UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

ANANDA LIMA FREIRE

ON THE EFFICIENT DESIGN OF EXTREME LEARNING MACHINES USING INTRINSIC PLASTICITY AND EVOLUTIONARY COMPUTATION APPROACHES

FORTALEZA
2015

ANANDA LIMA FREIRE

ON THE EFFICIENT DESIGN OF EXTREME LEARNING MACHINES USING INTRINSIC PLASTICITY AND EVOLUTIONARY COMPUTATION APPROACHES

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Sinais e sistemas

Orientador: Prof. Dr. Guilherme de Alencar Barreto

FORTALEZA

2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F933o Freire, Ananda Lima.
On the Efficient Design of Extreme Learning Machines Using Intrinsic Plasticity and Evolutionary Computation Approaches / Ananda Lima Freire. – 2015.
222 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2015.
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. Máquina de Aprendizado Extremo. 2. Robustez a Outliers. 3. Metaheurísticas. I. Título.
CDD 621.38

ANANDA LIMA FREIRE

ON THE EFFICIENT DESIGN OF EXTREME LEARNING MACHINES USING INTRINSIC PLASTICITY AND EVOLUTIONARY COMPUTATION APPROACHES

Thesis presented to the Graduate Program in Teleinformatics Engineering of the Federal University of Ceará in partial fulfillment of the requirements for the degree of Doctor in the area of Teleinformatics Engineering. Concentration Area of Study: Signals and Systems

Date Approved: February 10, 2015

EXAMINING COMMITTEE

Prof. Dr. Guilherme de Alencar Barreto
(Committee Chair / Advisor)
Federal University of Ceará (UFC)

Prof. Dr. George André Pereira Thé
Federal University of Ceará (UFC)

Prof. Dr. Marcos José Negreiros Gomes
State University of Ceará (UECE)

Prof. Dr. Felipe Maia Galvão França
Federal University of Rio de Janeiro (UFRJ)

In dedication to my mother and my beloved Victor for their endless love, support and encouragement.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my beloved ones, Odete and Victor, for their love and support through this journey. Thank you for giving me the strength to deal with the distresses of this path and helping me to chase my dreams.

I would like to sincerely thank my advisor, Prof. Dr. Guilherme Barreto, for his guidance, patient encouragement and confidence in me throughout this study, and especially his unhesitating support in a dark and personal moment I had been through. I would also like to thank Prof. Dr. Jochen Steil for accepting me to work at CoR-Lab under his supervision. I learned a lot from that experience, not only in a professional sphere but also as a person.

To the Professors Dr. George Thé, Dr. Marcos Gomes, Dr. Felipe França and Dr. Frederico Guimarães, thank you for your time and valuable contributions to the improvement of this work.

To my colleagues, both in UFC and in CoR-Lab, thank you all for making the work places such pleasant environments and for always being available for discussions.

To CAPES and the DAAD/CNPq scholarship program, thank you for the financial support.

To NUTEC and CENTAURO for providing their facilities.

I would also like to thank the secretaries and Coordination Departments of both UFC and Universität Bielefeld for their support.

Last but not least, thank you, God, for always being there for me.

RESUMO

A rede Máquina de Aprendizado Extremo (*Extreme Learning Machine* - ELM) tornou-se uma arquitetura neural bastante popular devido a sua propriedade de aproximadora universal de funções e ao rápido treinamento, dado pela seleção aleatória dos pesos e limiares dos neurônios ocultos. Apesar de sua boa capacidade de generalização, há ainda desafios consideráveis a superar. Um deles refere-se ao clássico problema de se determinar o número de neurônios ocultos, o que influencia na capacidade de aprendizagem do modelo, levando a um sobreajustamento, se esse número for muito grande, ou a um subajustamento, caso contrário. Outro desafio está relacionado à seleção aleatória dos pesos da camada oculta, que pode produzir uma matriz de ativações mal-condicionada, dificultando sensivelmente a solução do sistema linear construído para treinar os pesos da camada de saída. Tal situação leva a soluções com normas muito elevadas e, consequentemente, numericamente instáveis. Baseado nesses desafios, este trabalho oferece duas contribuições orientadas a um projeto eficiente da rede ELM. A primeira, denominada R-ELM/BIP, combina a versão *batch* de um método de aprendizado recente chamado Plasticidade Intrínseca com a técnica de estimação robusta conhecida como *Estimação M*. Esta proposta fornece solução confiável na presença de *outliers*, juntamente com boa capacidade de generalização e pesos de saída com normas reduzidas. A segunda contribuição, denominada *Adaptive Number of Hidden Neurons Approach* (ANHNA), está orientada para a seleção automática de um modelo de rede ELM usando metaheurísticas. A ideia subjacente consiste em definir uma codificação geral para o indivíduo de uma população que possa ser usada por diferentes metaheurísticas populacionais, tais como Evolução Diferencial e Enxame de Partículas. A abordagem proposta permite que estas metaheurísticas produzam soluções otimizadas para os vários parâmetros da rede ELM, incluindo o número de neurônios ocultos e as inclinações e limiares das funções de ativação dos mesmos, sem perder a principal característica da rede ELM: o mapeamento aleatório do espaço da camada oculta. Avaliações abrangentes das abordagens propostas são realizadas usando conjuntos de dados para regressão disponíveis em repositórios públicos, bem como um novo conjunto de dados gerado para o aprendizado da coordenação visuomotora de robôs humanoides.

Palavras-chave: Máquina de Aprendizado Extremo, Robustez, Metaheurística, Plasticidade Intrínseca em Batelada

ABSTRACT

The Extreme Learning Machine (ELM) has become a very popular neural network architecture due to its universal function approximation property and fast training, which is accomplished by setting randomly the hidden neurons' weights and biases. Although it offers a good generalization performance with little time consumption, it also offers considerable challenges. One of them is related to the classical problem of defining the network size, which influences the ability to learn the model and will overfit if it is too large or underfit if it is too small. Another is related to the random selection of input-to-hidden-layer weights that may produce an ill-conditioned hidden layer output matrix, which derails the solution for the linear system used to train the output weights. This leads to a solution with a high norm that becomes very sensitive to any contamination present in the data. Based on these challenges, this work provides two contributions to the ELM network design principles. The first one, named R-ELM/BIP, combines the maximization of the hidden layer's information transmission, through Batch Intrinsic Plasticity, with outlier-robust estimation of the output weights. This method generates a reliable solution in the presence of corrupted data with a good generalization capability and small output weight norms. The second method, named Adaptive Number of Hidden Neurons Approach (ANHNA), is defined as a general solution encoding that allows populational metaheuristics to evolve a close to optimal architecture for ELM networks combined with activation function's parameter optimization, without losing the ELM's main feature: the random mapping from input to hidden space. Comprehensive evaluations of the proposed approaches are performed using regression datasets available in public repositories, as well as using a new set of data generated for learning visuomotor coordination of humanoid robots.

Keywords: Extreme Learning Machine, Robustness, Metaheuristics, Batch Intrinsic Plasticity

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS AND ACRONYMS

ANHNA	Adaptive Number of Hidden Neurons Approach
ANN	Artificial Neural Network
BIP	Batch Intrinsic Plasticity
Cor-Lab	Research Institute of Cognition and Robotics
DE	Differential Evolution
EANN	Evolutionary Artificial Neural Networks
ECOD	Extended Complete Orthogonal Decomposition
ELM	Extreme Learning Machine
EI-ELM	Enhanced Incremental Extreme Learning Machine
EM-ELM	Error Minimized Extreme Learning Machine
ESN	Echo State Network
GA	Genetic Algorithm
GO-ELM	Genetically Optimized Extreme Learning Machine
I-ELM	Incremental Extreme Learning Machine
IP	Intrinsic Plasticity
LMS	Least Mean Square
LSM	Liquid State Machines
MLP	Multilayer Perceptron
OP-ELM	Optimally Pruned Extreme Learning Machine
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RMSE	Root Mean Square Error
RVFL	Random Vector Functional-Link
SaDE	Self-Adaptive Differential Evolution
SLFN	Single Hidden Layer Feedforward Network
SVD	Singular Value Decomposition

LIST OF SYMBOLS

$\boldsymbol{\beta}$	ELM's output weight vector
γ	Parameter that controls greedness of the mutation operator
$\boldsymbol{\nu}_i$	Velocity of i-th PSO particle
κ	Condition number of a matrix
λ	Regularization parameter
ψ	Derivative of *rho*
ρ	Weighting function for M-Estimators
σ_{max}	Largest eigenvalue of a matrix
σ_{min}	Smallet eigenvalue of a matrix
τ	Set of 3 bits that determines the weight function
υ	Tunning constant for a weight function
a	Slope parameter for sigmoid and hiperbolic tangent activation functions
b	Bias
$\mathbf{C}$	Population set of a metaheuristic algorithm
$\mathbf{c}$	Individual or particle vector of a metaheuristic algorithm
$\mathbf{c}'$	Offspring vector of a metaheuristic algorithm
$\mathbf{c}_{best}$	PSO's global best position of all swarm
CR	Crossover rate
CR_{m_k}	Median value of crossover rate for a given k strategy in SaDE algorithm
$CRmemory_k$	Memory that stores the CR values of sucessful k-th strategy
$\mathbf{D}$	Desired output matrix
$\mathbf{d}$	Desired output vector
d_n	Dimension of a individual or particle vector
F	Scaling factor
g	Current generation for a given metaheuristic
$\mathbf{H}$	ELM's hidden layer output

K_s	Number of strategies available for mutation in SaDE algorithm
LP	Learning period
LS	Least Squares
$\mathbf{m}$	ELM's hidden layer weights
N	Number of training samples
Pr	Dimensionality of the search space
p_k	k-th Strategy probability
p	ELM's input pattern dimension
$\mathbf{pl}_k$	PSO's k-th neighborhood best position
$\mathbf{p}_i$	PSO's previous personal best position of i-th particle
Q_{max}	Maximum number of hidden neurons for ANHNA
Q_{min}	Minimum number of hidden neurons for ANHNA
q	Number of hidden neurons
$\mathscr{N}$	Normal distribution
ns_k	Number of offsprings generated with k-th strategy that successfuly passed to the next generation
nf_k	Number of offsprings generated with k-th strategy that were discarded
n_v	Number of differences in a mutation operator
r	ELM's output layer dimension
S_k	Success rate of the offsprings by the k-th strategy
$\mathscr{U}$	Uniform distribution
$\mathbf{U}$	Input matrix
$\mathbf{u}$	Input vector
$\mathbf{v}$	Trial vector for DE and SaDE algorithms
$\mathbf{x}_i$	Vector of all synaptic sums from the i-th hidden neuron

CONTENTS

1 INTRODUCTION

Among the different artificial neural network (ANN) architectures available, the single layer feedforward one (SLFN) is the most popular (HUANG *et al.*, 2011). It is specially known by its universal approximation abilities (CAO *et al.*, 2012). In particular, the Extreme Learning Machine (ELM) is attracting considerable attention from the Computational Intelligence community due to its fast training speed by randomly selecting the weights and biases of the hidden neurons. This singular feature allows the problem of estimating the output weights to be cast as a linear model, which is then analytically determined by finding a least-squares solution (HUANG *et al.*, 2011).

However, there is no free lunch. Even though this network is capable of achieving a good generalization performance with little time consumption, it also offers considerable challenges. One of these issues is related to an inherited problem from standard SLFN: the definition of the number of hidden neurons. A small network may not provide a good performance due to its limited information processing which leads to a too simplistic model, while a large one may lead to overfitting and poor generalization. To correctly define the number of hidden neurons, the most common strategy is to resort to trial-and-error or exhaustive search strategies (YANG *et al.*, 2012).

Another issue is related to the random nature of the input-to-hidden-layer weights (hidden weights, for short). This random combination often generates a set of non-optimal weights and biases, which results in an ill-conditioned matrix of hidden neurons' activations $\mathbf{H}$. Such situation makes the solution of the linear output system numerically unstable due to the very high norm of the resulting output weight vector and, hence, make the whole network very sensitive to data perturbation (e.g. outliers) (ZHAO *et al.*, 2011). Furthermore, as described in the work of Bartlett (1998), their size is more relevant for the generalization capability than the configuration of the neural network itself.

It is important to further discuss the second of the two aforementioned issues since it has a direct effect on ELM's outlier robustness property. Since regression and classification problems are often contaminated by noise, it is known that it may influence the modeling accuracy as well as the estimated parameters (BELIAKOV; KELAREV; YEARWOOD, 2011). Bearing this in mind, Horata, Chiewchanwattana and Sunat (2013) proposed that a robust ELM must deal with the computational robustness issues related

to the conditioning of **H** matrix, as well as with outlier robustness by choosing estimation methods, other than the standard least-squares approach, which are capable of dealing with outliers.

Knowing that methods which control only the network size and the output weights are insufficient in tunning the hidden neurons to a good regime, where the encoding is optimal (NEUMANN; STEIL, 2011), we took the *Intrinsic Plasticity* (IP) notion as inspiration to tackle the computational robustness problem. This biologically motivated mechanism was first introduced by Triesch (2005) and consists in adapting the activation function's parameters (slope and bias) to force the output of the hidden neurons to follow an exponential distribution (stochastically speaking). This maximizes the information transmission caused by the high entropy of this probability distribution. (NEUMANN; STEIL, 2011). Empirically, it also results in a more stable **H** matrix followed by a smaller norm of the output weights than the ones provided by the standard least-squares method.

Regarding outlier robustness, even though robust ANN is a subject of interest in Machine Learning, the study of this property with ELM network is still in its infancy (HORATA *et al.*, 2013). Works such as Huynh, Won and Kim (2008), Barros and Barreto (2013), Horata, Chiewchanwattana and Sunat (2013) adopt an estimation method that is less sensitive to outliers than the ordinary least-squares estimation, known as *M-estimation* (HUBER, 1964). However, Horata's work is the only one that addresses both computational and outlier robustness problem.

Hence, one of the contributions of this work is a robust version of ELM network that incorporates both regularizing and outlier robustness features to ELM learning, by combining the optimization of the hidden layer output through a new method named Batch Intrinsic Plasticity (BIP) (NEUMANN; STEIL, 2011) with the M-estimation method.

Turning to the first mentioned ELM's challenge, to deal with its architecture design issue, we choose the use of population-based metaheuristics due to the complex task of finding suitable values for the number of hidden neurons and also to other learning parameters, such as the slopes and biases of the hidden activation functions. The use of evolutionary computation on ELM networks has been a quite popular subject. As examples, we mention works such as Zhu *et al.* (2005), Xu and Shu (2006), Silva *et al.* (2011), Cao, Lin and Huang (2012), Han *et al.* (2013), Matias et al. (2013), Figueiredo and Ludermir (2014) and Alencar and Rocha Neto (2014). However, the majority of

these works are not designed to automatically determine the optimal number of hidden neurons. In other words, most of them predefine the number of hidden neurons and they usually concentrate the efforts on using those metaheuristics to find optimal values for the input-to-hidden layer weights. This approach, however, goes against the very nature of the ELM network, that is, to be an SLFN whose parameters of the input-to-hidden projection is specified randomly.

In order to propose an improved design for the ELM network following the Intrinsic Plasticity learning principles, we introduce a new encoding scheme for the solution vector of a given metaheuristic optimization algorithm that allows automatic estimation of the number of hidden neurons and their activation function parameter values. This scheme, named *Automatic Number of Hidden Neurons Approach* (ANHNA), is very general and can be used by any population-based metaheuristic algorithms for continuous optimization. Computer experiments using Differential Evolution (DE), Self-Adaptive Differential Evolution (SaDE) and Particle Swarm Optimization (PSO) metaheuristics demonstrate the feasibility of the proposed approach.

Finally, as a third main contribution, we present the preliminary results of a robust version of ANHNA. This version provides, besides the number of hidden neurons and activation function's parameters, also the choice of the objective function to handle outliers properly. This setting provides a broader range of architectures to be chosen from.

Comprehensive evaluations of the proposed approaches were performed using regression datasets available in public repositories, as well as using a new set of data generated for learning visuomotor coordination of a humanoid robot named iCub. This last dataset was collected during an internship period at the Research Institute of Cognition and Robotics (CoR-Lab), University of Bielefeld, Germany.

1.1 General and specific objectives

Our main goal in this thesis lies on the improvement of ELM network in two aspects: (1) by designing an efficient method for model selection based on evolutionary computation and ideas emanating from the intrinsic plasticity learning paradigm, and (2) proposing an outlier robust version.

Given the main goal of this work, several specific ones were being defined and pursued over time, such as the ones listed below:

1. To travel abroad to an outstanding research group on Computational Intelligence and Cognitive Science in order to acquire expertise on the theory of ELM network and its application;
2. To develop a new solution based on the ELM network to a complex hand-eye coordination problem for a humanoid robot;
3. To execute tests on simulated and real iCub robot;
4. To build a new dataset for the problem mentioned in Item 2 in order to train different neural network models;
5. To study and implement the batch intrinsic plasticity (BIP) algorithm to train ELM networks;
6. To develop a population-based metaheuristic strategy for model selection of the ELM network;
7. To study and implement robust parameter estimation methods, such as the M-estimation, in order to evaluate their applicability to ELM training.

1.2 A word on the thesis' chronology and development

In this section, the research progress is described as it was performed during this Doctorate. We detail, chronologically, the development of the studies that culminated in the main contributions of this thesis.

The first contact of the doctorate student responsible for this thesis with neural networks using random projections in the hidden layer was around 2008, where the learned expertise was further explored to give rise to a thesis project in 2010.

Between 2011 and the beginning of 2012, during the internship period at the University of Bielefeld, Germany, more specifically at CoR-Lab (Research Institute for Cognition and Robotics) facilities, there was the opportunity to be introduced to humanoid robotics and to improve the knowledge about the ELM network and several training paradigms. Along this period, there was a comprehensive training to use the humanoid robot iCub (METTA *et al.*, 2010), both simulated and real one, where we designed the pointing problem as a part of a large study about the communicative ability for cognitive agents developed by the group. This particular problem was divided in two: imperative and precise pointing (LEMME *et al.*, 2013), where the first refers to children pre-verbal method of communication that consists in gesturing by extending their arms as if reaching for an

object far away from their grasp. The second, as it is named, refers to a precise gesturing which was achieved by applying the kinesthetic teaching approach. It is interesting to highlight that only a few studies have addressed the acquisition of pointing skills by robots.

Using the data generated with the simulated and real robot, we aimed to learn a direct mapping from the object's pixel coordinates in the visual field directly to hand positions (or to joint angles) in an eye-to-hand configuration, applying neural networks. From this study, two novelties were introduced: the Static Reservoir Computing (EMMERICH; STEIL, 2010), which is another ANN with random projections, and the Batch Intrinsic Plasticity (NEUMANN; STEIL, 2011; NEUMANN; STEIL, 2013), proposed to improve ELM's generalization performance. Both proposed recently by researchers in CoR-Lab. Three publications resulted from this period, which are listed in Section 1.3.

With the finishing of the experiments on the robot, we started to think about strategies for automatic model selection of the ELM network. Later in 2012, the work from Das *et al.* (2009), which proposed an automatic clustering using Differential Evolution, came as an inspiration to evolve the number of hidden neurons as well as the activation function's parameters of the ELM network. Then, in 2013, we developed one of the contributions of this thesis, a novel population-based metaheuristic approach for model selection of the ELM network, which resulted in two other publications (see Section 1.3). We also developed several variants of this method to attend different training requirements.

By the end of 2013, came to our knowledge the work by Horata, Chiewchanwattana, and Sunat (2013) and its contribution as an outlier-robust ELM network. In their work, they highlighted two issues that affect ELM performance: computational and robustness problems. With that in mind and taking our prior knowledge on the benefits of the intrinsic plasticity learning paradigm, we introduced an outlier-robust variant of the ELM network trained with the BIP learning introduced by Neumann and Steil (2011). This simple but efficient approach combines the best of two worlds: optimization of the parameters (weights and biases) of the activation functions of the hidden neurons through the BIP learning paradigm, with outlier robustness provided by the M-estimation method used to compute the output weights. By the end of 2014, the proposed outlier-robust ELM resulted in another accepted publication.

The last part of the contributions of this work consists in the on-going study on the development of a robust extension of the aforementioned population-based metaheuristic

method for model selection of the ELM network. A detailed timeline of this thesis development is shown in Table 1, with a reference to the corresponding thesis' chapters and appendices.

Table 1 – Thesis time-line.

Pre-doctorate period		
2008	First contact with the theory of random projection networks, such as the ELM and Echo-State Networks	(Chapter 2)
Doctorate		
2010	Compulsory subjects	
	Internship period at University of Bielefeld (Germany)	
2011	training with humanoid robot iCub simulator beginning of pointing problem with humanoid robots further studies with ELM network data harvesting with simulator introduction to *Batch Intrinsic Plasticity* introduction to *Static Reservoir Computing* participation at the CITEC^2 Summer School: Mechanisms of Attention –From Experimental Studies to Technical Systems held in University of Bielefeld (3rd-8th October) training with real iCub robot real robot data harvesting of pointing problem	(Appendix B) (Chapter 2)
2012	publication of Freire *et al.* (2012b) and Freire *et al.* (2012a) study of Das *et al.* (2009) work implementation of metaheuristics	(Appendix A)
2013	publication in joint effort of Lemme *et al.* (2013) development of the population-based metaheuristic approach for model selection of the ELM network and variants. study of Horata *et al.* (2013) work	(Chapter 4)
2014	development of the outlier-robust version of the ELM network trained with the BIO learning algorithm. development and preliminary tests of an outlier-robust extension of the proposed population-based metaheuristic approach for model selection of the ELM network publication of Freire and Barreto (2014a), Freire and Barreto (2014c) and Freire and Barreto (2014b).	(Chapter 3) (Chapter 4)

Source: author.

1.3 Scientific production

In this section, we list the publications produced along the development of this thesis. We also list a number of citations that this publication has already received.

- FREIRE, A., LEMME, A., STEIL, J., BARRETO, G. Learning visuo-motor coordination for pointing without depth calculation. *In*: Proceedings of the European Symposium on Artificial Neural Networks. [S.l.: s.n.], 2012. p. 91-96.
- FREIRE, A., LEMME, A., STEIL, J., BARRETO, G. Aprendizado de coordenação visomotora no comportamento de apontar em um espaço 3D. *In*: Anais do XIX Congresso Brasileiro de Automática (CBA2012). Campina Grande (Brazil), 2012. Available in: <http://cba2012.dee.ufcg.edu.br/anais>.
- LEMME, A., FREIRE, A., STEIL, J., BARRETO, G.. Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot iCub. *Neurocomputing*, v. 112, p. 179-188, 2013.

  Used as a reference by:
  - WREDE, B., ROHLFING, K., STEIL, J., WREDE, S., OUDEYER, P.-Y. et al. Towards robots with teleological action and language understanding. Ugur, Emre, and Nagai, Yukie and Oztop, Erhan and Asada, Minoru. Humanoids 2012 Workshop on Developmental Robotics: Can developmental robotics yield human-like cognitive abilities?, Nov 2012, Osaka, Japan. <hal-00788627>
  - NEUMANN, K., STRUB, C., STEIL, J. Intrinsic plasticity via natural gradient descent with application to drift compensation, *Neurocomputing*, v. 112, 2013, p. 26-33, Available in <http://www.sciencedirect.com/science/article/pii/S0925231213002221>.
  - QUEISSER, F., NEUMANN, K., ROLF, M., REINHART, F., STEIL, J. An active compliant control mode for interaction with a pneumatic soft robot. *In*: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), 2014, p. 573,579. Available in: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6942617&isnumber=6942370>.
  - ROLF, M., ASADA, M. Motor synergies are naturally observed during goal babbling. *In*: IROS 2013 Workshop: Cognitive Neuroscience Robotics, Tokyo, 2013.
  - CHAO, F., WANG, Z., SHANG, C., MENG, Q., JIANG, M., ZHOU, C.,

SHEN, Q. A developmental approach to robotic pointing via human-robot interaction. *Information Sciences*, v. 283, 2014, p. 288-303. Available in: <http://www.sciencedirect.com/science/article/pii/S0020025514004010>.

- HUANG, G., HUANG, G.-B., SONG, S., YOU, K. Trends in extreme learning machines: A review. *Neural Networks*, v. 61, 2015, p. 32-48. Available in: <http://www.sciencedirect.com/science/article/pii/S0893608014002214>.
- SEIDEL, D., EMMERICH, C., STEIL, J. Model-free path planning for redundant robots using sparse data from kinesthetic teaching. *In*: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), 2014, p.4381. Available in: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6943182&isnumber=6942370>.

- FREIRE, A., BARRETO, G. A new model selection approach for the ELM network using metaheuristic optimization. *In*: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014). Bruges (Belgium), 23-25 April 2014. Available from: <http://www.i6doc.com/fr/livre/?GCOI=28001100432440>.
- FREIRE, A., BARRETO, G. ANHNA: Uma nova abordagem de seleção de modelos para a Máquina de Aprendizado Extremo. *In*: XX Congresso Brasileiro de Automática (CBA 2014), Minas Gerais (Brazil),20-24 Setember 2014.
- FREIRE, A., BARRETO, G. A Robust and Regularized Extreme Learning Machine. *In*: Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2014), São Carlos (Brazil), 18-23 October 2014.

Other contributions of the work developed in this thesis, from a teaching perspective, are the following:

- four hours tutorial named "Introduction to the Simulator of the iCub Humanoid Robot" in the joint event *Latin American Robotics Symposium* and *Brazilian Robotics Symposium* held in Fortaleza (Brazil) on October, 16th to 19th, 2012.
- short course of 6 hours named "Simulador do Robô Humanóide iCub" in the *III Semana de Engenharia de Teleinformática* (SETI 2013), held in Federal University of Ceará from November, 6th to 8th, 2013.

1.4 Thesis structure

The remainder of this thesis is organized as follows.

Chapter 2 presents an introduction about the ELM network with a brief review of the literature on random projections for the design of feedforward neural networks. Also it contains a discussion on the main issues of this architecture.

Chapter 3 is dedicated to outlier robustness. Here, we demonstrate with a numerical example how the presence of outliers perturb the estimated solution. We also describe the M-estimation paradigm, followed by a brief review of robust neural networks and then we review approaches that apply robust methods to ELM design. Finally, we introduce the first of our proposals, namely the robust variant of the ELM network trained with the BIP learning algorithm.

Chapter 4 addresses another important issue: model selection. In this chapter, we introduce the second of our proposals, which is a population-based approach for the design of ELM networks. We also provide a literature review on how metaheuristic approaches are being applied for the ELM design. Afterwards, we discuss the proposed approach in detail and present a number of variants resulting from modifications of the fitness function.

Chapter 5 describes the methodology adopted, providing detailed information on how the datasets were treated and contaminated for the outlier robustness cases, how all tests were performed and which parameters were chosen for the different evaluated methods.

Chapter 6 reports the performance results of the proposed robust version of the ELM network trained with the BIP learning algorithm for five distinct datasets and several outlier-contaminated scenarios.

Chapter 7 reports the performance results of the proposed population-based metaheuristic approach for the efficient design of ELM networks with six different datasets.

Chapter 8 is dedicated to the conclusions of this work.

Appendix A describes in detail the algorithms of the metaheuristics evaluated in this work, how they differ from each other, their mutation strategies and how their parameters are adapted.

Appendix B provides the details of the six real-world regression problems datasets used in this work: Auto-MPG, Body Fat Breast Cancer, CPU, iCub, and Servo.

Appendix C reports tables with values of the resulting number of hidden neurons from the graphics presented in Chapter 6. We chose the following information to describe them: mean, median, maximum, minimum and standard deviation.

Appendix D reports the comparison of all variants of the proposed population-based metaheuristic approach for the ELM model selection, using all four chosen metaheuristics. The variants with best performances were chosen to be discussed in Chapter 7.

Appendix E shows typical convergence curves related to each of the independent runs of the proposed population-based metaheuristic for model selection of the ELM network, in order to illustrate how the fitness values evolved through generations.

Finally, Appendix F provides preliminary experiments on the outlier-robust extension of the proposed population-based metaheuristic approach for model selection of the ELM network.

2 THE EXTREME LEARNING MACHINE

Random projections in feedforward neural networks have been a subject of interest long before the famous works of Huang, Zhu, and Siew (2004, 2006). The essence of those random projections is that the feature generation does not suffer any learning adaptation, being randomly initialized and remaining fixed. This characteristic leads to a much simpler system, where only the hidden-to-output weights (*output weights* for short) must be learned. In this context, the Extreme Learning Machine (ELM) arose as an appealing option for single layer feedforward neural networks (SLFN), offering a learning efficiency with conceptual simplicity.

In this chapter, Section 2.1 makes a brief review through the literature on random projections in the hidden layer of feedforward neural networks. On Section 2.2, the ELM algorithm is detailed, followed by Section 2.3, where the issues and some proposed improvements are described, such as: regularization (Subsection 2.3.1), architecture design (Subsection 2.3.2) and intrinsic plasticity (Subsection 2.3.3). At last, Section 2.4 presents the final remarks over this chapter.

2.1 Random projections in the literature

In 1958, Rosenblatt already stated about the perceptron theory:

> A relatively small number of theorists, like Ashby (ASHBY, 1952) and von Neumann (NEUMANN, 1951; NEUMANN, 1956), have been concerned with the problems of how an imperfect neural network, containing many random connections, can be made to perform reliable. (ROSENBLATT, 1958, p. 387)

In his work, he discussed a perceptron network that responds to optical patterns as stimuli. It is formed by a set of two hidden layers, called *projection area* and *association area* and their connections are random and scattered. The output offers a recursive link to the latest hidden layer. One of the main conclusions of this paper is:

> In an environment of random stimuli, a system consisting of randomly connected units, subject to the parametric constraints discussed above, can learn to associate specific responses to specific stimuli. Even if many stimuli are

> associated to each response, they can still be recognized with a better-than-chance probability, although they may resemble one another closely and may activate many of the same sensory inputs to the system. (ROSENBLATT, 1958, p. 405)

At the end of the 1980's, a Radial Basis Function (RBF) network with random selected centers and training only the output layer was proposed (BROOMHEAD; LOWE; LOWE, 1988, 1989 apud WANG; WAN; HUANG, 2008, 2008). This work, however, chooses the scale factor heuristically and focuses on the data interpolation. Schmidt *et al.* (1992) proposed a randomly chosen hidden layer, whereas the output layer is trained by a single layer learning rule or a pseudoinverse technique. It is interesting to point out that the authors did not intend to present it as an alternative learning method: "this method is introduced only to analyze the functional behavior of the networks with respect to learning." (SCHMIDT *et al.*, 1992, p. 1)

Pao and Takefuji (1992) presented a *Random Vector Functional-Link* (RVFL) which is given by one hidden layer feedforward neural network with randomly selected input-to-hidden weights (*hidden weights* for short) and biases, while the output weights are learned using simple quadratic optimization. The difference here lies on the fact that the hidden layer output is viewed as an enhancement of the input vector and both are presented simultaneously to the output layer. A theoretical justification for the RVFL is given in (IGELNIK; PAO, 1995), where the authors prove that RVFL is indeed universal function approximators for continuous functions on bounded and finite dimensional sets, using random hidden weights and tuned hidden neuron biases (HUANG, 2014). They did not address the universal approximation capability of a standard SLFN with both random hidden weights and random hidden neuron biases (HUANG, 2014).

In 2001, H. Jaeger proposed a recurrent neural network named *Echo State Network* (ESN) (JAEGER, 2001a; JAEGER, 2001b) where a high dimensional hidden layer is viewed as a reservoir of dynamics and its weights are fixed and randomly initialized. They are stored into a sparse matrix that also allows communication between hidden neurons. The hidden layer receives as input, not only the input pattern, but its own activations and may take recurrent links from the network's output as well. The architecture equally admits communication between the input directly to the output layer. Finally, the learning phase is done by collecting the reservoir states during the application of the training data

followed by linear regression to estimate the output weights.

Different works over the concept of reservoir computing were also introduced, such as Maass *et al.* (2002) under the notion of *Liquid State Machines* (LSM) and *Static Reservoir Computing* presented in (EMMERICH; STEIL, 2010). For more details over reservoir computing approaches and their relation to random projections considering output feedback can be found in (REINHART, 2011).

In a most recent effort on random projections, Widrow *et al.* (2013) proposed the *No-Propagation* (No-Prop), which has also a random and fixed hidden weights and only the output weights are trained. However, it uses the steepest descent to minimize mean squared error, with the *Least Mean Square* (LMS) algorithm of Widrow and Hoff (WIDROW; HOFF, 1960 apud WIDROW *et al.*, 2013). Nevertheless, Lim (2013) stated that this work has been already proposed by G.-B. Huang and colleagues 10 years ago and intensively discussed and applied by other authors since.

So far, few examples of different approaches were presented to contextualize where the *Extreme Learning Machine* inserts itself in the literature. The remainder of this chapter is dedicated to describe this popular network.

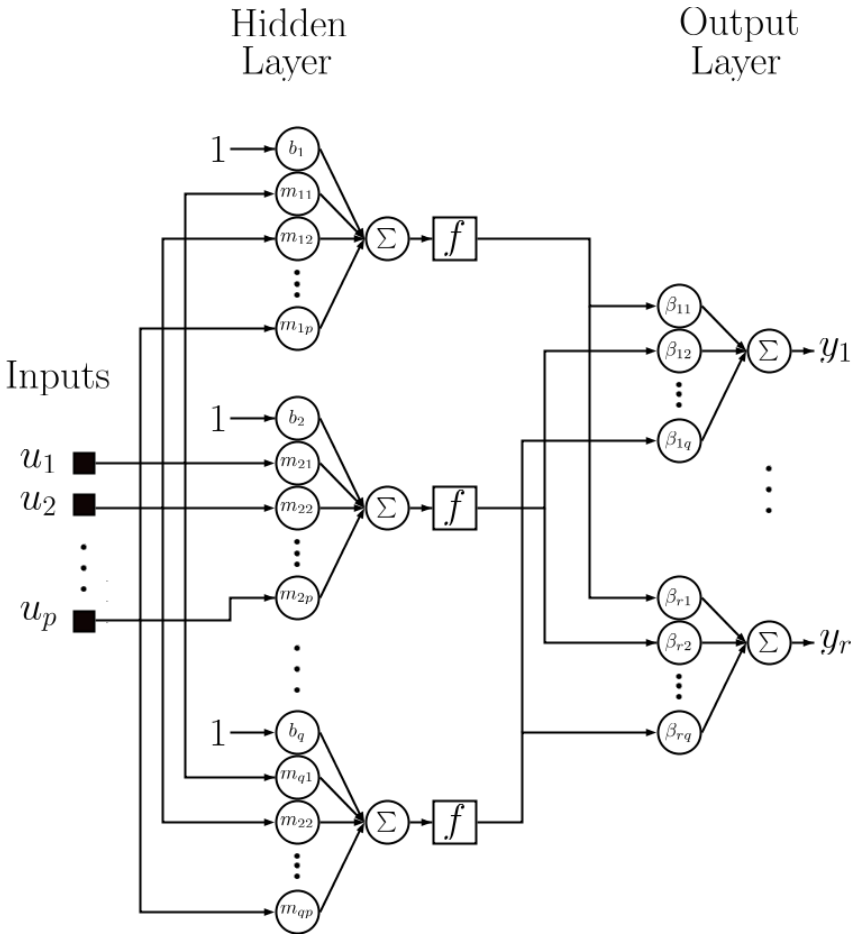
2.2 Introduction to ELM

As previously mentioned, ELM is a single hidden layer feedforward network with fixed and random projections of the input onto the hidden state space. Proposed by Huang, Zhu, and Siew (2004, 2006), it is a universal function approximator and admits not only sigmoidal networks, but also RBF networks, trigonometric networks, threshold networks, and fully complex neural networks (WIDROW *et al.*, 2013). Compared with other traditional computational intelligence techniques, such as the *Multilayer Perceptron* (MLP) and RBF, ELM provides a better generalization performance at a much faster learning speed and with minimal human intervention (HUANG; WANG; LAN, 2011).

Given a training set with N samples $\{(\mathbf{u}_i, \mathbf{d}_i)\}_{i=1}^{N}$, where $\mathbf{u}_i \in \mathbb{R}^p$ is the i-th input vector and $\mathbf{d}_i \in \mathbb{R}^r$ is its correspondent desired output. In the architecture described by Figure 1, with q hidden neurons and r output neurons, the i-th output at time step k is given by

$$\mathbf{y}(k) = \boldsymbol{\beta}^T \mathbf{h}(k), \tag{2.1}$$

Figure 1 – Extreme Learning Machine architecture, where only the output weights are trained.



Source: author.

where $\boldsymbol{\beta} \in \mathbb{R}^{q \times r}$ is the weight matrix connecting the hidden to output neurons.

For each input pattern $\mathbf{u}(k)$, the correspondent hidden state $\mathbf{h}(k) \in \mathbb{R}^q$ is given by

$$\mathbf{h}(k) = \left[f\left(\mathbf{m}_1^T \mathbf{u}(k) + b_1\right) \quad \ldots \quad f\left(\mathbf{m}_q^T \mathbf{u}(k) + b_q\right) \right], \tag{2.2}$$

where $\mathbf{m}_j \in \mathbb{R}^p$ is the weight vector of the j-th hidden neuron and is initially drawn from a uniform distribution, remaining unaffected by learning. The b_j is its bias value and $f(\cdot)$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems (HUANG *et al.*, 2006 apud HUANG, 2014, p. 379), for example:

1. Sigmoid function:

$$f(\cdot) = \frac{1}{1 + \exp\left(a(\mathbf{m}_j^T \mathbf{u}(k)) + b_j\right)} \tag{2.3}$$

2. Hyperbolic tangent:

$$f(\cdot) = \frac{1 - \exp\left(-\left(a(\mathbf{m}_j^T \mathbf{u}(k)) + b_j\right)\right)}{1 + \exp\left(-\left(a(\mathbf{m}_j^T \mathbf{u}(k)) + b_j\right)\right)} \tag{2.4}$$

3. Fourier function:

$$f(\cdot) = \sin\left(\mathbf{m}_j^T \mathbf{u}(k) + b_j\right) \tag{2.5}$$

4. Hard-limit function:

$$f(\cdot) = \begin{cases} 1 & \text{if } \left(\mathbf{m}_j^T \mathbf{u}(k) + b_j\right) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

5. Gaussian function:

$$f(\cdot) = \exp\left(-b_j \|\mathbf{u}(k) - \mathbf{m}_j\|^2\right) \tag{2.7}$$

where a is the function's slope, usually ignored being set to 1.

Let the hidden neurons output matrix be described by:

$$\mathbf{H} = \begin{pmatrix} h_1(1) & \ldots & h_q(1) \\ \vdots & \ddots & \vdots \\ h_1(N) & \ldots & h_q(N) \end{pmatrix}_{N \times q} . \tag{2.8}$$

The output weights $\boldsymbol{\beta}$ is given by the solution of an ordinary least-squares (see Equation 2.9).

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{D}, \tag{2.9}$$

where $\mathbf{D} \in \mathbb{R}^{N \times r}$ is the desired output matrix. This linear system can simply be resolved by least-squares method, that in its batch mode, is computed by

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{D} \tag{2.10}$$

where $\mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. Different methods can be used to calculate Moore-Penrose generalized inverse of a matrix, such as orthogonal projection method, orthogonalization method, singular value decomposition (SVD) and also through iterative methods (HUANG, 2014). In this work, the SVD approach was adopted.

That being said, the algorithm that describes the procedure to adapt an ELM network is given in Algorithm 1.

Algorithm 1: Extreme Learning Machine

1: Training set $\{(\mathbf{u}_i, \mathbf{d}_i)\}_{i=1}^{N}$ with $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{d} \in \mathbb{R}^r$;
2: Randomly generate the hidden weights $\mathbf{M} \in \mathbb{R}^{p \times q}$ and biases $\mathbf{b} \in \mathbb{R}^q$;
3: Calculate the hidden layer output matrix $\mathbf{H}$;
4: Calculate the output weight vector $\boldsymbol{\beta}$ (see Equation 2.10);
5: **return** $\boldsymbol{\beta}$.

2.3 Issues and improvements for ELM

Even though ELM networks and its variants are universal function approximators, their generalization performance is strongly influenced by the network size, regularization strength, and, in particular, the features provided by the hidden layer (NEUMANN, 2013). One of the issues inherited from the traditional SLFN is how to obtain the best architecture. It is known that a network with few nodes may not be able to model the data, in contrast, a network with too many neurons may lead to overfitting (MARTÍNEZ-MARTÍNEZ *et al.*, 2011). Another shortcoming is that the random choice of hidden weights and biases may result in an ill-conditioned hidden layer output matrix, which derails the solution for the linear system used to train the output weights (WANG *et al.*, 2011; HORATA *et al.*, 2013). An ill-conditioned hidden layer output matrix also provides unstable solutions, where the small errors in the data will lead to errors of a much higher magnitude in the solution (QUARTERONI *et al.*, 2006, p. 150). This is reflected on the high norm of output weights, which is not desirable as discussed in the works of Bartlett (1998) and Hagiwara and Fukumizu (2008), where they stated that the size of output weight vector is more relevant for the generalization capability than the configuration of the neural network itself. This can be observed as the training error enlarges and the test performance deteriorates (WANG *et al.*, 2011). To deal with this, ELM uses SVD even though it is computationally consuming.

In the following subsections, we presented some works and proposed improvements for ELM network that concerns the aforementioned issues.

2.3.1 Tikhonov's regularization

A problem is well-posed if its solution exists, is unique and depends continuously on its input data, as defined by Hadamard (1902) apud Haykin (2008, chap. 7). However, in many learning problems, those conditions are usually not satisfied. Violations may be

encountered when there is not a distinct output for every input, when there is not enough information in the training set that allows a unique reconstruction of the input-output mapping, and the inevitable presence of noise or outliers in real-world data which brings uncertainty to the reconstruction process (HAYKIN, 2008, chap. 7).

To deal with this limitation, it is expected that some prior information about the mapping is available. The most common form of prior knowledge involves the assumption that the mapping's underlying function is smooth, i.e., similar inputs produce similar outputs (HAYKIN, 2008, chap. 7). Based on that, the fundamental idea behind regularization theory is to restore well-posedness by appropriate constraints on the solution, which contains both the data and prior smoothness information (EVGENIOUA *et al.*, 2002).

In 1963, Tikhonov introduced a method named Regularization (TIKHONOV, 1963 apud HAYKIN, 2008, chap. 7), which became state of the art for solving ill-posed problems. For the regularized least-squares estimator, its cost function for the i-th output neuron is given by

$$\mathscr{J}(\boldsymbol{\beta}_i) = \|\boldsymbol{\varepsilon}_i\|^2 + \lambda \|\boldsymbol{\beta}_i\|^2, \tag{2.11}$$

$$\|\boldsymbol{\varepsilon}_i\|^2 = \boldsymbol{\varepsilon}_i^T \boldsymbol{\varepsilon}_i = (\mathbf{d}_i - \mathbf{H}\boldsymbol{\beta}_i)^T (\mathbf{d}_i - \mathbf{H}\boldsymbol{\beta}_i). \tag{2.12}$$

where $\boldsymbol{\varepsilon}_i \in \mathbb{R}^N$ is the vector of error between the desired output and the network's output, and $\lambda > 0$ is the regularization parameter.

The first term of the regularization cost function is responsible for minimizing the error squared norm (see Equation 2.12), enforcing closeness to the data. The second term of this function minimizes the norm of the output weight vector, introducing smoothness, while λ controls the trade-off between these two terms.

Minimizing Equation 2.11, we obtain

$$\hat{\boldsymbol{\beta}}_i = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H}^T \mathbf{d}_i, \tag{2.13}$$

where $\mathbf{I} \in \mathbb{R}^{q \times q}$ is an identity matrix.

2.3.2 Architecture optimization

As previously mentioned, one of the issues of the ELM network is to determine its architecture. On one hand, few hidden neurons may not provide enough information processing power and, consequently, a poor performance. On the other hand, a large hidden layer may create a very complex system and lead to overfitting. To try to handle this issue, three main approaches are pursued (XU *et al.*, 2007; ZHANG *et al.*, 2012):

- *constructive methods* start with a small network and then gradually adds new hidden neurons until a satisfactory performance is achieved (HUANG *et al.*, 2006; HUANG; CHEN, 2008; HUANG *et al.*, 2008; FENG *et al.*, 2009; ZHANG *et al.*, 2011; YANG *et al.*, 2012);
- *destructive methods* also known as *pruning methods*, start by training a much larger network and then removes the redundant hidden neurons (RONG *et al.*, 2008; MICHE *et al.*, 2008; MICHE *et al.*, 2011; FAN *et al.*, 2014);
- *evolutionary computation* uses population-based stochastic search algorithms that are developed from the natural evolution principle or inspired by biological group behavior (YAO, 1999). Besides the adaptation of an architecture, they may also perform weight and learning rule adaptation, input feature selection, weight initialization, etc (YAO, 1999). More about architecture design with metaheuristics is further described in Chapter 4 and some metaheuristics algorithms are detailed in Appendix A.

Huang *et al.* (2006) proved that an incremental ELM, named I-ELM, still maintain its properties as universal approximator. In 2008, two approaches were proposed to improve I-ELM performance. In (HUANG *et al.*, 2008), the I-ELM was extended from the real domain to the complex domain with the only constraint that the activation function must be complex continuous discriminatory or complex bounded nonlinear piecewise continuous. Also, in 2008, Huang and Chen proposed an enhanced I-ELM, named EI-ELM. It differs from I-ELM by picking the hidden neuron that leads to the smallest residual error at each learning step. This resulted in a more compact architecture and faster convergence.

Still on constructive methods, the *Error Minimized Extreme Learning Machine* (EM-ELM) (FENG *et al.*, 2009) grows hidden neurons one by one or group by group, updating the output weights incrementally each time. Based on EM-ELM, Zhang *et al.* (2011) proposed the AIE-ELM that grows randomly generated hidden neurons in a way that the existing hidden neurons may be replaced by some of the newly generated ones

with better performance instead of keeping the existing ones. The output weights are still updated incrementally just as EM-ELM. Yang *et al.* (2012) proposed the *Bidirectional Extreme Learning Machine* (B-ELM), in which some neurons are not randomly selected and takes into account the relationship between the residual error and the output weights to achieve a faster convergence.

On pruning methods, we can mention the *Optimally Pruned Extreme Learning Machine* (OP-ELM) (MICHE *et al.*, 2008; MICHE *et al.*, 2010), that starts with a large hidden layer, then each hidden neuron is ranked according with a Multiresponse Sparse Regression Algorithm and, finally, neurons are pruned according to a leave-one-out cross-validation. TROP-ELM (MICHE *et al.*, 2011) came as an OP-ELM improvement, that adds a cascade of two regularization penalties: first a L_1 penalty to rank the neurons of the hidden layer, followed by a L_2 penalty on the output weights for numerical stability and better pruning of the hidden neurons. Finally, and most recently, the work of Fan *et al.* (2014) proposes an Extreme Learning Machine with $L_{1/2}$ regularizer (ELMR), that identifies the unnecessary weights and prunes them based on their norms. Due to use of the $L_{1/2}$ regularizer, it is expected that the absolute values of the weights connecting relatively important hidden neurons become fairly large.

Although constructive and pruning methods address the architecture design, they explore only a limited number of available architectures (XU; LU; HO, 2007; ZHANG et al., 2012). As mentioned above, evolutionary algorithms allows that not only the architecture (number of layers, number of neurons, connections, etc) but also activation functions, weight adaptation and others to be optimized. Because of this feature, we adopted this method in this work and will be further detailed in Chapter 4.

2.3.3 Intrinsic plasticity

In artificial neural networks field, the majority of learning developments focus on exploring forms of synaptic plasticity, i.e., the adaptation of synaptic weights. This specific plasticity is a mechanism for memory formation and is the prevalent one in a normal adult brain (LENT, 2008, p. 126). Notwithstanding, it is not the only form of plasticity: "other forms also play a critical role in shaping adaptive changes within the nervous system, including intrinsic plasticity – a change in the intrinsic excitability of a neuron."(SEHGAL *et al.*, 2013, p. 186).

Biologically, this phenomenon helps neurons maintain appropriate levels of electrical activity, by shifting the positions and/or the slopes of the response curves to make the sensitive regions of the response curves always correspond well with input distributions (LI, 2011). Even though it is still uncertain how the underlying processes work, there is experimental evidence suggesting that it plays an important role as part of the memory engram[1] itself, as a regulator of synaptic plasticity underlying learning and memory, and as a component of homeostatic regulation (CUDMORE; DESAI, 2008).

Baddeley et al (1997) (BADDELEY *et al.*, 1997 apud LI, 2011) performed an experimental observation where they concluded that neurons exhibit approximately exponential distributions of the spike counts in a time window. Based on this information, computational approaches were developed to study the effects of intrinsic plasticity (IP) on various brain functions and dynamics.

In a mathematical manner, their goal is to obtain an approximately exponential distribution of the firing rates. Since the exponential distribution has the highest entropy among all distributions with fixed mean (TRIESCH, 2005), those approaches attempt to maximize the information transmission while maintaining a fixed average firing rate, or equivalently, minimize their average firing rate while carrying a fixed information capacity (LI, 2011). As examples, we may cite: (BELL; SEJNOWSKI, 1995), (BADDELEY *et al.*, 1997), (STEMMLER; KOCH, 1999), (TRIESCH, 2004), (SAVIN *et al.*, 2010) and (NEUMANN, 2013).

Triesch (2005) proposed a gradient rule for intrinsic plasticity to optimize the information transmission of a single neuron by adapting the slope $\boldsymbol{a}$ and bias $\boldsymbol{b}$ of the logistic sigmoid activation function (see Equation 2.3) in a way that the hidden neurons' outputs become exponentially distributed. Based on his works (TRIESCH, 2004; TRIESCH, 2005; TRIESCH, 2007), Neumann and Steil (2011) proposed a method named *Batch Intrinsic Plasticity* (BIP) to optimize ELM networks' hidden layer. The difference between those two methods lies on the parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ estimation that, in BIP, the samples are presented all at once and the parameters are calculated in only one shot.

Since ELM has random hidden weights, they may lead to saturated neurons or almost linear responses which may also compromise the generalization capability (NEUMANN; STEIL, 2013). Nevertheless, this can be avoided using activation functions

[1]"A hypothetical permanent change in the brain accounting for the existence of memory; a memory trace."(DICTIONARIES, 2014)

that provides a suitable regime just as the IP proposition, acting also as a feature regularizer (NEUMANN; STEIL, 2013).

With this in mind, the BIP approach is accomplished by forcing the j-th hidden neuron activation with a logistic activation function (Equation 2.3) or with a hyperbolic tangent (Equation 2.4) into a desired exponential distribution f_{des}. For each hidden neuron, all the incoming synaptic sum $\mathbf{x}_j = \mathbf{m}_j^T \mathbf{U}$ is collected, where $\mathbf{U} = (\mathbf{u}(1), ..., \mathbf{u}(N))^T$. Then random virtual targets $\mathbf{T}_{f_{des}} = (t_1, ..., t_N)^T$ from the desired exponential output distribution and the collected stimuli $\mathbf{x}_j$ are both drawn in ascending order. Next, we build the data matrix $\boldsymbol{\Phi}(\mathbf{x}_j) = (\mathbf{x}_j^T, (1, ..., 1)^T)$ and the parameter vector $\mathbf{v}_j = (a_j, b_j)$. The solution for the optimal $\mathbf{v}_j$ is obtained by computing the Moore-Penrose pseudo inverse is given by

$$(a_j, b_j)^T = (\boldsymbol{\Phi}(\mathbf{x}_j)^T \boldsymbol{\Phi}(\mathbf{x}_j) + \lambda I)^{-1} \boldsymbol{\Phi}(\mathbf{x}_j)^T f^{-1}(\mathbf{t}_{f_{des}}), \tag{2.14}$$

where f^{-1} is the inverse of the activation function. Thus, $\lambda > 0$ is the regularization parameter and $I \in \mathbb{R}^{q \times q}$ is an identity matrix.

It is important to highlight that, in case that the logistic sigmoid is used, the virtual targets $\mathbf{T}_{f_{des}}$ are in $[0, 1]$. Due to this fact, only truncated probability distributions are applied (NEUMANN, 2013). The algorithm that describes the BIP procedure is given in Algorithm 2.

Algorithm 2: Batch Intrinsic Plasticity

1: **Input:** $\mathbf{U} = (\mathbf{u}(1), \mathbf{u}(2)...\mathbf{u}(N))^T$;
2: **for** each hidden neuron j **do**
3: $\quad \mathbf{x}_j = \mathbf{m}_j^T \mathbf{U}$; {harvest the synaptic sum, $i = 1, ..., q$}
4: $\quad \mathbf{t}_{f_{des}} = (t_1, ..., t_N)^T$; {desired outputs from a exponential distribution f_{des}}
5: $\quad$ Put in ascending order $\mathbf{x}_j$ and $\mathbf{t}_{f_{des}}$
6: $\quad$ Build the model $\boldsymbol{\Phi}(\mathbf{x}_j) = (\mathbf{x}_j^T, (1, ...1)^T)$
7: $\quad$ Calculate Equation 2.14
8: **end for**
9: **return** $\{\mathbf{a}, \mathbf{b}\}$.

2.4 Concluding remarks

The ELM network is an SLFN with random hidden weights, bias and analytically determined output weights. It offers several advantages such as fast learning, simple implementation, and good generalization capability. One can argue that it has been

already proposed in the literature, however, as presented in our brief survey, there are distinctions in how the input features are presented, architecture design and/or in the main goal of processing. A thorough review can be seen in (HUANG, 2014).

The main challenges when using this network consist in defining an optimal number of hidden neurons, work through computational issues due to ill-posed $\mathbf{H}$ and dealing with the inevitable presence of noise or outliers in real-world data. Designing ELM's architecture has been extensively studied in these few years of the network's existence. Even though many researchers focus only on performance errors, there are only a few number of works that concern regularization and size of the output weights. Nevertheless, the study of its robustness, especially to outliers, is still in its infancy.

Having this problem in mind, in Chapter 3 we propose a robust ELM that combines the regularization effect of intrinsic plasticity and robustness to outliers.

3 A ROBUST ELM/BIP

Real world machine learning problems are often contaminated by noise caused by measurement errors, human mistakes, measurements of members of wrong populations, rounding errors etc (BELIAKOV *et al.*, 2011; RUSIECKI, 2013). They may also be heavy-tailed, which results in different sample distribution than the desired normal one. These inaccuracies may influence the data basically in two ways: affecting the observations (outputs **d**), creating outliers; and/or corrupting the explanatory variables (input data **u**) which are then named leverage points (ROUSSEEUW; LEROY, 1987). Thereupon, outliers can be defined intuitively as data points inconsistent with the remainder of dataset (HUYNH *et al.*, 2008).

Regression outliers (either in **u** or **d**) pose a serious threat to standard least squares analysis, influencing modeling accuracy as well as the estimated parameters, as shown in the works of Khamis *et al.* (2005) and Steege *et al.* (2012).

Hence, when fitting a model to contaminated data, we should adopt one of these methods: *regression diagnostic* or *robust regression* (ROUSSEEUW; LEROY, 1987). The intent of diagnostic methods is to erase the outliers and then fit the "good" data by Ordinary Least Squares (OLS), whilst a robust regression first fits a model with a "resistant" method and then discovers the outliers as those points which have the large residuals from that robust solution (ROUSSEEUW; LEROY, 1987).

Even though outliers are considered error or noise, they can also contain important information or even be the most important samples in the set (BEN-GAL, 2005 apud BARROS, 2013). By simply removing these points, that do not seem to behave as the remaining ones, there is a chance of biasing your regression line towards some raised hypothesis. Besides, adopting a diagnostic method is especially harder when dealing with datasets that have many outliers.

From the exposed, we investigated here a robust regression method and related works with neural networks, particularly with ELM. This method weights the contribution of each sample error designated to calculate a solution, avoiding that large errors influence the final estimation.

In this chapter, we provide a numerical example in Section 3.1 that demonstrates how the presence of a single outlier perturbs the regression line solution. In Section 3.2 we describe one of the most known robust regression methods, named M-Estimators, its weight

functions (Subsection 3.2.1) and then exhibit a numerical example (Subsection 3.2.2). In Section 3.3 we make a brief review of robust neural networks and introduce the approaches that apply robust methods to ELM networks. Then, in Section 3.4, we present our method named R-ELM/BIP. Finally, in Section 3.5, we provide our final remarks on this chapter.

3.1 Outliers influence

The following case exemplifies an outlier impact over model estimation. This specific example was presented in the work of Barros (2013) and reproduced here with permission of this author.

Table 2 – Cigarette consumption dataset.

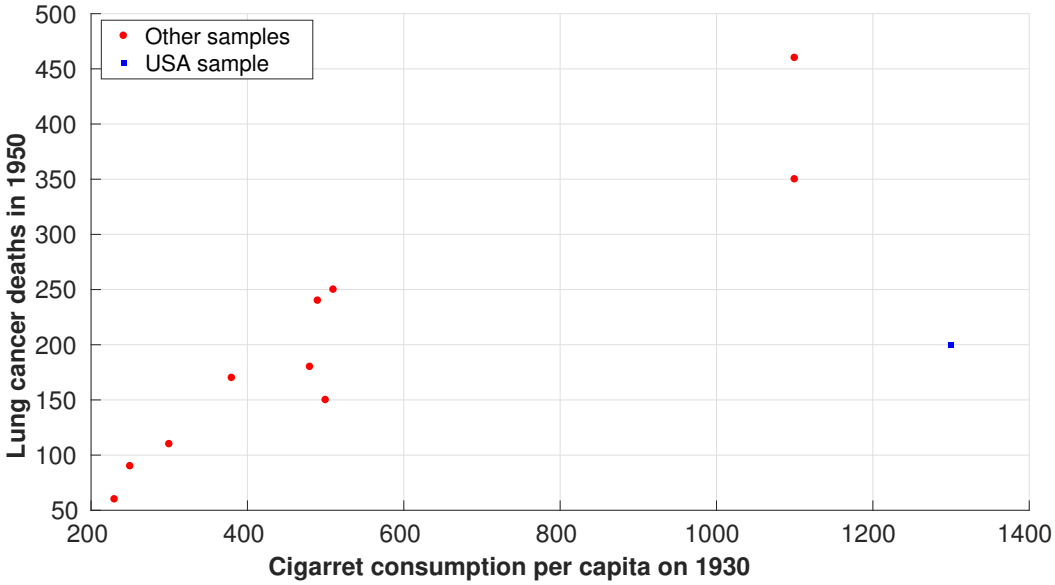
	Country	**Cigarette per capita**	**Deaths (millions)**
1	Australia	480	180
2	Canada	500	150
3	Denmark	380	170
4	Finland	1100	350
5	Great Britain	1100	460
6	Iceland	230	60
7	Holland	490	240
8	Norway	250	90
9	Sweden	300	110
10	Switzerland	510	250
11	**USA**	**1300**	**200**

Source: Barros (2013).

Table 2 shows the data corresponding to the set that associates the consumption of cigarettes per capita in 1930 in eleven different countries with lung cancer deaths (in million) that occurred in 1950. The sample related to the USA is recently added to this set and hence is highlighted. From the scatter plot in Figure 2, it is possible to observe that the majority of the data indicates a linear tendency, although this hypothesis is compromised with this new sample addition.

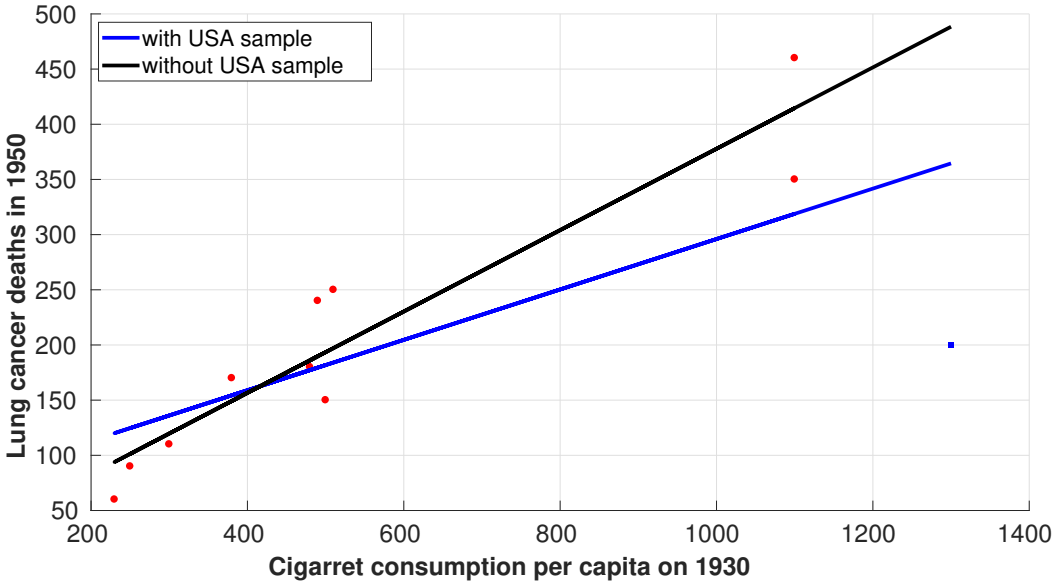
Figure 3 presents the two regression lines that result from the inclusion or not of the pair (1300, 200). It was used least squares algorithm to calculate them and it is clear that these two cases lead to different solutions. This new sample draws the regression line towards it, which alienates the line from a solution that could explain or even predict the number of deaths for a certain country.

Figure 2 – Scatter plot of Table 2 dataset.



Source: Barros (2013).

Figure 3 – Regression with and without the USA sample.



Source: Barros (2013).

Thinking of the linear system resulting from the estimation of ELM's output weights, the following sections discuss a robust linear regression method named *Maximum Likelihood Estimation*.

3.2 Fundamentals of Maximum Likelihood Estimation

The ordinary least squares (OLS) criterion has become the starting point for most introductory discussions of system modeling and is based on minimizing the squared error between the desired output **d** and the systems output **y**:

$$\mathscr{J}(\beta) = \sum_{n=1}^{N} \|\varepsilon\|^2 = \sum_{n=1}^{N} \|\mathbf{d}(n) - \mathbf{y}(n)\|^2. \tag{3.1}$$

However, its specific assumptions are ordinarily overlooked:

- we have an overdetermined system (more observations than output weights, in our case);
- the error ε is normally distributed with mean zero and unknown variance σ_ε^2;
- the observations $\{\varepsilon_i\}$ are uncorrelated.

The first assumption is easily fulfilled and the other two are usually supported by the Central Limit Theorem[1]. Nevertheless, as Huber said in his work: "Gauss was fully aware that his main reason for assuming an underlying normal distribution and a quadratic loss function was mathematical, i.e., computational convenience." (HUBER, 1964, p. 73 and 74). After all, real-world data are seldom so well-behaved. Besides the Central Limit Theorem can only explain that, in practice, we have distributions *approximately* normal and large samples are not always available.

In such situation, where the normality is failing at the tails[2], it is known that the OLS are far from optimal estimators (ANDREWS, 1974). Thus, one may ask whether it is possible to add robustness to the estimator by minimizing a different cost function than the sum of square errors.

It is worth noticing that the OLS criterion assigns the same importance to all errors (BARROS; BARRETO, 2013). Thereafter, outliers may produce large squared errors and then bias the solution towards the outliers locations, as demonstrated on Section 3.1.

An alternative was proposed by Huber (1964) and named *M-Estimation* ("M" for "maximum likelihood-type"). In short, it can be seen as a weighted mean, in which

[1] It states that the distribution of the sum, or average, of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.

[2] Heavy-tailed or fat-tailed distributions are the ones whose density tails tend to zero more slowly than the normal density tails (MARONNA *et al.*, 2006).

extreme error values are given reduced weight while those closer to the center of the distribution receive the largest weights.

A general M-estimator applied to the i-th output neuron of an ELM network minimizes the following cost function:

$$\mathscr{J}(\beta_i) = \sum_{n=1}^{N} \rho\left(\varepsilon_i(n)\right) = \sum_{n=1}^{N} \rho\left(d_i(n) - y_i(n)\right), \tag{3.2}$$

where the objective function $\rho(.)$ gives the contribution of each error $\varepsilon_i(n)$ to the cost function. It is interesting to notice that OLS is a particular M-estimator, achieved when $\rho\left(\varepsilon_i(n)\right) = \varepsilon_i^2(n)$.

The function ρ should possess the following properties:

- **Property 1**: $\rho(\varepsilon) \geq 0$;
- **Property 2**: $\rho(0) = 0$;
- **Property 3**: $\rho(\varepsilon) = \rho(-\varepsilon)$;
- **Property 4**: $\rho(\varepsilon(i)) \geq \rho(\varepsilon(j))$ for $|\varepsilon(i)| > |\varepsilon(j)|$.

In order to derive a learning rule, we differentiate the cost function with respect to the output weights β_i (that we will name here as *coefficients* for future distinction) by setting the partial derivatives to zero, as shown below:

$$\frac{\partial \mathscr{J}(\beta_i)}{\partial \beta_i} = \sum_{n=1}^{N} \frac{\partial \rho\left(\varepsilon_i(n)\right)}{\partial \varepsilon_i} \frac{\partial \varepsilon_i(n)}{\partial \beta_i} = \sum_{n=1}^{N} \psi(\varepsilon_i(n)) \frac{\partial \varepsilon_i(n)}{\partial \beta_i} = \mathbf{0}, \tag{3.3}$$

where the derivative $\psi = d\rho(\varepsilon)/d\varepsilon$ is named *influence function* and $\mathbf{0}$ is a $(q+1)$ dimensional row vector of zeros. It is interesting to notice that the influence function $\psi(\varepsilon)$ measures the influence of the error on the parameter estimation value. For example, for the least squares, $\rho(\varepsilon) = \varepsilon^2/2$ and $\psi(\varepsilon) = \varepsilon$, i.e. the influence of an error increases linearly with its size, which confirms the non-robustness of least squares estimation (ZHANG, 1997).

Based on this, the weight function is defined as

$$w(\varepsilon_i) = \frac{\psi(\varepsilon_i)}{\varepsilon_i}, \tag{3.4}$$

and then Equation 3.3 becomes

$$\sum_{n=1}^{N} w(\varepsilon_i(n))\varepsilon_i(n) \frac{\partial \varepsilon_i}{\partial \beta_i} = \mathbf{0}. \tag{3.5}$$

This is exactly the system of equations that we obtain if we solve a weighted least-squares problem, by minimizing $\sum_{n=1}^{N} w_i^2(n)\varepsilon_i^2(n)$ (ZHANG, 1997). Nonetheless, the

weights depend upon the errors, the errors depend upon the estimated coefficients and the estimated coefficients depend upon the weights (FOX, 2002). As a consequence, a closed-form equation for estimating β_i is not available. Therefore, an iterative solution, named *Iteratively Reweighted Least Squares* (IRWLS), is required (see Algorithm 3).

Algorithm 3: Iteratively Reweighted Least Squares

1: Provide an initial estimate of $\beta_i(0)$ using OLS (Equation 2.10);
2: **for** $t = 1$ until $\beta_i(t)$ converges **do**
3: Compute the errors $\varepsilon_{in}(t-1)$ associated with the i-th output neuron and $n = 1,...,N$.
4: Define weights $w_{in}(t-1) = w[\varepsilon_{in}(t-1)]$, creating $\mathbf{W}_i(t-1) = diag\{w_{in}(t-1)\}$
5: Solve for new weighted-least-squares estimate of $\beta_i(t)$:

$$\beta_i(t) = \left[\mathbf{H}^T\mathbf{W}_i(t-1)\mathbf{H}\right]^{-1}\mathbf{H}^T\mathbf{W}_i(t-1)\mathbf{d}_i \tag{3.6}$$

6: t = t+1;
7: **end for**
8: **return** β.

3.2.1 Objective and Weighting Functions for M-Estimators

In order to apply the IRWLS algorithm, it is imperative that the user chooses an objective function ρ. Nine of the most common functions for M-estimators are Andrews, Bisquare, Cauchy, Fair, Huber, Logistic, OLS, Talwar, and Welsch (see Table 3). This flexible choice of functions leads to different estimators, but independently of ρ, there is a positive parameter k, known as error (or outlier) threshold or tunning constant, that must also be defined.

The programming environment Matlab provides a function for robust estimation, named *ROBUSTFIT*, that has not only a list of several objective functions and their respective weight functions but also gives, if necessary, default values for the parameter k with the same aforementioned property. The objective and weight functions and their respective k are presented in Table 3 and demonstrated in Figures 4 and 5 with $k = 1$ respectively.

Observing the weight functions in Figure 5, it is possible to understand how this parameter influences the solution. Taking Huber (Figure 5e) as an example, errors within the defined error threshold ($|\varepsilon| \leq k$) are treated just as OLS, where they all contribute to the final estimation with weight $w(\varepsilon) = 1$. The difference lies at the boundary beyond k,

Table 3 – Objective functions and their respective weight functions and default thresholds provided by Matlab.

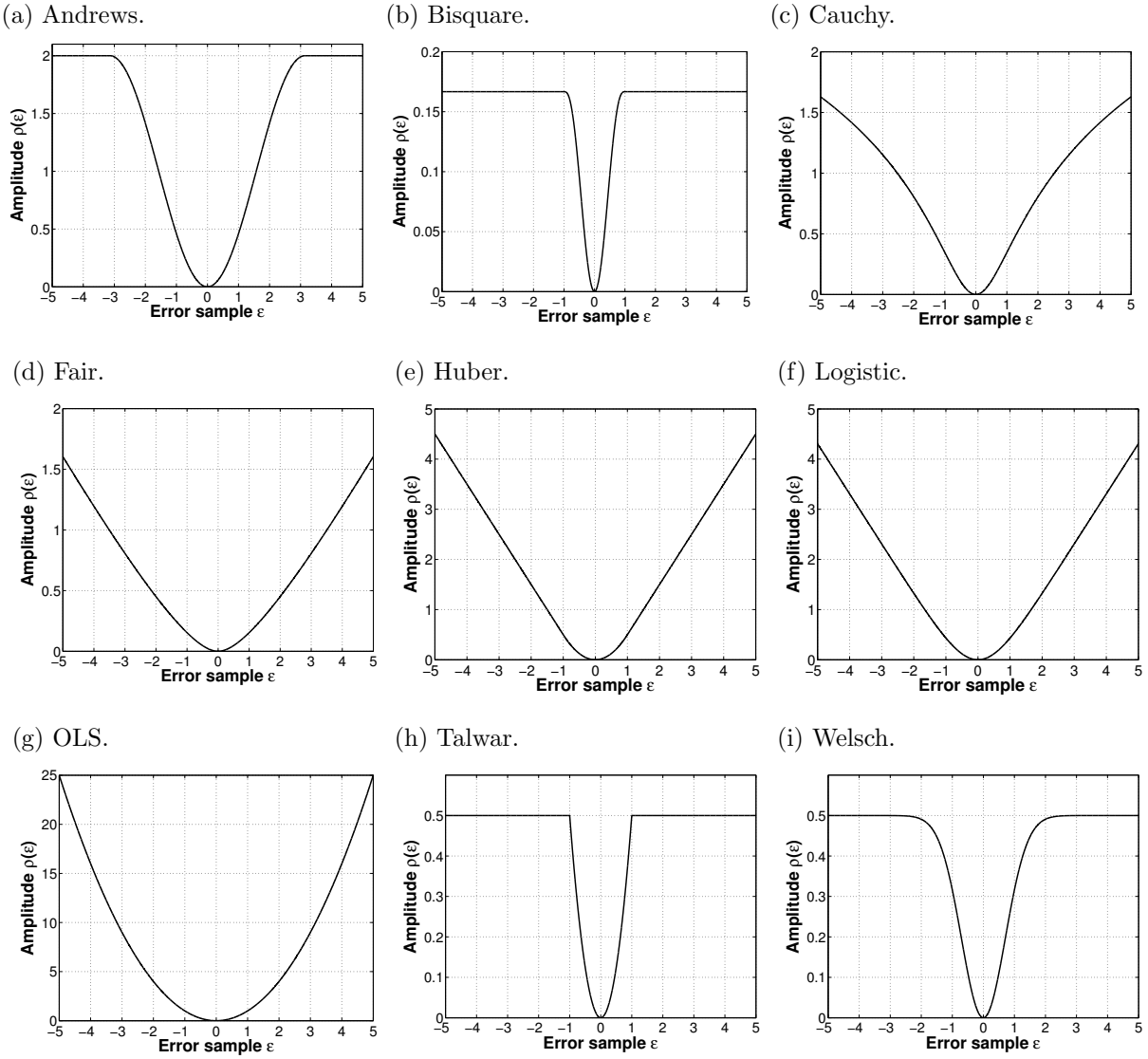
Name	**Objective Function** (ρ)	**Weight Function** (w)	**Threshold**(k)
Andrews	$\begin{cases} k^2\left[1-\cos\left(\frac{\varepsilon}{k}\right)\right], & \left\|\frac{\varepsilon}{k}\right\| \leq \pi \\ 2k^2, & \left\|\frac{\varepsilon}{k}\right\| > \pi \end{cases}$	$\begin{cases} \left(\frac{\varepsilon}{k}\right)^{-1}\sin\left(\frac{\varepsilon}{k}\right), & \left\|\frac{\varepsilon}{k}\right\| \leq \pi \\ 0, & \left\|\frac{\varepsilon}{k}\right\| > \pi \end{cases}$	1.339
Bisquare	$\begin{cases} \frac{k^2}{6}\left\{1-\left[1-\left(\frac{\varepsilon}{k}\right)^2\right]^3\right\}, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ \frac{k^2}{6}, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	$\begin{cases} \left[1-\left(\frac{\varepsilon}{k}\right)^2\right]^2, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ 0, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	4.685
Cauchy	$\frac{k^2}{2}\log\left(1+\left(\frac{\varepsilon}{k}\right)^2\right)$	$\frac{1}{1+\left(\frac{\varepsilon}{k}\right)^2}$	2.385
Fair	$k^2\left[\frac{\lvert\varepsilon\rvert}{k}-\log\left(1+\frac{\lvert\varepsilon\rvert}{k}\right)\right]$	$\frac{1}{1+\frac{\lvert\varepsilon\rvert}{k}}$	1.400
Huber	$\begin{cases} \frac{\varepsilon^2(n)}{2}, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ k\lvert\varepsilon\rvert-\frac{k^2}{2}, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	$\begin{cases} 1, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ \frac{k}{\lvert\varepsilon\rvert}, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	1.345
Logistic	$k^2\log\left[\cosh\left(\frac{\varepsilon}{k}\right)\right]$	$\left(\frac{\varepsilon}{k}\right)^{-1}\tanh\left(\frac{\varepsilon}{k}\right)$	1.205
OLS	$\varepsilon^2(n)$	1	–
Talwar	$\begin{cases} \frac{\varepsilon^2(n)}{2}, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ \frac{k^2}{2}, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	$\begin{cases} 1, & \left\|\frac{\varepsilon}{k}\right\| \leq 1 \\ 0, & \left\|\frac{\varepsilon}{k}\right\| > 1 \end{cases}$	2.795
Welsch	$\frac{k^2}{2}\left[1-\exp\left(-\left(\frac{\varepsilon}{k}\right)^2\right)\right]$	$\exp\left(-\left(\frac{\varepsilon}{k}\right)^2\right)$	2.985

Source: Barros and Barreto (2013).

where those weight values decrease as the absolute error values $|\varepsilon|$ increase.

Smaller values of k produce more resistance to outliers, but at the expense of lower efficiency when the errors are normally distributed (BARROS; BARRETO, 2013). This error threshold is usually chosen to provide reasonably high efficiency in the normal case (FOX, 2002). For example, $k = 1.345\sigma$ for the Huber objective function and $k = 4.685\sigma$ for Bisquare, where σ is the standard deviation of the errors. Those values give coefficient estimates that are approximately 95% as statistically efficient as the ordinary

Figure 4 – Different objective functions $\rho(\varepsilon)$ with $k = 1$.



