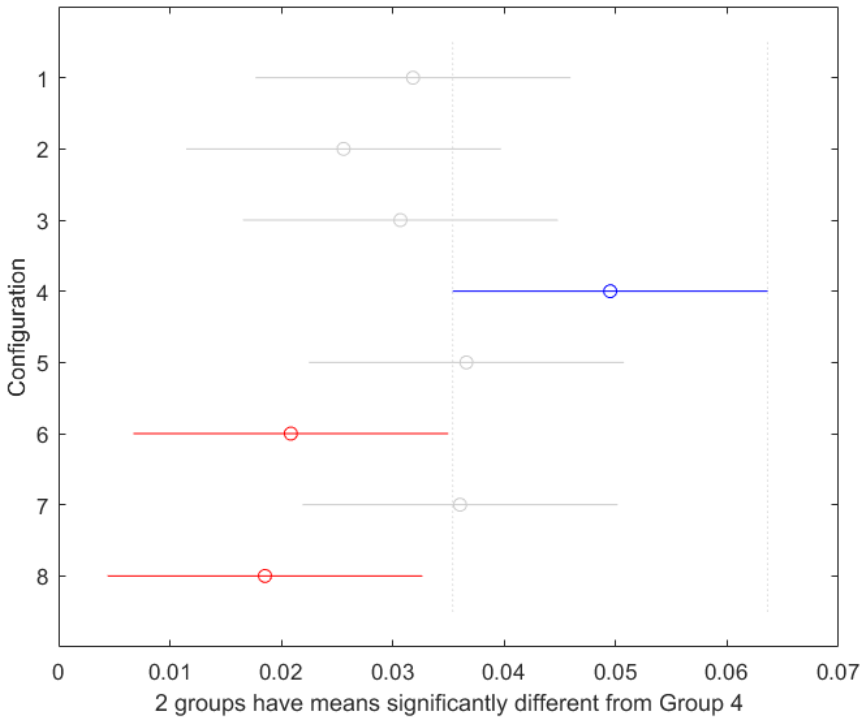
Table 5.7 – Results of *p-values* from the Friedman tests - Position control problem.

	NSGA-II	**SPEA2**
HV	0.1454	0.2562
S	0.0291	0.0012
RNI	0.8850	0.9172

Reference: author

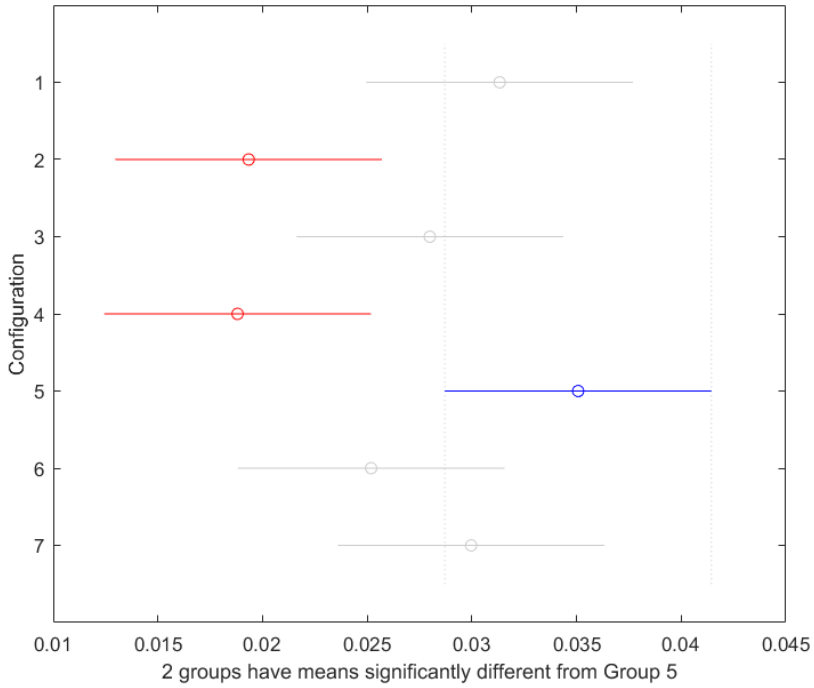
Similar to the speed control problem, there was no statistical difference between the configurations in terms of HV and RNI metrics, but statistically divergent results were obtained for both algorithms with regards to the Spread metric. To verify the existing differences, the results of the Durbin-Conover tests are presented in Figures 5.4 and 5.5.

Figure 5.4 – *Post-hoc* analysis of NSGA-II algorithm with Spread quality indicator for position-controlled DC motor drive.



Reference: author

Figure 5.5 – *Post-hoc* analysis of the SPEA2 algorithm with Spread quality indicator for position-controlled DC motor drive.



Reference: author

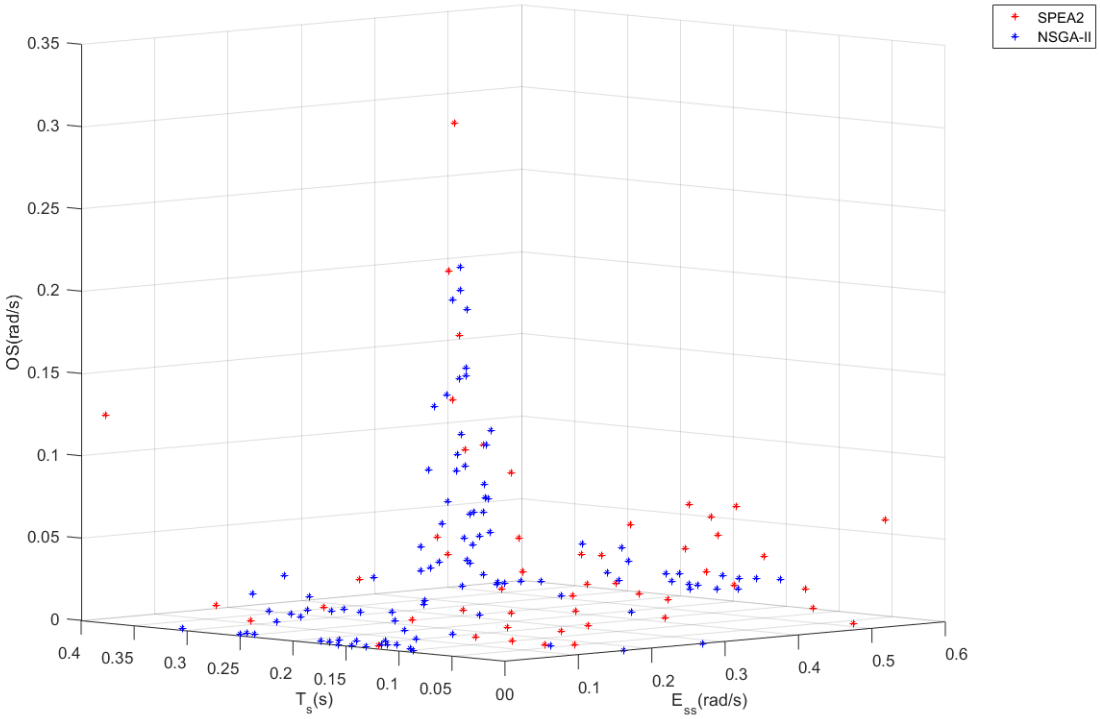
The *post-hoc* test associated with the NSGA-II algorithm revealed that configuration number 4 ($N_P = 100, p_c = 0.9$ and $p_m = 1/n$) was worse in terms of the Pareto Fronts distribution when compared to configurations 6 ($N_P = 100, p_c = 0.5$ and $p_m = 0.5$) and 8 ($N_P = 100, p_c = 0.9$ and $p_m = 0.5$). With regards to the SPEA2 algorithm, configuration number 5 ($N_P = 50, p_c = 0.5$ and $p_m = 0.5$) was worse when compared to configurations 2 ($N_P = 100, p_c = 0.5$ and $p_m = 1/n$) and 4 ($N_P = 100, p_c = 0.9$ and $p_m = 1/n$).

