Multi-objective optimization for variance counterbalancing in neural network training

A Thesis

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Dimitra Triantali

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION

IN DATA SCIENCE AND ENGINEERING

University of Ioannina

School of Engineering

Ioannina July 2021

Examining Committee:

- **Konstantinos E. Parsopoulos (Advisor)**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos Blekas**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina
- **Lisimachos P. Kondi**, Professor, Department of Computer Science and Engineering, University of Ioannina

Contents

List of Figures

List of Tables

Abstract

Dimitra Triantali, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, July 2021.
Multi-objective optimization for variance counterbalancing in neural network training.
Advisor: Konstantinos E. Parsopoulos, Associate Professor.

Variance counterbalancing (VCB) is a recently proposed method for large-scale stochastic learning. VCB aims at minimizing both the mean and the standard deviation of the squared error of the neural network over sets of randomly selected mini-batches of the training dataset. For this purpose, gradient-based single-objective optimization methods are typically used, despite the inherent biobjective nature of the underlying minimization problem.

The present thesis studies the use of multi-objective optimization (MO) methods for solving the VCB minimization problem. Both weighted aggregation and Pareto-based evolutionary algorithms are used, including the state-of-the-art random, bang-bang, and dynamic weighted aggregation approaches, as well as the widely used NSGA-II and MOPSO algorithms.

The proposed multi-objective VCB approach is demonstrated on a regression task using RBF neural networks. For this purpose, three different datasets are used. The first two datasets refer to the interpolation of a multimodal real-valued function using accurate and noisy training data, respectively, while the third one refers to the prediction of red-wine quality based on its physicochemical properties. The obtained results are statistically analyzed and compared to the standard VCB method. The analysis suggests that the multi-objective methods can be highly competitive, thereby enhancing the VCB approach. Useful insights regarding the best-performing approaches and the most influential parameters are derived.

Εκτεταμενη Περιληψη

Δήμητρα Τριανταλή, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2021.
Πολυκριτηριακή βελτιστοποίηση για την αντιστάθμιση της διασποράς στην εκπαίδευση νευρωνικών δικτύων.
Επιβλέπων: Κωνσταντίνος Ε. Παρσόπουλος, Αναπληρωτής Καθηγητής.

Η τεχνική της αντιστάθμισης της διασποράς (variance counterbalancing - VCB) προτάθηκε πρόσφατα ως μια εναλλακτική μέθοδος για στοχαστική μάθηση νευρωνικών δικτύων. Η τεχνική VCB στοχεύει στην ελαχιστοποίηση τόσο της μέσης τιμής όσο και της διασποράς του τετραγωνικού σφάλματος του νευρωνικού δικτύου, αξιοποιώντας τυχαία επιλεγμένα υποσύνολα του συνόλου εκπαίδευσης. Για το σκοπό αυτό, χρησιμοποιούνται συνήθως κλασικές μέθοδοι μονοκριτηριακής βελτιστοποίησης με παραγώγους, παρά την παρουσία δύο αντικειμενικών συναρτήσεων στο υποκείμενο πρόβλημα ελαχιστοποίησης.

Η παρούσα διατριβή μελετά τη χρήση πολυκριτηριακών μεθόδων βελτιστοποίησης στην τεχνική VCB. Στη μελέτη χρησιμοποιείται η μέθοδος συνάθροισης των αντικειμενικών συναρτήσεων με βάρη (weighted aggregation), συμπεριλαμβανομένων των προσεγγίσεων random, bang-bang και dynamic, καθώς και Pareto εξελικτικοί αλγόριθμοι, όπως οι NSGA-II και MOPSO.

Οι προτεινόμενες πολυκριτηριακές μέθοδοι εφαρμόζονται πειραματικά σε προβλήματα παλινδρόμησης με χρήση RBF δικτύων. Για το σκοπό αυτό, χρησιμοποιούνται τρία διαφορετικά σύνολα δεδομένων. Τα δύο πρώτα σύνολα αφορούν στην παρεμβολή της πραγματικής συνάρτησης ”Mexican Hat” χρησιμοποιώντας, αντίστοιχα, δεδομένα με και χωρίς θόρυβο, ενώ το τρίτο σύνολο αφορά στην πρόβλεψη της ποιότητας του κόκκινου κρασιού με βάση τις φυσικοχημικές του ιδιότητες. Όλα τα αποτελέσματα αναλύονται στατιστικά και συγκρίνονται με την υπάρχουσα μέθοδο

VCB. Τα αποτελέσματα δείχνουν ότι οι πολυκριτηριακές μέθοδοι είναι εξαιρετικά ανταγωνιστικές ως προς τις κλασικές μεθόδους. Επιπλέον, η στατιστική ανάλυση παρέχει χρήσιμα συμπεράσματα για την απόδοση των αλγορίθμων και την επίδραση των επιμέρους χαρακτηριστικών τους.

Chapter 1

Introduction

1.1 Aims and objectives

A *feedforward neural network* (FNN) is an artificial neural network (ANN) where connections between the nodes do not form a cycle. The FNN was the first and simplest type of neural networks. In this network, the information moves in only one direction, from the input nodes, through the hidden nodes, and to the output nodes. There are two types of FNN, the single-layer, the multi-layer perceptron. The first one is the simplest kind of neural networks, since the inputs are fed directly to the outputs via a series of weights. The second class consists of multiple layers of computational units, usually forward interconnected. This way each neuron in one layer has directed connections to the neurons of the subsequent layer.

The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds for a wide range of activation functions, e.g. for the sigmoid function.

Designing and training an FNN resembles training in different machine learning models. The training consists of the minimization of an error function, such as the

mean squared error (MSE). Once the network is trained, i.e., its weights and structure are fixed, it becomes a deterministic function and it can be used to make predictions on unknown data. However, when the number of training inputs is very large the training becomes time-intensive and learning slows down.

There are three terms involved inseparably in the training of a network, i.e., the input vectors, the epochs and the batch size. A training dataset is comprised of many *input vectors*. An input vector may also be called a pattern, a sample, an instance, an observation, or a feature vector. The second term, the number of *epochs*, defines the iterations of the learning algorithm over the entire training dataset. The *batch size* is another hyperparameter that defines the number of input vectors to work through before updating the internal model parameters. Small values of batch size correspond to a learning process that converges quickly, while large values result in a learning process that converges slowly with more accurate estimates of the error.

Gradient descent (GD) is an optimization algorithm often used for updating the parameters of the neural network by minimizing the underlying error function. GD can vary in terms of the number of training patterns used to calculate error. The three main types of gradient descent are batch, stochastic, and mini-batch.

Batch gradient descent (BGD) is a variation of the GD algorithm that calculates the error for each input vector in the training dataset, but only updates the model after all training vectors have been evaluated. So, all input vectors are used to create one batch. The decreased update frequency of this method results in a more stable error gradient and may result in a more stable convergence. Also, the method admits parallel implementation due to the separation of the calculation of prediction errors and the model update. On the other hand, the more stable error gradient may result in premature convergence of the model to a sub-optimal set of parameters. The most important disadvantage is that this method can become very slow for large datasets.

Stochastic gradient descent (SGD), calculates the error and updates the model for each input vector in the training dataset. The frequent updates of SGD give an insight into the performance of the model and the rate of improvement, while the increased model update frequency can result in faster learning. Also, the noisy update process can allow the model to avoid local minima (e.g. premature convergence). However, updating the model so frequently is computationally expensive, especially on large datasets. Moreover, the noisy learning process may disrupt the algorithm convergence towards a local minimum.

Mini-batch gradient descent (MBGD) is the third variation of the GD algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients. The batch size is more than one input vector and less than the size of the training dataset. MBGD seeks to find a balance between the robustness of SGD and the efficiency of BGD. The model update frequency is higher than BGD, which allows for a more robust convergence, avoiding local minima. Also, MBGD does not require having all training data in memory or in the algorithm implementation. However, mini-batch requires the configuration of an additional mini-batch size hyperparameter for the learning algorithm [1].

There are many extensions to the optimization methods presented above designed to improve the optimization process (same or better error in fewer iterations). Recently, Lagari *et al.* [2] proposed the *variance counterbalancing* (VCB) algorithm, which is a mini-batch algorithm for large-scale stochastic learning. This method minimizes an objective function that comprises the average squared error over a randomly selected set of mini-batches, along with a term that penalizes candidate solutions with large squared error variance. Their experimental analysis makes it clear that the VCB algorithm can overcome the high computational cost of the MSE of the network over the full training set and still lead to fast convergence. Despite the fact that this minimization problem is biobjective, i.e., it requires the minimization of both the average squared error and the corresponding variance, it was treated as a bound-constrained single-objective optimization problem.

The goal of this thesis is to investigate the performance of MO methods in solving the VCB minimization problem respecting its biobjective nature. Following the current state-of-the-art, both weighted aggregation and Pareto-based evolutionary algorithms are used and compared to the VCB method.

The employed testbed for experimentation comprised the application of VCB on RBF networks, due to their approximation capabilities, simple network structure, compact topology, and fast learning. Three different datasets are used for the demonstration of the proposed approaches. The first two datasets refer to the interpolation of the "Mexican Hat" function using accurate and noisy training data, respectively, while the last dataset refers to the prediction of red wine quality based on its acidity values.

All results are statistically analyzed and the NSGA-II and the MOPSO methods, as the best-performing approaches, are compared to the standard VCB method. The

results suggest that MO methods are highly competitive, providing better solutions in terms of the MSE of the network, without exceeding time constraints. Thus, they can be used as a viable alternative to the classic gradient-based algorithms with the VCB technique.

1.2 Structure of the thesis

The rest of the thesis is organized as follows: Chapter 2 contains the necessary background information. This includes the standard VCB algorithm, the general MO framework, and the RBF neural networks. Chapter 3 analyzes the proposed approach, while Chapter 4 is devoted to the experimental results. Chapter 5 concludes the thesis. The Appendix contains all the examined cases, as well as, boxplot representation of all the results.

Chapter 2

Background Information

2.1 Variance counterbalancing

We consider the problem of approximating a continuous function $g : X \subset \mathbb{R}^d \rightarrow \mathbb{R}$, using a neural network. Putting it formally, let S be a training set containing N pattern vectors:

$$S = \{(x_1, y_1), ..., (x_N, y_N)\}\,, \tag{2.1}$$

with $x_i \in X \subset \mathbb{R}^d$, d is the dimension of each pattern vector and $y_i = g(x_i) \in \mathbb{R}$, for all $i = 1, 2, ..., N$. Training is performed by minimizing an error metric over the whole training set S, such as the MSE:

$$E(w) = \frac{1}{N}\sum_{i=1}^{N}(N(x_i, w) - y_i)^2\,, \tag{2.2}$$

where $N(x_i, w)$ is the network's output using the i-th pattern vector x_i; w is the vector of network's parameters; and y_i is the desired output of the network. The minimization of the error function is usually performed using gradient-based optimization methods [3]. However, in large datasets this approach can be very time-consuming.

Stochastic learning ameliorates this problem by consecutively minimizing the network's MSE over randomly selected mini-batches.

Let K be the number of mini-batches of the training set S, with ${S_i}^{(M)}$, $i = 1, ..., K$ denoting the i-th one, containing $M \ll N$ pattern vectors selected at random:

$$ {S_i}^{(M)} = \{(x_{i1}, y_{i1}), ..., (x_{iM}, y_{iM})\} \subset S \,. \tag{2.3} $$

The associated minimization problem is given by:

$$ E_i(w) = \frac{1}{M} \sum_{j=1}^{M} (N(x_{i_j}, w) - y_{i_j})^2 \,. \tag{2.4} $$

A desirable minimizer w^* is one that gives nice generalization properties to the neural network. This means that w^* retains small changes of the MSE among the mini-batches. A convincingly hypothesis derived from this necessity is expressed as [2]:

$$ E(w^*) \approx E_i(w^*), \quad i = 1, 2, \ldots \tag{2.5} $$

Based on this assumption, a suitable objective function $\mathbf{f}$ can be constructed, where the variance term is equal to:

$$ \sigma^2(w^*) = \frac{1}{K} \sum_{i=1}^{K} \left(E_i(w^*) - E(w^*)\right)^2 , \tag{2.6} $$

and it can be used to counterbalance $E(w^*)$. The objective of the problem is then defined as the following vector function:

$$ \mathbf{f}(w^*) = [\, E(w^*) \,,\, \sigma^2(w^*) \,] \,. \tag{2.7} $$

In order to avoid the time-consuming computation of the MSE, $E(w^*)$, over the whole training set, it is suggested to approximate it with the average MSE over all mini-batches [2]:

$$ \bar{E}(w^*) = \frac{1}{K} \sum_{i=1}^{K} E_i(w^*). \tag{2.8} $$

If the corresponding approximate variance is calculated as:

$$ \bar{\sigma}^2(w^*) = \frac{1}{K} \sum_{i=1}^{K} \left(E_i(w^*) - \bar{E}(w^*)\right)^2 , \tag{2.9} $$

then, the vector objective function of the examined minimization problem can be defined as:

$$\bar{\mathbf{f}}(w^*) = [\, \bar{E}(w^*)\,,\, \bar{\sigma}^2(w^*)\,]\,. \tag{2.10}$$

2.2 Multi-objective optimization

A *multi-objective optimization* (MO) problem (also called multi-criteria, vector optimization or multi-performance problem) is defined as the problem of finding a vector of decision variables that satisfies constraints and optimizes a vector function whose components represent a number of objective functions. These functions form a mathematical description of performance criteria, which are usually in conflict with each other. Hence, the term "optimize" means finding a solution that would give the values of all the objective functions that are acceptable to the decision maker [4].

Let the MO problem at hand comprise M objective functions. These functions form a vector function defined as:

$$\mathbf{f}(x) = [\, f_1(x)\,,\, f_2(x)\,, ...,\, f_M(x)\,]. \tag{2.11}$$

Let also W contain all feasible decision vectors, i.e., it is the search space under consideration.

In most optimization problems there are restrictions imposed by its particular characteristics or the available resources. Such restrictions must be satisfied to consider a solution as being acceptable. All these restrictions are called *constraints* and they describe dependencies among decision variables and constants (or parameters) involved in the problem. The most common approach to handle constraints is the use of penalty functions. The idea of penalty functions is to transform a constrained optimization problem into an unconstrained one by adding (or subtracting) a certain value to (from) the objective function based on the amount of constraint violation present in a certain solution [4].

2.2.1 Pareto optimality

Minimization of vector functions gives rise to the concept of Pareto optimality.

Definition 2.1 (Pareto optimal). A decision vector $x^* \in W$ is Pareto optimal if there is no other vector $x \in W$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, .., M$, and $f_j(x) < f_j(x^*)$ for at least one $j \in \{1, .., M\}$. ■

In simple words, x^* is Pareto optimal if there exists no feasible vector $x \in W$ that would decrease one objective function without simultaneously increasing at least one of the rest of the objective functions [4].

Definition 2.2 (Pareto dominance). A vector $u = (u_1, ..., u_M)$ is said to dominate a vector $v = (v_1, ..., v_M)$ (denoted as $u \preceq v$) if and only if u is partially less than v, i.e., $\forall\ i \in \{1, .., M\}, u_i \leq v_i$ and $\exists\ j \in \{1, .., M\}$ such that $u_i < v_i$. ■

The concept of Pareto dominance implies that, for a solution to dominate another one, it should not be worse in any objective, while it must be strictly better in at least one of them. Consequently, when comparing two solutions A and B, using Pareto dominance, there are three possible outcomes: A dominates B, B dominates A, or A and B are incomparable, i.e., not dominated by each other. This concept defines a set of solutions called the *Pareto optimal set*. The vectors x^* corresponding to solutions included in the Pareto optimal set are called *non-dominated* [4]:

Definition 2.3 (Pareto optimal set). For a given MO problem $\mathbf{f}(x)$, the Pareto optimal set is defined as: $P^* = \{\ x \in W \mid \nexists\ x' \in W \quad \mathbf{f}(x') \preceq \mathbf{f}(x)\ \}$. ■

The image of the Pareto optimal set is called the *Pareto front*:

Definition 2.4 (Pareto front). For a given MO problem $\mathbf{f}(x)$ and Pareto optimal set P^*, the Pareto front is defined as: $PF^* = \{\ \mathbf{f} = [\ f_1(x)\ , ...,\ f_M(x)\ ] \mid x \in P^*\ \}$. ■

In general, it is impossible to find an analytical expression that represents the hyper-surface corresponding to the Pareto front [4].

2.2.2 Classification of multi-objective optimization methods

MO theory remained practically unexplored until the 1960s, when its mathematical foundations were consolidated. Since then, a plethora of algorithms have been developed. There have been several attempts to classify MO algorithms currently in use. Cohon and Marks [4] distinguished two stages in which, the solution of an MO problem can be divided: (i) the optimization of objective functions involved, and (ii) the process of deciding from the decision maker's perspective. They proposed the following popular classification of MO methods:

(a) Generating techniques (*search* $\Rightarrow$ *decide*).
Those approaches (such as the global criterion method and goal programming) assume that either a certain desired achievable goal or a certain pre-ordering of the objectives can be performed by the decision maker prior to the search.

(b) Techniques that rely on prior preferences (*decide* $\Rightarrow$ *search*).
These techniques (such as linear combination of weights and ϵ-constraint method) do not require prior preference information from the decision maker.

(c) Techniques that rely on progressive articulation of preferences (*decide* $\Leftrightarrow$ *search*).
These techniques (such as probabilistic trade-off development method and STEP method) normally operate in three stages. First, they find a non-dominated solution, they get the reaction of the decision maker regarding this non−dominated solution and modify the preferences of the objectives, accordingly. Finally, they repeat the two previous steps until the decision maker is satisfied or no further improvement is possible.

Up-to-date, there are various mathematical programming methods for solving non-linear MO problems. However, they have several limitations. For example, some of them require that the objectives (and the constraints) are differentiable. Other approaches are inapplicable to disconnected or non-convex Pareto fronts. Additionally, most of them generate a single solution per run of the algorithm. These weaknesses have motivated the use of metaheuristics.

A *metaheuristic* is a high-level search procedure that applies a set of rules based on some source of knowledge, in order to explore the search space more efficiently. A metaheuristic allows the generation of several elements of the Pareto optimal set in a single run. Also, they require only minor information of the objective functions (e.g., no derivatives are required, and they are less susceptible to the shape or continuity of the Pareto front). Metaheuristics have been established as efficient solvers in cases traditional algorithms are either inapplicable or ineffective, and they are currently widely used [4].

2.2.3 Weighted-aggregation methods

A straightforward approach for addressing MO problems transforms the problem to a single-objective one by aggregating all objective functions to a single one:

$$F(x,k) = \sum_{i=1}^{M} w_i(k)\, f_i(x), \tag{2.12}$$

where $w_i(k)$ are the non-negative weights, $i = 1, ..., M$, assuming that:

$$\sum_{i=1}^{M} w_i = 1. \tag{2.13}$$

The global minimizer of the derived function is a Pareto optimal point of the MO problem, otherwise there would exist a feasible solution that improves at least one of the objectives without increasing the others, hence, producing a smaller value of the weighted sum.

The most simple form of the weighted aggregation approach is the *conventional method*. In this case, the weights are fixed during the run. Using this approach, only a single Pareto optimal solution can be obtained per run, while *a priori* knowledge of the search space is required to choose the appropriate weights. Moreover, the procedure shall be repeated several times to obtain the desired number of Pareto optimal points. This is inefficient in most real-world problems. Also, this approach is unable to detect solutions in concave regions of the Pareto front [5].

On the other hand, if the weights are changing during the run, the optimizer can go through diverse points of the Pareto front. Changing the weights combination during the optimization process resembles rotating the coordinate system together with the Pareto front. Thus, when one weight decreases and the other increases, it is equivalent to rotating the coordinate system counter clockwise. Since the weights are non-negative, the maximal rotation angle is 90 degrees [5].

In the following paragraphs, we will focus on the biobjective case, which is the case of our problem of interest, as we explained earlier. There are various dynamic weighted aggregation variants in MO literature. The most popular ones are the following:

(a) *Random weighted aggregation (Random WA)*

In this case w_1 is randomly selected between 0 and 1 while w_2 is its complement:

$$w_1(t) = rand()\,,\; w_2(t) = 1 - w_1(t), \tag{2.14}$$

where t stands for the iteration of the solver used for minimizing the function of Eq.(2.12).

(b) *Bang-bang weighted aggregation (Bang WA)*

In this case, the Pareto front is abruptly rotated by 90 degrees. This means that w_1 is changed from 0 to 1, and w_2 from 1 to 0, as follows:

$$w_1(t) = sign\left(\sin\left(\frac{2\pi t}{33}\right)\right),\ w_2(t) = 1 - w_1(t). \tag{2.15}$$

This abrupt change is due to the usage of the $sign(.)$ function, which is defined as follows:

$$sign(x) = \begin{cases} -1, & \text{if } x < 0, \\ 0, & \text{if } x = 0, \\ 1, & \text{if } x > 0. \end{cases} \tag{2.16}$$

(c) *Dynamic weighted aggregation (Dynamic WA)*

In this case the Pareto front is gently rotated by 90 degrees. This means that w_1 is changed gradually from 0 to 1, and w_2 from 1 to 0, as follows:

$$w_1(t) = \left|\sin\left(\frac{2\pi t}{66}\right)\right|,\ w_2(t) = 1 - w_1(t). \tag{2.17}$$

The values 33 and 66 of the bang-bang and the dynamic wa approaches are the change frequency of the weights. These values were selected based on the number of iterations of the solver [6].