Source: author.

least squares estimates, provided that the response has a normal distribution with no outliers (FOX, 2002).

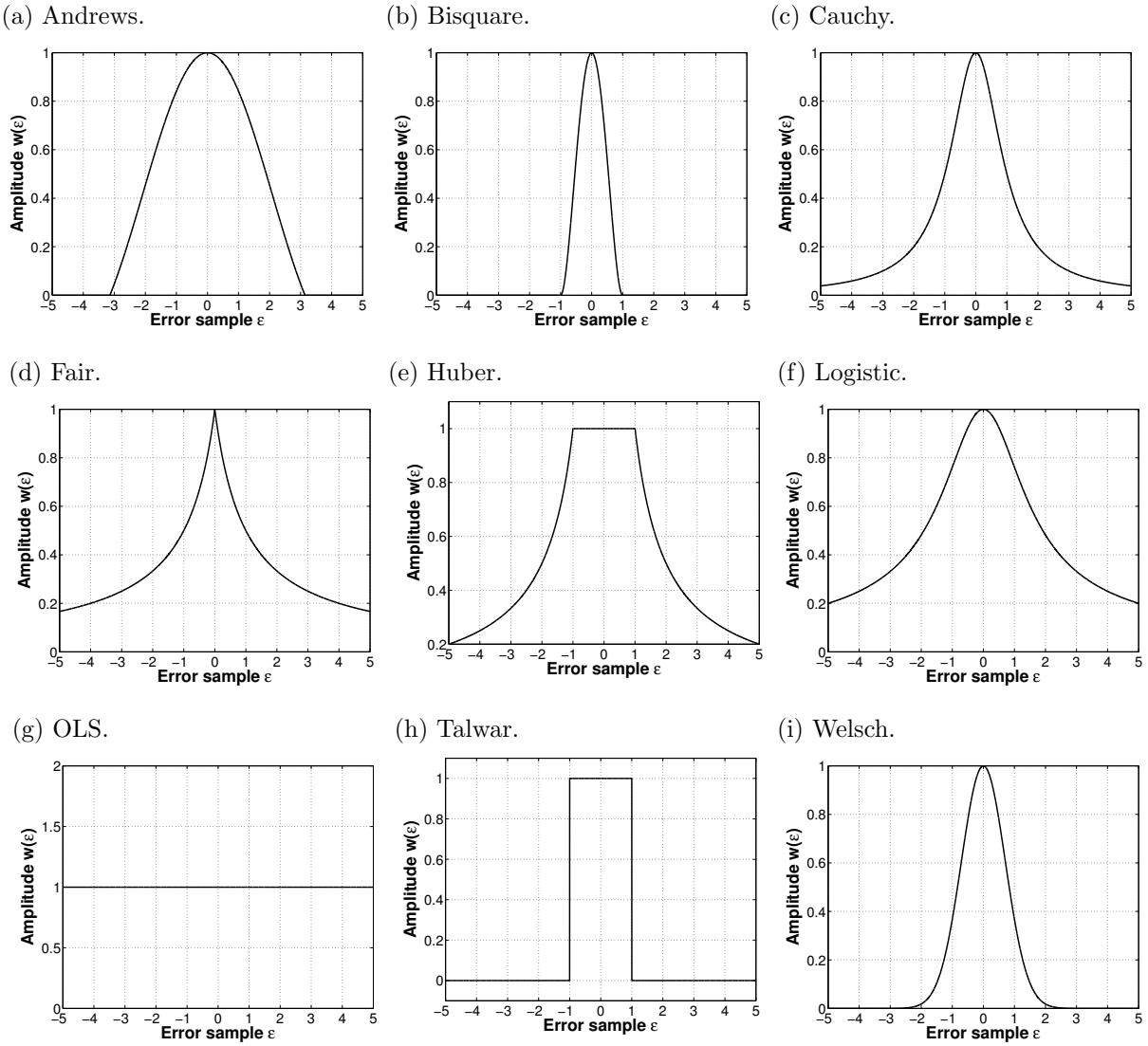
It is important to highlight that σ must also be estimated. Ordinarily, a robust estimation for *sigma* is the *normalized median absolute deviation* (*MADN*) described in Equation 3.7.

$$\hat{\sigma} = MADN(\varepsilon) = \frac{Med\left(|\varepsilon - Med(\varepsilon)|\right)}{0.6745}, \tag{3.7}$$

where $Med(.)$ is the median and the constant value 0.6745 makes $\hat{\sigma}$ an unbiased estimate for Gaussian errors (BARROS, 2013).

Another point to be highlighted refers to error treatment. As Barros and

Figure 5 – Different weight functions $w(\varepsilon)$ with $k = 1$.



Source: author.

Barreto (2013) mentioned, textbook equations of the weight functions shown in Table 3 are written using directly the raw error $\varepsilon_i(n)$ as argument. Notwithstanding, in practical applications, it is highly recommended to use standardized errors $e_i(n)$ instead (STEVENS, 1984), which are then computed as:

$$e_i(n) = \frac{\varepsilon_i(n)}{\hat{\sigma}\sqrt{1-h_{nn}^*}}, \tag{3.8}$$

where h_{nn}^*, with $0 \leq h_{nn}^* \leq 1$, is the n-th entry of the main diagonal of the hat matrix $\mathbf{H}^* = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, with the matrix $\mathbf{H}$ defined as in Equation 2.8.

In Matlab's implementation, they adopted the following equation:

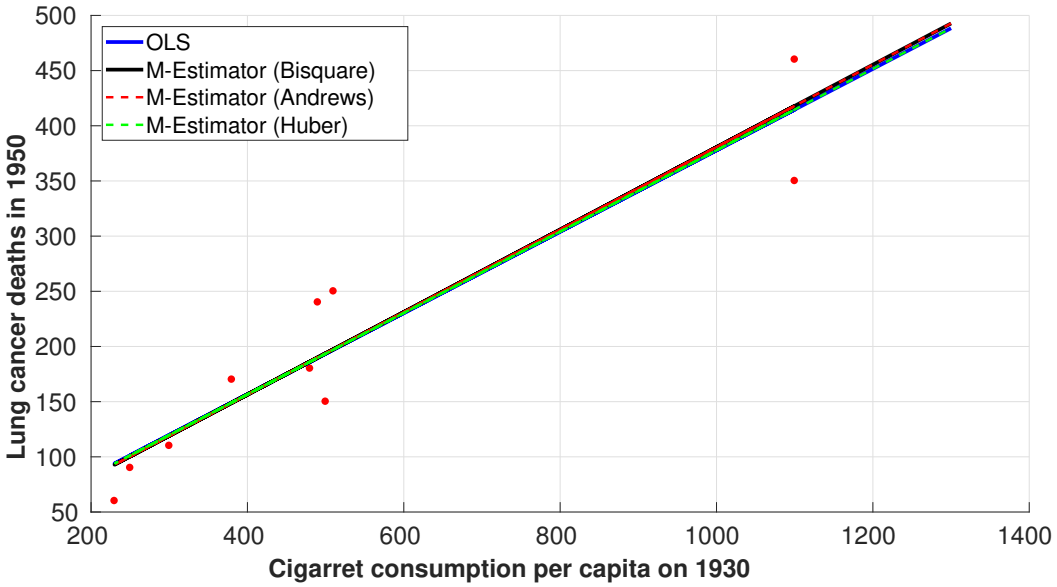
$$e_i(n) = \frac{\varepsilon_i(n)}{k\hat{\sigma}\sqrt{1-h_{nn}^*}}. \tag{3.9}$$

3.2.2 Numerical example

In order to observe the outlier robustness of M-Estimators, we take the same numerical example given in Subsection 3.1. Adopting the *ROBUSTFIT* tool provided by Matlab, the resulting regression lines are displayed in Figures 6 and 7 and are calculated using M-Estimators with OLS, Bisquare, Andrews and Huber objective functions.

In Figure 6, the data applied did not have the USA sample, i.e., it did not contain an outlier. We can observe that the regression lines almost overlap each other. Also by observing Table 4, that displays the provided weights for each sample error, we may notice that OLS and Huber had the same behavior with weight equal to one, while the other functions offered high values.

Figure 6 – Regression without USA outlier.

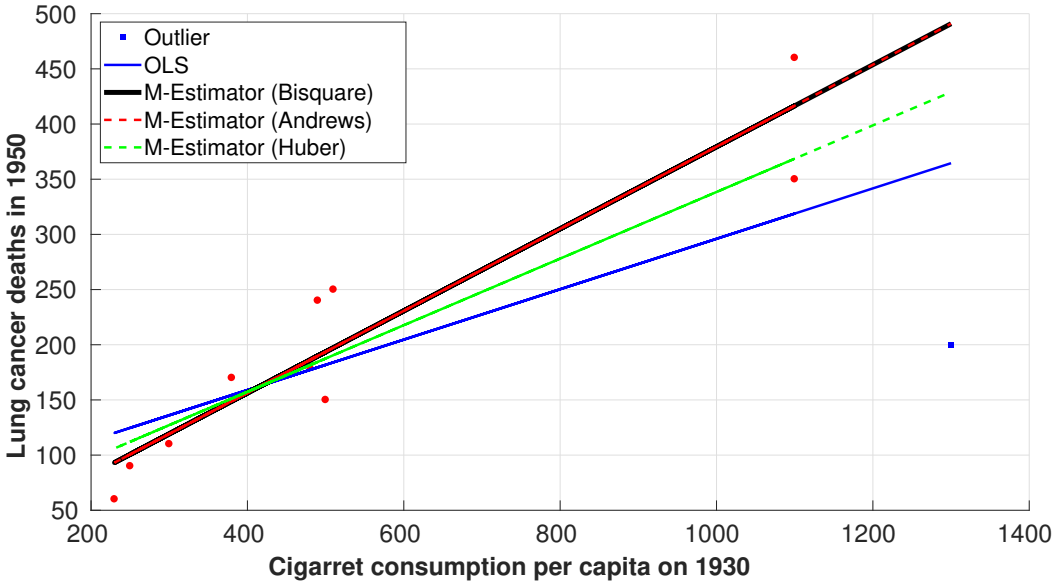


Source: author.

Figure 7 shows the resulting lines when there is an outlier in the data. Adding the USA sample, we may observe that the OLS line has been pulled towards this spurious sample, as expected. Moreover, the other M-Estimators are less affected by this sample. On one hand, the Bisquare and Andrews functions ignored it completely, as demonstrated in Table 4, where the respective weight is equal to zero. Huber's resulting line, on the other hand, was affected slightly with an approximately 0.3 weight. Even though, Huber's line was still more robust than the OLS alternative.

On the following sections, we will make a very brief review of robust neural

Figure 7 – Regression with USA outlier.



Source: author.

Table 4 – Weight values for different M-Estimators with Cigarette dataset.

Country	**Weights (without outlier)**				**Weights (with outlier)**			
	OLS	Bisquare	Andrews	Huber	OLS	Bisquare	Andrews	Huber
Australia	1	0.99914	0.99912	1	1	0.99926	0.99924	1
Canada	1	0.95897	0.95824	1	1	0.96286	0.96216	1
Denmark	1	0.99017	0.98997	1	1	0.99099	0.99081	1
Finland	1	0.8416	0.83959	1	1	0.89577	0.8941	1
Great Britain	1	0.9379	0.937	1	1	0.9549	0.95414	0.84893
Iceland	1	0.97379	0.97333	1	1	0.97594	0.97549	1
Holland	1	0.94674	0.94583	1	1	0.95081	0.94992	1
Norway	1	0.99742	0.99738	1	1	0.99755	0.9975	1
Sweden	1	0.9981	0.99807	1	1	0.99821	0.99817	1
Switzerland	1	0.94139	0.94042	1	1	0.94589	0.94493	1
USA	–	–	–	–	**1**	**0**	**0**	**0.29814**

Source: author.

networks and then present robust ELM solutions available in the literature.

3.3 Robustness in ELM networks

Robust neural networks have been a subject of interest for many years in different applications. Works such as Liu (1993) shows that the conventional back-propagation algorithm for neural network regression is robust to leverages, but not to outliers. Larsen *et al.* (1998) proposed a neural network optimized using the maximum a *posteriori* technique with a modified likelihood function which incorporates the potential risk of outliers in the

data. Lee *et al.* (2009) proposed a Welsch M-estimator radial basis function with pruning and growing techniques for noisy time series prediction. Łobos *et al.* (2000) presented on-line techniques for robust estimation of parameters of harmonic signals based on the total least-squares criteria, which can be implemented by analogue adaptive circuits. Feng *et al.* (2010) propose an algorithm for the neural network quantile regression adopted from a Majorization-Minimization algorithm for optimization and applied it on an empirical analysis of credit card portfolio data. Aladag *et al.* (2014) propose a median neuron model multilayer feedforward (MNM-MFF) model, trained with a modified particle swarm optimization metaheuristic, in order to deal with forecasting performance problems caused by outliers.

The study of robustness with ELM network is still in its beginning. As pointed out by Horata *et al.* (2013), two main aspects influence the robustness properties in a ELM network: *computational robustness* and *outlier robustness*.

The computational robustness is related to the ability of ELM to compute its output weights $\boldsymbol{\beta}$, even if $\mathbf{H}$ is not full rank or ill-conditioned (HORATA; CHIEWCHAN-WATTANA; SUNAT, 2013). This property has been usually ignored since many efforts emphasize on solutions accuracy only (ZHAO *et al.*, 2011). The hidden layer output matrix $\mathbf{H}$ may be ill-conditioned, as mentioned in Section 2.3, due to the random input weights and biases selection. This results in a solution with high norms and, consequently, sensitive to any data perturbation, which becomes a poor estimation to the truth (ZHAO *et al.*, 2011).

Besides, it is known that the size of the output layer weight is more relevant for the generalization capability than the configuration of the neural network, in terms of the number of neurons and format of activation function (KULAIF; ZUBEN, 2013; BARTLETT, 1998). Works such as (KULAIF; ZUBEN, 2013), (DENG *et al.*, 2009), (MARTÍNEZ-MARTÍNEZ et al., 2011) and (WANG; CAO; YUAN, 2011) explore this specific issue.

The second aspect, related to outliers robustness, has been explored in recent years within a few development proposals, using estimation methods that are known for being less sensitive to outliers than the OLS. Works such as (HUYNH *et al.*, 2008) substitutes the Singular Value Decomposition method by the Weighted Least Squares, though he did not detail which objective function he adopted. Another example is the

one proposed by Barros and Barreto (2013), who concentrate their efforts on robust classification problems with a proposal of an ELM, named *Robust ELM* (RELM), that applied IRWLS with Andrews weight function to estimate its output weights. Finally, one of the main references to this issue belongs to Horata *et al.* (2013), who addressed both robustness problems. They replace the SVD method for the *Extended Complete Orthogonal Decomposition* (ECOD) to overcome the not full column rank problem as well as the ill-conditioning. This method is then allied to the three proposed iterative algorithms to improve ELM's outlier robustness: IRWLS, the Multivariate Least-Trimmed Squares (MLTS) estimator and the One-Step Reweighted MLTS (RMLTS).

3.4 R-ELM/BIP

Based on the two robustness issues that Horata *et al.* (2013) described, we propose a new method that deals with both problems. We combine the regularizer effect and learning optimization property of Batch Intrinsic Plasticity (see Section 2.3.3) with the outlier robustness of the M-estimation framework.

The BIP method provides that, independently of the chosen hidden weights, we will have an optimized hidden layer which is then responsible for a stable output layer solution. This solution has usually a small Euclidean norm, which improves the generalization capability, as proven in works of Bartlett (1998) and Hagiwara and Fukumizu (2008).

This manner, we tackle efficiently the ELM's computational problem. Finally, inspired by the works of Horata *et al.* (2013) and Barros and Barreto (2013), we deal with outlier robustness by adopting the IRWLS algorithm, along with Bisquare objective function and Matlab's default error threshold, to estimate the output weights $\boldsymbol{\beta}$.

This new method is then named *Robust ELM with Batch Intrinsic Plasticity* (R-ELM/BIP) and the steps for its implementation follows in Algorithm 4.

3.5 Concluding remarks

Noisy datasets are almost a certainty when dealing with real-world problems. Issues related to human mistakes, measurement errors, noise with other distribution than the normal one etc, may cause the appearance of outliers. As extensively mentioned in

Algorithm 4: R-ELM/BIP

Training set $\mathbf{U} = \{(\mathbf{u}_i, \mathbf{d}_i)\}_{i=1}^{N}$ with $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{d} \in \mathbb{R}^r$;
Randomly generate the hidden weights $\mathbf{M} \in \mathbb{R}^{p \times q}$
for each hidden neuron j **do**
    $\mathbf{x}_j = \mathbf{m}_j^T \mathbf{U}$; {harvest the synaptic sum, $i = 1, ..., q$}
    $\mathbf{t}_{f_{des}} = (t_1, ..., t_N)^T$; {desired outputs from a exponential distribution f_{des}}
    Put in ascending order $\mathbf{x}_j$ and $\mathbf{t}_{f_{des}}$
    Build the model $\Phi(\mathbf{x}_j) = (\mathbf{x}_j^T, (1, ...1)^T)$
    Calculate Equation 2.14
end for
Re-introduce the training input data to the network and collect the network states $\mathbf{H}$;
Calculate an initial estimate of $\beta_i(0)$ using OLS (Equation 2.10);
for $t = 1$ until $\beta_i(t)$ converges **do**
    Compute the errors $\varepsilon_{in}(t-1)$ associated with the i-th output neuron and $n = 1, ..., N$.
    Define weights $w_{in}(t-1) = w[\varepsilon_{in}(t-1)]$, creating $\mathbf{W}_i(t-1) = diag\{w_{in}(t-1)\}$
    Solve for new weighted-least-squares estimate of $\beta_i(t)$ (see Equation 3.6);
    t = t+1;
end for
return β.

this chapter and demonstrated in Subsections 3.1 and 3.2.1, these spurious samples derail the OLS solution, which has no resource to differentiate a sample from an outlier.

To deal with such situation, we adopted the robust regression approach named M-Estimators, where the outliers are not necessarily excluded from the solution calculation, but rather has its influence diminished with a chosen objective function.

Robust regression with neural networks have been explored in the literature, although, concerning ELM architecture, there are only a few to count for. From the main reference on this matter, Horata *et al.* (2013) suggest that the ELM robustness property depends on two factors: computational robustness and outlier robustness.

Based on that, our proposed R-ELM/BIP adopts a robust regression estimator allied to the maximization of the hidden layer's information transmission, provided by BIP. This set up generates a reliable solution in the presence of outliers with a good generalization capability and small output weight norms.

In the following chapter, we deal with a different challenge for ELM networks: the architecture design. Our proposal, based on metaheuristics, provides the number of hidden neurons along with the respective activation functions parameters, which are all evolved through the iterations.

4 A NOVEL MODEL SELECTION APPROACH FOR THE ELM

The challenge of ANN's architecture design has been dealt for many years in a special class of neural networks named *Evolutionary Artificial Neural Networks* (EANN), in which evolution is, besides the learning, another fundamental form of adaptation (YAO, 1999; DING *et al.*, 2013). They can be regarded as a general framework for adaptive systems (YAO, 1999) once it can be used to optimize not only the network's topology, but also train synaptic weights, adapt learning rules, initialize weights and extract features.

Motivated by the ELM's architecture definition problem and inspired by those algorithms, we proposed a method based on metaheuristics that evolve not only the number of hidden neurons but also the activation function parameters: slope and bias. Named *Adaptive Number of Hidden Neurons Approach* (ANHNA), it is outlined as a general solution encoding for an individual (chromosome or particle) of a population followed by some specific constraints that allows any metaheuristic algorithm to evolve a solution for an ELM network, without losing its main characteristic: the random projections from the input units to the hidden neurons..

This chapter is organized as follows: in Section 4.1, metaheuristics are defined. In Section 4.2, it is made a brief review of the literature on how metaheuristics are being applied to ELM networks, some EANN that evolve their topologies and the few works that evolve topologies for ELM. In Section 4.3 we discuss the types of chromosome's encoding for architecture design using evolutionary algorithms. In Section 4.4, our proposal is discussed in detail. Subsection 4.4.1 presents ANHNA's seven fitness function variants and, in Subsection 4.4.2, it is discussed the application of this method to robust ELM networks. Lastly, Section 4.5 presents the final remarks of this chapter.

4.1 Introduction to metaheuristics

The nature-inspired heuristics, known as Evolutionary Algorithms, are based on some abstractions of the natural process of problem-solving (YANG, 2010). Since the beginning of time, nature has found its way to prevail over the adversities and one of its processes is described by Darwin's evolutionary theory, whose main concept is the survival of the fittest. In natural evolution, survival is achieved through reproduction. Two individuals combine their genetic material, in hope that their best characteristics are

passed on to the generated offspring. The weaker ones, as result of bad genes combination, will struggle to survive and most likely die. Another source of inspiration came from the study of swarms of social organisms (e.g. birds, fishes and ants) and how each individual's behavior, along with the limited knowledge of its neighbors' states, influences the goals achievement of the whole swarm.

The main feature in common with all phenomena that serve as an inspiration to metaheuristics is the use of a population of individuals (ENGELBRECHT, 2007). Metaheuristic algorithms can be classified as population-based or trajectory-based (YANG, 2010). Population-based algorithms are part of an area named Evolutionary Computation, which adopts the process of natural evolution and Darwin's theory to find an optimal solution for a specific problem. Each individual is an alternative to the solution and it is represented by a chromosome, which defines the individual's characteristics. As examples, there are Genetic Algorithms (GA), Genetic Programming, Evolutionary Programming and Differential Evolution. In trajectory-based algorithms, each individual performs a search in the space of an objective function by adjusting the trajectories towards the individuals with the best performance. Each individual is also an alternative solution and is seen as a position in search space. In nature, for instance, this behavior could be translated into the effort of a fish swarm finding sources of food and how they would group in the area that has the biggest concentration of food as soon as one of the fishes finds it. As examples of trajectory-based algorithms, we mention Particle Swarm Optimization and Ant Colony Optimization.

Apart from its classification, metaheuristics have two major components: selection of the best solutions and randomization (YANG, 2010). Selection ensures that the solutions will converge towards the optimality, while the randomness prevents the solutions to be trapped in local optima and, at the same time, increases the diversity of the solutions (YANG, 2010). The combination of these two components will usually ensure that the global optimality is achievable (YANG, 2010, p. 22).

ANNs have been extensively using metaheuristics to optimize their parameters (weights, connections, architecture, learning constants) in the literature. In this work, we adopt a few metaheuristic algorithms, detailed in Appendix A, to optimize an ELM's number of hidden neurons and activation function parameters at the same time. Through the remaining sections, we will discuss works published concerning ELM optimization

using metaheuristics, how the solutions are usually encoded for architecture optimization and, finally, our proposal of a general problem encoding to achieve the aforementioned goal.

4.2 Related works

The optimization of the ELM network using metaheuristic algorithms is an ongoing theme in several works over the past few years. Many of them have as goal finding suitable input weight values, usually maintaining the estimation of the output weights as the original ELM method. Works such as the *Evolutionary ELM* (E-ELM) (ZHU *et al.*, 2005) uses the Differential Evolution (STORN; PRICE, 1995; STORN; PRICE, 1997) to search for optimal hidden weights. The *Self Adaptive Evolutionary ELM* (CAO *et al.*, 2012) is similar to the E-ELM, although it allows automatic selection of different trial vector generation strategies and control parameters. Han *et al.* (2013) proposed an ELM network combined with an improved Particle Swarm Optimization to optimize the hidden weights and biases. The *Group Search Optimization ELM* (SILVA *et al.*, 2011) also aims at optimizing the hidden weights and biases. And more recently, the work from Figueiredo and Ludermir (2014) investigated the use of 8 topologies on the performance of a Particle Swarm Optimization ELM (XU; SHU, 2006). Unfortunately, this kind of approach demands that the number of hidden neurons must be determined beforehand by exhaustive search.

As already stated in Chapter 2, the design of an architecture is crucial for any successful SLFN. In the field of Evolutionary Artificial Neural Networks (EANN), there are works that evolve the topology of the networks as well. As examples, we can cite the work by Maniezzo (1994) that proposed the *Artificial Neural Networks Adaptation: Evolutionary Learning of Neural Optimal Running Abilities* (ANNA ELEONORA). It is based on a parallel genetic algorithm with an enhanced coding to evolve both the topology and all synaptic weights. This coding promotes a variable length binary gene where its first part indicates the existence/absence of a connection and the second part provides the value of the weights. Angeline *et al.* (1994) proposed the *Generalized Acquisition of Recurrent links* (GNARL) that evolves both architecture (number of hidden neurons, connections to and within the hidden layer, and connection from this layer to the output one) and weights for recurrent networks. Yao and Liu (1997) developed the EPNet, based on Fogel's

evolutionary programming, that evolves the architecture and weights from a feedforward neural network with sigmoidal activation function. Differently, from other methods, this one attempts neuron or connection elimination before addition within different mutations sequentially executed. Abbass (2002) published one of the few works that do not make use of genetic algorithms and called it *Memetic Pareto Artificial Neural Network* (MPANN). He adopted a method named Pareto-frontier Differential Evolution (ABBASS *et al.*, 2001 apud ABBASS, 2002) to evolve connections and number of hidden neurons of an EANN. Tsai *et al.* (2006) proposed a hybrid Taguchi-genetic algorithm to evolve the number of hidden neurons and connections in an SLFN, by increasing them from a small network until the learning performance is acceptable.

For ELM networks, evolutionary architecture design is still an area to be explored since there are still few published works. As examples, we can cite Matias *et al.* (2013) who proposed a *Genetically Optimized Extreme Learning Machine* (GO-ELM) that realizes a simultaneous evolution of architectures and connection weights. They applied a genetic algorithm to optimize the hidden weights, the number of hidden neurons and their activation function. The output weights are calculated using least squares with Tikhonov's regularization. Alencar and Rocha Neto (2014) proposed a genetic algorithm to prune ELM networks named AG-ELM for classification problems, where each binary chromosome indicates the number of hidden neurons by the number of genes equal to 1. Its fitness function is a trade-off between the classification error reduction and pruning importance, which is regulated by a parameter α.

There are some observations to be made about both works. In GO-ELM, since the hidden weights are adapted, it results in a mischaracterized ELM. As discussed in Chapter 2, the main characteristic of this network is a random choice of the hidden layer weights and once the authors evolve them, the resulting network becomes an evolutionary SLFN. About the AG-ELM, the authors did not make a fair comparison between the proposed method and a heuristic search, since the executed heuristic search started with 20 hidden neurons. This number is far higher than the one found by their method and it is also incremented in steps of 20 to 500 neurons. Being so, it is hardly possible to evaluate the real improvement of this proposal over a common design method such as cross-validation.

From the exposed so far, we identified three main points. The first one concerns

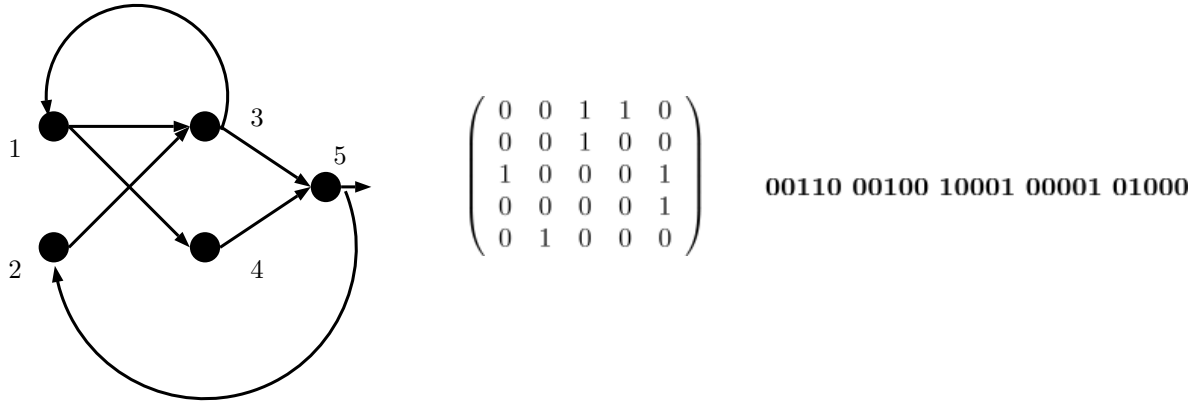
keeping the ELM's random nature of the input-to-output mapping, without any further adaptation of the input-to-hidden layer weights, so this network does not lose its main feature. The second one refers to the almost exclusive use of the genetic algorithm and its variants in literature. This leads us to the third point, which refers to chromosome encoding over the architectural information that we wish to evolve.

On the following sections, a brief introduction of the types of architecture optimization encoding schemes is given and our approach ANHNA is presented.

4.3 Encoding schemes for architecture evolution

The development of architecture optimization through metaheuristics is mainly reflected in how the architecture is coded into a chromosome (DING *et al.*, 2013). There are two main approaches: a direct encoding which specifies every connection of the network or evolves an indirect specification of the connectivity (CANTÚ-PAZ; KAMATH, 2003; DING *et al.*, 2013).

Figure 8 – An ANN's architecture, followed by its connectivity matrix and the respective binary string representation.



Source: Yao (1999).

Usually, a direct encoding scheme codes each EANN's weight connections in a binary string (YAO, 1999; DING *et al.*, 2013). As shown in Figure 8, it uses a square connection matrix that explains whether there are connections amid the neurons by switching on (1) and off (0) those connections. The chromosome's binary string representing an architecture is the concatenation result of this matrix rows. It is very suitable for the precise and fine-tuned search of a compact architecture since a single connection can be added or removed easily (YAO, 1999). The fitness value is calculated by converting each

chromosome into an ANN, initialize its weights, training and validating it. Observe that the obtained validation error is not the same thing as test error, which will be calculated with the best solution provided at the end of the search algorithm. Constraints to the architecture, such as the existence or not of recurrent neurons, may also be incorporated by constraints in the connection matrix.

For example, a feedforward EANN will only consider the upper-right triangle of that matrix. This kind of solution is interesting for small networks with few hidden layers and neurons, but for larger EANN, the chromosome will also be very large, increasing the search space as consequence.

Some works encode not only the connections but also all weights and biases. Maniezzo (1994) adopted a binary encoding scheme with variable length, where the first byte specifies the number of bits that will code the weight values, followed by the connectivity bits and their respective synaptic weights represented as binary values. In a more modern approach, Yao and Liu (1997), Leung *et al.* (2003) and Tsai *et al.* (2006) used floating numbers instead of binary representation.

To avoid specifying all connections, the user may choose to encode only some characteristics of an architecture. This approach is named *indirect encoding scheme*. In this case, the user chooses to commit to a specific topology (feedforward, recurrent, number of layers) with a particular learning algorithm, and then use a metaheuristic to find the parameter values that complete the network specification (CANTÚ-PAZ; KAMATH, 2003). For examples, in the work of Alencar and Rocha Neto (2014), the AG-ELM assumes a fully connected feedforward network, evolving only the number of hidden neurons by switching them on and off in a binary chromosome. Another binary representation is found in the work of Cantú-Paz and Kamath (2003), where the authors use the genetic algorithm to an MLP network in order to evolve the number of hidden neurons, learning rate, the activation function's slope a and also the initial set of weights.

A more sophisticated approach to indirect representations was introduced by Kitano (1990), named *grammar encoding method* which is also known as *developmental rule representation*. Different from other encoding scheme methods, the network configuration is not put directly into the chromosome, instead, a set of rules that generate this information is encoded. The advantage of this method is that it may create a smaller chromosome and it will also suffer less from the destructive effect that crossover may produce (YAO, 1999).

In the work of Matias *et al.* (2013), although the chromosomes have the same length as the vector of decision variables, it also uses the principles of grammar representation by mapping different types of variables into continuous values. Their problem involves binary, integer and real variables which are all represented in the chromosome as floating point number in the interval of [0,1]. They are then converted into the actual variable value according to their original domains by the use of specific equations and finally applied in networks to obtain their respective fitness values.

4.4 The Adaptive Number of Hidden Neurons Approach

Motivated by the ELM's architecture design challenge combined with the possibility of also changing hidden neuron's firing distributions and providing a larger variety to their responses, we propose the *Adaptive Number of Hidden Neurons Approach* (ANHNA). It can be defined as a pruning method, where an assembly of chromosome representation and maintenance rules that optimizes, simultaneously, the number of hidden neurons and their activation function parameters.

We found in the work of Das *et al.* (2009), who used a modified version of Differential Evolution (DE) together with a specific chromosome encoding scheme for automatic clustering, a great potential for ELM's network optimization. Instead of cluster centroids, we choose to apply the two activation function parameters and also extend it to be used by any population-based and some trajectory based metaheuristics. The resulting chromosome is a vector of real numbers that apply indirect representation.

Figure 9 illustrates the structure of the ANHNA's i-th chromosome, which has a dimension equal to $3Q_{max}$, where Q_{max} is the maximum number of hidden neurons. The first Q_{max} elements are real number switches, called *activation thresholds*, that control whether the corresponding set of parameters ($a_{i,j}$, $b_{i,j}$) is activated or not and, consequently, if the j-th neuron exists or not. The following Q_{max} elements are the activation functions' slope values ($a_{i,j}$) and the remaining are their respective biases ($b_{i,j}$).

Figure 9 – ANHNA's i-th chromosome representation.

$$\mathbf{c}_i(g) = \underbrace{\boxed{T_{i,1}}\boxed{T_{i,2}}\boxed{\ldots}\boxed{T_{i,Q_{max}}}}_{\text{Activation Thresholds}}\underbrace{\boxed{a_{i,1}}\boxed{\ldots}\boxed{a_{i,Q_{max}}}}_{\text{Slopes}}\underbrace{\boxed{b_{i,1}}\boxed{\ldots}\boxed{b_{i,Q_{max}}}}_{\text{Biases}}$$

Source: author.

The activation thresholds $T_{i,j}$ behave like control genes, ruled by an unit step function defined by Rule 1 as follows:

Rule 1: Activation thresholds rule for the *i*-th chromosome.

---

if $T_{i,j} > 0.5$ **then**

j-th hidden neuron with its parameters $(a_{i,j}, b_{i,j})$ is active

else

j-th neuron does not exist

end if

return q_i = total number of active hidden neurons of *i*-th chromosome.

There are some precautions that must be taken. One of them is to keep all activation thresholds within an interval, so the results do not get very large values and get stuck by not being able to test other combination possibilities.

Rule 2: Keep activation thresholds within an interval for the *i*-th chromosome.

---

if $T_{i,j} > 1$ **then**

$T_{i,j} = 1$

else

if $T_{i,j} < 0$ **then**

$T_{i,j} = 0$

end if

end if

Another precaution is to ensure the existence of a minimum number Q_{min} of active hidden neurons in every chromosome, where $\mathbf{T}_{i,\mathbf{j}_{new}}$ is sampled from an Uniform distribution $\mathscr{U}(0.5, 1)$:

Rule 3: Number of active hidden neurons in the *i*-th chromosome.

---

if $q_i < Q_{min}$ **then**

choose randomly Q_{min} activation thresholds: $\mathbf{j}_{new} \in \mathbb{R}^{Q_{min}}$;

substitutes their activation values by $\mathbf{T}_{i,\mathbf{j}_{new}} \sim \mathscr{U}(0.5, 1)$;

end if

The fourth and last rule is that, after every chromosome update, the slope values a_{ij} will be replaced by their absolute values $|a_{ij}|$.

Another point that must be addressed is one of the challenges of keeping ELM principles in a metaheuristic architectural optimization approach: how to deal with random initialization of the weights. Different random weight initializations using the same genotype may produce quite different fitness values, which becomes a source of noise and mislead the evolution process (YAO; LIU, 1997). One manner to deal with this situation is to choose a single set of weights and based on them, perform the evolutionary search process over the best number of hidden neurons, as adopted by Alencar and Rocha Neto (2014), and other parameters. The issue with this approach is that it is equivalent to find combinations of hidden neurons and their fixed weights that provide an acceptable performance, instead of searching for different solutions. Another manner to reduce this source of noise is to train and validate the same solution several times and then use the average as the chromosome's fitness (YAO; LIU, 1997). Although this increases the execution time, it also provides a more precise measure of the fitness we can expect for a chromosome, without restraining the search options as the aforementioned solution does. Based on that, the ANHNA approach is tied to a 5-fold cross-validation to evaluate the performance of a chromosome's solution, where its fitness value receives the average of the validation RMSE.

It is important to highlight that the purpose of ANHNA is to return only the design skeleton, that is the number of hidden neurons and the respective parameters of their activation functions. It does not provide any weight value (hidden or output ones), which can be quickly determined by applying Algorithm 1. The training set used to train the final ELM is formed by the combination of training and validation set applied in the 5-fold cross-validation and then tested with a set that was beforehand separated and not used during the metaheuristic search.

From the exposed, we show how ANHNA can be applied to different metaheuristics. Algorithm 5 describes ANHNA used with Differential Evolution and Algorithm 6 describes ANHNA with Particle Swarm Optimization. The details of those methods are further discussed in Appendix A.

Algorithm 5: ANHNA-DE Pseudocode $\{Q_{min}, Q_{max}\}$

Initialize the DE parameters;
Initialize population $\mathbf{C}(0)$, where $\mathbf{T}(0)$ and $\mathbf{a}(0) \sim \mathscr{U}(0,1)$ and $\mathbf{b}(0) \sim \mathscr{U}(-1,1)$;
Check activation thresholds;
while $g < MAXGEN$ **do**
  for each chromosome, $\mathbf{c}_i(g)$ **do**
    5-fold cross-validation of an ELM; {with q_i active hidden neurons and their $(\mathbf{a}_i,\mathbf{b}_i)$}
    Evaluate $f(\mathbf{c}_i)$; {fitness objective is the mean validation RMSE.}
    Apply the mutation operator according to Eq. A.3
    Check activation thresholds and $|\mathbf{a}_i|$;
    Apply the Crossover (Eq. A.6) and Selection (Eq. A.7) operators $\rightarrow \mathbf{c}_i(g+1)$;
  end for
end while
return q_{best} and $(\mathbf{a}_{best},\mathbf{b}_{best})$.

Algorithm 6: ANHNA-PSO Pseudocode $\{Q_{min}, Q_{max}\}$

Initialize the PSO parameters;
Initialize population $\mathbf{C}(0)$, where $\mathbf{T}(0)$ and $\mathbf{a}(0) \sim \mathscr{U}(0,1)$ and $\mathbf{b}(0) \sim \mathscr{U}(-1,1)$;
Check activation thresholds;
while $g < MAXGEN$ **do**
  for each chromosome, $\mathbf{c}_i(g)$ **do**
    5-fold cross-validation of an ELM; {with q_i active hidden neurons and their $(\mathbf{a}_i,\mathbf{b}_i)$};
    Evaluate $f(\mathbf{c}_i)$; {fitness objective is the mean validation RMSE.}
  end for
  for each chromosome, $\mathbf{c}_i(g)$ **do**
    Set $\mathbf{p}_i$ and $\mathbf{pk}_i$; {personal and neighborhood best positions.}
    Update velocity and position according to Eqs. A.11 and A.14, respectively;
    Check activation thresholds and $|\mathbf{a}_i|$;
  end for
end while
return q_{best} and $(\mathbf{a}_{best},\mathbf{b}_{best})$.

4.4.1 Variants of ANHNA

So far, our proposal focuses on trying to minimize the mean validation RMSE. As discussed on Chapter 2, an acceptable test error performance is not always the best measure to evaluate the future generalization performance. The size of the ELM's output weights influences on the generalization and also on the sensibility to outliers. Its size, that can be represented by the norm of the output weights, is directly related to how well-conditioned the hidden layer output matrix **H** is. Bearing this in mind, for comparison purposes, we developed seven different fitness functions in order to study the effect of multiple objectives in one single fitness function.

4.4.1.1 Using the condition number of $\mathbf{H}$ *and the norm of the output weight vector*

The version of ANHNA that utilizes the norm of the output weight vector besides the RMSE, named $ANHNA_{NO}$ for differentiation purposes only, adopts as fitness function the following equation:

$$f(\mathbf{c}_i) = \mu_{RMSE} + \delta_{RMSE} + 0.01\mu_{WNorm}, \tag{4.1}$$

where μ_{RMSE} represents the mean validation RMSE, δ_{RMSE} is its standard deviation and μ_{WNorm} is the mean euclidean norm of the respective output weight vector, all from the i-th chromosome.

With this setup, only 1% of the mean output weight norm is considered because its value may be overpowering the RMSE value. The standard deviation is also added to penalize those solutions that provide a large variation on the validation RMSE. From that, we expect to converge to a solution with smaller validation RMSE, standard deviation and output weight norm, which is also expected to diminish substantially the number of hidden neurons, since the fewer neurons the smaller the norm.

The second version aims at investigating how the condition number of the hidden layer output matrix may contribute to find an optimal ELM architecture. The condition number of a matrix (κ) is defined as the ratio of the largest eigenvalue (σ_{max}) to the smallest eigenvalue (σ_{min}) of the matrix (HAYKIN, 2008), as follows:

$$\kappa(\mathbf{H}) = \frac{\sigma_{max}(\mathbf{H})}{\sigma_{min}(\mathbf{H})}. \tag{4.2}$$

The closer to 1 this number is, the more well-conditioned this matrix is. It means that we can expect a solution for the linear system which is numerically more stable and less sensitive to noise in the data.

In our implementation, we applied a Matlab function named *rcond* that returns an estimation for the reciprocal of the condition number of a matrix. If it returns a value near to 1, the matrix is well-conditioned. If it returns a value near 0, this matrix is badly-conditioned. We choose this function to keep all involved variables in a small range.

With this configuration, we named $ANHNA_{CN}$ and its respective new fitness

function is given by

$$f(\mathbf{c}_i) = \mu_{RMSE} + \delta_{RMSE} - \iota\mu_{\kappa}, \tag{4.3}$$

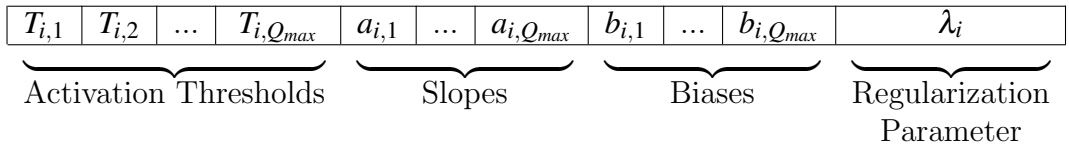
where, $\iota = 0.01$ and, just like ANHNA$_{NO}$, the μ_{RMSE} and δ_{RMSE} are the mean validation RMSE and the standard deviation, receptively, but μ_{κ} is the mean condition number of the generated output layer matrices of the i-th chromosome. Since the larger the last term gets, the better, which is the opposite from the first two, it will have a different sign within the function.

To study the impact of a stronger influence of the condition number in the evolution process, we determined a third variant, named $ANHNA_{CNv2}$. It is essentially the same as ANHNA$_{CN}$, although $ANHNA_{CNv2}$ contributes with a higher value, which is $\iota = 1$.

4.4.1.2 Using regularization

Another variant of ANHNA aims at investigating the effect of adding a regularization term into the fitness function. As discussed in Chapter 2, the idea of adding regularization is to restore well-posedness by appropriate constraints on the solution and provide, as consequence, smaller output weights. Nevertheless, as shown in Equation 2.13, there is a new parameter that must be determined: the regularization parameter λ. To solve this issue, we added it to the solution vector, as shown in Figure 10, that will be evolved along with the architectural solution.

Figure 10 – ANHNA's i-th chromosome representation with regularization.



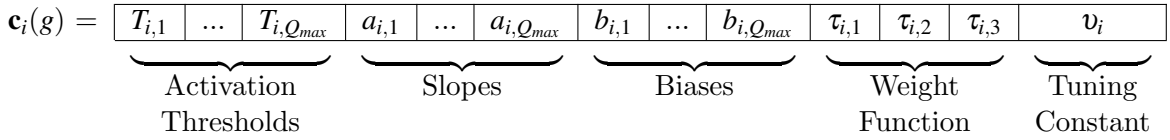
Source: author.

This new setup is named $ANHNA_{Re}$ and it has also the same variants as the first ANHNA version. $ANHNA_{Re}$'s fitness function utilizes only the mean validation RMSE, $ANHNA_{ReNO}$ follows the Equation 4.1, $ANHNA_{ReCN}$ uses Equation 4.3 with $\iota = 0.01$ and $ANHNA_{ReCNv2}$ the Equation 4.3 with $\iota = 1$.

4.4.2 ANHNA for robust regression

One may argue if it is possible to extend ANHNA to evolve a robust ELM network. The answer is yes, although it is still a work in progress. Our investigation, based on what was discussed in Chapter 3, compelled us to add two other parameters to the evolved solution: the weight function and the tuning constant, also known as error (or outlier) threshold. The choice of weight function is coded as three real-valued variables that, just like the activation threshold, are governed by a sign function. This results in three bits that encode eight different weight functions, which are demonstrated in Table 5. The tuning parameter is directly encoded as a real value variable at the end of ANHNA's chromosome, as shown in Figure 11.

Figure 11 – ANHNA's i-th chromosome representation for robust ELM networks.



Source: author.

Table 5 – Codification for choosing the weight function.

Code	Weight Function
000	Talwar
001	Huber
010	Bisquare
011	Fair
100	Cauchy
101	Andrews
110	Welsch
111	Logistic

Source: author.

Besides the chromosome modification, new rules are also implemented. Two of them are similar to the ones that govern the activation thresholds. The first, as already mentioned, is the use of sign function to generate the bits that will be compared to the translation Table 5. Note, however, that τ values are not replaced with the newly defined bits. A different variable vector, defined by the user, stores this code to each chromosome.

The second rule keeps them inside the [0,1] interval.

Rule 5: Weight function code determination for the *i*-th chromosome.

---

for j=1 until 3 **do**
  if $\tau_{i,j} > 0.5$ **then**
    j-th bit is considered equal to 1
  else
    j-th bit is considered equal to 0
  end if
end for
return q_i = total number of active hidden neurons of *i*-th chromosome.

Rule 6: Keep weight function code into an interval for the *i*-th chromosome.

---

for j=1 until 3 **do**
  if $\tau_{i,j} > 1$ **then**
    $\tau_{i,j} = 1$
  else
    if $\tau_{i,j} < 0$ **then**
      $\tau_{i,j} = 0$
    end if
  end if
end for

A third new rule is that, just like the slope parameters, υ must also be always a positive value. Hence, it will be replaced with its absolute value.

Another modification is within the 5-fold cross-validation process. The outliers contamination of the desired output in training and validation data required a different approach regarding the way the validation error will be faced. With the validation set also contaminated, which is nearly what we have in real life problems, it is not possible to simply use all results from the cross-validation to define the expected error. On one hand, if one of the validation results returns a low error, the ELM model has most likely learned also the outliers, which is not interesting. On the other hand, a large validation error could

reflect that the network did not handle the outliers correctly and, hence, has not learned the correct model at all. To handle these situations, we adopted more robust measures: trimmed mean or median of the validation errors. While the median is represented by the middle value taken after arranging the validation errors from lowest value to highest value, the trimmed mean takes the mean of the remaining validation errors after excluding a percentage of the largest and lowest values. In our specific case, the percentage for the trimmed mean is 10%, from where we exclude 10% of the largest absolute validation errors for each fold in cross-validation. The final fitness value for each chromosome is the mean of the resulting five trimmed RMSE.

Both cases are being investigated and the details of how the contamination was performed are in Chapter 5.

This modification of ANHNA is named $ANHNA_R$ and has all rules compiled in Algorithm 7. We elected Differential Evolution as the applied metaheuristic. It is important to highlight that, just as the original ANHNA, ANHNA$_R$ returns only the architectural information. The difference lies in the addition of the weight function to be applied to IRLS method and its error threshold (υ_{best}). The final robust ELM must be trained (with training and validation set) and tested (with unseen data), using IRLS as output weights estimation method.

Algorithm 7: ANHNA$_R$-DE Pseudocode $\{Q_{min}, Q_{max}\}$

1: Initialize the DE parameters;
2: Initialize population $\mathbf{C}(0)$, with $\mathbf{T}(0)$, $\mathbf{a}(0)$, $\tau(0)$ and $\upsilon(0) \sim \mathscr{U}(0,1)$ and $\mathbf{b}(0) \sim \mathscr{U}(-1,1)$;
3: Check activation thresholds;
4: **while** $g < MAXGEN$ **do**
5: **for** each chromosome, $\mathbf{c}_i(g)$ **do**
6: 5-fold cross-validation of an ELM; {with q_i active hidden neurons and their $(\mathbf{a}_i,\mathbf{b}_i)$, using the weight function given by τ_i and tuning constant υ_i.}
7: Evaluate $f(\mathbf{c}_i)$; {fitness objective: trimmed mean or median of validation RMSE.}
8: Apply the mutation operator according to Eq. A.3
9: Check activation thresholds, weight function code, $|\mathbf{a}_i|$ and $|\upsilon_i|$;
10: Apply the Crossover (Eq. A.6) and Selection (Eq. A.7) operators $\rightarrow \mathbf{c}_i(g+1)$;
11: **end for**
12: **end while**
13: **return** q_{best}, $(\mathbf{a}_{best},\mathbf{b}_{best})$, the respective weight function and υ_{best}.

4.5 Concluding remarks

Model selection for ELM networks using evolutionary algorithms has been a subject of interest for many years. The search and optimization properties of those methods make them suitable for this task. However, as discussed in this chapter, the majority of works involves the optimization of hidden weights, which goes against the fundamental principle of this network: the random choice of the input-to-hidden layer weights.

In this chapter, we proposed the *Adaptive Number of Hidden Neurons Approach* (ANHNA), a general encoding scheme, which can be adopted by different metaheuristics and evolves not only the number of hidden neurons but also the activation function parameters: slope and bias. It also keeps the ELM's principle of the random hidden weights.

ANHNA, besides the manner of encoding the architectural information in the chromosome, is also tied to a set of rules that governs the evolution process. Several other variants of ANHNA were proposed with different objective functions. A multiobjective task was implemented into single functions that will minimize the validation error as well as the output weight norm.

Another variant of ANHNA, named ANHNA$_{\boldsymbol{R}}$, is oriented towards the evolution of robust ELM networks. It promotes chromosome modification and a new set of rules, which include a change in how the validation error is treated in the fitness function.

In the following chapter, we present the experimental methodology as well as how the outlier contamination was performed. The parameters adopted for each method applied are also detailed and the chosen datasets are described.

5 METHODS AND EXPERIMENTS

The success of any experiment relies on how it is designed. The objective of this chapter is to present how the datasets were treated and contaminated for the outlier robustness cases (Section 5.1) and provide information about how the tests were performed (Section 5.2) and which parameters were adopted (Section 5.3). Finally, we make our final remarks over the content of this chapter in Section 5.4.

5.1 Data treatment

In this thesis, we investigate the performance of the proposed methods with 6 different datasets regarding regression problems, which are further described in Appendix B. We follow the work from Horata *et al.* (2013) as a major reference, which also includes their methodology for experiment design. In this regard, attributes from all sets were scaled to $[0,1]$ and their target values to $[-1,1]$.

5.1.1 Outlier contamination

The experiments on the outlier robustness involved deliberately contaminated datasets. Notwithstanding real-world datasets will mostly like have outliers, specific levels of noise will be added to the target outputs of the training set. These levels allow us to control and observe the behavior of the chosen algorithms with increasing distortion and if their performances will degrade or show robustness to it. It is also important to highlight that, in this thesis, we investigate only the contamination of the output, which interferes in the modeling precision, as discussed in Chapter 3.

From each dataset, eight sub-problems will emerge, depending on the type of outlier and the contamination rate. They could be contaminated by either one-sided outliers or two-sided outliers, where one-sided refers to absolute values only and two-sided may be positive as well as negative values. The contamination rates are 10%, 20%, 30% and 40%.

Following Horata *et al.* (2013), let $K \in 1,...,N$ be a subset of row indices of desired output matrix $\mathbf{D}$ that will be contaminated with outliers and $\Delta_k \in \mathbb{R}^{1\times r}$ is a vector of random errors drawn from a multivariate normal distribution of zero mean vector and diagonal covariance matrix, i.e. $\Delta_k \sim N(\mathbf{0}, \sigma^2\mathbf{I}_r)$, where $\mathbf{I}_r$ is a $r \times r$ identity matrix and

$\forall k \in K$. Thus, $\mathbf{d}_k \in \mathbb{R}^{1\times r}$ is a row from $\mathbf{D}$ and $\tilde{\mathbf{d}}_k$ will be its correspondent contaminated row due to one-sided outliers if

$$\tilde{\mathbf{d}}_k = \mathbf{d}_k + |\Delta_k|, \quad \forall k \in K, \tag{5.1}$$

or two-sided outliers if

$$\tilde{\mathbf{d}}_k = \mathbf{d}_k + \Delta_k, \quad \forall k \in K. \tag{5.2}$$

The contaminated target output matrix $\tilde{\mathbf{D}}$ is defined as

$$\tilde{\mathbf{D}} = \mathbf{D} + \Delta, \tag{5.3}$$

where $\Delta \in \mathbb{R}^{N\times r}$ is comprised of $[\Delta_1 ... \Delta_i ... \Delta_N]$ such that if $i \neq K$ then $\Delta_i = \mathbf{0}$.

It is also important to highlight that the test data do not suffer any contamination. This way, we can evaluate if the network has learned the true function underlying the data or if it has learned the noise as well.

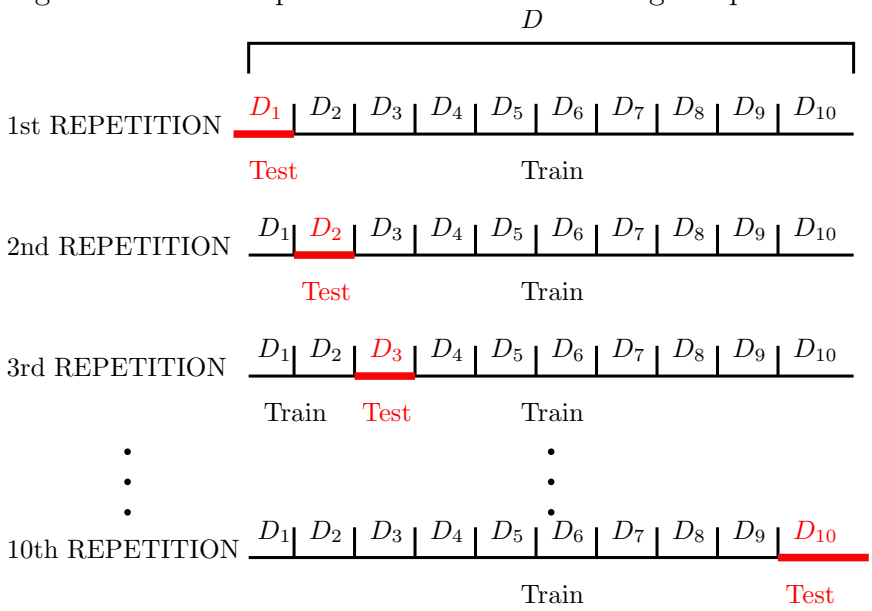
5.2 Experiments

In this thesis, there are three main experiments: the first is related to ELM robustness and a comparison to the proposed R-ELM/BIP; the verification of the performance of ANHNA's variants with different metaheuristics; and, finally, the evaluation of the work in progress over ANHNA$_R$, that will be presented with the first experiment.

All experiments involved basically two steps. First the data is randomized and then separated into ten almost evenly set of samples. We will use the idea of cross-validation here to guarantee that all samples were used to test the algorithms at least one time, which results in ten independent repetitions. Figure 12 shows the data partitioning into the test set and training set, where D represents here the whole dataset for demonstration purposes only.

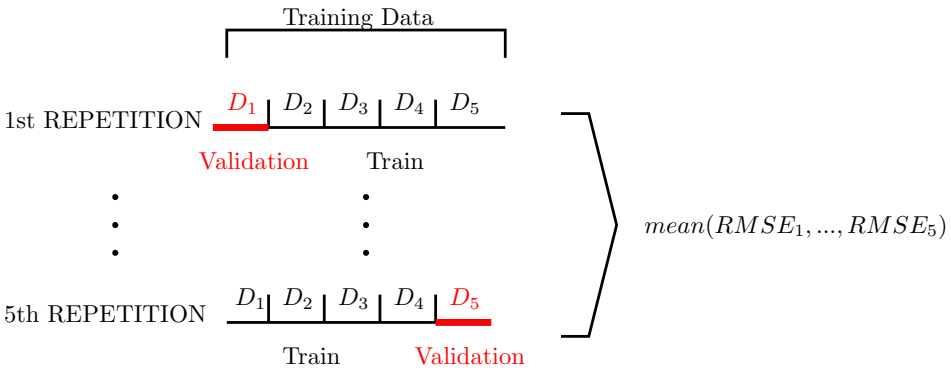
At each repetition, the second step depends on the method to be assessed. In the first experiment, for example, there is the issue of the estimation of parameters such as the number of hidden neurons. So, within every repetition, we applied a grid search, where the number of hidden neurons was increased from 2 to 100 neurons, in steps of 2. For each number of hidden neurons, it was performed a 5-fold cross-validation, as shown in Figure 13.

Figure 12 – Data separation in test and training samples.



Source: author.

Figure 13 – A 5-fold cross validation procedure.

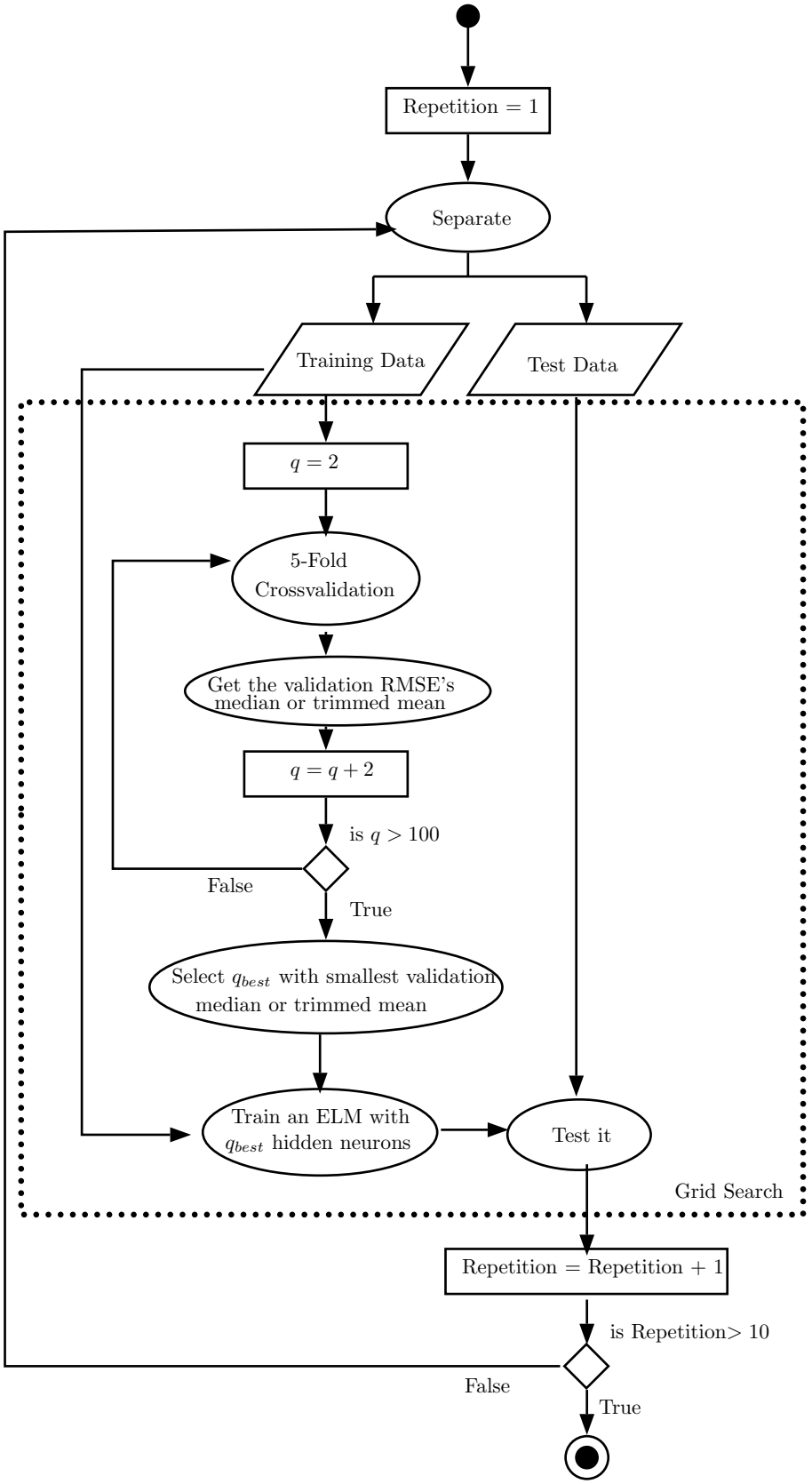


Source: author.

After the full grid was covered, the number of neurons that returned the smallest validation error is chosen. Later, the network using the chosen number of hidden neurons is trained applying training plus validation dataset, and then tested with unseen data. Figure 14 illustrates the steps of this procedure. We should highlight, however, that the calculated RMSE is a trimmed version, where 10% of the largest absolute error is discarded. This approach is used in an attempt of diminishing the outliers influences in the choice of the best architecture. With this setup, the following algorithms were compared: ELM, ELM/BIP, ROB-ELM, IRWLS-ELM and R-ELM/BIP.

In the second experiment, the parameter estimation is performed with ANHNA and its variants. In this specific case, the 5-fold cross-validation is inside the method itself, as described in Chapter 4. It is important to highlight that ANHNA and its

Figure 14 – Flow chart of the experiments with robust ELM networks.

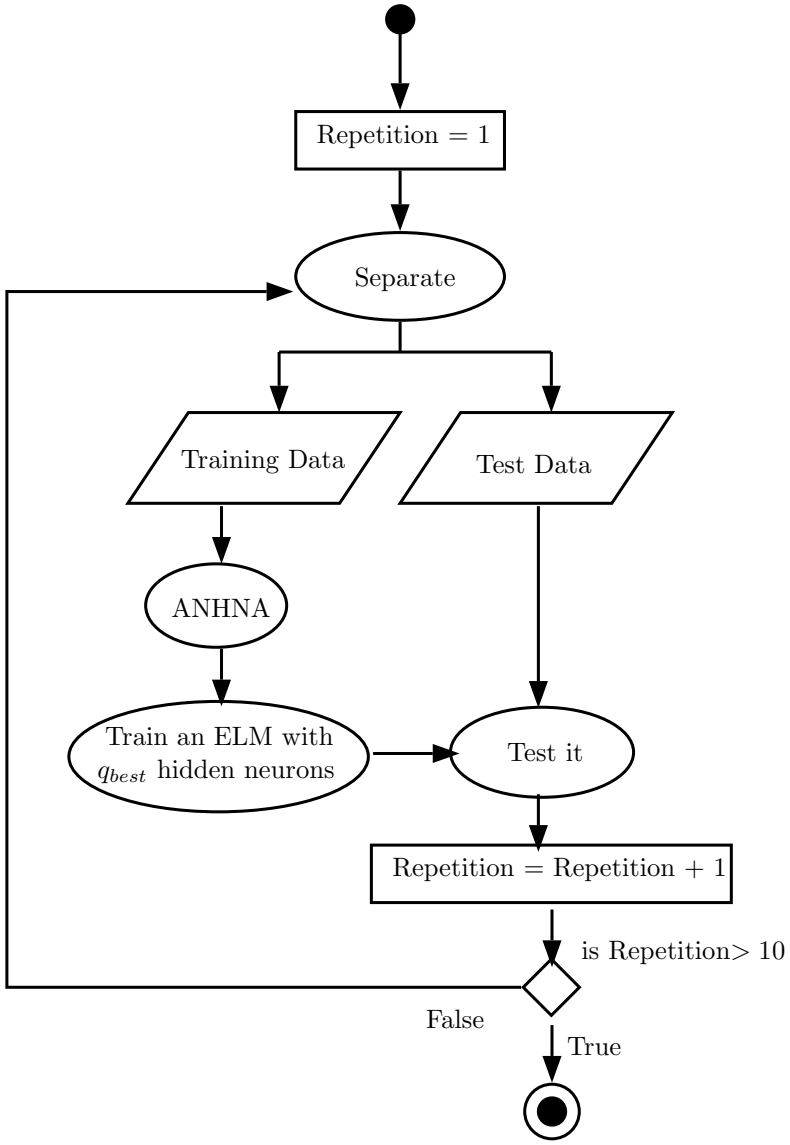


Source: author.

variants (ANHNA$_{NO}$, ANHNA$_{CN}$,ANHNA$_{CNv2}$, ANHNA$_{Re}$, ANHNA$_{ReNO}$, ANHNA$_{ReCN}$ and ANHNA$_{ReCNv2}$) utilize the mean validation RMSE, while ANHNA$_{R}$ may use the trimmed mean or median of validation RMSE after the 5-fold cross-validation. Figure 15 illustrates the executed steps. Those variants were compared only to ANHNA itself, but to evaluate ANHNA's performance, it was compared to ELM, ELM/BIP, ANHNA-DE, ANHNA-SADE, ANHNA-PSO$_g$ and ANHNA-PSO$_l$.

The third and last experiment, that refers the to ANHNA$_{R}$, follows the same structure as ANHNA in Figure 15.

Figure 15 – Flow chart of the experiments with ANHNA.



Source: author.

Table 6 indicates how the data was divided on the first repetition in training and test samples to test and training. Training samples are shown in five columns that

represent the five training and validation folds. Other repetitions are not presented due to the fact that the amount of samples varies very little: plus or minus one sample.

Table 6 – Division of samples for each fold in cross validation.

Datasets	**Test**	**Training/Validation**				
Auto-MPG	40	282/70	281/71	282/70	282/70	281/71
Body fat	25	182/45	181/46	181/46	182/45	182/45
Breast Cancer	20	140/34	139/35	139/35	139/35	139/35
CPU	21	150/38	150/38	151/37	151/37	150/38
iCub	49	353/89	354/88	353/89	354/88	354/88
Servo	17	120/30	120/30	120/30	120/30	120/30

Source: author.

In robustness experiments, the data partitioning is still the same, but now an error is added to the training and validation sets. As described in Section 5.1, there are eight combinations: 1 sided and 2 sided outliers with 10%,20%,30% and 40% of contamination each. For each combination, training and validation data will separately receive the percentage of corrupted samples. This means that, e.g., if there are 100 samples for training, 20 samples for validating and 10% of outliers, the training set will have 10 corrupted samples and validation set will have 2. It is important to highlight, once again, that the test set does not suffer any contamination.

ANHNA-DE$_R$ is compared with the following algorithms: ELM, ELM/BIP, ROB-ELM, IRWLS-ELM and R-ELM/BIP.

The experiments were executed simultaneously on six different computers with different versions of Matlab and variable computational resources. Because of this setup, the execution time will be given as a generic estimation based on empirical observation.

5.3 Parameters

Once the framework of the experiments is determined, we present the parameters values applied to the different algorithms during those tests.

For ELM networks, which had their number of hidden neurons found by grid search and did not suffer any modification on their activation function's parameters, there is a bias input equal to one only to the hidden layer. The other ELM generated in this work had their activation function's parameters altered, which already included a bias to them, that is why there is no need for an external fixed input as the example above.

All ELM use the hyperbolic tangent as activation function and the non-robust networks adopt SVD for estimation of output weights. Hidden weights are randomly initialized between $[-0.1, 0.1]$.

For the BIP procedure, the estimation of the activation function's parameters, as shown in Equation 2.14, needs a regularization parameter $\lambda = 10^{-2}$ and the desired exponential distribution mean $\mu_{f_{des}} = 0.2$. Those values are the same ones adopted in the original work (NEUMANN; STEIL, 2011) and were no longer optimized.

For the DE algorithm, the scaling and crossover parameters are calculated just as in the work of Das *et al.* (2009). The scaling factor varies in the range [0.5,1] by using the following equation:

$$F = 0.5(1 + rand(0,1)), \tag{5.4}$$

where $rand(0,1)$ is an uniformly distributed random number within the range [0,1].

$$CR = C_{min} + \left((Cmax - Cmin) \left(\frac{TotalGenerations - ActualGeneration}{TotalGenerations} \right) \right). \tag{5.5}$$

In association with this variable parameter, the mutation step follows the $DE/best/1$ strategy (see Appendix A). For the crossover, its parameter CR is also variable. It decreases linearly within the interval [0.5,1] along the generations as demonstrated in Equation 5.5. When $CR = 1$, it means that all components of the parent chromosome are replaced with the difference vector operator. As this parameter is decreasing, more components of the parent are inherited by the offspring. This approach allows the algorithm to perform an exhaustive exploration of the search space at the beginning and finely adjust the solutions at the final stages of evolution (DAS *et al.*, 2009).

Just as the adopted DE, SADE algorithm also does not need predefined scaling and crossover parameters. For both, we just need to set the number of population $NP = 100$ and the maximum number of generations $MAXIGEN = 300$, and for SADE there is also the learning period $LP = 20$.

For the PSO algorithm, we also adopted $NP = 100$ and $MAXGEN = 300$ for both *gbest* and *lbest* strategies. The acceleration constants $\varsigma_1 = \varsigma_2 = 2.05$ and the constriction coefficient $\chi = 0.72984$ were defined with values suggested by the works of Clerc and Kennedy (2002) and Bratton and Kennedy (2007).

In the robustness experiments, RELM and R-ELM/BIP do not have any adaptation of the weighting function. Thus, the Matlab's default bisquare weighting function is used as well as its respective default tuning constant equals to 4.685 for R-ELM/BIP. The RELM, as proposed in (BARROS; BARRETO, 2013), uses Andrews function with its default tuning constant equals to 1.339.

5.4 Concluding remarks

In this chapter, we detailed the three main experiments carried out in this thesis based on regression problems, how the data was treated and the adopted parameters for all applied algorithms.

From a data perspective, this work could be divided into two parts: non-robust and robust ELM. The non-robust part refers to the tests realized with ANHNA, ANHNA$_{NO}$, ANHNA$_{CN}$,ANHNA$_{CNv2}$, ANHNA$_{Re}$, ANHNA$_{ReNO}$, ANHNA$_{ReCN}$ and ANHNA$_{ReCNv2}$. The purpose of this experiment is to assess their capacity of providing good solutions preferentially with a, as small as possible, number of hidden neurons and norm of the output weight vector.

As for the robust part, it could also be divided into two parts: the evaluation of our method R-ELM/BIP and the work in progress about ANHNA$_{R}$. Section 5.1 presented the procedure for data contamination that we could observe, in a controlled manner, how the algorithms behave with increasing presence of outliers in the data.

In terms of data presentation, all experiments are based on two steps: ten external repetitions and a 5-fold cross-validation. This last procedure is applied either inside a grid search or within ANHNA and its variants. An overview of the process is exemplified in Figures 14 and 15 respectively.

In the next chapters, we present the results of the aforementioned experiments.

6 PERFORMANCE ANALYSIS OF THE ROBUST ELM

In this chapter, we present the results concerning the ELM network and its improved robust versions applied to data contaminated with outliers. We compare non-robust versions, such as ELM and ELM with Batch Intrinsic Plasticity (ELM/BIP), with versions that adopt a robust estimation method of output weights, e.g., Robust ELM (RELM), IRWLS-ELM and the proposed R-ELM/BIP (see Chapter 3 for more details). We also present the preliminary results from the robust versions ANHNA-DE$_R$ and ANHNA-DE$_{Rm}$, where the former uses a trimmed RMSE and the latter a median RMSE as the validation error. In appendix F it is illustrated the evolution along the generations of the choice of the weight functions and its respective error threshold for each robust ANHNA.

To evaluate their performances, we selected five real-world regression problems datasets: Auto-MPG, Breast Cancer, CPU, iCub, and Servo, although robust ANHNA versions do not have results for iCub dataset so far. For more details about them, see Appendix B. The evaluation metrics include the test RMSE, the selected number of hidden neurons and the Euclidean norm of the output weight vector.

As previously mentioned, the training and validation sets are contaminated with 10%, 20%, 30% e 40% of outliers that are also divided into two categories: one sided and two-sided. However, it is important to highlight that the testing set has no added noise but the one embedded in the dataset itself. That way, we can evaluate if the algorithm has learned the model or it has been profoundly affected (i.e. distorted) by the outliers.

This chapter is organized as follows. Section 6.1 displays all graphics and tables related to each dataset and its respective type of contamination followed by the discussion based on observations drawn from those graphics and tables.

6.1 Comparison of robust ELM networks

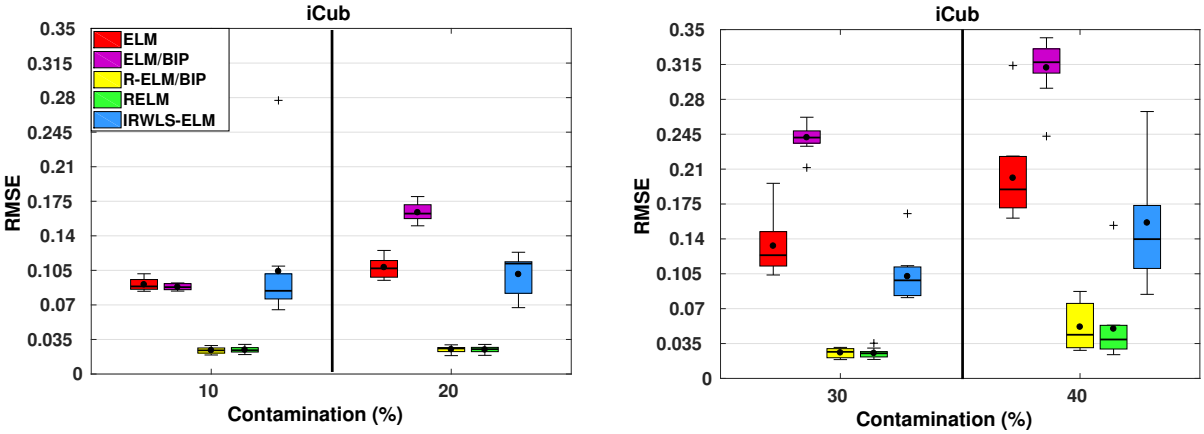
We present a set of boxplots that describes the range of the solutions found after each of the 10 repetitions of the aforementioned algorithms. The black dots represent the mean values of the evaluated metric and the black crosses indicate results that were far different from the distribution of the repetitions' results. Remember that each repetition is related to a cross-validation process to find the most suitable number of hidden layer neurons, which means that those repetitions may provide a different number of neurons

for the same method and problem. This is also described in boxplots. At the end, the behavior of output weight norms within the different outlier-contaminated scenario is demonstrated in tables. For a more precise description of the number of hidden neurons chosen see the tables displayed in Appendix C.

6.1.1 iCub dataset

Figure 16 – Testing RMSE of robust ELM networks with 1 sided contamination (iCub dataset).