It should be emphasised that throughout this section, comparisons were made only involving the performance between different parameter setting of the same algorithm. Later, in sections 5.4 and 5.5, statistical comparisons between the algorithms will be presented.

5.4 Case Study I: Speed Control

The Figure 5.6 shows the Pareto Fronts obtained by the NSGA-II and SPEA2 algorithms according to configuration 4 ($N_P = 100, p_c = 0.9$ and $p_m = 1/n$), for the speed control problem. A similarity between the occupied regions of the fronts in the normalised objectives space can be observed, a behaviour that is in accordance with the values obtained by the quality indicators.

Figure 5.6 – Pareto Fronts generated by NSGA-II and SPEA2 algorithms for speed control problem.



Reference: author

Table 5.8 displays the average results obtained for the quality indicators HV, Spread, and RNI, considering all tested configurations for each algorithm. The standard deviation for each result is indicated in parentheses below its corresponding mean value.

Table 5.8 – Quality indicators results - Speed control problem.

Algorithms	HV	S	RNI
NSGA-II	**0.9998 (0.0005)**	0.0228 (0.0146)	**1.0000 (0.0000)**
SPEA2	0.9995 (0.0015)	**0.0179 (0.0096)**	0.997 (0.0206)

Reference: author

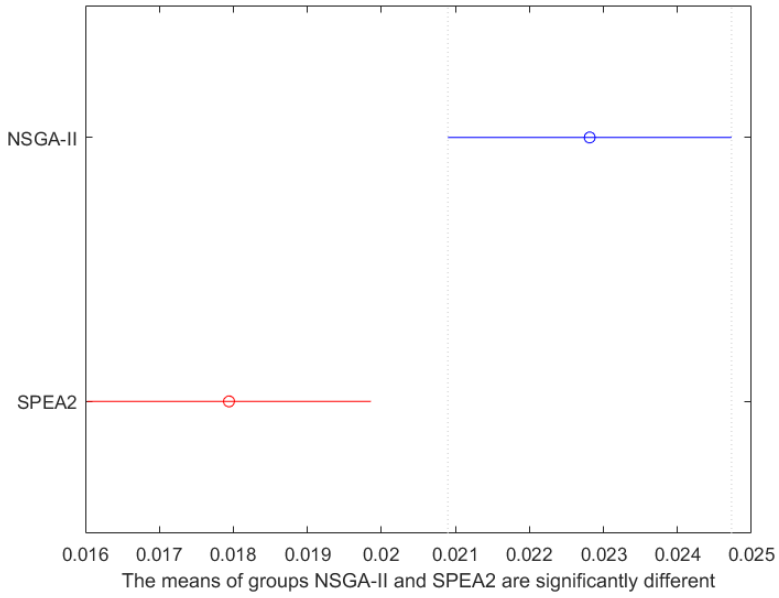
To make a statement whether there was superiority of one of the algorithms, a statistical treatment of the data was performed. Since the Shapiro-Wilk test had already indicated the non-normality of the data, the Mann-Whitney U test was used to compare the two algorithms. Thus, the test was performed three times, comparing both algorithms for each of the metrics, and the results are shown in Table 5.9. The test results indicate that in terms of HV and RNI quality indicators, both algorithms have shown statistically same performances. However, with regard to the S metric, the SPEA2 one was superior, as illustrated in Figure 5.7.

Table 5.9 – Mann-Whitney U test results - Speed control problem.

Comparison between NSGA-II and SPEA2	
Quality Indicator	*p-value*
HV	0.0725
S	0.0138
RNI	0.1943

Reference: author

Figure 5.7 – Comparison of results of NSGA-II and SPEA2 algorithms related to the quality indicator S - Speed control problem.



Reference: author

5.4.1 Decision Making

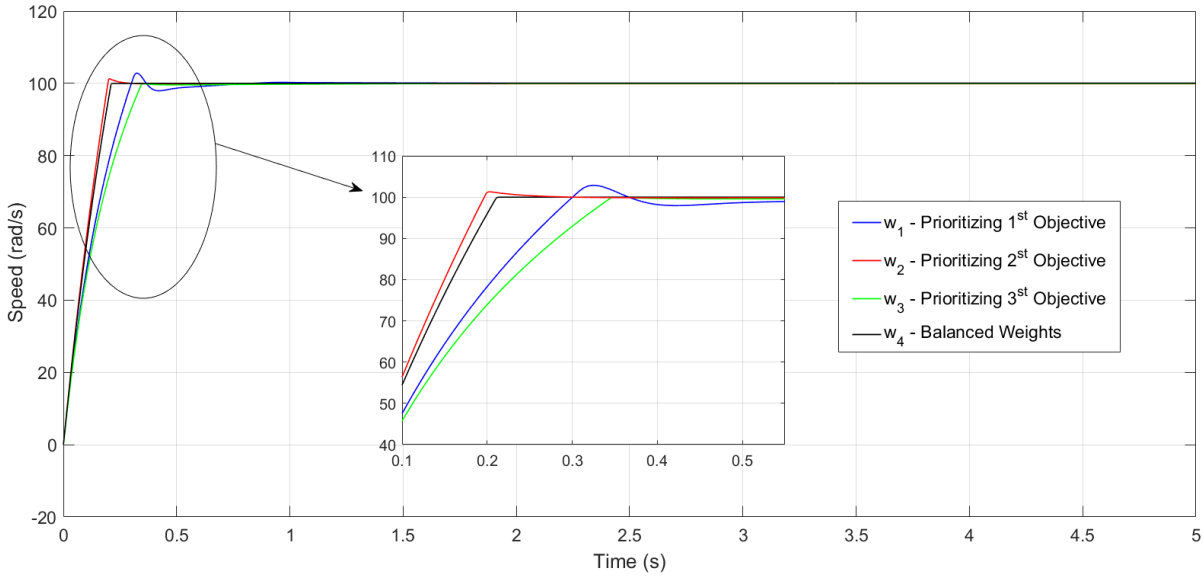
Due to conflicting objectives, each solution in the Pareto Front has a unique characteristic, resulting in different gain values for the PI controllers and the parameters associated with the anti-windup circuits. This leads to differing speed responses, directly related to the location of solutions within the objective space.