Regardless of the selected weighted aggregation scheme, a single-objective optimizer is required for the minimization of the objective function of Eq.(2.12). In [6], the popular *Particle Swarm Optimization* (PSO) algorithm is suggested. PSO is a population-based algorithm that simulates particle move. A group of individuals, referred to as *particles*, form the *swarm*. The position of each particle is a candidate solution. The particles interact and cooperate with each other to move toward better solutions.

Let a randomly generated population (within the given bounds) of size N that moves within a D-dimensional space, and M objective functions. The i-th particle at iteration t has a current *position*:

$$x_i^{(t)} = (x_{i1}, x_{i2}, ..., x_{iD}), \tag{2.18}$$

and it moves in the search space using an adaptable *velocity*:

$$v_i^{(t)} = (v_{i1}, v_{i2}, ..., v_{iD}). \tag{2.19}$$

The best position achieved so far by the particle is denoted as:

$$p_i^{(t)} = (p_{i1}, p_{i2}, ..., p_{iD}), \tag{2.20}$$

while the best solution achieved so far by the whole swarm is given as:

$$p_g^{(t)} = (p_{g1}, p_{g2}, ..., p_{gD}), \tag{2.21}$$

indicating the position where the smallest fitness has been achieved so far. The velocity and the position of the i-th particle at the next iteration $(t+1)$ are calculated according to the following equations:

$$v_i^{(t+1)} = w\, v_i^{(t)} + c_1\, rand()\, (p_i^{(t)} - x_i^{(t)}) + c_2\, rand()\, (p_g^{(t)} - x_i^{(t)}), \tag{2.22}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}, \tag{2.23}$$

where w is the inertia factor; c_1 and c_2 are two positive constants, called cognitive learning rate and social learning rate respectively; and $rand()$ is a vector of D random values in the range [0,1]. The parameters are set to their default values, $w = 0.729$, $c_1 = c_2 = 1.49$, as suggested in the theoretical analysis of PSO. In addition, a constant, v_{max}, is used to limit the velocities of particles [7].

2.2.4 Non-sorting genetic algorithm

Non-sorting genetic algorithm II (NSGA-II) is among the most popular MO algorithms [4]. It has two mechanisms that distinguish it from other MO algorithms. The first one is the fast non-dominated sorting of the population, and the second one is diversity preservation.

Let a randomly generated population of size N that probes the D-dimensional search space of a problem of M objectives. The i-th individual at iteration t has a current *position*:

$$x_i^{(t)} = (x_{i1}, x_{i2}, ..., x_{iD}), \tag{2.24}$$

which is a candidate solution of the problem. In NSGA-II, we firstly sort the population into different non-domination levels (*fronts*) F_i, where i is the front counter. For

each solution of the problem we calculate the domination count, which is the number of solutions that dominate it, and find the set of solutions that are dominated by this solution.

All solutions in the first non-dominated front F_1 have zero domination count. For each solution with zero domination count, we visit each member of its domination set and reduce its domination count by one. If for any member the domination count becomes zero, we put it in a separate list, indicating that it belongs to the next non-dominated front. After that, the above procedure is continued with each member of the separate list, and the third front is identified. The process continues until all fronts are identified. Every solution is assigned a non-domination level and is never visited again.

Secondly, we have to maintain good spread in the obtained set of solutions, in order to promote *diversity* among the population members. This is achieved by sorting the population according to each objective function value in ascending order. For each objective function, the solutions with the smallest and largest function values are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference of the function values of two adjacent solutions, as follows:

$$I_{[i]distance} = \frac{I_{[i+1]m} - I_{[i-1]m}}{f_m^{max} - f_m^{min}}\,, \tag{2.25}$$

where $I_{[i]m}$ refers to the m-th objective function value of the i-th solution in the current non-dominated set I, and f_m^{max}, f_m^{min}, are the maximum and minimum values of the m-th objective function, respectively. The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective. Thus, each solution is assigned a distance.

So far, every individual in the population has a non-domination rank and a distance. During the selection process of the algorithm, when selecting between two individuals, we prefer the one with the lowest rank. If both solutions belong to the same front and have the same rank, we prefer the one with the highest value of distance, because it is located in a less crowded region of the search space.

Having already initialized and sorted the population, the main loop of the method begins. At first, we perform the *binary tournament selection*. During each generation, a portion of the existing population is selected to breed a new generation. We create pairs of solutions and we select one of them based on rank and distance.

After that, we apply the recombination and mutation operators. More specifically, we first apply *real-valued recombination* to the selected parents. Two selected parents, ${x_{i_1}}^{(t)}$ and ${x_{i_2}}^{(t)}$, are recombined creating two children. The recombinated j-th factor of the offspring vector is:

$$o_j^{(t)} = r_j \, {x_{i_1,j}}^{(t)} + (1 - r_j) \, {x_{i_2,j}}^{(t)}, \tag{2.26}$$

where r_j is normally distributed in the interval $[-z, 1+z]$, and z is a positive number. Next, *real-valued mutation* is performed on the selected individuals. This operator perturbs the values of genes using a normally distributed perturbation in $[0, s^2]$, where s is equal to the 10% of the given bounds of the selected gene [8].

An offspring population is eventually generated and combined with the parent population. The *combined population* is sorted according to the previous procedure. The solutions belonging to the best non-dominated set F_1 are the best solutions in the combined population and must be emphasized more than any other solution. In order to choose exactly the desired number of population members, we choose the solutions with the smaller ranks. From the solutions of the last selected front, we take the solutions with the higher values of distance [9].

2.2.5 Multi-objective particle swarm optimization algorithm

Multi-objective particle swarm optimization (MOPSO) differs from other MO algorithms because it uses an external repository for non-dominated solutions. We assume that we have a randomly generated swarm of size N, a D-dimensional search space, and M objective functions. Similarly to standard PSO, the i-th particle at iteration t defines a current *position*:

$$x_i^{(t)} = (x_{i1}, x_{i2}, ..., x_{iD}), \tag{2.27}$$

that is a candidate solution of the problem at hand. The particle moves in the search space with an adaptable *velocity*:

$$v_i^{(t)} = (v_{i1}, v_{i2}, ..., v_{iD}), \tag{2.28}$$

while tracking the best position it achieved so far:

$$p_i^{(t)} = (p_{i1}, p_{i2}, ..., p_{iD}). \tag{2.29}$$

The positions of the particles that represent non-dominated vectors are stored separately in the *repository*. The positions of the non-dominated vectors define hypercubes in the objective space in order to maintain solution diversity. Having already initialized the population and the repository, the main loop of the method begins.

The velocity and the position of the i-th particle at the next iteration $(t+1)$ are calculated according to the following equations:

$$v_i^{(t+1)} = w\, v_i^{(t)} + c_1\, rand()\, (p_i^{(t)} - x_i^{(t)}) + c_2\, rand()\, (rep_h - x_i^{(t)}), \tag{2.30}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}, \tag{2.31}$$

where w is the inertia factor; c_1 and c_2 are the cognitive and the social learning rate, respectively; and $rand()$ is a random number generator in the range [0,1]. The parameters are set to their default values, $w = 0.729$, $c_1 = c_2 = 1.49$, as suggested for PSO. The vector rep_h is taken from the repository, while the index h is selected in the following way: each hypercube in objective space containing more than one particle is assigned a fitness value equal to the result of dividing a random number by the number of particles that it contains. Based on the implementation of MOPSO, as described in [10], this random number is the 10. The fitness value assignment is performed, such that less crowded hypercubes receive higher selection probabilities. Then, roulette-wheel selection is applied using these fitness values to select the hypercube from which we will take the corresponding particle. Once the hypercube has been selected, we select randomly a particle within it.

Then, a *mutation operator* is applied to the particles. The swarm is subdivided into three parts of equal size. Each part adopts a different mutation scheme: the first part has no mutation at all, the second part has uniform mutation (i.e., the range of each decision variable is kept constant over generations), and the third part assumes a non-uniform mutation (i.e., the range of each decision variable decreases over time). Using these operators, we aim at achieving both exploration (uniform mutation) and exploitation (non-uniform mutation) of the search space [11].

In order to keep the particles within the search space, in case of violation, the decision variables are stopped to their boundaries, while the velocity is multiplied by -1 so that it searches in the opposite direction in the next iteration.

The next step is to evaluate each of the particles in the swarm and update the contents of the repository as well as the hypercubes. This update consists of inserting

all the currently non-dominated solutions into the repository. Any dominated solution from the repository are eliminated in the process. Since the size of the repository is limited, whenever it gets full, we give priority to those particles located in less populated areas of the objective space.

Eventually, the best positions are updated. When the current position of the particle dominates its best position, it replaces it. Otherwise, if the current position is dominated by the existing one, the latter is kept. If neither of them is dominated, then we select one at random [10].

2.3 Radial basis function neural networks

The VCB technique can be applied on any type of neural networks. Our approach adopts the widely used radial basis function (RBF) neural networks, which constitute a special class of ANNs. An RBF is a feed-forward three-layered and fully-connected neural network, having a topology as shown in Fig.(2.1). The first layer is the input layer, i.e., a set of source nodes that connects the network to the environment. The second layer is the hidden layer, which includes K neurons, each one implementing a radial activation function. For this purpose, the Gaussian function is frequently used as the radial basis function:

$$f_k(x, u_k, \sigma_k) = exp(-\frac{\|(x - u_k)^2\|}{2\sigma_k^2}) \,, \tag{2.32}$$

where x is the input of the network; $||.||$ represents the Euclidean (l_2) norm and u_k and σ_k are the center and the width of the k-th neuron, respectively.

The weighted outputs of the hidden layer are transmitted to the third layer, also called the output layer. The output layer calculates the linear combination of hidden layer outputs and bias to obtain the final output of the network for the specific activation pattern of the input layer [12] :

$$y(x) = \sum_{k=1}^{K} w_k \, f_k(x, u_k, \sigma_k) + b, \tag{2.33}$$

where x is the activation pattern; K is the number of RBFs used; w_k are the weights of the network; b is the bias; and $f_k(.)$ is the activation function.

The aim of training an RBF network of K neurons is the determination of the

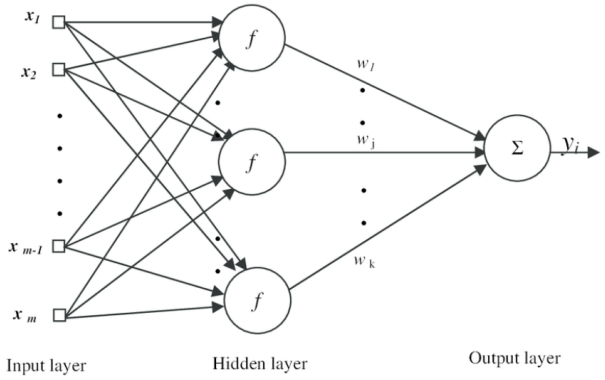


Figure 2.1: Architecture of an RBF network.

weights w_k between the hidden layer and the output layer, the width σ_k of the hidden layer base function, the center vectors u_{ik} of the hidden layer and the bias b, where $i = 1, ..., d$, with d being the dimension of the training vector, and $k = 1, ..., K$. All these parameters can be encoded in a decision vector as follows:

$$dv = [\, b\,,\, w_1\,,\, \sigma_1\,,\, u_{11}\,,\, u_{12}\,, ...,\, u_{1d}\,, ...,\, w_k\,,\, \sigma_K\,,\, u_{K1}\,,\, u_{K2}\,, ...,\, u_{Kd}\,]. \tag{2.34}$$

Thus, the dimension of the network's parameter space is equal to $D = 1 + K\,d$. The predictions of the network are compared to the expected output outcomes, calculating an error function, such as MSE [12].

A wide range of methods are used during the training of an RBF network, including clustering and optimization algorithms. Using clustering techniques, such as k-means, may result in dealing with two intrinsic disadvantages. The first is due to its iterative nature, which can lead to long convergence times, and the second originates from its inability to automatically determine the number of RBF centers, thus resulting in a time consuming trial-and-error procedure for establishing the size of the hidden layer [13]. A multitude of alternative techniques have been proposed to tackle these disadvantages, such as gradient-based optimization methods [3]. However, the necessity of gradient information of the function, results in high computational requirements. For this reason, evolutionary algorithms are, recently, used during the training of networks [7] [14]. They have lower computational requirements and at the same time they have strong global search ability, robustness and can solve difficult problems with functions that do not possess properties, such as continuity and differentiability.

Chapter 3

Proposed Approach

3.1 Variance counterbalancing with multi-objective solvers

In the proposed approaches, the VCB algorithm described in Section 2.1 is studied using the MO solvers described in Sections 2.2.3, 2.2.4 and 2.2.5. According to the VCB method, the training of the neural network proceeds in cycles. At each cycle, a set of mini-batches is selected at random, each mini-batch consisting of randomly selected training vectors. Then, the selected MO method is applied, and a set of non-dominated solutions is returned. We, having the role of the decision maker, may follow different strategies for the evaluation of these solutions. We can either evaluate all the non-dominated solutions, or evaluate a random number of them, or evaluate only one randomly selected solution, in order to compute the MSE of the current iteration using the whole training set.

At the end of each cycle a solution is available corresponding to a specific parameter setting of the neural network, as defined in the Eq.(2.34). At the same time, the solution with the lowest MSE (final solution) over the whole training set for all cycles is tracked and updated. In Fig.(3.1) we illustrate the objective values of the obtained

solutions for consecutive VCB cycles (blue dots), as well as, the final solution of the method (red dot) for a single experiment of each MO Method.

The stopping conditions for the VCB method are as follows:

- The method reaches a predefined maximum number of network evaluations. One network evaluation corresponds to the evaluation of the neural network on a single training pattern.
- The method reaches a predefined maximum number of VCB cycles.

The algorithm stops as soon as either of the condition is satisfied.

3.2 Application details for weighted aggregation approaches

The weighted aggregation approaches equipped with the PSO solver are implemented as described in Section 2.2.3. However, we made some modifications in order to gain better results. The first modification is the use of an external archive storing the non-dominated solutions visited by the algorithm. Any dominated solution from the archive are eliminated in the process. Since the size of the archive is limited, whenever it gets full, we select randomly such a number of individuals as the predefined archive size.

The second modification refers to the initialization of the population. In every call of the PSO solver, the swarm is initialized using the non-dominated solutions stored at the external archive of the previous VCB cycle. When the number of these solutions is smaller than the desired swarm size, we generate the remaining number of individuals randomly in the search space of the problem.

One of the well-known weaknesses of population-based algorithms when they are used to study functions with many local minima, is the tendency for premature convergence to local minimizers. Partial or full population restarting is one of the possible modifications addressing this issue. Our modification is the ability to restart the swarm when the distance of the particles is lower than a given bound, assuming that the individuals have stuck in a local minimum. In order to restart the population we use the non-dominated solutions that are stored in the archive. If the number of the archive's solutions is smaller than the desired size of the population, we generate

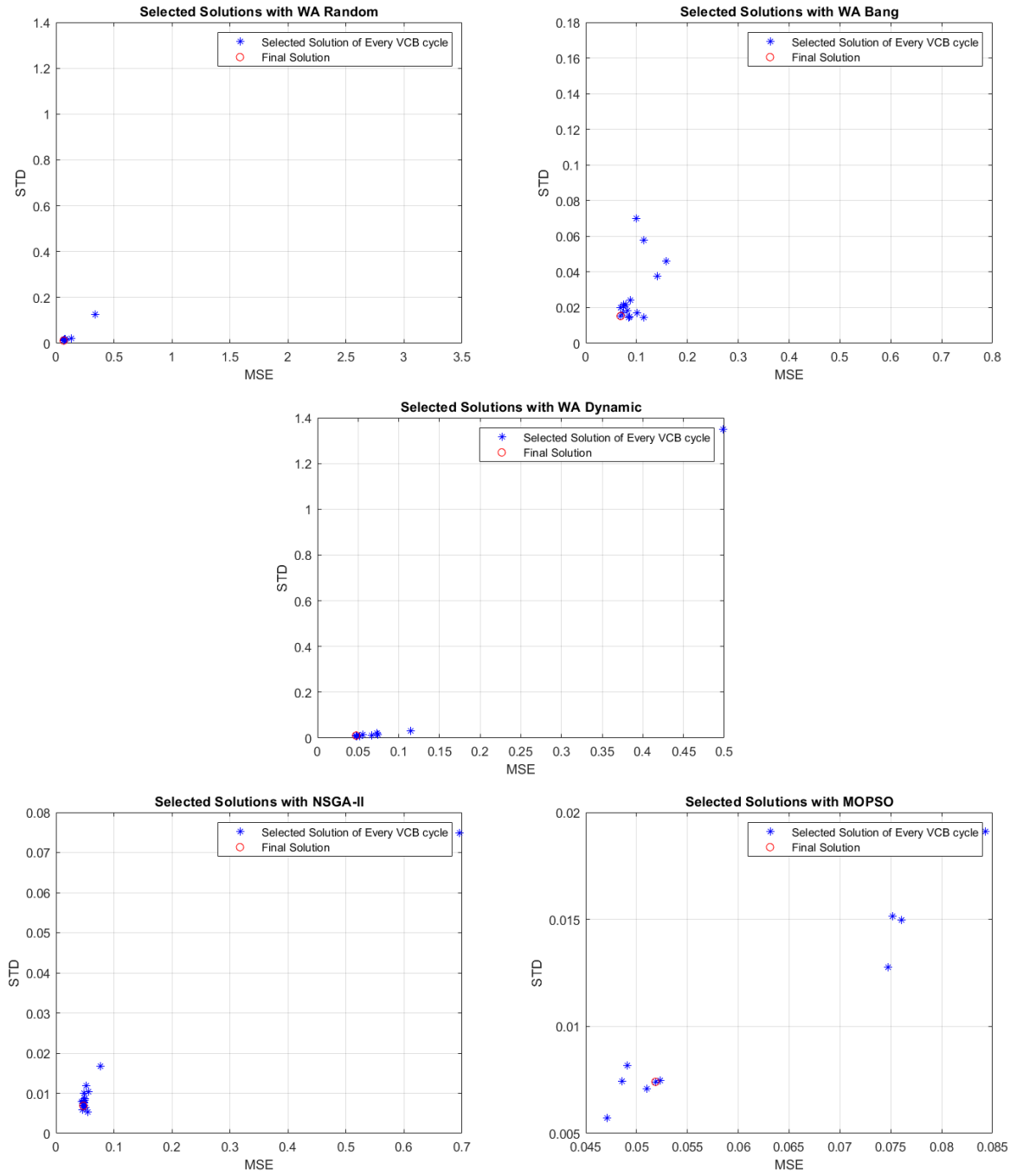


Figure 3.1: Objective values of the obtained solutions for consecutive VCB cycles (blue dots) and the final solution of the method (red dot) for a single experiment of each MO method.

the remaining number of individuals randomly in the search space of the problem we solve.

In order to keep the particles within the search space we do not only use the constant v_{max} to limit the velocities of particles in the nearest bound. We select to multiply the velocity by -1 so that they search in the opposite direction in the next iteration.

A solution (particle) is evaluated using one of the three available weighted aggregation methods, i.e., random, bang-bang, or dynamic approach. In each case, we select to transform every objective function f_i as follows,

$$f_i(x) = \frac{f_i(x) - f_i^{\min}(x)}{f_i^{\max}(x) - f_i^{\min}(x)} \tag{3.1}$$

where x is a solution; $i = 1, ..., M$, M the number of objective functions; $f_i^{\max}(x)$ and $f_i^{\min}(x)$ are the maximum and minimum value of the objective function until the time the method is called, respectively [15]. All objective functions after normalization will be bounded by the values 0 and 1. This last modification for the weighted aggregation approaches arised out of our desire to give the same magnitude to each objective function.

3.3 Application details for non-sorting genetic algorithm

The NSGA-II algorithm is implemented as described in Section 2.2.4. We introduced 3 modifications in order to gain results of higher quality. The first modification refers to the initialization of the population. In every call of the NSGA-II solver, the population is initialized using the Pareto solutions obtained of the previous VCB cycle. When the number of these solutions is smaller than the desired population size, we produce the remaining number of individuals randomly in the search space of the problem at hand.

One disadvantage of NSGA-II is the premature convergence to local optima of the objective functions. The problem is related to the loss of genetic diversity of the population. For this reason, we test the individual's genetic material, before the crossover operation. If the individuals have the same genes, then the recombination of their genetic material is ineffective, since the offsprings are simply clones of their parents. So, we use only one individual in the recombination procedure as the first

parent, and we introduce a randomly generated individual as the second parent [16].

An increasing mutation rate is the third modification in our proposed approach. This way, we have the opportunity to use an adaptive mutation rate if there is not an improvement of the lowest value of the MSE using the entire training dataset over a given bound of single-pattern evaluations [16].