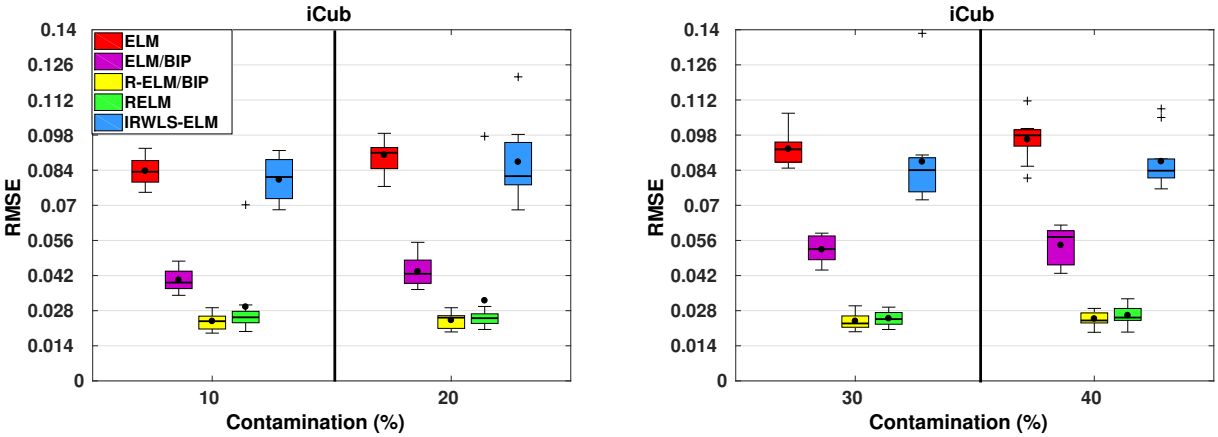
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 17 – Testing RMSE of robust ELM networks with 2 sided contamination (iCub dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

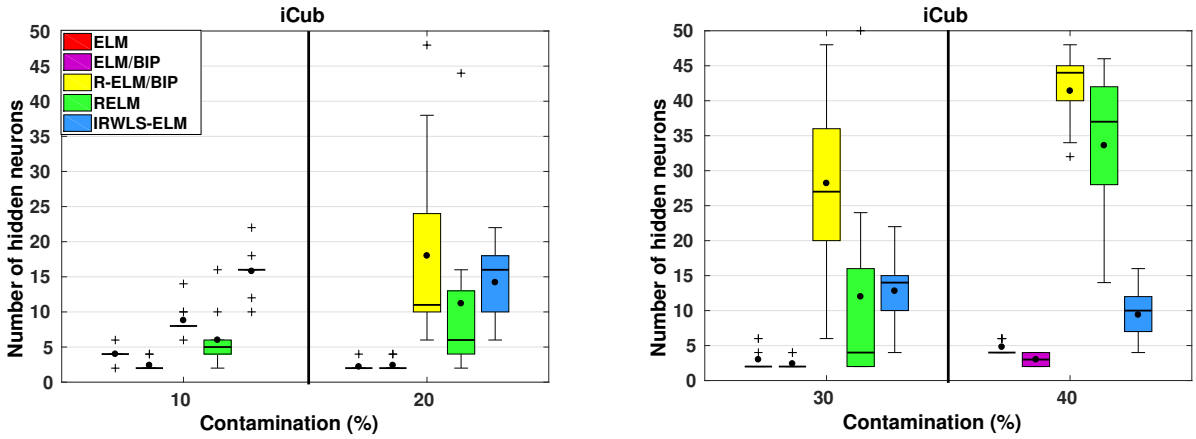
From Figures 16 and 17, it is clear the deterioration of the RMSE performance of the non-robust methods when we increase the percentage of contamination with outliers. This behavior confirms our expectation that non-robust methods would be highly affected

by the presence of outliers, resulting in poor estimations of the real phenomenon that generated the data. The larger the amount of noise, the worst are their performances. This result also demonstrates that outliers may influence ELM performance when this issue is not correctly addressed.

The proposed R-ELM/BIP and the RELM showed to be less affected in those cases, where RELM demonstrated a slightly better RMSE performance in the two-sided case.

Figure 18 – Number of hidden neurons of robust ELM networks with 1 sided contamination (iCub dataset).

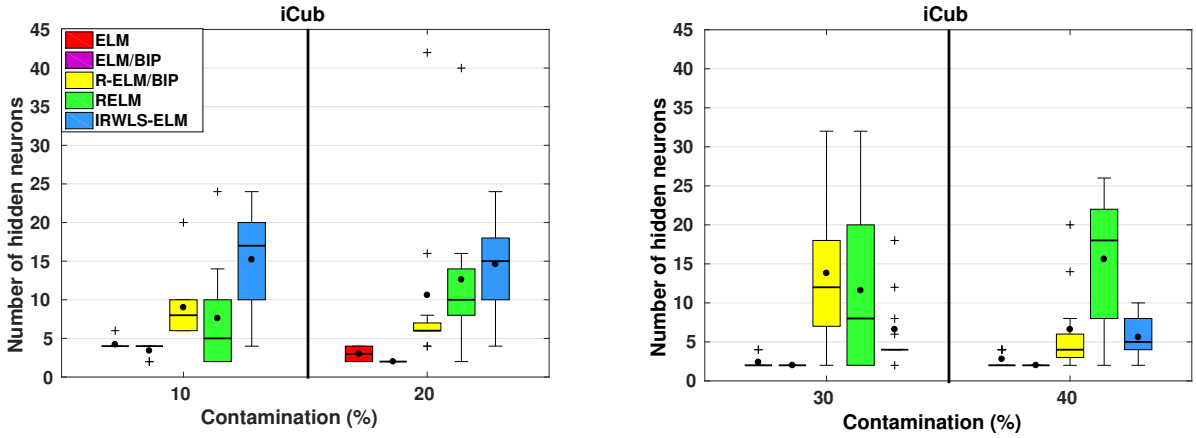
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 19 – Number of hidden neurons of robust ELM networks with 2 sided contamination (iCub dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Interestingly, the ELM/BIP network offered some outlier robustness when compared to the ELM in the two-sided contamination scenario. This is due to the fact that

absolute-valued noise is a much harder problem than the two-sided one. Considering that an outlier exerts an influence that draws the estimated model towards it when all outliers are positive values, they will all exert an attraction towards the same direction. Moreover, when we deal with outliers both positive and negative, their influence may be deadened by each other to some extent. We may observe, as well, that the ability to optimize the outputs of the hidden neurons and consequently providing small norms of the norm of the output weight vector, has a direct influence, not only on the generalization capacity but also in the ELM's outlier robustness property. Hence the importance of addressing correctly ELM numerical issues.

Observing the number of hidden neurons in Figures 18 and 19, it is interesting to notice that, after all repetitions of grid search and cross validation, the ELM and ELM/BIP best networks have less than five hidden neurons. Usually, a large number is closely related to a better RMSE performance, although this is not the case of IRWLS-ELM. Even though this method deals with computational and outlier robustness problems, its RMSE performance is only slightly better than the traditional ELM. We hypothesized that its high norms of output weight vectors (see Tables 7 and 8) make it too sensitive to any possible noise contained in the dataset.

As for the norms themselves in Tables 7 and 8 , ELM/BIP and R-ELM/BIP provided not only the smallest ones, but also presented the smallest variation considering the increasing contamination. Those tables also illustrate how much a randomly initialized hidden layer may affect the norms of the output weight vectors and how important it is to use methods that improve it.

Table 7 – Output weight euclidean norm with 1 sided contamination (iCub dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	8.130×10^{2}	4.334×10^{2}	9.844	2.100×10^{3}	7.524×10^{2}
	ELM-BIP	4.594×10^{-1}	4.094×10^{-1}	3.738×10^{-1}	8.527×10^{-1}	1.473×10^{-1}
	R-ELM/BIP	2.416	1.654	6.702×10^{-1}	9.407	2.543
	RELM	7.813×10^{3}	55.03	9.840×10^{-1}	6.697×10^{4}	2.104×10^{4}
	IRWLS-ELM	1.602×10^{6}	1.240×10^{6}	2.676×10^{4}	5.329×10^{6}	1.657×10^{6}
20%	ELM	1.906×10^{2}	7.690	2.543	1.698×10^{3}	5.314×10^{2}
	ELM-BIP	1.745	6.487×10^{-1}	4.741×10^{-1}	5.930	1.851
	R-ELM/BIP	86.34	4.152	6.947×10^{-1}	5.415×10^{2}	1.771×10^{2}
	RELM	4.231×10^{7}	90.47	4.941	4.224×10^{8}	1.335×10^{8}
	IRWLS-ELM	2.637×10^{6}	2.122×10^{6}	9.197×10^{3}	7.775×10^{6}	2.857×10^{6}
30%	ELM	1.625×10^{3}	43.92	2.555	9.270×10^{3}	3.086×10^{3}
	ELM-BIP	1.700	7.957×10^{-1}	7.192×10^{-1}	7.789	2.192
	R-ELM/BIP	1.676×10^{2}	1.021×10^{2}	1.627	5.712×10^{2}	1.952×10^{2}
	RELM	1.475×10^{8}	1.971×10^{2}	8.674×10^{-1}	1.474×10^{9}	4.661×10^{8}
	IRWLS-ELM	3.801×10^{6}	4.225×10^{5}	1.799×10^{2}	1.804×10^{7}	7.184×10^{6}
40%	ELM	2.555×10^{3}	2.080×10^{3}	5.761×10^{2}	6.379×10^{3}	2.098×10^{3}
	ELM-BIP	3.599	1.913	8.639×10^{-1}	12.04	4.231
	R-ELM/BIP	1.260×10^{3}	8.482×10^{2}	2.472×10^{2}	4.041×10^{3}	1.224×10^{3}
	RELM	3.821×10^{8}	3.491×10^{8}	7.354×10^{4}	9.577×10^{8}	3.710×10^{8}
	IRWLS-ELM	7.146×10^{5}	8.219×10^{4}	7.855×10^{2}	3.905×10^{6}	1.317×10^{6}

Source: author.

Table 8 – Output weight euclidean norm with 2 sided contamination (iCub dataset).

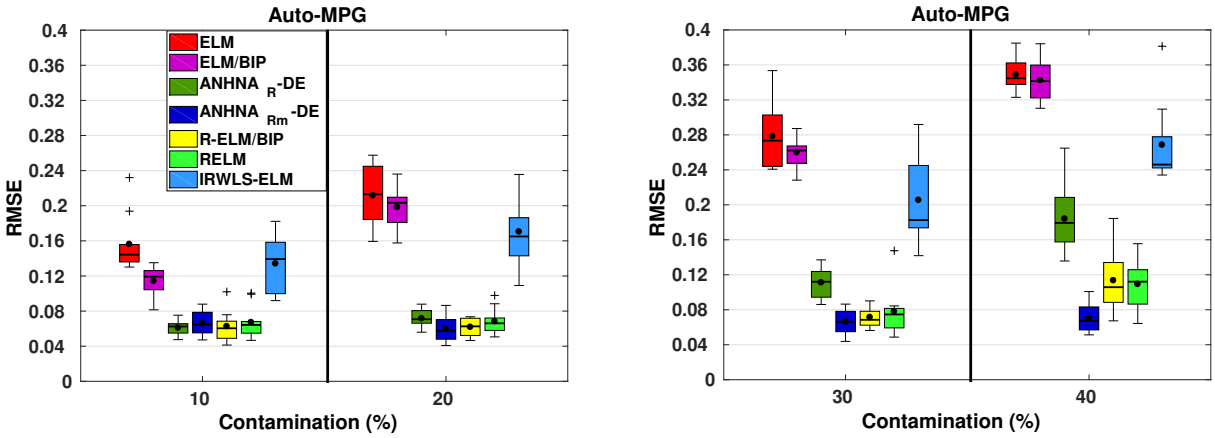
		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	6.117×10^{2}	4.678×10^{2}	1.279×10^{2}	1.688×10^{3}	5.063×10^{2}
	ELM-BIP	1.479	1.003	2.715×10^{-1}	5.839	1.618
	R-ELM/BIP	2.314	1.419	7.378×10^{-1}	8.907	2.562
	RELM	5.551×10^{5}	1.220×10^{2}	2.248	5.409×10^{6}	1.706×10^{6}
	IRWLS-ELM	4.126×10^{6}	1.127×10^{6}	2.261×10^{2}	1.659×10^{7}	6.317×10^{6}
20%	ELM	2.368×10^{2}	1.567×10^{2}	1.508	7.338×10^{2}	2.797×10^{2}
	ELM-BIP	6.620×10^{-1}	3.495×10^{-1}	2.602×10^{-1}	1.938	6.147×10^{-1}
	R-ELM/BIP	43.61	1.717	4.223×10^{-1}	4.151×10^{2}	1.306×10^{2}
	RELM	1.246×10^{7}	5.681×10^{3}	8.547×10^{-1}	1.242×10^{8}	3.926×10^{7}
	IRWLS-ELM	3.838×10^{6}	1.016×10^{6}	5.454×10^{2}	1.621×10^{7}	6.336×10^{6}
30%	ELM	1.209×10^{2}	3.009	1.236	9.658×10^{2}	3.043×10^{2}
	ELM-BIP	1.275	3.502×10^{-1}	2.837×10^{-1}	8.513	2.557
	R-ELM/BIP	35.12	3.128	2.550×10^{-1}	2.776×10^{2}	86.07
	RELM	5.664×10^{6}	2.528×10^{4}	1.858	5.322×10^{7}	1.673×10^{7}
	IRWLS-ELM	1.363×10^{7}	5.734×10^{2}	1.820	1.361×10^{8}	4.304×10^{7}
40%	ELM	6.250×10^{2}	11.63	1.694	2.606×10^{3}	9.931×10^{2}
	ELM-BIP	6.424×10^{-1}	4.293×10^{-1}	3.147×10^{-1}	2.491	6.591×10^{-1}
	R-ELM/BIP	3.927	1.106	2.609×10^{-1}	21.54	6.792
	RELM	1.548×10^{6}	6.504×10^{5}	12.01	6.373×10^{6}	2.256×10^{6}
	IRWLS-ELM	4.164×10^{4}	2.334×10^{3}	1.517	2.147×10^{5}	7.825×10^{4}

Source: author.

6.1.2 Auto-MPG dataset

Figure 20 – Testing RMSE of robust ELM networks with 1 sided contamination (Auto-MPG dataset).

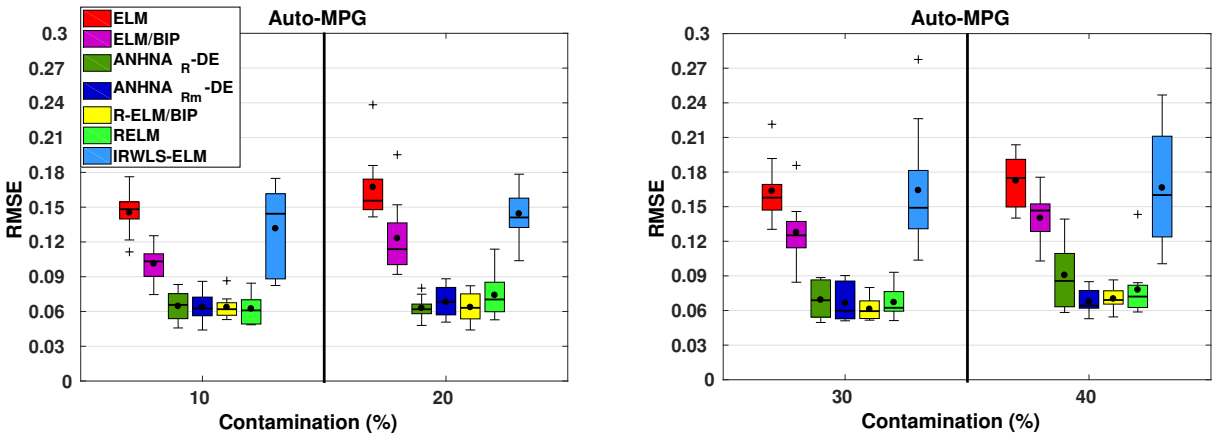
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 21 – Testing RMSE of robust ELM networks with 2 sided contamination (Auto-MPG dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

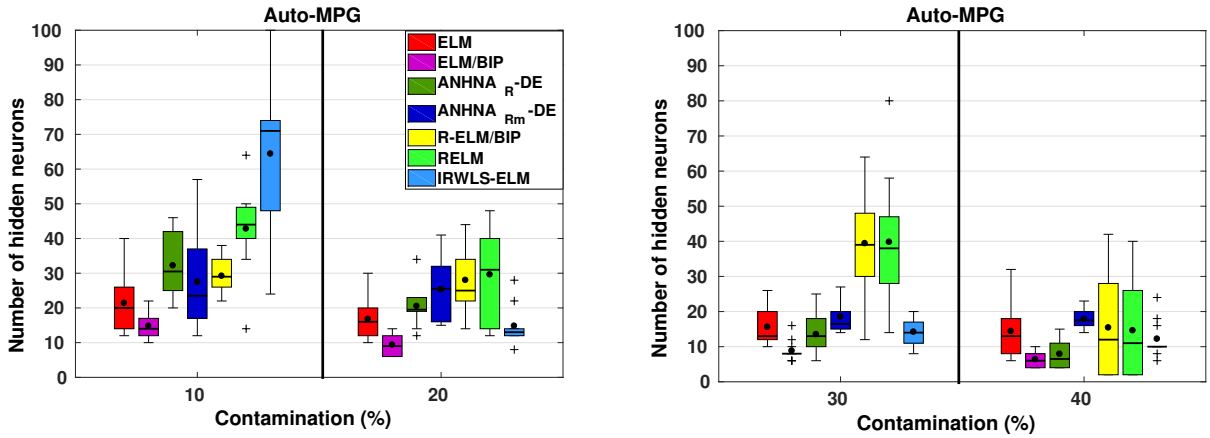
From this example on, we also present the results of ANHNA-DE$_R$ (represented in dark green) and ANHNA-DE$_{Rm}$ (represented in dark blue).

As it can be observed in Figures 20 and 21, the two robust ANHNA provided similar performances to RELM and R-ELM/BIP, which suffer far less from the increasing outlier contamination. ANHNA-DE$_{Rm}$ shows to be more robust than ANHNA-DE$_R$ in most cases. This is due to the fact that the trimmed RMSE applied in ANHNA-DE$_R$ only trims 10% of the largest errors, which becomes less effective when the rate of contamination

increases. The median, in this case, has shown to be a more robust method to be adopted. The IRWLS-ELM was the worst of all robust approaches and, as expected, the non-robust versions were highly affected by outliers.

Figure 22 – Number of hidden neurons of robust ELM networks with 1 sided contamination (Auto-MPG dataset).

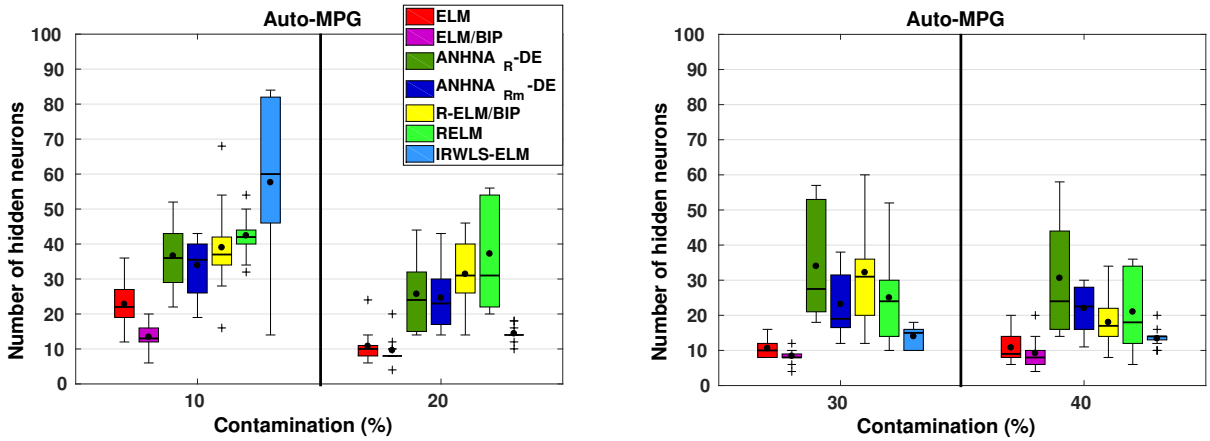
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 23 – Number of hidden neurons of robust ELM networks with 2 sided contamination (Auto-MPG dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

As for the number of hidden neurons in Figures 22 and 23, IRWLS-ELM stood out among the robust methods with less than 20 neurons in 20% to 40% of contamination in both cases. We can also notice that, between ANHNA versions, ANHNA-DE$_R$ provided the smallest amount of hidden neurons in one-sided contamination case, while ANHNA-DE$_{Rm}$ stood out in the two-sided case.

Once again, the methods that applied BIP has shown norms of the output

Table 9 – Output weight euclidean norm with 1 sided contamination (Auto-MPG dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	1.821×10^{2}	1.399×10^{2}	5.027×10^{2}	3.555×10^{2}	1.069×10^{2}
	ELM/BIP	1.438	1.220	8.676×10^{-1}	2.667	5.898×10^{-1}
	ANHNA$_{R}$-DE	6.390×10^{8}	7.152×10^{5}	1.198×10^{2}	6.341×10^{9}	2.003×10^{9}
	ANHNA$_{Rm}$-DE	4.077×10^{9}	2.447×10^{4}	1.635×10^{2}	4.045×10^{10}	1.278×10^{10}
	R-ELM/BIP	1.810	1.568	1.135	2.861	5.698×10^{-1}
	RELM	1.577×10^{2}	1.379×10^{2}	2.961×10^{2}	3.442×10^{2}	1.006×10^{2}
	IRWLS-ELM	2.260×10^{4}	1.290×10^{4}	1.815×10^{2}	1.186×10^{5}	3.469×10^{4}
20%	ELM	1.437×10^{2}	1.308×10^{2}	2.760×10^{2}	3.089×10^{2}	7.648×10^{2}
	ELM/BIP	1.271	1.212	4.008×10^{-1}	1.961	5.918×10^{-1}
	ANHNA$_{R}$-DE	6.278×10^{6}	1.831×10^{5}	6.256×10^{2}	5.188×10^{7}	1.630×10^{7}
	ANHNA$_{Rm}$-DE	6.637×10^{9}	1.158×10^{4}	4.178×10^{2}	6.636×10^{10}	2.098×10^{10}
	R-ELM/BIP	1.984	1.637	1.032	3.890	9.762×10^{-1}
	RELM	7.504×10^{2}	5.339×10^{2}	1.254×10^{2}	2.064×10^{2}	6.053×10^{2}
	IRWLS-ELM	6.801×10^{2}	5.159×10^{2}	1.963×10^{2}	1.783×10^{2}	5.462×10^{2}
30%	ELM	1.523×10^{2}	1.304×10^{2}	6.258×10^{2}	2.910×10^{2}	8.763×10^{2}
	ELM/BIP	1.705	1.099	6.520×10^{-1}	4.404	1.436
	ANHNA$_{R}$-DE	9.520×10^{9}	3.581×10^{2}	13.30	9.520×10^{10}	3.010×10^{10}
	ANHNA$_{Rm}$-DE	1.876×10^{6}	1.235×10^{4}	1.599×10^{2}	1.031×10^{7}	3.425×10^{6}
	R-ELM/BIP	4.252	4.185	1.129	8.300	2.653
	RELM	3.278×10^{2}	9.907×10^{2}	2.261×10^{2}	2.280×10^{4}	6.889×10^{2}
	IRWLS-ELM	9.403×10^{2}	8.157×10^{2}	1.436×10^{2}	1.776×10^{2}	5.670×10^{2}
40%	ELM	1.428×10^{2}	1.281×10^{2}	45.92	3.585×10^{2}	1.200×10^{2}
	ELM/BIP	1.478	9.445×10^{-1}	6.188×10^{-1}	3.574	1.020
	ANHNA$_{R}$-DE	2.713×10^{4}	94.56	4.715	2.529×10^{5}	7.947×10^{4}
	ANHNA$_{Rm}$-DE	3.445×10^{5}	3.977×10^{4}	2.154×10^{2}	2.683×10^{6}	8.366×10^{5}
	R-ELM/BIP	2.061	1.118	4.165×10^{-1}	9.134	2.637
	RELM	5.088×10^{2}	1.652×10^{2}	1.038	2.194×10^{2}	7.257×10^{2}
	IRWLS-ELM	1.115×10^{2}	6.424×10^{2}	15.93	4.548×10^{2}	1.383×10^{2}

Source: author.

weight vectors (see Table 9 and 10) with smaller mean and variation than the remaining algorithms along the different rate of contamination. Moreover, the two robust ANHNA provided the largest norms of them all. This is due to the fact that their fitness function takes into account only the trimmed validation RMSE or its median. So, these versions of robust ANHNA will search only for the best RMSE performance solution. It is interesting to notice that, even though ANHNA's solution provide large output weight norms, it does not overfit, as demonstrated by their testing results.

We can also see that the trade-off between RMSE performance, number of hidden neurons and norm of output weight vector is difficult to balance. So far, the network that stood out considering all cases was the R-ELM/BIP with one of the best RMSE performances allied to the smallest norm among the robust methods. Nevertheless,

Table 10 – Output weight euclidean norm with 2 sided contamination (Auto-MPG dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	1.542×10^{2}	1.326×10^{2}	4.183×10^{2}	2.817×10^{2}	7.372×10^{2}
	ELM/BIP	1.680	1.697	4.966×10^{-1}	2.891	7.763×10^{-1}
	ANHNA$_{R}$-DE	3.676×10^{9}	2.041×10^{7}	3.622×10^{2}	2.742×10^{10}	8.637×10^{9}
	ANHNA$_{Rm}$-DE	1.174×10^{9}	2.549×10^{5}	1.258×10^{2}	1.108×10^{10}	3.481×10^{9}
	R-ELM/BIP	3.051	2.355	7.738×10^{-1}	7.933	2.042
	RELM	1.428×10^{2}	1.278×10^{2}	8.711×10^{2}	2.403×10^{2}	5.737×10^{2}
	IRWLS-ELM	2.080×10^{4}	7.567×10^{2}	6.667×10^{2}	8.060×10^{4}	2.624×10^{4}
20%	ELM	5.147×10^{2}	4.272×10^{2}	9.855	1.408×10^{2}	4.855×10^{2}
	ELM/BIP	1.136	7.889×10^{-1}	5.221×10^{-1}	2.684	6.988×10^{-1}
	ANHNA$_{R}$-DE	7.697×10^{9}	8.232×10^{4}	2.994×10^{2}	7.692×10^{10}	2.432×10^{10}
	ANHNA$_{Rm}$-DE	4.243×10^{9}	1.111×10^{5}	52.24	4.242×10^{10}	1.342×10^{10}
	R-ELM/BIP	2.640	2.463	1.233	5.880	1.317
	RELM	1.223×10^{2}	7.238×10^{2}	3.944×10^{2}	3.014×10^{2}	9.490×10^{2}
	IRWLS-ELM	6.449×10^{2}	5.557×10^{2}	3.187×10^{2}	1.142×10^{2}	2.482×10^{2}
30%	ELM	6.738×10^{2}	5.035×10^{2}	1.331×10^{2}	2.297×10^{2}	6.679×10^{2}
	ELM/BIP	1.348	1.126	5.740×10^{-1}	3.355	8.205×10^{-1}
	ANHNA$_{R}$-DE	1.072×10^{9}	6.187×10^{5}	9.024×10^{2}	9.225×10^{9}	2.881×10^{9}
	ANHNA$_{Rm}$-DE	3.829×10^{5}	6.817×10^{4}	3.096×10^{2}	1.797×10^{6}	6.971×10^{5}
	R-ELM/BIP	3.649	2.703	9.821×10^{-1}	9.599	2.919
	RELM	7.769×10^{2}	5.971×10^{2}	1.250×10^{2}	2.400×10^{2}	6.521×10^{2}
	IRWLS-ELM	7.876×10^{2}	8.376×10^{2}	24.24	1.892×10^{2}	5.798×10^{2}
40%	ELM	4.835×10^{2}	2.552×10^{2}	12.78	1.227×10^{2}	4.580×10^{2}
	ELM/BIP	1.158	9.475×10^{-1}	5.144×10^{-1}	2.884	7.082×10^{-1}
	ANHNA$_{R}$-DE	2.772×10^{6}	7.864×10^{4}	6.123×10^{2}	8.472×10^{6}	3.693×10^{6}
	ANHNA$_{Rm}$-DE	1.639×10^{9}	2.089×10^{4}	8.121×10^{2}	1.399×10^{10}	4.405×10^{9}
	R-ELM/BIP	1.348	1.189	6.977×10^{-1}	2.172	5.332×10^{-1}
	RELM	8.007×10^{2}	5.155×10^{2}	3.927	2.478×10^{2}	8.641×10^{2}
	IRWLS-ELM	7.028×10^{2}	5.862×10^{2}	1.157×10^{2}	1.676×10^{2}	4.946×10^{2}

Source: author.

it was not so successful, considering the number of hidden neurons.

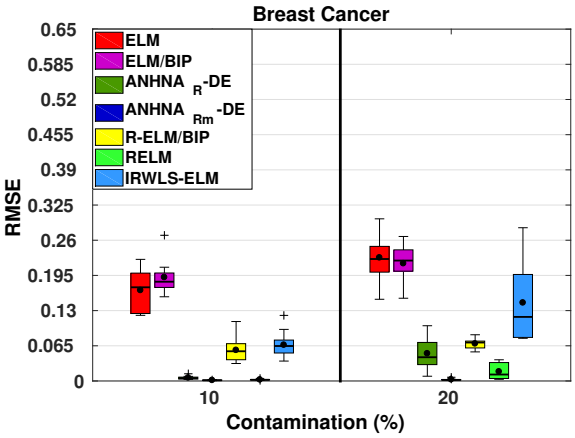
6.1.3 Breast Cancer dataset

From the exposed in Figures 24 and 25, it is clear that the ANHNA-DE$_{Rm}$ is the best performing method. Since all methods, including ELM and ELM/BIP, were chosen based on the trimmed RMSE, we hypothesized that the median used in ANHNA-DE$_{Rm}$ provided a more robust measure considering that both training and validation sets are corrupted. The two other best performances belong to RELM and ANHNA-DE$_{R}$, although their performances deteriorated with the increasing rate of contamination.

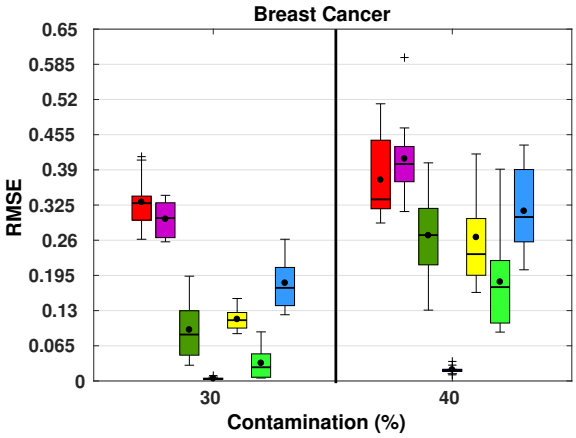
The ELM/BIP and R-ELM/BIP, as in the previously presented datasets, still maintains the smallest norms of all tested algorithms in all cases studied.

Figure 24 – Testing RMSE of robust ELM networks with 1 sided contamination (Breast Cancer dataset).

(a) 10% and 20%.



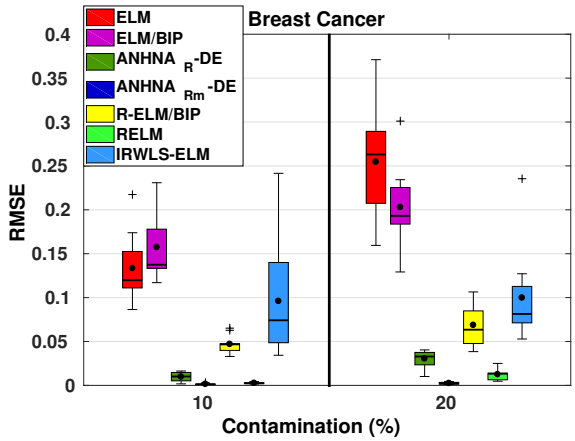
(b) 30% and 40%.



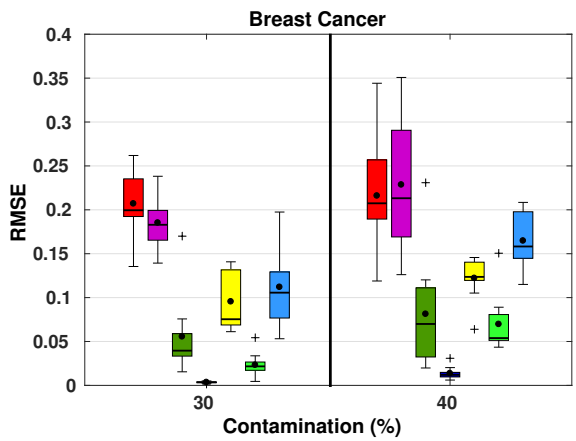
Source: author.

Figure 25 – Testing RMSE of robust ELM networks with 2 sided contamination (Breast Cancer dataset).

(a) 10% and 20%.



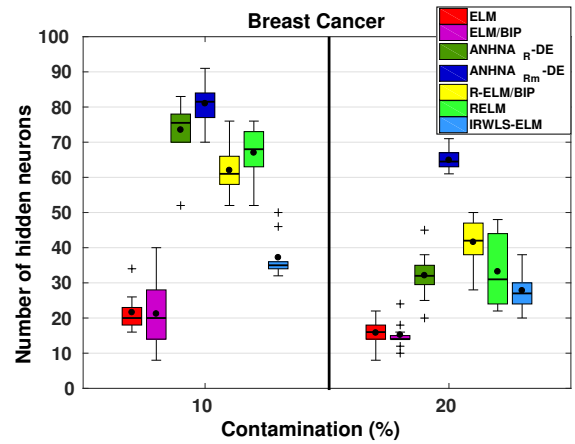
(b) 30% and 40%.



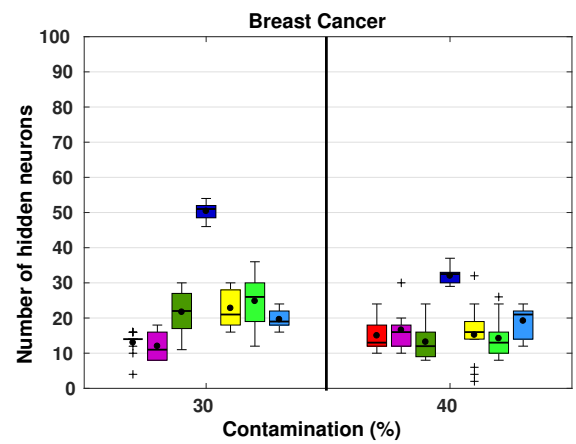
Source: author.

Figure 26 – Number of hidden neurons of robust ELM networks with 1 sided contamination (Breast Cancer dataset).

(a) 10% and 20%.



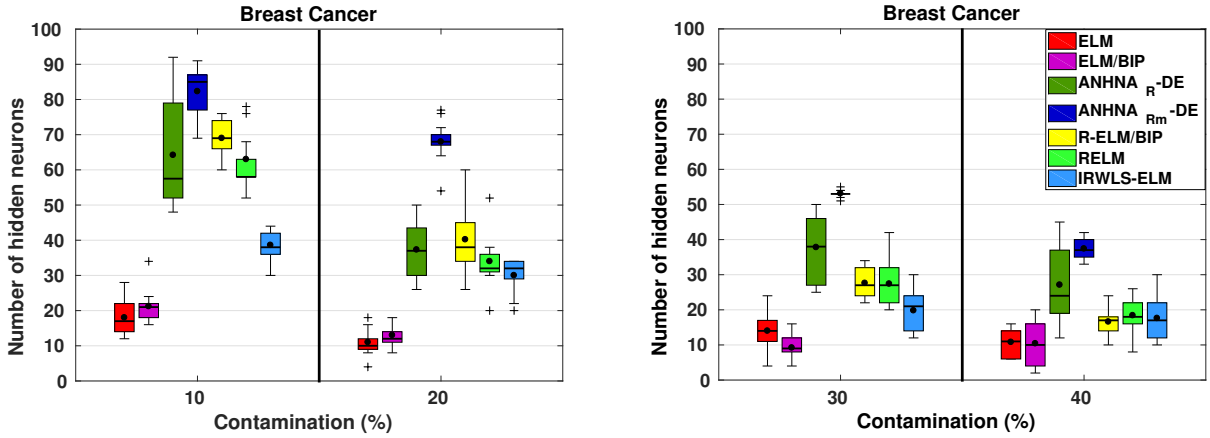
(b) 30% and 40%.



Source: author.

Figure 27 – Number of hidden neurons of robust ELM networks with 2 sided contamination (Breast Cancer dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Table 11 – Output weight euclidean norm with 1 sided contamination (Breast Cancer dataset).

		Mean	Median	Minimum	Maximum	Std.
10%	ELM	11.72	7.193	3.409	43.74	12.09
	ELM/BIP	9.096×10^{-1}	9.134×10^{-1}	7.073×10^{-1}	1.138	1.280×10^{-1}
	ANHNA$_R$-DE	6.547×10^{2}	39.56	9.096	6.065×10^{2}	1.902×10^{2}
	ANHNA$_{Rm}$-DE	42.23	29.23	8.923	1.271×10^{2}	36.12
	R-ELM/BIP	7.176×10^{-1}	7.075×10^{-1}	5.403×10^{-1}	9.495×10^{-1}	1.564×10^{-1}
	RELM	2.451	2.421	2.031	3.134	3.257×10^{-1}
	IRWLS-ELM	15.38	12.53	9.279	30.38	7.123
20%	ELM	6.196	5.517	3.783	10.29	1.978
	ELM/BIP	8.694×10^{-1}	8.883×10^{-1}	6.557×10^{-1}	1.050	1.426×10^{-1}
	ANHNA$_R$-DE	6.391×10^{7}	2.437×10^{4}	1.397×10^{2}	6.382×10^{8}	2.018×10^{8}
	ANHNA$_{Rm}$-DE	34.54	15.75	7.807	1.648×10^{2}	48.29
	R-ELM/BIP	7.270×10^{-1}	7.323×10^{-1}	6.078×10^{-1}	8.879×10^{-1}	9.021×10^{-2}
	RELM	3.424	3.535	2.713	4.531	6.047×10^{-1}
	IRWLS-ELM	16.86	8.884	5.196	59.18	16.54
30%	ELM	7.696	6.108	3.826	17.66	4.022
	ELM/BIP	8.591×10^{-1}	8.350×10^{-1}	4.731×10^{-1}	1.214	2.521×10^{-1}
	ANHNA$_R$-DE	1.053×10^{9}	4.252×10^{2}	54.62	1.053×10^{10}	3.331×10^{9}
	ANHNA$_{Rm}$-DE	41.36	26.97	11.01	1.016×10^{2}	33.36
	R-ELM/BIP	9.742×10^{-1}	9.309×10^{-1}	6.920×10^{-1}	1.196	1.595×10^{-1}
	RELM	4.447	4.179	3.045	6.206	1.274
	IRWLS-ELM	7.564	7.419	4.987	10.61	2.162
40%	ELM	7.589	5.837	4.186	14.78	3.582
	ELM/BIP	1.210	1.208	6.889×10^{-1}	1.598	2.665×10^{-1}
	ANHNA$_R$-DE	5.897×10^{4}	9.327×10^{2}	10.80	4.631×10^{5}	1.458×10^{5}
	ANHNA$_{Rm}$-DE	62.97	49.25	20.30	1.789×10^{2}	46.54
	R-ELM/BIP	1.129	9.661×10^{-1}	5.522×10^{-1}	2.010	5.372×10^{-1}
	RELM	4.940	3.417	2.661	16.34	4.168
	IRWLS-ELM	10.26	8.887	5.967	21.04	4.729

Source: author.

Table 12 – Output weight euclidean norm with 2 sided contamination (Breast Cancer dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	7.305	7.135	5.560	10.23	1.508
	ELM/BIP	8.719×10^{-1}	8.731×10^{-1}	6.522×10^{-1}	1.094	1.645×10^{-1}
	ANHNA$_{R}$-DE	5.996×10^{9}	2.373×10^{2}	17.81	5.983×10^{10}	1.892×10^{10}
	ANHNA$_{Rm}$-DE	27.84	16.02	7.195	75.83	24.84
	R-ELM/BIP	6.559×10^{-1}	6.465×10^{-1}	4.910×10^{-1}	8.032×10^{-1}	9.020×10^{-2}
	RELM	2.557	2.602	1.925	3.102	3.712×10^{-1}
	IRWLS-ELM	24.20	18.18	12.37	60.13	14.46
20%	ELM	4.193	3.773	1.711	8.023	1.672
	ELM/BIP	8.107×10^{-1}	7.863×10^{-1}	5.180×10^{-1}	1.070	1.875×10^{-1}
	ANHNA$_{R}$-DE	1.462×10^{10}	5.648×10^{5}	2.716×10^{2}	1.462×10^{11}	4.622×10^{10}
	ANHNA$_{Rm}$-DE	38.59	27.75	10.21	1.380×10^{2}	38.23
	R-ELM/BIP	8.062×10^{-1}	7.797×10^{-1}	6.380×10^{-1}	1.152	1.441×10^{-1}
	RELM	3.609	3.346	2.366	5.274	9.031×10^{-1}
	IRWLS-ELM	14.26	14.30	3.310	28.74	8.999
30%	ELM	6.605	4.986	1.626	19.16	4.934
	ELM/BIP	8.210×10^{-1}	8.274×10^{-1}	4.607×10^{-1}	1.148	1.876×10^{-1}
	ANHNA$_{R}$-DE	8.318×10^{10}	7.841×10^{6}	67.55	6.109×10^{11}	1.958×10^{11}
	ANHNA$_{Rm}$-DE	29.08	23.97	8.589	78.18	20.96
	R-ELM/BIP	8.443×10^{-1}	8.012×10^{-1}	6.842×10^{-1}	1.100	1.375×10^{-1}
	RELM	3.752	3.172	2.559	5.767	1.135
	IRWLS-ELM	6.337	5.397	2.629	12.65	3.116
40%	ELM	4.570	3.405	2.467	8.423	2.401
	ELM/BIP	1.012	9.393×10^{-1}	4.043×10^{-1}	1.803	4.751×10^{-1}
	ANHNA$_{R}$-DE	8.772×10^{6}	7.049×10^{2}	1.031×10^{2}	8.733×10^{7}	2.760×10^{7}
	ANHNA$_{Rm}$-DE	94.17	71.99	7.714	2.618×10^{2}	89.64
	R-ELM/BIP	8.908×10^{-1}	7.976×10^{-1}	5.500×10^{-1}	1.353	2.915×10^{-1}
	RELM	4.780	3.877	2.488	9.408	2.534
	IRWLS-ELM	8.338	5.910	2.427	32.33	8.581

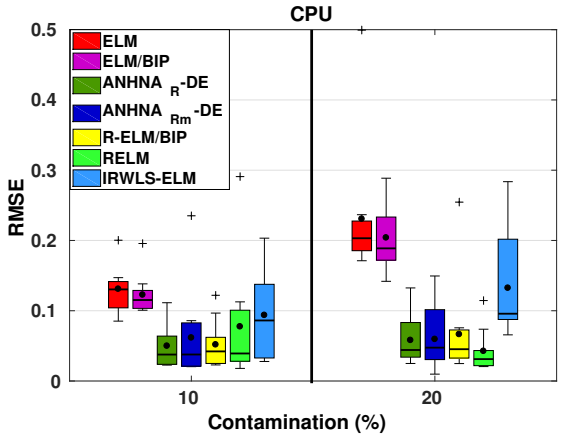
Source: author.

For this dataset ANHNA-DE$_{Rm}$ provided the the best performance when considering RMSE performance and a small norms of output weight vectors (see Tables 11 and 12). Although it needed a large number of hidden neurons (see Figures 26 and 27), it also accomplished performances that varied less than the other evaluated methods plus combined with norms surprisingly small considering the size of the hidden layer. The ANHNA-DE$_{R}$ showed a good RMSE performance, but considering the number of hidden neurons and their high weight norms, it did not excel as a viable solution. That being said, plus the fact that IRWLS-ELM provided the poorest performance among the evaluated robust methods, RELM and R-ELM/BIP offered the second and third best solutions, respectively, for this set.

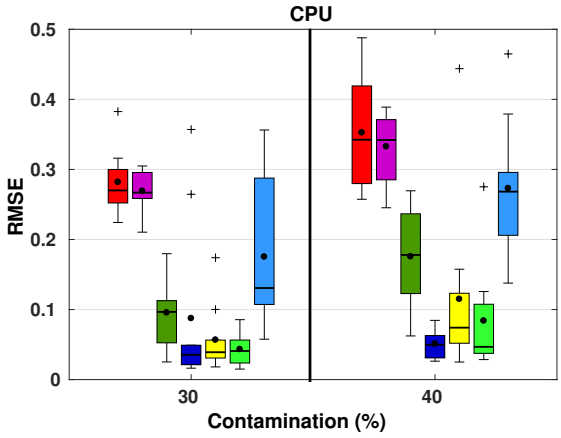
6.1.4 CPU dataset

Figure 28 – Testing RMSE of robust ELM networks with 1 sided contamination (CPU dataset).

(a) 10% and 20%.



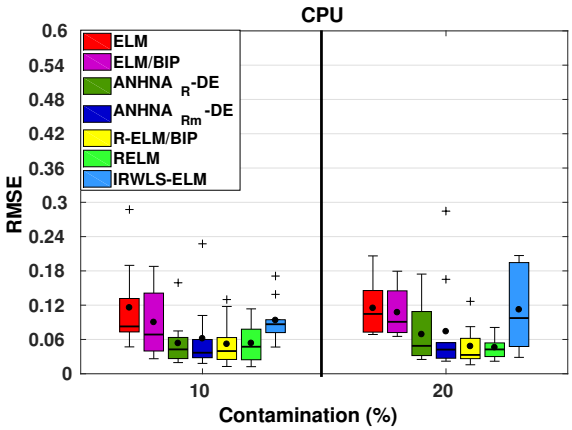
(b) 30% and 40%.



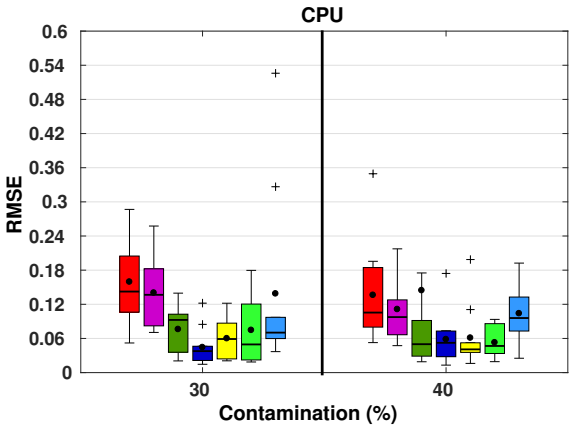
Source: author.

Figure 29 – Testing RMSE of robust ELM networks with 2 sided contamination (CPU dataset).

(a) 10% and 20%.



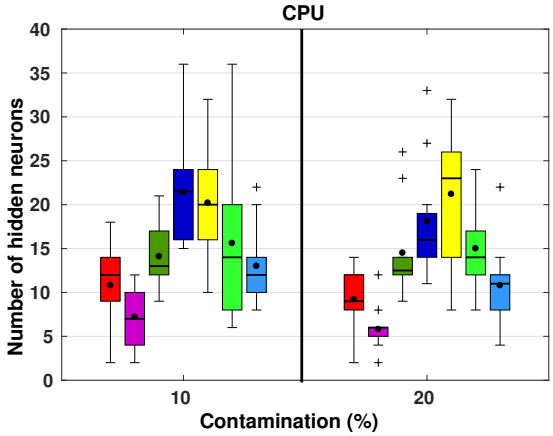
(b) 30% and 40%.



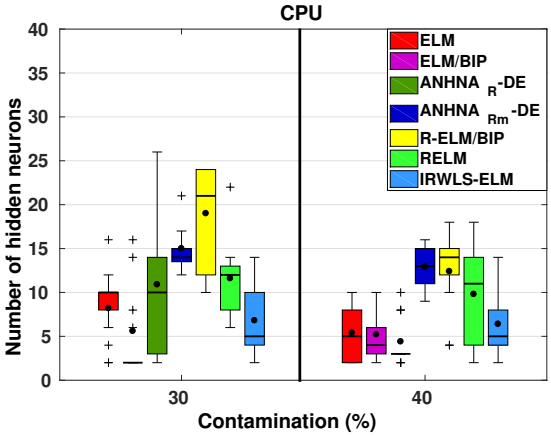
Source: author.

Figure 30 – Number of hidden neurons of robust ELM networks with 1 sided contamination (CPU dataset).

(a) 10% and 20%.



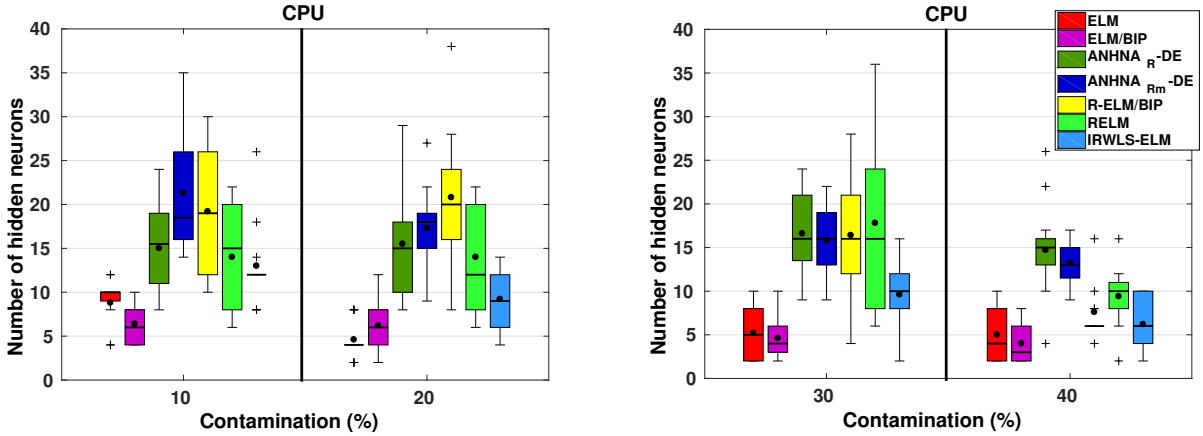
(b) 30% and 40%.



Source: author.

Figure 31 – Number of hidden neurons of robust ELM networks with 2 sided contamination (CPU dataset).

(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Table 13 – Output weight euclidean norm with 1 sided contamination (CPU dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	1.715×10^{3}	1.257×10^{3}	4.149	5.593×10^{3}	1.846×10^{3}
	ELM/BIP	8.101×10^{-1}	8.058×10^{-1}	3.536×10^{-1}	1.229	3.102×10^{-1}
	ANHNA$_R$-DE	1.457×10^{7}	5.564×10^{4}	2.304×10^{2}	1.345×10^{8}	4.221×10^{7}
	ANHNA$_{Rm}$-DE	2.170×10^{9}	4.086×10^{5}	1.312×10^{3}	2.165×10^{10}	6.846×10^{9}
	R-ELM/BIP	7.595×10^{-1}	7.740×10^{-1}	2.823×10^{-1}	1.532	3.999×10^{-1}
	RELM	8.874×10^{2}	4.431×10^{2}	13.75	5.449×10^{3}	1.624×10^{3}
	IRWLS-ELM	1.017×10^{3}	7.781×10^{2}	6.395×10^{2}	2.479×10^{3}	5.756×10^{2}
20%	ELM	1.306×10^{3}	1.092×10^{3}	4.333	4.533×10^{3}	1.392×10^{3}
	ELM/BIP	8.131×10^{-1}	7.052×10^{-1}	4.127×10^{-1}	1.450	3.508×10^{-1}
	ANHNA$_R$-DE	2.462×10^{9}	6.770×10^{4}	1.882×10^{2}	2.461×10^{10}	7.781×10^{9}
	ANHNA$_{Rm}$-DE	2.319×10^{9}	1.800×10^{4}	3.211×10^{3}	1.714×10^{10}	5.545×10^{9}
	R-ELM/BIP	1.269	9.110×10^{-1}	3.459×10^{-1}	3.059	8.874×10^{-1}
	RELM	5.970×10^{2}	3.927×10^{2}	2.131×10^{2}	1.521×10^{3}	4.822×10^{2}
	IRWLS-ELM	1.409×10^{3}	7.997×10^{2}	4.635	5.299×10^{3}	1.595×10^{3}
30%	ELM	7.667×10^{2}	7.340×10^{2}	4.495	2.647×10^{3}	8.230×10^{2}
	ELM/BIP	7.410×10^{-1}	5.129×10^{-1}	3.000×10^{-1}	1.522	4.189×10^{-1}
	ANHNA$_R$-DE	8.867×10^{9}	2.598×10^{3}	4.709	8.866×10^{10}	2.804×10^{10}
	ANHNA$_{Rm}$-DE	2.731×10^{4}	6.875×10^{3}	5.366×10^{2}	8.774×10^{4}	3.680×10^{4}
	R-ELM/BIP	1.686	1.488	3.249×10^{-1}	4.617	1.316
	RELM	5.439×10^{2}	3.202×10^{2}	1.674	1.871×10^{3}	5.362×10^{2}
	IRWLS-ELM	6.223×10^{2}	2.525×10^{2}	2.264	2.738×10^{3}	8.828×10^{2}
40%	ELM	5.278×10^{2}	14.67	3.231	2.507×10^{3}	8.695×10^{2}
	ELM/BIP	9.217×10^{-1}	8.725×10^{-1}	4.323×10^{-1}	1.692	4.266×10^{-1}
	ANHNA$_R$-DE	1.604×10^{3}	1.208×10^{2}	3.079	7.359×10^{3}	2.811×10^{3}
	ANHNA$_{Rm}$-DE	1.442×10^{5}	1.211×10^{4}	4.112×10^{2}	1.026×10^{6}	3.262×10^{5}
	R-ELM/BIP	1.265	1.207	3.102×10^{-1}	3.217	9.147×10^{-1}
	RELM	7.633×10^{2}	3.472×10^{2}	1.395	3.561×10^{3}	1.117×10^{3}
	IRWLS-ELM	4.844×10^{2}	10.32	3.534	2.568×10^{3}	8.425×10^{2}

Source: author.

In Figures 28 and 29, ANHNA-DE$_{Rm}$, RELM and R-ELM/BIP presented similar RMSE performances. On the number of hidden neurons, from the evaluated robust methods, the IRWLS-ELM provided solutions with fewer neurons than the others.

As for the norms of the output weight vectors (see Tables 13 and 14), we may observe once more that ELM/BIP and R-ELM/BIP provided the smallest and stabler norms of all methods, independently of the rate of outlier contamination. Just as in iCub and Auto-MPG examples, all robust ANHNA versions provided the highest norms.

Table 14 – Output weight euclidean norm with 2 sided contamination (CPU dataset).

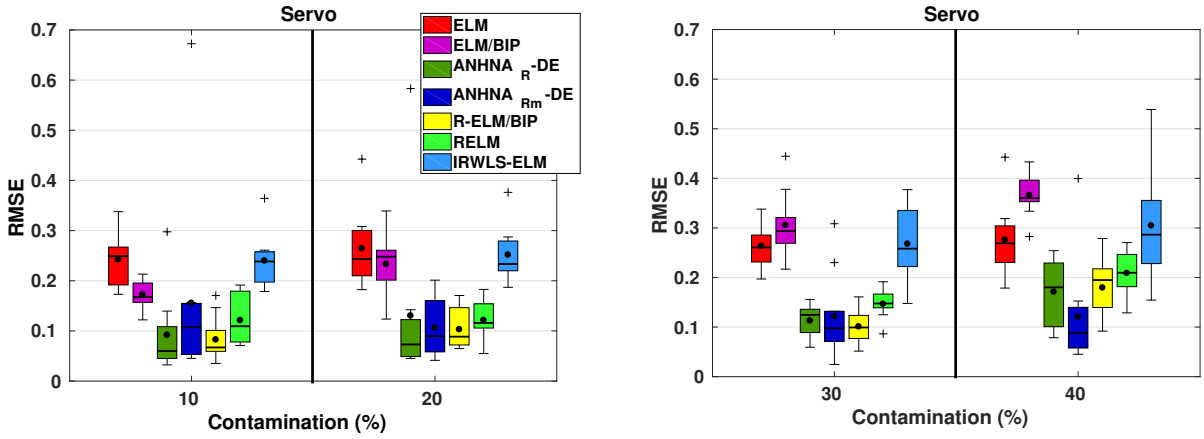
		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	6.278×10^{2}	4.877×10^{2}	4.262	2.144×10^{3}	6.077×10^{2}
	ELM/BIP	5.687×10^{-1}	5.369×10^{-1}	3.128×10^{-1}	9.218×10^{-1}	2.024×10^{-1}
	ANHNA$_{R}$-DE	3.180×10^{5}	2.044×10^{4}	6.065×10^{2}	2.924×10^{6}	9.160×10^{5}
	ANHNA$_{Rm}$-DE	2.439×10^{8}	5.998×10^{4}	5.967×10^{2}	2.386×10^{9}	7.526×10^{8}
	R-ELM/BIP	7.726×10^{-1}	6.786×10^{-1}	3.036×10^{-1}	1.430	3.708×10^{-1}
	RELM	8.148×10^{2}	6.502×10^{2}	1.833	3.035×10^{3}	8.631×10^{2}
	IRWLS-ELM	1.483×10^{3}	1.121×10^{3}	1.946×10^{2}	5.112×10^{3}	1.467×10^{3}
20%	ELM	2.308×10^{2}	12.23	1.889	9.586×10^{2}	3.677×10^{2}
	ELM/BIP	6.997×10^{-1}	3.982×10^{-1}	2.608×10^{-1}	1.951	5.784×10^{-1}
	ANHNA$_{R}$-DE	2.242×10^{6}	7.113×10^{4}	1.762×10^{3}	1.904×10^{7}	5.959×10^{6}
	ANHNA$_{Rm}$-DE	5.296×10^{5}	5.229×10^{3}	2.888×10^{2}	4.642×10^{6}	1.452×10^{6}
	R-ELM/BIP	9.012×10^{-1}	5.583×10^{-1}	3.876×10^{-1}	2.505	7.240×10^{-1}
	RELM	4.356×10^{2}	4.153×10^{2}	4.237	8.386×10^{2}	3.122×10^{2}
	IRWLS-ELM	6.843×10^{2}	5.398×10^{2}	4.502	2.445×10^{3}	7.653×10^{2}
30%	ELM	2.104×10^{2}	15.37	2.011	8.700×10^{2}	3.332×10^{2}
	ELM/BIP	6.900×10^{-1}	6.628×10^{-1}	1.867×10^{-1}	1.428	4.395×10^{-1}
	ANHNA$_{R}$-DE	3.200×10^{7}	1.807×10^{4}	1.067×10^{2}	2.992×10^{8}	9.402×10^{7}
	ANHNA$_{Rm}$-DE	9.037×10^{4}	6.416×10^{3}	8.776×10^{2}	8.462×10^{5}	2.656×10^{5}
	R-ELM/BIP	7.281×10^{-1}	5.820×10^{-1}	2.253×10^{-1}	1.654	4.699×10^{-1}
	RELM	1.477×10^{3}	7.300×10^{2}	1.914	5.061×10^{3}	1.771×10^{3}
	IRWLS-ELM	1.708×10^{3}	8.776×10^{2}	1.194	9.569×10^{3}	2.845×10^{3}
40%	ELM	2.641×10^{2}	3.259	1.614	1.605×10^{3}	4.991×10^{2}
	ELM/BIP	4.262×10^{-1}	3.460×10^{-1}	1.633×10^{-1}	1.286	3.213×10^{-1}
	ANHNA$_{R}$-DE	1.883×10^{5}	9.369×10^{4}	1.065×10^{2}	6.287×10^{5}	2.275×10^{5}
	ANHNA$_{Rm}$-DE	1.439×10^{10}	1.158×10^{4}	7.999×10^{2}	1.439×10^{11}	4.549×10^{10}
	R-ELM/BIP	4.081×10^{-1}	3.822×10^{-1}	2.015×10^{-1}	7.398×10^{-1}	1.402×10^{-1}
	RELM	2.856×10^{2}	1.634×10^{2}	2.859	7.576×10^{2}	2.579×10^{2}
	IRWLS-ELM	2.877×10^{2}	1.231×10^{2}	9.430×10^{-1}	1.063×10^{3}	3.915×10^{2}

Source: author.

6.1.5 Servo dataset

Figure 32 – Testing RMSE of robust ELM networks with 1 sided contamination (Servo dataset).

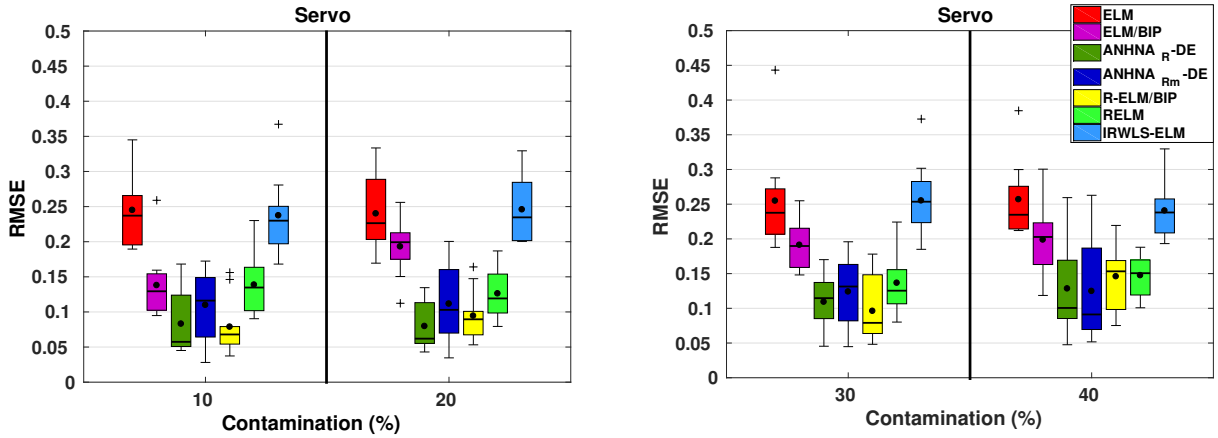
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 33 – Testing RMSE of robust ELM networks with 2 sided contamination (Servo dataset).

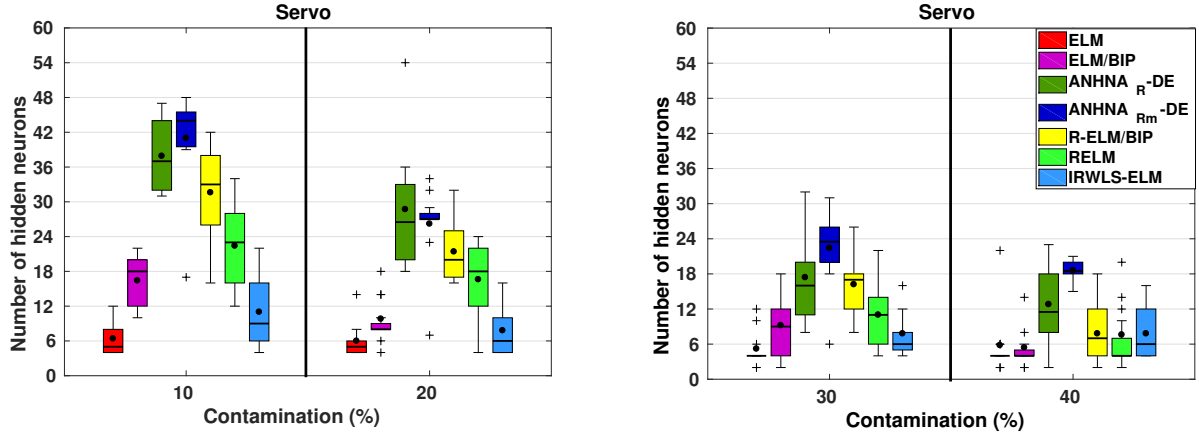
(a) 10% and 20%. (b) 30% and 40%.



Source: author.

Figure 34 – Number of hidden neurons of robust ELM networks with 1 sided contamination (Servo dataset).

(a) 10% and 20%. (b) 30% and 40%.

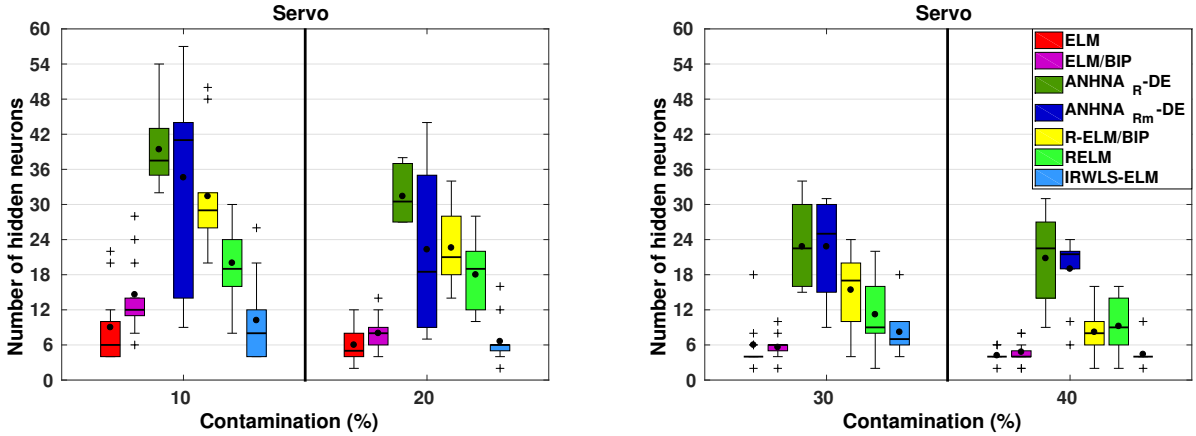


Source: author.

Figure 35 – Number of hidden neurons of robust ELM networks with 2 sided contamination (Servo dataset).

(a) 10% and 20%.

(b) 30% and 40%.



Source: author.

Table 15 – Output weight euclidean norm with 1 sided contamination (Servo dataset).

		Mean	Median	Minimum	Maximum	Std.
10%	ELM	1.003×10^{2}	1.816×10^{2}	5.168	5.230×10^{2}	1.632×10^{2}
	ELM/BIP	5.316	5.080	1.426	11.22	3.085
	ANHNA$_R$-DE	2.829×10^{9}	2.739×10^{7}	1.909×10^{5}	2.718×10^{10}	8.559×10^{9}
	ANHNA$_{Rm}$-DE	3.204×10^{9}	1.786×10^{8}	2.673×10^{4}	1.535×10^{10}	6.015×10^{9}
	R-ELM/BIP	4.549	4.282	1.998	7.614	1.852
	RELM	5.067×10^{4}	1.786×10^{4}	7.961×10^{2}	1.765×10^{5}	6.238×10^{4}
	IRWLS-ELM	3.546×10^{2}	6.445×10^{2}	6.328	1.330×10^{4}	4.836×10^{2}
20%	ELM	7.109×10^{2}	3.636×10^{2}	3.837	2.349×10^{2}	8.800×10^{2}
	ELM/BIP	1.792	1.322	5.014×10^{-1}	3.801	1.168
	ANHNA$_R$-DE	1.644×10^{10}	4.986×10^{5}	2.197×10^{2}	1.644×10^{11}	5.198×10^{10}
	ANHNA$_{Rm}$-DE	5.239×10^{8}	2.189×10^{6}	1.419×10^{4}	5.077×10^{9}	1.600×10^{9}
	R-ELM/BIP	3.494	2.898	1.599	5.984	1.714
	RELM	5.086×10^{2}	2.187×10^{2}	5.986	1.881×10^{4}	6.828×10^{2}
	IRWLS-ELM	1.418×10^{2}	3.772×10^{2}	5.664	7.693×10^{2}	2.373×10^{2}
30%	ELM	6.680×10^{2}	12.13	2.014	2.856×10^{2}	1.149×10^{2}
	ELM/BIP	2.803	1.213	4.610×10^{-1}	12.54	3.759
	ANHNA$_R$-DE	7.839×10^{5}	1.056×10^{4}	2.124×10^{2}	5.361×10^{6}	1.749×10^{6}
	ANHNA$_{Rm}$-DE	1.004×10^{8}	2.477×10^{5}	4.781×10^{2}	6.784×10^{8}	2.262×10^{8}
	R-ELM/BIP	2.635	2.357	1.137	5.140	1.189
	RELM	1.837×10^{2}	9.061×10^{2}	2.340	9.769×10^{2}	2.921×10^{2}
	IRWLS-ELM	1.584×10^{2}	1.468×10^{2}	10.66	4.105×10^{2}	1.594×10^{2}
40%	ELM	2.544×10^{4}	16.40	3.671	2.535×10^{5}	8.014×10^{4}
	ELM/BIP	1.176	6.230×10^{-1}	5.164×10^{-1}	4.950	1.356
	ANHNA$_R$-DE	1.760×10^{9}	4.936×10^{4}	14.88	1.746×10^{10}	5.518×10^{9}
	ANHNA$_{Rm}$-DE	1.745×10^{8}	1.211×10^{5}	6.987×10^{2}	1.695×10^{9}	5.343×10^{8}
	R-ELM/BIP	1.583	1.085	3.472×10^{-1}	4.301	1.268
	RELM	1.723×10^{2}	4.976	9.141×10^{-1}	8.461×10^{2}	3.040×10^{2}
	IRWLS-ELM	3.128×10^{2}	5.983×10^{2}	5.464	1.412×10^{4}	4.674×10^{2}

Source: author.

Table 16 – Output weight euclidean norm with 2 sided contamination (Servo dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	4.130×10^{2}	3.837×10^{2}	7.071	2.025×10^{4}	7.832×10^{2}
	ELM/BIP	3.230	2.340	5.375×10^{-1}	8.466	2.840
	ANHNA$_{R}$-DE	2.396×10^{9}	9.618×10^{6}	3.637×10^{5}	2.266×10^{10}	7.126×10^{9}
	ANHNA$_{Rm}$-DE	1.386×10^{9}	1.726×10^{8}	3.581×10^{2}	6.056×10^{9}	2.137×10^{9}
	R-ELM/BIP	5.137	3.879	1.991	10.25	2.929
	RELM	1.897×10^{4}	4.192×10^{2}	4.187×10^{2}	8.113×10^{4}	3.032×10^{4}
	IRWLS-ELM	8.176×10^{2}	6.385×10^{2}	11.48	5.805×10^{4}	1.833×10^{4}
20%	ELM	4.372×10^{2}	2.282×10^{2}	1.117	1.443×10^{2}	5.454×10^{2}
	ELM/BIP	1.453	7.873×10^{-1}	5.074×10^{-1}	5.208	1.565
	ANHNA$_{R}$-DE	1.457×10^{8}	1.684×10^{7}	5.456×10^{4}	7.912×10^{8}	2.814×10^{8}
	ANHNA$_{Rm}$-DE	2.598×10^{8}	6.373×10^{5}	2.648×10^{2}	2.586×10^{9}	8.173×10^{8}
	R-ELM/BIP	3.280	2.843	1.569	7.219	1.624
	RELM	1.678×10^{4}	2.536×10^{2}	6.538×10^{2}	1.280×10^{5}	3.940×10^{4}
	IRWLS-ELM	8.528×10^{2}	2.405×10^{2}	2.483	5.380×10^{2}	1.647×10^{2}
30%	ELM	9.334×10^{2}	37.20	1.961	8.502×10^{2}	2.664×10^{2}
	ELM/BIP	9.030×10^{-1}	6.155×10^{-1}	3.329×10^{-1}	3.267	8.809×10^{-1}
	ANHNA$_{R}$-DE	1.090×10^{10}	1.934×10^{6}	7.750×10^{2}	1.088×10^{11}	3.440×10^{10}
	ANHNA$_{Rm}$-DE	6.234×10^{8}	1.532×10^{6}	2.631×10^{2}	6.219×10^{9}	1.966×10^{9}
	R-ELM/BIP	2.077	2.073	9.629×10^{-1}	3.299	8.627×10^{-1}
	RELM	2.760×10^{2}	1.119×10^{2}	1.350	1.612×10^{4}	4.840×10^{2}
	IRWLS-ELM	3.047×10^{2}	8.760×10^{2}	95.14	1.628×10^{4}	5.115×10^{2}
40%	ELM	1.938×10^{2}	8.201	2.393	1.587×10^{2}	4.963×10^{2}
	ELM/BIP	6.225×10^{-1}	4.428×10^{-1}	2.654×10^{-1}	2.518	6.745×10^{-1}
	ANHNA$_{R}$-DE	8.250×10^{6}	1.022×10^{6}	1.823×10^{2}	6.234×10^{7}	1.927×10^{7}
	ANHNA$_{Rm}$-DE	6.026×10^{6}	2.595×10^{5}	3.490×10^{2}	5.646×10^{7}	1.773×10^{7}
	R-ELM/BIP	1.302	1.193	1.815×10^{-1}	2.393	8.339×10^{-1}
	RELM	1.293×10^{2}	8.044×10^{2}	8.445×10^{-1}	6.045×10^{2}	1.817×10^{2}
	IRWLS-ELM	2.901×10^{2}	18.69	1.163	2.572×10^{2}	8.041×10^{2}

Source: author.

As illustrated by Figures 32 and 33, we may observe again that ELM/BIP provided some outlier robustness when compared to ELM and even to IRWLS-ELM in the 2 sided case. We hypothesize that the combination of a small number of hidden neurons and a larger norm of output weight vector than other methods, as it is demonstrated in following figures and tables, contributes to the poor performance of IRWLS-ELM.

The robust ANHNA versions presents the largest number of neurons of all methods, followed by R-ELM/BIP, RELM and IRWLS-ELM in most of the cases illustrated in Figures 34 and 35.

Just as the majority of the examples presented previously, Tables 15 and 16 show that ELM/BIP and R-ELM/BIP provided small norms of output weight vector even with the worst case scenario of 40% of outlier contamination. The robust versions of

ANHNA presented very high norms, as in most datasets evaluated.

In the following chapter, we present the results from ANHNA for optimizing the traditional ELM network.

7 PERFORMANCE ANALYSIS OF THE ANHNA

In this chapter, we present the results from ANHNA using four different metaheuristics: Differential Evolution (ANHNA-DE), Self-Adaptive Differential Evolution (ANHNA-SADE), Particle Swarm Optimization with global neighborhood (ANHNA-PSO$_g$) and Particle Swarm Optimization with local neighborhood (ANHNA-PSO$_l$). More details about these algorithms, see Appendix A.

As mentioned before, besides ANHNA$_R$, we have eight variants, independently of the chosen metaheuristic. In order to optimize the evaluation of the proposed approach, we choose only one variant to represent each aforementioned ANHNA variant.

Just as in the previous chapter, we adopt three different information to evaluate their performance instead of only observing its numerical error: we search for the trade-off between the RMSE during testing, the selected number of neurons and the Euclidean norm of the output weight vector.

From what is exposed in Appendix D, on one hand, ANHNA-DE variant that uses only the RMSE as fitness function has the best RMSE performance overall, although it seldom exceeds the other variants regarding the minimum number of hidden neurons. Moreover, it is the worst of ANHNA's variants considering the final norm of the output weight vector. This behavior was already anticipated because it is usually the scenario found in many works in the literature. On the other hand, ANHNA-DE$_{NO}$ provides a solution with the smallest norm of the output weight vector of all and also the minimum number of hidden neurons among all methods compared. However, its RMSE performance is the worst of them. This result was also anticipated since it was expected that ANHNA-DE$_{NO}$ would choose solutions with fewer neurons and consequently smaller norms. Furthermore, due to the fact that better performances were associated with higher norms, they were then excluded in the selection step. Another consequence of this choice is that using fewer neurons, the network will most likely underfit.

With the trade-off in mind, only two variants succeeded evenly: ANHNA-DE$_{Re}$ and ANHNA-DE$_{ReCNv2}$. The former had a slightly better RMSE performance and norm of the output weight vector than the other ANHNA versions, while the latter provides a slightly fewer number of hidden neurons. Thus, henceforward, we elected ANHNA-DE$_{Re}$ to represent ANHNA-DE results.

For ANHNA-PSO variants, we found similar issues, where we also had to

compromise between precision performance, its number of neurons and norm of the output weight vector. For both ANHNA-PSO$_g$ and ANHNA-PSO$_l$, we elected their regularized variants ANHNA-PSO$_{gRe}$ and ANHNA-PSO$_{lRe}$, as well. With ANHNA-SADE there are, so far, only results without regularization. Furthermore, the test results for its variants were similar to one another. Thus, we adopted ANHNA-SADE original version to be represented in this chapter.