In a multi-objective problem formulation, it is common to have a moment when a decision must be made in terms of which weights are established for each objective and the best solution is chosen based on the application. The WSM method (described in section 3.5) was used to visualise different speed responses for the controlled DC motor. For each normalised objective (E_{ss}, T_s and OS), an importance weight is assigned, forming the vector $w = (w_1, w_2, w_3)$.

Four weight vectors were defined: $w_1 = (1, 0, 0)$, $w_2 = (0, 1, 0)$, $w_3 = (0, 0, 1)$ and $w_4 = (1/3, 1/3, 1/3)$. Weights vectors w_1, w_2 and w_3 prioritise only one objective at a time,

while w_4 means that all three objectives have equal significance. Vector w_1 means that minimising the steady-state error is more important; w_2, the settling time whereas w_3 means that, the smaller overshoot, the better. Because of this, four different types of response are obtained as shown in Figure 5.8. It is important to note that the different responses presented are related to only one simulation of the SPEA2 algorithm that had good characteristics with respect to evaluation metrics, being chosen only to illustrate some characteristics of multi-objective problems.

Figure 5.8 – Speed responses for different weight vectors.



Reference: author

As expected, the speed response changes when different objective is prioritised. The response with the faster settling time displays less overshoot whereas the one with the smallest stead-state error presents bigger overshoot and long settling time. According to the trade-off between all the objectives, the speed response that complies with the objectives having the same degree of importance, w_4, is more appropriate when the fast speed control with minimal steady-state error and overshoot matters.

Table 5.10 displays the different values of the PI controller's gains found by the SPEA2 algorithm, together with the anti-windup circuits window, which generated the speed responses shown in Figure 5.8.

Table 5.10 – Results of the tuning of the PI controllers - Speed control problem.

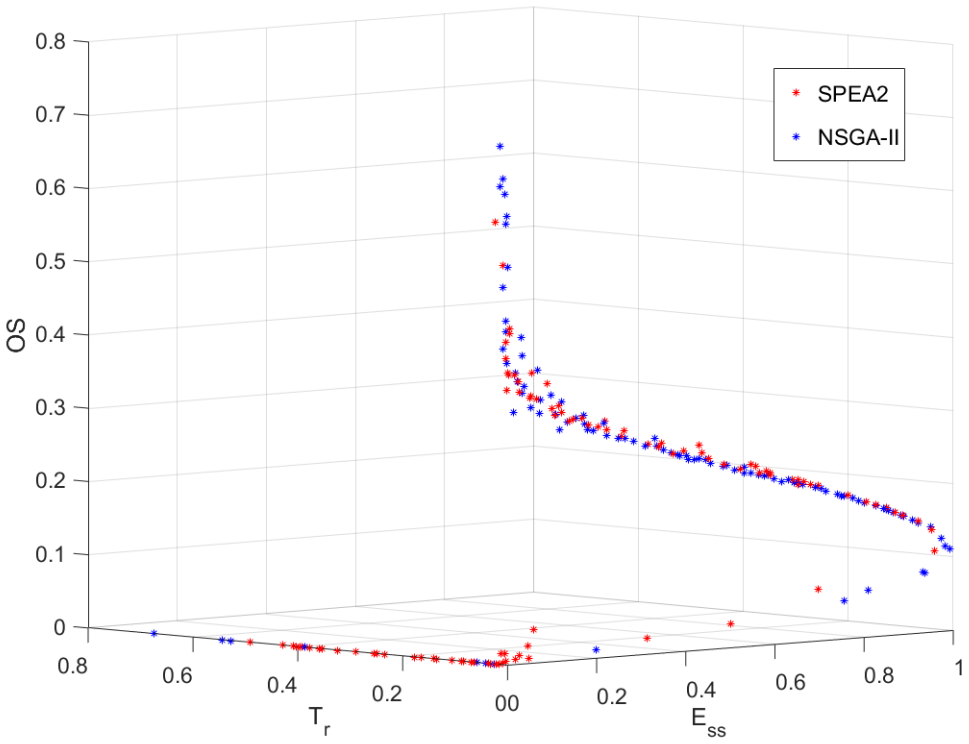
	Variables	w_1	w_2	w_3	w_4
Speed Controller	$k_{p\omega}$	1.4992	14.1185	13.0929	14.4042
	$k_{i\omega}$	18.5622	6.2892	2.1067	2.7818
	dz_{ω}	11.9174	7.6886	12.5946	9.0746
Current Controller	k_{pi}	6.8058	20.0000	18.0826	17.0799
	k_{ii}	13.5622	18.9716	12.8567	11.9865
	dz_i	91.85318	87.9296	114.3284	118.7233

Reference: author

5.5 Case Study II: Position Control

The Pareto Fronts generated by the NSGA-II and SPEA2 algorithms for position control are presented in Figure 5.9. As it can be seen, the results provided by both algorithms display similar characteristics. However, differently from the problem of tuning the controllers for the speed-controlled DC motor drive, the Pareto Front displays a more interesting curve pattern within the objective space.

Figure 5.9 – Pareto Fronts generated by NSGA-II and SPEA2 algorithms for position control.



Reference: author

The mean and standard deviation values of the quality indicators employed are presented in Table 5.11, suggesting a superior performance of the SPEA2 algorithm over the

NSGA-II. Therefore, statistical tests were performed to analyse the data. Due to the non-normality of the sample data, the Mann-Whitney U test was applied considering the three quality indicators, and the results are presented in Table 5.12. The hypervolume *p-value* of less than 0.0001 suggests that the algorithms had statistically different performances, as depicted in Figure 5.10. However, for the Spread and RNI metrics, the algorithms had the same performance.

Table 5.11 – Quality indicators results - Position control problem.