3.4 Application details for multi-objective particle swarm optimization algorithm

The MOPSO algorithm is implemented as described in Section 2.2.5. The only modification that we made is related to the initialization of the swarm. Similarly to the rest of the methods described above, in every call of MOPSO the swarm is initialized using the non-dominated solutions from the repository of the previous VCB cycle. When the number of these solutions is smaller than the desired size of the swarm, we generate the remaining particles randomly in the search space of the problem.

Chapter 4

Experimental Analysis

The experimental analysis was based on the application of VCB on RBF networks. Taking into consideration the stochasticity of the VCB algorithm as well as of the selected MO methods, an appropriate analysis shall include a number of independent experiments, and a complete statistical analysis of the obtained solutions.

The first test problem was the two-dimensional "Mexican Hat" function, defined as:

$$f(x_1, x_2) = \frac{\sin\left(x_1^2 + x_2^2\right)}{\sqrt{x_1^2 + x_2^2}}, \tag{4.1}$$

which is depicted in Fig.(4.1). For this problem, 40000 two-dimensional pattern vectors were taken as our dataset:

$$(x_1, x_2) \in [-5, 5] \times [-5, 5]. \tag{4.2}$$

In order to solve the VCB minimization problem, we used the MO methods presented in Chapters 2 and 3. The basic VCB-related, algorithm-related, and network-related

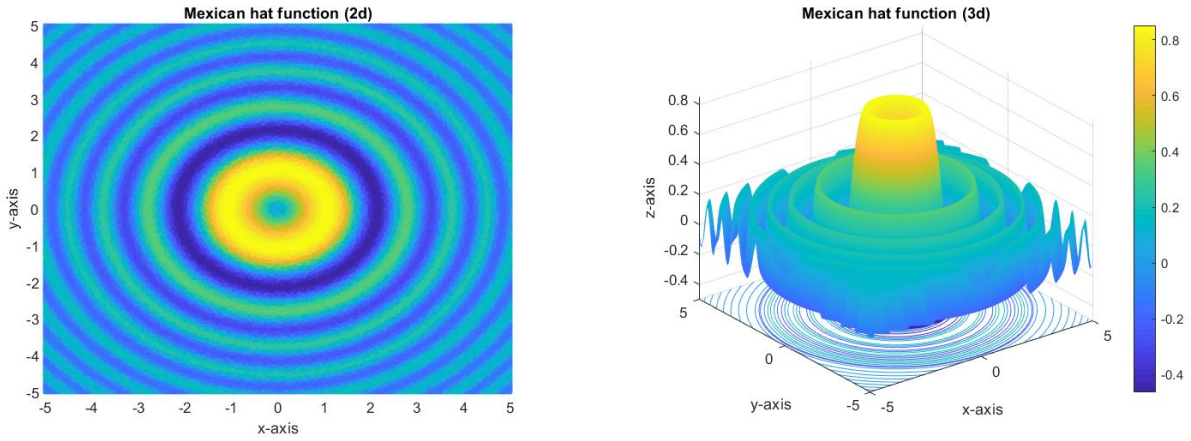


Figure 4.1: The "Mexican Hat" function.

parameters of our experimental setup are reported in Tables 4.1, 4.2 and 4.3, respectively. Besides the fixed parameters, we considered a number of parameters with variable values, aiming at studying the performance of the MO algorithms under their different settings. These parameters are:

- pd; percentage of the dataset that constitutes the training set.
- n; number of neurons for the network.
- vcb; number of cycles of the VCB method.
- str; strategy of evaluating the non-dominated solutions.
- ps; population size.
- mut; mutation strategy.
- vel; maximum velocity.
- $rest$; restart mechanism.

The values of these parameters are defined for each MO approach separately.

For each parameter setting 25 experiments were performed. The software was developed and tested on MATLAB R2018a running on a Linux system. For the statistical analysis, the one-sample Kolmogorov-Smirnov normalization test was used to check if the data of each sample came from the normal distribution [17]. The results showed that, in all cases, the data did not come from the normal distribution.

Table 4.1: Fixed VCB-related parameters.

Parameters	Value
number of patterns	40000
number of mini-batches	20
patterns per mini-batch	50
training patterns dimension	2
maximum pattern evaluations	10000000

Table 4.2: Fixed algorithm-related parameters.

Parameters	PSO	NSGA-II	MOPSO
w	0.729	-	0.729
c_1	1.49	-	1.49
c_2	1.49	-	1.49
number of grids	-	-	20
mutation rate	-	-	0.5
tournament size	-	2	-
crossover probability	-	0.5	-

Since each sample was acquired independently and its size was big enough, we used the non-parametric Wilcoxon ranksum test to compare samples on the assumption of having equal medians [17]. In case that the samples did not come from distributions of equal medians, the one with the lower median was retained. In all comparisons, a level of significance 0.05 was considered.

The results reported in all the provided tables are rounded to 4 decimal digits. Also, the lowest value of interest is boldfaced whenever required.

Our experimental analysis comprised of the following four phases:

(a) *Phase 1*: Each MO algorithm was applied on the considered regression problem for all combinations of the variable parameters. The experiments were divided in different settings. For each setting, one of the parameters was of interest. Thus, the performances of the algorithm under the different values of this parameter were compared among them, for each combination of the rest of the parameters, separately. This phase aimed at identifying the most influential parameter(s) for each MO algorithm.

Table 4.3: Fixed RBF network-related parameters.

Parameters	Value
min value of center components	-5
max value of center components	5
min value of weights	-20
max value of weights	20
min value of width	0.1
max value of width	2.0

(b) *Phase 2*: This phase consisted of statistical comparisons of different MO methods among them.

(c) *Phase 3*: This phase included comparisons between MO methods and BFGS of the standard VCB approach.

(d) *Phase 4*: This phase consisted of comparisons of the most representative methods on different datasets.

4.1 Experimental phase 1: individual performance under different settings

We conducted a number of experiments to identify promising parameter settings for each MO method. Due to the huge number of experiments in phase 1, we provide tables with the relevant settings for each algorithm, as well as boxplots of the attained MSE values, in the Appendix, while summarizing the results in the following paragraphs.

4.1.1 Weighted aggregation approaches

The three weighted aggregation methods, i.e., random wa, bang-bang wa, and dynamic wa were all applied, individually, under each one of the settings reported in Table 4.4. For each setting, the corresponding parameter of interest was considered and its effect on the algorithm performance was analyzed for all combinations of the rest of the parameters. The considered parameter settings are reported in Tables

Table 4.4: Parameter settings for the weighted aggregation approaches.

Setting	Examined parameter	Values
1	pd	{ 0.6 , 0.8 }
2	n	{ 5 , 10 }
3	vcb	{ 10 , 20 }
4	str	{ 1 , 2 , 3 }
5	ps	{ 20 , 40 }
6	vel	{ 1 , 2 }
7	rest	{ 0 , 1 }

A.1-A.7 and the obtained solutions are illustrated in boxplots in Figs. A.1-A.21 in the Appendix.

Random weighted aggregation

Regarding setting 1 (different pd value), there were 10 cases in total where the samples of random wa with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon test of different medians. Moreover, half of these cases performed better using a 60% of the dataset as the training set. Thus, there seemed to be no special preference on the training dataset size.

About setting 2 (number of neurons), all cases with 5 neurons outperformed those with 10 neurons. As a result, using 5 neurons is clearly beneficial for this approach.

Concerning setting 3 (number of VCB cycles) we observed that there were 105 cases, out of 192, where the samples parsed the Wilcoxon test for different medians. Since in all these cases 10 VCB cycles were preferred to run, using 10 VCB cycles, instead of 20, is helpful for the approach.

Relating to setting 4 (evaluation strategy) we noticed that when we examined the relation of the medians between the first and the second strategy there were 12 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the first strategy was preferred, we recommend using the first strategy to evaluate the non-dominated solutions, instead of the second. Next, we examined the relation of the medians between the first and the third strategy and we observed that there were 7 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the first strategy was preferred, we recommend using the

first strategy to evaluate the Pareto solutions, instead of the third. When we examined the relation of the medians between the second and the third strategy there were 12 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the third. Eventually, we advocate evaluating all the non-dominated solutions, using the first strategy.

About setting 5 (different swarm size) we noticed that there were 40 cases where the samples parsed the Wilcoxon test for different medians. Since in all these cases a swarm with 20 particles was preferred to run, we recommend using 20 individuals for our population, instead of 40.

Regarding setting 6 (percentage of the velocity) we saw that there were 16 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases a maximum velocity equal to 5% of the search space per dimension was preferred, we advocate using it, instead of a maximum velocity equal to 20% of the search space per dimension.

Concerning setting 7 (existence of a restart mechanism) we observed that there were 14 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases a restart mechanism was performed, we recommend restarting the population, if it is essential, in the implementation of the random wa approach.

Almost all of the parameters of interest were selected for the random wa method, based on exclusively the results of the experiments. Only the proportion of the dataset that constitutes the training set was not strictly specified. We propose a 60% of the dataset to constitute the training set, since the corresponding sample had lower MSE standard deviation in both training and testing sets, compared to the sample that used a 80% of the dataset as the training set.

Bang-bang weighted aggregation

Regarding setting 1 (different pd value), there were 7 cases in total where the samples of bang-bang wa with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon test of different medians. Since in the majority of these cases a 80% of the dataset was preferred to constitute the training set of the approach, we recommend using 0.8 as the selected pd value, instead of 0.6.

About setting 2 (number of neurons), all cases with 5 neurons outperformed these with 10 neurons. As a result, using 5 neurons is clearly beneficial for this approach.

Concerning setting 3 (number of VCB cycles) we observed that there were 36 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases 20 VCB cycles were preferred to run, using 20 VCB cycles, instead of 10, seems helpful.

Relating to setting 4 (evaluation strategy) we noticed that when we examined the relation of the medians between the first and the second strategy there were 10 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the first. Next, we examined the relation of the medians between the first and the third strategy and we observed that there were 5 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the third strategy was preferred, we recommend using the third strategy to evaluate the Pareto solutions, instead of the first. When we examined the relation of the medians between the second and the third strategy there were 7 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the third. Eventually, we advocate evaluating a random number of non-dominated solutions, using the second evaluation strategy.

About setting 5 (different swarm size) we noticed that there were 42 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases a swarm with 40 particles was preferred to run, we recommend using 40 individuals for our population, instead of 20.

Regarding setting 6 (percentage of the velocity) we saw that there were 8 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases a maximum velocity equal to 5% of the search space per dimension was preferred, we advocate using it, instead of a maximum velocity equal to 20% of the search space per dimension.

Concerning setting 7 (existence of a restart mechanism) we observed that there were 6 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases a restart mechanism was applied, we recommend restarting the population, if it is essential, in the implementation of the bang-bang

wa approach.

Dynamic weighted aggregation

Regarding setting 1 (different pd value), there were 5 cases in total where the samples of dynamic wa with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon test of different medians. Moreover, most of these cases performed better using a 60% of the dataset as the training set. Thus, the training set size is proposed to be equal to 60% of the dataset.

About setting 2 (number of neurons), all cases with 5 neurons surpassed those with 10 neurons. As a result, using 5 neurons is clearly beneficial for this approach.

Concerning setting 3 (number of VCB cycles) we observed that there were 84 cases, out of 192, where the samples parsed the Wilcoxon test for different medians. Since in all these cases 10 VCB cycles were preferred to run, using 10 VCB cycles, instead of 20, seems helpful for the method.

Relating to setting 4 (evaluation strategy) we noticed that when we examined the relation of the medians between the first and the second strategy there were 3 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the first. Next, we examined the relation of the medians between the first and the third strategy and we observed that there were 5 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the third strategy was preferred, we recommend using the third strategy to evaluate the Pareto solutions, instead of the first. When we examined the relation of the medians between the second and the third strategy there were 10 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the third. Eventually, we advocate evaluating a random number of the non-dominated solutions, using the second strategy.

About setting 5 (different swarm size) we noticed that there were 72 cases where the samples parsed the Wilcoxon test for different medians. Since in all these cases a swarm with 20 particles was preferred to run, we recommend using 20 individuals for our population, instead of 40.

Regarding setting 6 (percentage of the velocity) we saw that there were 6 cases

Table 4.5: Parameter settings for NSGA-II.

Setting	Examined parameter	Values
1	pd	{ 0.6 , 0.8 }
2	n	{ 5 , 10 }
3	vcb	{ 10 , 20 }
4	str	{ 1 , 2 , 3 }
5	ps	{ 20 , 40 }
6	mut	{ 1 , 2 }

where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases a maximum velocity equal to 20% of the search space per dimension was preferred, we advocate using it, instead of a maximum velocity equal to 5% of the search space per dimension.

Concerning setting 7 (existence of a restart mechanism) we observed that there were 4 cases where the samples parsed the Wilcoxon test for different medians. Half of these cases performed better using a restart mechanism. Thus, there seemed to be no special preference on the existence or not of a population restart. However, having already specified almost all of the parameters of interest for the dynamic wa method, based on exclusively the results of the experiments, we propose offering the ability of applying a restart mechanism, since the corresponding sample had lower MSE standard deviation and mean in both training and testing sets, compared to the sample that could not apply restart.

4.1.2 Non-sorting genetic algorithm

The NSGA-II algorithm was applied, under each one of the settings reported in Table 4.5. Probing the effect of the examined parameter under consideration of the rest. The settings are reported in Tables A.8-A.13 and the obtained solutions are illustrated in boxplots in Figs. A.22-A.27 in the Appendix.

Regarding setting 1 (different pd value), there were 2 cases, out of 96, where the samples of NSGA-II with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon test of different medians. Moreover, half of these cases performed better using a 60% of the dataset as the training set. Thus, there seemed to be no special preference on the training dataset size.

About setting 2 (number of neurons), all cases with 5 neurons outperformed those with 10 neurons. As a result, using 5 neurons is clearly helpful for the method.

Concerning setting 3 (number of VCB cycles) we observed that there were 4 cases where the samples parsed the Wilcoxon test for different medians. Since in all these cases 10 VCB cycles were preferred to run, using 10 VCB cycles, instead of 20, is beneficial for this approach.

Relating to setting 4 (evaluation strategy) we noticed that when we examined the relation of the medians between the first and the second strategy there were 2 cases where the samples parsed the Wilcoxon test for different medians. Since in these cases, the second strategy was preferred, we recommend using the second strategy to evaluate the non-dominated solutions, instead of the first. Next, we examined the relation of the medians between the first and the third strategy and we observed that there were 7 cases where the samples parsed the Wilcoxon test for different medians. Since in these cases, the third strategy was preferred, we recommend using the third strategy to evaluate the Pareto solutions, instead of the first. When we examined the relation of the medians between the second and the third strategy there was only 1 case where the samples parsed the Wilcoxon test for different medians. In that case the third strategy was preferred. So, we recommend using the third strategy to evaluate the non-dominated solutions, instead of the second. Eventually, we advocate evaluating only one randomly selected non-dominated solution, using the third strategy.

About setting 5 (different population size) we noticed that there were 27 cases where the samples parsed the Wilcoxon test for different medians. Since in all these cases a population with 40 individuals was preferred to run, we recommend using 40 individuals, instead of 20.

Regarding setting 6 (mutation strategy) we saw that there were 14 cases where the samples parsed the Wilcoxon test for different medians. Since in all these cases an adaptive mutation rate was preferred, we advocate using it, instead of using a fixed one.

Almost all of the parameters of interest were selected for NSGA-II, based on exclusively the results of the experiments. Only the proportion of the dataset that constitutes the training set is not strictly specified. We propose a 60% of the dataset as the training set, since the corresponding sample had lower MSE standard deviation and mean in both training and testing sets, compared to the sample that used a 80%

Table 4.6: Parameter settings for MOPSO.

Setting	Examined parameter	Values
1	pd	{ 0.6 , 0.8 }
2	n	{ 5 , 10 }
3	vcb	{ 10 , 20 }
4	str	{ 1 , 2 , 3 }
5	ps	{ 20 , 40 }
6	vel	{ 1 , 2 }

of the dataset to constitute the training set.

4.1.3 Multi-objective particle swarm optimization algorithm

The MOPSO algorithm was applied, under each one of the settings reported in Table 4.6. For each setting, the corresponding parameter of interest was considered and its effect on the algorithm performance was analyzed for all combinations of the rest of the parameters. The considered parameter settings are reported in Tables A.14-A.19 and the obtained solutions are illustrated in boxplots in Figs. A.28-A.33 in the Appendix of the thesis.

Regarding setting 1 (different pd value), there were 5 cases in total where the samples of MOPSO with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon test of different medians. Since in the majority of these cases a 60% of the dataset is preferred to constitute the training set, we recommend using a 60%, instead of 80%, of the dataset to train our network.

About setting 2 (number of neurons) we noticed that in 94, out of 96, cases the samples of MOPSO with 5 neurons outstriped those with 10 neurons. As a result, using 5 neurons is clearly beneficial for this approach.

Concerning setting 3 (number of VCB cycles) we observed that in 53 cases the samples that used 10 VCB cycles exceeded those with 20 cycles. Thus, using 10 cycles is useful for the MOPSO method.

Relating to setting 4 (evaluation strategy) we noticed that when we examined the relation of the medians between the first and the second strategy there were 3 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the second strategy was preferred, we recommend using the

second strategy to evaluate the non-dominated solutions, instead of the first. Next, we examined the relation of the medians between the first and the third strategy and we observed that there were 7 cases where the samples parsed the Wilcoxon test for different medians. Since in most of these cases, the third strategy was preferred, we recommend using the third strategy to evaluate the Pareto solutions, instead of the first. When we examined the relation of the medians between the second and the third strategy there were 9 cases where the samples parsed the Wilcoxon test for different medians. Since in the majority of these cases, the third strategy was preferred, we recommend using the third strategy to evaluate the non-dominated solutions, instead of the second. Eventually, we advocate evaluating only one randomly selected non-dominated solution, using the third evaluation strategy.

About setting 5 (different swarm size) we noticed that in 64 cases the samples parsed the Wilcoxon test for different medians, using 20 particles for the swarm. So, we recommend using 20 individuals for our population, instead of 40.

Regarding setting 6 (percentage of the velocity) we saw that in 42 cases the samples parsed the Wilcoxon test for different medians. Since in most of these cases having a maximum velocity equal to 20% of the search space per dimension was preferred, we advocate using a maximum velocity equal to 20%, instead of 5%, of the search space per dimension.

4.2 Experimental phase 2: comparisons among multi-objective approaches

The most promising variants (set of parameters) of the MO methods were compared among them in experimental phase 2. The basic statistical values of MSE in the training and testing sets are illustrated in Tables 4.7 and 4.8 for the compared MO methods. Figure 4.2 illustrates the training and testing MSE of the MO methods together, indicating the lack of significant differences in the values of MSE between the two sets for each method.

From the pairwise comparisons between the weighted aggregation methods reported in Table 4.9, we can see that the performances of random wa and dynamic wa were superior to the bang-bang wa. At the same time, these methods had lower MSE mean and standard deviation in both training and testing sets, compared to the

Table 4.7: Training MSE of the methods.

Method	min	mean	median	max	std
Random WA	0.0490	0.0598	0.0547	**0.0782**	0.0095
Bang WA	0.0585	0.0973	0.0783	0.2889	0.0509
Dynamic WA	0.0470	0.0603	0.0546	0.1093	0.0139
NSGA-II	**0.0450**	**0.0513**	**0.0486**	0.0819	**0.0080**
MOPSO	0.0472	**0.0513**	0.0489	0.0885	0.0088
BFGS	0.0496	0.1574	0.0705	1.8721	0.3631

Table 4.8: Testing MSE of the methods.

Method	min	mean	median	max	std
Random WA	0.0495	0.0617	0.0563	0.0873	0.0109
Bang WA	0.0594	0.0995	0.0808	0.2782	0.0553
Dynamic WA	0.0467	0.0602	0.0533	0.1160	0.0153
NSGA-II	**0.0451**	**0.0518**	**0.0492**	**0.0833**	**0.0082**
MOPSO	0.0479	0.0529	0.0497	0.0957	0.0112
BFGS	0.0484	0.1548	0.0684	1.9073	0.3711

bang-bang wa method, indicating that they can produce more robust solutions for the minimization problem we study.

After, we performed pairwise comparisons between the best-performing MO methods, that are reported in Table 4.10. We observe that the performances of Pareto-based evolutionary algorithms were superior to the wa methods. At the same time, the NSGA-II algorithm had the lowest MSE mean and standard deviation in both training and testing sets among the MO methods.

4.3 Experimental phase 3: comparisons between multi-objective methods and BFGS

The proposed MO methods were compared also against the BFGS method that is used with the standard VCB algorithm. Similarly to the MO methods, a tuning procedure based on different settings was also applied for the BFGS method, and it is

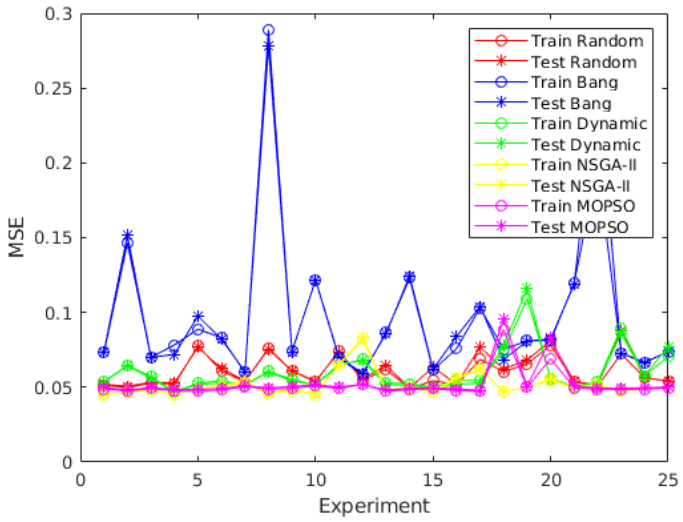


Figure 4.2: Training and Testing MSE of MO methods.