We selected six real-world regression problems datasets: Auto-MPG, Body Fat, Breast Cancer, CPU, iCub, and Servo. For more details about them, see Appendix B.

This chapter is organized as follows. Section 7.1 holds all figures and tables related to the comparative study carried in this chapter, followed by the discussion over those results.

7.1 ANHNA's comparative study

7.1.1 Auto-MPG dataset

Table 17 – Evaluation results of the norm of the output weight vector (Auto-MPG dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	2.411×10^{4}	1.273×10^{4}	5.878×10^{3}	6.912×10^{4}	2.313×10^{4}
ELM-BIP	4.559	3.714	2.663	9.287	2.177
ANHNA-DE$_{Re}$	93.65	16.30	8.554	7.832×10^{2}	2.424×10^{2}
ANHNA-PSO$_{gRe}$	25.37	16.56	7.522	65.26	19.29
ANHNA-PSO$_{lRe}$	41.18	18.34	9.324	1.709×10^{2}	53.26
ANHNA-SADE	9.578×10^{3}	5.094×10^{2}	1.729×10^{2}	5.337×10^{4}	1.925×10^{4}

Source: author.

Table 18 – Regularization parameters for different ANHNA (Auto-MPG dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	2.263×10^{-4}	1.132×10^{-4}	2.801×10^{-8}	6.994×10^{-4}	2.464×10^{-4}
ANHNA-PSO$_{gRe}$	2.401×10^{-4}	1.796×10^{-4}	9.394×10^{-6}	9.809×10^{-4}	2.883×10^{-4}
ANHNA-PSO$_{lRe}$	1.732×10^{-4}	1.350×10^{-4}	8.126×10^{-7}	4.459×10^{-4}	1.600×10^{-4}

Source: author.

From the exposed in Table 17, the ELM network has the solutions with the highest norm of all examples. This result combined with the ones in the previous chapter

illustrates how much the random choice of hidden weights and biases influences the estimation of the output weights. As already expected, ELM/BIP provided the smallest norm of all methods. When we compare the median and minimum values ANHNA-DE$_{Re}$, ANHNA-PSO$_{gRe}$ and ANHNA-PSO$_{lRe}$, it is possible to observe that their norms were very similar. Their maximum values are what distinguishes their performances and consequently moves their average value.

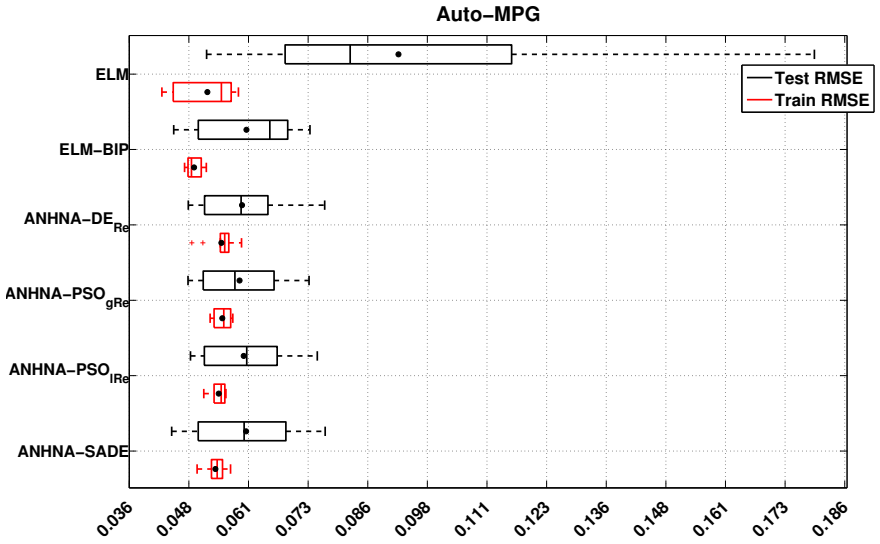
Considering the values of regularization parameter λ in Table 18, the values presented are small but not zero (with exception of the minimum value of ANHNA-DE$_{Re}$). This means that regularization was in fact needed to improve the performances, otherwise, the evolved λ would always be close to zero, i.e. no regularization needed.

Figure 36 shows the RMSE of training and testing, where the black dots represent their respective mean values. It is clear that ELM achieves the worst performance overall, but the remaining methods have very similar results. ANHNA-SADE provides the smallest testing RMSE combined with a small number of hidden neurons (about 30), while ANHNA-PSO$_{gRe}$ provides the best results taking the median and mean as reference.

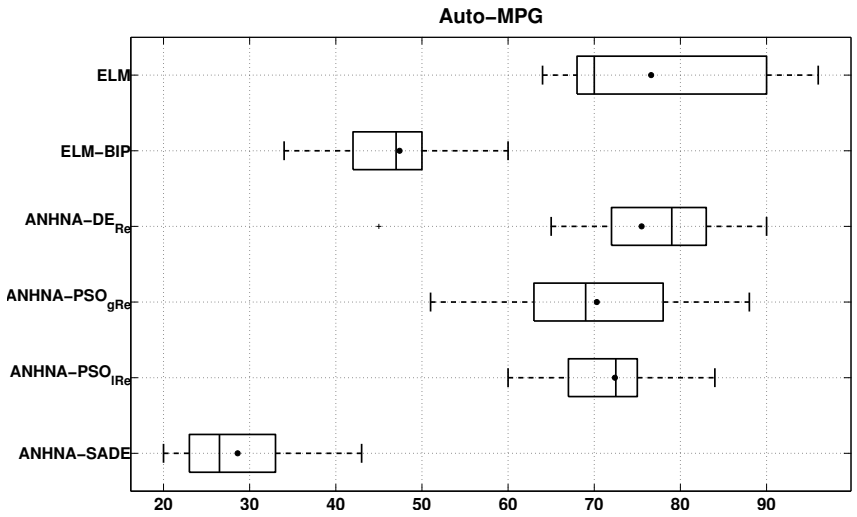
Overall, considering also the time of execution to select a network, train and test it, ELM/BIP offered the best solution. It provided a competitive RMSE performance, combined with the smallest norm of all, used fewer neurons than ANHNA-DE$_{Re}$, ANHNA-PSO$_{gRe}$, and ANHNA-PSO$_{lRe}$ and required only minutes to run, compared to hours of ANHNA versions.

Figure 36 – ANHNA comparison with different metaheuristics (Auto-MPG dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



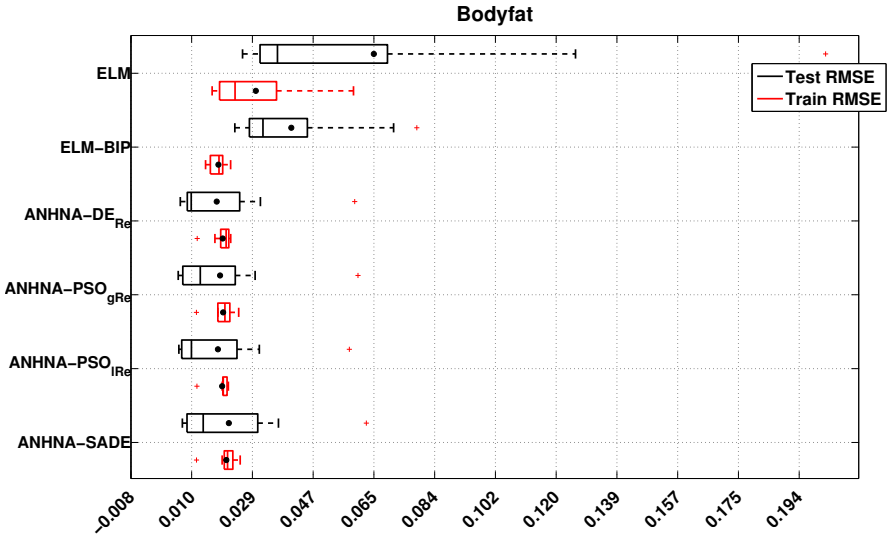
Source: author.

7.1.2 Bodyfat dataset

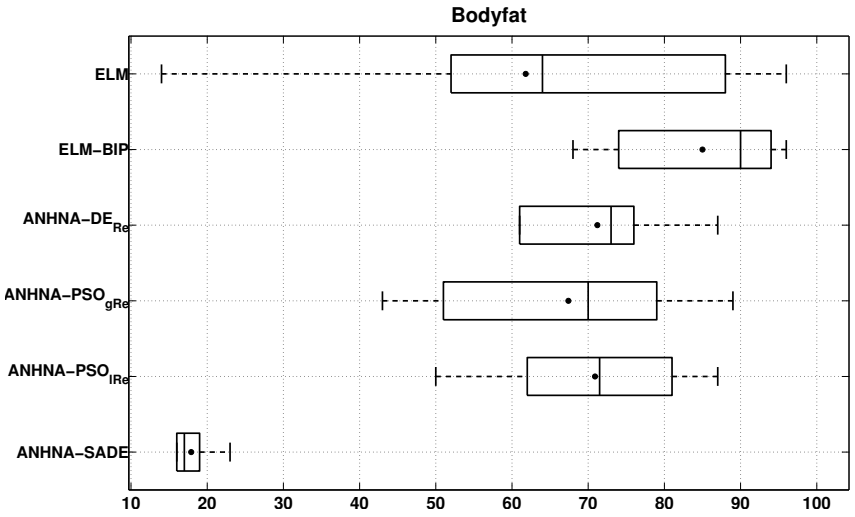
For this dataset, as shown in Figure 37, all ANHNA versions reported better performances than ELM and ELM/BIP. On average, those versions are very similar although their medians demonstrate slight differences, where ANHNA-PSO$_{lRe}$ and ANHNA-DE$_{Re}$ stand out. ANHNA-SADE, once more, offered the minimum number of hidden neurons but with higher norms than ANHNA-DE$_{Re}$, ANHNA-PSO$_{gRe}$, ANHNA-PSO$_{lRe}$.

Figure 37 – ANHNA comparison with different metaheuristics (Bodyfat dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 19 – Evaluation results of the norm of the output weight vector (Bodyfat dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	5.505×10^{2}	5.104×10^{2}	13.36	1.051×10^{3}	3.593×10^{2}
ELM-BIP	1.038	1.020	7.538×10^{-1}	1.249	1.425×10^{-1}
ANHNA-DE$_{Re}$	5.225	4.047	2.551	14.82	3.674
ANHNA-PSO$_{gRe}$	4.713	3.065	2.392	15.46	3.949
ANHNA-PSO$_{lRe}$	3.787	3.271	1.705	7.262	1.891
ANHNA-SADE	45.70	40.22	16.05	80.37	23.06

Source: author.

Table 20 – Regularization parameters for different ANHNA (Bodyfat dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	9.045×10^{-4}	5.731×10^{-4}	3.865×10^{-5}	3.613×10^{-3}	1.059×10^{-3}
ANHNA-PSO$_{gRe}$	8.133×10^{-4}	4.376×10^{-4}	3.096×10^{-3}	4.221×10^{-3}	1.895×10^{-3}
ANHNA-PSO$_{lRe}$	1.464×10^{-3}	1.184×10^{-3}	1.668×10^{-4}	3.927×10^{-3}	1.262×10^{-3}

Source: author.

Even though ELM/BIP still provides the smallest norms of output weight vectors, ANHNA-SADE provides the best solution for this example. It achieved a performance similar to the best ones with a significantly smaller number of hidden neurons and a norm of acceptable magnitude.

7.1.3 Breast Cancer dataset

Table 21 – Evaluation results of the norm of the output weight vector (Breast Cancer dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	23.16	21.72	10.60	40.06	10.24
ELM-BIP	6.433×10^{-1}	6.251×10^{-1}	5.411×10^{-1}	7.950×10^{-1}	7.899×10^{-2}
ANHNA-DE$_{Re}$	5.441	4.693	3.339	9.504	2.051
ANHNA-PSO$_{gRe}$	6.921	6.646	3.661	9.651	1.768
ANHNA-PSO$_{lRe}$	7.002	6.722	4.753	9.506	1.529
ANHNA-SADE	18.59	6.211	3.518	1.052×10^{2}	31.71

Source: author.

Table 22 – Regularization parameters for different ANHNA (Breast Cancer dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	6.130×10^{-6}	2.730×10^{-6}	7.267×10^{-8}	3.842×10^{-5}	1.150×10^{-5}
ANHNA-PSO$_{gRe}$	4.327×10^{-6}	3.234×10^{-6}	1.176×10^{-8}	1.287×10^{-5}	4.326×10^{-6}
ANHNA-PSO$_{lRe}$	5.667×10^{-6}	3.578×10^{-6}	3.288×10^{-7}	1.407×10^{-5}	4.824×10^{-6}

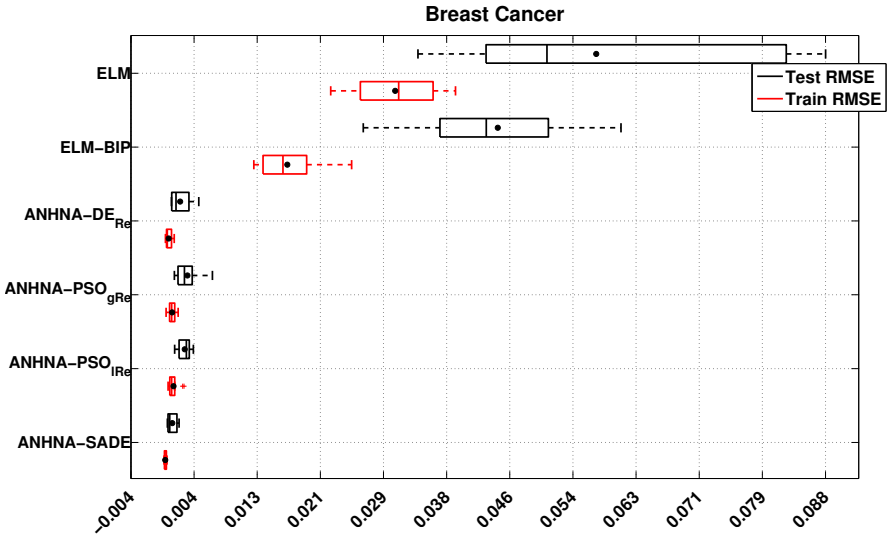
Source: author.

Observing Table 21, we can already notice a pattern, where ELM/BIP offer the smallest norms of all, ANHNA-SADE provides the largest ones and ANHNA-DE$_{Re}$, ANHNA-PSO$_{gRe}$ and ANHNA-PSO$_{lRe}$ are very similar to each other.

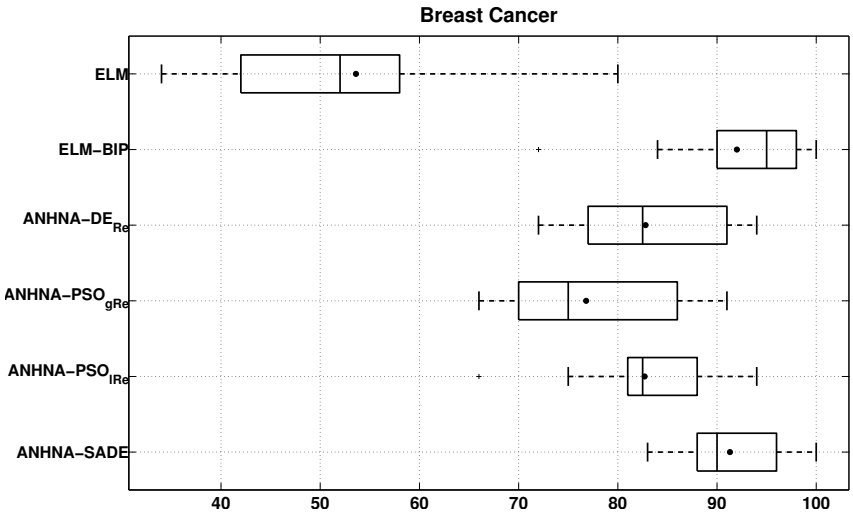
Figure 38 shows that ANHNA-based methods provide solutions with a noticeable improvement on the testing performance. Although the four versions suggest a larger

Figure 38 – ANHNA comparison with different metaheuristics (Breast Cancer dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

number of hidden neurons, the resulting norms of the output weight vectors are not as high as ELM ones. Considering that ANHNA versions' RMSE performances are similar, ANHNA-PSO$_{gRe}$ provided the best solution with fewer neurons and a small norm.

Table 22 shows that the chosen λ are closer to zero than the ones obtained with the previously presented datasets.

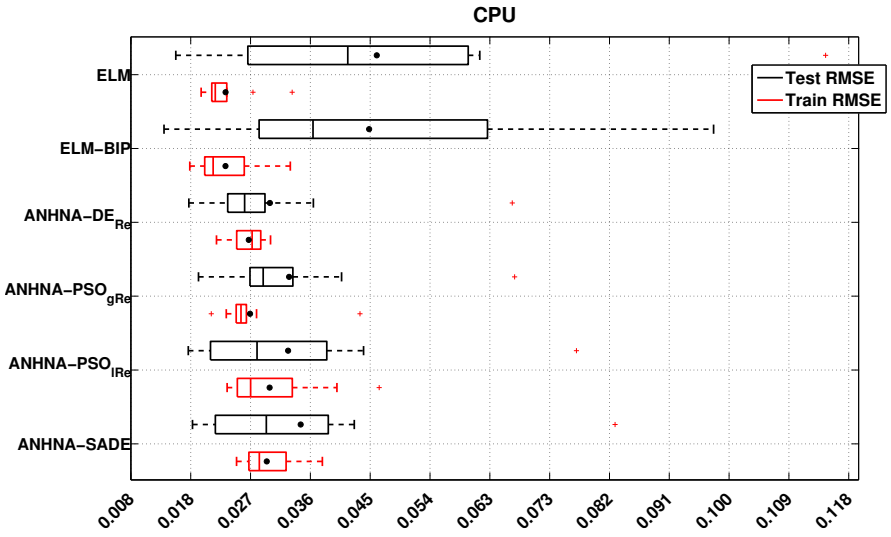
7.1.4 CPU dataset

Figure 39 shows that, once more, ANHNA versions provided better performances, although ANHNA-DE$_{Re}$ offers the best mean and median values of all ANHNA.

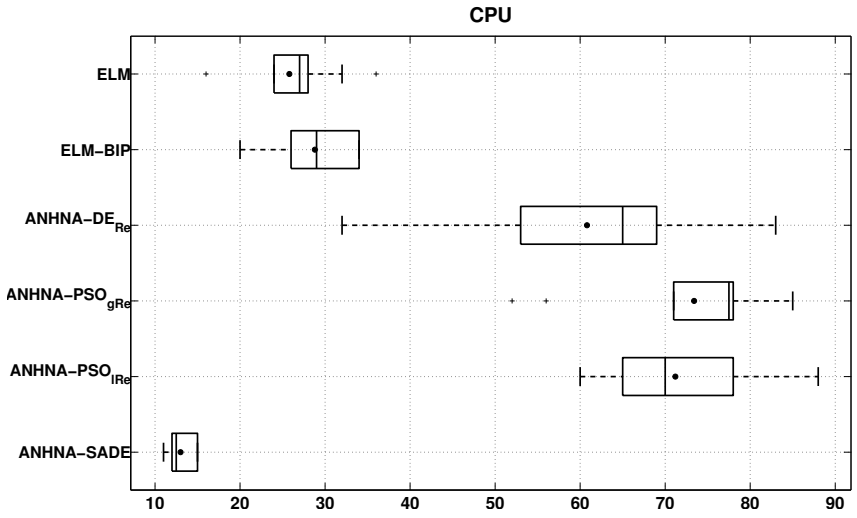
ANHNA-SADE still offers the minimum number of hidden neurons, but also the largest norms of output weight vectors of all ANHNA versions (see Table 23). Another point to highlight is that, while ELM and ELM/BIP show signs of overfitting (small training errors and large test errors), we can observe that all ANHNA versions provides compatible training and testing errors. Such result shows the reliability of our proposed method. That being said, ANHNA-DE$_{Re}$ provided the best solution with the best RMSE performance, the second minimum number of hidden neurons among ANHNA versions and a small norm.

Figure 39 – ANHNA comparison with different metaheuristics (CPU dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 23 – Evaluation results of the norm of the output weight vector (CPU dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	1.665×10^{3}	1.438×10^{3}	7.715×10^{2}	3.295×10^{3}	7.652×10^{2}
ELM-BIP	1.437	1.380	6.323×10^{-1}	2.093	4.048×10^{-1}
ANHNA-DE$_{Re}$	18.21	13.24	8.083	54.47	13.73
ANHNA-PSO$_{gRe}$	15.69	12.60	8.917×10^{-1}	54.42	14.67
ANHNA-PSO$_{lRe}$	8.574	8.875	6.874×10^{-1}	14.03	4.629
ANHNA-SADE	4.492×10^{2}	3.135×10^{2}	1.447×10^{2}	1.442×10^{3}	4.102×10^{2}

Source: author.

Table 24 – Regularization parameters for different ANHNA (CPU dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	1.186×10^{-4}	7.556×10^{-5}	2.101×10^{-6}	4.346×10^{-4}	1.326×10^{-4}
ANHNA-PSO$_{gRe}$	1.162×10^{-2}	1.587×10^{-4}	5.507×10^{-6}	1.150×10^{-1}	3.632×10^{-2}
ANHNA-PSO$_{lRe}$	4.443×10^{-2}	2.456×10^{-4}	9.392×10^{-5}	2.667×10^{-1}	9.566×10^{-2}

Source: author.

7.1.5 iCub dataset

Analyzing Figure 40, it is clear that there is no significative difference in performance between ELM/BIP and ANHNA versions. What is clear is that all of the former are better than the traditional ELM.

Considering also the number of hidden neurons, ANHNA-SADE still offers the minimum number, but ELM/BIP also provides a small amount. To finalize, observing the norms of the output weight matrices in Table 25, it is clear that ELM/BIP provided the best solution, given the fact of the smallest norm plus the fact that it takes minutes to execute this evaluation.

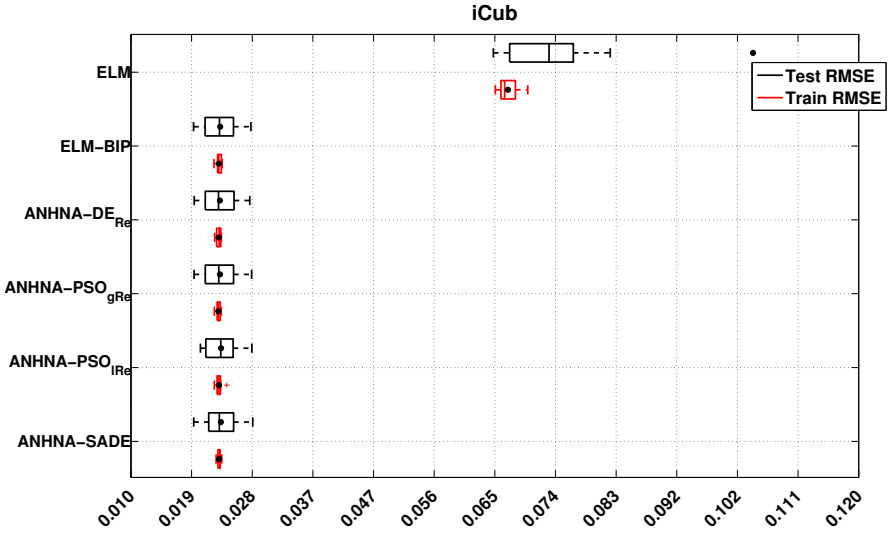
Table 25 – Evaluation results of the norm of the output weight matrix (iCub dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	4.488×10^{7}	3.455×10^{7}	4.594×10^{5}	1.095×10^{8}	4.495×10^{7}
ELM-BIP	1.954	1.837	5.483×10^{-1}	4.052	1.127
ANHNA-DE$_{Re}$	6.486	5.255	9.970×10^{-1}	16.48	4.779
ANHNA-PSO$_{gRe}$	3.379	3.128	2.478	4.754	6.804×10^{-1}
ANHNA-PSO$_{lRe}$	4.920	2.805	1.155	21.07	5.850
ANHNA-SADE	1.332×10^{2}	83.71	34.66	5.553×10^{2}	1.550×10^{2}

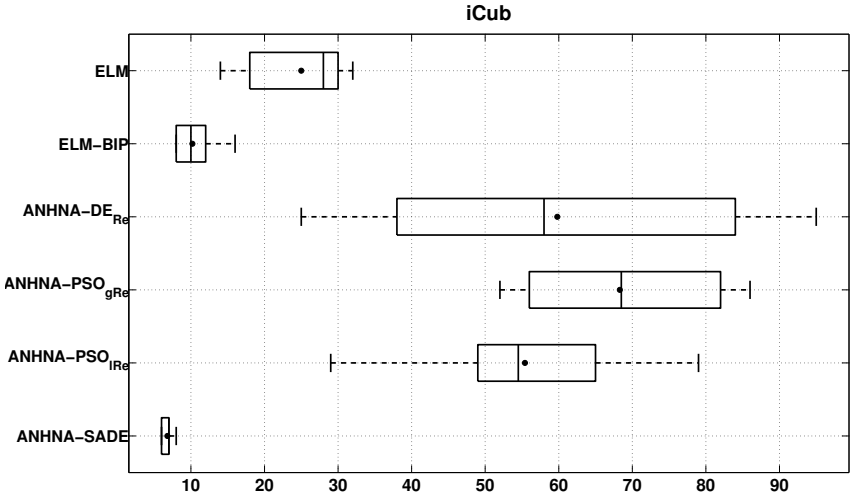
Source: author.

Figure 40 – ANHNA comparison with different metaheuristics (iCub dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 26 – Regularization parameters for different ANHNA (iCub dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	2.334×10^{-3}	2.104×10^{-4}	2.810×10^{-5}	1.523×10^{-2}	4.789×10^{-3}
ANHNA-PSO$_{gRe}$	9.698×10^{-4}	1.087×10^{-3}	1.685×10^{-4}	1.330×10^{-3}	3.710×10^{-4}
ANHNA-PSO$_{lRe}$	1.546×10^{-3}	8.891×10^{-4}	-3.509×10^{-4}	4.873×10^{-3}	1.747×10^{-3}

Source: author.

7.1.6 Servo dataset

It is interesting to notice in Table 28 that the regularization parameters are very small tending to zero. This behavior is clearly reflected in the resulting large norms of output weight vectors of ANHNA versions (see Table 27).

Although ANHNA-DE$_{Re}$ showed a better performance, its respective norms prevents it to be chosen as the best solution. In this case, ELM/BIP also provides the best cost-benefit ratio.

In the following chapter, we present the conclusions of this work and future works.

Table 27 – Evaluation results of the norm of the output weight vector (Servo dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	2.154×10^{4}	3.522×10^{3}	1.286×10^{2}	1.578×10^{5}	4.875×10^{4}
ELM-BIP	12.95	10.74	5.546	32.43	8.349
ANHNA-DE$_{Re}$	2.597×10^{3}	1.068×10^{3}	61.36	1.519×10^{4}	4.502×10^{3}
ANHNA-PSO$_{gRe}$	7.797×10^{3}	2.150×10^{3}	2.184×10^{2}	5.047×10^{4}	1.536×10^{4}
ANHNA-PSO$_{lRe}$	3.286×10^{3}	3.571×10^{3}	1.502×10^{2}	9.226×10^{3}	2.933×10^{3}
ANHNA-SADE	9.863×10^{6}	1.325×10^{6}	3.553×10^{4}	8.295×10^{7}	2.577×10^{7}

Source: author.

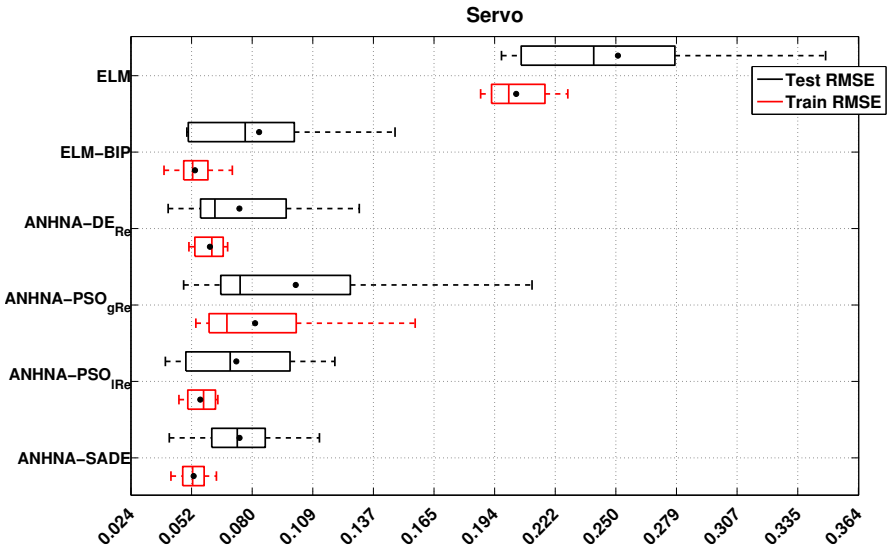
Table 28 – Regularization parameters for different ANHNA (Servo dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ANHNA-DE$_{Re}$	2.083×10^{-6}	4.416×10^{-8}	7.657×10^{-11}	2.023×10^{-5}	6.377×10^{-6}
ANHNA-PSO$_{gRe}$	3.528×10^{-7}	4.506×10^{-9}	-7.557×10^{-7}	1.843×10^{-6}	8.234×10^{-7}
ANHNA-PSO$_{lRe}$	2.831×10^{-7}	2.545×10^{-9}	3.169×10^{-10}	1.921×10^{-6}	6.038×10^{-7}

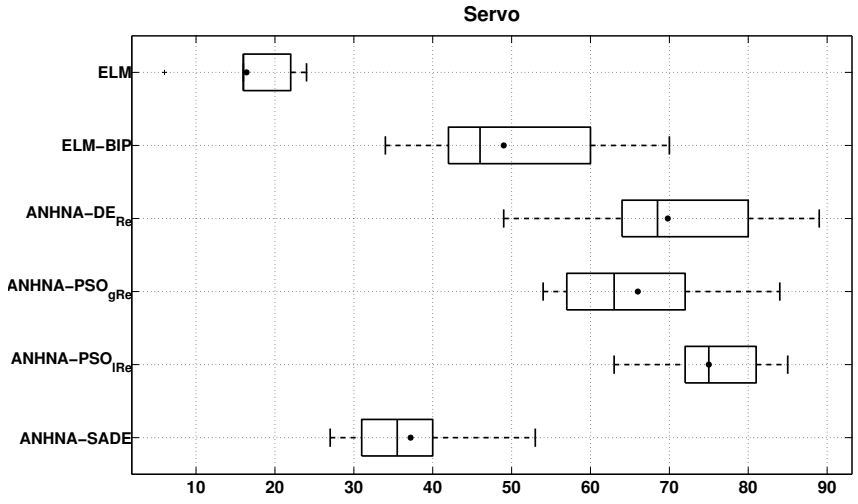
Source: author.

Figure 41 – ANHNA comparison with different metaheuristics (Servo dataset).

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

8 CONCLUSIONS AND FUTURE WORK

This thesis deals with the challenge of improving the Extreme Learning Machine network by designing an efficient method for model selection and proposing an outlier robust version. Considering that traditional ELM networks have their performances strongly influenced by the hidden layer size, the conditioning of the hidden layer's output and the features provided by the hidden layer, any improvement should address those issues.

That being said, one of our contributions is the R-ELM/BIP that combines the outlier robustness property provided by M-Estimators with the optimization of the hidden layer's output using Intrinsic Plasticity principles. The second contribution towards an improved ELM network is the Automatic Number of Hidden Neurons Approach (ANHNA), which is a new encoding scheme for automatic architecture design based on metaheuristics that suggests the number of hidden neurons and their respective activation function's parameters. Deriving from ANHNA, a third contribution is a robust version that also searches for the weight function and its respective error threshold, besides the architecture itself.

An additional contribution of this work is the data and the work related to the learning of visuomotor coordination in a pointing problem with the humanoid robot iCub. This contribution provided one of the datasets used here and it was harvested with the simulated version of the robot.

In this thesis, we made a brief review in the literature about the different neural networks with random projections in the hidden layer and how, within this context, the ELM arose as an appealing alternative. We also discussed the issues that affect its performance and introduced the concept of Intrinsic Plasticity, which is based on a biologically plausible phenomenon and optimizes the information transmission of the hidden layer. This method, implemented as Batch Intrinsic Plasticity, also acts as a feature regularizer and provides a better conditioned hidden layer's output matrix $\mathbf{H}$, that affect (positively) the quality of the output weights solution.

Another aspect of ELM's performance discussed here is related to its robustness to outliers. It is known that real-world datasets are commonly contaminated with outliers and/or leverage points. So, it is only realistic that we also tried to incorporate such property into an improved version. Based on Horata *et al.* (2013), we described the two main problems that must be addressed in order to achieve outlier robustness: the numerical

stability and the robust estimation of output weights.

We also presented a brief review about optimization of ELM network using metaheuristics, where we showed that most of the works seek to optimize the hidden weights. This specific category of works, by adapting the hidden weights, ceases to deal with an ELM network to deal with an evolutionary neural network that adopts a metaheuristic to perform the weight adaptation. Based on the work of Das *et al.* (2009), our proposed encoding scheme, ANHNA, which can be applied to any populational based metaheuristic, is further described. Four different metaheuristics algorithms, that were adopted in this thesis, are described as well.

We carried out a comprehensive set of tests to evaluate our proposed methods with traditional ELM and its improved version using BIP, for commonly used datasets, and also with robust methods such as RELM and IRWLS-ELM, for purposely increasingly contaminated datasets. For the robust cases, with exception of robust ANHNA versions, all networks were selected through a grid search associated with a 5-Fold cross-validation that used a 10% trimmed validation RMSE to choose the best network in each of the 10 independent repetitions. Each of the methods was subjected to training and validation using data deliberately contaminated with outliers with different rates (10% to 40%) and different cases (1 or 2 sided).

The performed comparison involved the evaluation of three different information: the test RMSE, the number of hidden neurons and the norm of the output weight vectors. The respective results showed that the non-robust ELM versions did suffer performance deterioration with the growing rate of outliers, which reinforces the importance of the study on outlier robustness. In addition, there was not a single method that was the best in all datasets and cases, but there are some patterns observed. One of them is related to BIP property of maintaining a small norm of the output weight vector, independently of the rate of outlier contamination. ELM/BIP provided better results than ELM network, especially in two-sided cases, and showed that small norms of the output weights influence the network robustness. Such results indicate that BIP is a very reliable method even in adverse situations.

The proposed R-ELM/BIP performed similarly to RELM in most of the cases, but better than the non-robust methods and the IRWLS-ELM. It presented, just as RELM and robust ANHNA versions, to be less influenced by the rate of outliers. Even though the

usual number of hidden neurons is larger than the RELM's number in half of the examples, the provided norms of R-ELM/BIP are the smallest of all robust methods and also than the traditional ELM, independently of the contamination rate or size of the hidden layer. Hence, it represents a valid and reliable alternative to the other methods.

The two robust ANHNA-DE evaluated, ANHNA-DE$_{R}$ and ANHNA-DE$_{Rm}$, differentiate from each other by the use of trimmed validation RMSE or median validation RMSE to choose the best solution along the generations, respectively. They presented accuracies that are comparable to, or even better than, the other robust methods. However, the number of hidden neurons was similar or higher than the others and their norms of output weight vectors were extremely high. This indicates that using only RMSE as the fitness function for ELM's model selection is not a sufficient measure and shows that you can achieve great accuracy with the ELM at the expense of a non-robust network that may also present generalization problems as well.

For the evaluation of the proposed method ANHNA, where there is no other noise added but the one inherent to the dataset itself, comprehensive tests were performed with ANHNA-DE, ANHNA-SADE, ANHNA-PSO$_{g}$, and ANHNA-PSO$_{l}$. For each of them, eight different fitness functions were adopted and comprised the use of RMSE, two different percentages of the conditioning number and the norm of the output weight vector, besides the same variants but using regularized ELM. The only exception was with ANHNA-SADE that had only the former four fitness functions evaluated. From all of those tests, we chose only one for each metaheuristic to be represented in the results chapter.

From the comparisons of the variants, the difficulty of the trade-off between the RMSE performance, the number of hidden neurons and norm of the output weight vector became clear. Those variants that provided best RMSE performance rarely offered small hidden layers, much less a small norm of the output weight vector, and vice versa. With the given options, we chose the regularized versions ANHNA-DE$_{Re}$, ANHNA-PSO$_{gRe}$ and ANHNA-PSO$_{lRe}$, who presented the good RMSE performances and smaller norms than the variants with highest RMSE accuracy. For ANHNA-SADE, we chose its original version, since there were no results available with regularization.

When those chosen ANHNA variants were compared with ELM, ELM/BIP and among themselves, we observed that its principle of modifying the activation function parameters as well as the number of hidden neurons has provided better RMSE performances

than traditional ELM and similar or better performances than ELM/BIP. It also was clear that the use of DE or PSO is indifferent to RMSE performance, but SADE gave similar RMSE performances with an unusually small number of hidden neurons. Nevertheless, this last method also resulted in solutions with the highest norm of all selected algorithms.

Even though, in some cases, the cost-benefit ratio (considering the norm and execution time) still leans towards the ELM/BIP, ANHNA proposal still presented examples with improved accuracy associated with norms equivalent to the ELM/BIP ones. Such results indicate that this method should be further refined, but already shows promising results.

So far, this thesis presented three main contributions: the R-ELM/BIP, ANHNA versions and the iCub data and study. We focused on describing the former two formulations and their contexts, associating them with a comprehensive amount of tests which included the iCub's harvest data. From what it was exposed, our proposals showed valid and promising results. In what follows, we discuss the main directions that we wish to pursue from now on.

8.1 Future Work

The fitness function for ANHNA will be further studied and evaluated in order to find the best balance between the previously mentioned trade-off. Time optimization of ANHNA's execution is another matter that must be addressed. We will also execute the evaluations of ANHNA-SADE with regularized ELM networks and perform additional studies to understand which mechanisms enabled it to provide a fewer number of hidden neurons than the other tested metaheuristics.

For robust versions of ANHNA, new robust measures to evaluate the validation errors, when the validation set is also contaminated, should be tested. The investigation of robustness properties will be performed using noise of distributions other than the Normal one plus the adoption of different robust estimation methods that could reduce the execution time without affecting its robustness.

BIBLIOGRAPHY

ABBASS, H. A. An evolutionary artificial neural networks approach for breast cancer diagnosis. **Artificial Intelligence in Medicine**, v. 25, n. 3, p. 265–281, 2002. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0933365702000283>.

ABBASS, H. A.; SARKER, R.; NEWTON, C. A pareto differential evolution approach to vector optimization problems. In: **IEEE Congress on Evolutionary Computation**. Seoul, Korea: IEEE Publishing, 2001. v. 2, p. 971–978.

ALADAG, C. H.; EGRIOGLU, E.; YOLCU, U. Robust multilayer neural network based on median neuron model. **Neural Computing and Applications**, v. 24, n. 3-4, p. 945–956, 2014.

ALENCAR, A. S. C.; Rocha Neto, A. R. Uma abordagem de poda para máquinas de aprendizado extremo via algoritmos genéticos. In: **Anais do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2014)**. [s.n.], 2014. Available from Internet: <http://www.lbd.dcc.ufmg.br/colecoes/eniac/2014/0060.pdf>.

ANDREWS, D. F. A robust method for multiple linear regression. **Technometrics**, Taylor & Francis, Ltd. on behalf of American Statistical Association and American Society for Quality, v. 16, n. 4, p. 523–531, 1974. ISSN 00401706. Available from Internet: <http://www.jstor.org/stable/1267603>.

ANGELINE, P.; SAUNDERS, G.; POLLACK, J. An evolutionary algorithm that constructs recurrent neural networks. **IEEE Transactions on Neural Networks**, v. 5, n. 1, p. 54–65, January 1994.

ASHBY, W. R. **Design for a brain**. [S.l.]: New York: Wiley, 1952.

BADDELEY, R.; ABBOTT, L. F.; BOOTH, M. C.; SENGPIEL, F.; FREEMAN, T.; WAKEMAN, E. A.; ROLLS, E. T. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. **Proceedings of the Royal Society B: Biological Sciences**, v. 264, p. 1775–1783, 1997.

BAKER, J. E. Reducing bias and inefficiency in the selection algorithm. In: GREFENSTETTE, J. J. (Ed.). **Proceedings of the 2nd International Conference on Genetic Algorithms**. [S.l.]: Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1987. p. 14–21. ISBN 0-8058-0158-8.

BARROS, A. L. B.; BARRETO, G. A. Building a robust extreme learning machine for classification in the presence of outliers. In: PAN, J.-S.; POLYCARPOU, M.; WOźNIAK, M.; CARVALHO, A. C.; QUINTIáN, H.; CORCHADO, E. (Ed.). **Hybrid Artificial Intelligent Systems**. [S.l.]: Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8073). p. 588–597.

BARROS, A. L. B. D. P. **Revisitando O Problema de Classificação de Padrões na Presença de Outliers Usando Técnicas de Regressão Robusta**. Tese (Doutorado) — Universidade Federal do Ceará, 2013.

BARTLETT, P. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. **IEEE Transactions on Information Theory**, v. 44, n. 5, p. 525–536, March 1998.

BATES, E.; CAMAIONI, L.; VOLTERRA, V. The acquisition of performatives prior to speech. **Merrill-Palmer Quarterly**, v. 21, n. 3, p. 205–224, 1975.

BELIAKOV, G.; KELAREV, A.; YEARWOOD, J. Robust artificial neural networks and outlier detection. technical report. **CoRR**, 2011. Available from Internet: <http://dblp.uni-trier.de/db/journals/corr/corr1110.html#abs-1110-0169>.

BELL, A. J.; SEJNOWSKI, T. J. An information-maximization approach to blind separation and blind deconvolution. **Neural Computation**, v. 7, n. 6, p. 1129–1159, November 1995.

BEN-GAL, I. Outlier detection. In: MAIMON, O.; ROKACH, L. (Ed.). **Data Mining and Knowledge Discovery Handbook**. [S.l.]: Springer US, 2005. p. 131–146.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: **Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)**. [S.l.: s.n.], 2007. p. 120–127.

BROOMHEAD, D. S.; LOWE, D. Multivariable functional interpolation and adaptive networks. **Complex Systems**, v. 2, n. 3, p. 321–355, 1988.

CANTÚ-PAZ, E.; KAMATH, C. Evolving neural networks to identify bent-double galaxies in the first survey. **Neural Networks**, v. 16, p. 507–517, 2003.

CAO, J.; LIN, Z.; HUANG, G.-B. Self-adaptive evolutionary extreme learning machine. **Neural Processing Letters**, v. 36, n. 3, p. 285–305, 2012. Available from Internet: <http://link.springer.com/article/10.1007%2Fs11063-012-9236-y#>.

CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 1, p. 58–73, February 2002.

CUDMORE, R. H.; DESAI, N. S. Intrinsic plasticity. **Scholarpedia**, v. 3, n. 2, p. 1363, 2008. revision 89024.

DAS, S.; ABRAHAM, A.; KONAR, A. **Metaheuristic Clustering**. [S.l.]: Springer, 2009. (Studies in Computational Intelligence, Vol. 178). ISBN 978-3-540-93964-1.

DENG, W.; ZHENG, Q.; CHEN, L. Regularized extreme learning machine. In: **CIDM**. [S.l.]: IEEE, 2009. p. 389–395.

DICTIONARIES, O. **"Engram"**. 2014. Oxford University Press.

DING, S.; LI, H.; SU, C.; YU, J.; JIN, F. Evolutionary artificial neural networks: a review. **Artificial Intelligence Review**, v. 39, n. 3, p. 251–260, 2013.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: **Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95.** [S.l.: s.n.], 1995. p. 39–43.

EMMERICH, F. R. R. C.; STEIL, J. J. Recurrence enhances the spatial encoding of static inputs in reservoir networks. In: **Proceedings of International Conference on Artificial Neural Networks**. [S.l.: s.n.], 2010. p. 148–153.

ENGELBRECHT, A. **Computational Interlligence: an introduction**. 2nd. ed. [S.l.]: John Wiley & Sons, 2007.

EVGENIOUA, T.; POGGIO, T.; PONTIL, M.; VERRI, A. Regularization and statistical learning theory for data analysis. **Computational Statistics & Data Analysis**, v. 38, p. 421–432, 2002. Available from Internet: <http://www0.cs.ucl.ac.uk/staff/M.Pontil/reading/EPPV.pdf>.

FAN, Y.-t.; WU, W.; YANG, W.-y.; FAN, Q.-w.; WANG, J. A pruning algorithm with L1/2 regularizer for extreme learning machine. **Journal of Zhejiang University SCIENCE C**, v. 15, n. 2, p. 119–125, 2014.

FENG, G.; HUANG, G.-B.; LIN, Q.; GAY, R. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. **IEEE Transactions on Neural Networks**, v. 20, n. 8, p. 1352–1357, August 2009.

FENG, Y.; LI, R.; SUDJIANTO, A.; ZHANG, Y. Robust neural network with applications to credit portfolio data analysis. **Stat Interface**, v. 3, n. 4, p. 437–444, 2010.

FIGUEIREDO, E. M.; LUDERMIR, T. B. Investigating the use of alternative topologies on performance of the PSO-ELM. **Neurocomputing**, v. 127, n. 0, p. 4–12, 2014. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0925231213007807>.

FOX, J. An R and S-plus companion to applied regression. In: _____. SAGE Publications, Inc, 2002. cap. Appendix. Available from Internet: <http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-robust-regression.pdf>.

FREIRE, A.; BARRETO, G. A new model selection approach for the elm network using metaheuristic optimization. In: **Proceedings of 22th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014)**. Bruges (Belgium): [s.n.], 2014. p. 619–624. Available from Internet: <http://www.i6doc.com/fr/livre/?GCOI=28001100432440>.

FREIRE, A.; BARRETO, G. A robust and regularized extreme learning machine. In: **Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2014**. São Carlos (Brazil): [s.n.], 2014.

FREIRE, A.; BARRETO, G. Uma nova abordagem de seleção de modelos para a máquina de aprendizado extremo. In: **XX Congresso Brasileiro de Automática (CBA 2014)**. Minas Gerais (Brazil): [s.n.], 2014.

FREIRE, A.; LEMME, A.; BARRETO, G.; STEIL, J. Aprendizado de coordenação visomotora no comportamento de apontar em um espaço 3d. In: **Anais do XIX Congresso Brasileiro de Automática (CBA2012)**. [s.n.], 2012. Available from Internet: <http://cba2012.dee.ufcg.edu.br/anais>.

FREIRE, A.; LEMME, A.; BARRETO, G.; STEIL, J. Learning visuo-motor coordination for pointing without depth calculation. In: **Proceedings of the European Symposium on Artificial Neural Networks**. [S.l.: s.n.], 2012. p. 91–96.

HADAMARD, J. Sur les problémes aux dérivées partielles et leur signification physique. **Bulletin, Princeton University**, v. 13, p. 49–52, 1902.

HAGIWARA, K.; FUKUMIZU, K. Relation between weight size and degree of over-fitting in neural network regression. **Neural Networks**, v. 21, n. 1, p. 48–58, 2008.

HAN, F.; YAO, H.-F.; LING, Q.-H. An improved evolutionary extreme learning machine based on particle swarm optimization. **Neurocomputing**, v. 116, p. 87–93, 2013.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. [S.l.]: Prentice Hall, 2008. ISBN 0131471392.

HORATA, P.; CHIEWCHANWATTANA, S.; SUNAT, K. Robust extreme learning machine. **Neurocomputing**, v. 102, p. 31–34, 2013. Advances in Extreme Learning Machines (ELM 2011).

HU, Z.; XIONG, S.; SU, Q.; ZHANG, X. Sufficient conditions for global convergence of differential evolution algorithm. **Journal Applied Mathematics**, v. 2013, 2013. Article ID 193196.

HUANG, G.-B. Reply to comments on "the extreme learning machine". **IEEE Transactions On Neural Networks**, v. 19, n. 8, p. 1494–1495, August 2008.

HUANG, G.-B. An insight into extreme learning machines: Random neurons, random features and kernels. **Cognitive Computation**, v. 6, p. 376–390, 2014.

HUANG, G.-B.; CHEN, L. Enhanced random search based incremental extreme learning machine. **Neurocomputing**, v. 71, n. 16–18, p. 3460–3468, 2008. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).

HUANG, G.-B.; CHEN, L.; SIEW, C.-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. **IEEE Transactions on Neural Networks**, v. 17, n. 4, p. 879–892, July 2006.

HUANG, G.-B.; LI, M.-B.; CHEN, L.; SIEW, C.-K. Incremental extreme learning machine with fully complex hidden nodes. **Neurocomputing**, v. 71, n. 4-6, p. 576–583, 2008.

HUANG, G.-B.; WANG, D.; LAN, Y. Extreme learning machines: a survey. **International Journal of Machine Learning and Cybernetics**, v. 2, n. 2, p. 107–122, 2011.

HUBER, P. J. Robust estimation of a location parameter. **The Annals of Mathematical Statistics**, v. 35, n. 1, p. 73–101, 1964. Available from Internet: <http://projecteuclid.org/euclid.aoms/1177703732>.

HUYNH, H. T.; WON, Y.; KIM, J.-J. An improvement of extreme learning machine for compact single-hidden-layer feedforward neural networks. **International Journal of Neural Systems**, v. 18, n. 5, p. 433–441, 2008.

IGELNIK, B.; PAO, Y.-H. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. **IEEE Transactions on Neural Networks**, v. 6, n. 6, p. 1320–1329, November 1995.

JAEGER, H. **The "echo state" approach to analysing and training recurrent neural networks**. 2001. GMD Report 148, GMD - German National Research Institute for Computer Science.

JAEGER, H. **The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note**. 2001. GMD Report 148, GMD - German National Research Institute for Computer Science. Available from Internet: <http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf>.

KHAMIS, A.; ISMAIL, Z.; HARON, K.; MOHAMMED, A. T. The effects of outliers data on neural network performance. **Journal of Applied Sciences**, v. 5, n. 8, p. 1394–1398, 2005.

KITANO, H. Designing neural networks using genetic algorithms with graph generation system. **Complex Systems**, v. 4, n. 4, p. 461–476, 1990. Available from Internet: <http://www.complex-systems.com/abstracts/v04_i04_a06.html>.

KULAIF, A. C. P.; ZUBEN, F. J. V. Improved regularization in extreme learning machines. In: **Annals of Congresso Brasileiro de Inteligência Computacional (CBIC)**. [S.l.: s.n.], 2013.

LARSEN, J.; NONBOE, L.; HINTZ-MADSEN, M.; HANSEN, L. Design of robust neural network classifiers. In: **Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on**. [S.l.: s.n.], 1998. v. 2, p. 1205–1208 vol.2. ISSN 1520-6149.

LEE, C.-C.; TSAI, C.-L.; CHIANG, Y.-C.; SHIH, C.-Y. Noisy time series prediction using m-estimator based robust radial basis function neural networks with growing and pruning techniques. **Expert Systems and Applications**, v. 36, n. 3, p. 4717–4724, 2009.

LEMME, A.; FREIRE, A.; BARRETO, G.; STEIL, J. Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub. **Neurocomputing**, v. 112, p. 179–188, 2013.

LENT, R. **Neurociência - Da Mente e do Comportamento**. [S.l.]: Guanabara Koogan, 2008. ISBN 9788527713795.

LEUNG, F. H. F.; LAM, H.; LING, S.; TAM, P.-S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. **IEEE Transactions on Neural Networks**, v. 14, n. 1, p. 79–88, January 2003.

LI, C. A model of neuronal intrinsic plasticity. **IEEE Transactions on Autonomous Mental Development**, v. 3, n. 4, p. 277–284, December 2011.

LIM, M.-H. Comments on the "No-Prop" algorithm. **Neural Networks**, v. 48, p. 59–60, 2013.

LISZKOWSKI, U.; CARPENTER, M.; STRIANO, T.; TOMASELLO, M. 12- and 18-month-olds point to provide information for others. **J. of Cognition and Development**, v. 7, n. 2, p. 173–187, 2006.

LIU, Y. Robust parameter estimation and model selection for neural network regression. In: COWAN, J. D.; TESAURO, G.; ALSPECTOR, J. (Ed.). **Advances in Neural Information Processing Systems (NIPS)**. [S.l.]: Morgan Kaufmann, 1993. p. 192–199.

Łobos, T.; KOSTYŁA, P.; WACŁAWEK, Z.; CICHOCKI, A. Adaptive neural networks for robust estimation of signal parameters. **The International Journal for Computation and Mathematics in Electrical and Electronic Engineering**, v. 19, n. 3, p. 903–912, 2000.

LOWE, D. Adaptive radial basis function nonlinearities, and the problem of generalisation. In: **First IEEE International Conference on Artificial Neural Networks**. [S.l.: s.n.], 1989. p. 171–175.

MAASS, W.; NATSCHLÄGER, T.; MARKRAM, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. **Neural Computation**, v. 14, n. 11, p. 2531–2560, November 2002.

MANIEZZO, V. Genetic evolution of the topology and weight distribution of neural networks. **IEEE Transactions on Neural Networks**, v. 5, n. 1, p. 39–53, January 1994.

MARONNA, R.; MARTIN, D.; YOHAI, V. **Robust Statistics Theory and Methods**. England: John Wiley & Sons, 2006.

MARTÍNEZ-MARTÍNEZ, J. M.; ESCANDELL-MONTERO, P.; SORIA-OLIVAS, E.; MARTÍN-GUERRERO, J. D.; MAGDALENA-BENEDITO, R.; GÓMEZ-SANCHIS, J. Regularized extreme learning machine for regression problems. **Neurocomputing**, v. 74, n. 17, p. 3716–3721, 2011.

MATIAS, T.; ARAUJO, R.; ANTUNES, C. H.; GABRIEL, D. Genetically optimized extreme learning machine. In: **2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)**. [S.l.: s.n.], 2013. p. 1–8.

METTA, G.; NATALE, L.; NORI, F.; SANDINI, G.; VERNON, D.; FADIGA, L.; HOFSTEN, C. von; ROSANDER, K.; LOPES, M.; SANTOS-VICTOR, J.; BERNARDINO, A.; MONTESANO, L. The icub humanoid robot: an open-systems platform for research in cognitive development. **Neural Networks**, v. 23, n. 8-9, p. 1125–1134, 2010.

MEYER, M.; VLACHOS, P. **StatLib: Data, Software and News from the Statistics Community**. 1989. Available from Internet: <http://lib.stat.cmu.edu/index.php>.

MICHE, Y.; HEESWIJK, M. van; BAS, P.; SIMULA, O.; LENDASSE, A. TROP-ELM: A double-regularized {ELM} using {LARS} and tikhonov regularization. **Neurocomputing**, v. 74, n. 16, p. 2413–2421, 2011. Advances in Extreme Learning Machine: Theory and Applications Biological Inspired Systems. Computational and Ambient Intelligence Selected papers of the 10th International Work-Conference on Artificial Neural Networks (IWANN2009).

MICHE, Y.; SORJAMAA, A.; BAS, P.; SIMULA, O.; JUTTEN, C.; LENDASSE, A. OP-ELM: Optimally pruned extreme learning machine. **IEEE Transactions on Neural Networks**, v. 21, n. 1, p. 158–162, January 2010.

MICHE, Y.; SORJAMAA, A.; LENDASSE, A. OP-ELM: Theory, experiments and a toolbox. In: KŮRKOVÁ, V.; NERUDA, R.; KOUTNÍK, J. (Ed.). **Artificial Neural Networks - ICANN 2008**. [S.l.]: Springer Berlin Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5163). p. 145–154.

NEUMANN, J. V. The general and logical theory of automata. In: JEFFRESS, L. A. (Ed.). **Cerebral mechanisms in behavior**. [S.l.]: New York: Wiley, 1951. p. 1–41.

NEUMANN, J. V. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In: SHANNON, C. E.; MCCARTHY, J. (Ed.). **Automata studies**. [S.l.]: Princeton: Princeton University Press, 1956. p. 43–98.

NEUMANN, K. **Reliability of Extreme Learning Machines**. Tese (Doutorado) — Faculty of Technology, Bielefeld University, October 2013.

NEUMANN, K.; STEIL, J. Batch intrinsic plasticity for extreme learning machines. In: **Artificial Neural Networks and Machine Learning - ICANN**. [S.l.: s.n.], 2011. p. 339–346. ISBN 978-3-642-21734-0.

NEUMANN, K.; STEIL, J. Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity. **Neurocomputing**, v. 102, p. 23 – 30, February 2013. Advances in Extreme Learning Machines (ELM 2011). Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0925231212005619>.

PAO, Y.-H.; TAKEFUJI, Y. Functional-link net computing: theory, system architecture, and functionalities. **Computer**, v. 25, n. 5, p. 76–79, May 1992.

QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. Differential evolution algorithm with strategy adaptation for global numerical optimization. **IEEE Transactions on Evolutionary Computation**, v. 13, n. 2, p. 398–417, 2009.

QUARTERONI, A.; SALERI, F.; GERVASIO, P. **Scientific computing with MATLAB and Octave**. 3rd. ed. Berlin: Springer, 2006.

REINHART, F. R. **Reservoir Computing with Output Feedback**. Tese (Doutorado) — Faculty of Technology, Bielefeld University, 2011.

RONG, H.-J.; ONG, Y.-S.; TAN, A.-H.; ZHU, Z. A fast pruned-extreme learning machine for classification problem. **Neurocomputing**, v. 72, n. 1-3, p. 359–366, 2008. Machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007).

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, November 1958.

ROUSSEEUW, P. J.; LEROY, A. M. **Robust Regression and Outlier Detection**. [S.l.]: John Wiley & Sons, 1987. (Wiley Series in Probability and Statistics).

RUSIECKI, A. Robust learning algorithm based on lta estimator. **Neurocomputing**, v. 120, n. 0, p. 624–632, 2013. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0925231213004554>.

SAVIN, C.; JOSHI, P.; TRIESCH, J. Independent component analysis in spiking neurons. **PLoS Computational Biology**, v. 6, n. 4, April 2010.

SCHMIDT, W.; KRAAIJVELD, M.; DUIN, R. Feedforward neural networks with random weights. In: **Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on**. [S.l.: s.n.], 1992. p. 1–4.

SEHGAL, M.; SONG, C.; EHLERS, V. L.; JR., J. R. M. Learning to learn – intrinsic plasticity as a metaplasticity mechanism for memory formation. **Neurobiology of Learning and Memory**, v. 105, n. 0, p. 186–199, 2013.

SILVA, D. N. G.; PACIFICO, L. D. S.; LUDERMIR, T. An evolutionary extreme learning machine based on group search optimization. In: **2011 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2011. p. 574–580.

STEEGE, F.; STEPHAN, V.; GROB, H. Effects of noise-reduction on neural function approximation. In: **Proc. 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning**. [S.l.: s.n.], 2012. p. 73–78.

STEMMLER, M.; KOCH, C. How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. **Nature Neuroscience**, v. 2, p. 521–527, 1999.

STEVENS, J. P. Outliers and influential data points in regression analysis. **Psychological Bulletin**, v. 95, n. 2, p. 334–344, 1984. Available from Internet: <http://isites.harvard.edu/fs/docs/icb.topic477909.files/outliers_regression.pdf>.

STORN, R. On the usage of differential evolution for function optimization. In: **1996 Biennial Conference of the North American Fuzzy Information Processing Society, 1996. NAFIPS.** [s.n.], 1996. p. 519–523. Available from Internet: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=534789>.

STORN, R.; PRICE, K. **Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces**. 1995. Technical Report. International Computer Science Institute.

STORN, R.; PRICE, K. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. **Journal of Global Optimization**, v. 11, n. 4, p. 341–359, 1997. Available from Internet: <http://link.springer.com/article/10.1023%2FA%3A1008202821328>.

TAIANA, M.; SANTOS, J.; GASPAR, J.; NASCIMENTO, J.; BERNARDINO, A.; LIMA, P. Tracking objects with generic calibrated sensors: An algorithm based on color and 3D shape features. **Robotics and Autonomous Systems**, v. 58, p. 784–795, 2010.

TIKHONOV, A. On solving incorrectly posed problems and method of regularization. **Doklady Akademii Nauk USSR**, v. 151, p. 501–504, 1963.

TRIESCH, J. Synergies between intrinsic and synaptic plasticity in individual model neurons. In: SAUL, L. K.; WEISS, Y.; BOTTOU, L. (Ed.). **Advances in Neural Information Processing Systems 17**. Cambridge, MA: MIT Press, 2004. p. 1417–1424. Available from Internet: <http://books.nips.cc/papers/files/nips17/NIPS2004_0591.pdf>.

TRIESCH, J. A gradient rule for the plasticity of a neuron's intrinsic excitability. In: **International Conference on Artificial Neural Networks**. [S.l.: s.n.], 2005. p. 65–79.

TRIESCH, J. Synergies between intrinsic and synaptic plasticity mechanisms. **Neural Computation**, v. 19, n. 4, p. 885–909, April 2007.

TSAI, J.-T.; CHOU, J.-H.; LIU, T.-K. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. **IEEE Transactions on Neural Networks**, v. 17, n. 1, p. 69–80, January 2006.

WANG, L. P.; WAN, C. R. Comments on "the extreme learning machine". **IEEE Transactions On Neural Networks**, v. 19, n. 8, p. 1494–1495, August 2008.

WANG, Y.; CAO, F.; YUAN, Y. A study on effectiveness of extreme learning machine. **Neurocomputing**, v. 74, n. 16, p. 2483–2490, 2011. Advances in Extreme Learning Machine: Theory and Applications Biological Inspired Systems. Computational and Ambient Intelligence Selected papers of the 10th International Work-Conference on Artificial Neural Networks (IWANN2009).

WIDROW, B.; GREENBLATT, A.; KIM, Y.; PARK, D. The No-Prop algorithm: A new learning algorithm for multilayer neural networks. **Neural Networks**, v. 37, p. 182–188, 2013.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: **Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4**. [S.l.: s.n.], 1960. p. 96–104.

XU, J.; LU, Y.; HO, D. W. C. A combined genetic algorithm and orthogonal transformation for designing feedforward neural networks. In: **Proceedings of Third International Conference on Natural Computation (ICNC 2007)**. [S.l.: s.n.], 2007. p. 10–14.

XU, Y.; SHU, Y. Evolutionary extreme learning machine – based on particle swarm optimization. In: WANG, J.; YI, Z.; ZURADA, J.; LU, B.-L.; YIN, H. (Ed.). **Advances in Neural Networks - ISNN 2006**. [S.l.]: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3971). p. 644–652.

YANG, X.-S. **Engineering Optimization: An Introduction with Metaheuristic Applications**. [S.l.]: John Wiley & Sons, 2010. ISBN 978-0-470-58246-6.

YANG, Y.; WANG, Y.; YUAN, X. Bidirectional extreme learning machine for regression problem and its learning effectiveness. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 9, p. 1498–1505, September 2012.

YAO, X. Evolving artificial neural networks. **Proceedings of the IEEE**, v. 87, n. 9, p. 1423–1447, September 1999.

YAO, X.; LIU, Y. A new evolutionary system for evolving artificial neural networks. **IEEE Transactions on Neural Networks**, v. 8, n. 3, p. 694–713, May 1997.

ZAHARIE, D. Differential evolution: from theoretical analysis to practical insights. In: **Proceedings of 20th International Conference on Soft Computing**. [s.n.], 2014. Available from Internet: <http://web.info.uvt.ro/~dzaharie/lucrari/mendel2012.pdf>.

ZHANG, R.; LAN, Y.; HUANG, G.-B.; SOH, Y. Extreme learning machine with adaptive growth of hidden nodes and incremental updating of output weights. In: KAMEL, M.; KARRAY, F.; GUEAIEB, W.; KHAMIS, A. (Ed.). **Autonomous and Intelligent Systems**. [S.l.]: Springer Berlin Heidelberg, 2011, (Lecture Notes in Computer Science, v. 6752). p. 253–262.

ZHANG, R.; LAN, Y.; HUANG, G.-B.; XU, Z.-B. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 2, p. 365–371, February 2012.

ZHANG, Z. Parameter estimation techniques: A tutorial with application to conic fitting. In: **Image and Vision Computing**. [S.l.: s.n.], 1997. v. 15, n. 1, p. 59–76.

ZHAO, G.; SHEN, Z.; MAN, Z. Robust input weight selection for well-conditioned extreme learning machine. **International Journal of Information Technology**, v. 17, n. 1, 2011.

ZHU, Q.-Y.; QIN, A.; SUGANTHAN, P.; HUANG, G.-B. Evolutionary extreme learning machine. **Pattern Recognition**, v. 38, n. 10, p. 1759–1763, 2005.

APPENDIX A – EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms refer to a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution or social behavior of biological organisms (YAO, 1999). They are particularly useful for dealing with large complex problems because they are less likely to be trapped in local minima then traditional gradient-based algorithms (YAO, 1999). In our work, we choose two of these algorithms and their variations: Differential Evolution (DE), Self-Adaptive Differential Evolution (SaDE), Particle Swarm Optimization (PSO) with global or local best neighborhoods.

This chapter is organized as follows: Section A.1 describes the Differential Evolution algorithm, emphasizing its three main parts: mutation (Subsection A.1.1), crossover (Subsection A.1.2) and selection (Subsection A.1.3). Section A.2 describes a variation of DE, named Self-Adaptive Differential Evolution, and how it differs from the original method: mutation strategies (Subsection A.2.1) and parameters adaptation (Subsection A.2.2). At the end, Section A.3 describes the Particle Swarm Optimization algorithm and their two main variations: with global or local best neighborhood setup for particles adaptation, and on Subsection A.3.1 the PSO's velocity component is detailed.

A.1 Differential Evolution

Differential Evolution is a population-based stochastic metaheuristic, designed by Storn and Price in 1995 (STORN; PRICE, 1995; STORN; PRICE, 1997). It is very popular due to its simplicity and effectiveness in solving various types of problems, including multi-objective, multi-modal, dynamic and constrained optimization problems (ZAHARIE, 2014). This method works through a cycle of reproduction and selection operators, where reproduction includes mutation and crossover operators. Each cycle corresponds to an evolutionary generation, which evolves a population of NP solution candidates called *individuals* towards the global optimum. The population of individuals corresponding to generation g will be denoted by $\mathbf{C}(g) = \{\mathbf{c}_1(g), \mathbf{c}_2(g), ..., \mathbf{c}_{NP}(g)\}$ and the components of a vector $\mathbf{c}_i$ will be denoted by $(c_i^1, c_i^2, ..., c_i^{d_n})$, where d_n is the size of the individual vector. The initial population is generated by assigning random values in the search space to the variables of every solution (HU *et al.*, 2013).

Differently than other evolutionary algorithms, the mutation operator is applied first to generate a trial vector $\mathbf{V}_i(g)$ for every individual of the current population, which will then be used by the crossover operator to implement a discrete recombination of $\mathbf{v}_i(g)$ and $\mathbf{c}_i(g)$ to produce the offspring $\mathbf{c}'_i(g)$. That being said, the general structure of DE is described in Algorithm 8, where the eventual improvements are placed on mutation and crossover operators. In almost all implementations, the selection operator is working as follows: the trial element is compared with the current element and the best of them is transferred to the new population (ZAHARIE, 2014).

Algorithm 8: General DE Algorithm

initialize the parameters from DE;
$g \leftarrow 0$;
set $\mathbf{c}_{min}$ and $\mathbf{c}_{max}$;
initialize population $\mathbf{C}(0) \sim \mathscr{U}(\mathbf{c}_{min}, \mathbf{c}_{max})$ of NP individuals;
compute fitness: $\{f(\mathbf{c}_1(0), ..., f(\mathbf{c}_{NP}(0)\}$;
while stopping condition(s) not true **do**
  for each individual, $\mathbf{c}_i(g) \in \mathbf{C}(g)$, $i = 1, ..., NP$ **do**
    create: $\mathbf{v}_i(g) \leftarrow generateMutation(\mathbf{C}(g))$;
    create an offspring: $\mathbf{c}'_i(g) \leftarrow crossover(\mathbf{C}(g), \mathbf{V}(g))$;
    compute fitness: $f(\mathbf{c}'_i(g))$;
    if $f(\mathbf{c}'_i(g)) < f(\mathbf{c}_i(g))$ **then**
      $\mathbf{c}_i(g+1) \leftarrow \mathbf{c}'_i(g)$;
      $f(\mathbf{c}_i(g+1)) \leftarrow f(\mathbf{c}'_i(g))$;
    else
      $\mathbf{c}_i(g+1) \leftarrow \mathbf{c}_i(g)$;
      $f(\mathbf{c}_i(g+1)) \leftarrow f(\mathbf{c}_i(g))$;
    end if
  end for
  $g \leftarrow g+1$;
end while
return individual with best fitness: $\mathbf{c}_{best}$.

As mentioned before, the DE variants mainly differ with respect to the mutation and crossover operators. The most used ones are briefly described in the following.

A.1.1 Mutation

Applied to produce a trial vector $\mathbf{v}_i$, the most frequently used mutation strategies implemented are listed as follows and more details about them can be seen in Storn (1996).

- **DE/rand/1**: the most common one, generates $\mathbf{v}_i$ with three elements i_1, i_2 and i_3, where $\mathbf{c}_{i_1}(g)$ is a target vector, $i \neq i_1 \neq i_2 \neq i_3$ and $i_1, i_2, i_3 \sim \mathscr{U}(1, NP)$. They are used

to construct a difference term to be added to the target vector (see Equation A.1) and the constant $F \in (0,\infty)$ is a *scale factor* or *mutation factor* that controls the amplification of the differential variation (ENGELBRECHT, 2007).

$$\mathbf{v}_i(g) = \mathbf{c}_{i_1}(g) + F(\mathbf{c}_{i_2}(g) - \mathbf{c}_{i_3}(g)). \tag{A.1}$$

- **DE/rand/$\mathbf{n_v}$**: for this strategy, more than one difference is used to calculate $\mathbf{v}_i$, as shown below:

$$\mathbf{v}_i(g) = \mathbf{c}_{i_1}(g) + F\sum_{k=1}^{n_v}(\mathbf{c}_{i_{2,k}}(g) - \mathbf{c}_{i_{3,k}}(g)), \tag{A.2}$$

  where $\mathbf{c}_{i_{2,k}}(g) - \mathbf{c}_{i_{3,k}}(g)$ shows the k-th difference vector. The larger the value of n_v, the more directions can be explored per generation (ENGELBRECHT, 2007).
- **DE/best/1**: is a small variance from *DE/rand/1*, where the target vector $\mathbf{c}_{i_1}(g)$ becomes the best individual $\mathbf{c}_{best}(g)$ in g-th generation (see Equation A.3).

$$\mathbf{v}_i(g) = \mathbf{c}_{best}(g) + F(\mathbf{c}_{i_2}(g) - \mathbf{c}_{i_3}(g)). \tag{A.3}$$

- **DE/rand-to-best/$\mathbf{n_v}$**: other common variant is based on changing the selected target vector with linear combinations involving the best individual $\mathbf{c}_{best}(g)$ and/or the target element $\mathbf{c}_{i_1}(g)$ from the current population (see Equation A.4) (ZAHARIE, 2014).

$$\mathbf{v}_i(g) = \gamma\mathbf{c}_{best}(g) + (1-\gamma)\mathbf{c}_{i_1}(g) + F\sum_{k=1}^{n_v}(\mathbf{c}_{i_{2,k}}(g) - \mathbf{c}_{i_{3,k}}(g)), \tag{A.4}$$

  where $\gamma \in (0,1)$ controls the influence of the best individual on the target vector.
- **DE/current-to-rand/$\mathbf{1+n_v}$**: with this strategy, the parent is mutated using at least two difference vectors (ENGELBRECHT, 2007). One is calculated from the current individual and a random vector, while the remaining are calculated using only randomly selected vectors, as shown on Equation A.5.

$$\mathbf{v}_i(g) = \mathbf{c}_i(g) + F_1(\mathbf{c}_{i_1}(g) - \mathbf{c}_i(g)) + F_2\sum_{k=1}^{n_v}(\mathbf{c}_{i_{2,k}}(g) - \mathbf{c}_{i_{3,k}}(g)), \tag{A.5}$$

  where F_1 may be equal or not to F_2.

A.1.2 Crossover

After the mutation phase, crossover operator is applied to each pair of individuals $\mathbf{c}_i(g)$ and respective trial vectors $\mathbf{v}_i(g)$ to produce the offspring $\mathbf{c}'_i(g)$. The basic DE algorithm employs the *binomial crossover*, defined as follows:

$$c_i^{j\prime}(g) = \begin{cases} c_i^j(g) & \text{if } \text{rand}_j \leq CR \text{ and } j = j_{rand} \\ v_i^j(g) & \text{otherwise.} \end{cases} \tag{A.6}$$

In Equation A.6, the crossover rate $CR \in [0,1)$ is a user-specified constant that controls the number of elements of parent $\mathbf{c}_i(g)$ that will change, and $j = 1,...,d_n$. The larger CR value is, the more elements of the trial vector will be used to produce the offspring, and less of the parent. To enforce that at least one element will differ from the parent, a randomly selected $j_{rand} \sim \mathscr{U}(1,d_n)$ is chosen from the trial vector to pass to the next generation.

Besides, there is an *exponential crossover* operator, which is an alternative to the aforementioned method. In this case, the elements of the offspring $\mathbf{c}'_i(g)$ are inherited from the respective trial vector $\mathbf{v}_i(g)$, starting from a randomly chosen element until the first time $\text{rand}_j > CR$. The remaining elements of $\mathbf{c}'_i(g)$ are then copied from the corresponding parent $\mathbf{c}_i(g)$ by at least one element.

A.1.3 Selection

Finally, the selection operator is responsible for constructing the population for the next generation. As already mentioned, a very simple rule based on fitness is applied (see Equation A.7), which ensures that the average fitness of the population does not deteriorate (ENGELBRECHT, 2007).

$$\mathbf{c}_i(g+1) = \begin{cases} \mathbf{c}'_i(g) & f(\mathbf{c}'_i(g)) < f(\mathbf{c}_i(g)) \\ \mathbf{c}_i(g) & \text{otherwise.} \end{cases} \tag{A.7}$$

A.2 Self-Adaptive Differential Evolution

Proposed by Qin *et al.* (2009), the Self-Adaptive Differential Evolution determines adaptively an appropriate strategy and its associated parameter values at different stages of the evolution process. Just like many other methods, DE has control parameters

that significantly influence its optimization performance, such as population size NP, scaling factor F and crossover rate CR. Therefore, in an attempt of avoiding trial-and-error searching processes, which are time-consuming and has high computational cost, trial vector generation strategies and their parameters are gradually self-adapted from their previous experiences in generating promising solutions.

A.2.1 Mutation Strategies

Within this DE variation, we keep a candidate pool of K_s strategies from where each individual $\mathbf{c}_i(g)$ will elect one based on the probability $p_k(g)$ $(k = 1, ..., K_s)$, which represents the probability of that k-th strategy to be chosen. The more successful a strategy has behaved to generate promising solutions, the more likely it will be chosen in the next generations. This pool contains four different strategies:

- *DE/rand/1* (see Equation A.1)
- *DE/rand/2* (see Equation A.2)
- *DE/current-to-rand/1* (see Equation A.5), where $F_1 \neq F_2$ and
- *DE/rand-to-best/2*: where the implementation suggested in Qin *et al.* (2009) is different from the one described in Storn (1996) and Engelbrecht (2007). In his work, this strategy is given by:

$$\mathbf{v}_i(g) = \mathbf{c}_i(g) + F(\mathbf{c}_{best}(g) - \mathbf{c}_i(g)) + F(\mathbf{c}_{i_1}(g) - \mathbf{c}_{i_2}(g)) + F(\mathbf{c}_{i_3}(g) - \mathbf{c}_{i_4}(g)). \tag{A.8}$$

A learning period LP, constituted of a fixed number of generations, is established so that the algorithm can acquire information about the success and failure of those strategies through the first generations in order to determine real values to $p_k(g)$. Therefore, those probabilities during the learning period are equal, i.e., $p_k(g_{LP}) = \frac{1}{K_s}$.