Algorithms	HV	S	RNI
NSGA-II	0.9979 (0.014)	0.0312 (0.0215)	0.9990 (0.089)
SPEA2	**0.9990 (0.013)**	**0.0268 (0.0106)**	**0.9999 (0.0011)**

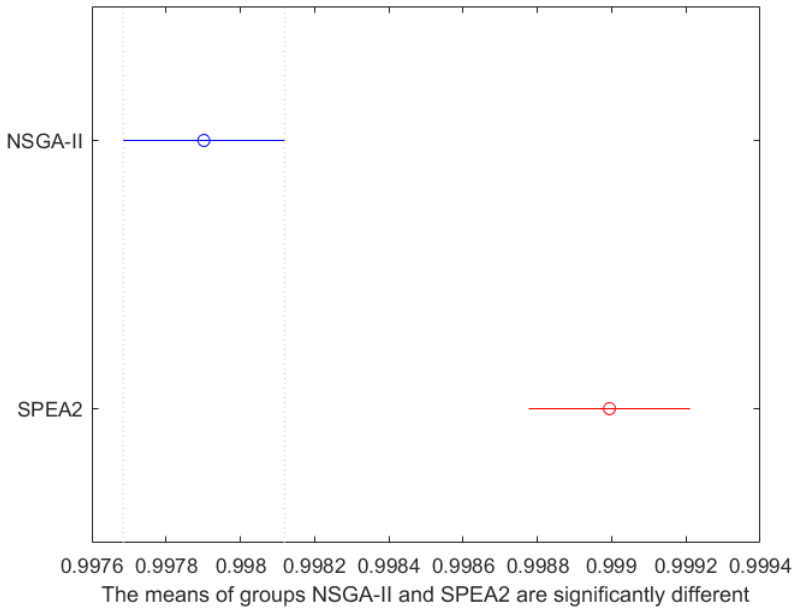
Reference: author

Table 5.12 – Mann-Whitney U test results - Position control problem.

Comparison between NSGA-II and SPEA2.	
Quality indicator	*p-value*
HV	<0.0001
S	0.2252
RNI	0.3866

Reference: author

Figure 5.10 – Comparison of results of NSGA-II and SPEA2 algorithms related to the quality indicator HV - Position control problem.

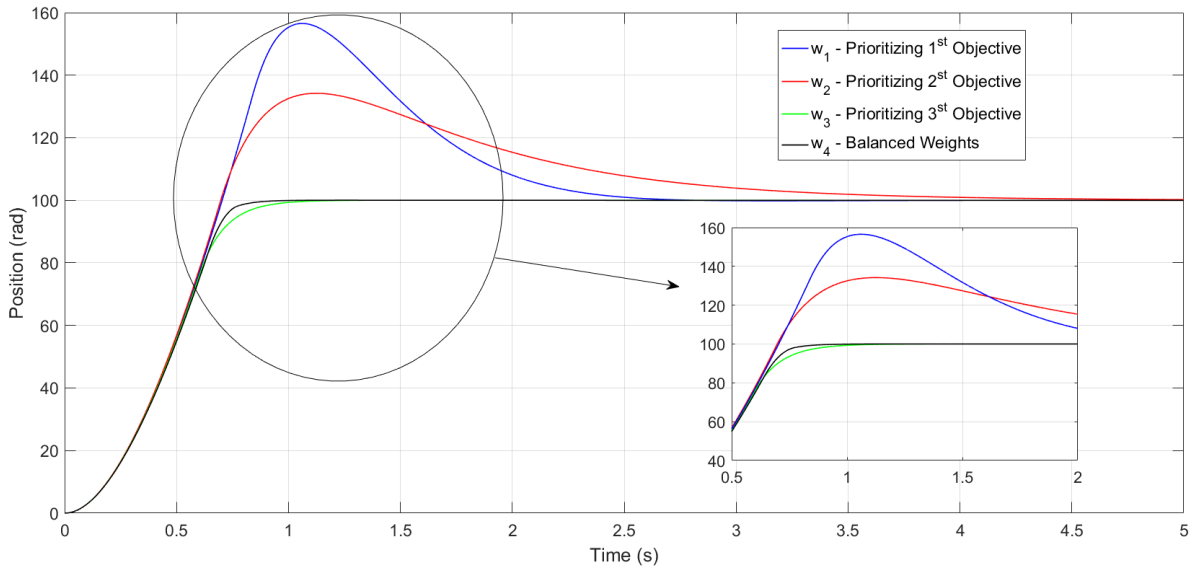


Reference: author

5.5.1 Decision Making

The same strategy of defining four weight vectors, referred to as $w_1 = (1, 0, 0)$, $w_2 = (0, 1, 0)$, $w_3 = (0, 0, 1)$, and $w_4 = (1/3, 1/3, 1/3)$, was again used in position-controlled electric drive. Figure 5.11 shows the different response behaviours associated with each weight vector. The blue response had the smallest steady-state error but exhibited an overshoot of approximately 58%. The response that prioritised rise time also had a high overshoot and the largest steady-state error among the four responses. The green response, associated with the smallest overshoot, exhibited critically damped characteristics and took longer to achieved the reference value. Finally, the response associated with the vector of balanced weights provided good characteristics associated with the chosen objectives, without overshoot, with a low rise time, and small steady-state error. This figure clearly displays the conflict between the objectives, and the possible different responses presented in the Pareto Fronts obtained for this problem.

Figure 5.11 – Position responses for different weight vectors.



Reference: author

The gains values used to generate the position responses illustrated in Figure 5.11 are presented in Table 5.13.

5.6 Conclusion

This chapter presented the results associated with multi-objective formulations for the classical problem of tuning controllers for speed and position control of a DC motor drive. Firstly, several combinations of objectives associated with the type of response were investigated to choose the one that presents a higher degree of conflict among them. Afterwards, a series of runs of the algorithms regarding the possible setting of the main

Table 5.13 – Results of the tuning of the PI controllers - Position control problem.

	Variables	w_1	w_2	w_3	w_4
Position Controller	k_{pp}	3.2279	6.5318	8.4301	12.0320
	k_{ip}	0.0000	10.9163	4.9791	0.0000
Speed Controller	$k_{p\omega}$	17.5728	10.5028	16.9011	14.4051
	$k_{i\omega}$	8.8929	7.7277	9.8654	11.2369
	dz_{ω}	3.3667	7.5325	13.8741	9.6373
Current Controller	k_{pi}	14.5467	20.0000	20.0000	17.0497
	k_{ii}	11.8344	12.4420	20.0000	13.2961
	dz_i	126.6931	161.9172	120.5214	107.8293

Reference: author

parameters of the NSGA-II and SPEA2 evolutionary algorithms was performed, revealing similar performance for most configurations. In the end, comparisons were made showing that for the speed control problem, the SPEA2 algorithm showed superiority over the NSGA-II with respect to the quality indicator S, while in the position control problem, the average HV obtained by SPEA2 was higher than the one given by NSGA-II. Within this chapter, the decision-making process was also addressed, in which four different types of speed and position responses were presented to illustrate the wide range of possibilities within the Pareto fronts displayed by both algorithms.