Table 4.9: Wilcoxon ranksum tests of the WA methods. The corresponding p-values are given in the parentheses.

Compared algorithms A -vs- B		Training set	Testing set
Random WA	Bang WA	+ (0.0000)	+ (0.0000)
Random WA	Dynamic WA	= (0.7710)	= (0.1870)
Bang WA	Dynamic WA	− (0.0000)	− (0.0000)
["+": A is better than B]	["−": B is better than A]		["=": indifferent]

summarized below.

4.3.1 Parameter tuning of the BFGS method

We conducted a number of experiments to identify promising parameters similarly to the MO methods. We examined 3 settings, as reported in Table 4.11. For each setting, the corresponding parameter of interest was considered and its effect on the algorithm performance was analyzed for all combinations of the rest of the parameters. The considered parameter settings are reported in Tables A.20-A.22 and the obtained solutions are illustrated in boxplots in Figs. A.34-A.36 in the Appendix of the thesis.

Regarding setting 1 (different pd value), there were 2 cases in total where the samples of BFGS with $pd = 0.6$ and $pd = 0.8$, respectively, did pass the Wilcoxon

Table 4.10: Wilcoxon ranksum tests of the MO methods. The corresponding p-values are given in the parentheses.

Compared algorithms A -vs- B		Training set	Testing set
Random WA	NSGA-II	– (0.0000)	– (0.0000)
Random WA	MOPSO	– (0.0000)	– (0.0000)
Dynamic WA	NSGA-II	– (0.0001)	– (0.0001)
Dynamic WA	MOPSO	– (0.0000)	– (0.0000)
NSGA-II	MOPSO	= (0.6415)	= (0.3320)
["+": A is better than B]	["−": B is better than A]		["=": indifferent]

Table 4.11: Parameter settings for BFGS.

Setting	Examined parameter	Values
1	pd	{ 0.6 , 0.8 }
2	n	{ 5 , 10 }
3	vcb	{ 10 , 20 }

test of different medians. Moreover, these cases performed better using a 80% of the dataset as the training set. Thus, using 80% of the dataset to constitute the training set is helpful for this approach.

About setting 2 (number of neurons), we observed that there were 2 cases where the samples parsed the Wilcoxon test for different medians. Since in these cases 10 neurons, instead of 5, were preferred for the network, we recommend using 10 neurons.

Concerning setting 3 (number of VCB cycles) we observed that there was only 1 case where the samples parsed the Wilcoxon test for different medians. Since in that case 10 VCB cycles were preferred to run, we propose applying 10 VCB cycles, instead of 20.

4.3.2 Comparisons between the methods

The best-performing approaches of both weighted aggregation and Pareto-based MO algorithms were compared with the standard VCB method with the BFGS solver. In Tables 4.7 and 4.8 the basic statistical values of MSE in the training and testing sets

Table 4.12: Wilcoxon ranksum tests of the optimization methods. The corresponding p-values are given in the parentheses.

Algorithm	Case 1		Case 2		Case 3	
A	Training set	Testing set	Training set	Testing set	Training set	Testing set
Random WA	+ (0.0004)	+ (0.0076)	+ (0.0001)	+ (0.0026)	− (0.0001)	− (0.0000)
Dynamic WA	+ (0.0001)	+ (0.0009)	+ (0.0026)	+ (0.0002)	− (0.0031)	+ (0.0008)
NSGA-II	+ (0.0000)	+ (0.0000)	+ (0.0000)	+ (0.0000)	= (0.1253)	= (0.3320)
MOPSO	+ (0.0000)	+ (0.0000)	+ (0.0000)	+ (0.0000)	+ (0.0036)	+ (0.0181)
["+": A is better than BFGS] ["−": BFGS is better than A] ["=": indifferent]						

are reported, using the proposed set of parameters for each method. According to the statistical analysis of the optimization methods, we distinguished 3 cases. In Table 4.12 all the p-values of the pairwise comparisons are reported and explained below.

In the first case, we compared the methods using their proposed set of parameters. From the pairwise comparisons, we can see that the performances of the MO methods were superior to the BFGS method. At the same time, the MO methods exhibited lower MSE mean and standard deviation in both training and testing sets compared to the BFGS method. Interestingly, NSGA-II and MOPSO had 45.26 and 33.13 times smaller MSE standard deviation in testing set compared to the BFGS method, respectively.

According to the parameter tuning, we proposed to use 5 neurons when we employed the MO methods, while for the BFGS method we proposed 10 neurons for the neural network. For this reason, in the second case, we compared the optimization methods when all of them used 5 neurons. From the pairwise comparisons, we observe that the performances of the MO methods were again superior to the BFGS method.

In the third case, we compared the methods when all of them used 10 neurons. The pairwise comparisons between them produced mixed results, but we can notice that the performance of NSGA-II was competitive to BFGS, while the performance of MOPSO was superior to the BFGS method.

4.3.3 Running-time requirements

In the previous sections, we focused on comparisons of the optimization methods with respect to their MSE values. For this section we compared the best-performing MO methods between them, as well as against the BFGS method, based on the required running-time (in seconds) in training phase of each experiment. We ignored the time

Table 4.13: Statistical values of training/running time (seconds) for all methods.

Values	min	mean	median	max	std
Random WA	19.6000	20.7480	21.0000	22.1000	0.7714
Dynamic WA	32.4000	34.4360	34.8000	35.8000	1.1467
NSGA-II	18.1000	18.3560	18.4000	18.6000	0.1261
MOPSO	16.5000	**16.6240**	**16.6000**	**16.8000**	**0.0663**
BFGS (n=5)	**10.8000**	32.0120	34.9000	46.0000	9.2848
BFGS (n=10)	30.9000	58.3240	58.0000	78.6000	13.5579

of the testing phase that was negligible in each case. The basic statistical values of running-time for the training set are illustrated in Table 4.13, using the proposed set of parameters for each method.

From the pairwise comparisons between the MO methods we got that the median of the random wa sample was greater than the median of NSGA-II (p-value $= 0.0000$). The median of the random wa sample was greater than the median of MOPSO (p-value $= 0.0000$). The median of the dynamic wa sample was greater than the median of NSGA-II (p-value $= 0.0000$). The median of the dynamic wa sample was greater than the median of MOPSO (p-value $= 0.0000$). The median of NSGA-II sample was greater than the median of MOPSO (p-value $= 0.0000$). Eventually, the performance of MOPSO was superior to the other MO methods. Also, MOPSO had the lowest mean and standard deviation of running-time.

From the pairwise comparisons between the MO methods and the BFGS method, we got that the performance of each MO method was superior to BFGS (p-value $= 0.0000$). We received the same results when we compared the time complexity of the methods when all of them used 5 and 10 neurons.

4.4 Experimental phase 4: comparisons on different datasets

4.4.1 A noisy dataset

This part of our analysis contained comparisons on different datasets, starting with our regression model for the "Mexican Hat" function, using a noisy dataset, where 25% of the patterns in training were contained with noise. The noise follows the

Table 4.14: Training MSE using noisy dataset.

Values	min	mean	median	max	std
Random WA	0.0909	0.1261	0.1186	0.2775	0.0491
Bang WA	0.0983	0.1234	0.1225	0.1831	0.0195
Dynamic WA	0.0830	0.1160	0.1048	0.3409	0.0501
NSGA-II	0.0833	**0.0894**	0.0894	**0.1098**	**0.0051**
MOPSO	0.0817	0.0936	0.0907	0.1687	0.0159
BFGS	**0.0494**	0.4396	**0.0680**	9.1855	1.8226

Table 4.15: Testing MSE using noisy dataset.

Values	min	mean	median	max	std
Random WA	0.0529	0.0834	0.0720	0.2903	0.0556
Bang WA	0.0562	0.0793	0.0734	0.1496	0.0213
Dynamic WA	0.0471	0.0758	0.0609	0.2973	0.0490
NSGA-II	**0.0450**	**0.0519**	0.0521	**0.0728**	**0.0053**
MOPSO	0.0471	0.0545	**0.0512**	0.1189	0.0138
BFGS	0.0494	0.4591	0.0694	9.6090	1.9067

Gaussian distribution with mean value equal to the true function value and sigma (standard deviation) equal to 0.01. The network was trained using the noisy function values, while tested using the true function values. The basic statistical values of MSE in both training and testing sets are reported in Tables 4.14 and 4.15.

The first step of the statistical analysis was the application of normality tests in both training and testing samples using the one-sample Kolmogorov-Smirnov test. The results showed that none of the samples came from the normal distribution at a 0.05 significance level. So, we compared the methods, using the non-parametric Wilcoxon ranksum test.

From the pairwise comparisons between the methods reported in Table 4.16, we can see that the performances of NSGA-II and MOPSO were superior to the other methods. At the same time, NSGA-II had the lowest MSE mean and standard deviation in both training and testing sets.

Subsequently, we statistically compared NSGA-II and MOPSO with the BFGS method, distinguishing 3 cases as for the noiseless case (original dataset). In Ta-

Table 4.16: Wilcoxon ranksum tests of the MO methods using noisy dataset. The corresponding p-values are given in the parentheses.

Compared algorithms A -vs- B		Training set	Testing set
Random WA	Bang WA	$=$ (0.2523)	$=$ (0.1352)
Random WA	Dynamic WA	$=$ (0.1403)	$=$ (0.3224)
Random WA	NSGA-II	$-$ (0.0000)	$-$ (0.0000)
Random WA	MOPSO	$-$ (0.0000)	$-$ (0.0000)
Bang WA	Dynamic WA	$-$ (0.0047)	$-$ (0.0181)
Bang WA	NSGA-II	$-$ (0.0000)	$-$ (0.0000)
Bang WA	MOPSO	$-$ (0.0000)	$-$ (0.0000)
Dynamic WA	NSGA-II	$-$ (0.0000)	$-$ (0.0001)
Dynamic WA	MOPSO	$-$ (0.0009)	$-$ (0.0004)
NSGA-II	MOPSO	$+$ (0.0181)	$=$ (0.5605)
[“+”: A is better than B]		[“$-$”: B is better than A]	[“=”: indifferent]

ble 4.17 all the p-values of the pairwise comparisons are reported. In the first case we compared the methods using the proposed set of parameters of each method. From the pairwise comparisons we can see that the performances of the MO methods were superior to the BFGS method in the testing phase, while they were inferior in the training phase. Interestingly, NSGA-II and MOPSO had a 359.75 and 138.17 times smaller MSE standard deviation compared to the BFGS method in testing phase, respectively.

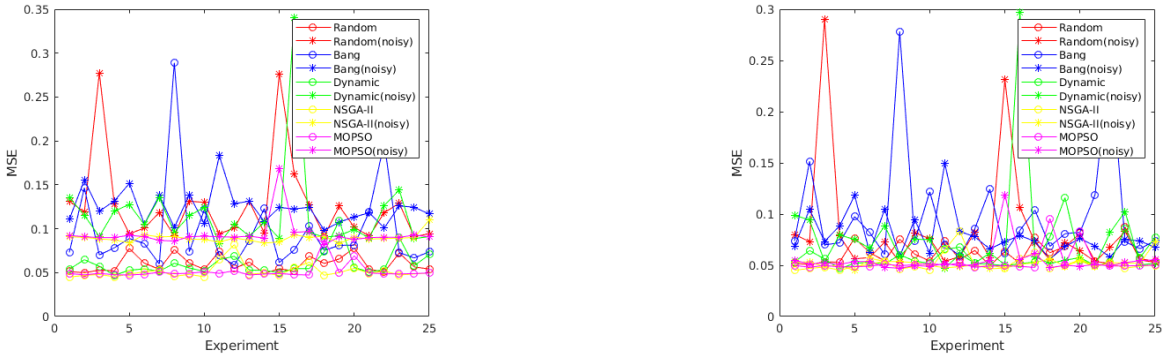
In the second case we compared the optimization methods when all of them used 5 neurons. Just like in the first case, we can see that the performances of the MO methods were superior to the BFGS method in testing, while they were inferior in training.

In the third case we compared the methods when all of them used 10 neurons. The pairwise comparisons between them produced mixed results, but the performance of NSGA-II was competitive to the BFGS method in testing and inferior in training, while the performance of MOPSO was superior to the BFGS method in testing and inferior in training.

In Fig.(4.3), the training and testing MSE for both original and noisy datasets

Table 4.17: Wilcoxon ranksum tests of the optimization methods using noisy dataset. The corresponding p-values are given in the parentheses.

Algorithm	Case 1		Case 2		Case 3	
A	Training set	Testing set	Training set	Testing set	Training set	Testing set
NSGA-II	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$=$ (0.2604)
MOPSO	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$+$ (0.0069)
[“+”: A is better than BFGS] [“−”: BFGS is better than A] [“=”: indifferent]						



(a) Training MSE

(b) Testing MSE

Figure 4.3: Training and Testing MSE for original and noisy datasets.

using the MO methods are illustrated. Afterwards, we compared statistically each MO method for these datasets. From the pairwise comparisons between the methods reported in Table 4.18, we can see that the performances of the MO methods in training were better when they used the original dataset, while in testing phase the results were mixed, since random wa and MOPSO were better when they used the original dataset, while the remaining MO methods could perform equally efficient in both datasets.

4.4.2 The red wine dataset

The last dataset that we used in our analysis was the Red Wine Quality dataset. The dataset contains a total of 12 variables, which are recorded for 1599 observations. We examined the quality of wine based on acidity. So, we used only 2 of the variables, the fixed acidity, which are non-volatile acids that do not evaporate readily, and the volatile acidity, which are high acetic acid in wine which leads to an unpleasant vinegar taste [18]. We chose that dataset because industry players are of interest in relating the human quality of tasting to wine's chemical properties so that certification

Table 4.18: Wilcoxon ranksum tests of the MO methods using original and noisy datasets. The corresponding p-values are given in the parentheses.

Compared algorithms A -vs- B		Training set	Testing set
Random WA (original)	Random WA (noisy)	+ (0.0000)	+ (0.0164)
Bang WA (original)	Bang WA (noisy)	+ (0.0000)	= (0.1567)
Dynamic WA (original)	Dynamic WA (noisy)	+ (0.0000)	= (0.1031)
NSGA-II (original)	NSGA-II (noisy)	+ (0.0000)	= (0.1744)
MOPSO (original)	MOPSO (noisy)	+ (0.0000)	+ (0.0199)
["+": A is better than B]	["−": B is better than A]	["=": indifferent]	

Table 4.19: Training MSE using the red wine dataset.

	Setting 1					Setting 2				
Algorithm	min	mean	median	max	std	min	mean	median	max	std
Random WA	0.5478	0.5744	0.5571	0.7515	0.0443	**0.0214**	0.0371	0.0257	0.1196	0.0245
Bang WA	0.5542	0.6033	0.5915	0.6971	0.0423	0.0250	0.0574	0.0373	0.1533	0.0370
Dynamic WA	0.5338	0.5645	0.5426	0.7660	0.0493	0.0221	0.0342	0.0315	0.0851	0.0144
NSGA-II	0.5568	0.5605	0.5606	0.5669	**0.0019**	0.0225	0.0230	0.0228	0.0249	**0.0005**
MOPSO	**0.5247**	**0.5295**	**0.5287**	**0.5409**	0.0045	**0.0214**	**0.0223**	**0.0221**	**0.0247**	0.0007
BFGS	0.5421	815.4536	0.5486	4193.6810	1401.0972	0.0220	489.9167	0.0226	3914.3024	999.6690

and quality assessment and assurance processes are more controlled.

In order to analyze the red wine dataset we made some modifications in our implementation. First, the size of the dataset was reduced from 40000 to 1599 patterns. Also, we noticed that the values of the fixed acidity are in range 4.6 to 15.9 and the values of volatile acidity are in range 0.12 to 1.58. So, we chose to normalize the input data. As a consequence the minimum and maximum value of the center components became equal to 0 and 1, respectively. As concerns the output data we distinguished 2 settings. In the first setting, we do not normalize the output data, while in the second one, we normalize it. The basic statistical values of MSE in both training and testing sets in the examined settings are reported in Tables 4.19 and 4.20.

As for the former datasets, the first step of the statistical analysis was the application of normality tests in all samples using the one-sample Kolmogorov-Smirnov test. The results showed that none of the samples came from the normal distribution at a 0.05 significance level. So, we compared the methods, using the non-parametric Wilcoxon ranksum test.

From the pairwise comparisons between the methods reported in Table 4.21, we notice that in both examined settings, the performances of NSGA-II and MOPSO were superior to the other MO methods.

Table 4.20: Testing MSE using the red wine dataset.

	Setting 1					Setting 2				
Algorithm	min	mean	median	max	std	min	mean	median	max	std
Random WA	0.5532	0.5774	0.5608	0.7301	0.0405	0.0243	0.0437	0.0287	0.1530	0.0309
Bang WA	0.5359	0.5904	0.5806	0.7056	0.0490	0.0237	0.0543	0.0375	0.1468	0.0346
Dynamic WA	0.5710	0.6282	0.5894	1.0600	0.1043	0.0221	0.0346	0.0316	0.0857	0.0150
NSGA-II	**0.5346**	**0.5417**	**0.5399**	**0.5680**	0.0070	**0.0208**	**0.0215**	**0.0214**	**0.0228**	**0.0005**
MOPSO	0.5866	0.5934	0.5924	0.6076	**0.0057**	0.0226	0.0243	0.0243	0.0267	0.0011
BFGS	0.5592	822.1290	0.5758	4194.9224	1409.2182	0.0210	493.5759	0.0228	3939.9349	1005.6667

Table 4.21: Wilcoxon ranksum tests of the MO methods using the red wine dataset. The corresponding p-values are given in the parentheses.

Compared algorithms		Setting 1		Setting 2	
A	B	Training set	Testing set	Training set	Testing set
Random WA	Bang WA	$+$ (0.0003)	$=$ (0.5475)	$+$ (0.0008)	$=$ (0.1116)
Random WA	Dynamic WA	$-$ (0.0099)	$+$ (0.0000)	$=$ (0.3933)	$=$ (0.2523)
Random WA	NSGA-II	$=$ (0.8009)	$-$ (0.0000)	$-$ (0.0090)	$-$ (0.0000)
Random WA	MOPSO	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
Bang WA	Dynamic WA	$-$ (0.0001)	$=$ (0.0598)	$-$ (0.0029)	$-$ (0.0085)
Bang WA	NSGA-II	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
Bang WA	MOPSO	$-$ (0.0000)	$=$ (0.0991)	$-$ (0.0000)	$-$ (0.0000)
Dynamic WA	NSGA-II	$+$ (0.0164)	$-$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
Dynamic WA	MOPSO	$-$ (0.0000)	$=$ (0.5869)	$-$ (0.0000)	$-$ (0.0018)
NSGA-II	MOPSO	$-$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$+$ (0.0000)
[“+”: A is better than B]		[“−”: B is better than A]		[“=”: indifferent]	

Subsequently, we statistically compared NSGA-II and MOPSO with the BFGS method, distinguishing 3 cases, as in the previous datasets. In Table 4.22 all the p-values of the pairwise comparisons are reported. In the first case we compared the methods using the proposed set of parameters of each method. From the pairwise comparisons in both examined settings we can see that the performance of NSGA-II was superior to BFGS in testing, while it was competitive to BFGS in training. Also, the performance of MOPSO was superior to BFGS in training, while it was competitive to BFGS in testing.

In the second case we compared the optimization methods when all of them used 5 neurons. Just like in the first case, from the pairwise comparisons in both examined settings the performance of NSGA-II was superior to the BFGS method in testing, while it was competitive to BFGS in training. Also, the performance of MOPSO was superior to the BFGS method in training, while it was competitive to the BFGS method in testing.

Table 4.22: Wilcoxon ranksum tests of the optimization methods using the red wine dataset. The corresponding p-values are given in the parentheses.

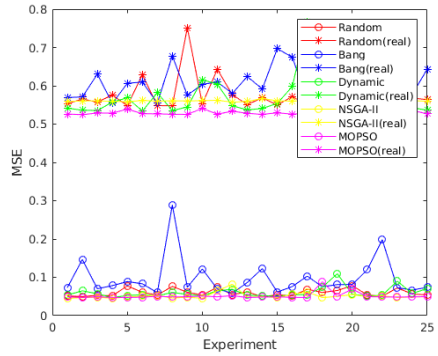
Setting	Algorithm	Case 1		Case 2		Case 3	
		Training set	Testing set	Training set	Testing set	Training set	Testing set
1	NSGA-II	$=$ (0.2990)	$+$ (0.0000)	$=$ (0.1744)	$+$ (0.0000)	$+$ (0.0000)	$=$ (0.1073)
	MOPSO	$+$ (0.0000)	$=$ (0.0775)	$+$ (0.0000)	$=$ (0.7269)	$+$ (0.0000)	$=$ (0.0598)
2	NSGA-II	$=$ (0.6837)	$+$ (0.0002)	$=$ (0.1511)	$+$ (0.0000)	$=$ (0.3618)	$=$ (0.9690)
	MOPSO	$+$ (0.0002)	$=$ (0.2604)	$=$ (0.0743)	$=$ (0.5737)	$=$ (0.2444)	$=$ (0.1253)
["+": A is better than BFGS] ["−": BFGS is better than A] ["=": indifferent]							

Table 4.23: Wilcoxon ranksum tests of the MO methods using original and red wine datasets. The corresponding p-values are given in the parentheses.