On the original paper (QIN *et al.*, 2009), the authors used the *stochastic universal selection method* (BAKER, 1987) to choose one strategy for each individual in current population. However, posterior works, such as Cao *et al.* (2012) does not mention any particular method for such task, which gives us the option of using it or not.

After the mutation is done, depending on the strategy, the crossover operator changes as well. For the first three strategies, the binomial crossover is chosen because of its popularity (QIN *et al.*, 2009), and for the last one, the offspring receives the entire trial vector $\mathbf{v}_i(g)$. Once again, in Cao *et al.* (2012), it differs by using only binomial crossover.

Table 29 – Success memory.

Index	**Strategy 1**	**Strategy 2**	...	**Strategy K_s**
1	$ns_1(g-LP)$	$ns_2(g-LP)$	...	$ns_{K_s}(g-LP)$
2	$ns_1(g-LP+1)$	$ns_2(g-LP+1)$	...	$ns_{K_s}(g-LP+1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$ns_1(g-1)$	$ns_2(g-1)$	...	$ns_{K_s}(g-1)$

Source: Qin *et al.* (2009).

Table 30 – Failure memory.

Index	**Strategy 1**	**Strategy 2**	...	**Strategy K_s**
1	$nf_1(g-LP)$	$nf_2(g-LP)$	...	$nf_{K_s}(g-LP)$
2	$nf_1(g-LP+1)$	$nf_2(g-LP+1)$	...	$nf_{K_s}(g-LP+1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$nf_1(g-1)$	$nf_2(g-1)$	...	$nf_{K_s}(g-1)$

Source: Qin *et al.* (2009).

Subsequently, all offspring are evaluated and the number of those generated by k-th strategy that succeeded to pass to next generation is recorded as $ns_k(g)$, as well as the number of those who failed is recorded as $nf_k(g)$. LP generations' success ns_k and failures nf_k are stored in *success* and *Failure memory*, which are illustrated on Tables 29 and 30. Once the generations reach a value larger than LP, the earliest records on these memories, i.e. $\mathbf{ns}(g-LP)$ and $\mathbf{nf}(g-LP)$, are replaced by new ones. Also, the probabilities of choosing different strategies will be updated at each subsequent generation based on the success and failure memories, accordingly with Equation A.9.

$$p_k(g) = \frac{S_k(g)}{\sum_{k=1}^{K_s} S_k(g)}, \tag{A.9}$$

where $S_k(g)$ is defined as follows.

$$S_k(g) = \frac{\sum_{\delta=g-LP}^{g-1} ns_k(\delta)}{\sum_{\delta=g-LP}^{g-1} ns_k(\delta) + \sum_{\delta=g-LP}^{g-1} nf_k(\delta)} + \varepsilon, \tag{A.10}$$

with $k = 1, ..., K_s$, $g > LP$ and $\varepsilon = 0.01$, which is used to avoid possible null success rates. Equation A.10 presents the success rate of offspring generated by the k-th strategy and succeeded to pass to next generation within the previous LP generations with respect

to generation g. The larger $S_k(g)$, the larger the probability $p_k(g)$ of applying the k-th strategy to generate the offspring $\mathbf{c}'_i(g)$ at current generation g.

A.2.2 Parameter Adaptation

In basic DE there are three control parameters: the scaling factor F, crossover rate CR and the number of individuals NP, whereas SaDE needs only NP and the amount of learning period LP to be adjusted by the user. The population size and learning period do not need a fine-tuning, however, F and CR may be sensitive to different problems and F is directly related to the speed of the algorithm (QIN *et al.*, 2009). In SaDE, for every $\mathbf{c}_i(g)$, F is randomly generated according to the normal distributions $\mathcal{N}(0.5, 0.3)$ and CR is continually adapted as $\mathcal{N}(CR_{m_k}, 0.1)$. The CR_{m_k} represents the mean value and it is initialized as 0.5. Within the first LP generations, a $CRMemory_k$ is created to store those CR values related to k-th strategy that were successful to create offspring that passed to the next generation. After those LP generations, the median value value stored in $CRMemory_k$ will be calculated to overwrite CR_{m_k}.

All of those steps aforementioned are summarized in Algorithm 9.

A.3 Particle Swarm Optimization

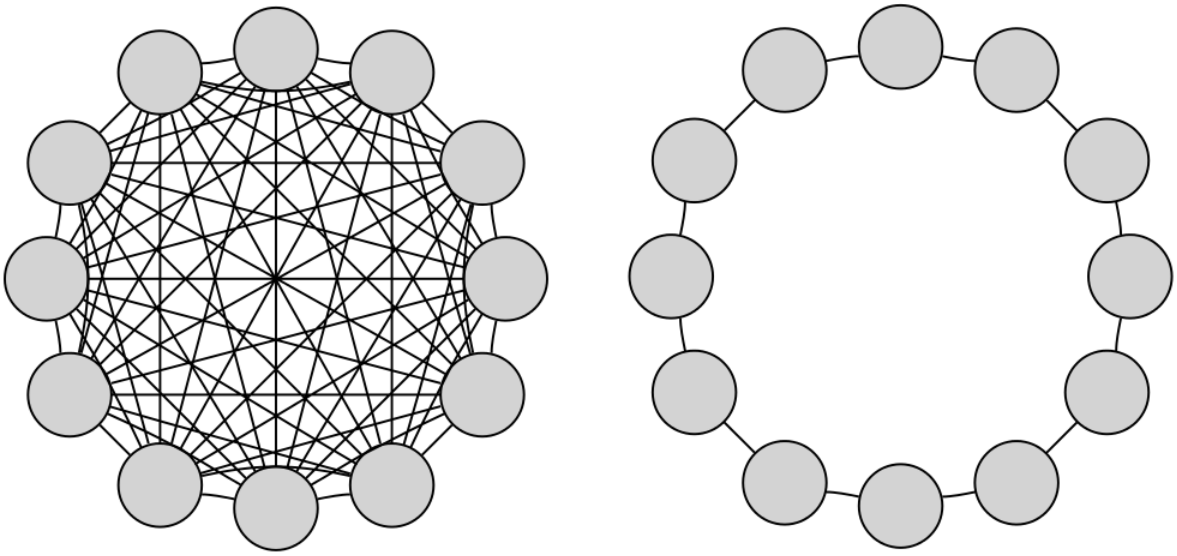
The Particle Swarm Optimization was developed as a population-based search that simulates the social behavior of fish schooling or bird flocking, first proposed in (KENNEDY; EBERHART, 1995). In most of PSO implementations, particles of a swarm move through the search space using a combination of an attraction to the best solution that each of them has found, and an attraction to the best solution that all particles in their neighborhood have found (BRATTON; KENNEDY, 2007). This neighborhood is defined for each individual particle as the subset of particles it is able to communicate with (BRATTON; KENNEDY, 2007) and its range determines the kind of topology the algorithm will assume.

The neighborhood in a global topology comprises all particles, while in local topology, which was first introduced by Eberhart and Kennedy (1995), the neighborhood comprises only a small number of particles that are linked by their indexes. Figure 42 illustrates the relationship differences between particles within both types of neighborhood.

Algorithm 9: General SaDE Algorithm (QIN *et al.*, 2009)

$g \leftarrow 0$;
set $\mathbf{c}_{min}$ and $\mathbf{c}_{max}$;
initialize population $\mathbf{C}(0) \sim \mathscr{U}(\mathbf{c}_{min}, \mathbf{c}_{max})$ of NP individuals;
initialize CR_{m_k}, $p_k(g)$ with $k \leftarrow 1, ..., K_s$ and LP;
compute fitness: $\{f(\mathbf{c}_1(0), ..., f(\mathbf{c}_{NP}(0)\}$;
while stopping condition(s) not true **do**
  if $g > LP$ **then**
    for $k \leftarrow 1$ to K_s **do**
      update $p_k(g)$; {Calculate $p_k(g)$ and update *Success* and *Failure Memory.*}
      remove $ns_k(g-LP)$ and $nf_k(g-LP)$ out of *Success* and *Failure Memory*;
    end for
  end if
  use stochastic universal sampling to select one strategy k for each individual in $\mathbf{c}_i(g)$;
  for $i \leftarrow 1$ to NP **do**
    $F_i \sim \mathscr{N}(0.5, 0.3)$; {Assign control parameter F.}
  end for
  if $g > LP$ **then**
    for $k \leftarrow 1$ to K_s **do**
      $CR_{m_k} \leftarrow median(CRMemory_k)$; {Assign control parameter CR.}
    end for
  end if
  for $k \leftarrow 1$ to K_s **do**
    for $i \leftarrow 1$ to NP **do**
      $CR_{k,i} \sim \mathscr{N}(CR_{m_k}, 0.1)$;
      while $CR_{k,i} < 0$ or $CR_{k,i} > 1$ **do**
        $CR_{k,i} \sim \mathscr{N}(CR_{m_k}, 0.1)$;
      end while
    end for
  end for
  Produce offspring $\mathbf{C}'(g)$ using the associated strategy k and parameters F_i and $CR_{k,i}$;
  if $\mathbf{c}'_i(g)$ outside boundaries **then**
    $\mathbf{c}'_i(g) \sim \mathscr{U}(\mathbf{c}_{min}, \mathbf{c}_{max})$;
  end if
  for $i \leftarrow 1$ to NP **do**
    compute fitness: $f(\mathbf{c}'_i(g))$; {Selection}
    if $f(\mathbf{c}'_i(g)) \leq f(\mathbf{c}_i(g))$ **then**
      $\mathbf{c}_i(g+1) \leftarrow \mathbf{c}'_i(g)$;
      $f(\mathbf{c}_i(g+1)) \leftarrow f(\mathbf{c}'_i(g))$;
      $ns_k(g) = ns_k(g) + 1$;
      $CRMemory_k \leftarrow CR_{k,i}$;
      if $f(\mathbf{c}'_i(g)) < f(\mathbf{c}_{best})$ **then**
        $\mathbf{c}_{best} \leftarrow \mathbf{c}'_i(g)$;
        $f(\mathbf{c}_{best}) \leftarrow f(\mathbf{c}'_i(g))$;
      end if
    else
      $\mathbf{c}_i(g+1) \leftarrow \mathbf{c}_i(g)$;
      $f(\mathbf{c}_i(g+1)) \leftarrow f(\mathbf{c}_i(g))$;
      $nf_k(g) = nf_k(g) + 1$;
    end if
  end for
  $g \leftarrow g+1$;
end while
return individual with best fitness: $\mathbf{c}_{best}$.

Figure 42 – PSO topologies.
(a) Global (gbest) topology. (b) Local topology (lbest) in ring shape.



Source: Bratton and Kennedy (2007).

In the case of *ring* social structure (Figure 42b), each particle is connected to two adjacent members in the population, although this can be generalized to a number bigger than two, accordingly with the user will.

Local topologies present a slower convergence due to the information exchange between small groups of particles, which creates several independent search groups at the beginning. However, this feature may prevent early convergence in a suboptimal solution (ENGELBRECHT, 2007). The algorithms that use global topology are known as *gbest* PSO and the ones with local topology are *lbest* PSO.

The internal structures aforementioned as particles are individually composed by three vectors: the current position of i-th particle $\mathbf{c}_i \in \mathbb{R}^{Pr}$, the previous i-th best position $\mathbf{p}_i \in \mathbb{R}^{Pr}$ and the i-th particle velocity $\boldsymbol{\nu}_i \in \mathbb{R}^{Pr}$, where Pr is the search space dimensionality. Let $\mathbf{c}_i(g)$ be a solution candidate described as a set of coordinates of a point in space, whose position is changed by adding a velocity $\boldsymbol{\nu}_i$, as demonstrated on Equations A.11, Equation A.12 for *gbest* PSO and Equation A.13 for *lbest* PSO.

$$\mathbf{c}_i(g+1) = \mathbf{c}_i(g) + \boldsymbol{\nu}_i(g+1). \tag{A.11}$$

$$\nu_i^j(g+1) = \nu_i^j(g) + \varsigma_1 r_1^j(p_i^j(g) - c_i^j(g)) + \varsigma_2 r_2^j(p_{best}^j(g) - c_i^j(g)). \tag{A.12}$$

$$v_i^j(g+1) = v_i^j(g) + \varsigma_1 r_1^j(p_i^j(g) - c_i^j(g)) + \varsigma_2 r_2^j(pl_k^j(g) - c_i^j(g)), \quad \text{(A.13)}$$

where ς_1 and ς_2 are positive acceleration constants, r_1^j and r_2^j, with $j = 1, ..., Pr$, are random values sampled from an uniform distribution at each velocity update. They are responsible for introducing a stochastic element to the algorithm.

An overview about the PSO process is presented on Algorithm 10, for *gbest* PSO, and Algorithm 11, for *lbest* PSO.

A.3.1 Velocity component

The velocity vector drives the optimization process and reflects both the particle's experience knowledge and the information exchanged within the neighborhood (ENGELBRECHT, 2007). It is adjusted mainly by two components: the *cognitive component* and *social component*. The first one is represented by the difference between $\mathbf{p}_i$, that is the personal best position found so far, and the individual's current position. The social component is the socially exchanged information given by the stochastically weighted difference between the neighborhood's best position $\mathbf{pl}_k$ (*lbest*) or $\mathbf{c}_{best}$ (*gbest*) and the individual's current position. These adjustments to the particle's movement through space cause it to search around the two best positions (CLERC; KENNEDY, 2002).

The association of the acceleration constants ς_1 and ς_2 with the random values r_1^j and r_2^j scales the contribution of cognitive and social components respectively. This stochastic amount will be mentioned from now on as $\varphi_1 = \varsigma_1 r_1$ and $\varphi_2 = \varsigma_2 r_2$.

A number of basic modifications to the PSO have been developed to improve the speed of convergence and the quality of its solutions (ENGELBRECHT, 2007). In the work from Bratton and Kennedy (2007), they define a new standard algorithm designed to be a straightforward extension of the original algorithm while taking into account those developments that can be expected to improve performance on standard measures. One of the suggested changes was proposed to balance the exploration-exploitation trade-off and prevent that the velocities explode to large values. This is accomplished by constricting the velocities by a *constriction coefficient* χ, proposed in (CLERC; KENNEDY, 2002).

The new velocity update equation is now given by:

$$\mathbf{v}_i^j(g+1) = \chi \left[v_i^j(g) + \varphi_1(p_i^j(g) - c_i^j(g)) + \varphi_2(p_{best}^j(g) - c_i^j(g)) \right], \tag{A.14}$$

where

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi(\varphi-4)}|}, \tag{A.15}$$

with $\kappa \in [0,1]$, $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$.

Algorithm 10: *gbest* PSO

$g \leftarrow 0$;
set $\mathbf{c}_{min}$ and $\mathbf{c}_{max}$;
initialize population $\mathbf{C}(0) \sim \mathscr{U}(\mathbf{c}_{min}, \mathbf{c}_{max})$ of NP particles;
compute fitness $\{f(\mathbf{c}_1(0), ..., f(\mathbf{c}_{NP}(0)\}$;
set large numbers to $f(\mathbf{p}_i(0)$ and $f(\mathbf{p}_{best}(0))$;
while stopping condition(s) not true **do**
  for each particle $i = 1, ..., NP$ **do**
    if $f(\mathbf{c}_i(g)) < f(\mathbf{p}_i(g))$ **then**
      $\mathbf{p}_i(g) \leftarrow \mathbf{c}_i(g)$; {set personal best position}
      $f(\mathbf{p}_i(g)) \leftarrow f(\mathbf{c}_i(g))$;
    end if
    if $f(\mathbf{c}_i(g)) < f(\mathbf{p}_{best}(g))$ **then**
      $f(\mathbf{p}_{best}(g)) \leftarrow f(\mathbf{c}_i(g))$; {set global best position}
      $\mathbf{p}_{best}(g) \leftarrow \mathbf{c}_i(g)$;
    end if
  end for
  for each particle $i = 1, ..., NP$ **do**
    update the velocity;
    update the position;
  end for
  $g \leftarrow g + 1$;
end while

Algorithm 11: *lbest* PSO

1: $g \leftarrow 0$;
2: set $\mathbf{c}_{min}$ and $\mathbf{c}_{max}$;
3: initialize population $\mathbf{C}(0) \sim \mathscr{U}(\mathbf{c}_{min}, \mathbf{c}_{max})$ of NP particles;
4: compute fitness $\{f(\mathbf{c}_1(0), ..., f(\mathbf{c}_{NP}(0)\}$;
5: set large numbers to $f(\mathbf{p}_i(0)$ and $f(\mathbf{pl}_k(0))$;
6: **while** stopping condition(s) not true **do**
7: **for** each particle $i = 1, ..., NP$ **do**
8: **if** $f(\mathbf{c}_i(g)) < f(\mathbf{c}_i(g))$ **then**
9: $\mathbf{c}_i(g) \leftarrow \mathbf{c}_i(g)$; {set personal best position}
10: $f(\mathbf{c}_i(g) \leftarrow f(\mathbf{c}_i(g))$;
11: **end if**
12: **if** $f(\mathbf{c}_i(g)) < f(\mathbf{pl}_k(g))$ **then**
13: $f(\mathbf{pl}_k(g)) \leftarrow f(\mathbf{c}_i(g))$; {set neighborhood best position}
14: $\mathbf{pl}_k(g) \leftarrow \mathbf{c}_i(g)$;
15: **end if**
16: **end for**
17: **for** each particle $i = 1, ..., NP$ **do**
18: update the velocity;
19: update the position;
20: **end for**
21: $g \leftarrow g + 1$;
22: **end while**

APPENDIX B – DATASETS

In this work, we investigate the performance of the proposed methods with 6 different datasets regarding regression problems.

B.1 Real-world problems

- **Auto-MPG**: taken from UCI database, this set concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 discrete and 5 continuous attributes. The original set has 1 nominal attribute which refers to the car name and it is unique for each sample. In this case, we ignored this information, resulting in a set of only 7 attributes and one output. Another observation is the absence of 6 attributes values, which leave us with 392 full samples.
- **Body Fat**: it is a set from Statlib database (MEYER; VLACHOS, 1989) that concerns the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.

  There are 14 attributes available, all continuous, and one output which refers to the percentage of body fat.
- **Wisconsin Prognostic Breast Cancer**: it is a set from UCI database for regression problems, where each sample represents follow-up data for one breast cancer case, including only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis. The goal here is to predict the time for cancer to recur.

  There are 198 samples with 34 attributes, but the first one is just an ID number, which is ignored. The second one is the nominal predicting field that indicates the outcome: R = recurrent, N = nonrecurrent. In this case, the recurrent cases were substituted by the numerical value 1, and the nonrecurrent by -1. At the end, we are left with 32 attributes.
- **CPU**: also taken from UCI database, the estimated relative performance values of a CPU, described in terms of its cycle time, memory size etc. There are 209 samples and the first two data columns are ignored because they inform the vendor name and model, which is almost a unique identification for each sample. The last data column is also ignored since it is the estimation result of this dataset owner, which

leaves us with 6 input attributes and 1 output.

- **iCub imperative pointing**: this dataset was collected during the sandwich Ph.D. period spent at Research Institute for Cognition and Robotics (CoR-Lab) in Bielefeld, Germany, supported by CNPq and DAAD, the German Academic Exchange Service. Publications over this set can be seen in (FREIRE *et al.*, 2012a; FREIRE *et al.*, 2012b; LEMME *et al.*, 2013).

  As a visually guided sensorimotor behavior, pointing has been an important topic of research in cognitive science, especially in what concerns the development of infants' preverbal communicative skill (LEMME *et al.*, 2013). Classically, infants are thought to point for two main reasons (BATES *et al.*, 1975; LISZKOWSKI *et al.*, 2006). Firstly, they point when they want a nearby adult to do something for them (e.g. give them something). This is called imperative pointing and consists in extending the arm as if reaching for an object. It has been proposed that children use imperative pointing as a result of not being able to reach objects that are too far away to grasp. Secondly, infants point when they want an adult to share attention with them to some interesting event or object.

Figure 43 – Setup for iCub's dataset harvesting.

(a) iCub's simulator.

(b) On the left, the physical setup is shown. A webcam captures video of an user moving a red ball. The video is then projected on a screen inside the simulator (center panel). On the right, the view from the left and right simulated cameras is displayed.

Source: Lemme *et al.* (2013).

This behavior was executed with the aid of the iCub's simulator, however, based on real images from a webcam, as demonstrated on Figure 43. A red ball of approximately 6 cm of diameter is the target object within the visual field of the robot. This is accomplished by recording the image of a user moving this ball

freely in space with a webcam. The video is then projected on a screen inside the simulator, as shown on center panel in Figure 43. This configuration emulates a direct interaction with the real world, which is seen by the simulated robot with simulated camera eyes. In the experiment, recorded data comprise pixel coordinates of the ball from both eye-cameras (i_L, j_L) and (i_R, j_R), the 3D position of the ball (x_b, y_b, z_b) estimated from the simulated left camera image by using an object tracker available on iCub's software repository (TAIANA *et al.*, 2010), end-effector position (x_e, y_e, ze) and joint angles (θ_1 , ..., θ_7) (LEMME *et al.*, 2013).

The dataset resulted in 491 samples and was first presented on Freire *et al.* (2012b) and later on Lemme *et al.* (2013). The goal is to map pixel coordinates obtained from binocular vision directly to joint angles, defining 4 attributes as inputs and 7 outputs.

- **Servo Motor**: taken from UCI, its goal relies on predicting the rise time of a servomechanism in terms of gain settings and choices of mechanical linkages. There are 167 samples with 4 attributes and one output. Two of those attributes (motor and screw) has nominal values which were substituted as shown on Table 31.

Table 31 – Substitution of nominal attribute in Servo dataset into a numerical one.

Motor and Screw Values	A	B	C	D	E
New Value	1	2	3	4	5

Source: author.

APPENDIX C – DETAILED TABLES FOR ROBUST ELM

In this appendix, we present the correspondent mean, median, maximum, minimum and standard deviation values of the number of hidden neurons and output weight euclidean norms from the graphics presented in Chapter 6.

C.1 iCub dataset

Table 32 – Number of hidden neurons with 1 sided contamination (iCub dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	4	4	2	6	9.428×10^{-1}
	ELM-BIP	2.400	2	2	4	8.433×10^{-1}
	R-ELM/BIP	8.800	8	6	14	2.150
	RELM	6	5	2	16	4.216
	IRWLS-ELM	15.80	16	10	22	3.190
20%	ELM	2.200	2	2	4	6.325×10^{-1}
	ELM-BIP	2.400	2	2	4	8.433×10^{-1}
	R-ELM/BIP	18	11	6	48	14.24
	RELM	11.20	6	2	44	12.55
	IRWLS-ELM	14.20	16	6	22	5.287
30%	ELM	3	2	2	6	1.700
	ELM-BIP	2.400	2	2	4	8.433×10^{-1}
	R-ELM/BIP	28.20	27	6	48	13.71
	RELM	12	4	2	50	15.43
	IRWLS-ELM	12.80	14	4	22	6.339
40%	ELM	4.800	4	4	6	1.033
	ELM-BIP	3	3	2	4	1.054
	R-ELM/BIP	41.40	44	32	48	6.041
	RELM	33.60	37	14	46	10.74
	IRWLS-ELM	9.400	10	4	16	3.893

Source: author.

Table 33 – Number of hidden neurons with 2 sided contamination (iCub dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	4.200	4	4	6	6.325×10^{-1}
	ELM-BIP	3.400	4	2	4	9.661×10^{-1}
	R-ELM/BIP	9	8	6	20	4.243
	RELM	7.600	5	2	24	6.979
	IRWLS-ELM	15.20	17	4	24	6.746
20%	ELM	3	3	2	4	1.054
	ELM-BIP	2	2	2	2	
	R-ELM/BIP	10.60	6	4	42	11.55
	RELM	12.60	10	2	40	10.63
	IRWLS-ELM	14.60	15	4	24	6.186
30%	ELM	2.400	2	2	4	8.433×10^{-1}
	ELM-BIP	2	2	2	2	
	R-ELM/BIP	13.80	12	2	32	9.773
	RELM	11.60	8	2	32	10.36
	IRWLS-ELM	6.600	4	2	18	4.904
40%	ELM	2.800	2	2	4	1.033
	ELM-BIP	2	2	2	2	
	R-ELM/BIP	6.600	4	2	20	5.967
	RELM	15.60	18	2	26	8.208
	IRWLS-ELM	5.600	5	2	10	2.951

Source: author.

C.2 Auto-MPG dataset

Table 34 – Number of hidden neurons with 1 sided contamination (Auto-MPG dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	21.40	20	12	40	8.695
	ELM-BIP	14.80	14	10	22	3.676
	R-ELM/BIP	29.20	29	22	38	5.350
	RELM	42.80	44	14	64	12.93
	IRWLS-ELM	64.40	71	24	100	22.25
20%	ELM	16.80	16	10	30	5.978
	ELM-BIP	9.400	9	6	14	3.134
	R-ELM/BIP	28	25	14	44	9.189
	RELM	29.60	31	12	48	13.19
	IRWLS-ELM	14.80	13	8	28	5.827
30%	ELM	15.60	13	10	26	5.317
	ELM-BIP	8.800	8	6	16	3.155
	R-ELM/BIP	39.40	39	12	64	15.09
	RELM	39.80	38	14	80	19.52
	IRWLS-ELM	14.20	14	8	20	4.264
40%	ELM	14.40	13	6	32	7.648
	ELM-BIP	6.400	6	4	10	2.459
	R-ELM/BIP	15.40	12	2	42	14.27
	RELM	14.60	11	2	40	13.70
	IRWLS-ELM	12.20	10	6	24	5.453

Source: author.

Table 35 – Number of hidden neurons with 2 sided contamination (Auto-MPG dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	22.80	22	12	36	7.495
	ELM-BIP	13.40	13	6	20	4.006
	R-ELM/BIP	39	37	16	68	14.02
	RELM	42.40	42	32	54	6.720
	IRWLS-ELM	57.60	60	14	84	24.25
20%	ELM	10.80	10	6	24	5.266
	ELM-BIP	9.600	8	4	20	4.195
	R-ELM/BIP	31.40	31	14	46	9.845
	RELM	37.20	31	20	56	15.44
	IRWLS-ELM	14.40	14	10	18	2.459
30%	ELM	10.60	10	8	16	2.836
	ELM-BIP	8.400	8	4	12	2.271
	R-ELM/BIP	32.20	31	12	60	13.97
	RELM	25	24	10	52	12.52
	IRWLS-ELM	14	15	10	18	2.981
40%	ELM	10.80	9	6	20	4.237
	ELM-BIP	9.200	8	4	20	4.826
	R-ELM/BIP	18	17	8	34	7.542
	RELM	21	18	6	36	11.13
	IRWLS-ELM	13.40	14	10	20	3.134

Source: author.

C.3 Bodyfat dataset

Table 36 – Number of hidden neurons with 1 sided contamination (Bodyfat dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	13.80	14	10	20	3.190
	ELM-BIP	10.80	10	6	18	3.425
	R-ELM/BIP	84.20	82	68	98	9.449
	RELM	76.20	76	44	100	16.45
	IRWLS-ELM	16.20	16	14	20	1.989
20%	ELM	11.20	11	8	14	2.348
	ELM-BIP	10.40	11	4	20	4.971
	R-ELM/BIP	45.80	50	8	66	17.24
	RELM	34.40	33	10	62	23.49
	IRWLS-ELM	15	14	10	20	3.018
30%	ELM	11	11	6	16	3.432
	ELM-BIP	8.800	8	4	14	3.553
	R-ELM/BIP	30	27	12	50	13.70
	RELM	19.80	13	2	50	19.15
	IRWLS-ELM	14.40	14	10	18	2.459
40%	ELM	12.80	13	4	24	5.514
	ELM-BIP	8	7	4	20	4.807
	R-ELM/BIP	14.20	19	2	28	10.09
	RELM	10	3	2	34	12.33
	IRWLS-ELM	10	10	4	16	3.771

Source: author.

Table 37 – Number of hidden neurons with 2 sided contamination (Bodyfat dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	13	14	10	14	1.700
	ELM-BIP	13.40	13	8	22	4.719
	R-ELM/BIP	82.80	82	66	98	12.41
	RELM	62.20	62	34	90	16.75
	IRWLS-ELM	16.40	16	14	24	3.098
20%	ELM	10.40	10	6	14	2.459
	ELM-BIP	8.200	8	4	14	3.190
	R-ELM/BIP	43.60	39	20	70	17.91
	RELM	45.20	50	12	64	17.97
	IRWLS-ELM	16.40	15	14	24	3.373
30%	ELM	8.800	8	6	14	2.700
	ELM-BIP	6.200	6	2	14	3.706
	R-ELM/BIP	25.80	23	8	56	15.36
	RELM	24.20	14	8	52	18.02
	IRWLS-ELM	13.20	12	10	18	2.348
40%	ELM	10.20	10	6	16	3.048
	ELM-BIP	8.600	6	4	22	5.582
	R-ELM/BIP	19.60	20	12	26	3.864
	RELM	14.80	13	6	24	6.680
	IRWLS-ELM	10.40	10	8	14	2.066

Source: author.

C.4 Breast Cancer dataset

Table 38 – Number of hidden neurons with 1 sided contamination (Breast Cancer dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	21.60	20	16	34	5.317
	ELM-BIP	21.20	20	8	40	9.247
	R-ELM/BIP	62	61	52	76	7.180
	RELM	67	68	52	76	8.124
	IRWLS-ELM	37.20	35	32	50	5.903
20%	ELM	15.80	16	8	22	3.706
	ELM-BIP	15.20	14	10	24	3.795
	R-ELM/BIP	41.60	42	28	50	6.851
	RELM	33.20	31	22	48	10.21
	IRWLS-ELM	27.80	27	20	38	5.692
30%	ELM	13	14	4	16	3.682
	ELM-BIP	12	11	8	18	4
	R-ELM/BIP	22.80	21	16	30	5.266
	RELM	24.80	26	12	36	8.066
	IRWLS-ELM	19.60	19	16	24	2.459
40%	ELM	15	13	10	24	4.447
	ELM-BIP	16.60	16	10	30	5.739
	R-ELM/BIP	15.20	16	2	32	9.390
	RELM	14.20	13	8	26	6.286
	IRWLS-ELM	19.20	21	12	24	4.733

Source: author.

Table 39 – Number of hidden neurons with 2 sided contamination (Breast Cancer dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	18	17	12	28	4.989
	ELM-BIP	21.20	21	16	34	5.266
	R-ELM/BIP	69	69	60	76	6.055
	RELM	63	58	52	78	8.907
	IRWLS-ELM	38.60	38	30	44	4.526
20%	ELM	11	10	4	18	4.137
	ELM-BIP	13	12	8	18	3.432
	R-ELM/BIP	40.20	38	26	60	9.998
	RELM	34	32	20	52	8.165
	IRWLS-ELM	30	32	20	34	5.333
30%	ELM	14	14	4	24	5.888
	ELM-BIP	9.200	9	4	16	3.676
	R-ELM/BIP	27.60	27	22	34	4.195
	RELM	27.40	27	20	42	6.736
	IRWLS-ELM	19.80	21	12	30	6.143
40%	ELM	10.80	11	6	16	3.910
	ELM-BIP	10.40	10	2	20	6.381
	R-ELM/BIP	16.60	17	10	24	4.526
	RELM	18.40	18	8	26	5.400
	IRWLS-ELM	17.60	17	10	30	5.948

Source: author.

C.5 CPU dataset

Table 40 – Number of hidden neurons with 1 sided contamination (CPU dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	10.80	12	2	18	4.917
	ELM-BIP	7.200	7	2	12	3.293
	R-ELM/BIP	20.20	20	10	32	7.084
	RELM	15.60	14	6	36	9.276
	IRWLS-ELM	13	12	8	22	4.830
20%	ELM	9.200	9	2	14	4.022
	ELM-BIP	5.800	6	2	12	2.741
	R-ELM/BIP	21.20	23	8	32	7.671
	RELM	15	14	8	24	5.099
	IRWLS-ELM	10.80	11	4	22	5.181
30%	ELM	8.200	10	2	16	4.566
	ELM-BIP	5.600	2	2	16	5.400
	R-ELM/BIP	19	21	10	24	5.518
	RELM	11.60	12	6	22	4.600
	IRWLS-ELM	6.800	5	2	14	4.022
40%	ELM	5.400	5	2	10	3.406
	ELM-BIP	5.200	4	2	10	3.155
	R-ELM/BIP	12.40	14	4	18	5.060
	RELM	9.800	11	2	18	5.203
	IRWLS-ELM	6.400	5	2	14	3.627

Source: author.

Table 41 – Number of hidden neurons with 2 sided contamination (CPU dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	8.800	10	4	12	2.860
	ELM-BIP	6.400	6	4	10	2.459
	R-ELM/BIP	19.20	19	10	30	7.315
	RELM	14	15	6	22	5.888
	IRWLS-ELM	13	12	8	26	5.518
20%	ELM	4.600	4	2	8	2.503
	ELM-BIP	6.200	6	2	12	2.898
	R-ELM/BIP	20.80	20	8	38	8.443
	RELM	14	12	6	22	6.394
	IRWLS-ELM	9.200	9	4	14	3.425
30%	ELM	5.200	5	2	10	2.860
	ELM-BIP	4.600	4	2	10	2.503
	R-ELM/BIP	16.40	16	4	28	8.422
	RELM	17.80	16	6	36	9.636
	IRWLS-ELM	9.600	10	2	16	3.864
40%	ELM	5	4	2	10	3.162
	ELM-BIP	4	3	2	8	2.309
	R-ELM/BIP	7.600	6	4	16	3.373
	RELM	9.400	10	2	16	3.777
	IRWLS-ELM	6.200	6	2	10	3.327

Source: author.

C.6 Servo dataset

Table 42 – Number of hidden neurons with 1 sided contamination (Servo dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	6.400	5	4	12	2.951
	ELM-BIP	16.40	18	10	22	4.600
	R-ELM/BIP	31.60	33	16	42	8.475
	RELM	22.40	23	12	34	7.106
	IRWLS-ELM	11	9	4	22	6.616
20%	ELM	6	5	4	14	3.127
	ELM-BIP	9.800	8	4	18	4.264
	R-ELM/BIP	21.40	20	16	32	5.420
	RELM	16.60	18	4	24	6.603
	IRWLS-ELM	7.800	6	4	16	4.566
30%	ELM	5.200	4	2	12	3.293
	ELM-BIP	9.200	9	2	18	5.903
	R-ELM/BIP	16.20	17	8	26	5.453
	RELM	11	11	4	22	5.831
	IRWLS-ELM	7.800	6	4	16	4.158
40%	ELM	5.800	4	2	22	5.846
	ELM-BIP	5.400	4	2	14	3.534
	R-ELM/BIP	7.800	7	2	18	5.203
	RELM	7.600	4	2	20	6.096
	IRWLS-ELM	7.800	6	4	16	4.662

Source: author.

Table 43 – Number of hidden neurons with 2 sided contamination (Servo dataset).

		Mean	**Median**	**Minimum**	**Maximum**	**Std.**
10%	ELM	9	6	4	22	6.815
	ELM-BIP	14.60	12	6	28	7.121
	R-ELM/BIP	31.40	29	20	50	9.891
	RELM	20	19	8	30	6.667
	IRWLS-ELM	10.20	8	4	26	7.451
20%	ELM	6	5	2	12	2.981
	ELM-BIP	8	8	4	14	3.266
	R-ELM/BIP	22.60	21	14	34	6.186
	RELM	18	19	10	28	6.394
	IRWLS-ELM	6.600	6	2	16	4.222
30%	ELM	6	4	2	18	4.522
	ELM-BIP	5.600	6	2	10	2.271
	R-ELM/BIP	15.40	17	4	24	6.328
	RELM	11.20	9	2	22	6.125
	IRWLS-ELM	8.200	7	4	18	3.938
40%	ELM	4.200	4	2	6	1.135
	ELM-BIP	4.800	4	2	8	2.150
	R-ELM/BIP	8.200	8	2	16	4.566
	RELM	9.200	9	2	16	4.917
	IRWLS-ELM	4.400	4	2	10	2.066

Source: author.

APPENDIX D – RESULTS OF ANHNA'S VARIANTS

This appendix is dedicated to present all results related to ANHNA with Differential Evolution and its variants performance using the following datasets: Abalone, Auto-MPG, Body Fat, Boston Housing, Breast Cancer, CPU, iCub and Servo (more details in Appendix B). We also present here ANHNA's performance using different metaheuristics (Section D.2) where the variants were also applied. It is important to highlight that ANHNA using Self-Adaptive Differential Evolution does not have regularization variants versions for this thesis.

D.1 ANHNA with Differential Evolution

Below we present the evaluation of all variants of ANHNA with Differential Evolution, as described in Section 4.4.1. For more details over this metaheuristic, see Appendix A.1). One of the figures presents the RMSE from training and test in the 10 repetitions (see details of the test methodology in Chapter 5), where the black dots represent the mean value, and the other figure shows the number of hidden neurons chosen by its respective ANHNA in those same repetitions. The tables provide the information about the behavior (mean, median, minimum value, maximum value and standard deviation) of the output weight norms given by the architecture and parameters provided by ANHNA.

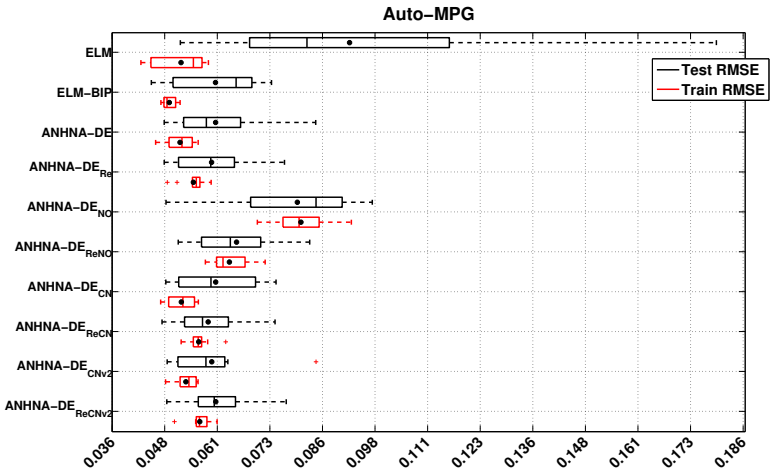
From this information, we elect the version that presents itself with the best RMSE performance, least number of hidden neurons chosen and the smaller average of weight norms in most of the tested datasets, to represent ANHNA in the main results of this thesis.

It is also important to highlight that these tests were made with no addition of noise besides the one already contained in the set itself.

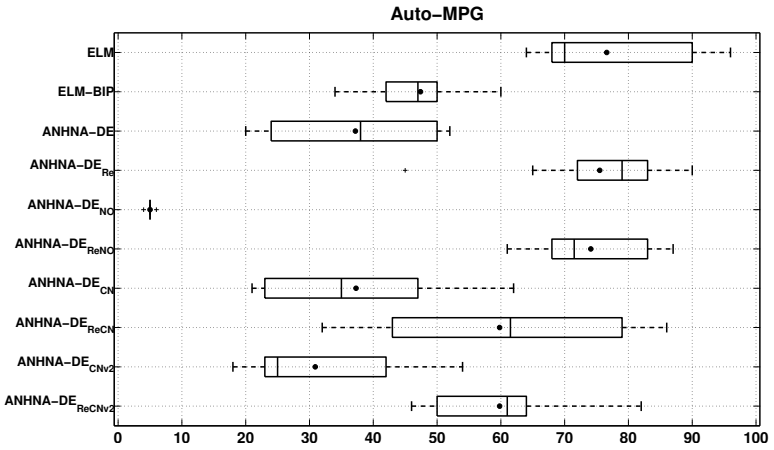
D.1.1 Auto-MPG

Figure 44 – ANHNA-DE's variants for Auto-MPG dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 44 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (Auto-MPG dataset).

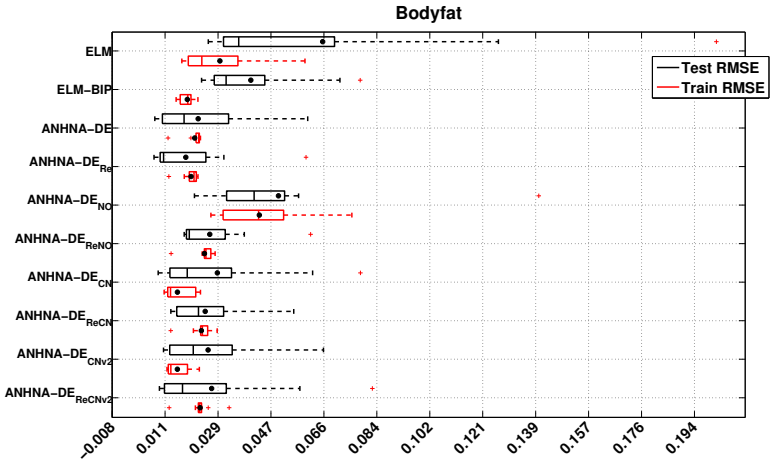
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	2.411×10^{4}	1.273×10^{4}	5.878×10^{3}	6.912×10^{4}	2.313×10^{4}
ELM-BIP	4.559	3.714	2.663	9.287	2.177
ANHNA-DE	1.478e+09	3.108×10^{4}	3.160×10^{2}	1.478×10^{10}	4.673e+09
ANHNA-DE$_{Re}$	93.65	16.30	8.554	7.832×10^{2}	2.424×10^{2}
ANHNA-DE$_{NO}$	2.036	1.711	6.635×10^{-1}	4.056	1.082
ANHNA-DE$_{ReNO}$	6.497×10^{-1}	6.424×10^{-1}	4.559×10^{-1}	8.617×10^{-1}	1.517×10^{-1}
ANHNA-DE$_{CN}$	6.856×10^{8}	4.450×10^{4}	8.484×10^{2}	3.515e+09	1.403e+09
ANHNA-DE$_{ReCN}$	25.83	19.40	7.569	92.02	24.49
ANHNA-DE$_{CNv2}$	4.110×10^{5}	7.117×10^{3}	3.694×10^{2}	3.751×10^{6}	1.176×10^{6}
ANHNA-DE$_{ReCNv2}$	46.86	35.57	12.07	1.979×10^{2}	54.17

Source: author.

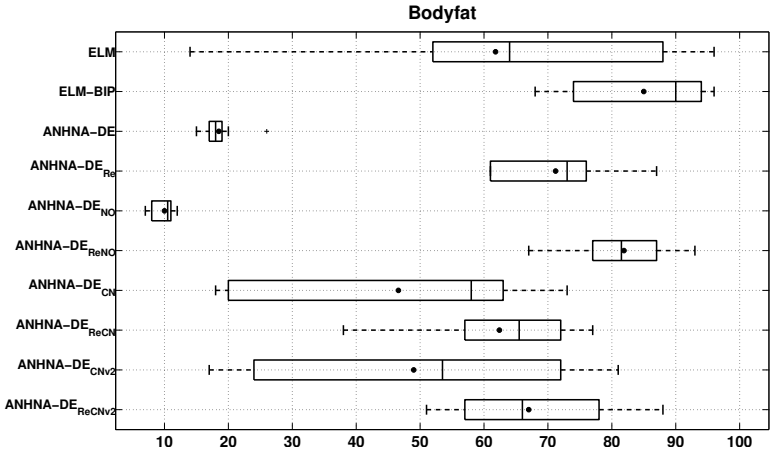
D.1.2 Bodyfat

Figure 45 – ANHNA-DE's variants for Bodyfat dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 45 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (Bodyfat dataset).

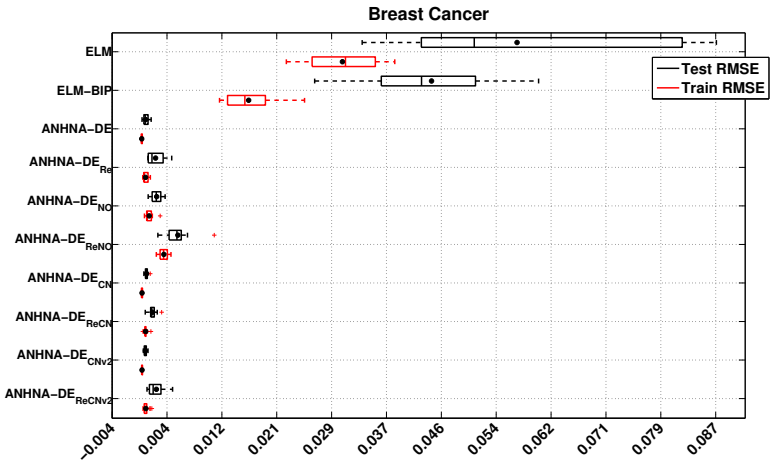
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	5.505×10^{2}	5.104×10^{2}	13.36	1.051×10^{3}	3.593×10^{2}
ELM-BIP	1.038	1.020	7.538×10^{-1}	1.249	1.425×10^{-1}
ANHNA-DE	1.302×10^{4}	2.915×10^{2}	24.77	1.258×10^{5}	3.962×10^{4}
ANHNA-DE$_{Re}$	5.225	4.047	2.551	14.82	3.674
ANHNA-DE$_{NO}$	5.260	5.073	1.837	9.490	2.620
ANHNA-DE$_{ReNO}$	1.052	1.032	7.651×10^{-1}	1.284	1.490×10^{-1}
ANHNA-DE$_{CN}$	4.942×10^{6}	3.495×10^{4}	25.10	4.848×10^{7}	1.530×10^{7}
ANHNA-DE$_{ReCN}$	3.087	3.266	1.303	4.435	1.075
ANHNA-DE$_{CNv2}$	7.014×10^{6}	2.089×10^{4}	1.172×10^{2}	6.909×10^{7}	2.181×10^{7}
ANHNA-DE$_{ReCNv2}$	2.472	2.467	1.425	3.106	5.445×10^{-1}

Source: author.

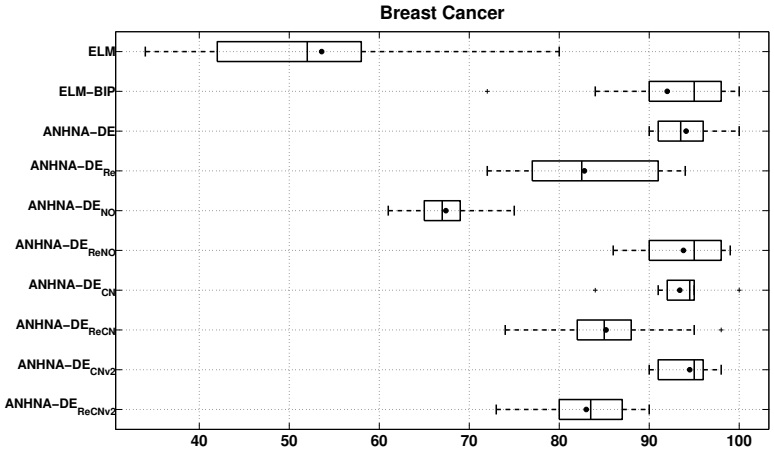
D.1.3 Breast Cancer

Figure 46 – ANHNA-DE's variants for Breast Cancer dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 46 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (Breast Cancer dataset).

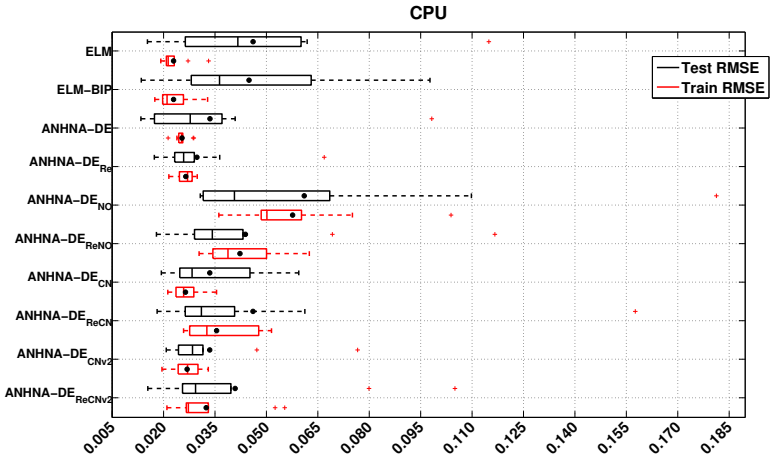
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	23.16	21.72	10.60	40.06	10.24
ELM-BIP	6.433×10^{-1}	6.251×10^{-1}	5.411×10^{-1}	7.950×10^{-1}	7.899×10^{-2}
ANHNA-DE	20.91	15.64	5.509	43.22	13.39
ANHNA-DE$_{Re}$	5.441	4.693	3.339	9.504	2.051
ANHNA-DE$_{NO}$	5.218	5.299	4.092	6.492	8.314×10^{-1}
ANHNA-DE$_{ReNO}$	1.870	1.852	1.511	2.420	2.449×10^{-1}
ANHNA-DE$_{CN}$	41.20	16.38	5.979	1.346×10^{2}	46.11
ANHNA-DE$_{ReCN}$	5.599	4.442	2.417	10.91	2.828
ANHNA-DE$_{CNv2}$	13.06	10.30	7.537	34.08	8.039
ANHNA-DE$_{ReCNv2}$	8.617	7.623	4.179	16.35	3.530

Source: author.

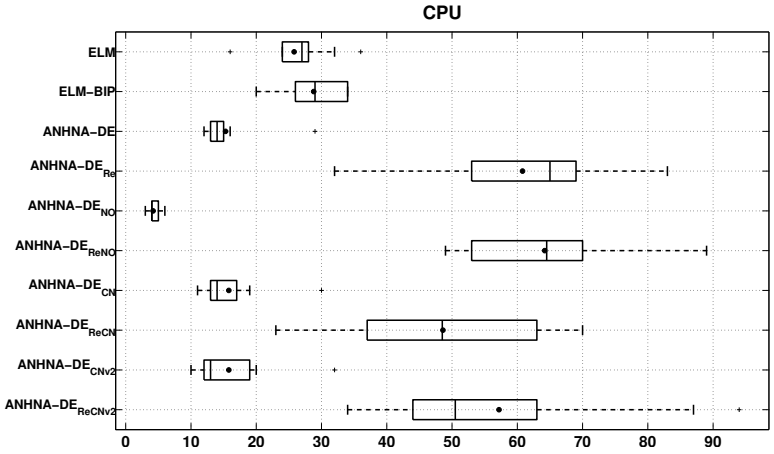
D.1.4 CPU

Figure 47 – ANHNA-DE's variants for CPU dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 47 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (CPU dataset).

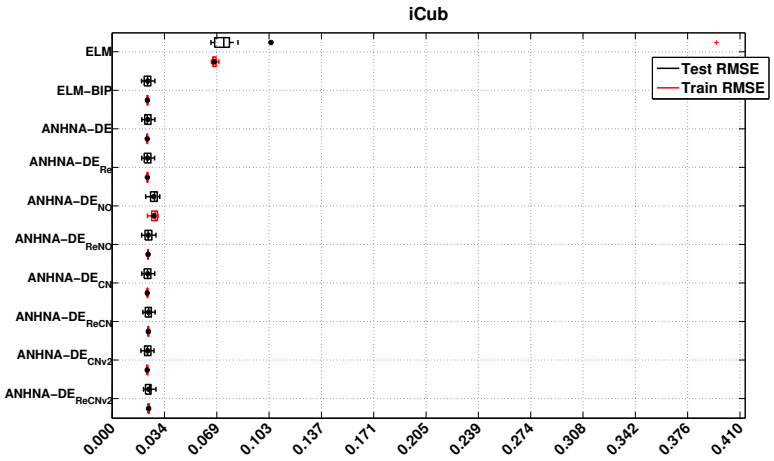
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	1.665×10^{3}	1.438×10^{3}	7.715×10^{2}	3.295×10^{3}	7.652×10^{2}
ELM-BIP	1.437	1.380	6.323×10^{-1}	2.093	4.048×10^{-1}
ANHNA-DE	2.072×10^{5}	2.866×10^{3}	3.543×10^{2}	1.869×10^{6}	5.865×10^{5}
ANHNA-DE$_{Re}$	18.21	13.24	8.083	54.47	13.73
ANHNA-DE$_{NO}$	2.746	2.647	8.141×10^{-1}	5.481	1.480
ANHNA-DE$_{ReNO}$	7.166×10^{-1}	6.000×10^{-1}	3.719×10^{-1}	1.323	3.116×10^{-1}
ANHNA-DE$_{CN}$	8.244×10^{4}	2.110×10^{3}	84.14	5.924×10^{5}	1.884×10^{5}
ANHNA-DE$_{ReCN}$	13.93	8.772	1.610	37.80	12.94
ANHNA-DE$_{CNv2}$	2.703×10^{4}	4.051×10^{3}	2.391×10^{2}	1.341×10^{5}	4.646×10^{4}
ANHNA-DE$_{ReCNv2}$	8.852	6.499	1.394	22.57	6.733

Source: author.

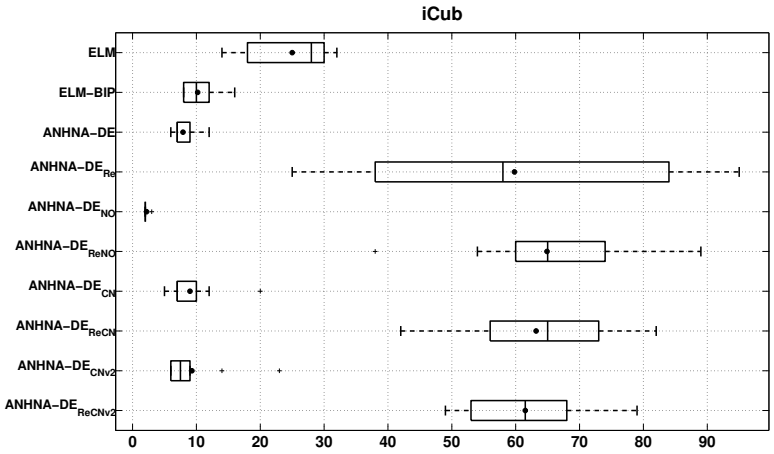
D.1.5 iCub

Figure 48 – ANHNA-DE's variants for iCub dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 48 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (iCub dataset).

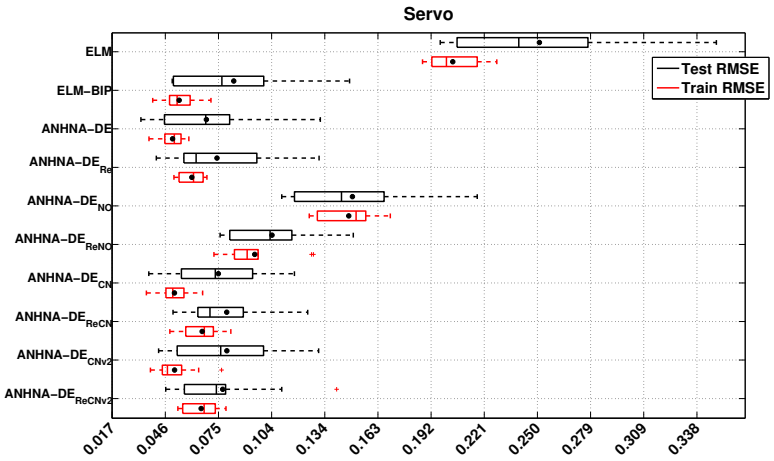
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	4.488×10^{7}	3.455×10^{7}	4.594×10^{5}	1.095×10^{8}	4.495×10^{7}
ELM-BIP	1.954	1.837	5.483×10^{-1}	4.052	1.127
ANHNA-DE	2.530×10^{4}	3.058×10^{2}	72.00	2.498×10^{5}	7.889×10^{4}
ANHNA-DE$_{Re}$	6.486	5.255	9.970×10^{-1}	16.48	4.779
ANHNA-DE$_{NO}$	5.210×10^{-1}	5.276×10^{-1}	2.452×10^{-1}	9.002×10^{-1}	2.022×10^{-1}
ANHNA-DE$_{ReNO}$	7.816×10^{-2}	7.560×10^{-2}	5.586×10^{-2}	1.103×10^{-1}	1.497×10^{-2}
ANHNA-DE$_{CN}$	1.567×10^{5}	5.499×10^{2}	39.37	1.232×10^{6}	3.904×10^{5}
ANHNA-DE$_{ReCN}$	1.263×10^{-1}	1.090×10^{-1}	8.636×10^{-2}	1.911×10^{-1}	3.558×10^{-2}
ANHNA-DE$_{CNv2}$	3.659×10^{8}	3.642×10^{2}	21.95	3.658e+09	1.157e+09
ANHNA-DE$_{ReCNv2}$	7.026×10^{-2}	7.513×10^{-2}	5.234×10^{-2}	8.134×10^{-2}	1.042×10^{-2}

Source: author.

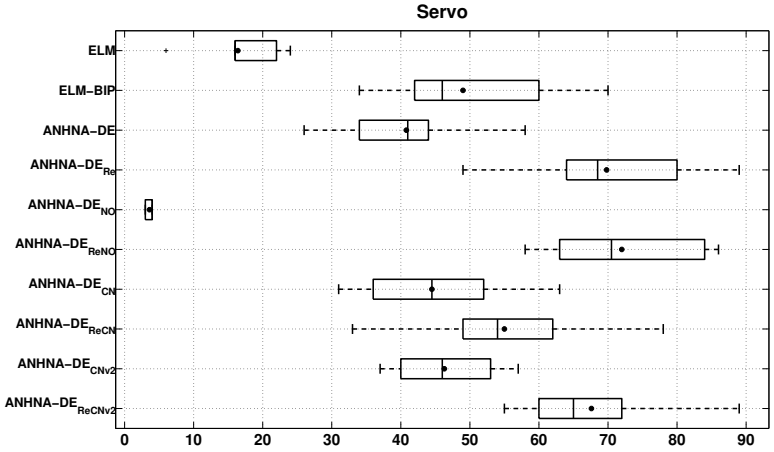
D.1.6 Servo

Figure 49 – ANHNA-DE's variants for Servo dataset with DE.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 49 – ANHNA-DE's comparison study of euclidean norm of output weight vectors (Servo dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	2.154×10^{4}	3.522×10^{3}	1.286×10^{2}	1.578×10^{5}	4.875×10^{4}
ELM-BIP	12.95	10.74	5.546	32.43	8.349
ANHNA-DE	7.187e+09	1.435e+09	1.811×10^{5}	3.002×10^{10}	1.150×10^{10}
ANHNA-DE$_{Re}$	2.597×10^{3}	1.068×10^{3}	61.36	1.519×10^{4}	4.502×10^{3}
ANHNA-DE$_{NO}$	1.561	1.342	5.898×10^{-1}	2.640	6.969×10^{-1}
ANHNA-DE$_{ReNO}$	1.315	1.274	8.087×10^{-1}	2.332	4.468×10^{-1}
ANHNA-DE$_{CN}$	1.601×10^{10}	3.487e+09	1.639×10^{5}	1.011×10^{11}	3.121×10^{10}
ANHNA-DE$_{ReCN}$	3.504×10^{3}	4.397×10^{2}	53.69	1.383×10^{4}	5.553×10^{3}
ANHNA-DE$_{CNv2}$	9.740e+09	2.607e+09	1.035×10^{6}	4.925×10^{10}	1.641×10^{10}
ANHNA-DE$_{ReCNv2}$	2.759×10^{3}	7.341×10^{2}	86.11	1.874×10^{4}	5.759×10^{3}

Source: author.

D.2 ANHNA with different metaheuristics

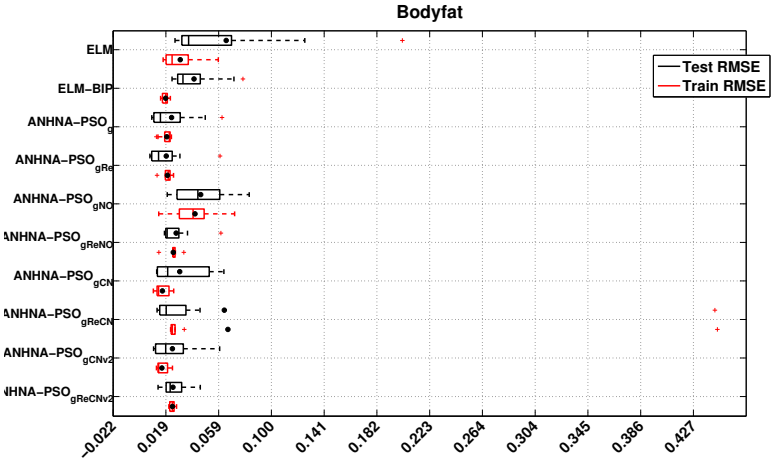
In this section, we present ANHNA and its variants' performance using different metaheuristics, such as: Particle Swarm Optimization using global (Subsection D.2.1) and local (Subsection D.2.2) neighborhood, and also with Self-Adaptive Differential Evolution (Subsection D.2.3). We present the results of only two datasets for illustration purposes. More details over these metaheuristics, see Appendix A.3 and Appendix A.2.

D.2.1 Variations with ANHNA-PSO$_g$

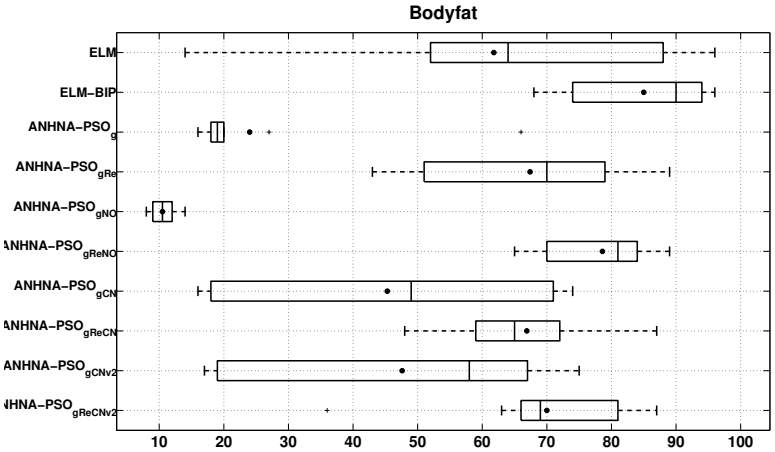
D.2.1.1 Bodyfat

Figure 50 – ANHNA's variants for Bodyfat dataset with PSO$_g$.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 50 – ANHNA-PSO$_g$'s comparison study of euclidean norm of output weight vectors (Bodyfat dataset).

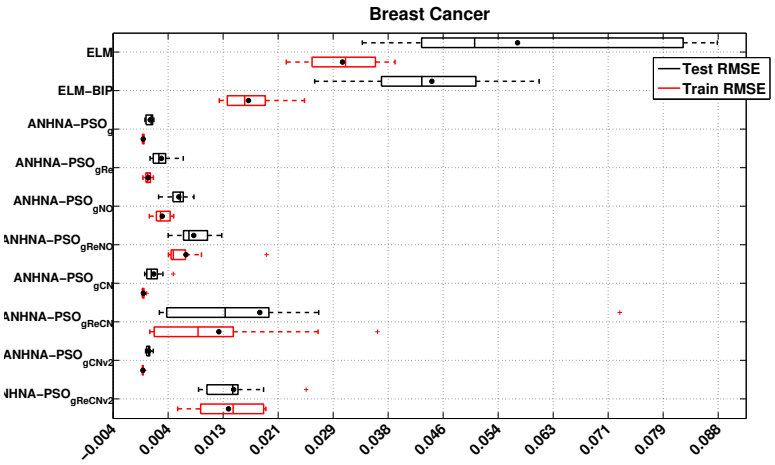
	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	5.505×10^{2}	5.104×10^{2}	13.36	1.051×10^{3}	3.593×10^{2}
ELM-BIP	1.038	1.020	7.538×10^{-1}	1.249	1.425×10^{-1}
ANHNA-PSO$_{g}$	1.222×10^{3}	3.121×10^{2}	14.99	9.297×10^{3}	2.856×10^{3}
ANHNA-PSO$_{gRe}$	4.713	3.065	2.392	15.46	3.949
ANHNA-PSO$_{gNO}$	4.594	4.436	2.256	6.470	1.532
ANHNA-PSO$_{gReNO}$	1.055	1.030	7.140×10^{-1}	1.370	1.983×10^{-1}
ANHNA-PSO$_{gCN}$	1.388×10^{5}	1.728×10^{3}	62.17	1.329×10^{6}	4.181×10^{5}
ANHNA-PSO$_{gReCN}$	1.681	1.867	1.053e-32	2.215	6.613×10^{-1}
ANHNA-PSO$_{gCNv2}$	8.105×10^{3}	2.050×10^{3}	27.60	4.079×10^{4}	1.417×10^{4}
ANHNA-PSO$_{gReCNv2}$	2.195	2.138	1.580	3.681	5.749×10^{-1}

Source: author.

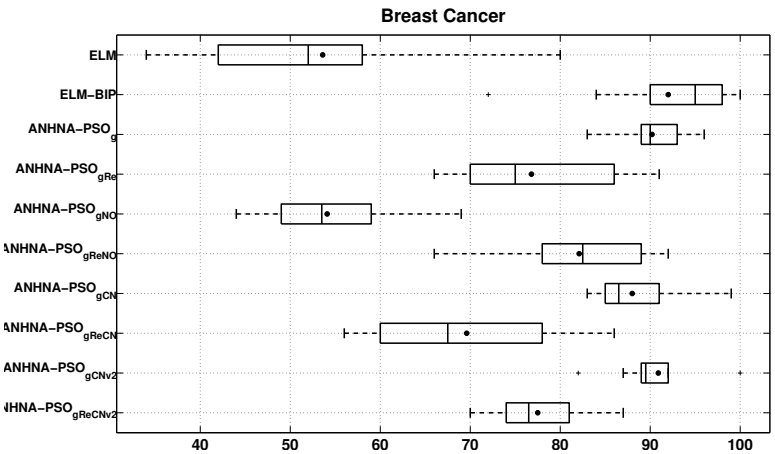
D.2.1.2 Breast Cancer

Figure 51 – ANHNA's variants for Breast Cancer dataset with PSO$_g$.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

Table 51 – ANHNA-PSO$_g$'s comparison study of euclidean norm of output weight vectors (Breast Cancer dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	23.16	21.72	10.60	40.06	10.24
ELM-BIP	6.433×10^{-1}	6.251×10^{-1}	5.411×10^{-1}	7.950×10^{-1}	7.899×10^{-2}
ANHNA-PSO$_g$	42.15	33.80	10.75	1.027×10^{2}	28.54
ANHNA-PSO$_{gRe}$	6.921	6.646	3.661	9.651	1.768
ANHNA-PSO$_{gNO}$	4.650	4.858	3.076	5.269	7.153×10^{-1}
ANHNA-PSO$_{gReNO}$	2.591	2.006	1.639	7.815	1.858
ANHNA-PSO$_{gCN}$	55.79	41.64	10.93	1.248×10^{2}	44.89
ANHNA-PSO$_{gReCN}$	3.846	3.076	2.440	7.725	1.655
ANHNA-PSO$_{gCNv2}$	41.00	33.88	10.18	1.113×10^{2}	31.96
ANHNA-PSO$_{gReCNv2}$	2.479	2.479	1.951	2.967	2.929×10^{-1}

Source: author.

D.2.2 Variants with ANHNA-PSO$_l$

D.2.2.1 Bodyfat

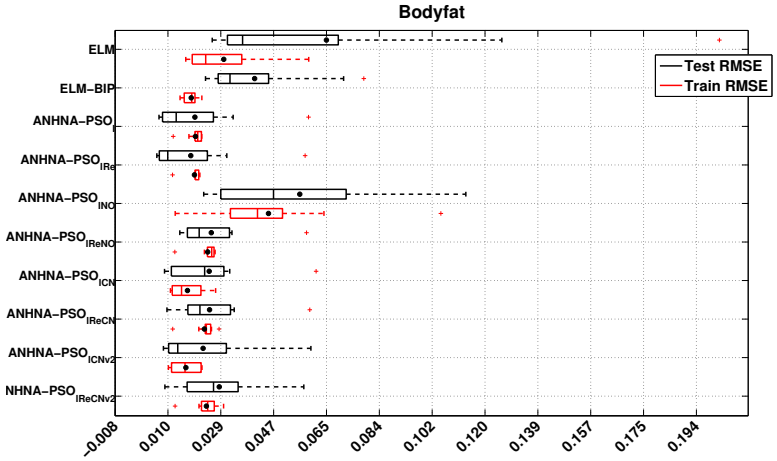
Table 52 – ANHNA-PSO$_l$'s comparison study of euclidean norm of output weight vectors (Bodyfat dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	5.505×10^{2}	5.104×10^{2}	13.36	1.051×10^{3}	3.593×10^{2}
ELM-BIP	1.038	1.020	7.538×10^{-1}	1.249	1.425×10^{-1}
ANHNA-PSO$_l$	76.79	39.20	17.70	4.242×10^{2}	1.232×10^{2}
ANHNA-PSO$_{lRe}$	3.787	3.271	1.705	7.262	1.891
ANHNA-PSO$_{lNO}$	5.386	5.354	2.341	9.331	2.487
ANHNA-PSO$_{lReNO}$	1.079	1.126	8.135×10^{-1}	1.261	1.308×10^{-1}
ANHNA-PSO$_{lCN}$	3.757×10^{5}	2.518×10^{3}	56.59	3.208×10^{6}	1.009×10^{6}
ANHNA-PSO$_{lReCN}$	2.226	2.317	1.312	3.830	6.818×10^{-1}
ANHNA-PSO$_{lCNv2}$	4.512×10^{3}	4.411×10^{2}	20.71	3.797×10^{4}	1.181×10^{4}
ANHNA-PSO$_{lReCNv2}$	2.269	2.279	1.529	3.879	7.200×10^{-1}

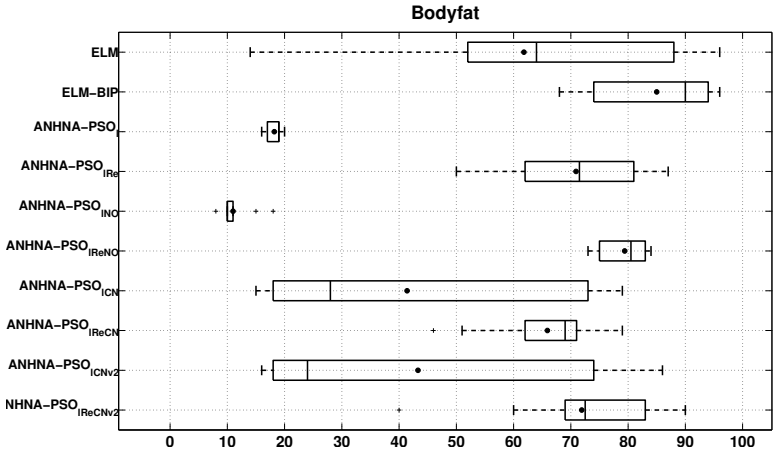
Source: author.

Figure 52 – ANHNA's variants for Bodyfat dataset with PSO_l.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

D.2.2.2 Breast Cancer

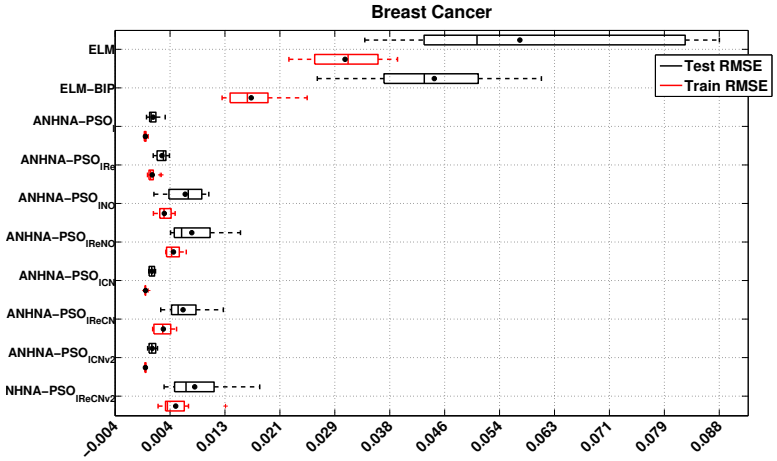
Table 53 – ANHNA-PSO_l's comparison study of euclidean norm of output weight vectors (Breast Cancer dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	23.16	21.72	10.60	40.06	10.24
ELM-BIP	6.433×10^{-1}	6.251×10^{-1}	5.411×10^{-1}	7.950×10^{-1}	7.899×10^{-2}
ANHNA-PSO_l	89.30	72.41	27.45	2.356×10^{2}	59.79
ANHNA-PSO_{lRe}	7.002	6.722	4.753	9.506	1.529
ANHNA-PSO_{lNO}	6.247	5.995	4.861	8.911	1.230
ANHNA-PSO_{lReNO}	1.951	1.958	1.671	2.289	2.128×10^{-1}
ANHNA-PSO_{lCN}	86.99	74.05	37.61	2.285×10^{2}	56.55
ANHNA-PSO_{lReCN}	4.528	3.348	2.361	8.237	2.034
ANHNA-PSO_{lCNv2}	92.81	72.42	30.28	2.108×10^{2}	68.70
ANHNA-$PSO_{lReCNv2}$	5.761	5.180	3.494	11.02	2.436

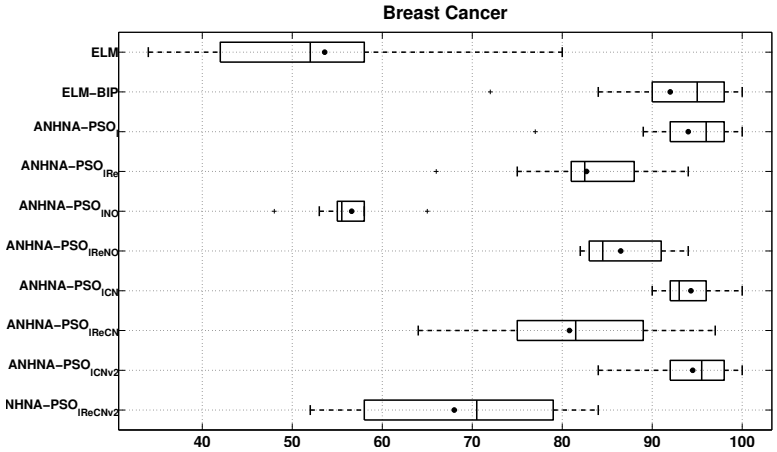
Source: author.

Figure 53 – ANHNA's variants for Breast Cancer dataset with PSO_l.

(a) Training and Testing RMSE Results.



(b) Number of hidden neurons.



Source: author.