CHAPTER 6

General Conclusions

In first place, the aim of this work was to use and compare different optimisation algorithms applied to a DC motor drive system. Four algorithms were utilized to perform optimal tuning of the controllers' gains and anti-windup circuits for closed-loop speed and position control. The non-linearity of the system, caused by limitations on the output of the armature current demand and armature voltage applied to the DC motor at their respective rated values, made finding the optimal tuning for regulators a challenging task. To explore different objectives, formulations, and algorithms, two scenarios were studied, including a single-objective formulation using the integral absolute error as a metric to compare PSO and SA algorithms and a second formulation using multi-objective evolutionary algorithms NSGA-II and SPEA2 to address three objectives simultaneously. For both formulations, the large search space made the task of finding the optimal tuning even more difficult when compared to the single-objective problem.

This work demonstrates a rigorous and systematic approach used to evaluate optimisation algorithms applied to a DC motor drive system. A distinguished aspect is the analysis which has been made, where a couple of distinct parameter setting configurations were evaluated for each single-objective algorithm (PSO and SA) either for the speed and position control problems. Furthermore, statistical tests were carried out for a fair of algorithm performance, given the stochastic nature of metaheuristics. This resulted in 1440 runs for PSO and SA, with an average of approximately 594.1 (s) for each run. For the multi-objective algorithms, 640 runs were performed. Although the number of different parameter setting for the multi-objective algorithms was reduced to 8 due to the absence of statistically significant differences, the approach taken ensured a thorough evaluation

of the algorithms. Overall, this work presents a comprehensive analysis of optimisation algorithms for DC motor drive systems, with a total of 2080 runs to obtain enough data before computing the results.

For the speed control problem, formulated to minimise the integral of the absolute speed error, it was found that there was no significant difference between the different parameter configurations considered for the PSO and SA algorithms. However, using the Mann-Whitney U test, it was identified that the PSO algorithm performed better than SA, converging faster towards the optimal solutions. Regarding the tuning of controller gains, it was observed that the proportional and integral gains were set to high values, an expected result, as it leads to faster speed responses towards the reference value. An important point to note is related to the dead-zone window of the anti-windup circuit of the current PI controller, which showed a high standard deviation, indicating the absence of windup phenomenon in the current loop.

In the second case study, the position control problem was considered, and the PSO and SA algorithms were utilized to minimise the integral of the absolute position error. The majority of PSO algorithm's parameters configurations performed statistically the same, except for the configuration with a population size of 20 with equal values for the acceleration coefficients and random variation of the inertia weight, which resulted in inferior performance compared to others. On the other hand, the SA algorithm had equivalent performance across the different configurations evaluated. Once again, in the comparison between the two algorithms, PSO demonstrated superiority, achieving lower IAE objective values and showing greater robustness than SA.

In the last part of this work, Chapter 5 presents the results associated with the multi-objective formulations. Four classic control theory objectives (E_{ss}, T_r, T_s and OS) were taken into consideration, and after a series of preliminary analysis, it was decided to formulate the speed control problem aiming to minimise the E_{ss}, T_s and OS, and the position control problem with the objectives E_{ss}, T_r and OS. This choice was made because these objectives express more conflicting characteristics. In the end, simulations were carried out considering different parameter configurations for both algorithms. From the obtained Pareto Fronts, three quality indicators were used to evaluate the performance of multi-objective algorithms: Hypervolume, Spread, and RNI.

For the multi-objective speed control problem, the NSGA-II algorithm showed no difference in performance regarding any quality indicator, while for SPEA2, configuration number 4 ($N_P = 100, p_c = 0.9$ and $p_m = 1/n$) came up with better Spread values than configuration number 5 ($N_P = 50, p_c = 0.5$ and $p_m = 0.5$). Regarding Hypervolume and RNI, NSGA-II and SPEA2 have shown statistically equal performances, while with regards to the Spread indicator, SPEA2 displayed an advantage over the NSGA-II.

In the comparison between algorithms for the multi-objective position control problem, there was a superiority for SPEA2 concerning the HV metric, while for the Spread and RNI indicators, there was a statistical equality.

The WSM method was used to visualise different types of speed and position response present in the Pareto Fronts displayed by the multi-objective algorithms. Four different objective weights were assigned, which led to different tuning and, consequently, different speed and position responses based on the objectives weight. This clearly displayed the conflicts between the different objectives in the Pareto Front generated by the optimisation process.

Despite the numbers of parameters to be optimised, the algorithms reached convergence after a relatively small number of runs when compared to the universe of different possibilities to be exploited, demonstrating their effectiveness as a powerful tool for application in electrical drive systems.

6.1 Future Works

As future works, the following are highlighted:

- The use of other metaheuristics, such as Artificial Bee Colony (ABC), Ant Colony Optimisation (ACO), Harmony Search (HS), Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D), and Multi-objective Particle Swarm Optimization (MOPSO);
- Automatically investigate the parameter settings of these algorithms when applied to controller tuning problems;
- Replace the PI controllers with fuzzy logic controllers, tuning their parameters through the implemented algorithms;
- Apply the developed algorithms in the control of different types of electric machines, such as the electrically excited synchronous machine.

6.2 List of Publications

Papers published at conferences:

- SANTOS, G. F. D.; SILVA, W. G. D.; JÚNIOR, G. D. C. Tuning of pi controllers for electric drives using evolutionary multi-objective optimization algorithm. In: IEEE. 2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI). [S.l.], 2022. p. 1–6.
- SANTOS, G. F. dos; SILVA, W. G. da; PICKERT, V.; JÚNIOR, G. da C. Tuning of a pi speed regulator for electric drives by using particle swarm optimization algorithm. In: IEEE. 2021 Brazilian Power Electronics Conference (COBEP). [S.l.], 2021. p. 1–8.
- SANTOS, G. F. dos; SILVA, W. G. da; PICKERT, V.; CRUZ, G. da. Comparison of pso and sa optimization algorithms on the tuning of a pi speed regulator for electric drives. In: IEEE. 2021 Brazilian Power Electronics Conference (COBEP). [S.l.], 2021. p. 1–8.

Bibliography

AGRAWAL, P.; ABUTARBOUSH, H. F.; GANESH, T.; MOHAMED, A. W. Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access*, IEEE, v. 9, p. 26766–26791, 2021. Cited 4 times on pages 23, 28, 29, and 30.

BARA'A, A. A.; ABBOOD, A. D.; HASAN, A. A.; PIZZUTI, C.; AL-ANI, M.; ÖZDEMIR, S.; AL-DABBAGH, R. D. A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerging development and future directions. *Swarm and Evolutionary Computation*, Elsevier, v. 63, p. 100885, 2021. Cited on page 29.