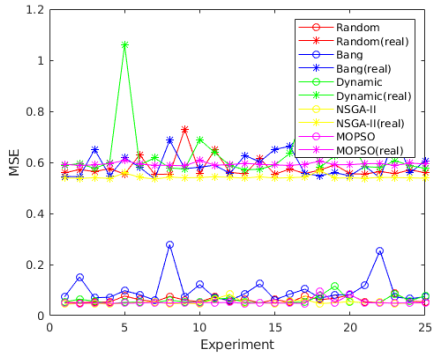
Compared algorithms		Setting 1		Setting 2	
A -vs-	B	Training set	Testing set	Training set	Testing set
Random WA (original)	Random WA (red wine)	$+$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
Bang WA (original)	Bang WA (red wine)	$+$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
Dynamic WA (original)	Dynamic WA (red wine)	$+$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
NSGA-II (original)	NSGA-II (red wine)	$+$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
MOPSO (original)	MOPSO (red wine)	$+$ (0.0000)	$+$ (0.0000)	$-$ (0.0000)	$-$ (0.0000)
["+": A is better than B] ["−": B is better than A] ["=": indifferent]					

In the third case we compared the methods when all of them used 10 neurons. From the pairwise comparisons we can notice that in the first setting the performances of the MO methods were competitive to BFGS in testing, while they were superior to the BFGS method in training. In the second setting the performances of the MO methods were competitive to BFGS in both the training and testing phases.

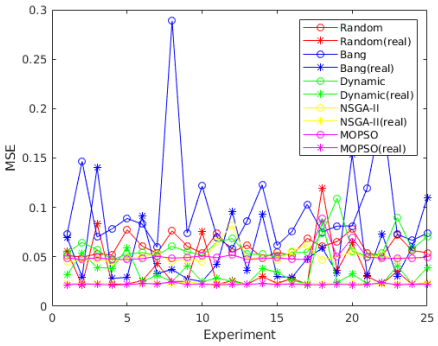
In Fig.(4.4), the training and testing MSE of both original and red wine datasets using MO methods in the examined settings are illustrated. Next, we compared statistically each MO method for these datasets. From the pairwise comparisons between the methods reported in Table 4.23, we can see that as concerns the first setting the performances of the MO methods were better when they used the original dataset, while in the second setting the performances of the MO methods were better when they used the red wine dataset.



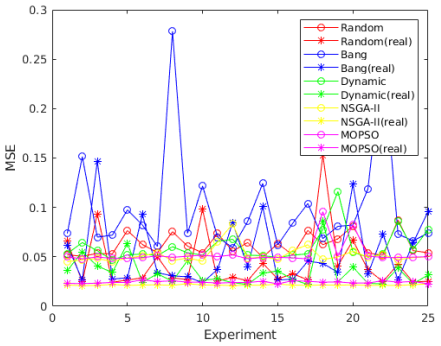
(a) Training MSE (Setting 1)



(b) Testing MSE (Setting 1)



(c) Training MSE (Setting 2)



(d) Testing MSE (Setting 2)

Figure 4.4: Training and Testing MSE FOR original and red wine datasets.

4.5 Why do we select variance counterbalancing?

The present thesis studies the use of MO methods for solving the VCB minimization problem. The last part of our analysis contained comparisons of NSGA-II, MOPSO and the BFGS method when they used and when they ignored the VCB technique in training of the neural network, based on the resulting MSE of the whole network. The basic statistical values of MSE of the three methods in both training and testing sets are illustrated in Tables 4.24 and 4.25, while the pairwise comparisons of the methods in the examined cases are reported in Table 4.26.

To begin with, we selected the original dataset. From the pairwise comparisons between the methods we can see that there is no statistically significant difference between the methods when they used or when they ignored the VCB technique. Only MOPSO performed better in training when it employed the VCB. At the same time, as concerns the NSGA-II algorithm in both training and testing phases had smaller MSE mean when it used the VCB technique, while the results about the standard

Table 4.24: Training MSE for VCB selection.

		Original dataset					Noisy dataset				
	Algorithm	min	mean	median	max	std	min	mean	median	max	std
VCB	NSGA-II	0.0450	**0.0513**	**0.0486**	0.0819	0.0080	0.0833	0.0894	0.0894	0.1098	**0.0051**
	MOPSO	0.0472	**0.0513**	0.0489	0.0885	0.0088	0.0817	0.0936	0.0907	0.1687	0.0159
	BFGS	0.0496	0.1574	0.0705	1.8721	0.3631	**0.0494**	0.4396	**0.0680**	9.1855	1.8226
no VCB	NSGA-II	**0.0439**	0.0521	0.0495	**0.0674**	**0.0063**	0.0814	0.0916	0.0889	0.1301	0.0106
	MOPSO	0.0448	0.0557	0.0504	0.1022	0.0131	0.0800	0.0991	0.0909	0.1424	0.0171
	BFGS	0.0501	0.1042	0.0692	0.9539	0.1772	0.0517	**0.0698**	0.0697	**0.0853**	0.0076

Table 4.25: Testing MSE for VCB selection.

		Original dataset					Noisy dataset				
	Algorithm	min	mean	median	max	std	min	mean	median	max	std
VCB	NSGA-II	0.0451	**0.0518**	**0.0492**	0.0833	0.0082	0.0833	0.0894	0.0894	0.1098	**0.0051**
	MOPSO	0.0479	0.0529	0.0497	0.0957	0.0112	0.0471	0.0545	0.0512	0.1189	0.0138
	BFGS	0.0484	0.1548	0.0684	1.9073	0.3711	0.0494	0.4591	0.0694	9.6090	1.9067
no VCB	NSGA-II	**0.0439**	0.0523	0.0499	**0.0695**	**0.0060**	0.0459	**0.0538**	0.0519	**0.0809**	0.0081
	MOPSO	0.0444	0.0548	0.0504	0.0829	0.0106	**0.0432**	0.0582	**0.0509**	0.1150	0.0157
	BFGS	0.0494	0.1064	0.0686	0.9542	0.1773	0.0510	0.0712	0.0695	0.1049	0.0120

deviation of MSE were better when the VCB was not applied. As concerns the MOPSO method we observe that in training both the MSE mean and standard deviation were smaller when the VCB was used, while in testing the results were mixed, since the MSE mean was smaller when the VCB was applied and the standard deviation of MSE was smaller when the VCB was ignored.

These results were the motivation to make the same comparisons using the noisy dataset. From the pairwise comparisons between the methods we observe that there is no statistically significant difference between the methods when they used or when they neglected the VCB technique. Moreover, NSGA-II in training had smaller values in both the MSE mean and standard deviation when it used the VCB technique, while in testing the results were mixed, since the MSE mean was smaller when the VCB was ignored and the standard deviation of MSE was smaller when the VCB was used. As concerns the MOPSO algorithm we can see that in both phases the MSE mean and standard deviation were smaller when the VCB was used. These results indicate that applying the VCB technique in training of an RBF network using NSGA-II and MOPSO can provide robust results.

Table 4.26: Wilcoxon ranksum tests of the methods for VCB selection. The corresponding p-values are given in the parentheses.

Compared algorithms		Original dataset		Noisy dataset	
A -vs-	B	Training set	Testing set	Training set	Testing set
NSGA-II (VCB)	NSGA-II (no VCB)	$=$ (0.2143)	$=$ (0.2003)	$=$ (0.8009)	$=$ (0.6276)
MOPSO (VCB)	MOPSO (no VCB)	$+$ (0.0105)	$=$ (0.3130)	$=$ (0.3826)	$=$ (0.6837)
BFGS (VCB)	BFGS (no VCB)	$=$ (0.1683)	$=$ (0.7562)	$=$ (0.3320)	$=$ (0.8462)
["+": A is better than B] ["−": B is better than A] ["=": indifferent]					

Chapter 5

Conclusions

We studied the use of multi-objective methods for the recently proposed variance counterbalancing algorithm for large-scale stochastic learning. The employed optimization methods included the state-of-the-art weighted aggregation approaches, the non-sorting genetic algorithm, and the multi-objective particle swarm optimization algorithm. Our experimental analysis was based on the application of variance counterbalancing on RBF networks, using three different datasets.

Having determined appropriate sets of parameters for each method, we statistically compared them. Also, we compared the best-performing multi-objective approaches with the standard single-objective formulation of variance counterbalancing, that employs the BFGS solver. The numerical experiments that we conducted and the accompanying statistical analysis suggest that the multi-objective methods with our proposed modifications, can be competitive and frequently superior to the original single-objective variance counterbalancing approach.

In future extensions, the variance counterbalancing algorithm can be applied on a variety of neural networks using real-world datasets, as well as on problems of higher dimension.

Bibliography

[1] J. Brownlee. A gentle introduction to mini-batch gradient descent and how to configure batch size. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

[2] P. Lagari, L. Tsoukalas, and I. Lagaris, *Variance Counterbalancing for Stochastic Large-scale Learning*, ser. International Journal on Artificial Intelligence Tools. World Scientific, 2020, vol. 29.

[3] N. Karayiannis, "Reformulated radial basis neural networks trained by gradient descent," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 657–671, 1999.

[4] C. Coello Coello, G. Lamont, and D. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed., ser. Genetic Algorithms and Evolutionary Computation. Springer US, 2007.

[5] Y. Jin, M. Olhofer, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?" in *Genetic and Evolutionary Computation Conference*, 2001, pp. 1042–1049.

[6] K. Parsopoulos and M. Vrahatis, "Particle swarm optimization method in multi-objective problems," in *ACM Symposium on Applied Computing (SAC 2002)*, 2002, pp. 603–607.

[7] Y. Liu, Q. Zheng, Z. Shi, and J. Chen, "Training radial basis function networks with particle swarms," in *Advances in Neural Networks – ISNN 2004*, 2004, pp. 317–322.

[8] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, 2nd ed., ser. Natural Computing Series. Springer, 2015.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[10] C. Coello and G. Pulido, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[11] C. Coello and R. Sierra, "Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance," in *Evolutionary Multi-Criterion Optimization*, 2005, pp. 505–519.

[12] T. Kurban and E. Beşdok, "A comparison of rbf neural network training algorithms for inertial sensor based terrain classification," *Sensors (Basel)*, vol. 9, no. 8, pp. 6312–29, 2009.

[13] G. Montager, D. Giveki, M. Karami, and H. Rastegar, "Radial basis function neural networks: A review," *Computer Reviews*, vol. 1, no. 1, 2018.

[14] S. Ding, L. Xu, C. Su, and F. Jin, "An optimizing method of rbf neural network based on genetic algorithm," *Neural Comput Applic*, vol. 21, pp. 333–336, 2012.

[15] R. Marler and J. Arora, *Survey of multi-objective optimization methods for engineering*, ser. Structural and Multidisciplinary Optimization. Springer, 2004, vol. 26.

[16] M. Rocha and J. Neves, *Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation*, ser. Multiple Approaches to Intelligent Systems. Springer, 1999, vol. 1611.

[17] G. Kanji, *100 Statistical Tests*, 3rd ed. SAGE Publications, 2006.

[18] D. Nguyen. Red wine quality prediction using regression modeling and machine learning. [Online]. Available: https://towardsdatascience.com/red-wine-quality-prediction-using-regression-modeling-and-machine-learning

Appendix A

Appendix

The following pages contain all tables with the relevant settings and the boxplots of the attained MSE values for each optimization method for the experimental phase 1.

A.1 Examined cases for the weighted aggregation approaches

Table A.1: Examined cases for setting 1 (*pd*) of the wa approaches.

ID	Examined cases	ID	Examined cases
1(49)	vcb=10 str=1 ps=20 vel=1 rest=0	25(73)	vcb=20 str=1 ps=20 vel=1 rest=0
2(50)	vcb=10 str=1 ps=20 vel=1 rest=1	26(74)	vcb=20 str=1 ps=20 vel=1 rest=1
3(51)	vcb=10 str=1 ps=20 vel=2 rest=0	27(75)	vcb=20 str=1 ps=20 vel=2 rest=0
4(52)	vcb=10 str=1 ps=20 vel=2 rest=1	28(76)	vcb=20 str=1 ps=20 vel=2 rest=1
5(53)	vcb=10 str=1 ps=40 vel=1 rest=0	29(77)	vcb=20 str=1 ps=40 vel=1 rest=0
6(54)	vcb=10 str=1 ps=40 vel=1 rest=1	30(78)	vcb=20 str=1 ps=40 vel=1 rest=1
7(55)	vcb=10 str=1 ps=40 vel=2 rest=0	31(79)	vcb=20 str=1 ps=40 vel=2 rest=0
8(56)	vcb=10 str=1 ps=40 vel=2 rest=1	32(80)	vcb=20 str=1 ps=40 vel=2 rest=1
9(57)	vcb=10 str=2 ps=20 vel=1 rest=0	33(81)	vcb=20 str=2 ps=20 vel=1 rest=0
10(58)	vcb=10 str=2 ps=20 vel=1 rest=1	34(82)	vcb=20 str=2 ps=20 vel=1 rest=1
11(59)	vcb=10 str=2 ps=20 vel=2 rest=0	35(83)	vcb=20 str=2 ps=20 vel=2 rest=0
12(60)	vcb=10 str=2 ps=20 vel=2 rest=1	36(84)	vcb=20 str=2 ps=20 vel=2 rest=1
13(61)	vcb=10 str=2 ps=40 vel=1 rest=0	37(85)	vcb=20 str=2 ps=40 vel=1 rest=0
14(62)	vcb=10 str=2 ps=40 vel=1 rest=1	38(86)	vcb=20 str=2 ps=40 vel=1 rest=1
15(63)	vcb=10 str=2 ps=40 vel=2 rest=0	39(87)	vcb=20 str=2 ps=40 vel=2 rest=0
16(64)	vcb=10 str=2 ps=40 vel=2 rest=1	40(88)	vcb=20 str=2 ps=40 vel=2 rest=1
17(65)	vcb=10 str=3 ps=20 vel=1 rest=0	41(89)	vcb=20 str=3 ps=20 vel=1 rest=0
18(66)	vcb=10 str=3 ps=20 vel=1 rest=1	42(90)	vcb=20 str=3 ps=20 vel=1 rest=1
19(67)	vcb=10 str=3 ps=20 vel=2 rest=0	43(91)	vcb=20 str=3 ps=20 vel=2 rest=0
20(68)	vcb=10 str=3 ps=20 vel=2 rest=1	44(92)	vcb=20 str=3 ps=20 vel=2 rest=1
21(69)	vcb=10 str=3 ps=40 vel=1 rest=0	45(93)	vcb=20 str=3 ps=40 vel=1 rest=0
22(70)	vcb=10 str=3 ps=40 vel=1 rest=1	46(94)	vcb=20 str=3 ps=40 vel=1 rest=1
23(71)	vcb=10 str=3 ps=40 vel=2 rest=0	47(95)	vcb=20 str=3 ps=40 vel=2 rest=0
24(72)	vcb=10 str=3 ps=40 vel=2 rest=1	48(96)	vcb=20 str=3 ps=40 vel=2 rest=1
The first ID number is for n=5, while inside the parenthesis is for n=10			

Table A.2: Examined cases for setting 2 (n) of the wa approaches.

ID	Examined cases	ID	Examined cases
1(49)	vcb=10 str=1 ps=20 vel=1 rest=0	25(73)	vcb=20 str=1 ps=20 vel=1 rest=0
2(50)	vcb=10 str=1 ps=20 vel=1 rest=1	26(74)	vcb=20 str=1 ps=20 vel=1 rest=1
3(51)	vcb=10 str=1 ps=20 vel=2 rest=0	27(75)	vcb=20 str=1 ps=20 vel=2 rest=0
4(52)	vcb=10 str=1 ps=20 vel=2 rest=1	28(76)	vcb=20 str=1 ps=20 vel=2 rest=1
5(53)	vcb=10 str=1 ps=40 vel=1 rest=0	29(77)	vcb=20 str=1 ps=40 vel=1 rest=0
6(54)	vcb=10 str=1 ps=40 vel=1 rest=1	30(78)	vcb=20 str=1 ps=40 vel=1 rest=1
7(55)	vcb=10 str=1 ps=40 vel=2 rest=0	31(79)	vcb=20 str=1 ps=40 vel=2 rest=0
8(56)	vcb=10 str=1 ps=40 vel=2 rest=1	32(80)	vcb=20 str=1 ps=40 vel=2 rest=1
9(57)	vcb=10 str=2 ps=20 vel=1 rest=0	33(81)	vcb=20 str=2 ps=20 vel=1 rest=0
10(58)	vcb=10 str=2 ps=20 vel=1 rest=1	34(82)	vcb=20 str=2 ps=20 vel=1 rest=1
11(59)	vcb=10 str=2 ps=20 vel=2 rest=0	35(83)	vcb=20 str=2 ps=20 vel=2 rest=0
12(60)	vcb=10 str=2 ps=20 vel=2 rest=1	36(84)	vcb=20 str=2 ps=20 vel=2 rest=1
13(61)	vcb=10 str=2 ps=40 vel=1 rest=0	37(85)	vcb=20 str=2 ps=40 vel=1 rest=0
14(62)	vcb=10 str=2 ps=40 vel=1 rest=1	38(86)	vcb=20 str=2 ps=40 vel=1 rest=1
15(63)	vcb=10 str=2 ps=40 vel=2 rest=0	39(87)	vcb=20 str=2 ps=40 vel=2 rest=0
16(64)	vcb=10 str=2 ps=40 vel=2 rest=1	40(88)	vcb=20 str=2 ps=40 vel=2 rest=1
17(65)	vcb=10 str=3 ps=20 vel=1 rest=0	41(89)	vcb=20 str=3 ps=20 vel=1 rest=0
18(66)	vcb=10 str=3 ps=20 vel=1 rest=1	42(90)	vcb=20 str=3 ps=20 vel=1 rest=1
19(67)	vcb=10 str=3 ps=20 vel=2 rest=0	43(91)	vcb=20 str=3 ps=20 vel=2 rest=0
20(68)	vcb=10 str=3 ps=20 vel=2 rest=1	44(92)	vcb=20 str=3 ps=20 vel=2 rest=1
21(69)	vcb=10 str=3 ps=40 vel=1 rest=0	45(93)	vcb=20 str=3 ps=40 vel=1 rest=0
22(70)	vcb=10 str=3 ps=40 vel=1 rest=1	46(94)	vcb=20 str=3 ps=40 vel=1 rest=1
23(71)	vcb=10 str=3 ps=40 vel=2 rest=0	47(95)	vcb=20 str=3 ps=40 vel=2 rest=0
24(72)	vcb=10 str=3 ps=40 vel=2 rest=1	48(96)	vcb=20 str=3 ps=40 vel=2 rest=1
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			

Table A.3: Examined cases for setting 3 (vcb) of the wa approaches.

ID	Examined cases	ID	Examined cases
1(49)	n=5 str=1 ps=20 vel=1 rest=0	25(73)	n=10 str=1 ps=20 vel=1 rest=0
2(50)	n=5 str=1 ps=20 vel=1 rest=1	26(74)	n=10 str=1 ps=20 vel=1 rest=1
3(51)	n=5 str=1 ps=20 vel=2 rest=0	27(75)	n=10 str=1 ps=20 vel=2 rest=0
4(52)	n=5 str=1 ps=20 vel=2 rest=1	28(76)	n=10 str=1 ps=20 vel=2 rest=1
5(53)	n=5 str=1 ps=40 vel=1 rest=0	29(77)	n=10 str=1 ps=40 vel=1 rest=0
6(54)	n=5 str=1 ps=40 vel=1 rest=1	30(78)	n=10 str=1 ps=40 vel=1 rest=1
7(55)	n=5 str=1 ps=40 vel=2 rest=0	31(79)	n=10 str=1 ps=40 vel=2 rest=0
8(56)	n=5 str=1 ps=40 vel=2 rest=1	32(80)	n=10 str=1 ps=40 vel=2 rest=1
9(57)	n=5 str=2 ps=20 vel=1 rest=0	33(81)	n=10 str=2 ps=20 vel=1 rest=0
10(58)	n=5 str=2 ps=20 vel=1 rest=1	34(82)	n=10 str=2 ps=20 vel=1 rest=1
11(59)	n=5 str=2 ps=20 vel=2 rest=0	35(83)	n=10 str=2 ps=20 vel=2 rest=0
12(60)	n=5 str=2 ps=20 vel=2 rest=1	36(84)	n=10 str=2 ps=20 vel=2 rest=1
13(61)	n=5 str=2 ps=40 vel=1 rest=0	37(85)	n=10 str=2 ps=40 vel=1 rest=0
14(62)	n=5 str=2 ps=40 vel=1 rest=1	38(86)	n=10 str=2 ps=40 vel=1 rest=1
15(63)	n=5 str=2 ps=40 vel=2 rest=0	39(87)	n=10 str=2 ps=40 vel=2 rest=0
16(64)	n=5 str=2 ps=40 vel=2 rest=1	40(88)	n=10 str=2 ps=40 vel=2 rest=1
17(65)	n=5 str=3 ps=20 vel=1 rest=0	41(89)	n=10 str=3 ps=20 vel=1 rest=0
18(66)	n=5 str=3 ps=20 vel=1 rest=1	42(90)	n=10 str=3 ps=20 vel=1 rest=1
19(67)	n=5 str=3 ps=20 vel=2 rest=0	43(91)	n=10 str=3 ps=20 vel=2 rest=0
20(68)	n=5 str=3 ps=20 vel=2 rest=1	44(92)	n=10 str=3 ps=20 vel=2 rest=1
21(69)	n=5 str=3 ps=40 vel=1 rest=0	45(93)	n=10 str=3 ps=40 vel=1 rest=0
22(70)	n=5 str=3 ps=40 vel=1 rest=1	46(94)	n=10 str=3 ps=40 vel=1 rest=1
23(71)	n=5 str=3 ps=40 vel=2 rest=0	47(95)	n=10 str=3 ps=40 vel=2 rest=0
24(72)	n=5 str=3 ps=40 vel=2 rest=1	48(96)	n=10 str=3 ps=40 vel=2 rest=1
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			

Table A.4: Examined cases for setting 4 (str) of the wa approaches.