D.2.3 Variants with ANHNA-SADE

D.2.3.1 Bodyfat

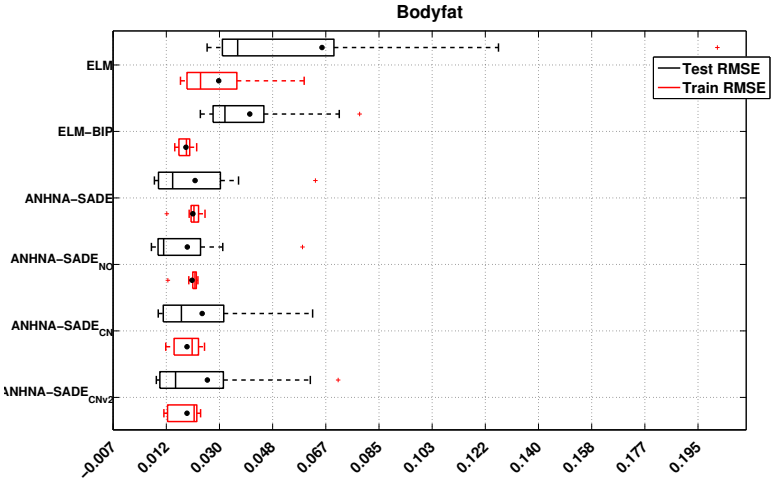
Table 54 – ANHNA's comparison study of euclidean norm of output weight vectors (Bodyfat dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	5.505×10^{2}	5.104×10^{2}	13.36	1.051×10^{3}	3.593×10^{2}
ELM-BIP	1.038	1.020	7.538×10^{-1}	1.249	1.425×10^{-1}
ANHNA-SADE	45.70	40.22	16.05	80.37	23.06
ANHNA-SADE$_{NO}$	42.20	31.42	10.35	1.251×10^{2}	36.73
ANHNA-SADE$_{CN}$	5.915×10^{2}	57.44	12.67	4.149×10^{3}	1.284×10^{3}
ANHNA-SADE$_{CNv2}$	1.864×10^{3}	54.57	27.25	9.271×10^{3}	3.785×10^{3}

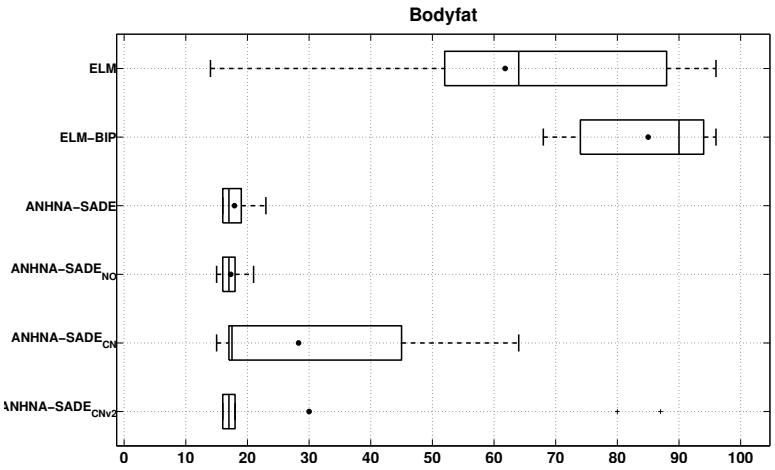
Source: author.

Figure 54 – ANHNA's variants for Bodyfat dataset with SADE.

(a) Training and Test RMSE Results.



(b) Number of hidden neurons.



Source: author.

D.2.3.2 Breast Cancer

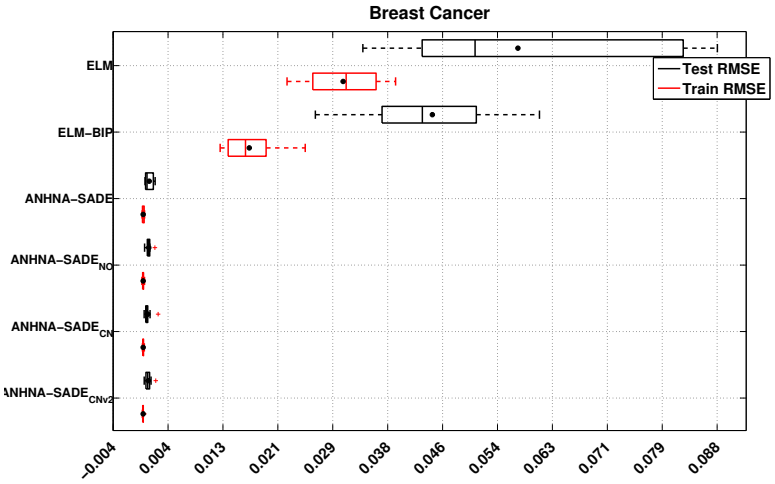
Table 55 – ANHNA's comparison study of euclidean norm of output weight vectors (Breast Cancer dataset).

	Mean	**Median**	**Minimum**	**Maximum**	**Std.**
ELM	23.16	21.72	10.60	40.06	10.24
ELM-BIP	6.433×10^{-1}	6.251×10^{-1}	5.411×10^{-1}	7.950×10^{-1}	7.899×10^{-2}
ANHNA-SADE	18.59	6.211	3.518	1.052×10^{2}	31.71
ANHNA-SADE$_{NO}$	45.05	8.421	3.829	2.398×10^{2}	73.41
ANHNA-SADE$_{CN}$	5.850	5.668	4.498	8.052	1.240
ANHNA-SADE$_{CNv2}$	6.057	4.947	4.376	13.33	2.788

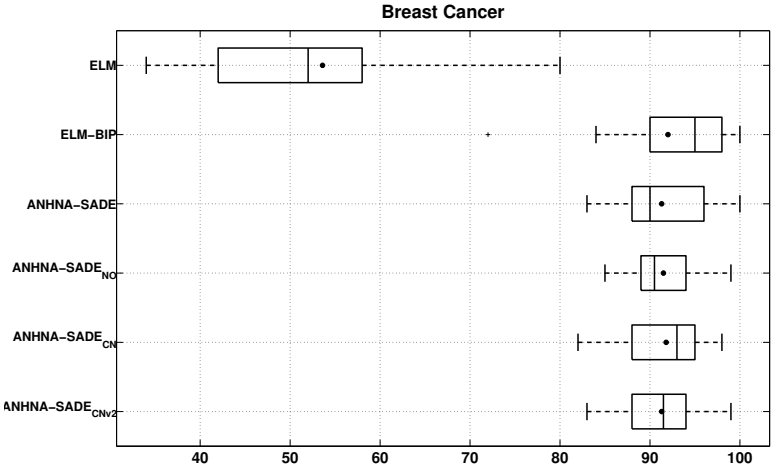
Source: author.

Figure 55 – ANHNA's variants for Breast Cancer dataset with SADE.

(a) Training and Test RMSE Results.



(b) Number of hidden neurons.



Source: author.

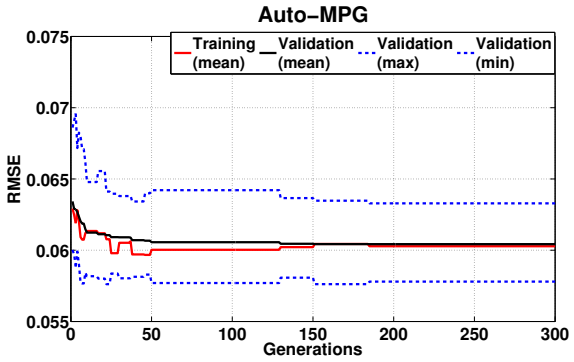
APPENDIX E – ANHNA'S CONVERGENCE

In this appendix, the convergence graphics for the chosen variant ANHNA-DE$_{Re}$ are presented for each of the ten repetitions performed, as detailed in Chapter 5. Considering a 5-Fold internal cross-validation within ANHNA, each graph shows the evolution of the best solution provided by the metaheuristic. The mean training RMSE is represented in a red line, while the mean validation RMSE, used as the fitness function, is given by a black one. The minimum and maximum validation RMSE of the best solution are represented by a dashed blue line.

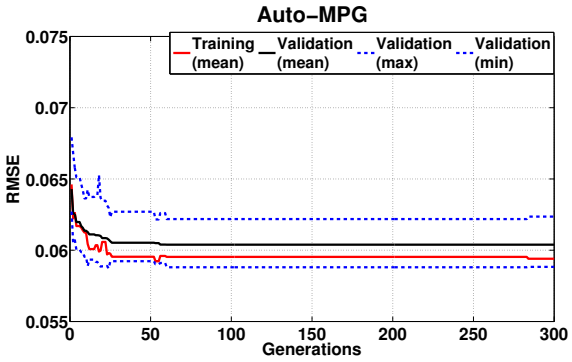
E.1 Auto-MPG

Figure 56 – ANHNA-DE$_{Re}$ convergence in 4 of 10 repetitions (Auto-MPG dataset).

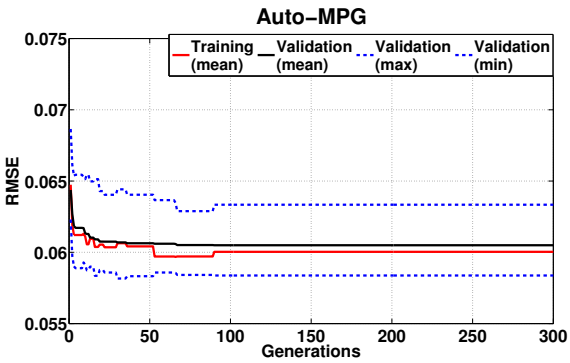
(a) Repetition 1



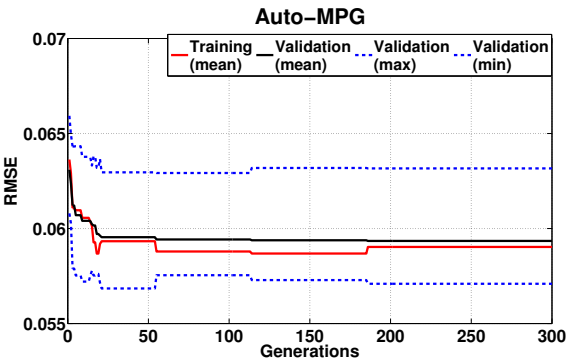
(b) Repetition 2



(c) Repetition 3



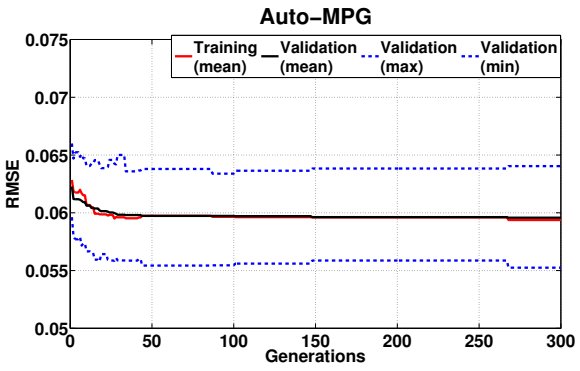
(d) Repetition 4



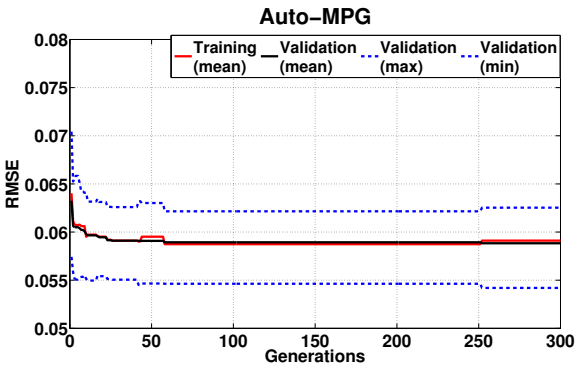
Source: author.

Figure 57 – ANHNA-DE$_{Re}$ convergence of 5 to 10 from 10 repetitions (Auto-MPG dataset).

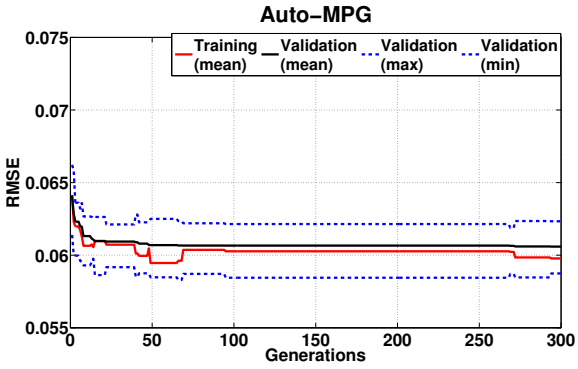
(a) Repetition 5



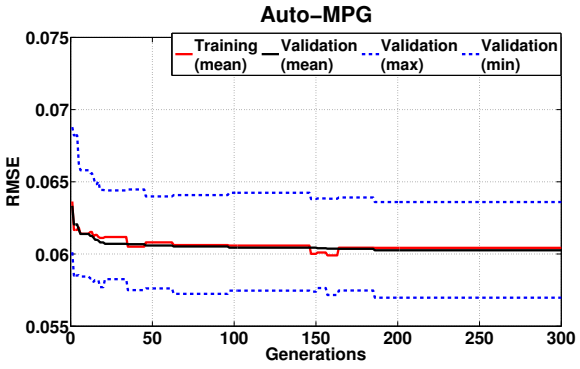
(b) Repetition 6



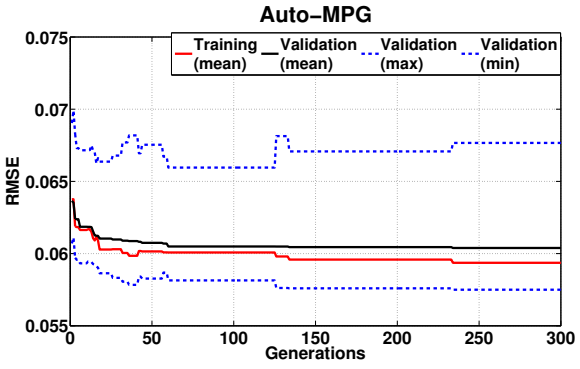
(c) Repetition 7



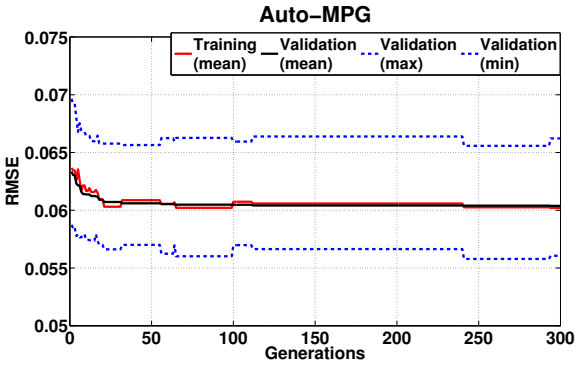
(d) Repetition 8



(e) Repetition 9



(f) Repetition 10

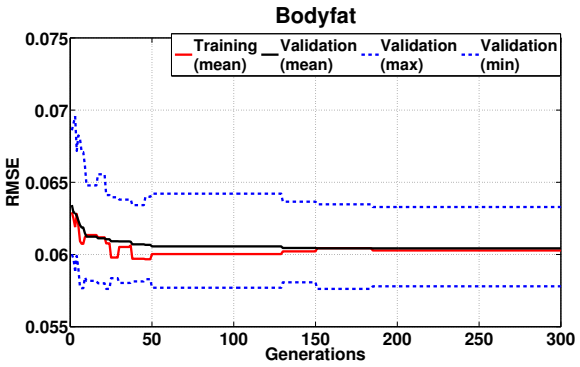


Source: author.

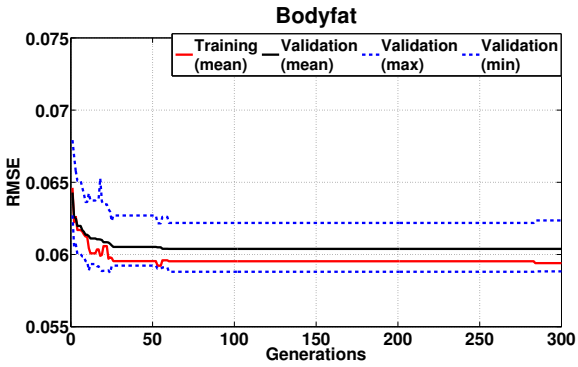
E.2 Bodyfat

Figure 58 – ANHNA-DE$_{Re}$ convergence of 1 to 6 of 10 repetitions (Bodyfat dataset).

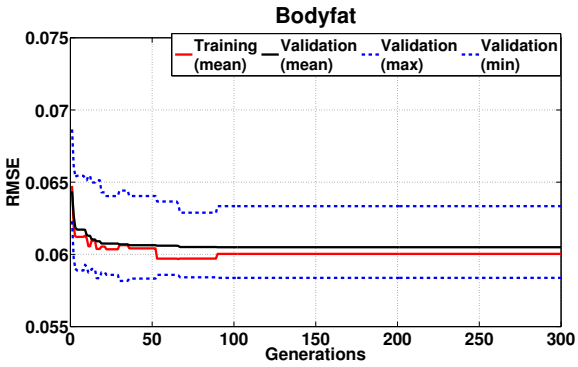
(a) Repetition 1



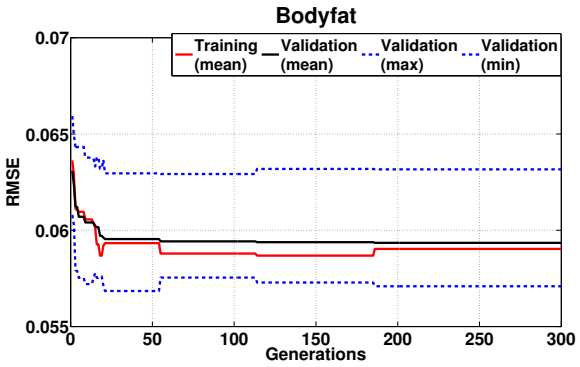
(b) Repetition 2



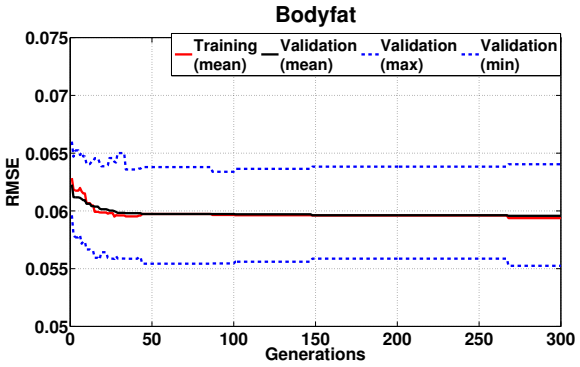
(c) Repetition 3



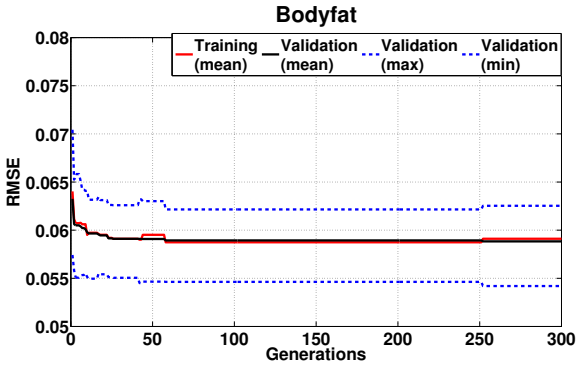
(d) Repetition 4



(e) Repetition 5



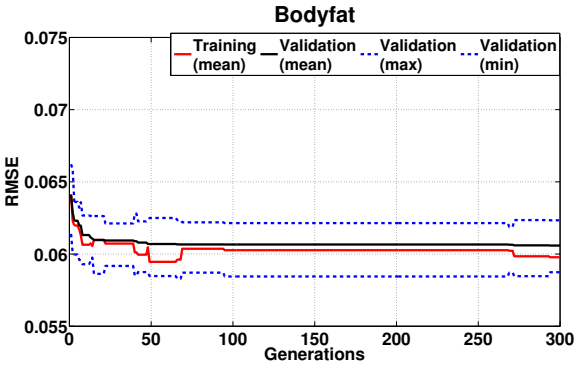
(f) Repetition 6



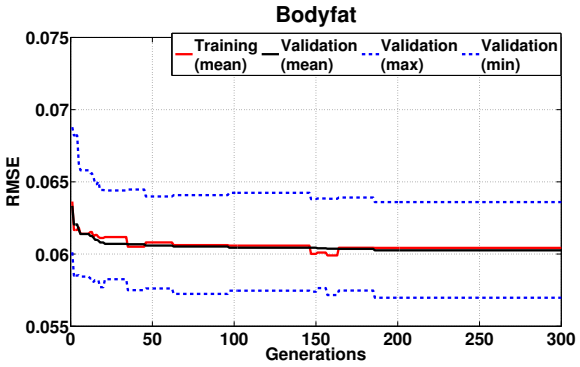
Source: author.

Figure 59 – ANHNA-DE$_{Re}$ convergence of 9 to 10 from 10 repetitions (Bodyfat dataset).

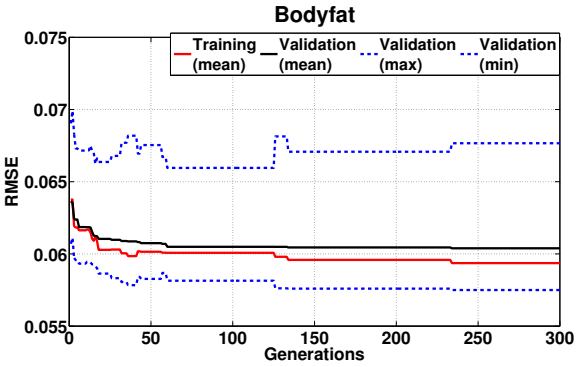
(a) Repetition 7



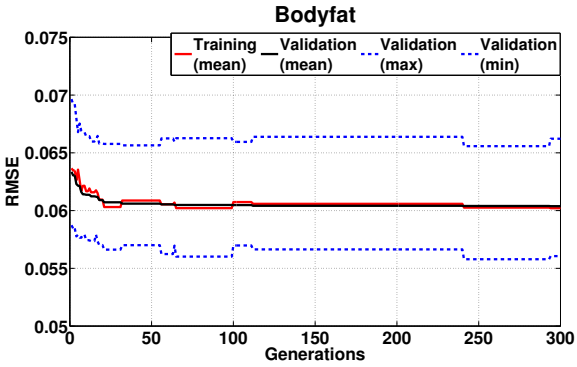
(b) Repetition 8



(c) Repetition 9



(d) Repetition 10

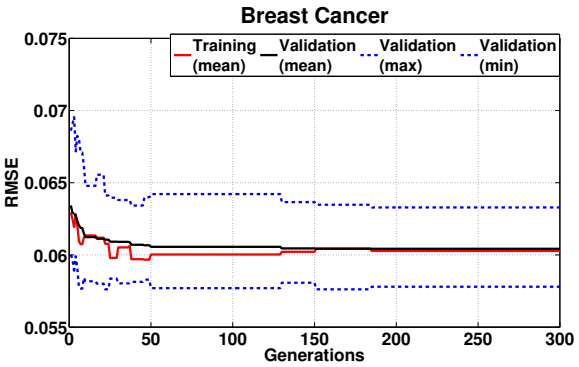


Source: author.

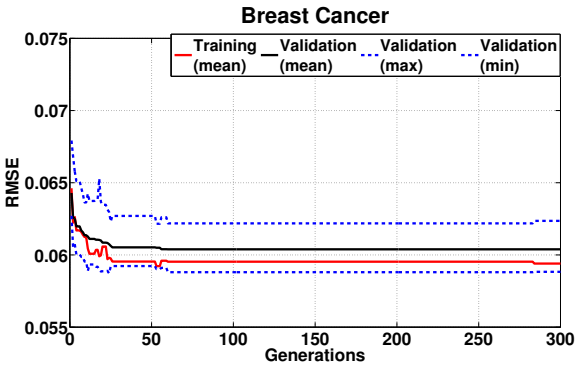
E.3 Breast Cancer

Figure 60 – ANHNA-DE$_{Re}$ convergence in 2 of 10 repetitions (Breast Cancer dataset).

(a) Repetition 1



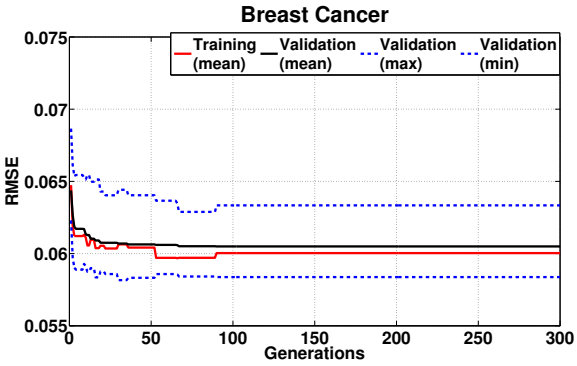
(b) Repetition 2



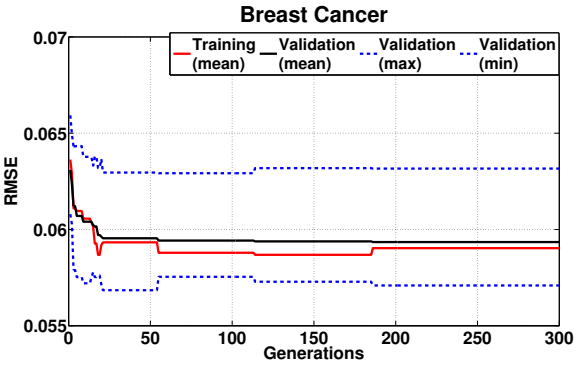
Source: author.

Figure 61 – ANHNA-DE$_{Re}$ convergence of 3 to 10 from 10 repetitions (Breast Cancer dataset).

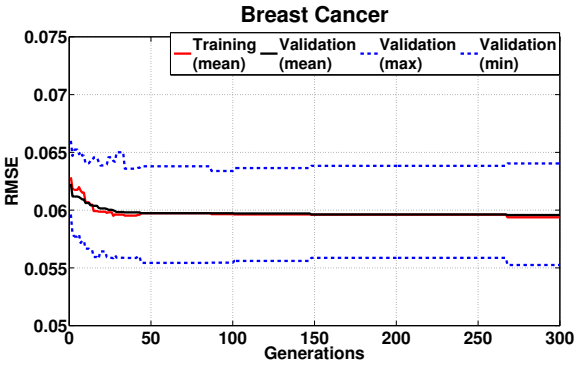
(a) Repetition 3



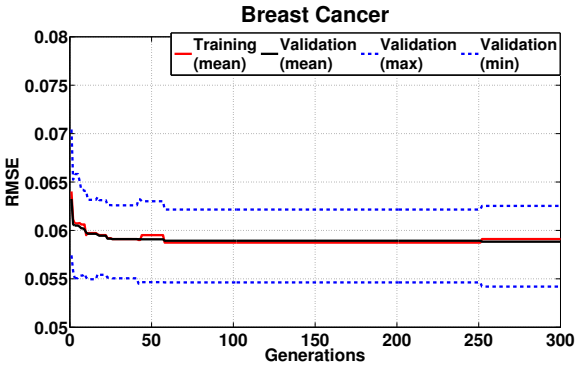
(b) Repetition 4



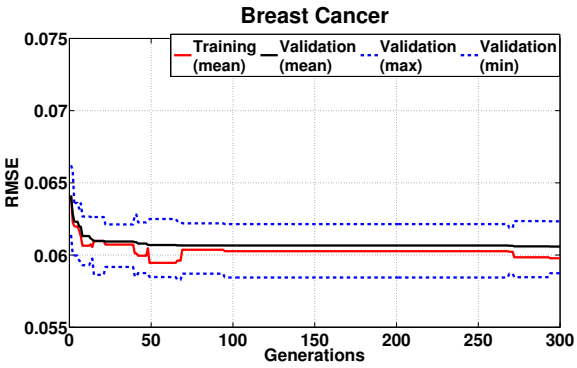
(c) Repetition 5



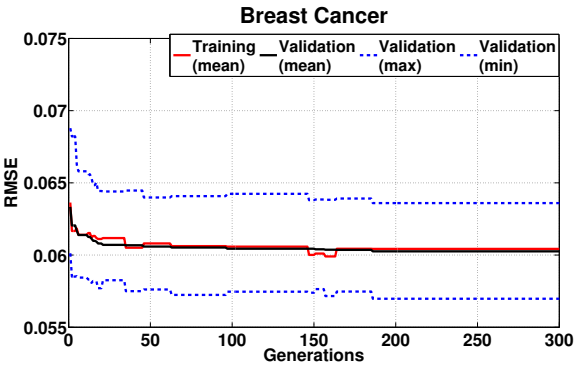
(d) Repetition 6



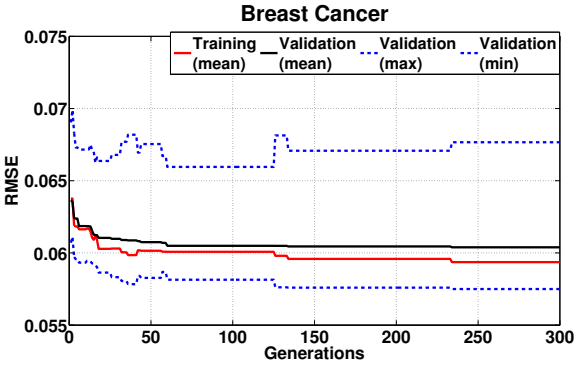
(e) Repetition 7



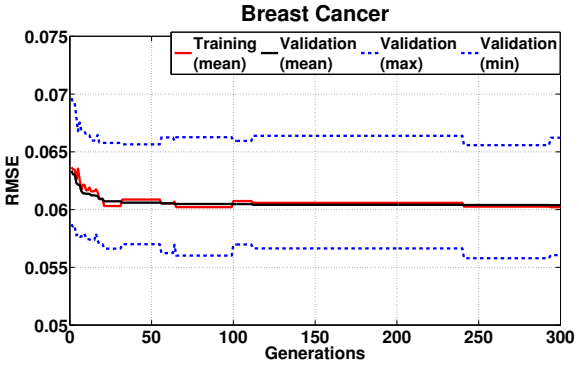
(f) Repetition 8



(g) Repetition 9



(h) Repetition 10

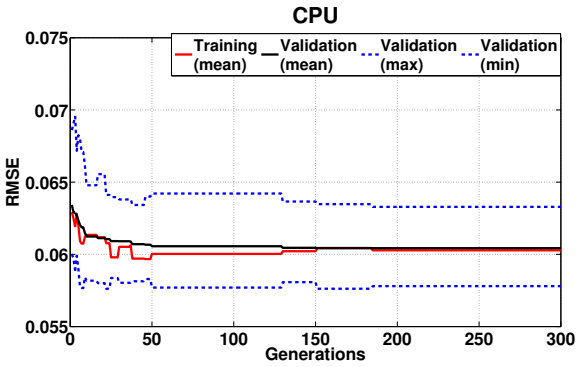


Source: author.

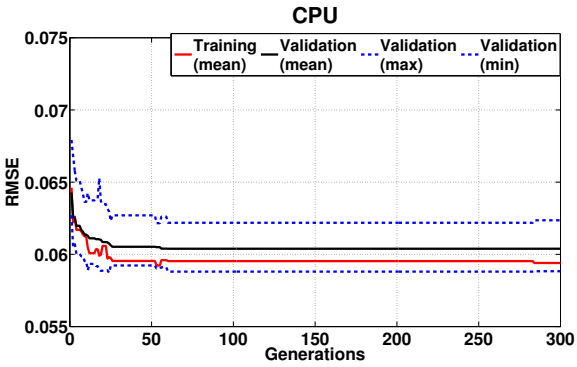
E.4 CPU

Figure 62 – ANHNA-DE$_{Re}$ convergence in 8 of 10 repetitions (CPU dataset).

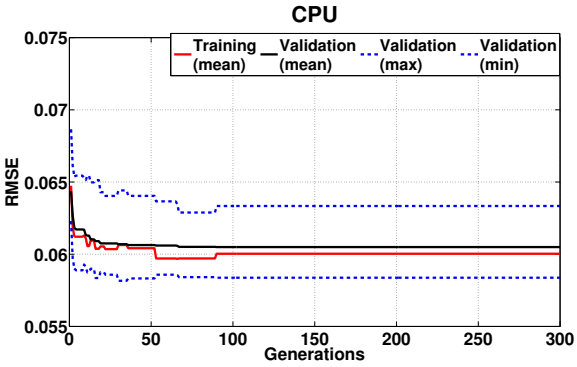
(a) Repetition 1



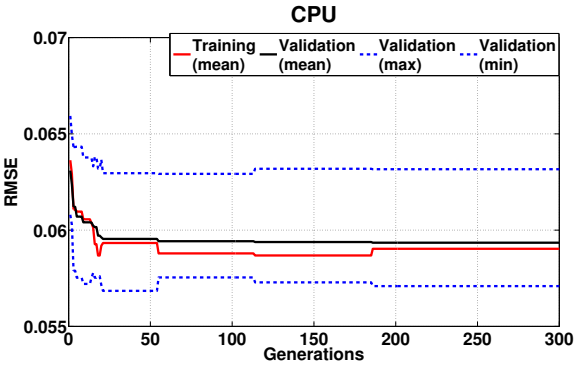
(b) Repetition 2



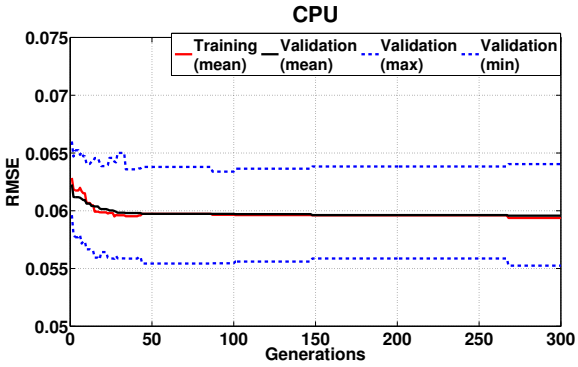
(c) Repetition 3



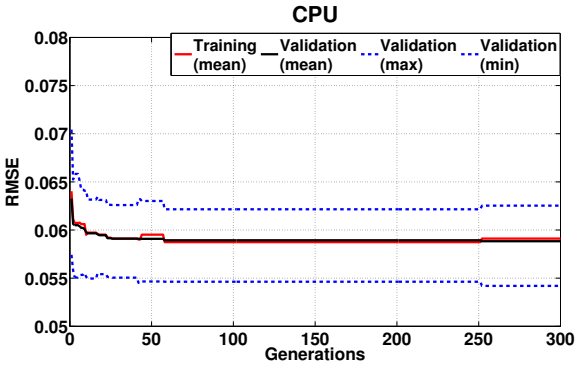
(d) Repetition 4



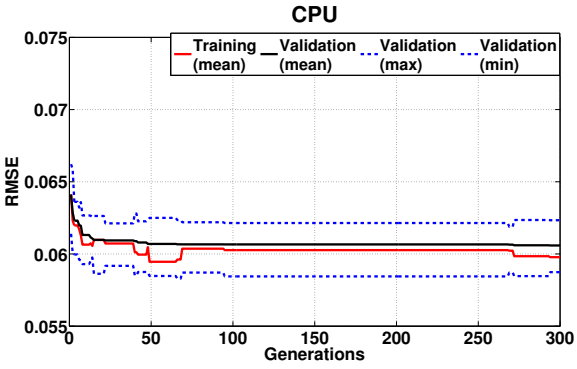
(e) Repetition 5



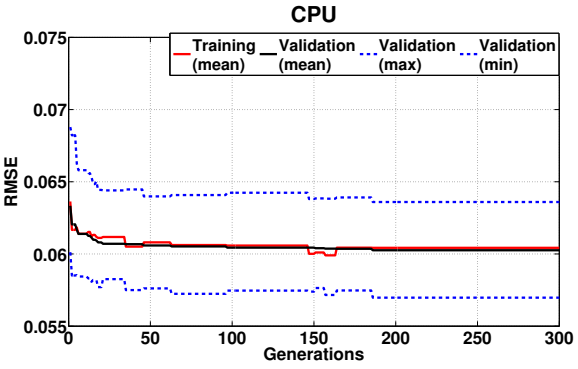
(f) Repetition 6



(g) Repetition 7



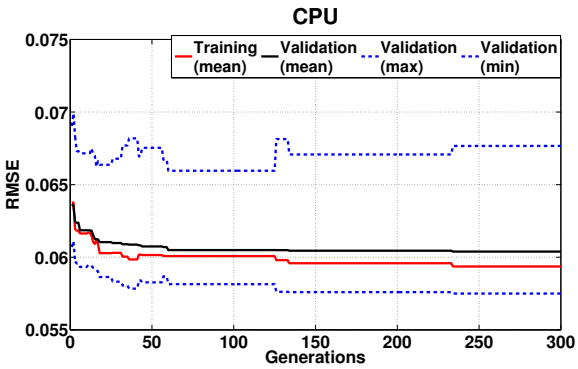
(h) Repetition 8



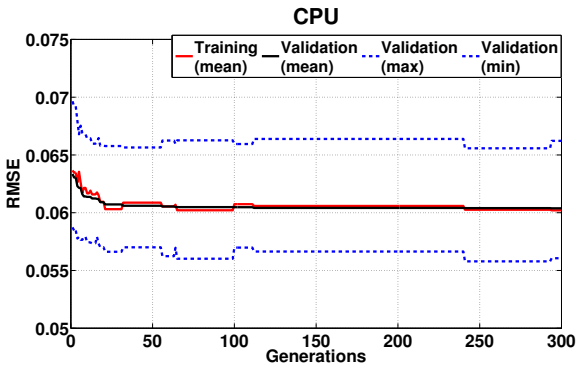
Source: author.

Figure 63 – ANHNA-DE$_{Re}$ convergence of 9 to 10 from 10 repetitions (CPU dataset).

(a) Repetition 9



(b) Repetition 10

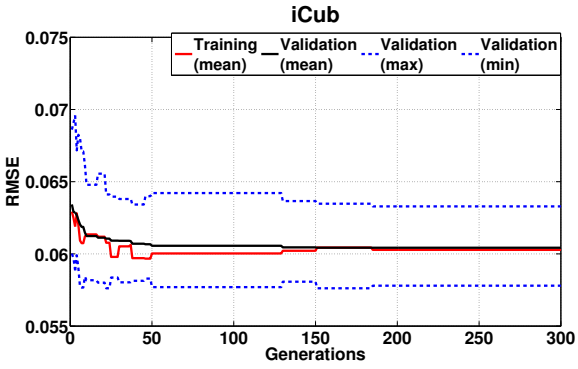


Source: author.

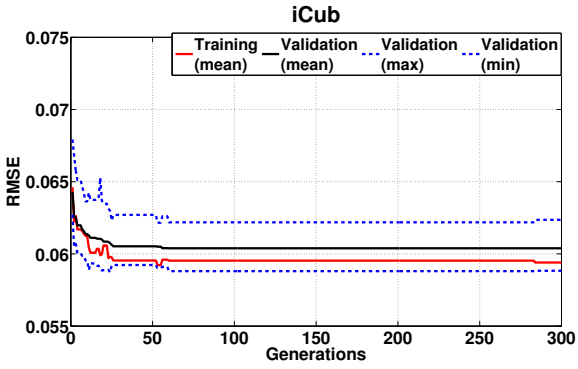
E.5 iCub

Figure 64 – ANHNA-DE$_{Re}$ convergence in 4 of 10 repetitions (iCub dataset).

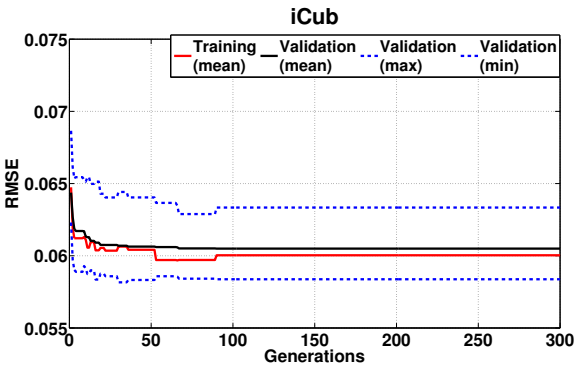
(a) Repetition 1



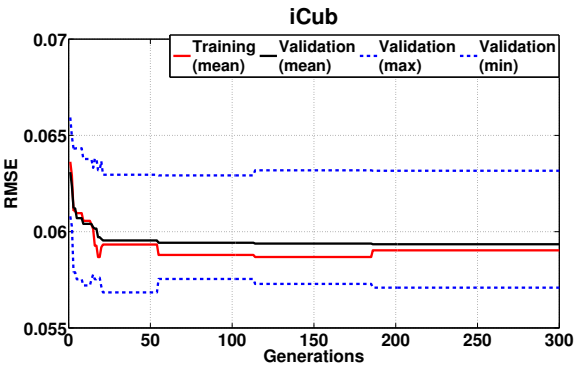
(b) Repetition 2



(c) Repetition 3



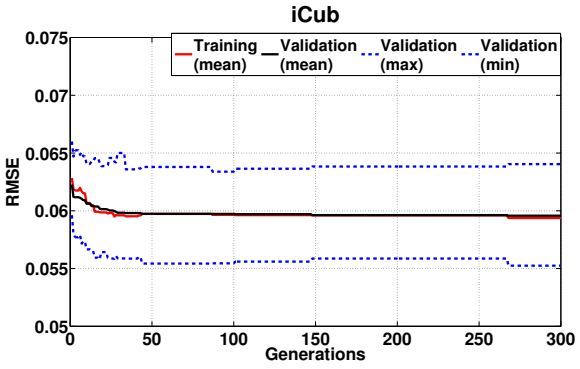
(d) Repetition 4



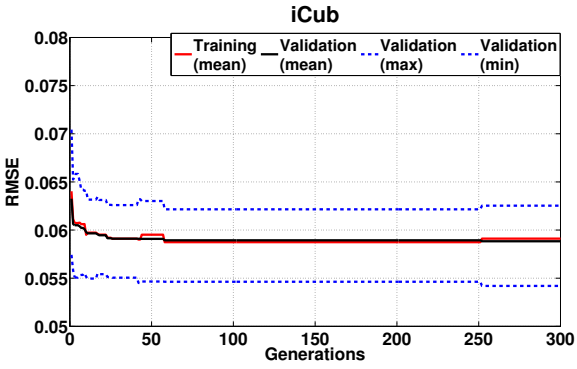
Source: author.

Figure 65 – ANHNA-DE$_{Re}$ convergence of 5 to 10 from 10 repetitions (iCub dataset).

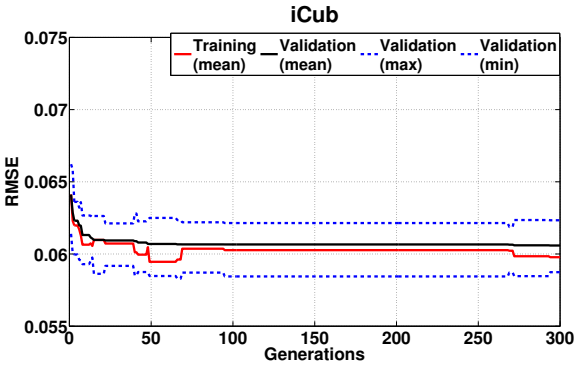
(a) Repetition 5



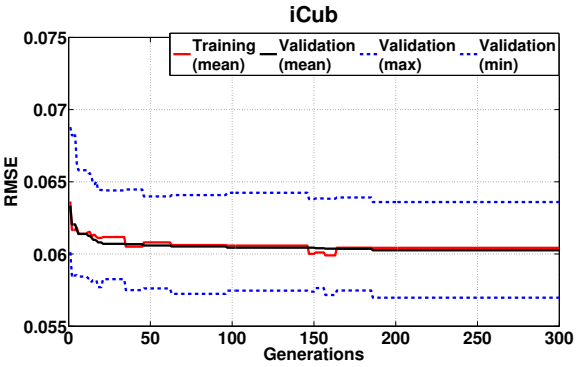
(b) Repetition 6



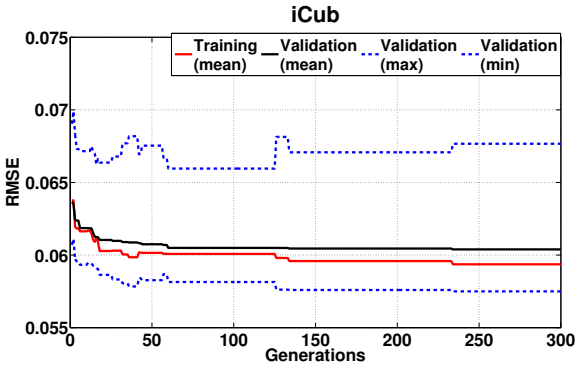
(c) Repetition 7



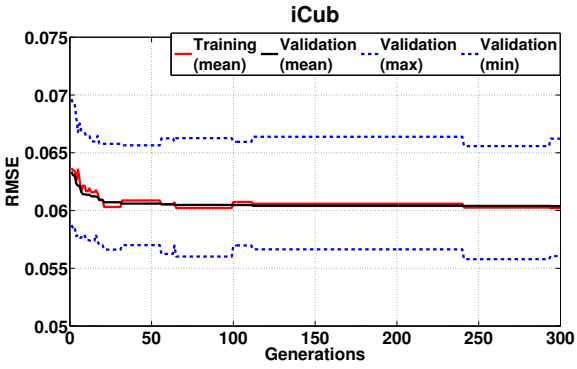
(d) Repetition 8



(e) Repetition 9



(f) Repetition 10

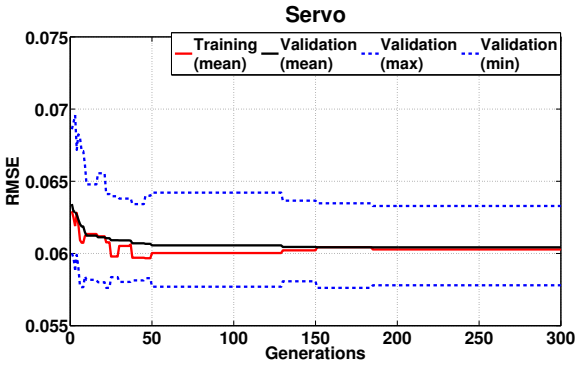


Source: author.

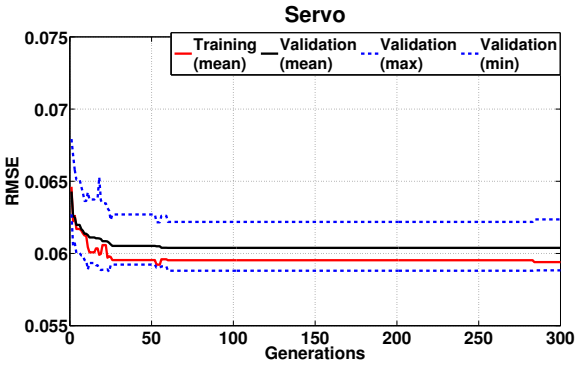
E.6 Servo

Figure 66 – ANHNA-DE$_{Re}$ convergence in 8 of 10 repetitions (Servo dataset).

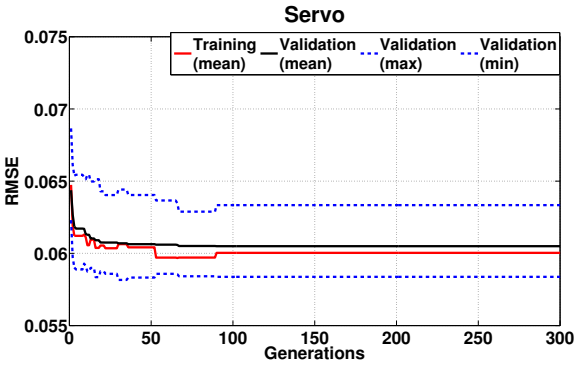
(a) Repetition 1



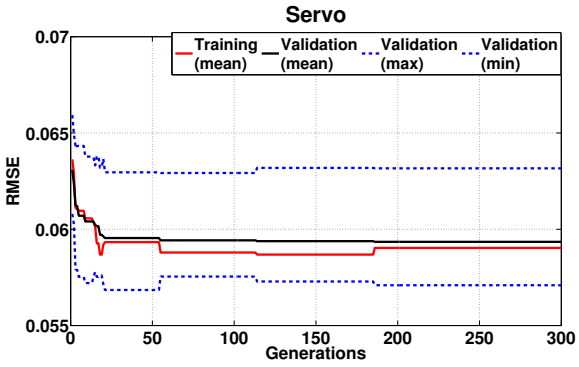
(b) Repetition 2



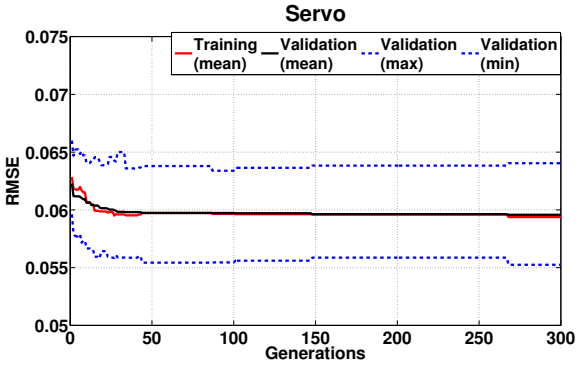
(c) Repetition 3



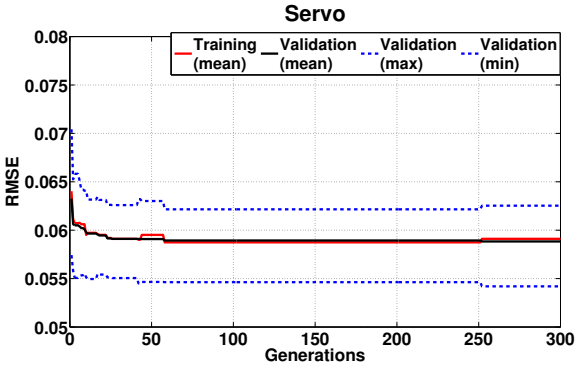
(d) Repetition 4



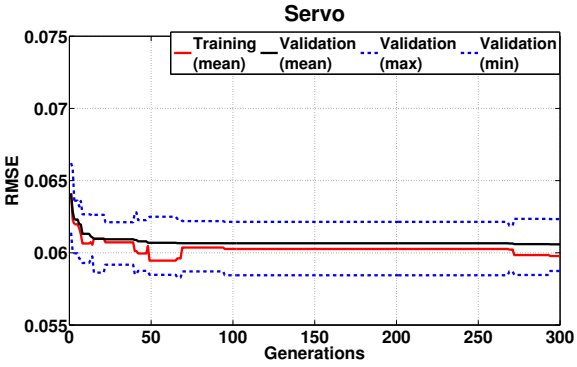
(e) Repetition 5



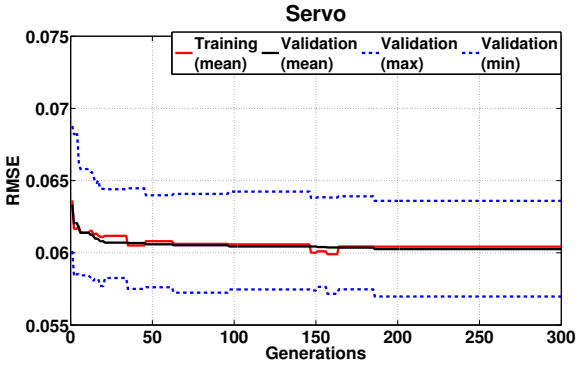
(f) Repetition 6



(g) Repetition 7



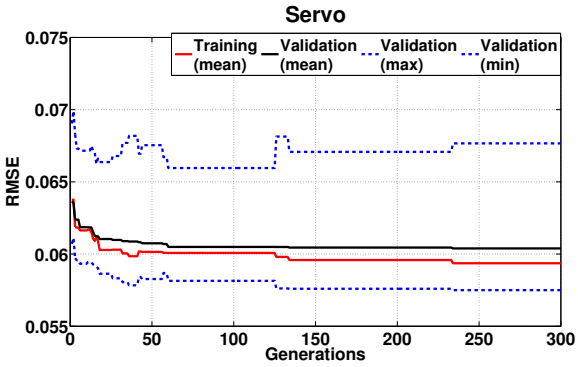
(h) Repetition 8



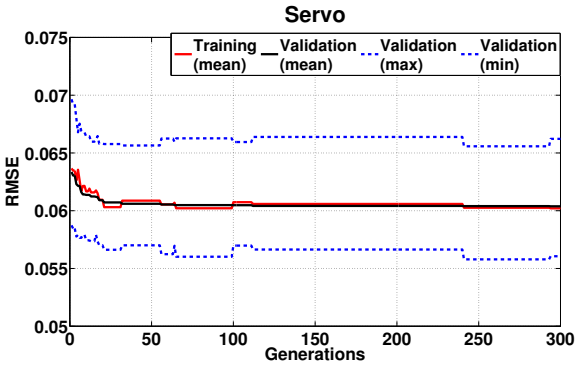
Source: author.

Figure 67 – ANHNA-DE$_{Re}$ convergence of 9 to 10 from 10 repetitions (Servo dataset).

(a) Repetition 9



(b) Repetition 10



Source: author.

APPENDIX F – ROBUST ANHNA'S CONVERGENCE

This appendix is dedicated to the exhibition of the weight function's choice evolution over the generations, as well as its respective error threshold (or tunning constant) of the two versions of robust ANHNA: using trimmed mean (ANHNA$_R$-DE) or median ANHNA$_{Rm}$-DE.

Here we present the five adopted real-world regression problems datasets: Auto-MPG, Breast Cancer, CPU, and Servo. For each of them, it was applied 1 sided or 2 sided contaminations, which had also different percentages of contamination: 10%, 20%, 20% and 20%.

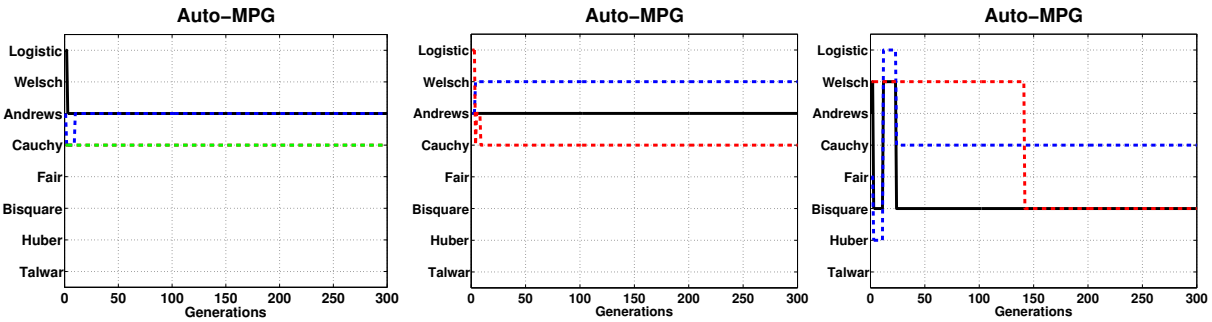
F.1 ANHNA$_R$-DE

F.1.1 Auto-MPG

F.1.1.1 Auto-MPG with 1 sided and 10% of contamination

Figure 68 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 1 sided and 10% of outlier contamination.

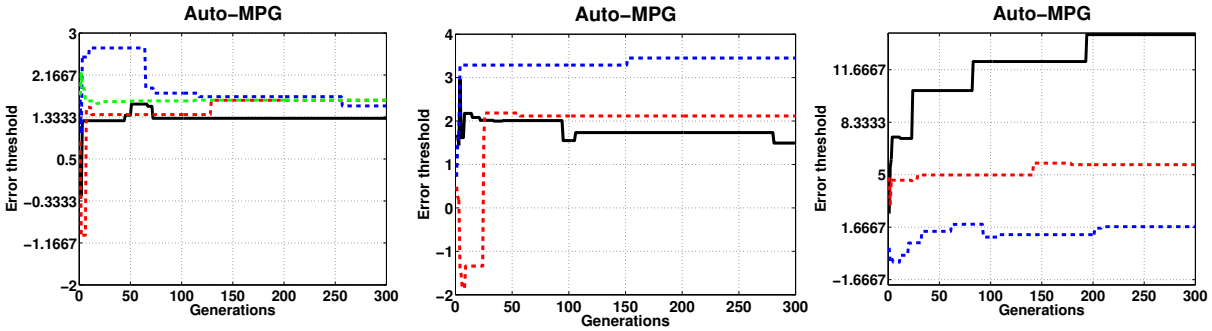
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 69 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

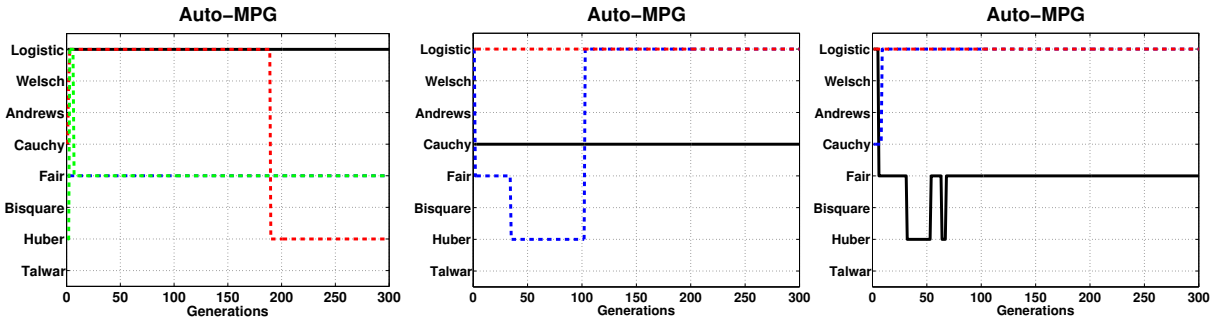


Source: author.

F.1.1.2 Auto-MPG with 1 sided and 20% of contamination

Figure 70 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 1 sided and 20% of outlier contamination.

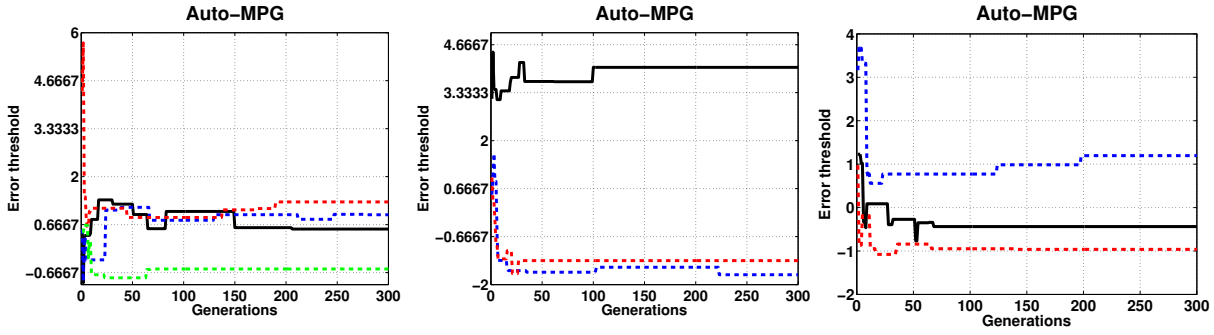
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 71 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

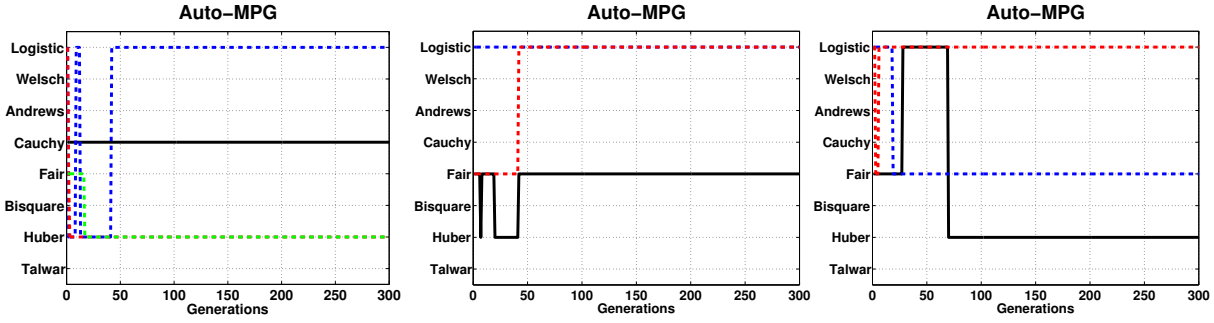


Source: author.

F.1.1.3 Auto-MPG with 1 sided and 30% of contamination

Figure 72 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 1 sided and 30% of outlier contamination.

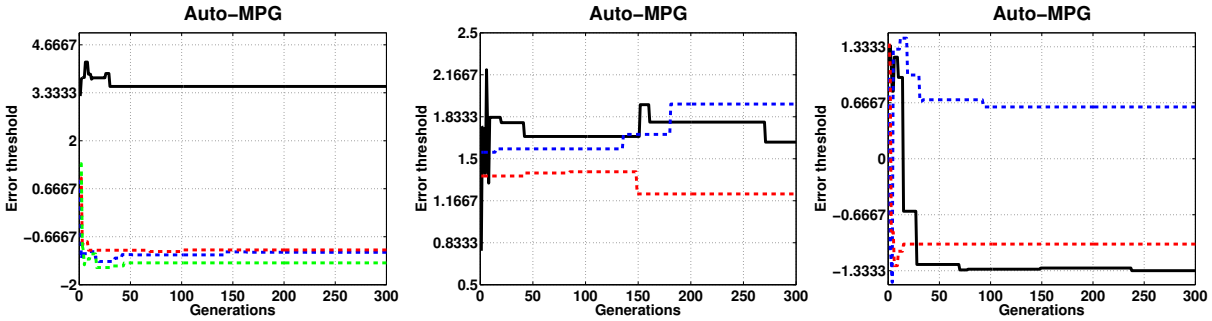
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 73 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

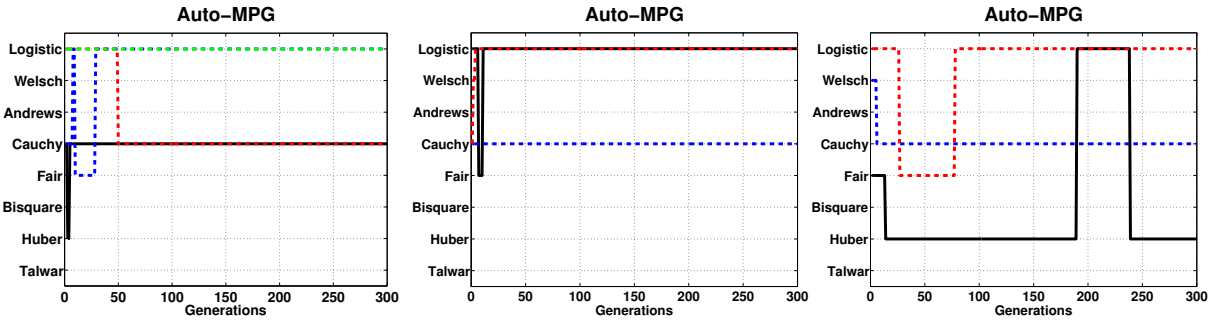


Source: author.

F.1.1.4 Auto-MPG with 1 sided and 40% of contamination

Figure 74 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 1 sided and 40% of outlier contamination.

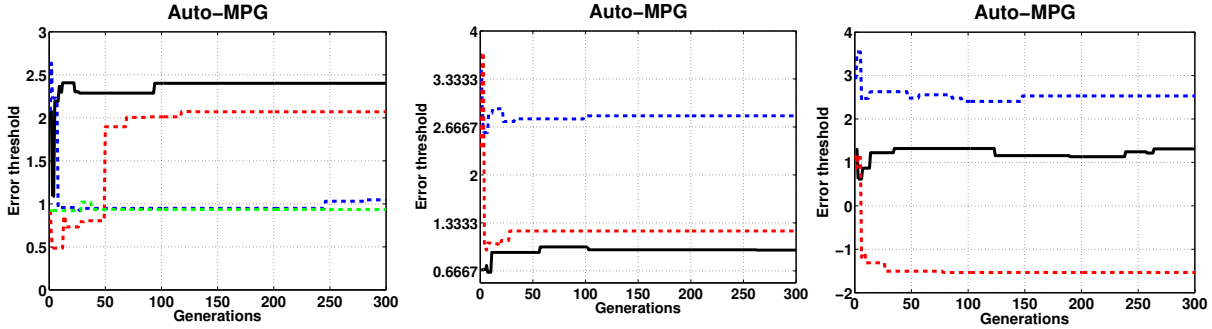
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 75 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

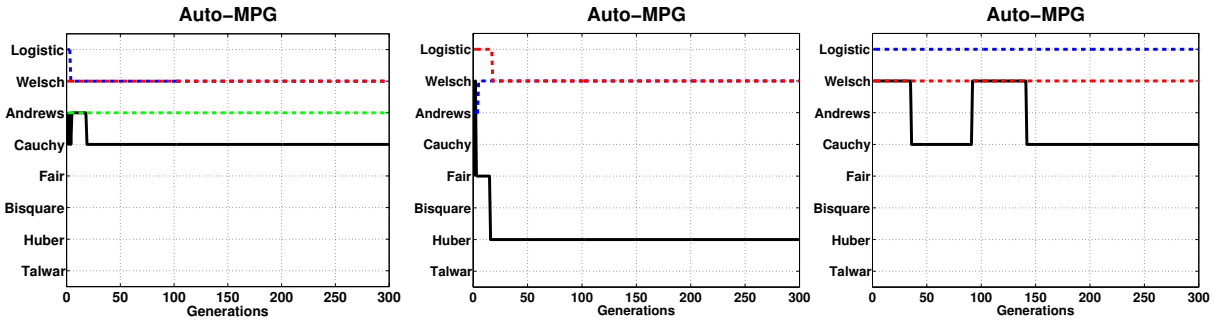


Source: author.

F.1.1.5 Auto-MPG with 2 sided and 10% of contamination

Figure 76 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 2 sided and 10% of outlier contamination.

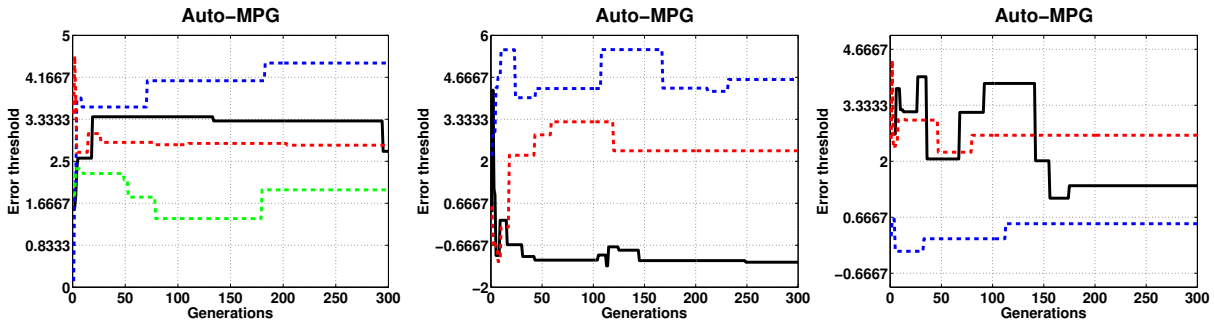
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 77 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

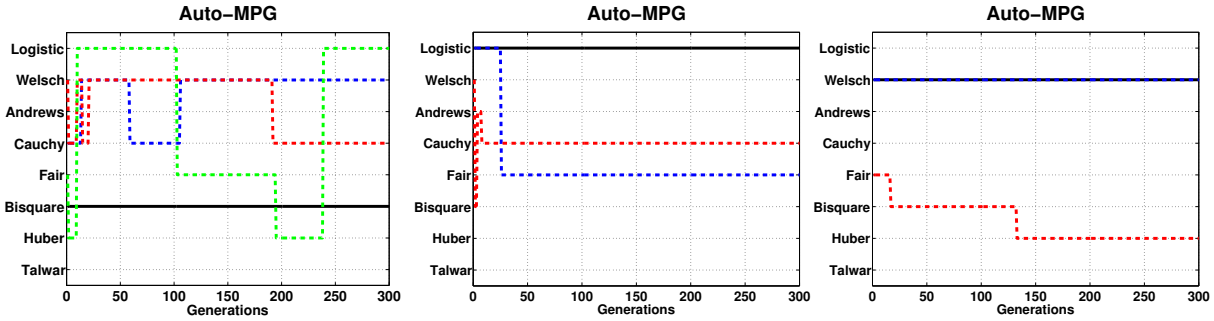


Source: author.

F.1.1.6 Auto-MPG with 2 sided and 20% of contamination

Figure 78 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 2 sided and 20% of outlier contamination.

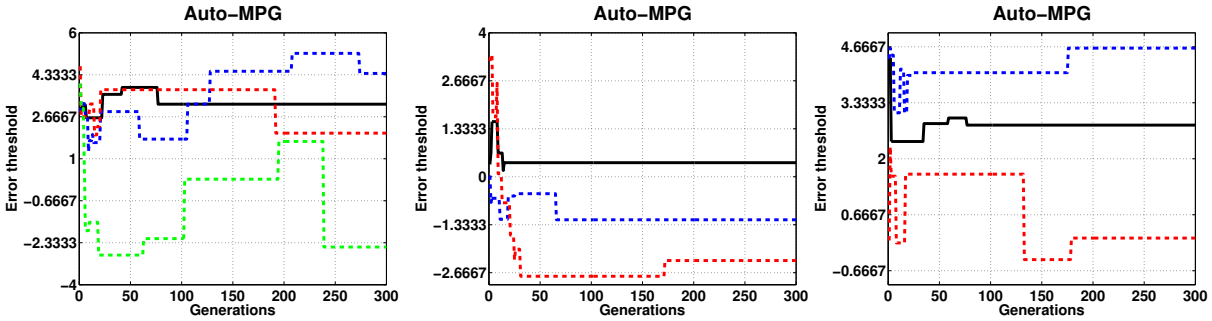
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 79 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

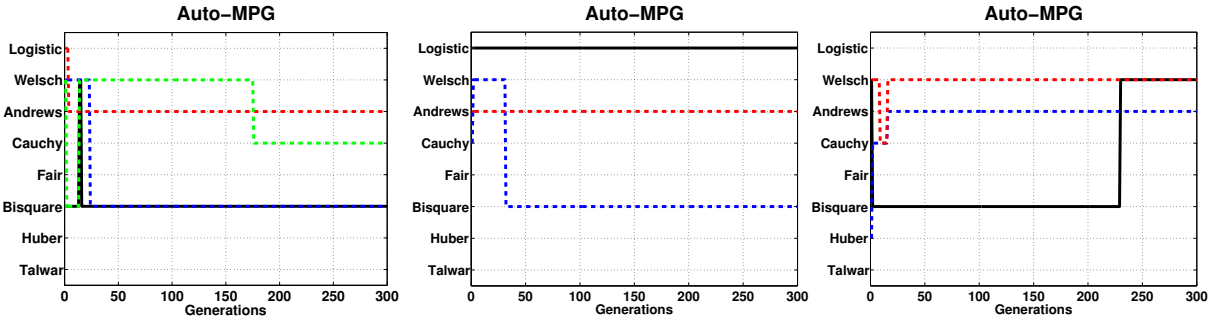


Source: author.

F.1.1.7 Auto-MPG with 2 sided and 30% of contamination

Figure 80 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 2 sided and 30% of outlier contamination.

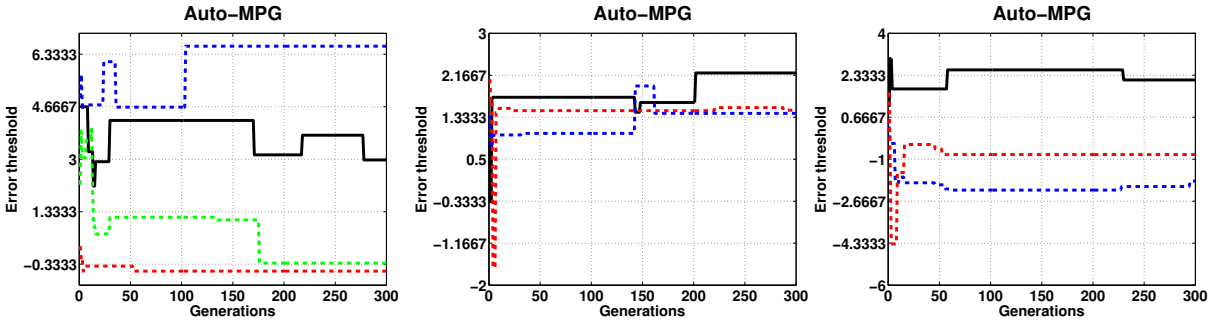
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 81 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

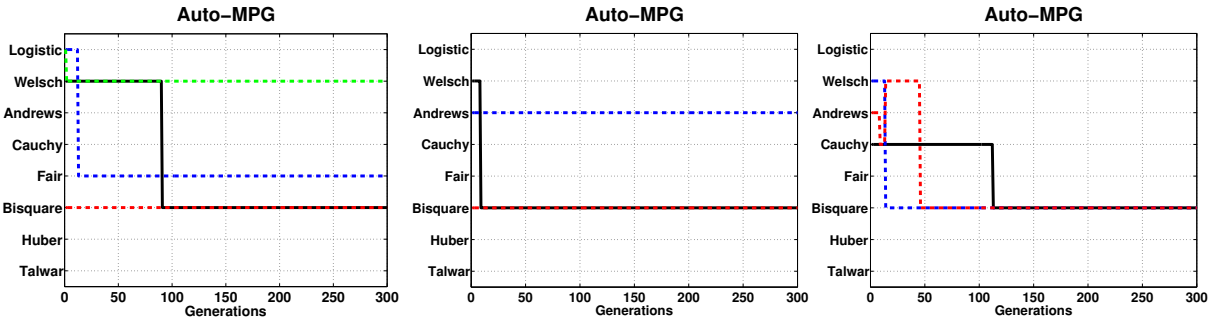


Source: author.

F.1.1.8 Auto-MPG with 2 sided and 40% of contamination

Figure 82 – ANHNA$_R$-DE weight function choice for Auto-MPG dataset with 2 sided and 40% of outlier contamination.

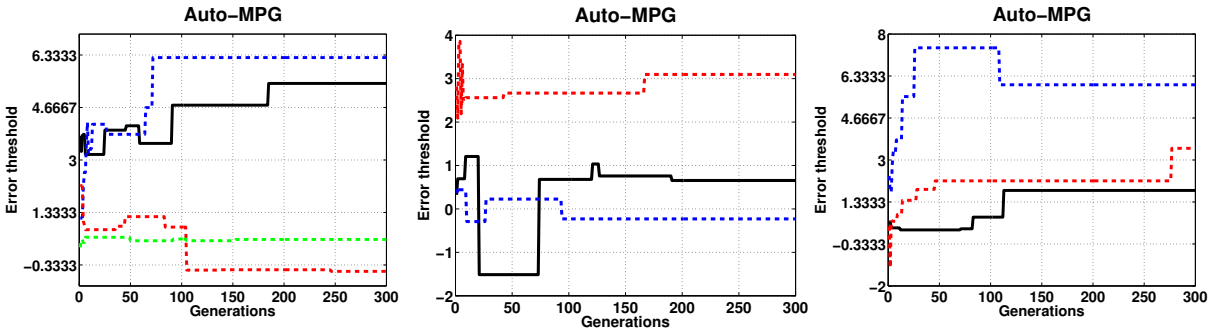
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 83 – ANHNA$_R$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



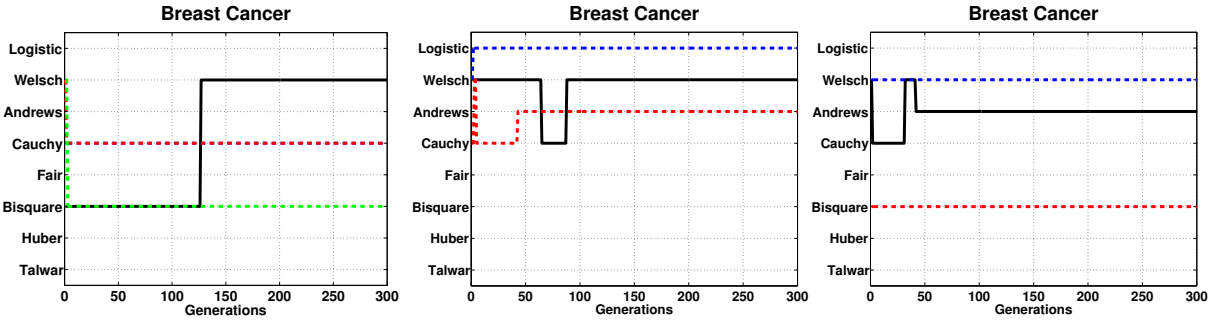
Source: author.