BARBOSA, C. E. M. *Algoritmos bio-inspirados para solução de problemas de otimização.* Dissertation (Master) — Universidade Federal de Pernambuco, 2017. Cited 2 times on pages 26 and 27.

BECCENERI, J. C. Meta-heurísticas e otimização combinatória: Aplicações em problemas ambientais. *INPE, Sao José dos Campos*, 2008. Cited on page 26.

BERGH, F. Van den; ENGELBRECHT, A. P. A study of particle swarm optimization particle trajectories. *Information sciences*, Elsevier, v. 176, n. 8, p. 937–971, 2006. Cited on page 34.

CAPRARO, C. T.; BRADARIC, I.; CAPRARO, G. T.; LUE, T. K. Using genetic algorithms for radar waveform selection. In: IEEE. *2008 IEEE Radar Conference.* [S.l.], 2008. p. 1–6. Cited on page 104.

CARRASCO, J.; GARCÍA, S.; RUEDA, M.; DAS, S.; HERRERA, F. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, Elsevier, v. 54, p. 100665, 2020. Cited 3 times on pages 51, 61, and 63.

CARVALHO, V. R. d. *Using multi-agent systems and social choice theory to design hyper-heuristics for multi-objective optimization problems.* Thesis (PhD) — Universidade de São Paulo, 2022. Cited 2 times on pages 38 and 42.

CASTILLO-VALDIVIESO, P. A.; MERELO, J. J.; PRIETO, A.; ROJAS, I.; ROMERO, G. Statistical analysis of the parameters of a neuro-genetic algorithm. *IEEE Transactions on Neural Networks*, IEEE, v. 13, n. 6, p. 1374–1394, 2002. Cited on page 63.

CHAPMAN, S. J. *Fundamentos de máquinas elétricas*. [S.l.]: AMGH editora, 2013. Cited on page 52.

CHAUHAN, S.; SINGH, M.; AGGARWAL, A. K. Diversity driven multi-parent evolutionary algorithm with adaptive non-uniform mutation. *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, v. 33, n. 5, p. 775–806, 2021. Cited on page 45.

CHEN, Q.; MA, X.; YU, Y.; SUN, Y.; ZHU, Z. Multi-objective evolutionary multi-tasking algorithm using cross-dimensional and prediction-based knowledge transfer. *Information Sciences*, Elsevier, v. 586, p. 540–562, 2022. Cited on page 68.

CHIDAMBARAM, M.; SAXENA, N. Relay tuning of pid controllers. *Tamil Nadu: Springer*, Springer, 2018. Cited on page 22.

CHIEN, K. L.; HRONES, J.; RESWICK, J. On the automatic control of generalized passive systems. *Transactions of the American Society of Mechanical Engineers*, American Society of Mechanical Engineers, v. 74, n. 2, p. 175–183, 1952. Cited on page 22.

CHOŁODOWICZ, E.; ORŁOWSKI, P. Comparison of spea2 and nsga-ii applied to automatic inventory control system using hypervolume indicator. *Studies in Informatics and Control*, v. 26, n. 1, p. 67–74, 2017. Cited on page 44.

COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. et al. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. Cited on page 38.

COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. et al. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. Cited 2 times on pages 46 and 58.

COHEN, G.; COON, G. Theoretical consideration of retarded control. *Transactions of the American Society of Mechanical Engineers*, American Society of Mechanical Engineers, v. 75, n. 5, p. 827–834, 1953. Cited on page 22.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. Cited 2 times on pages 40 and 104.

DELAHAYE, D.; CHAIMATANAN, S.; MONGEAU, M. Simulated annealing: From basics to applications. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2019. p. 1–35. Cited on page 75.

DEVECI, M.; DEMIREL, N. C. Evolutionary algorithms for solving the airline crew pairing problem. *Computers & Industrial Engineering*, Elsevier, v. 115, p. 389–406, 2018. Cited on page 50.

DIABAT, A.; DESKOORES, R. A hybrid genetic algorithm based heuristic for an integrated supply chain problem. *Journal of Manufacturing Systems*, Elsevier, v. 38, p. 172–180, 2016. Cited on page 104.

DUTTA, R.; KUMAR, N.; PANKAJ, D. Pid control for ambulatory gait orthosis: application of different tuning methods. *Advances in Biomedical Engineering Research*, Science and Engineering Publishing Company, v. 2, p. 44–49, 2014. Cited on page 22.

EBERHART, R. C.; SHI, Y.; KENNEDY, J. *Swarm intelligence*. [S.l.]: Elsevier, 2001. Cited on page 33.

ELARBI, M.; BECHIKH, S.; SAID, L. B.; DATTA, R. Multi-objective optimization: classical and evolutionary approaches. In: *Recent advances in evolutionary multi-objective optimization*. [S.l.]: Springer, 2017. p. 1–30. Cited on page 48.

ENGELBRECHT, A. P. *Computational intelligence: an introduction*. [S.l.]: John Wiley & Sons, 2007. Cited 2 times on pages 32 and 33.

EUZÉBIO, T. A. M. et al. Sintonia ótima de controlador pid descentralizado para processos mimo. Universidade Federal de Campina Grande, 2015. Cited on page 58.

FASIH, H.; TAVAKOLI, S.; SADEGHI, J.; TORABI, H. Kalman filter-based centralized controller design for non-square multi-input multi-output processes. *Chemical Engineering Research and Design*, Elsevier, v. 132, p. 187–198, 2018. Cited on page 22.

FEI, Z.; LI, B.; YANG, S.; XING, C.; CHEN, H.; HANZO, L. A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems. *IEEE Communications Surveys & Tutorials*, IEEE, v. 19, n. 1, p. 550–586, 2016. Cited on page 46.

FIX, E.; HODGES, J. L. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, JSTOR, v. 57, n. 3, p. 238–247, 1989. Cited on page 44.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. Cited on page 63.

GARCIA, C. E.; MORARI, M. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, ACS Publications, v. 21, n. 2, p. 308–323, 1982. Cited on page 22.

GENDREAU, M.; POTVIN, J.-Y. et al. *Handbook of metaheuristics*. [S.l.]: Springer, 2010. Cited on page 22.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, Elsevier, v. 13, n. 5, p. 533–549, 1986. Cited on page 29.

HASSANAT, A.; ALMOHAMMADI, K.; ALKAFAWEEN, E.; ABUNAWAS, E.; HAMMOURI, A.; PRASATH, V. S. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, MDPI, v. 10, n. 12, p. 390, 2019. Cited on page 104.

HENDERSON, D.; JACOBSON, S. H.; JOHNSON, A. W. The theory and practice of simulated annealing. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2003. p. 287–319. Cited on page 36.