ID	Examined cases	ID	Examined cases
1(33)	n=5 vcb=10 ps=20 vel=1 rest=0	17(49)	n=10 vcb=10 ps=20 vel=1 rest=0
2(34)	n=5 vcb=10 ps=20 vel=1 rest=1	18(50)	n=10 vcb=10 ps=20 vel=1 rest=1
3(35)	n=5 vcb=10 ps=20 vel=2 rest=0	19(51)	n=10 vcb=10 ps=20 vel=2 rest=0
4(36)	n=5 vcb=10 ps=20 vel=2 rest=1	20(52)	n=10 vcb=10 ps=20 vel=2 rest=1
5(37)	n=5 vcb=10 ps=40 vel=1 rest=0	21(53)	n=10 vcb=10 ps=40 vel=1 rest=0
6(38)	n=5 vcb=10 ps=40 vel=1 rest=1	22(54)	n=10 vcb=10 ps=40 vel=1 rest=1
7(39)	n=5 vcb=10 ps=40 vel=2 rest=0	23(55)	n=10 vcb=10 ps=40 vel=2 rest=0
8(40)	n=5 vcb=10 ps=40 vel=2 rest=1	24(56)	n=10 vcb=10 ps=40 vel=2 rest=1
9(41)	n=5 vcb=20 ps=20 vel=1 rest=0	25(57)	n=10 vcb=20 ps=20 vel=1 rest=0
10(42)	n=5 vcb=20 ps=20 vel=1 rest=1	26(58)	n=10 vcb=20 ps=20 vel=1 rest=1
11(43)	n=5 vcb=20 ps=20 vel=2 rest=0	27(59)	n=10 vcb=20 ps=20 vel=2 rest=0
12(44)	n=5 vcb=20 ps=20 vel=2 rest=1	28(60)	n=10 vcb=20 ps=20 vel=2 rest=1
13(45)	n=5 vcb=20 ps=40 vel=1 rest=0	29(61)	n=10 vcb=20 ps=40 vel=1 rest=0
14(46)	n=5 vcb=20 ps=40 vel=1 rest=1	30(62)	n=10 vcb=20 ps=40 vel=1 rest=1
15(47)	n=5 vcb=20 ps=40 vel=2 rest=0	31(63)	n=10 vcb=20 ps=40 vel=2 rest=0
16(48)	n=5 vcb=20 ps=40 vel=2 rest=1	32(64)	n=10 vcb=20 ps=40 vel=2 rest=1
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			

Table A.5: Examined cases for setting 5 (ps) of the wa approaches.

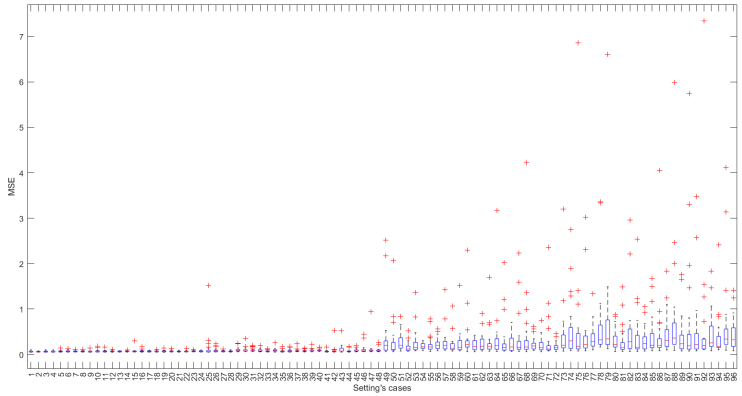
ID	Examined cases	ID	Examined cases
1(49)	n=5 vcb=10 str=1 vel=1 rest=0	25(73)	n=10 vcb=10 str=1 vel=1 rest=0
2(50)	n=5 vcb=10 str=1 vel=1 rest=1	26(74)	n=10 vcb=10 str=1 vel=1 rest=1
3(51)	n=5 vcb=10 str=1 vel=2 rest=0	27(75)	n=10 vcb=10 str=1 vel=2 rest=0
4(52)	n=5 vcb=10 str=1 vel=2 rest=1	28(76)	n=10 vcb=10 str=1 vel=2 rest=1
5(53)	n=5 vcb=10 str=2 vel=1 rest=0	29(77)	n=10 vcb=10 str=2 vel=1 rest=0
6(54)	n=5 vcb=10 str=2 vel=1 rest=1	30(78)	n=10 vcb=10 str=2 vel=1 rest=1
7(55)	n=5 vcb=10 str=2 vel=2 rest=0	31(79)	n=10 vcb=10 str=2 vel=2 rest=0
8(56)	n=5 vcb=10 str=2 vel=2 rest=1	32(80)	n=10 vcb=10 str=2 vel=2 rest=1
9(57)	n=5 vcb=10 str=3 vel=1 rest=0	33(81)	n=10 vcb=10 str=3 vel=1 rest=0
10(58)	n=5 vcb=10 str=3 vel=1 rest=1	34(82)	n=10 vcb=10 str=3 vel=1 rest=1
11(59)	n=5 vcb=10 str=3 vel=2 rest=0	35(83)	n=10 vcb=10 str=3 vel=2 rest=0
12(60)	n=5 vcb=10 str=3 vel=2 rest=1	36(84)	n=10 vcb=10 str=3 vel=2 rest=1
13(61)	n=5 vcb=20 str=1 vel=1 rest=0	37(85)	n=10 vcb=20 str=1 vel=1 rest=0
14(62)	n=5 vcb=20 str=1 vel=1 rest=1	38(86)	n=10 vcb=20 str=1 vel=1 rest=1
15(63)	n=5 vcb=20 str=1 vel=2 rest=0	39(87)	n=10 vcb=20 str=1 vel=2 rest=0
16(64)	n=5 vcb=20 str=1 vel=2 rest=1	40(88)	n=10 vcb=20 str=1 vel=2 rest=1
17(65)	n=5 vcb=20 str=2 vel=1 rest=0	41(89)	n=10 vcb=20 str=2 vel=1 rest=0
18(66)	n=5 vcb=20 str=2 vel=1 rest=1	42(90)	n=10 vcb=20 str=2 vel=1 rest=1
19(67)	n=5 vcb=20 str=2 vel=2 rest=0	43(91)	n=10 vcb=20 str=2 vel=2 rest=0
20(68)	n=5 vcb=20 str=2 vel=2 rest=1	44(92)	n=10 vcb=20 str=2 vel=2 rest=1
21(69)	n=5 vcb=20 str=3 vel=1 rest=0	45(93)	n=10 vcb=20 str=3 vel=1 rest=0
22(70)	n=5 vcb=20 str=3 vel=1 rest=1	46(94)	n=10 vcb=20 str=3 vel=1 rest=1
23(71)	n=5 vcb=20 str=3 vel=2 rest=0	47(95)	n=10 vcb=20 str=3 vel=2 rest=0
24(72)	n=5 vcb=20 str=3 vel=2 rest=1	48(96)	n=10 vcb=20 str=3 vel=2 rest=1
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			

Table A.6: Examined cases for setting 6 (vel) of the wa approaches.

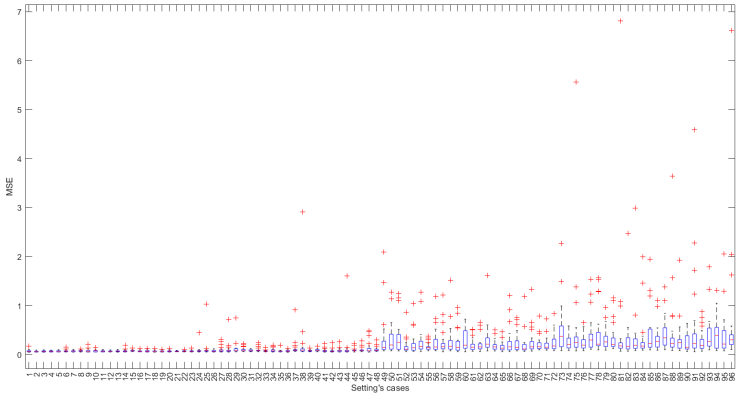
ID	Examined cases	ID	Examined cases
1(49)	n=5 vcb=10 str=1 ps=20 rest=0	25(73)	n=10 vcb=10 str=1 ps=20 rest=0
2(50)	n=5 vcb=10 str=1 ps=20 rest=1	26(74)	n=10 vcb=10 str=1 ps=20 rest=1
3(51)	n=5 vcb=10 str=1 ps=40 rest=0	27(75)	n=10 vcb=10 str=1 ps=40 rest=0
4(52)	n=5 vcb=10 str=1 ps=40 rest=1	28(76)	n=10 vcb=10 str=1 ps=40 rest=1
5(53)	n=5 vcb=10 str=2 ps=20 rest=0	29(77)	n=10 vcb=10 str=2 ps=20 rest=0
6(54)	n=5 vcb=10 str=2 ps=20 rest=1	30(78)	n=10 vcb=10 str=2 ps=20 rest=1
7(55)	n=5 vcb=10 str=2 ps=40 rest=0	31(79)	n=10 vcb=10 str=2 ps=40 rest=0
8(56)	n=5 vcb=10 str=2 ps=40 rest=1	32(80)	n=10 vcb=10 str=2 ps=40 rest=1
9(57)	n=5 vcb=10 str=3 ps=20 rest=0	33(81)	n=10 vcb=10 str=3 ps=20 rest=0
10(58)	n=5 vcb=10 str=3 ps=20 rest=1	34(82)	n=10 vcb=10 str=3 ps=20 rest=1
11(59)	n=5 vcb=10 str=3 ps=40 rest=0	35(83)	n=10 vcb=10 str=3 ps=40 rest=0
12(60)	n=5 vcb=10 str=3 ps=40 rest=1	36(84)	n=10 vcb=10 str=3 ps=40 rest=1
13(61)	n=5 vcb=20 str=1 ps=20 rest=0	37(85)	n=10 vcb=20 str=1 ps=20 rest=0
14(62)	n=5 vcb=20 str=1 ps=20 rest=1	38(86)	n=10 vcb=20 str=1 ps=20 rest=1
15(63)	n=5 vcb=20 str=1 ps=40 rest=0	39(87)	n=10 vcb=20 str=1 ps=40 rest=0
16(64)	n=5 vcb=20 str=1 ps=40 rest=1	40(88)	n=10 vcb=20 str=1 ps=40 rest=1
17(65)	n=5 vcb=20 str=2 ps=20 rest=0	41(89)	n=10 vcb=20 str=2 ps=20 rest=0
18(66)	n=5 vcb=20 str=2 ps=20 rest=1	42(90)	n=10 vcb=20 str=2 ps=20 rest=1
19(67)	n=5 vcb=20 str=2 ps=40 rest=0	43(91)	n=10 vcb=20 str=2 ps=40 rest=0
20(68)	n=5 vcb=20 str=2 ps=40 rest=1	44(92)	n=10 vcb=20 str=2 ps=40 rest=1
21(69)	n=5 vcb=20 str=3 ps=20 rest=0	45(93)	n=10 vcb=20 str=3 ps=20 rest=0
22(70)	n=5 vcb=20 str=3 ps=20 rest=1	46(94)	n=10 vcb=20 str=3 ps=20 rest=1
23(71)	n=5 vcb=20 str=3 ps=40 rest=0	47(95)	n=10 vcb=20 str=3 ps=40 rest=0
24(72)	n=5 vcb=20 str=3 ps=40 rest=1	48(96)	n=10 vcb=20 str=3 ps=40 rest=1
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			

Table A.7: Examined cases for setting 7 ($rest$) of the wa approaches.

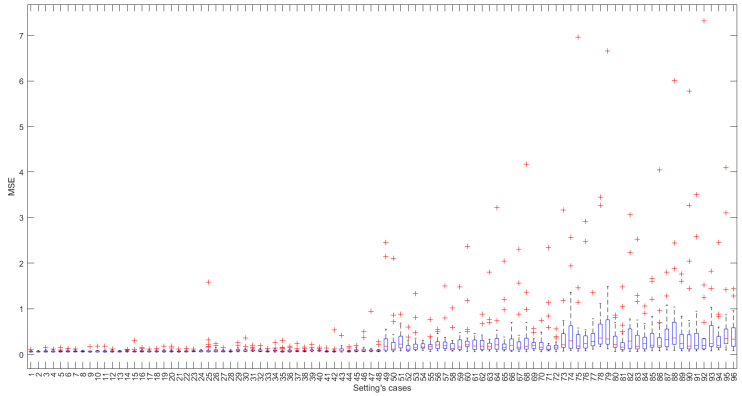
ID	Examined cases	ID	Examined cases
1(49)	n=5 vcb=10 str=1 ps=20 vel=1	25(73)	n=10 vcb=10 str=1 ps=20 vel=1
2(50)	n=5 vcb=10 str=1 ps=20 vel=2	26(74)	n=10 vcb=10 str=1 ps=20 vel=2
3(51)	n=5 vcb=10 str=1 ps=40 vel=1	27(75)	n=10 vcb=10 str=1 ps=40 vel=1
4(52)	n=5 vcb=10 str=1 ps=40 vel=2	28(76)	n=10 vcb=10 str=1 ps=40 vel=2
5(53)	n=5 vcb=10 str=2 ps=20 vel=1	29(77)	n=10 vcb=10 str=2 ps=20 vel=1
6(54)	n=5 vcb=10 str=2 ps=20 vel=2	30(78)	n=10 vcb=10 str=2 ps=20 vel=2
7(55)	n=5 vcb=10 str=2 ps=40 vel=1	31(79)	n=10 vcb=10 str=2 ps=40 vel=1
8(56)	n=5 vcb=10 str=2 ps=40 vel=2	32(80)	n=10 vcb=10 str=2 ps=40 vel=2
9(57)	n=5 vcb=10 str=3 ps=20 vel=1	33(81)	n=10 vcb=10 str=3 ps=20 vel=1
10(58)	n=5 vcb=10 str=3 ps=20 vel=2	34(82)	n=10 vcb=10 str=3 ps=20 vel=2
11(59)	n=5 vcb=10 str=3 ps=40 vel=1	35(83)	n=10 vcb=10 str=3 ps=40 vel=1
12(60)	n=5 vcb=10 str=3 ps=40 vel=2	36(84)	n=10 vcb=10 str=3 ps=40 vel=2
13(61)	n=5 vcb=20 str=1 ps=20 vel=1	37(85)	n=10 vcb=20 str=1 ps=20 vel=1
14(62)	n=5 vcb=20 str=1 ps=20 vel=2	38(86)	n=10 vcb=20 str=1 ps=20 vel=2
15(63)	n=5 vcb=20 str=1 ps=40 vel=1	39(87)	n=10 vcb=20 str=1 ps=40 vel=1
16(64)	n=5 vcb=20 str=1 ps=40 vel=2	40(88)	n=10 vcb=20 str=1 ps=40 vel=2
17(65)	n=5 vcb=20 str=2 ps=20 vel=1	41(89)	n=10 vcb=20 str=2 ps=20 vel=1
18(66)	n=5 vcb=20 str=2 ps=20 vel=2	42(90)	n=10 vcb=20 str=2 ps=20 vel=2
19(67)	n=5 vcb=20 str=2 ps=40 vel=1	43(91)	n=10 vcb=20 str=2 ps=40 vel=1
20(68)	n=5 vcb=20 str=2 ps=40 vel=2	44(92)	n=10 vcb=20 str=2 ps=40 vel=2
21(69)	n=5 vcb=20 str=3 ps=20 vel=1	45(93)	n=10 vcb=20 str=3 ps=20 vel=1
22(70)	n=5 vcb=20 str=3 ps=20 vel=2	46(94)	n=10 vcb=20 str=3 ps=20 vel=2
23(71)	n=5 vcb=20 str=3 ps=40 vel=1	47(95)	n=10 vcb=20 str=3 ps=40 vel=1
24(72)	n=5 vcb=20 str=3 ps=40 vel=2	48(96)	n=10 vcb=20 str=3 ps=40 vel=2
The first ID number is for pd=0.6, while inside the parenthesis is for pd=0.8			



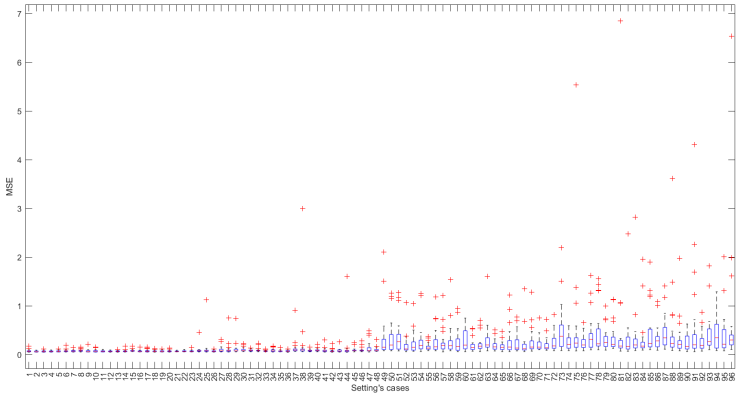
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)

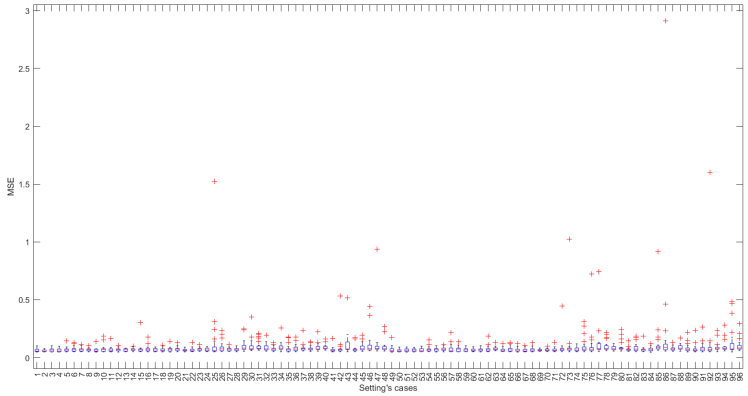


(c) Testing MSE (pd=0.6)

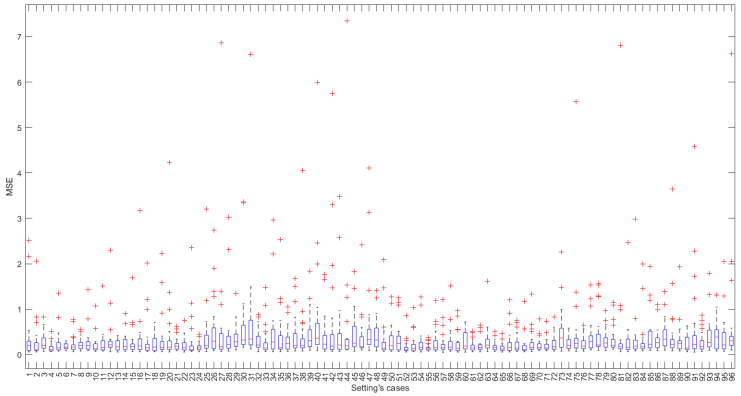


(d) Testing MSE (pd=0.8)

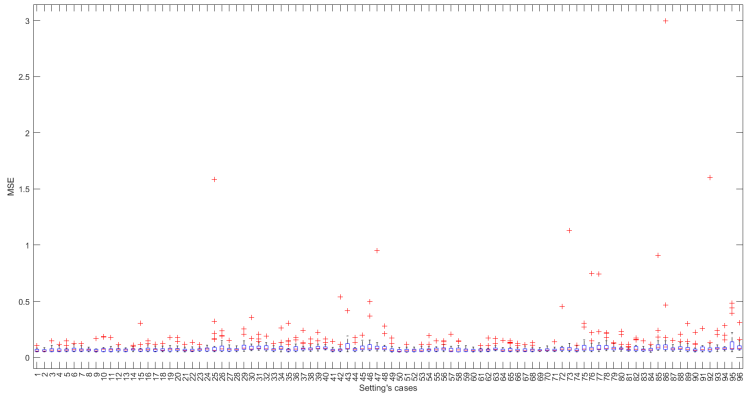
Figure A.1: Boxplots of MSE for setting 1 (pd) of random wa