F.1.2 Breast Cancer

F.1.2.1 Breast Cancer with 1 sided and 10% of contamination

Figure 84 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 1 sided and 10% of outlier contamination.

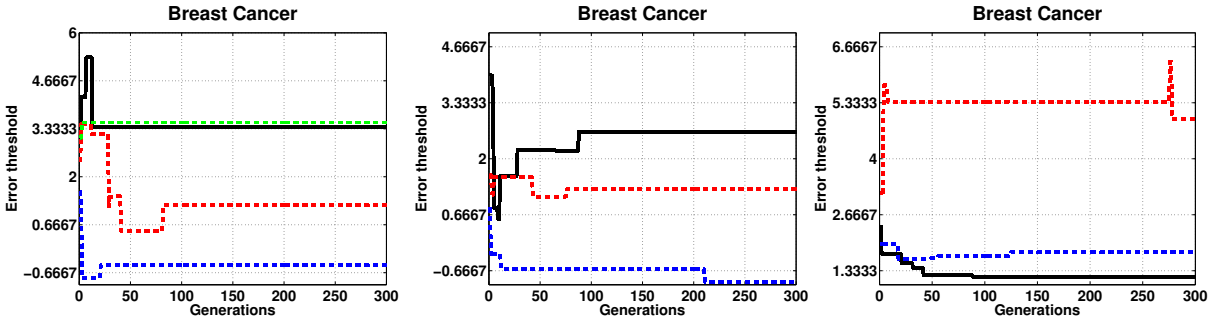
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 85 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

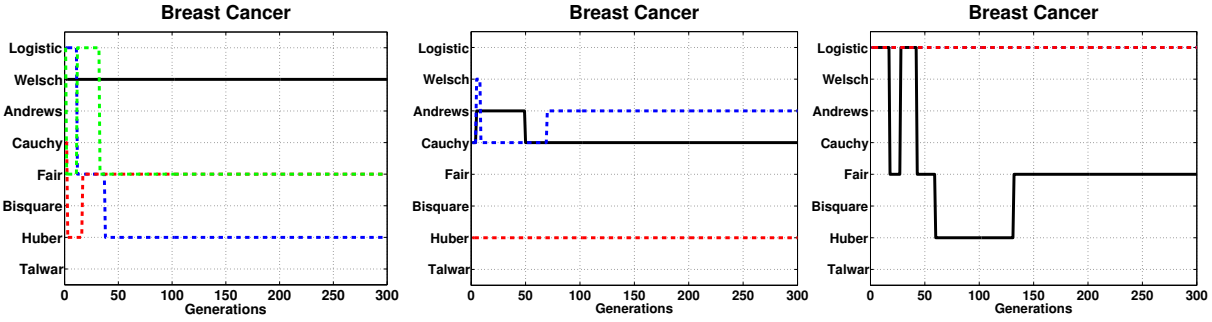


Source: author.

F.1.2.2 Breast Cancer with 1 sided and 20% of contamination

Figure 86 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 1 sided and 20% of outlier contamination.

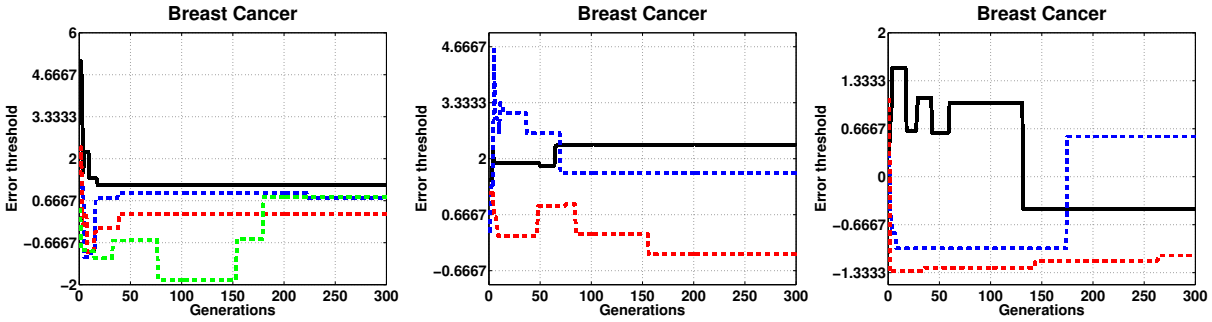
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 87 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

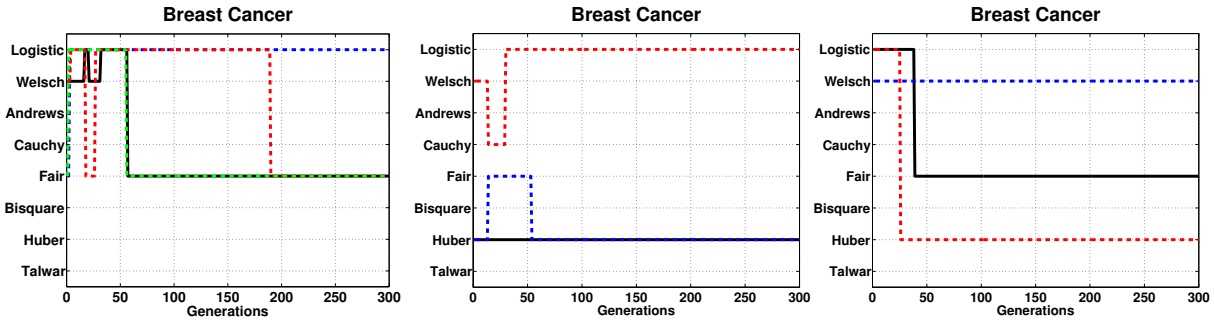


Source: author.

F.1.2.3 Breast Cancer with 1 sided and 30% of contamination

Figure 88 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 1 sided and 30% of outlier contamination.

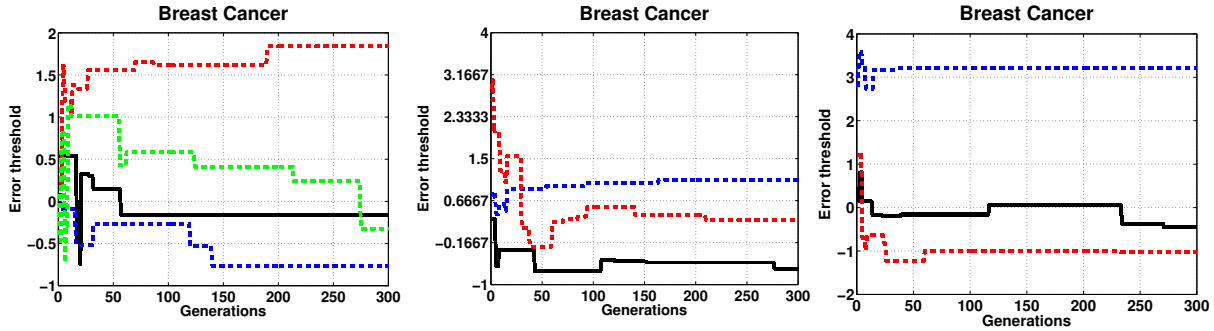
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 89 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

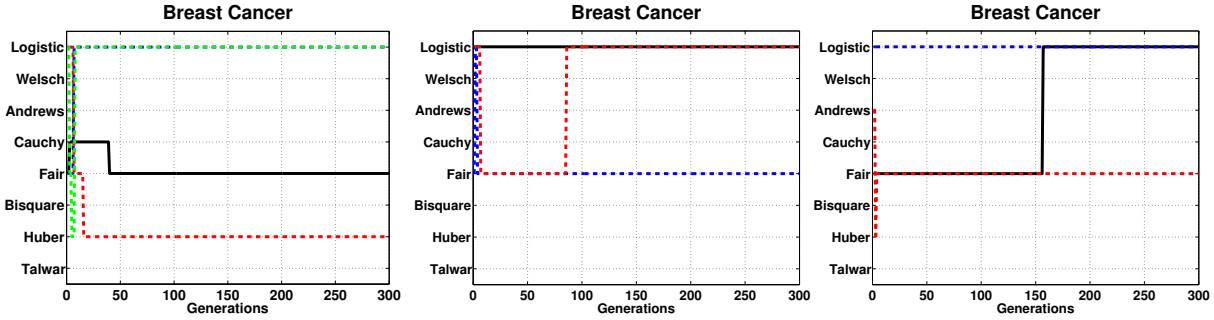


Source: author.

F.1.2.4 Breast Cancer with 1 sided and 40% of contamination

Figure 90 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 1 sided and 40% of outlier contamination.

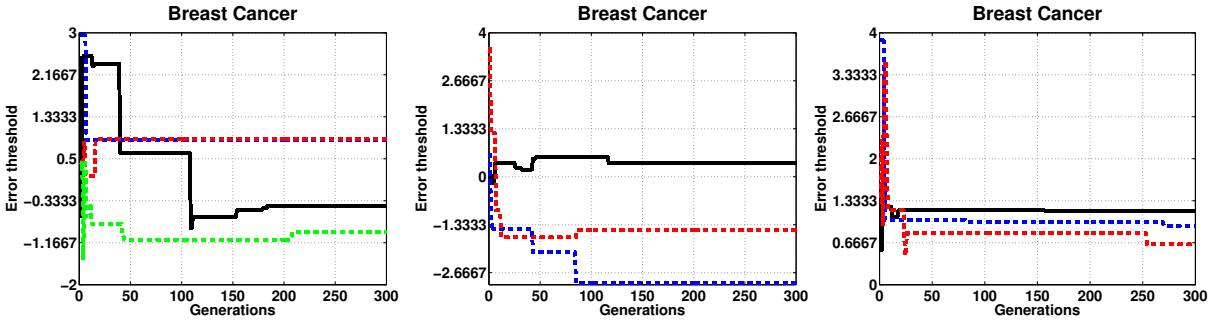
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 91 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

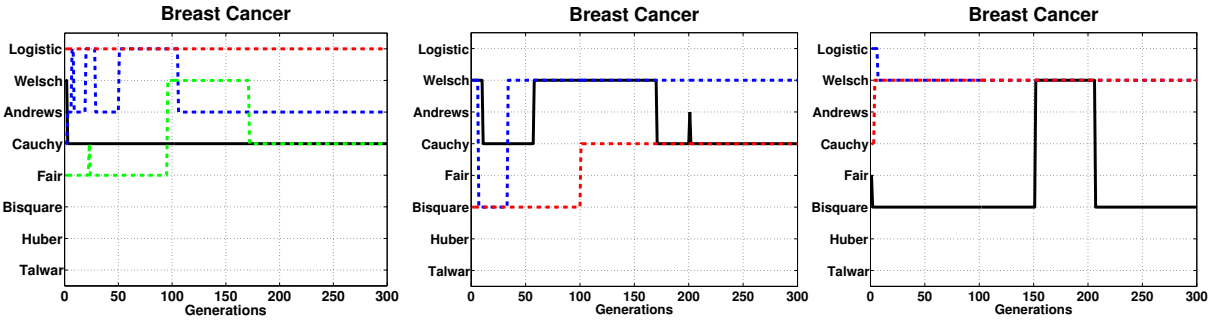


Source: author.

F.1.2.5 Breast Cancer with 2 sided and 10% of contamination

Figure 92 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 2 sided and 10% of outlier contamination.

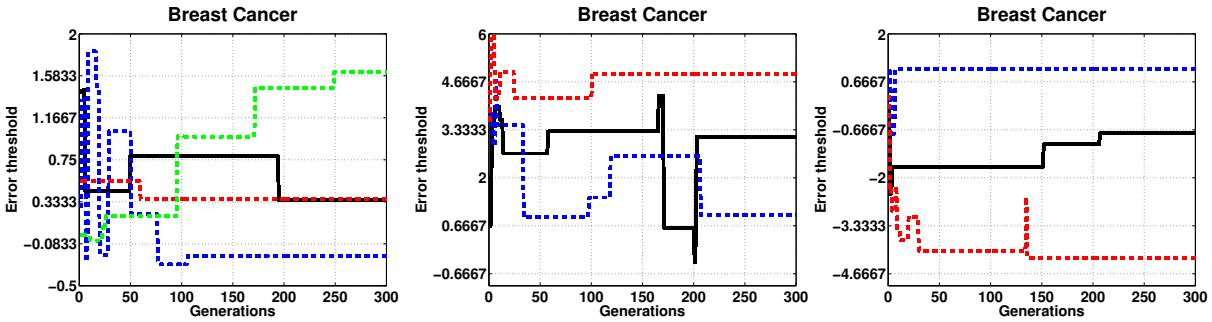
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 93 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

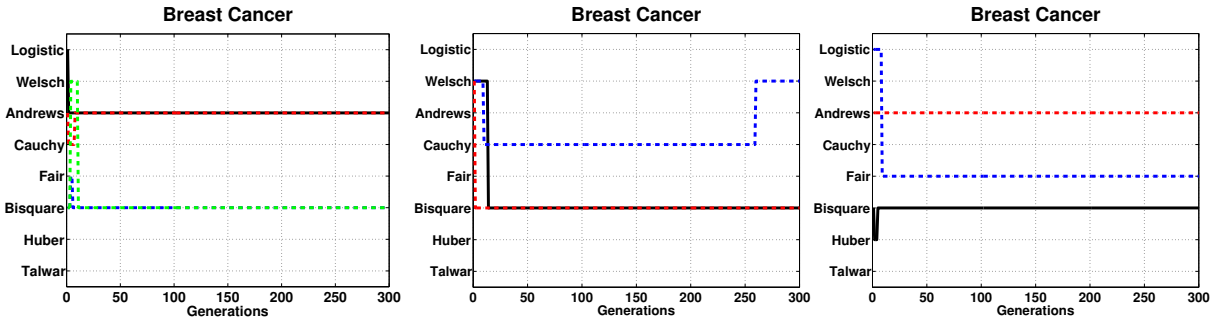


Source: author.

F.1.2.6 Breast Cancer with 2 sided and 20% of contamination

Figure 94 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 2 sided and 20% of outlier contamination.

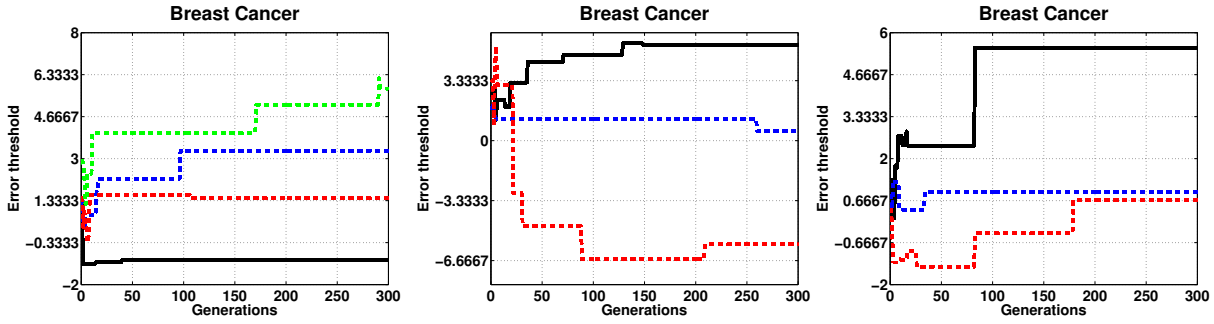
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 95 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

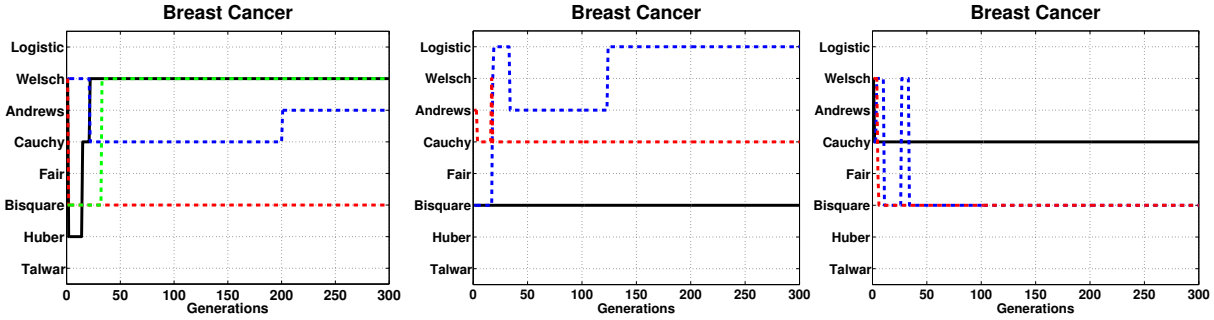


Source: author.

F.1.2.7 Breast Cancer with 2 sided and 30% of contamination

Figure 96 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 2 sided and 30% of outlier contamination.

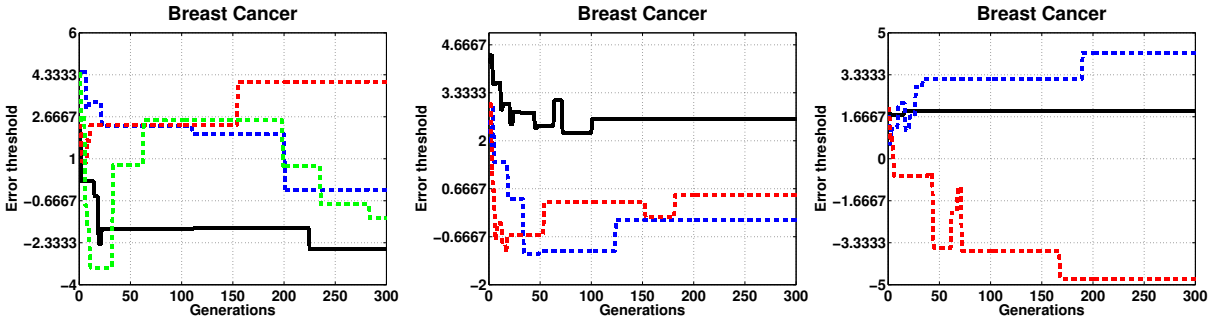
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 97 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

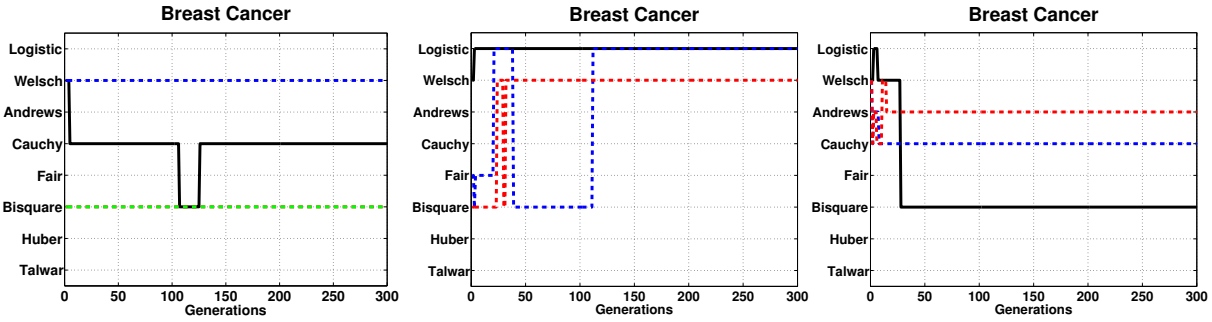


Source: author.

F.1.2.8 Breast Cancer with 2 sided and 40% of contamination

Figure 98 – ANHNA$_R$-DE weight function choice for Breast Cancer dataset with 2 sided and 40% of outlier contamination.

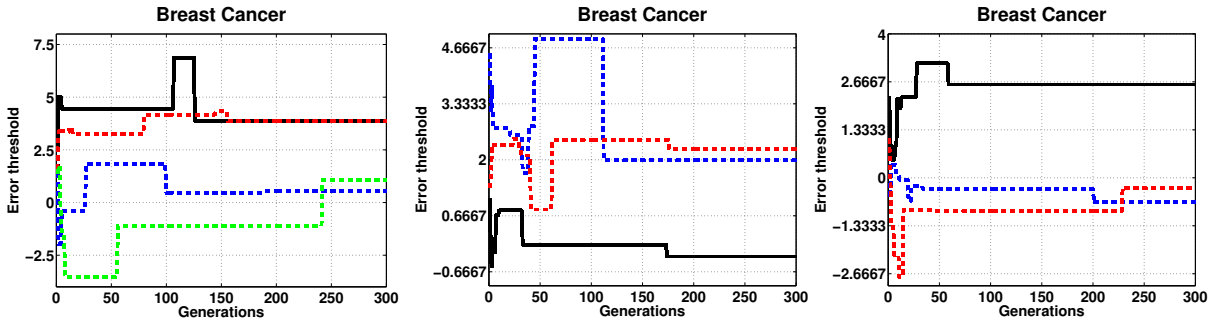
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 99 – ANHNA$_R$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



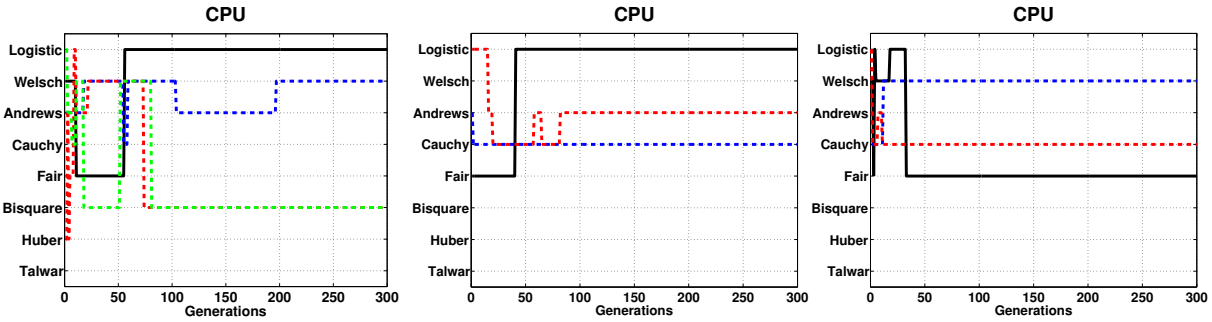
Source: author.

F.1.3 CPU

F.1.3.1 CPU with 1 sided and 10% of contamination

Figure 100 – ANHNA$_R$-DE weight function choice for CPU dataset with 1 sided and 10% of outlier contamination.

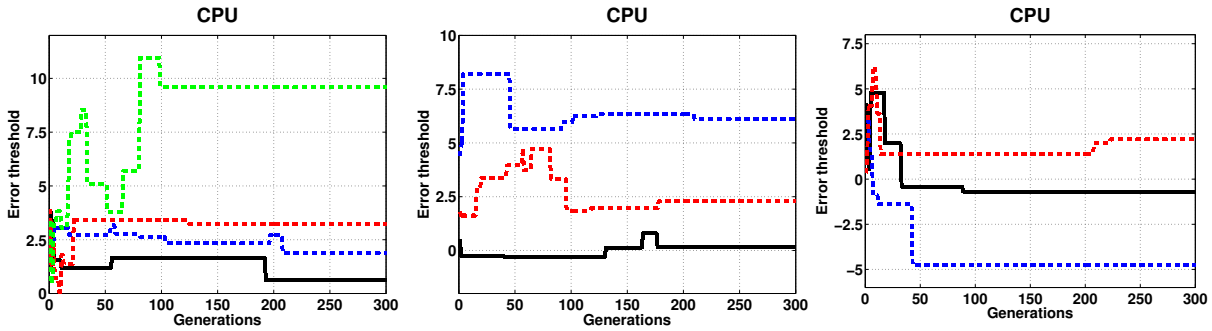
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 101 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

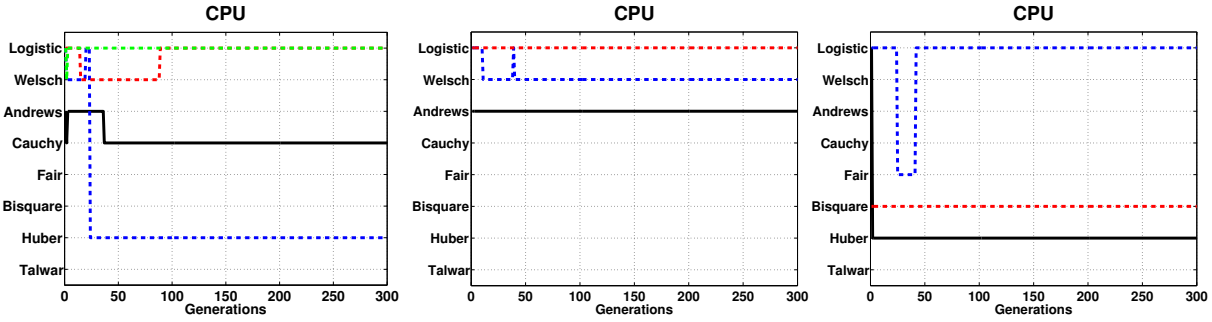


Source: author.

F.1.3.2 CPU with 1 sided and 20% of contamination

Figure 102 – ANHNA$_R$-DE weight function choice for CPU dataset with 1 sided and 20% of outlier contamination.

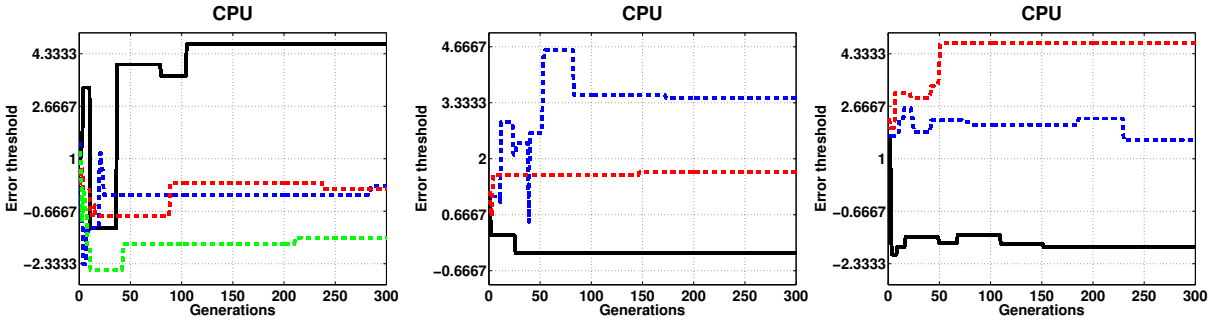
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 103 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

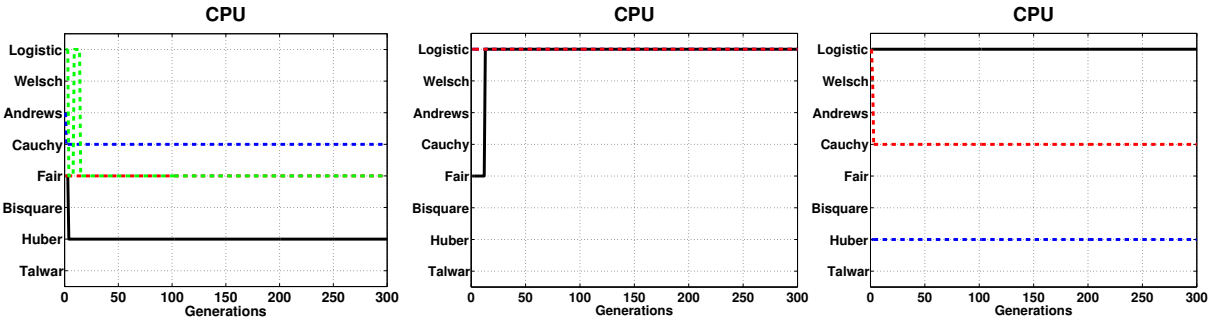


Source: author.

F.1.3.3 CPU with 1 sided and 30% of contamination

Figure 104 – ANHNA$_R$-DE weight function choice for CPU dataset with 1 sided and 30% of outlier contamination.

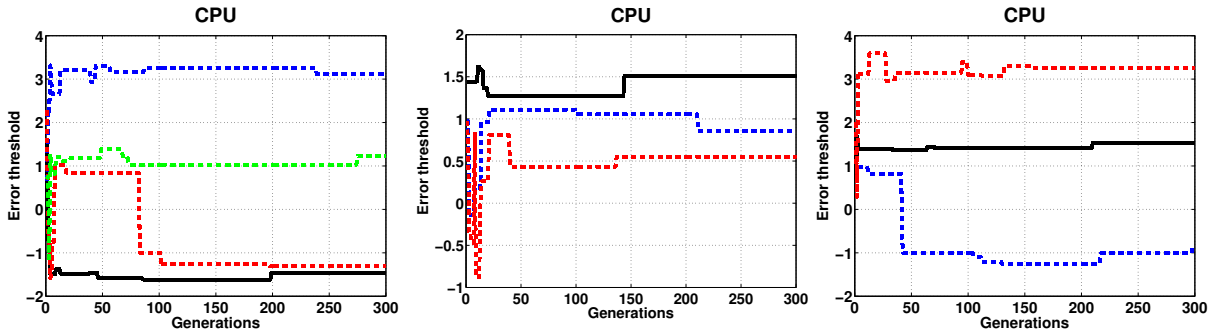
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 105 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

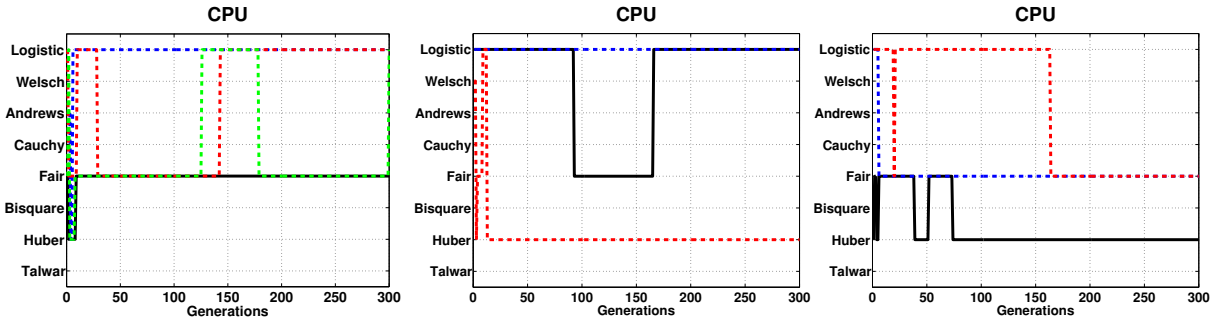


Source: author.

F.1.3.4 CPU with 1 sided and 40% of contamination

Figure 106 – ANHNA$_R$-DE weight function choice for CPU dataset with 1 sided and 40% of outlier contamination.

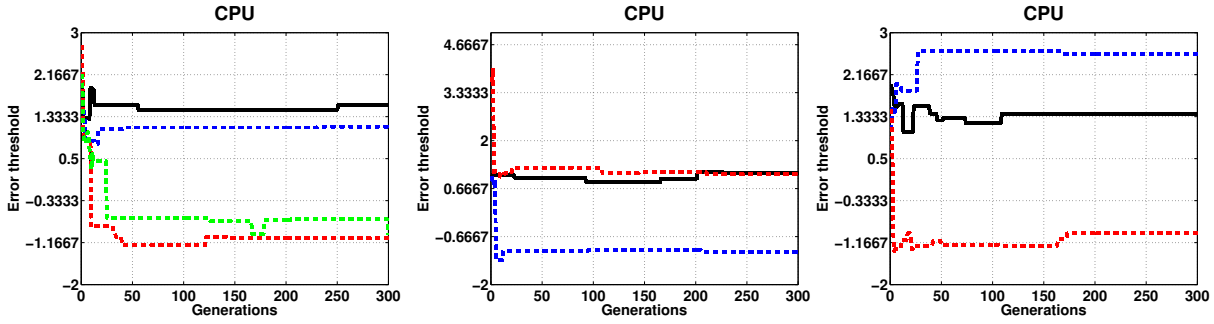
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 107 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

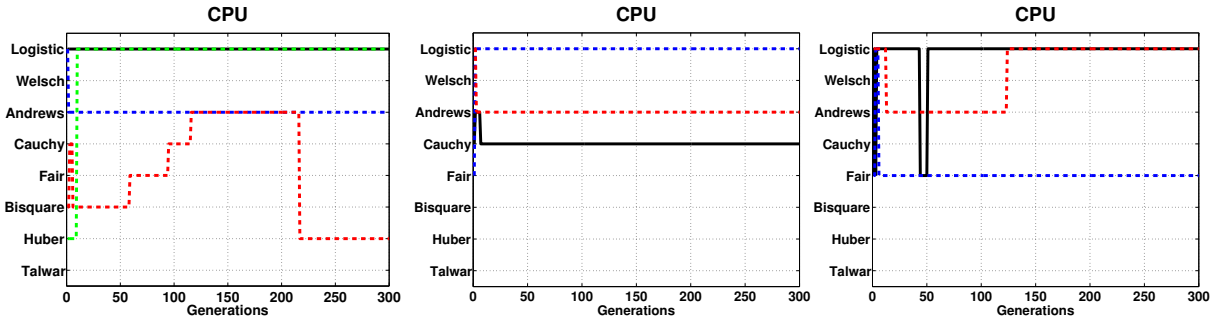


Source: author.

F.1.3.5 CPU with 2 sided and 10% of contamination

Figure 108 – ANHNA$_R$-DE weight function choice for CPU dataset with 2 sided and 10% of outlier contamination.

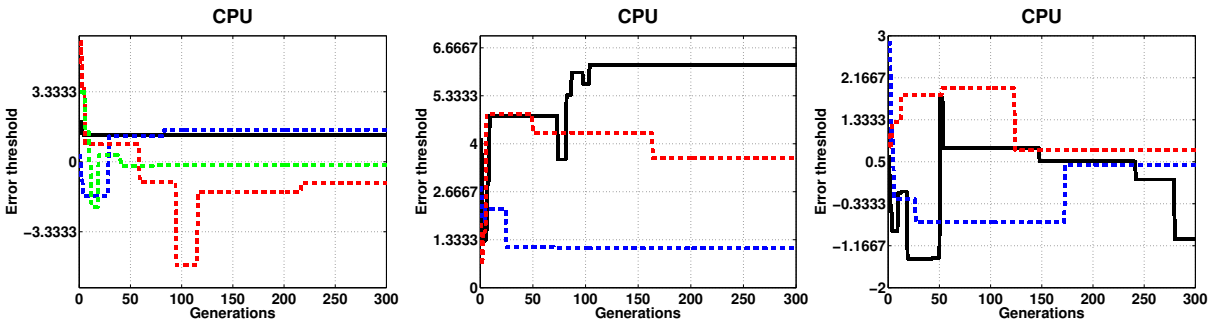
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 109 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

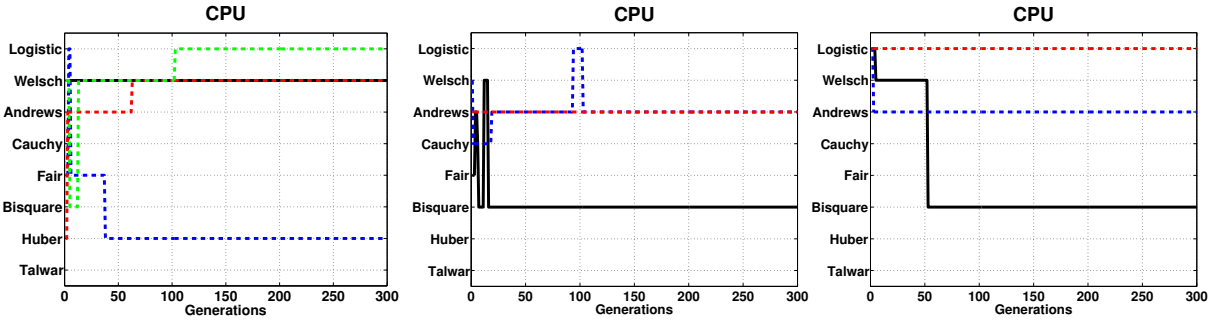


Source: author.

F.1.3.6 CPU with 2 sided and 20% of contamination

Figure 110 – ANHNA$_R$-DE weight function choice for CPU dataset with 2 sided and 20% of outlier contamination.

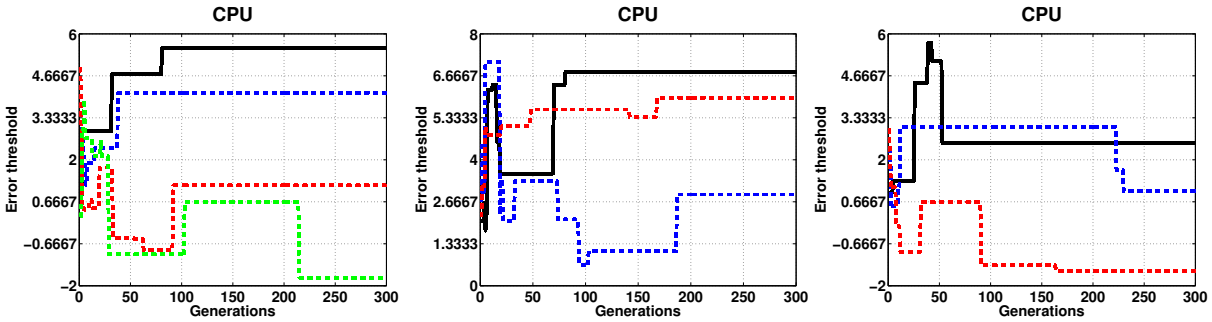
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 111 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

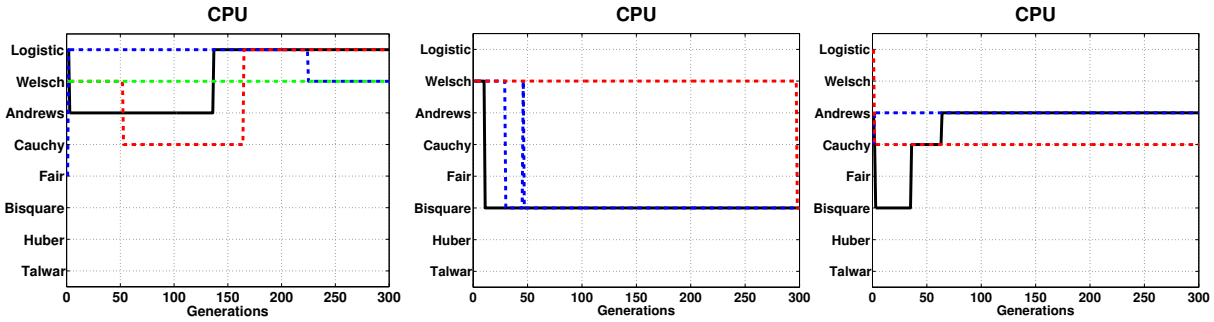


Source: author.

F.1.3.7 CPU with 2 sided and 30% of contamination

Figure 112 – ANHNA$_R$-DE weight function choice for CPU dataset with 2 sided and 30% of outlier contamination.

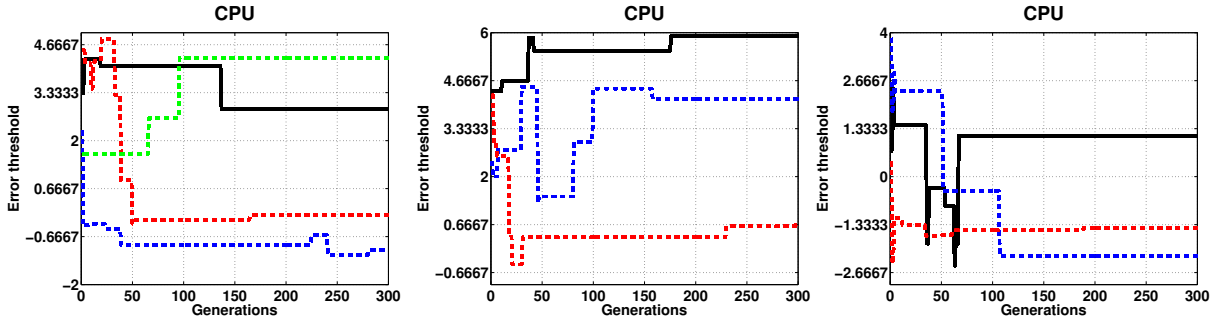
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 113 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

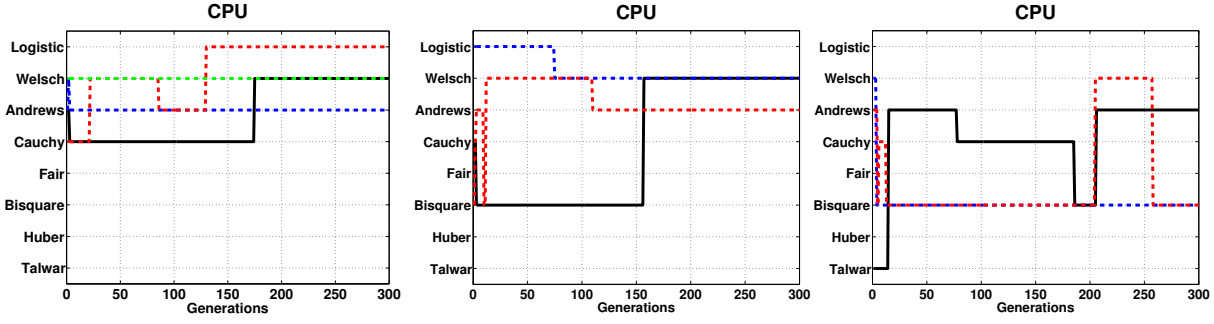


Source: author.

F.1.3.8 CPU with 2 sided and 40% of contamination

Figure 114 – ANHNA$_R$-DE weight function choice for CPU dataset with 2 sided and 40% of outlier contamination.

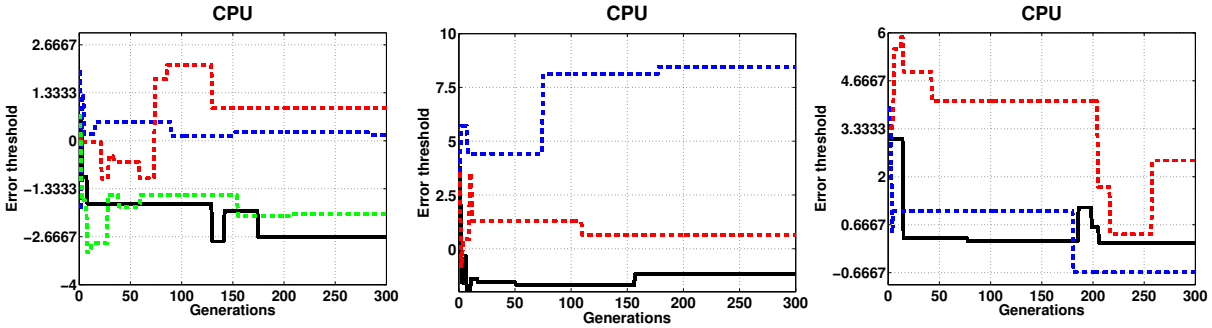
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 115 – ANHNA$_R$-DE error threshold evolution for CPU dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



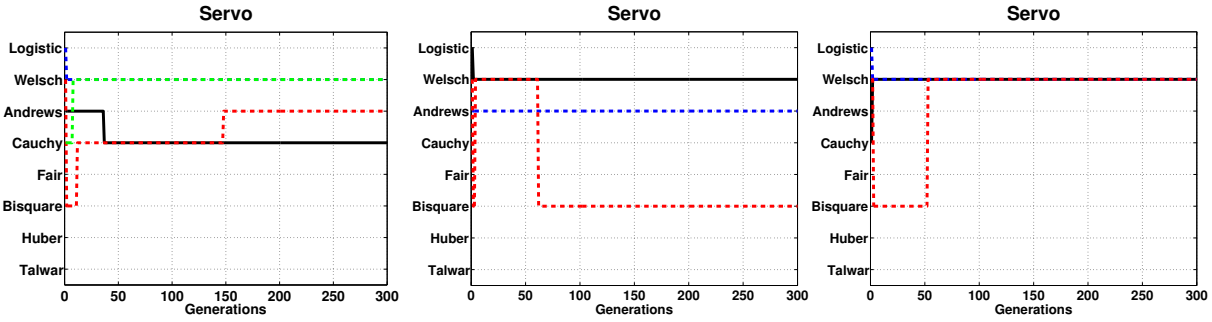
Source: author.

F.1.4 Servo

F.1.4.1 Servo with 1 sided and 10% of contamination

Figure 116 – ANHNA$_R$-DE weight function choice for Servo dataset with 1 sided and 10% of outlier contamination.

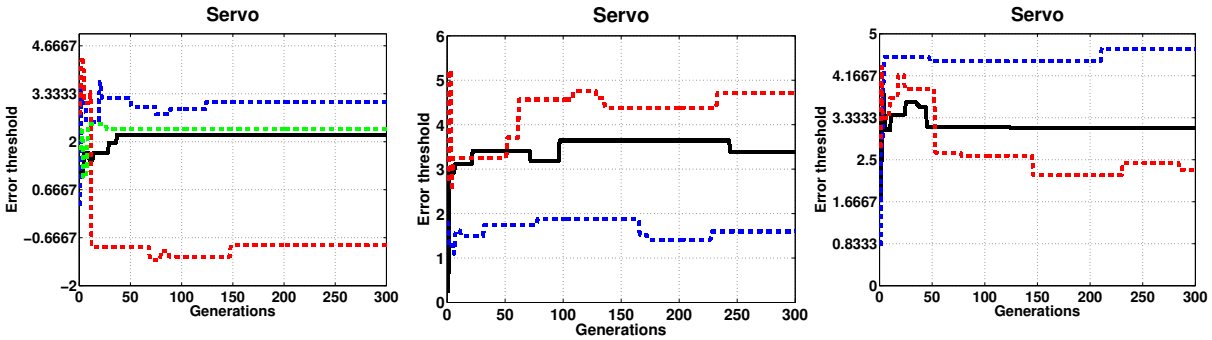
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 117 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

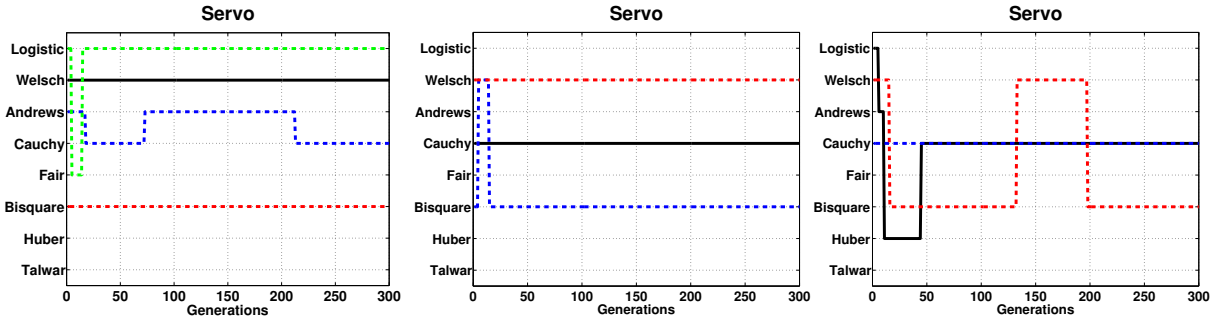


Source: author.

F.1.4.2 Servo with 1 sided and 20% of contamination

Figure 118 – ANHNA$_R$-DE weight function choice for Servo dataset with 1 sided and 20% of outlier contamination.

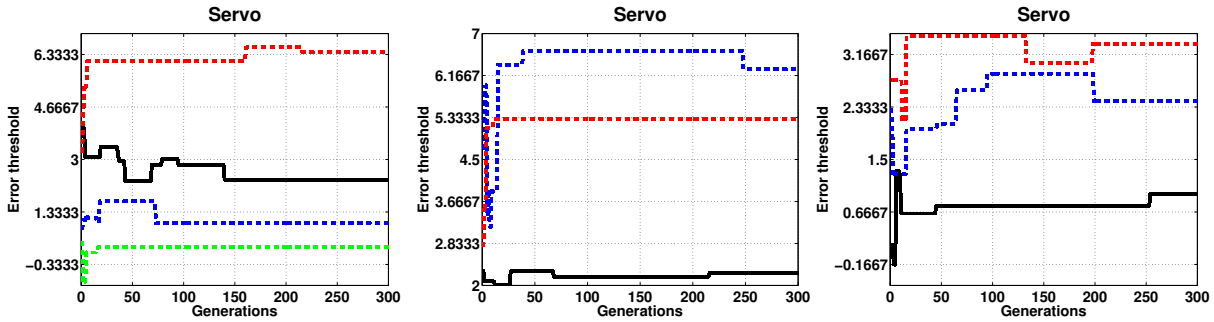
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 119 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

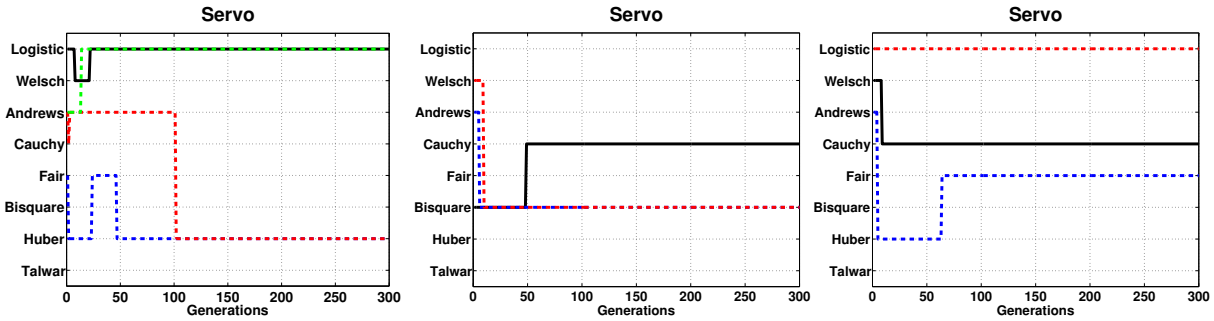


Source: author.

F.1.4.3 Servo with 1 sided and 30% of contamination

Figure 120 – ANHNA$_R$-DE weight function choice for Servo dataset with 1 sided and 30% of outlier contamination.

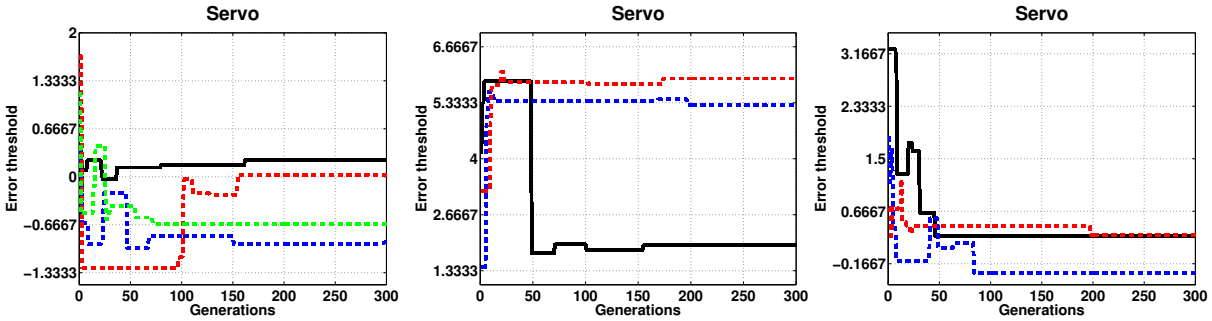
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 121 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

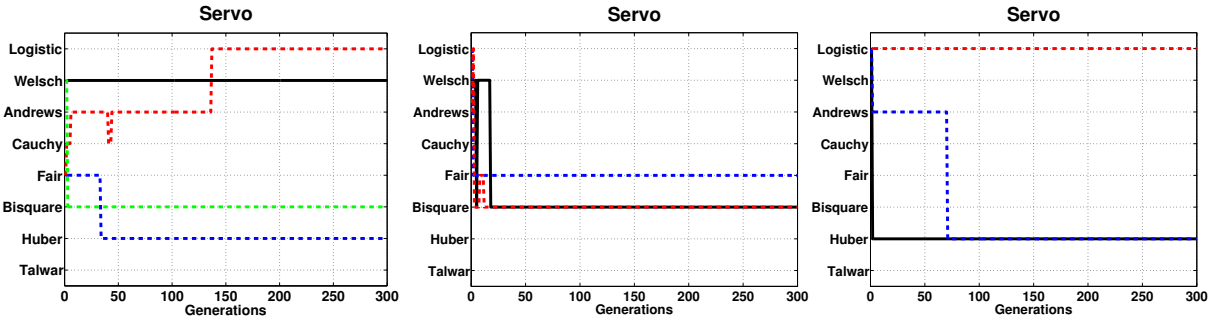


Source: author.

F.1.4.4 Servo with 1 sided and 40% of contamination

Figure 122 – ANHNA$_R$-DE weight function choice for Servo dataset with 1 sided and 40% of outlier contamination.

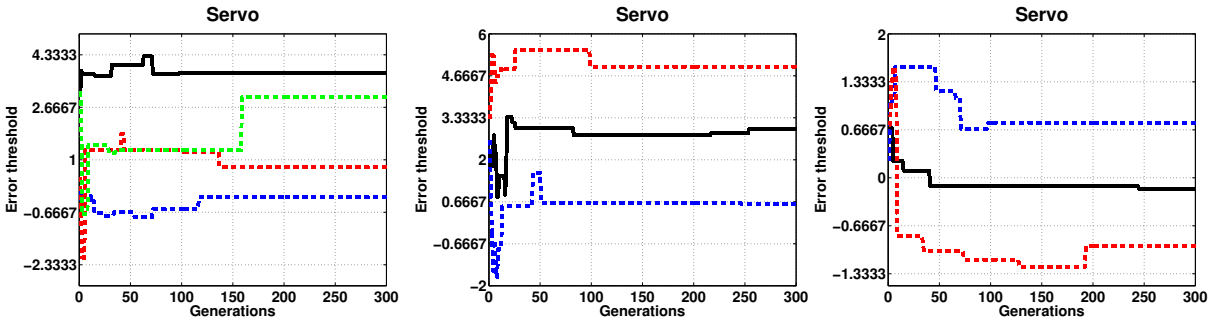
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 123 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

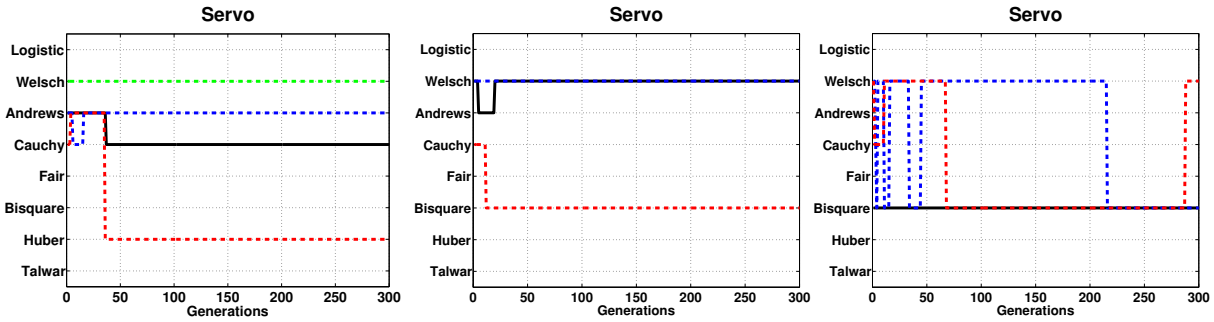


Source: author.

F.1.4.5 Servo with 2 sided and 10% of contamination

Figure 124 – ANHNA$_R$-DE weight function choice for Servo dataset with 2 sided and 10% of outlier contamination.

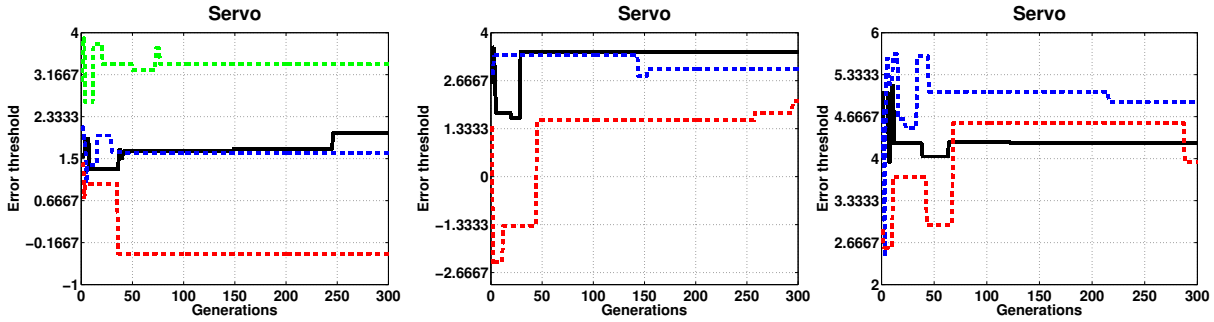
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 125 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

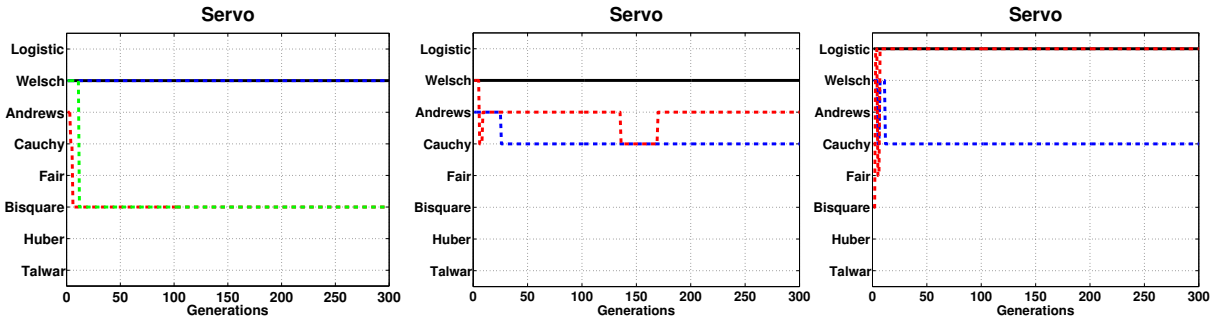


Source: author.

F.1.4.6 Servo with 2 sided and 20% of contamination

Figure 126 – ANHNA$_R$-DE weight function choice for Servo dataset with 2 sided and 20% of outlier contamination.

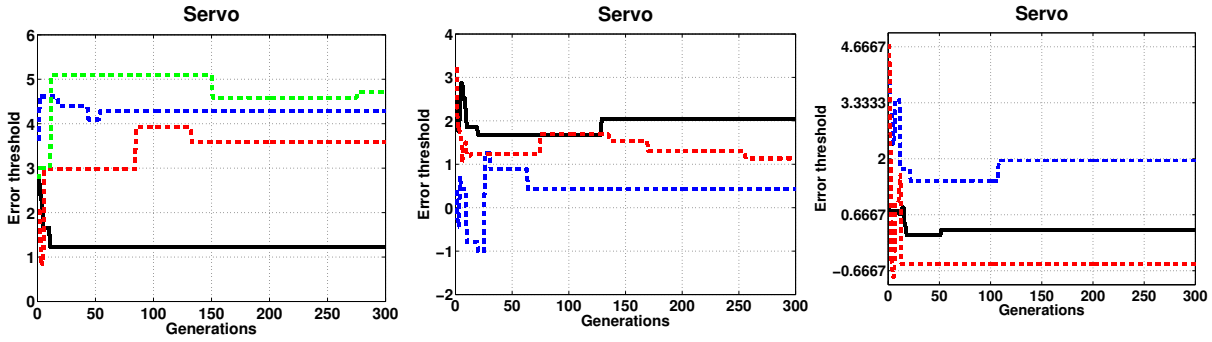
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 127 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

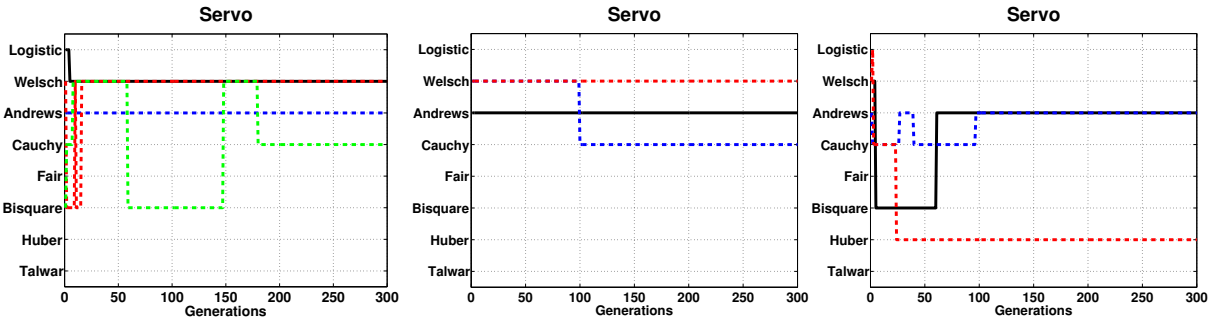


Source: author.

F.1.4.7 Servo with 2 sided and 30% of contamination

Figure 128 – ANHNA$_R$-DE weight function choice for Servo dataset with 2 sided and 30% of outlier contamination.

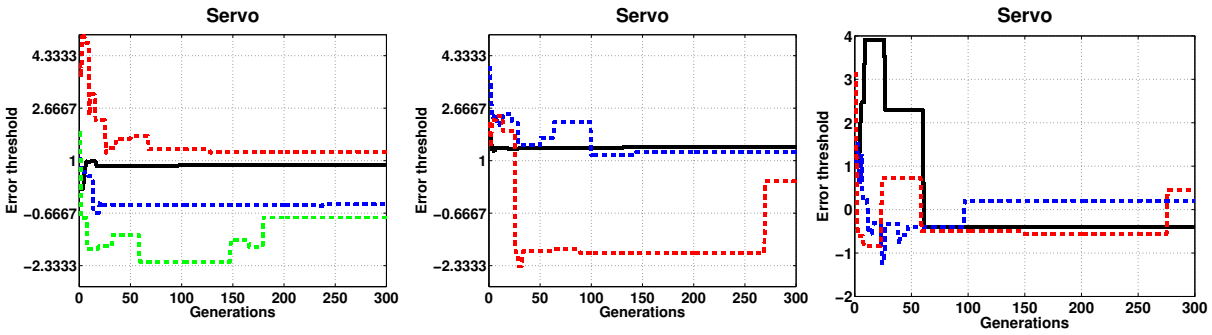
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 129 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

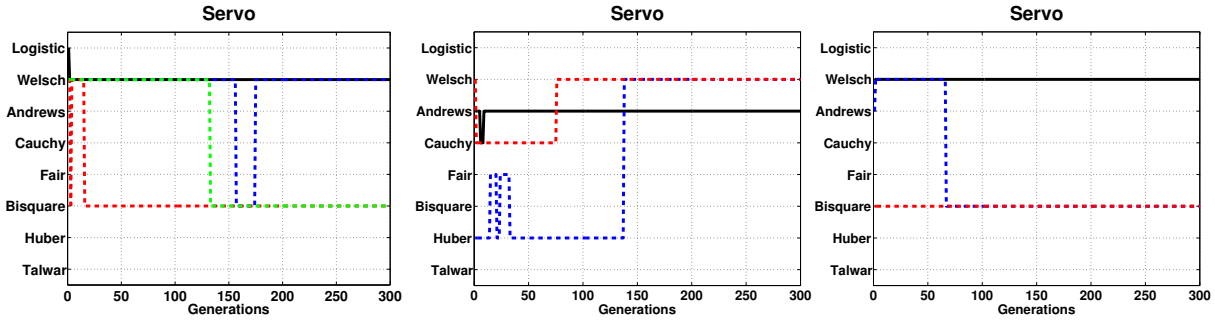


Source: author.

F.1.4.8 Servo with 2 sided and 40% of contamination

Figure 130 – ANHNA$_R$-DE weight function choice for Servo dataset with 2 sided and 40% of outlier contamination.

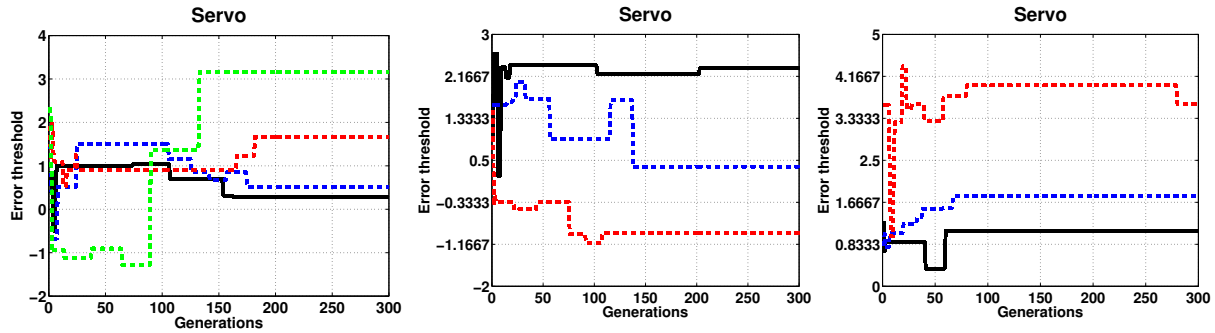
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 131 – ANHNA$_R$-DE error threshold evolution for Servo dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

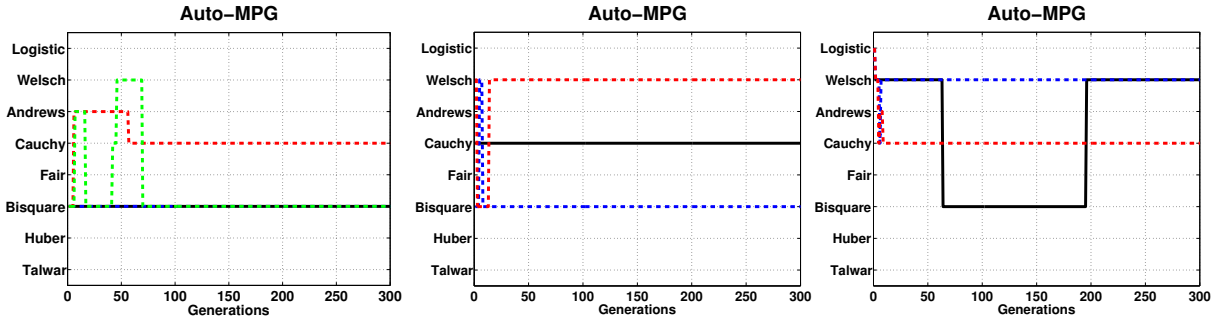
F.2 ANHNA$_{Rm}$-DE

F.2.1 Auto-MPG

F.2.1.1 Auto-MPG with 1 sided and 10% of contamination

Figure 132 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 1 sided and 10% of outlier contamination.

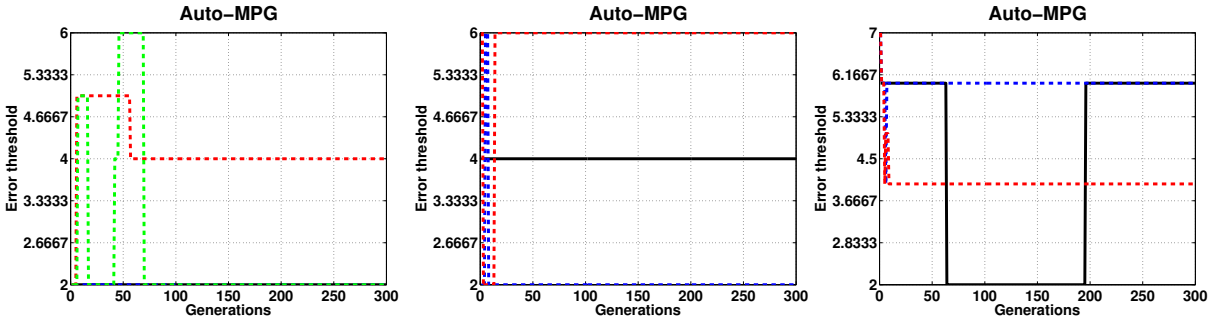
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 133 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

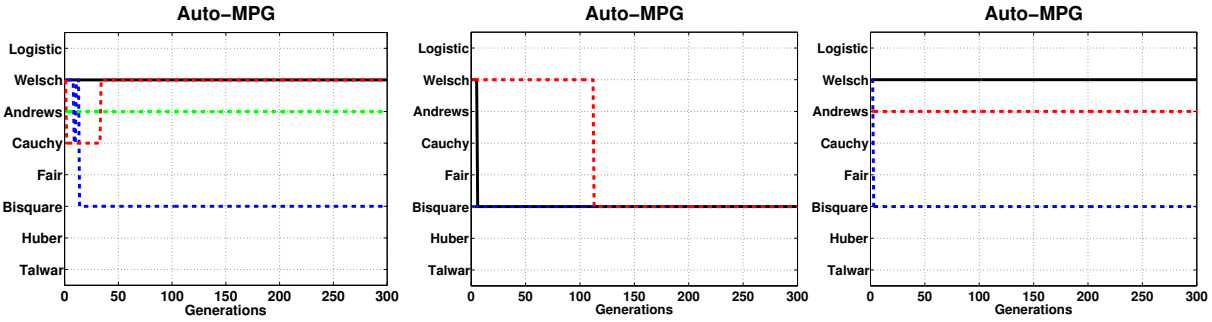


Source: author.

F.2.1.2 Auto-MPG with 1 sided and 20% of contamination

Figure 134 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

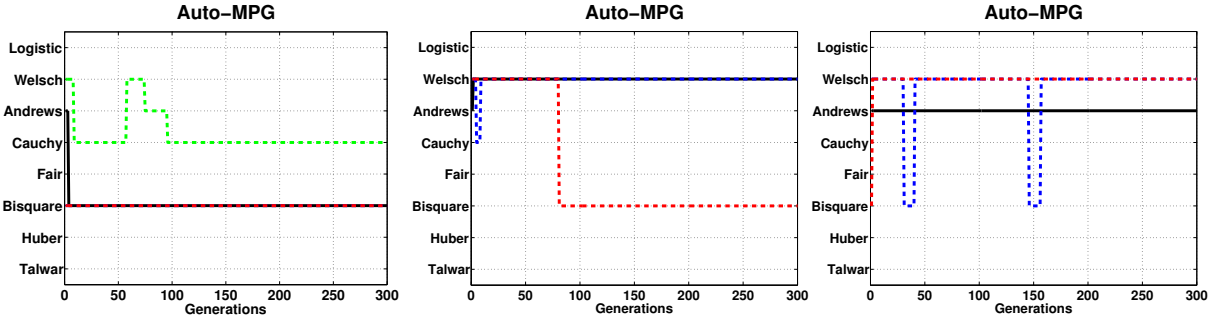


Source: author.

F.2.1.3 Auto-MPG with 1 sided and 30% of contamination

Figure 136 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 1 sided and 30% of outlier contamination.

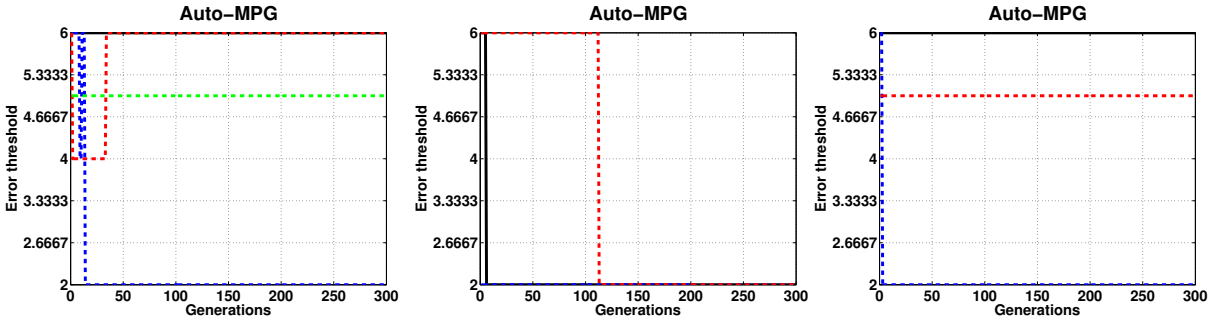
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 135 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 20% of outlier contamination.

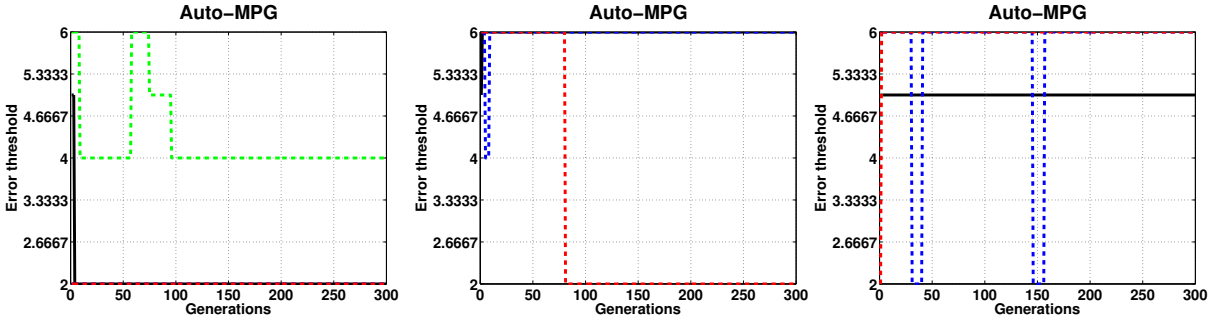
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 137 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

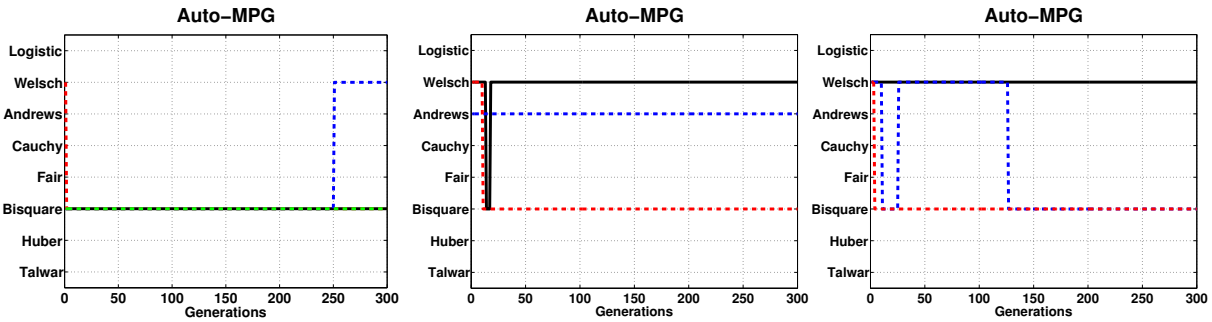


Source: author.

F.2.1.4 Auto-MPG with 1 sided and 40% of contamination

Figure 138 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 1 sided and 40% of outlier contamination.