HU, Z.; LI, Z.; SUN, H.; WEI, L. Moea3h: Multi-objective evolutionary algorithm based on hierarchical decision, heuristic learning and historical environment. *ISA transactions*, Elsevier, 2022. Cited on page 68.

HUANG, C.; LI, Y.; YAO, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE transactions on evolutionary computation*, IEEE, v. 24, n. 2, p. 201–216, 2019. Cited on page 105.

HUSSAIN, K.; SALLEH, M. N. M.; CHENG, S.; SHI, Y. Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, Springer, v. 52, n. 4, p. 2191–2233, 2019. Cited on page 30.

HUTTER, F.; LÓPEZ-IBÁNEZ, M.; FAWCETT, C.; LINDAUER, M.; HOOS, H. H.; LEYTON-BROWN, K.; STÜTZLE, T. Aclib: A benchmark library for algorithm configuration. In: SPRINGER. *International Conference on Learning and Intelligent Optimization*. [S.l.], 2014. p. 36–40. Cited on page 66.

JOVIC, S.; NEDELJKOVIC, B.; GOLUBOVIC, Z.; KOSTIC, N. Evolutionary algorithm for reference evapotranspiration analysis. *Computers and Electronics in Agriculture*, Elsevier, v. 150, p. 1–4, 2018. Cited on page 50.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. *Proceedings of ICNN'95-international conference on neural networks*. [S.l.], 1995. v. 4, p. 1942–1948. Cited on page 31.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. *science*, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983. Cited on page 36.

KORA, P.; YADLAPALLI, P. Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications*, Foundation of Computer Science, v. 162, n. 10, 2017. Cited on page 45.

KRİSHNAN, R. *Electric Motor Drives-Modeling, Analysis and Control, Virginia Tech, Blacksburg, VA*. [S.l.]: Prentice Hall, 2001. Cited on page 65.

LEQUIN, O.; GEVERS, M.; TRIEST, L. Optimizing the settling time with iterative feedback tuning. *IFAC Proceedings Volumes*, Elsevier, v. 32, n. 2, p. 4659–4663, 1999. Cited on page 51.

LI, B.; LI, J.; TANG, K.; YAO, X. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, Acm New York, NY, USA, v. 48, n. 1, p. 1–35, 2015. Cited on page 40.

LIMA, J. A. d. P. *Uma abordagem baseada em hiper-heurística e otimização multi-objetivo para o teste de mutação de ordem superior*. Dissertation (Master) — Universidade Federal do Parana, 2017. Cited on page 68.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016. Cited on page 66.

LOPEZ, L. S. Investigação de técnicas eficientes para algoritmos evolutivos multiobjetivo baseados em decomposição. Universidade Federal de Minas Gerais, 2017. Cited on page 38.

MAASHI, M. *An investigation of multi-objective hyper-heuristics for multi-objective optimisation*. Thesis (PhD) — University of Nottingham, 2014. Cited 3 times on pages 29, 36, and 48.

MANIKANDAN, R.; ARULMOZHIYAL, R. Position control of dc servo drive using fuzzy logic controller. In: IEEE. *2014 International Conference on Advances in Electrical Engineering (ICAEE)*. [S.l.], 2014. p. 1–5. Cited on page 23.

MELO, V. V. d. *Técnicas de Aumento de Eficiência para metaheurísticas aplicadas a otimização global contínua e discreta*. Thesis (PhD) — Universidade de São Paulo, 2009. Cited 3 times on pages 27, 28, and 62.

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H.; TELLER, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, American Institute of Physics, v. 21, n. 6, p. 1087–1092, 1953. Cited on page 36.

MIETTINEN, K.; HAKANEN, J.; PODKOPAEV, D. Interactive nonlinear multiobjective optimization methods. *Multiple criteria decision analysis: state of the art surveys*, Springer, p. 927–976, 2016. Cited on page 59.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. *Advances in engineering software*, Elsevier, v. 69, p. 46–61, 2014. Cited on page 29.

NEUBAUER, A. Adaptive non-uniform mutation for genetic algorithms. In: SPRINGER. *International Conference on Computational Intelligence*. [S.l.], 1997. p. 24–34. Cited on page 45.

PEER, E. S.; BERGH, F. van den; ENGELBRECHT, A. P. Using neighbourhoods with the guaranteed convergence pso. In: IEEE. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*. [S.l.], 2003. p. 235–242. Cited on page 32.

PENG, J.; CHEN, Y.; EBERHART, R. Battery pack state of charge estimator design using computational intelligence approaches. In: IEEE. *Fifteenth Annual Battery Conference on Applications and Advances (Cat. No. 00TH8490)*. [S.l.], 2000. p. 173–177. Cited on page 35.

PIOTROWSKI, A. P.; NAPIORKOWSKI, J. J.; PIOTROWSKA, A. E. Population size in particle swarm optimization. *Swarm and Evolutionary Computation*, Elsevier, v. 58, p. 100718, 2020. Cited on page 67.

PURSHOUSE, R. C.; DEB, K.; MANSOR, M. M.; MOSTAGHIM, S.; WANG, R. A review of hybrid evolutionary multiple criteria decision making methods. In: IEEE. *2014 IEEE congress on evolutionary computation (CEC)*. [S.l.], 2014. p. 1147–1154. Cited on page 59.

RATNAWEERA, A. Particle swarm optimization with self-adaptive acceleration coefficients. In: *Proc. Int'l Conf. Fuzzy Syst. & Knowledge Discovery (FSKD 2002), Singapore, Nov.* [S.l.: s.n.], 2002. v. 1, p. 264–268. Cited 2 times on pages 35 and 67.

REIS, M. R. d. C. et al. Análise comparativa de métodos de otimização aplicados à sintonia do controlador pi. Universidade Federal de Goiás, 2014. Cited on page 52.

RIBEIRO, V. H. A.; REYNOSO-MEZA, G. Multi-objective pid controller tuning for an industrial gasifier. In: IEEE. *2018 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.], 2018. p. 1–6. Cited on page 51.

RODRÍGUEZ-MOLINA, A.; MEZURA-MONTES, E.; VILLARREAL-CERVANTES, M. G.; ALDAPE-PÉREZ, M. Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem. *Applied Soft Computing*, Elsevier, v. 93, p. 106342, 2020. Cited 6 times on pages 22, 23, 40, 51, 57, and 58.

RONCHI, F. P. Construção e análise do desempenho de um motor de corrente contínua utilizando materiais magnéticos macios a partir da metalurgia do pó. 2015. Cited on page 52.

RUANO, A. E.; GE, S. S.; GUERRA, T. M.; LEWIS, F. L.; PRINCIPE, J. C.; COLNARIČ, M. Computational intelligence in control. *Annual Reviews in Control*, Elsevier, v. 38, n. 2, p. 233–242, 2014. Cited on page 23.