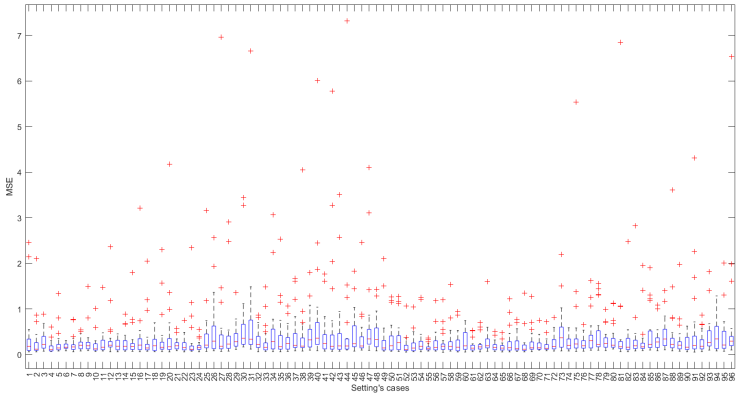
(a) Training MSE (n=5)



(b) Training MSE (n=10)

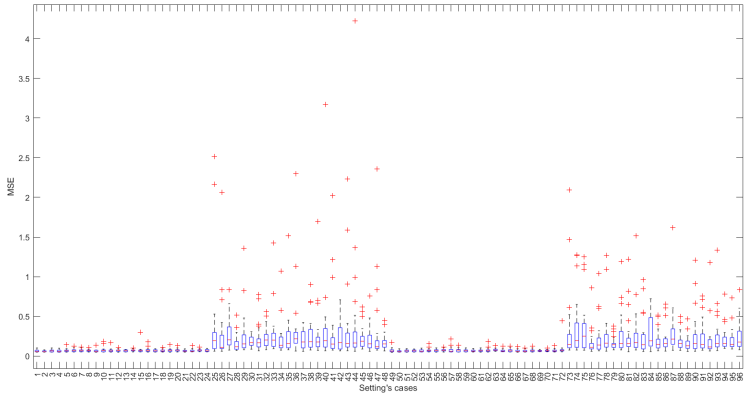


(c) Testing MSE (n=5)

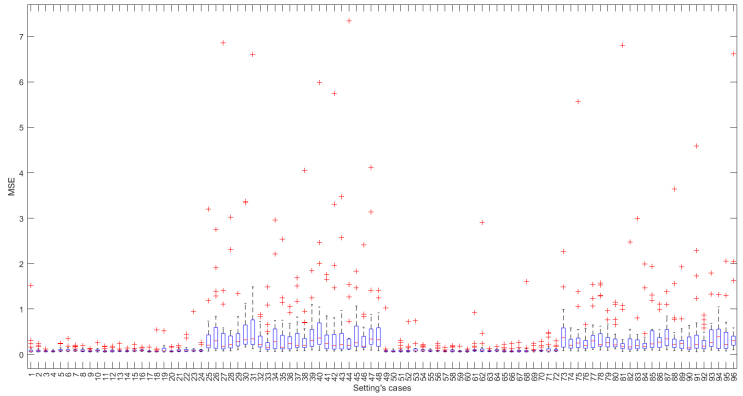


(d) Testing MSE (n=10)

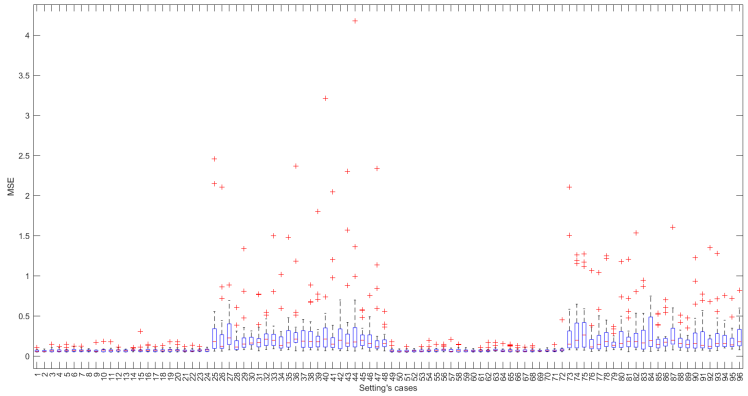
Figure A.2: Boxplots of MSE for setting 2 (n) of random wa.



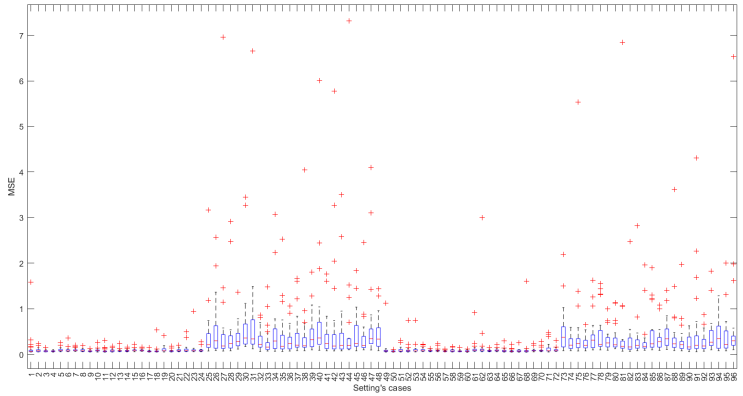
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)

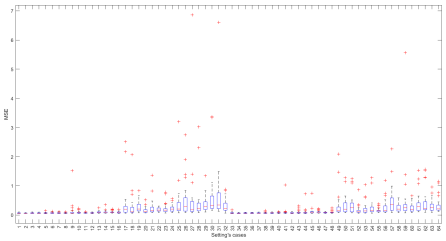


(c) Testing MSE (vcb=10)

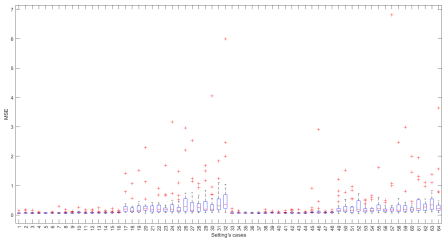


(d) Testing MSE (vcb=20)

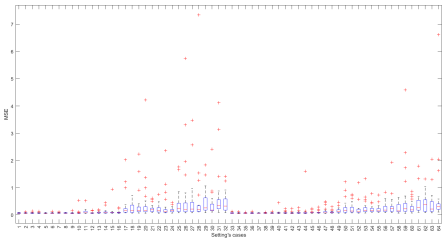
Figure A.3: Boxplots of MSE for setting 3 (*vcb*) of random wa.



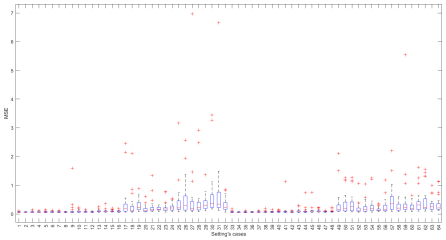
(a) Training MSE (str=1)



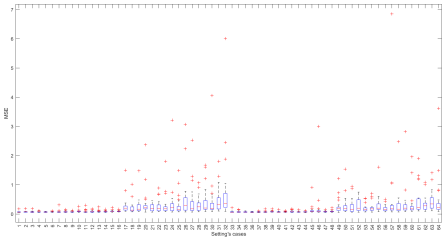
(b) Training MSE (str=2)



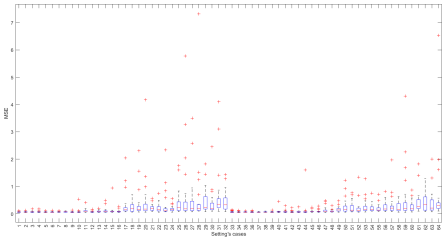
(c) Training MSE (str=3)



(d) Testing MSE (str=1)

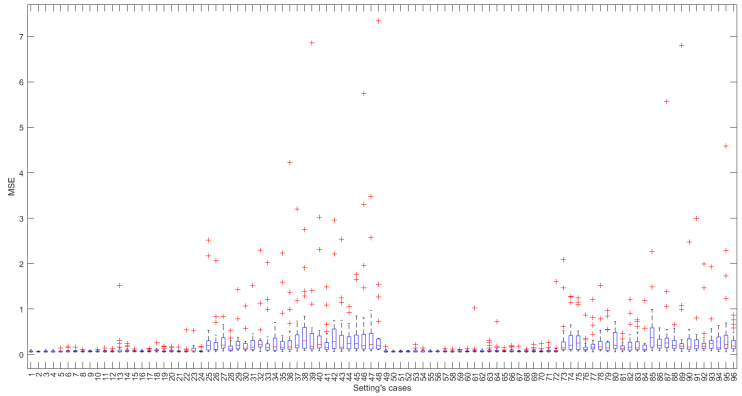


(e) Testing MSE (str=2)

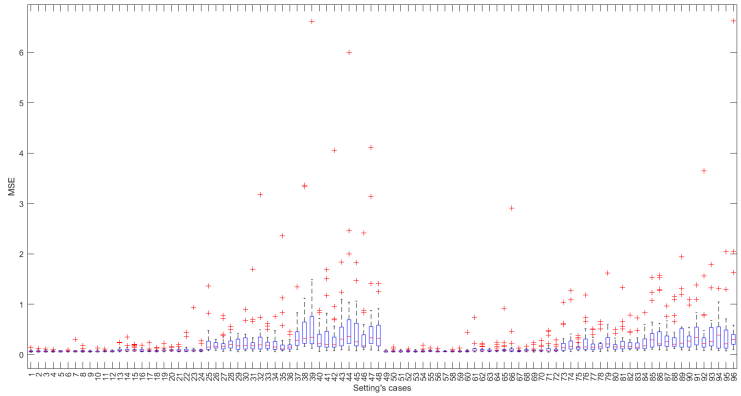


(f) Testing MSE (str=3)

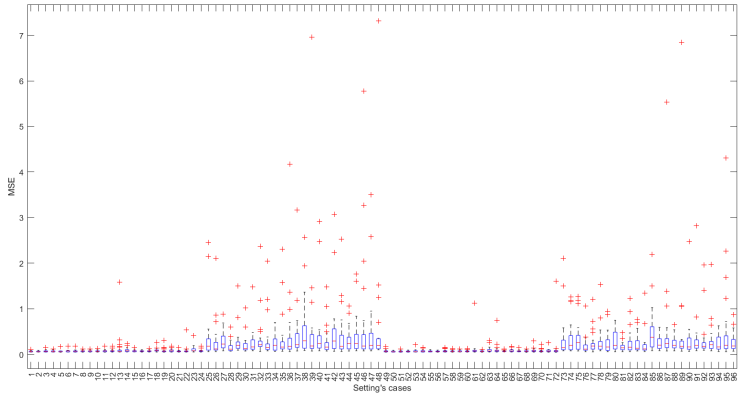
Figure A.4: Boxplots of MSE for setting 4 (*str*) of random wa.



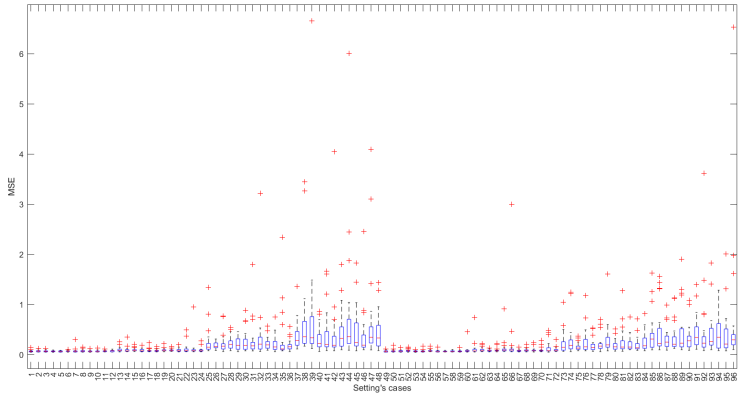
(a) Training MSE (ps=20)



(b) Training MSE (ps=40)

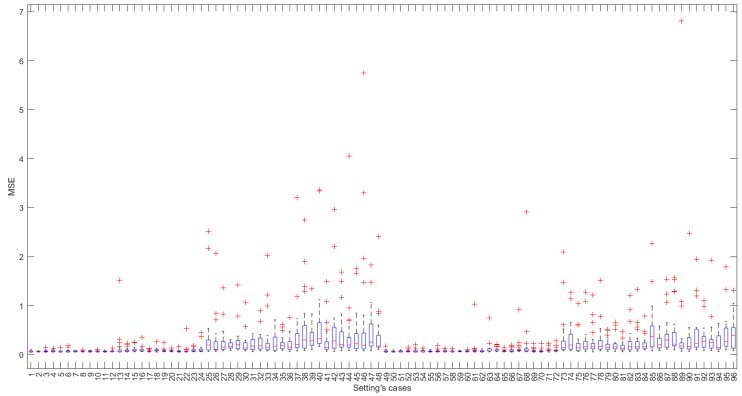


(c) Testing MSE (ps=20)

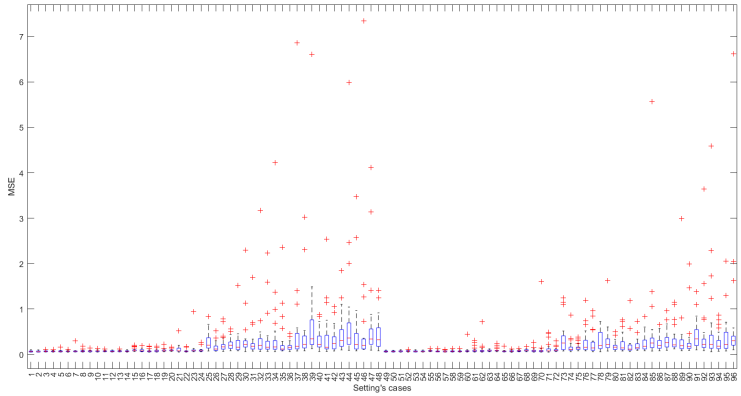


(d) Testing MSE (ps=40)

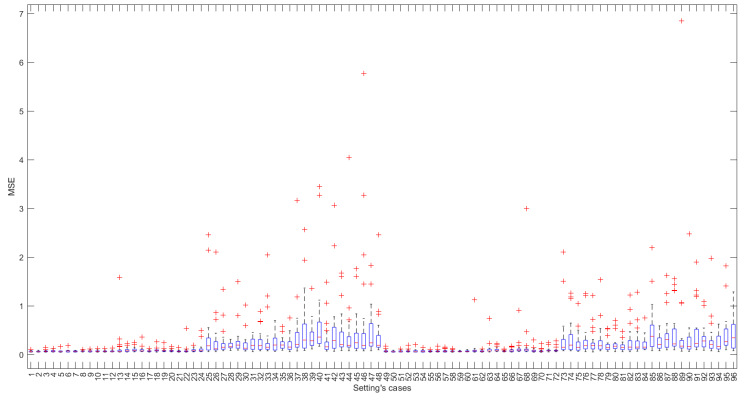
Figure A.5: Boxplots of MSE for setting 5 (ps) of random wa.



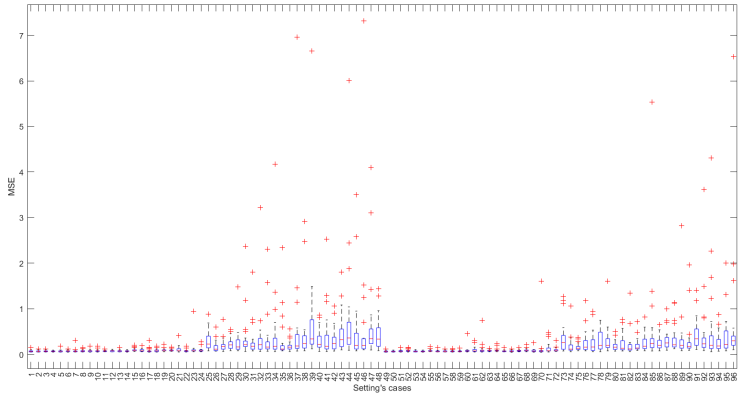
(a) Training MSE (vel=1)



(b) Training MSE (vel=2)

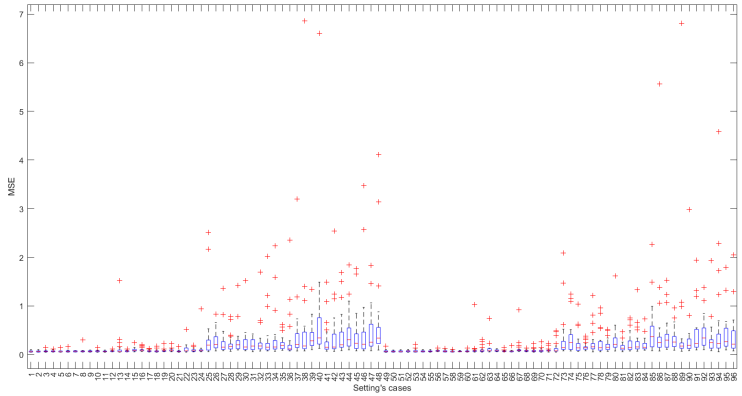


(c) Testing MSE (vel=1)

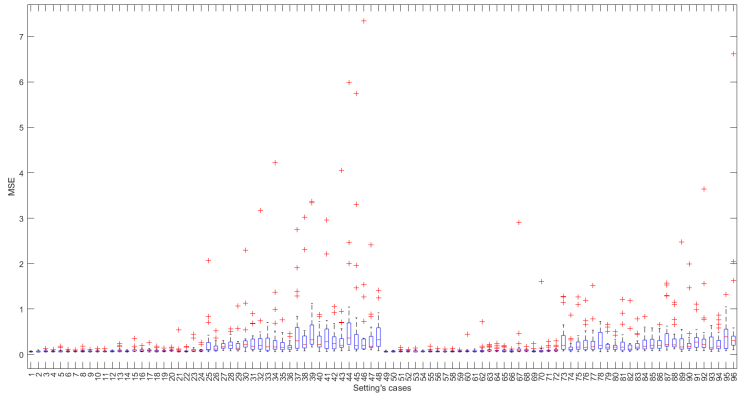


(d) Testing MSE (vel=2)

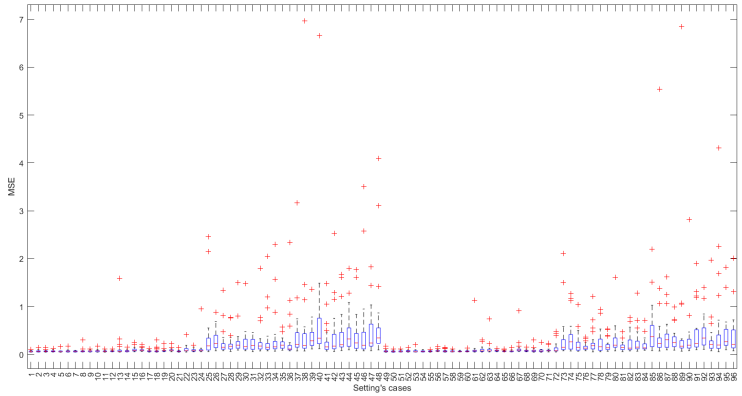
Figure A.6: Boxplots of MSE for setting 6 (vel) of random wa.