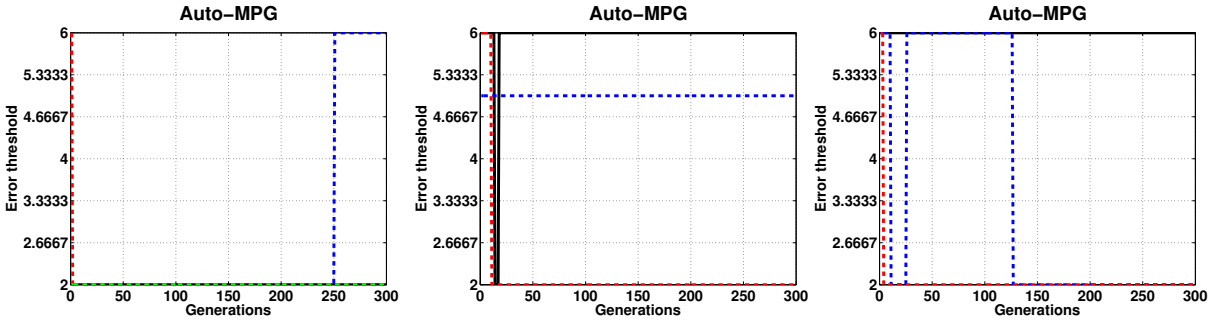
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 139 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

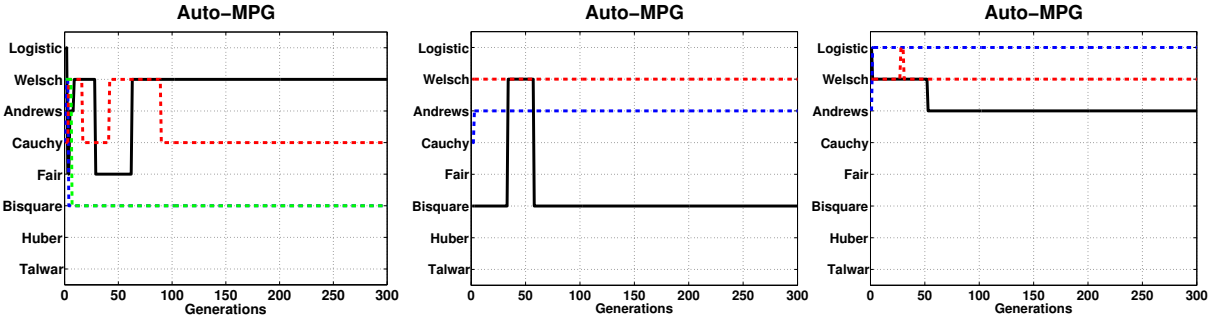


Source: author.

F.2.1.5 Auto-MPG with 2 sided and 10% of contamination

Figure 140 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 2 sided and 10% of outlier contamination.

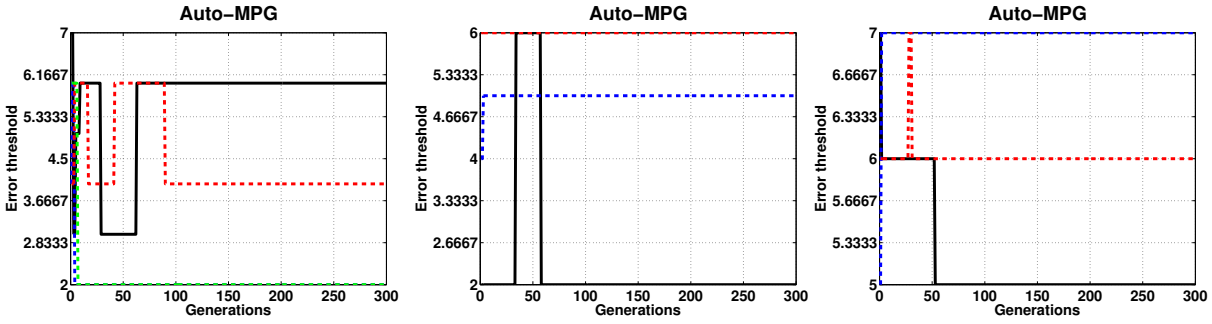
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 141 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

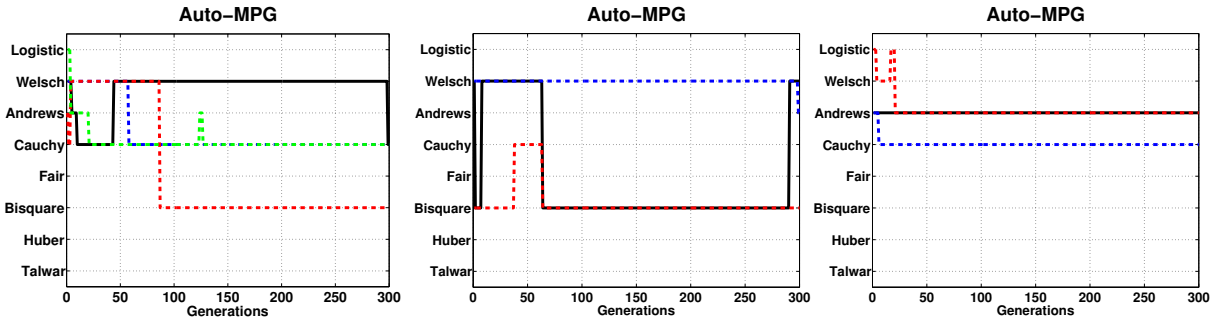


Source: author.

F.2.1.6 Auto-MPG with 2 sided and 20% of contamination

Figure 142 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 2 sided and 20% of outlier contamination.

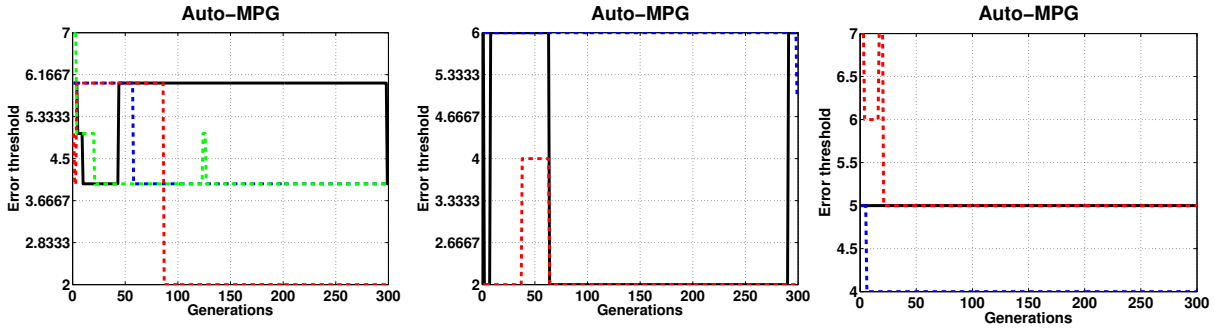
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 143 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

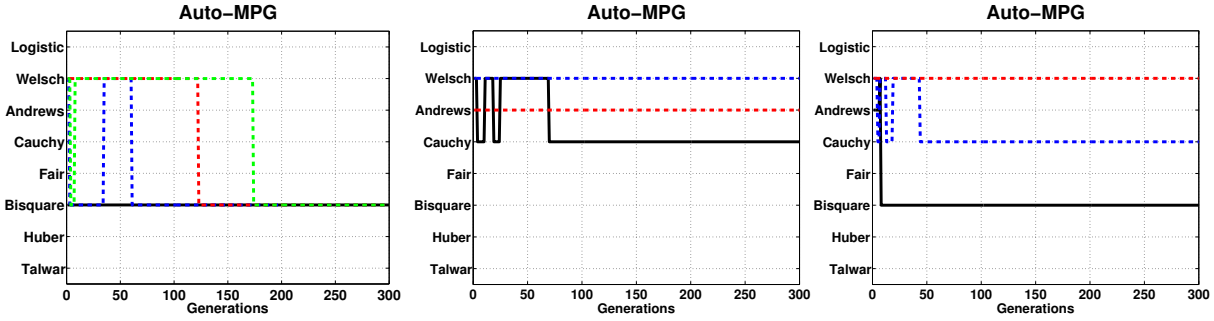


Source: author.

F.2.1.7 Auto-MPG with 2 sided and 30% of contamination

Figure 144 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 2 sided and 30% of outlier contamination.

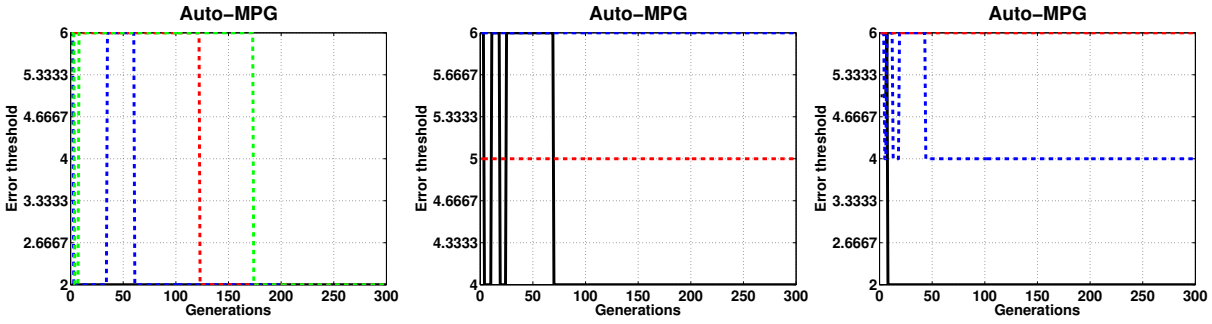
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 145 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

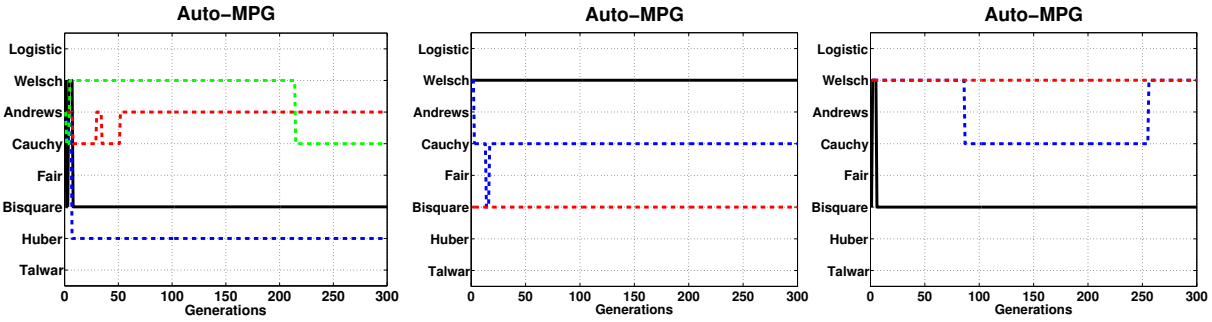


Source: author.

F.2.1.8 Auto-MPG with 2 sided and 40% of contamination

Figure 146 – ANHNA$_{Rm}$-DE weight function choice for Auto-MPG dataset with 2 sided and 40% of outlier contamination.

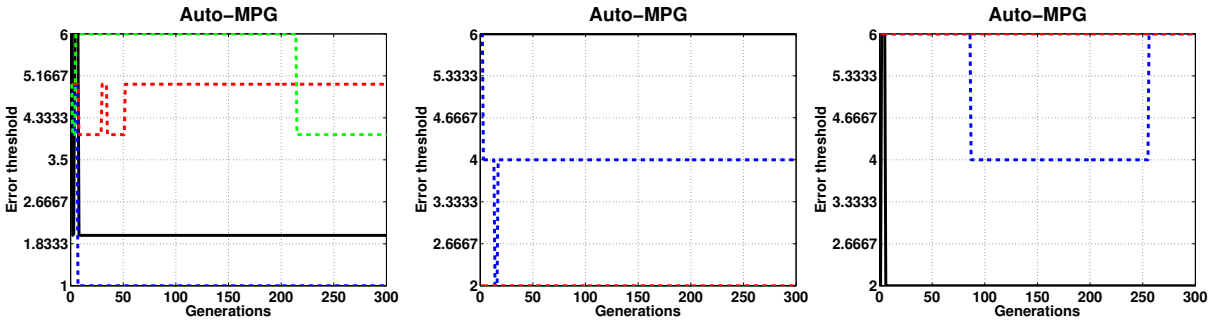
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 147 – ANHNA$_{Rm}$-DE error threshold evolution for Auto-MPG dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



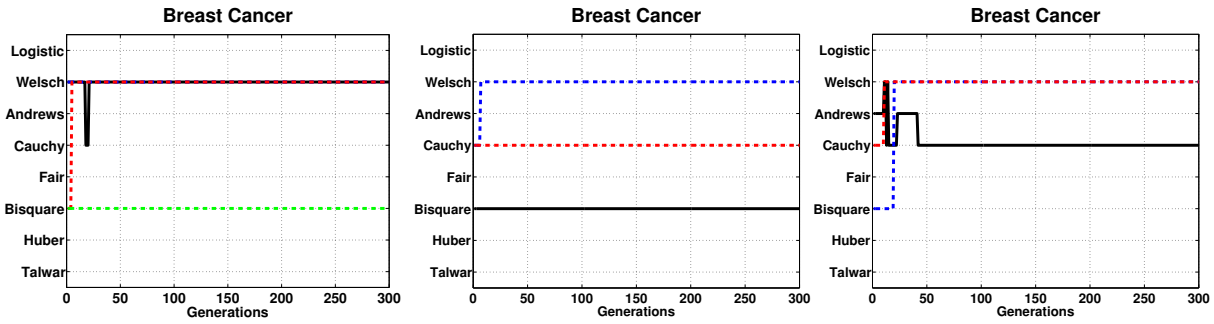
Source: author.

F.2.2 Breast Cancer

F.2.2.1 Breast Cancer with 1 sided and 10% of contamination

Figure 148 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 1 sided and 10% of outlier contamination.

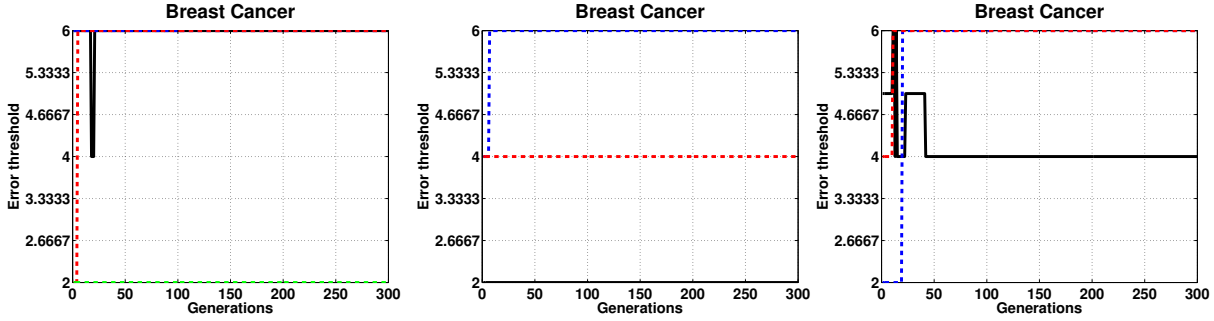
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 149 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

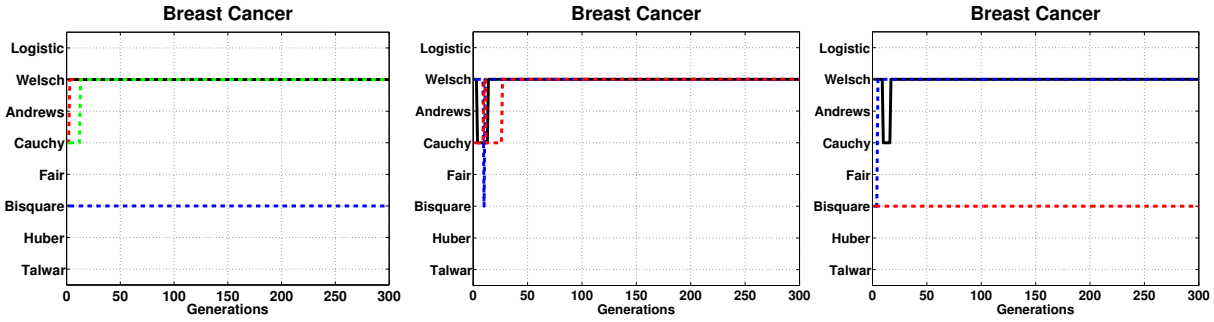


Source: author.

F.2.2.2 Breast Cancer with 1 sided and 20% of contamination

Figure 150 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 1 sided and 20% of outlier contamination.

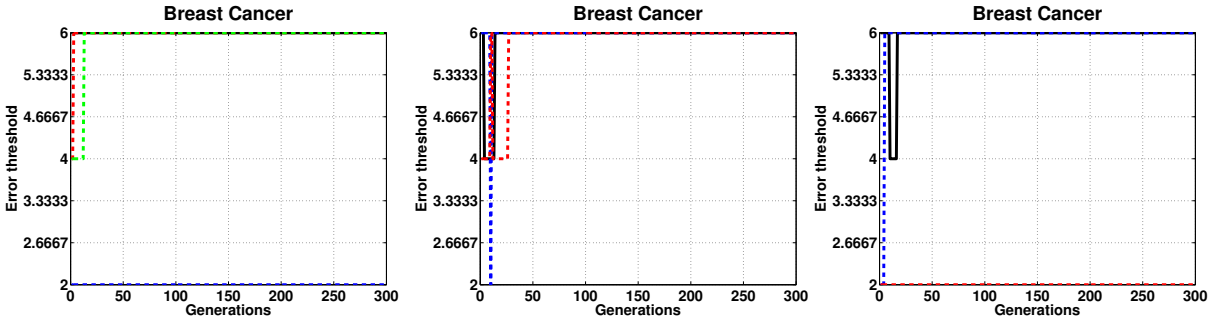
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 151 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

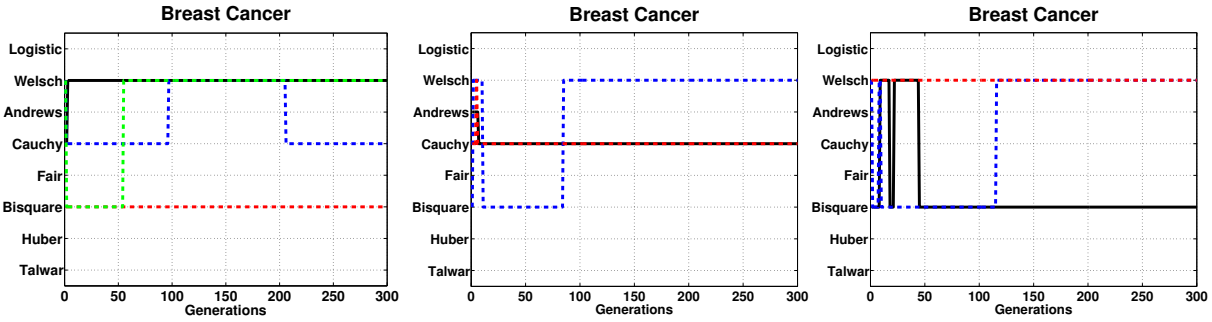


Source: author.

F.2.2.3 Breast Cancer with 1 sided and 30% of contamination

Figure 152 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 1 sided and 30% of outlier contamination.

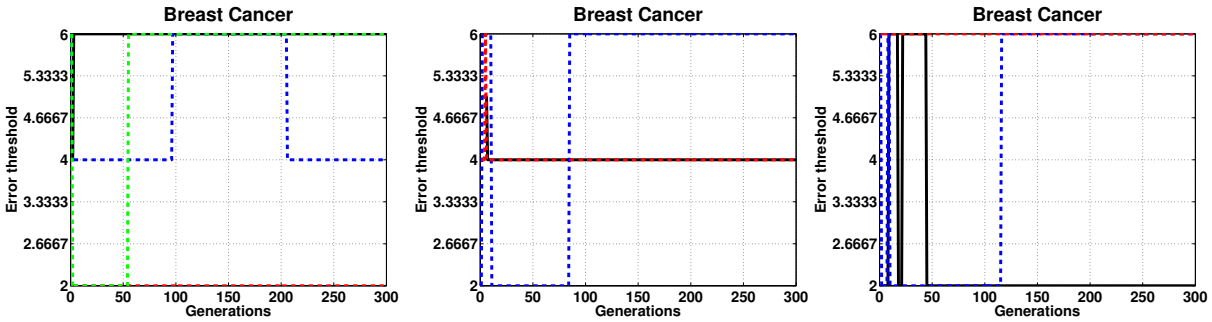
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 153 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

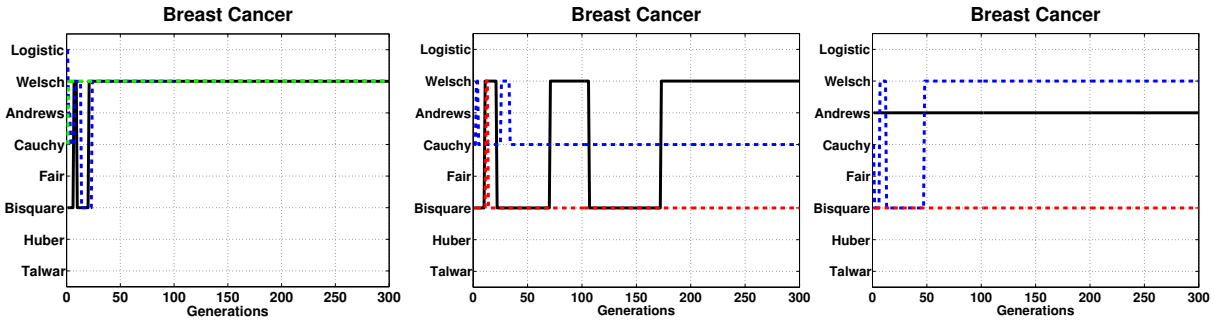


Source: author.

F.2.2.4 Breast Cancer with 1 sided and 40% of contamination

Figure 154 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 1 sided and 40% of outlier contamination.

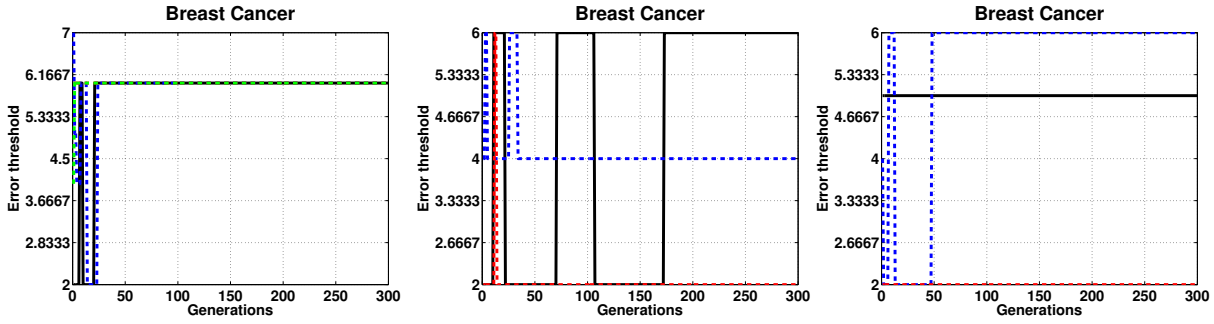
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 155 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

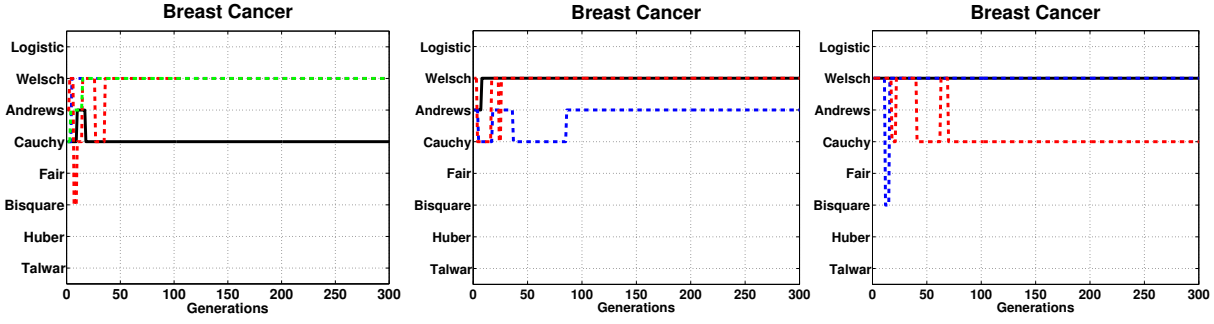


Source: author.

F.2.2.5 Breast Cancer with 2 sided and 10% of contamination

Figure 156 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 2 sided and 10% of outlier contamination.

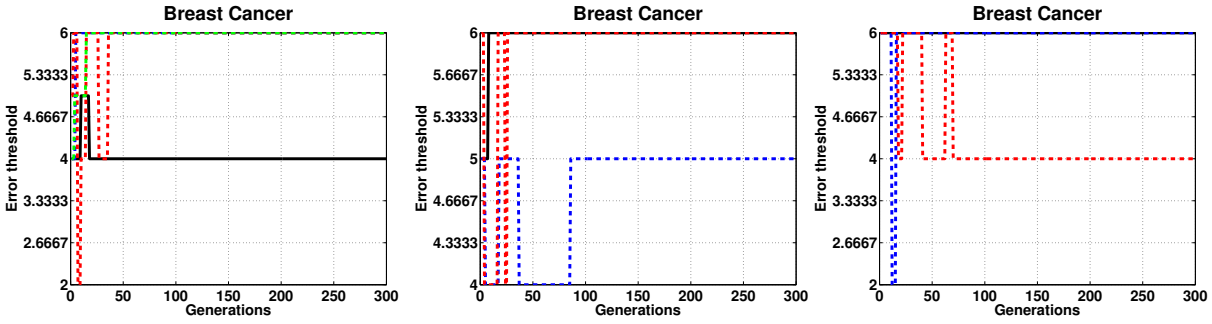
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 157 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

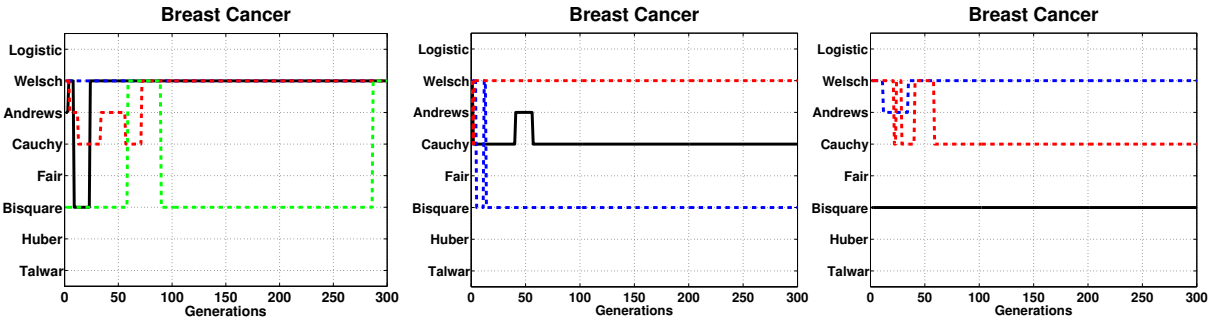


Source: author.

F.2.2.6 Breast Cancer with 2 sided and 20% of contamination

Figure 158 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

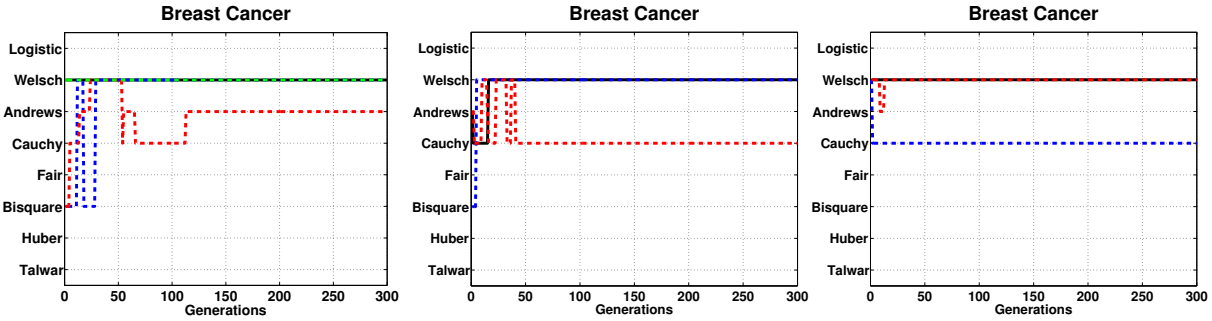


Source: author.

F.2.2.7 Breast Cancer with 2 sided and 30% of contamination

Figure 160 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 2 sided and 30% of outlier contamination.

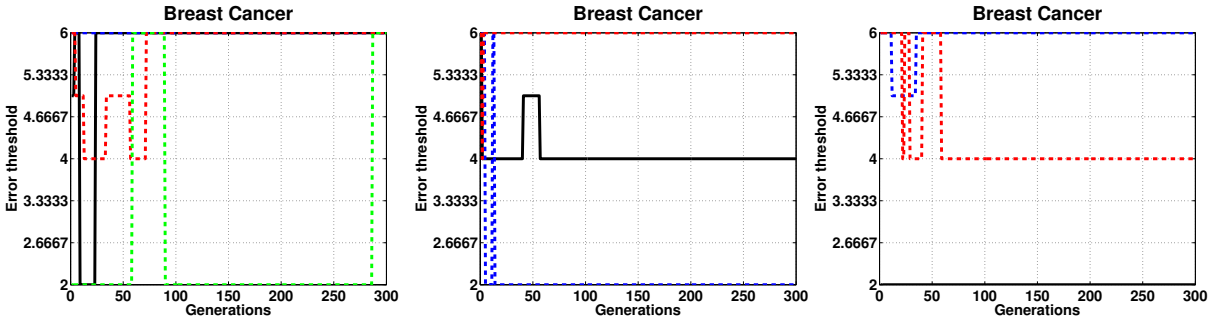
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 159 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 20% of outlier contamination.

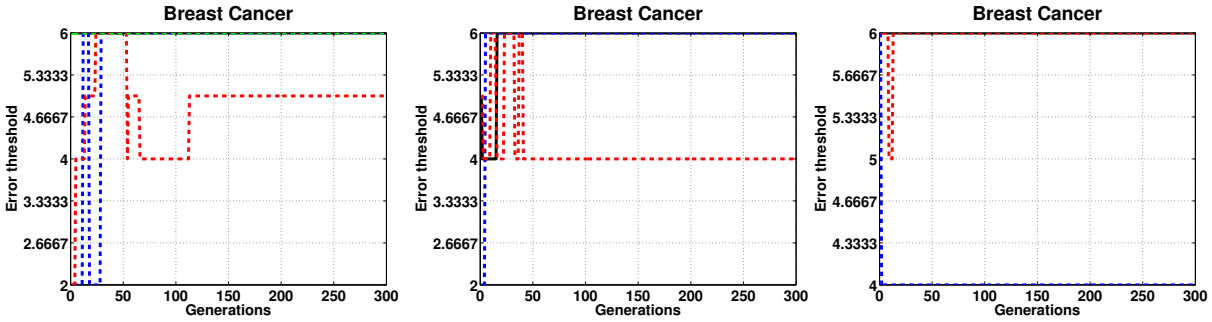
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 161 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

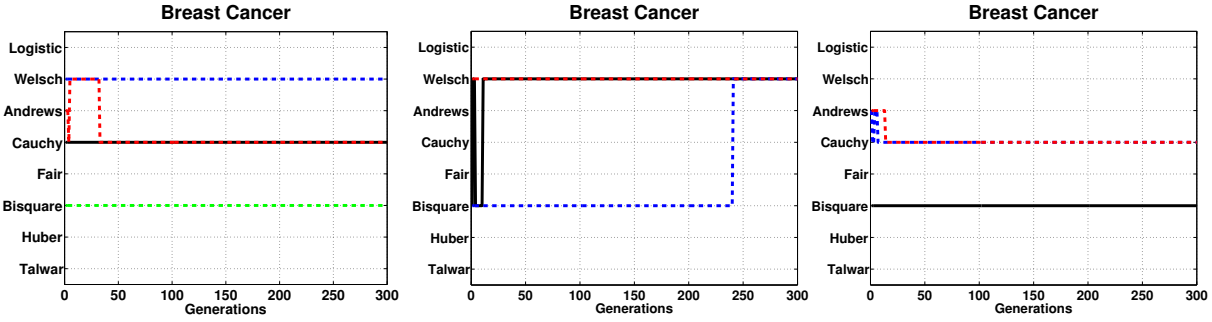


Source: author.

F.2.2.8 Breast Cancer with 2 sided and 40% of contamination

Figure 162 – ANHNA$_{Rm}$-DE weight function choice for Breast Cancer dataset with 2 sided and 40% of outlier contamination.

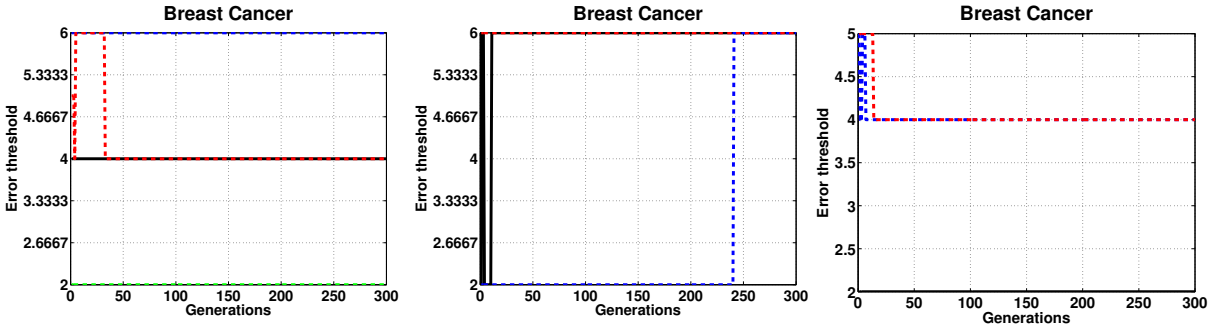
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 163 – ANHNA$_{Rm}$-DE error threshold evolution for Breast Cancer dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



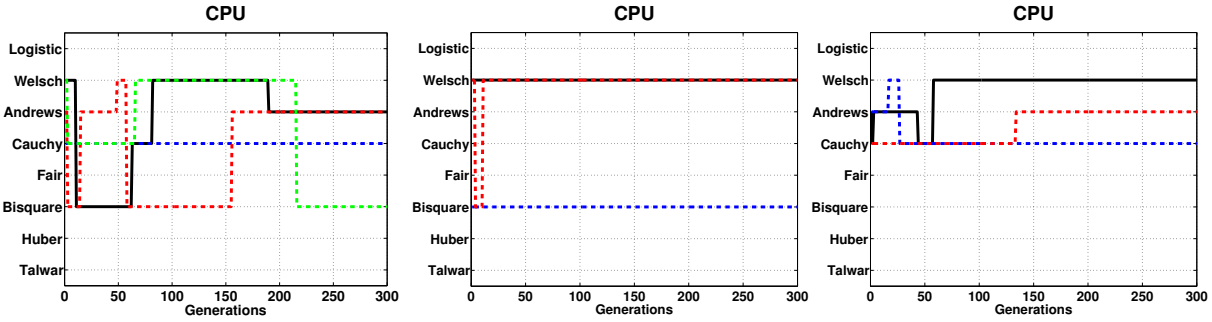
Source: author.

F.2.3 CPU

F.2.3.1 CPU with 1 sided and 10% of contamination

Figure 164 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 1 sided and 10% of outlier contamination.

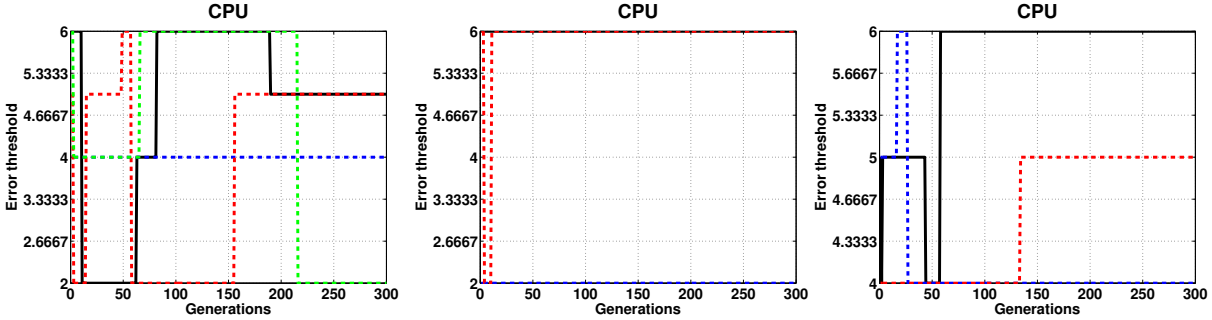
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 165 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

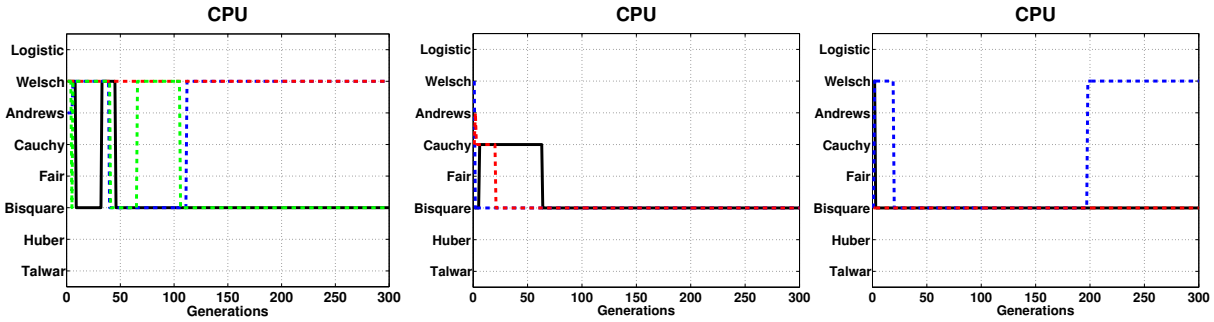


Source: author.

F.2.3.2 CPU with 1 sided and 20% of contamination

Figure 166 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 1 sided and 20% of outlier contamination.

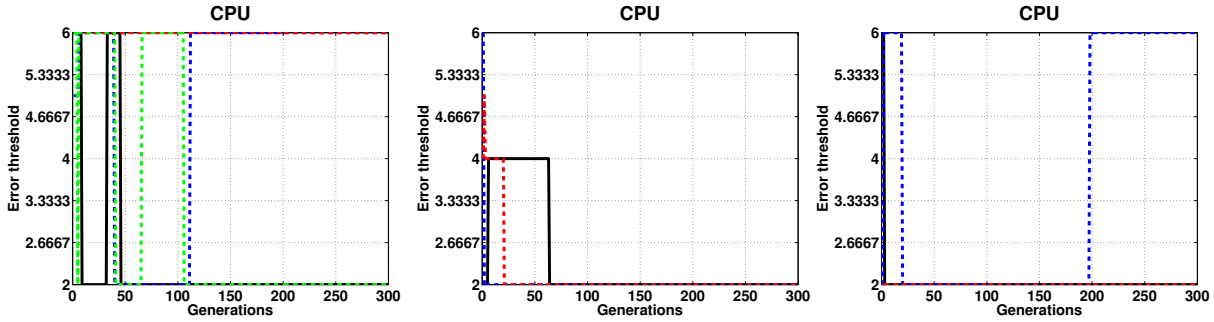
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 167 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

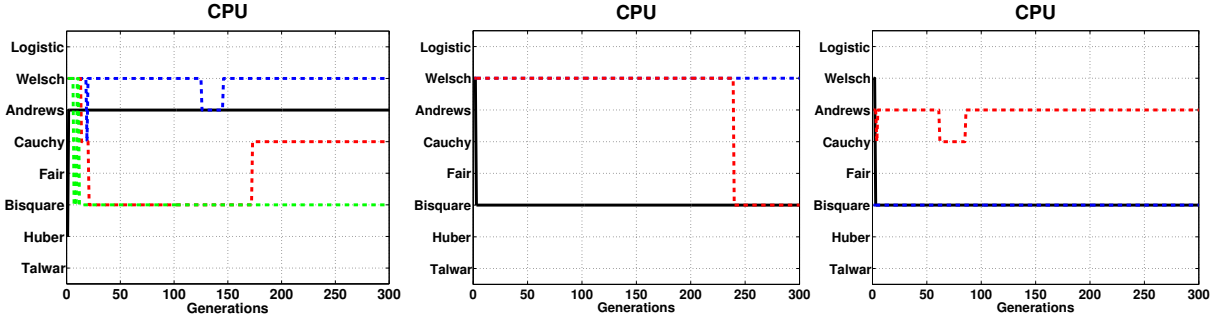


Source: author.

F.2.3.3 CPU with 1 sided and 30% of contamination

Figure 168 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 1 sided and 30% of outlier contamination.

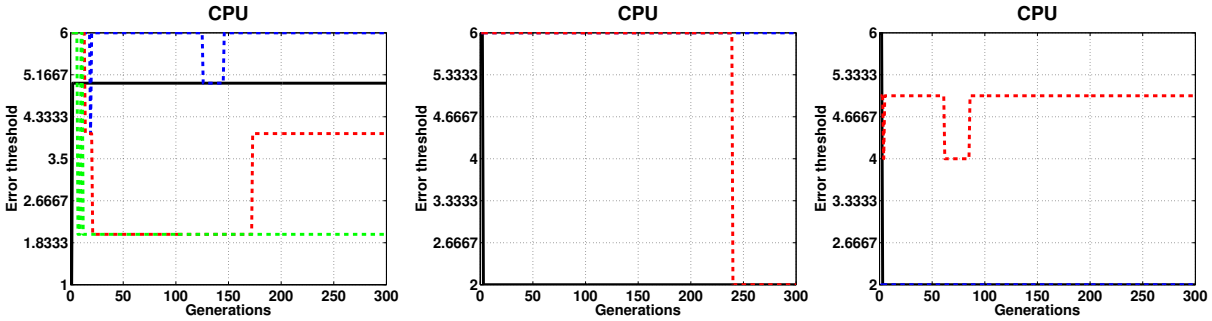
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 169 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

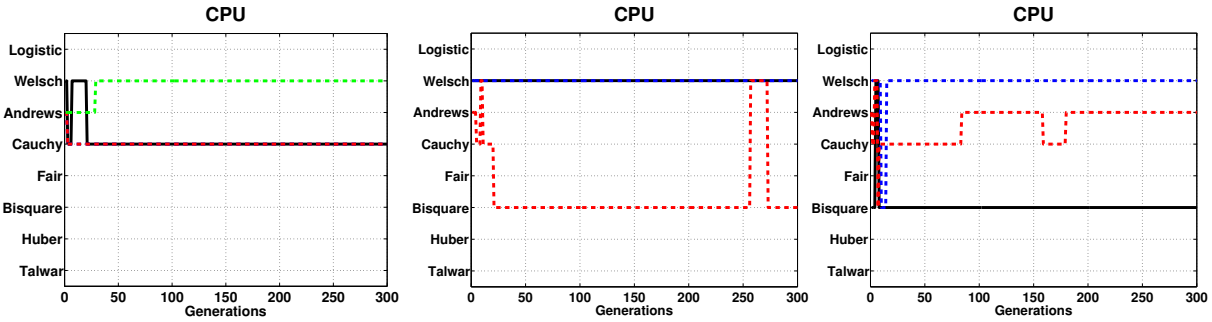


Source: author.

F.2.3.4 CPU with 1 sided and 40% of contamination

Figure 170 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

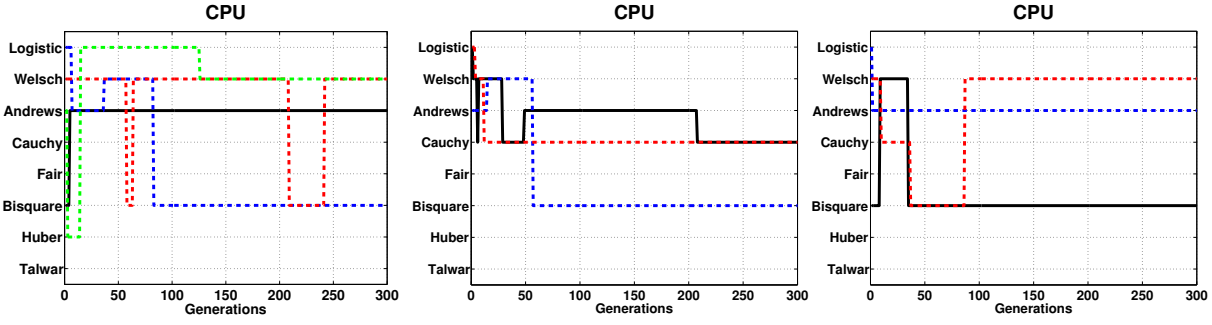


Source: author.

F.2.3.5 CPU with 2 sided and 10% of contamination

Figure 172 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 2 sided and 10% of outlier contamination.

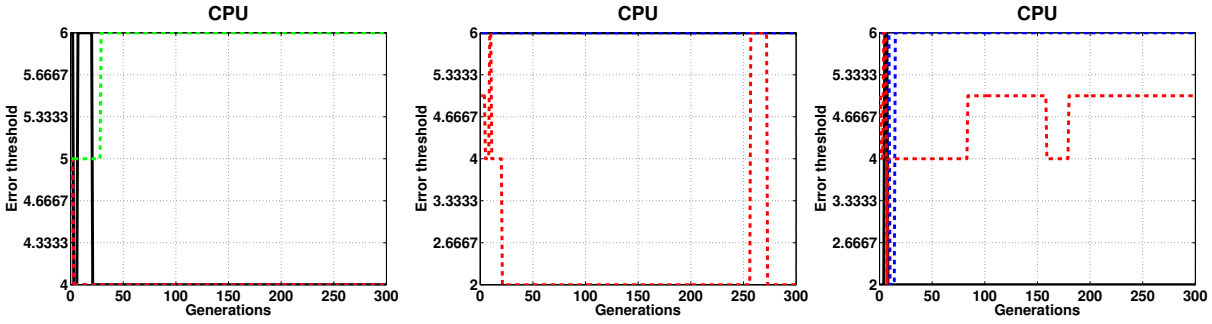
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 171 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 1 sided and 40% of outlier contamination.

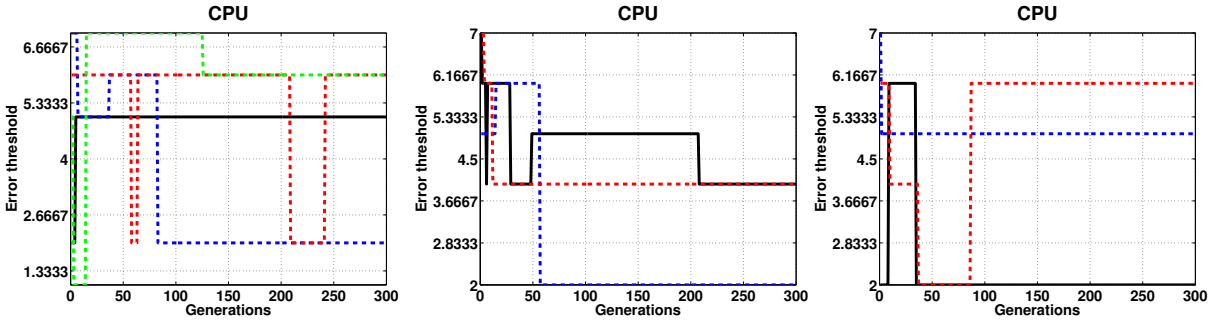
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 173 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

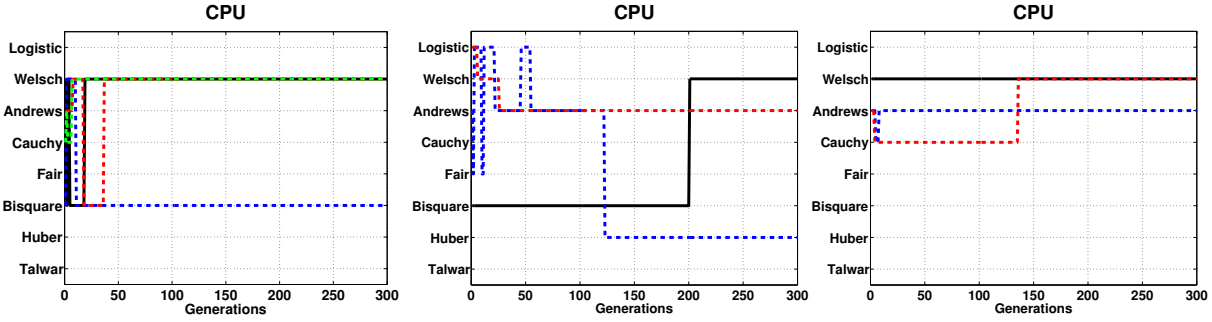


Source: author.

F.2.3.6 CPU with 2 sided and 20% of contamination

Figure 174 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 2 sided and 20% of outlier contamination.

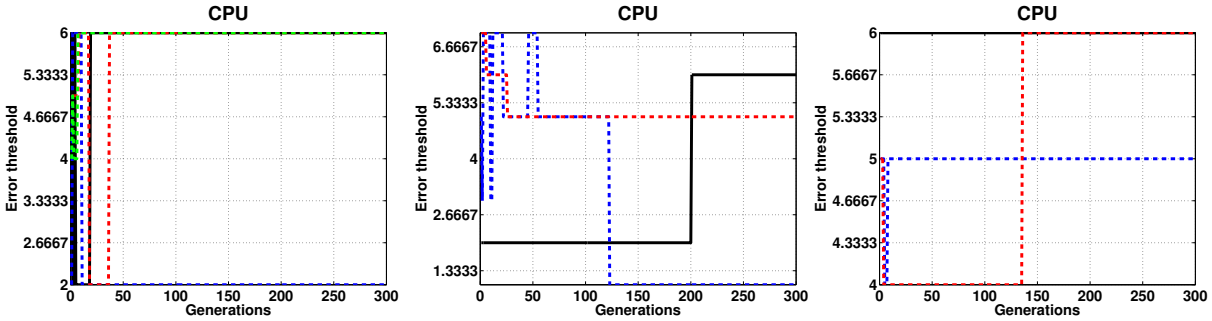
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 175 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

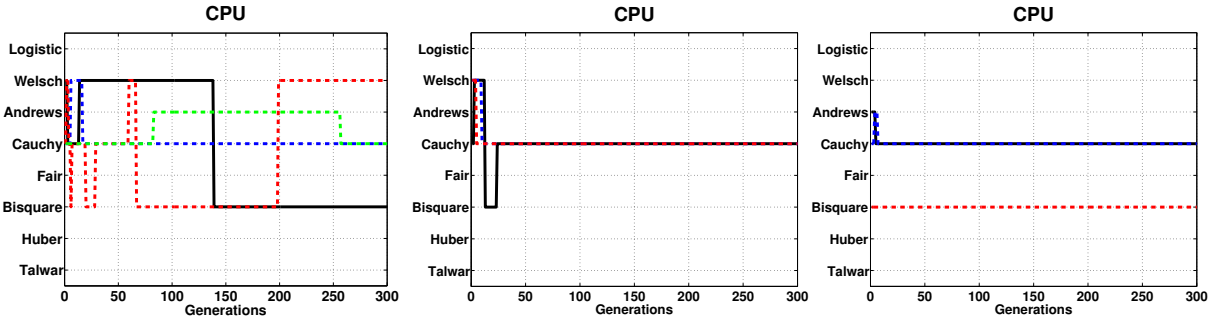


Source: author.

F.2.3.7 CPU with 2 sided and 30% of contamination

Figure 176 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 2 sided and 30% of outlier contamination.

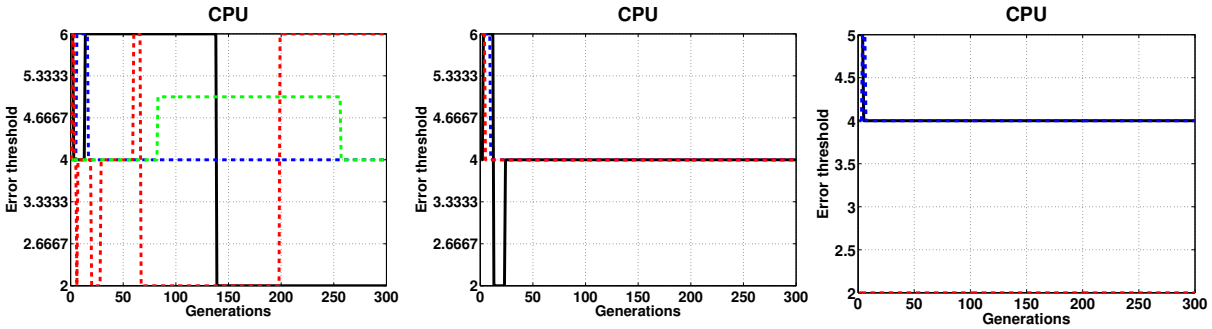
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 177 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

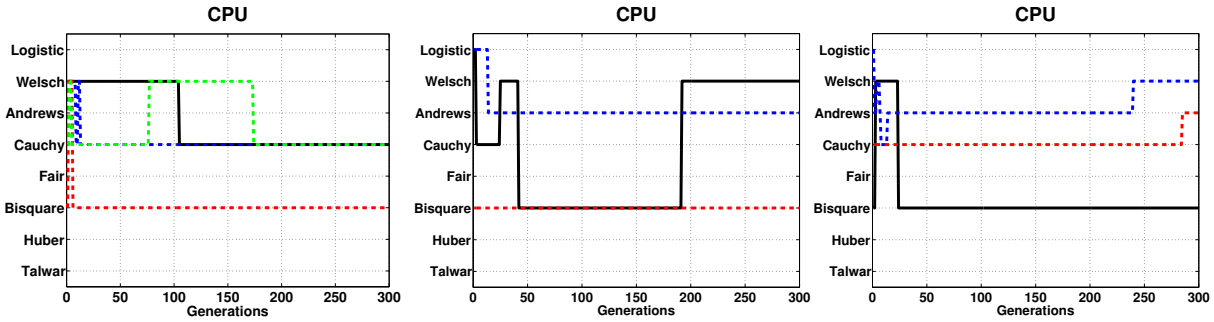


Source: author.

F.2.3.8 CPU with 2 sided and 40% of contamination

Figure 178 – ANHNA$_{Rm}$-DE weight function choice for CPU dataset with 2 sided and 40% of outlier contamination.

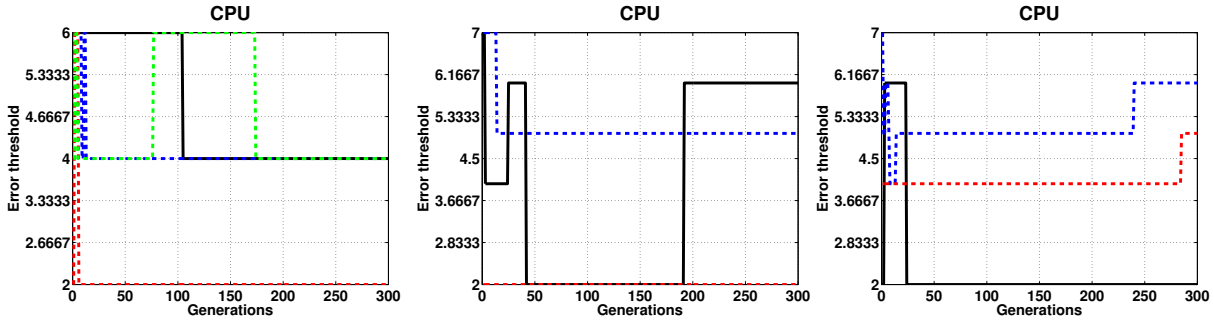
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 179 – ANHNA$_{Rm}$-DE error threshold evolution for CPU dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



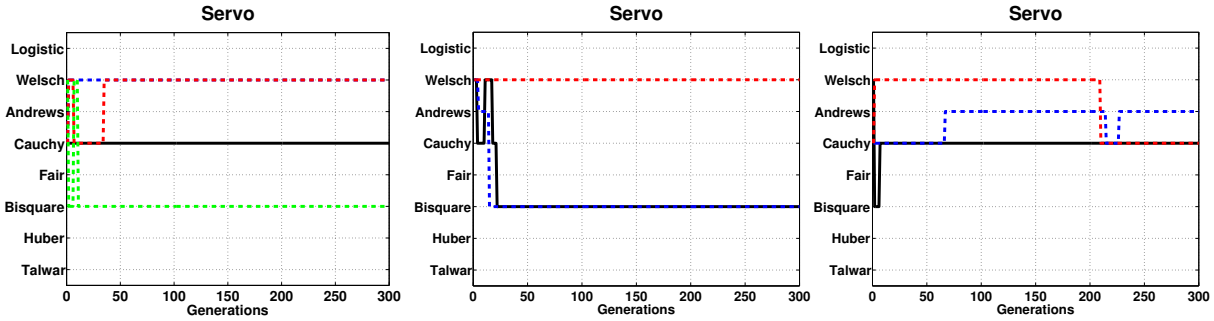
Source: author.

F.2.4 Servo

F.2.4.1 Servo with 1 sided and 10% of contamination

Figure 180 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 1 sided and 10% of outlier contamination.

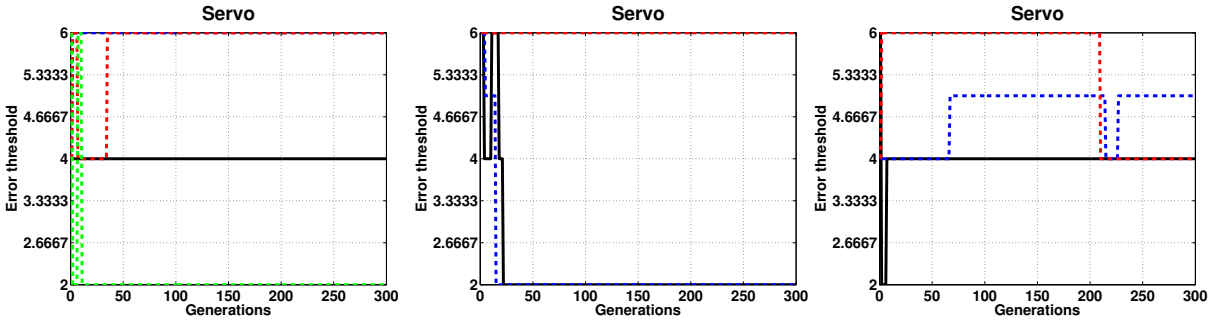
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 181 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 1 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

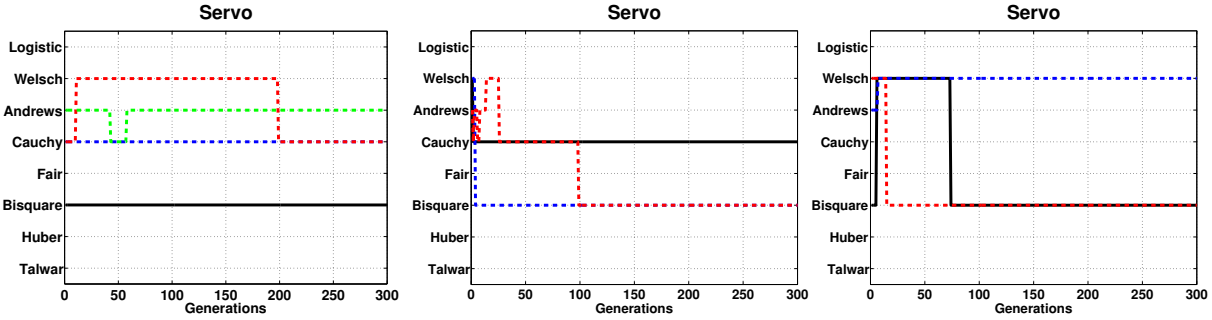


Source: author.

F.2.4.2 Servo with 1 sided and 20% of contamination

Figure 182 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 1 sided and 20% of outlier contamination.

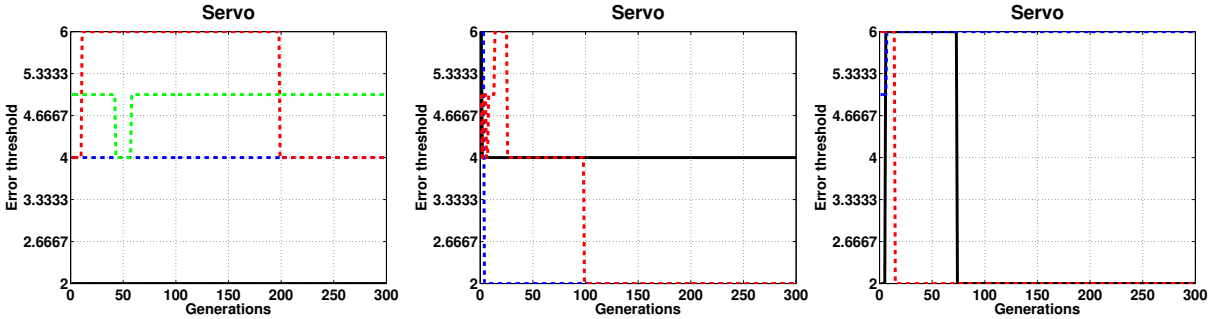
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 183 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 1 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

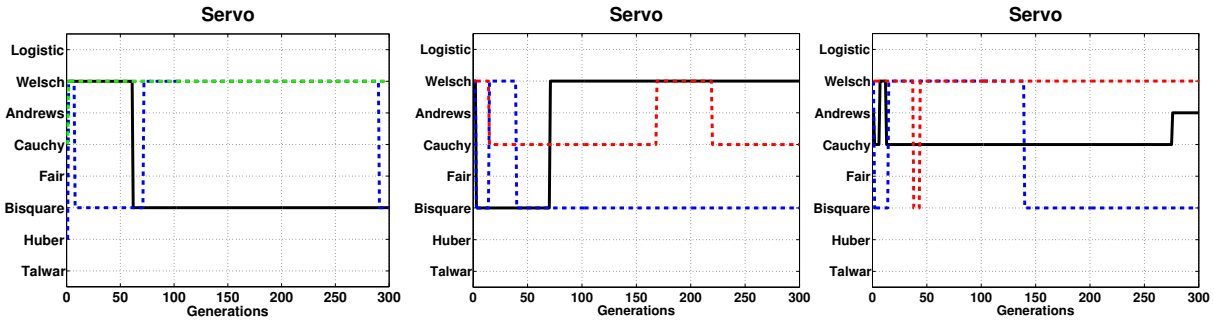


Source: author.

F.2.4.3 Servo with 1 sided and 30% of contamination

Figure 184 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 1 sided and 30% of outlier contamination.

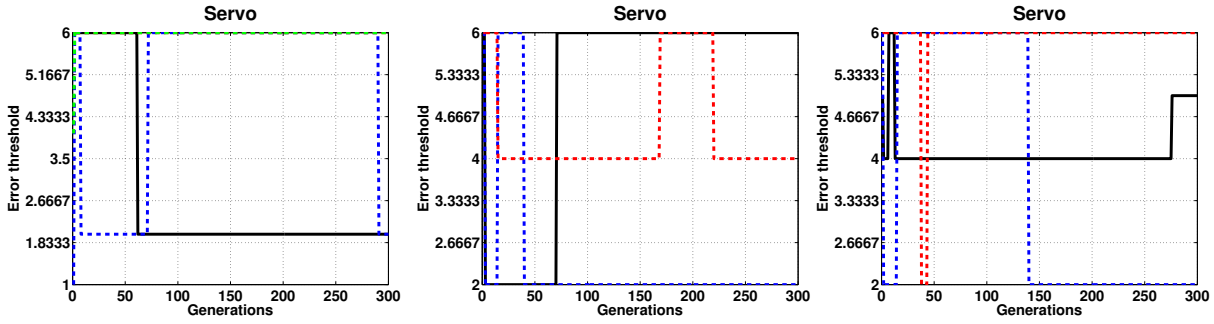
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 185 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 1 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

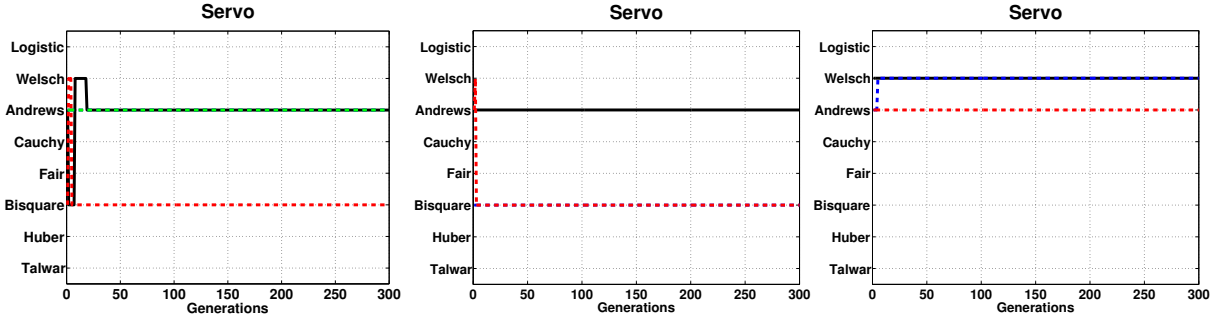


Source: author.

F.2.4.4 Servo with 1 sided and 40% of contamination

Figure 186 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 1 sided and 40% of outlier contamination.

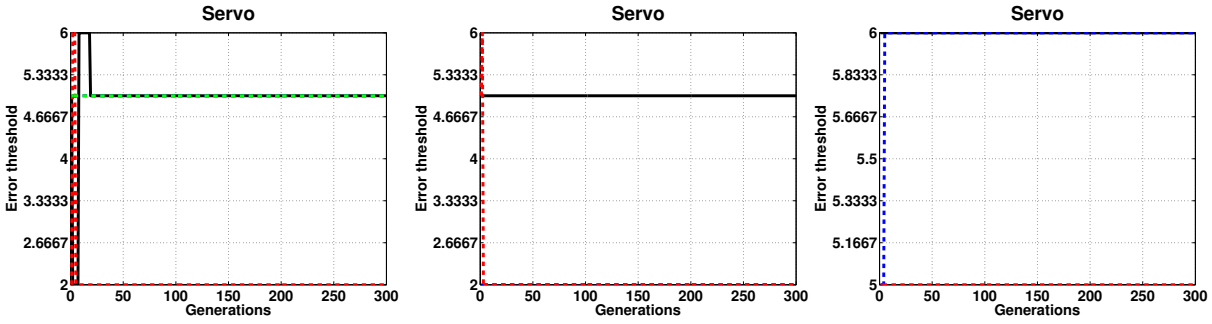
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 187 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 1 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

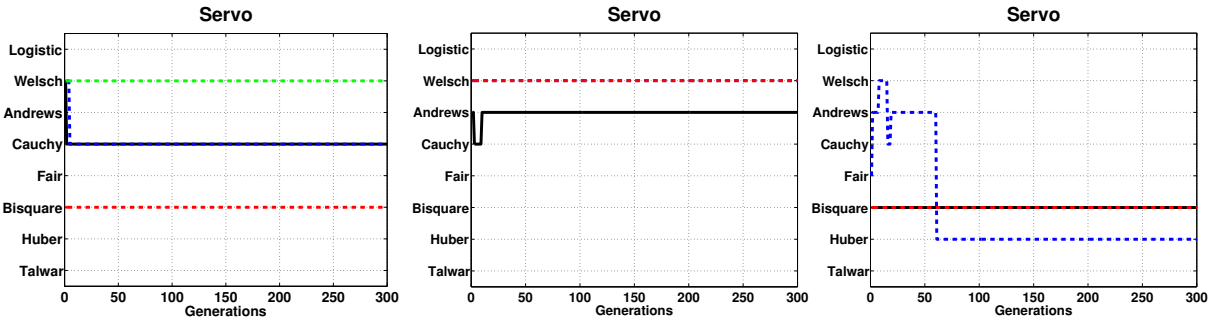


Source: author.

F.2.4.5 Servo with 2 sided and 10% of contamination

Figure 188 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

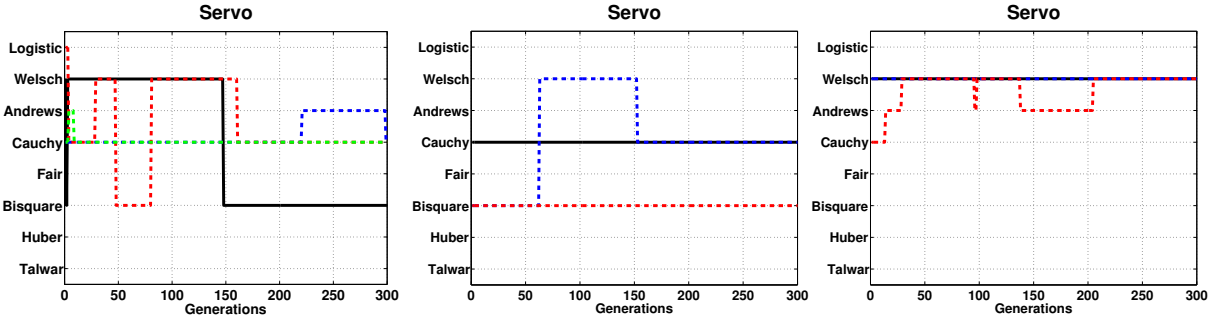


Source: author.

F.2.4.6 Servo with 2 sided and 20% of contamination

Figure 190 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

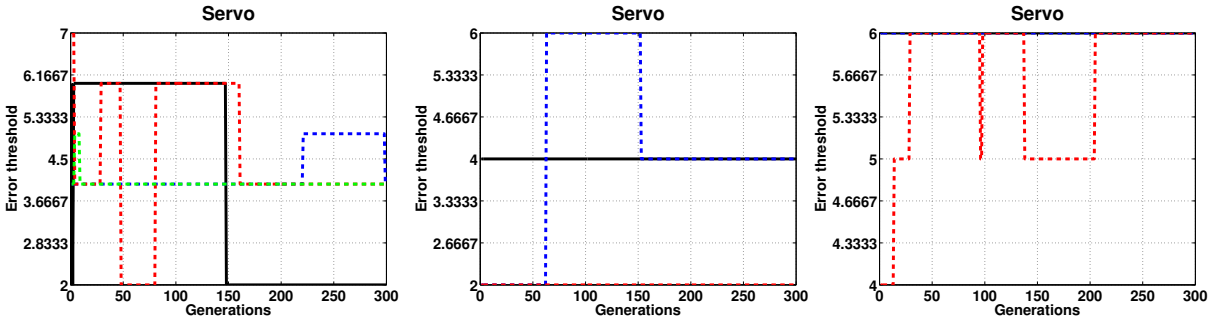
Figure 189 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 2 sided and 10% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

Source: author.

Figure 191 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 2 sided and 20% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

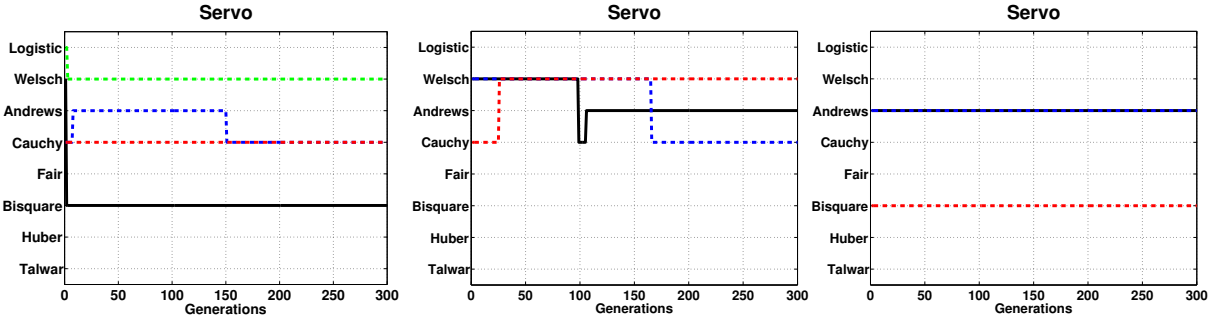


Source: author.

F.2.4.7 Servo with 2 sided and 30% of contamination

Figure 192 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 2 sided and 30% of outlier contamination.

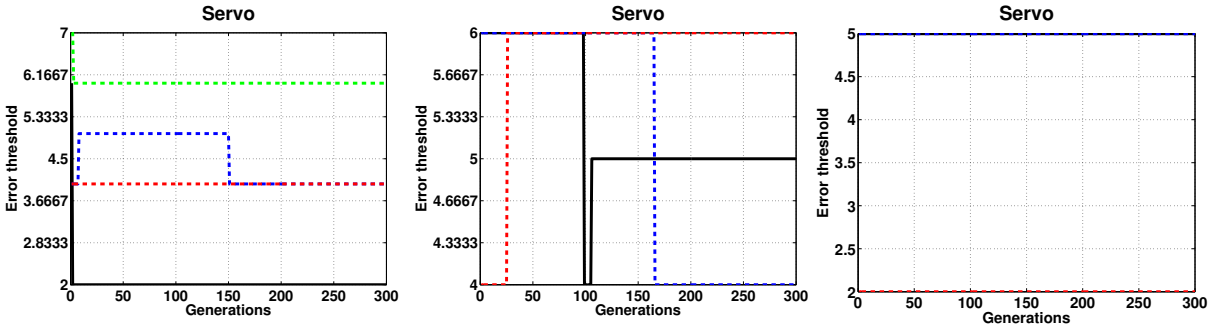
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 193 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 2 sided and 30% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10

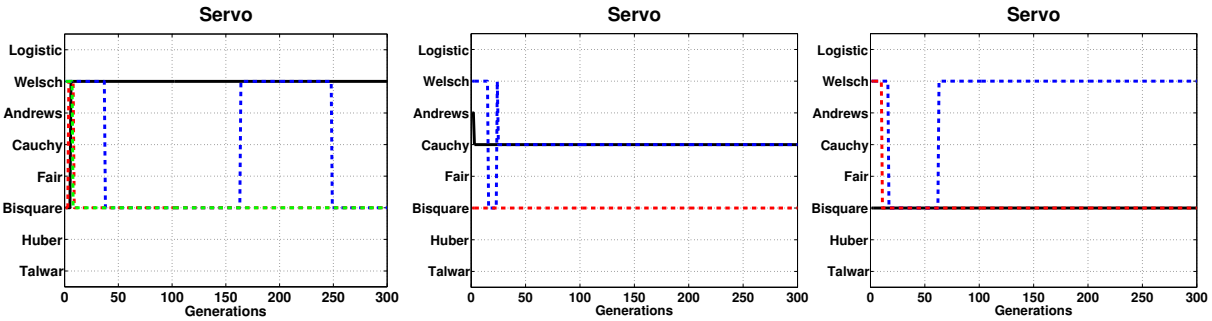


Source: author.

F.2.4.8 Servo with 2 sided and 40% of contamination

Figure 194 – ANHNA$_{Rm}$-DE weight function choice for Servo dataset with 2 sided and 40% of outlier contamination.

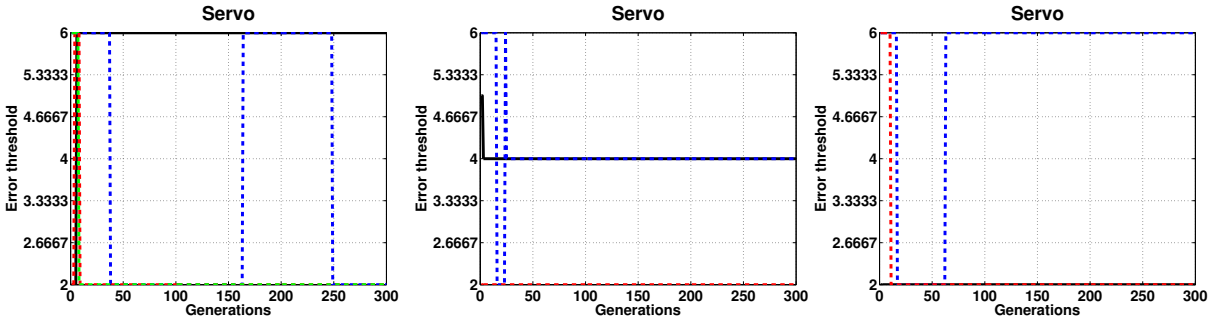
(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.

Figure 195 – ANHNA$_{Rm}$-DE error threshold evolution for Servo dataset with 2 sided and 40% of outlier contamination.

(a) Repetitions 1 to 4 (b) Repetitions 5 to 7 (c) Repetitions 8 to 10



Source: author.