SANPRASIT, K.; ARTRIT, P. Optimal comparison using mowoa and mogwo for pid tuning of dc servo motor. *Journal of Automation and Control Engineering Vol*, v. 7, n. 1, 2019. Cited on page 23.

SANTOS, G. F. D.; SILVA, W. G. D.; JÚNIOR, G. D. C. Tuning of pi controllers for electric drives using evolutionary multi-objective optimization algorithm. In: IEEE. *2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. [S.l.], 2022. p. 1–6. Cited on page 105.

SANTOS, G. F. dos; SILVA, W. G. da; PICKERT, V.; CRUZ, G. da. Comparison of pso and sa optimization algorithms on the tuning of a pi speed regulator for electric drives. In: IEEE. *2021 Brazilian Power Electronics Conference (COBEP)*. [S.l.], 2021. p. 1–8. Cited 2 times on pages 23 and 36.

SANTOS, G. F. dos; SILVA, W. G. da; PICKERT, V.; JÚNIOR, G. da C. Tuning of a pi speed regulator for electric drives by using particle swarm optimization algorithm. In: IEEE. *2021 Brazilian Power Electronics Conference (COBEP)*. [S.l.], 2021. p. 1–8. Cited 2 times on pages 31 and 67.

SARDAHI, Y.; BOKER, A. Multi-objective optimal design of four-parameter pid controls. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *Dynamic Systems and Control Conference*. [S.l.], 2018. v. 51890, p. V001T01A001. Cited on page 23.

SCHERVISH, M. J. P values: what they are and what they are not. *The American Statistician*, Taylor & Francis, v. 50, n. 3, p. 203–206, 1996. Cited on page 62.

SEN, R.; PATI, C.; DUTTA, S.; SEN, R. Comparison between three tuning methods of pid control for high precision positioning stage. *Mapan*, Springer, v. 30, n. 1, p. 65–70, 2015. Cited on page 22.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, JSTOR, v. 52, n. 3/4, p. 591–611, 1965. Cited on page 63.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. [S.l.], 1998. p. 69–73. Cited on page 34.

SILVA, J. C. et al. Comparação de abordagens mopso no planejamento da operação de sistemas hidrotérmicos. Universidade Federal de Goiás, 2014. Cited 4 times on pages 40, 46, 48, and 67.

SILVA, L. R. da; FLESCH, R. C.; NORMEY-RICO, J. E. Analysis of anti-windup techniques in pid control of processes with measurement noise. *IFAC-PapersOnLine*, Elsevier, v. 51, n. 4, p. 948–953, 2018. Cited on page 56.

SILVA, W. D.; ACARNLEY, P.; FINCH, J. Anti-windup circuits with on-line optimisation by genetic algorithm. In: *9th European Conference on Power Electronics and Applications-EPE 2001*. [S.l.: s.n.], 2001. Cited 2 times on pages 23 and 57.

SILVA, W. G. D. *Speed control of electric drives in the presence of load disturbances*. Thesis (PhD) — University of Newcastle upon Tyne, 1999. Cited 4 times on pages 45, 51, 60, and 104.

SILVA, W. G. da; SANTOS, G. F. dos; PICKERT, V. On the tuning of a pi speed controller for electric drives by using simulated annealing algorithm. In: *Congresso Brasileiro de Automatica-CBA*. [S.l.: s.n.], 2020. v. 2, n. 1. Cited on page 75.

SOUZA, M. d. Automatic algorithm configuration: methods and applications. 2022. Cited on page 67.

SRINIVAS, N.; DEB, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, MIT Press, v. 2, n. 3, p. 221–248, 1994. Cited on page 40.

TALBI, E.-G. *Metaheuristics: from design to implementation*. [S.l.]: John Wiley & Sons, 2009. Cited on page 50.

TAN, K. C.; LEE, T. H.; KHOR, E. F. Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial intelligence review*, Springer, v. 17, n. 4, p. 251–290, 2002. Cited on page 48.

TIAN, Y.; SI, L.; ZHANG, X.; CHENG, R.; HE, C.; TAN, K. C.; JIN, Y. Evolutionary large-scale multi-objective optimization: A survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, v. 54, n. 8, p. 1–34, 2021. Cited on page 23.

VACHHANI, V. L.; DABHI, V. K.; PRAJAPATI, H. B. Survey of multi objective evolutionary algorithms. In: IEEE. *2015 international conference on circuits, power and computing technologies [ICCPCT-2015]*. [S.l.], 2015. p. 1–9. Cited on page 39.

VERMA, S.; PANT, M.; SNASEL, V. A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *Ieee Access*, IEEE, v. 9, p. 57757–57791, 2021. Cited 3 times on pages 40, 41, and 42.

VILLARREAL-CERVANTES, M. G.; RODRIGUEZ-MOLINA, A.; GARCIA-MENDOZA, C.-V.; PENALOZA-MEJIA, O.; SEPÚLVEDA-CERVANTES, G. Multi-objective on-line optimization approach for the dc motor controller tuning using differential evolution. *IEEE Access*, IEEE, v. 5, p. 20393–20407, 2017. Cited on page 23.

VRANCIC, D.; PENG, Y. Some practical recommendations in anti-windup design. In: IET. *UKACC International Conference on Control'96 (Conf. Publ. No. 427)*. [S.l.], 1996. v. 1, p. 108–113. Cited on page 57.

WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 196–202. Cited on page 63.

WONG, K. K.; HOO, C. L.; MOHYI, M. H. H. Anti-windup pi controller, sipic for motor position control. In: EDP SCIENCES. *MATEC Web of Conferences*. [S.l.], 2018. v. 152, p. 02022. Cited on page 23.

YE, F.; WANG, H.; DOERR, C.; BÄCK, T. Benchmarking a genetic algorithm with configurable crossover probability. In: SPRINGER. *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II*. [S.l.], 2020. p. 699–713. Cited on page 104.

ZIEGLER, J. G.; NICHOLS, N. B. et al. Optimum settings for automatic controllers. *trans. ASME*, v. 64, n. 11, 1942. Cited on page 22.

ZITZLER, E. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. [S.l.]: Citeseer, 1999. Cited on page 48.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische ..., v. 103, 2001. Cited 2 times on pages 42 and 104.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, IEEE, v. 3, n. 4, p. 257–271, 1999. Cited on page 46.

ZOLFAGHARIAN, A.; NOSHADI, A.; ZAIN, M. Z. M. et al. Practical multi-objective controller for preventing noise and vibration in an automobile wiper system. *Swarm and Evolutionary Computation*, Elsevier, v. 8, p. 54–68, 2013. Cited on page 58.