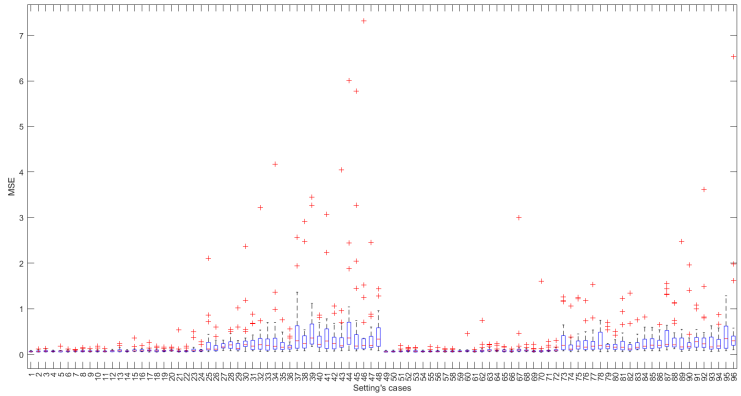
(a) Training MSE (rest=0)



(b) Training MSE (rest=1)

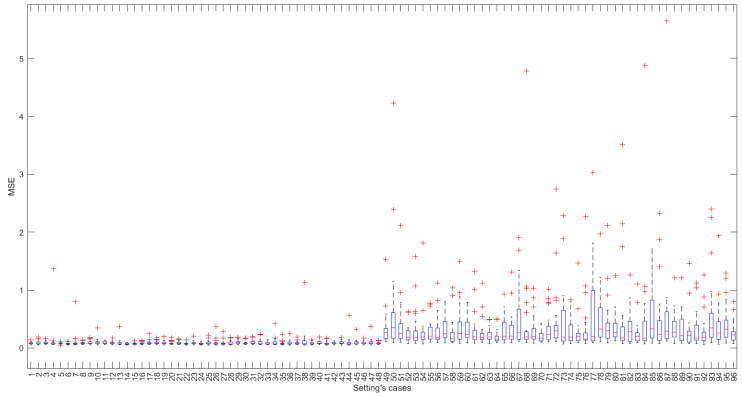


(c) Testing MSE (rest=0)

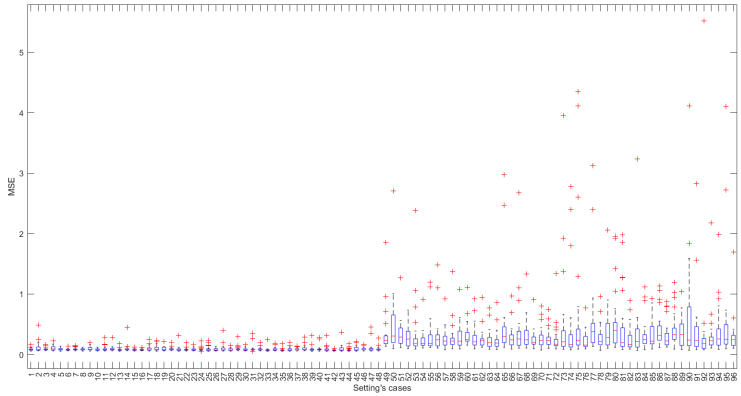


(d) Testing MSE (rest=1)

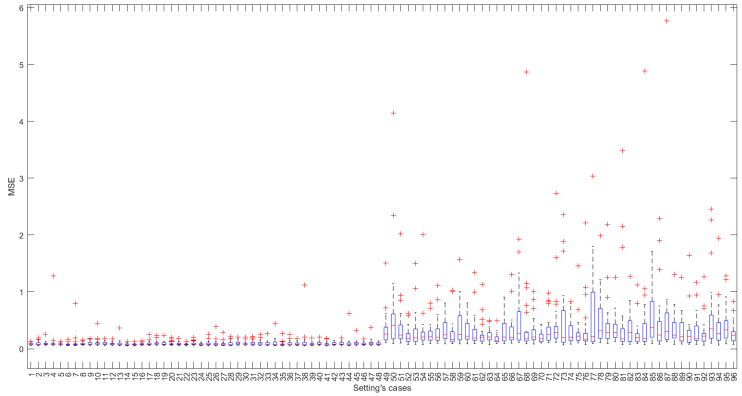
Figure A.7: Boxplots of MSE for setting 7 ($rest$) of random wa.



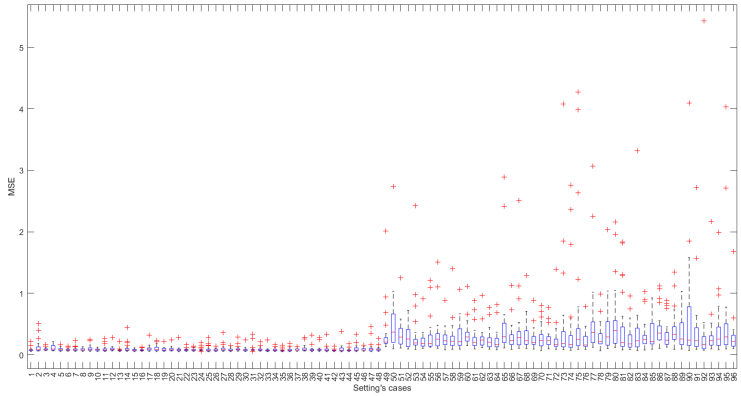
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)

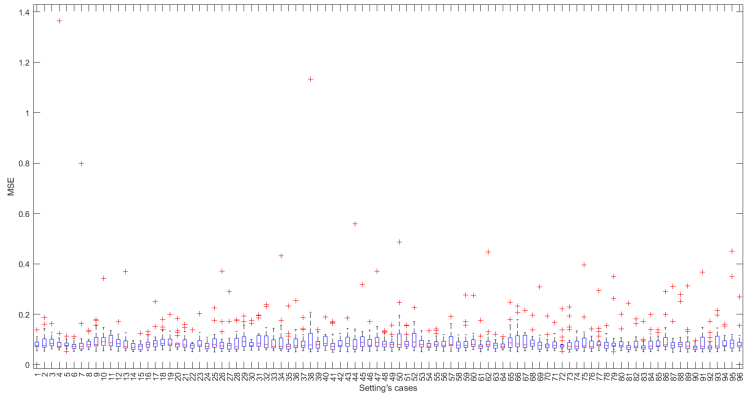


(c) Testing MSE (pd=0.6)

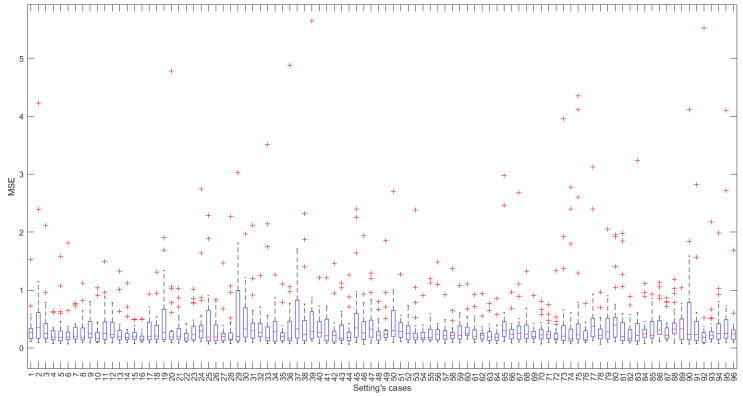


(d) Testing MSE (pd=0.8)

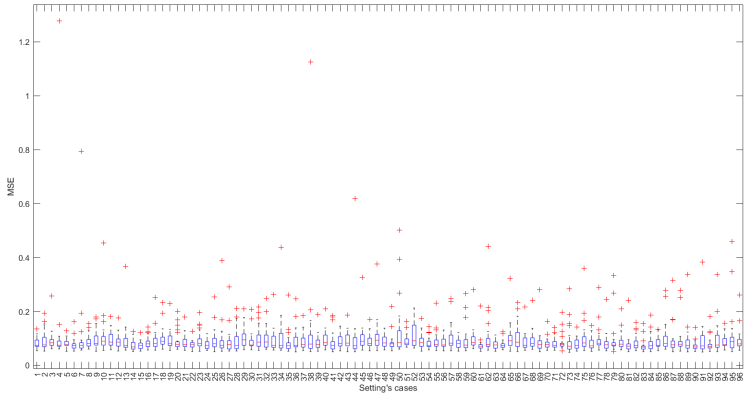
Figure A.8: Boxplots of MSE for setting 1 (pd) of bang-bang wa.



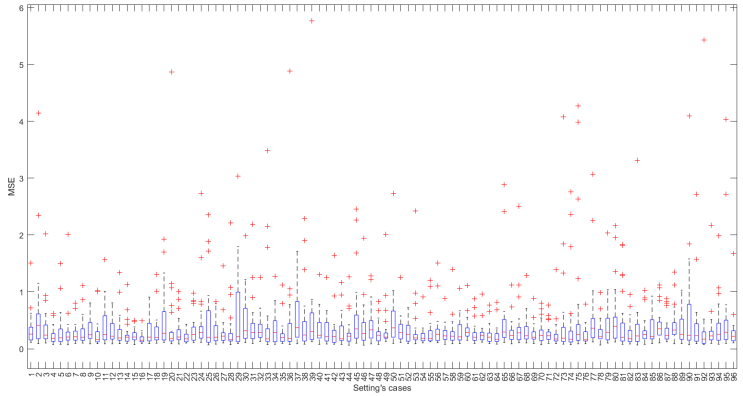
(a) Training MSE (n=5)



(b) Training MSE (n=10)

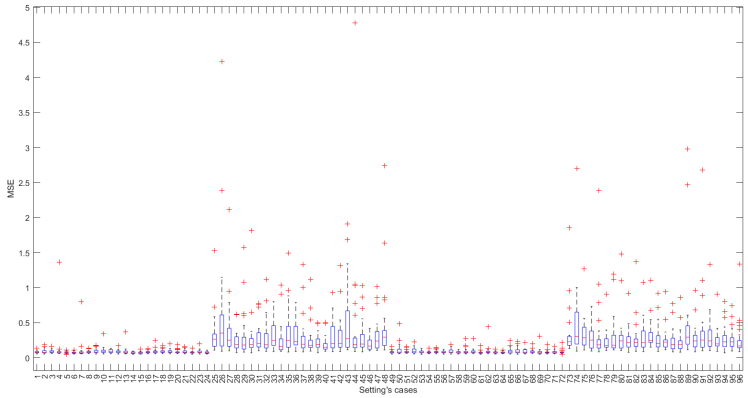


(c) Testing MSE (n=5)

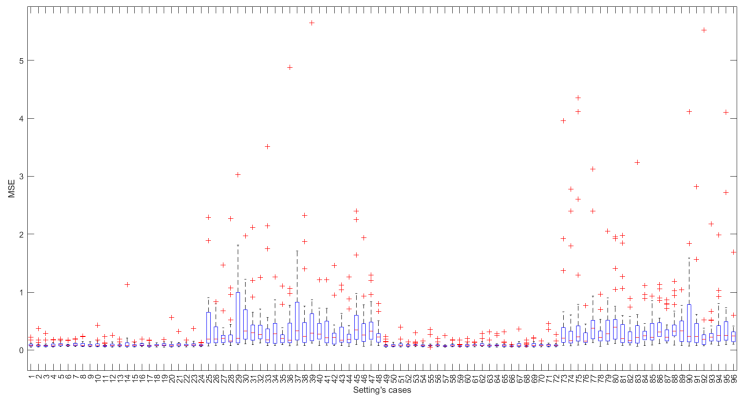


(d) Testing MSE (n=10)

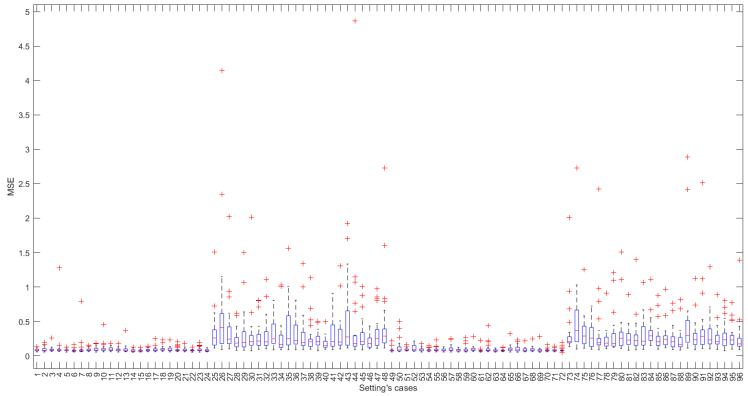
Figure A.9: Boxplots of MSE for setting 2 (n) of bang-bang wa.



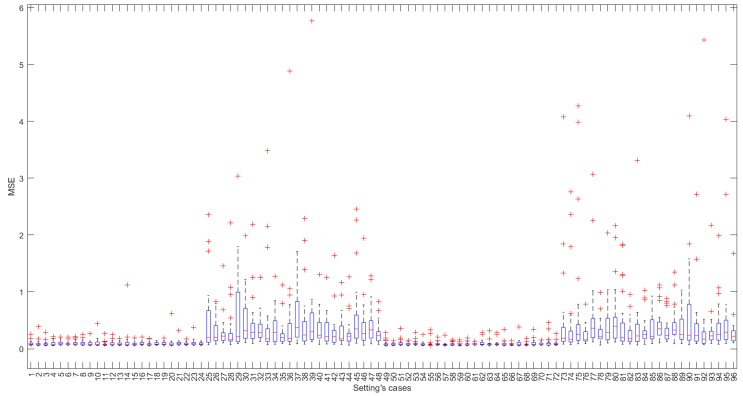
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)

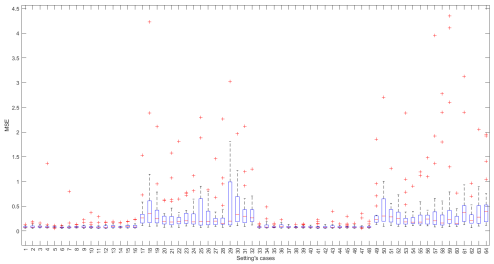


(c) Testing MSE (vcb=10)

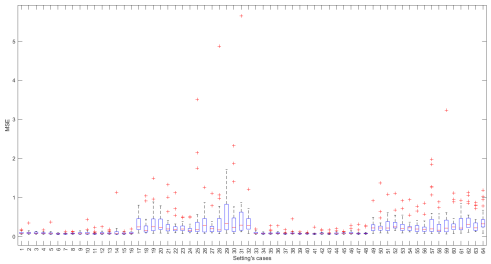


(d) Testing MSE (vcb=20)

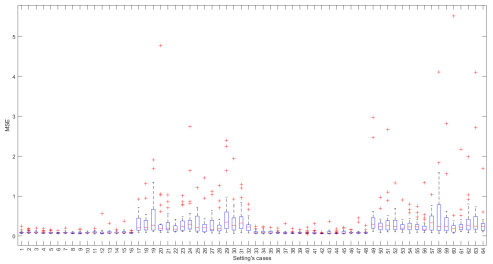
Figure A.10: Boxplots of MSE for setting 3 (vcb) of bang-bang wa.



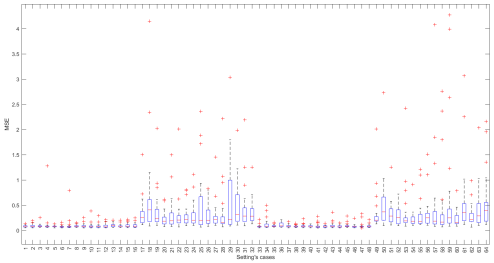
(a) Training MSE (str=1)



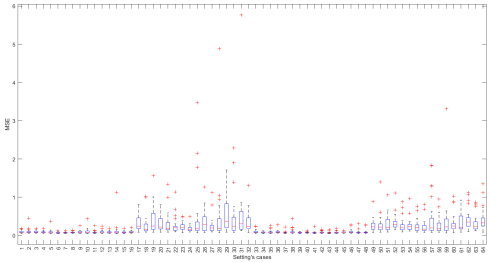
(b) Training MSE (str=2)



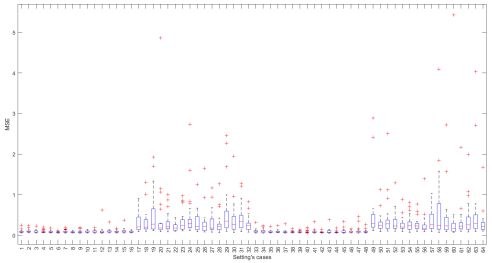
(c) Training MSE (str=3)



(d) Testing MSE (str=1)

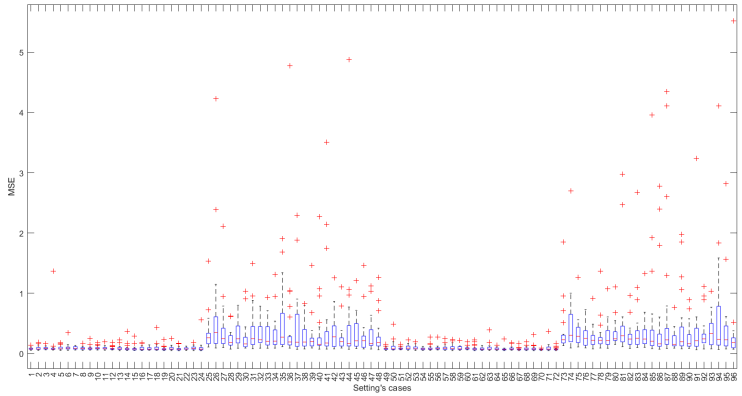


(e) Testing MSE (str=2)

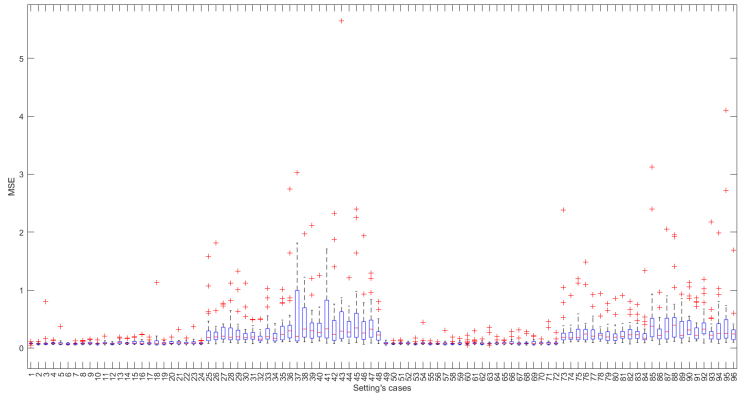


(f) Testing MSE (str=3)

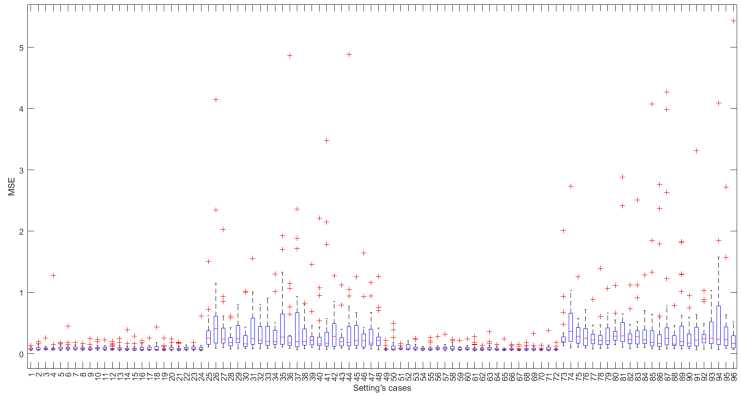
Figure A.11: Boxplots of MSE for setting 4 (str) of bang-bang wa.



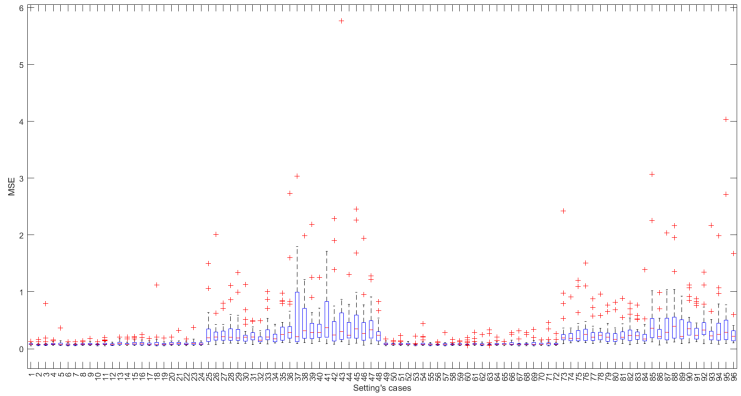
(a) Training MSE (ps=20)



(b) Training MSE (ps=40)

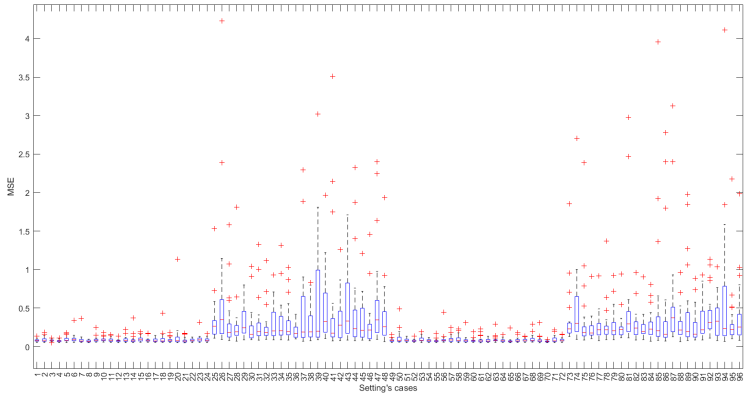


(c) Testing MSE (ps=20)

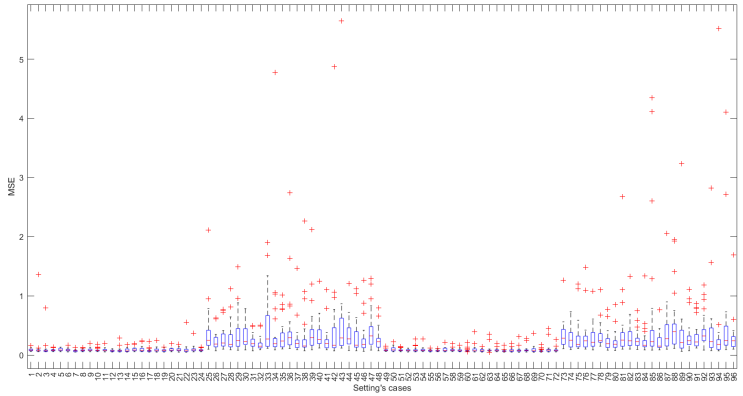


(d) Testing MSE (ps=40)

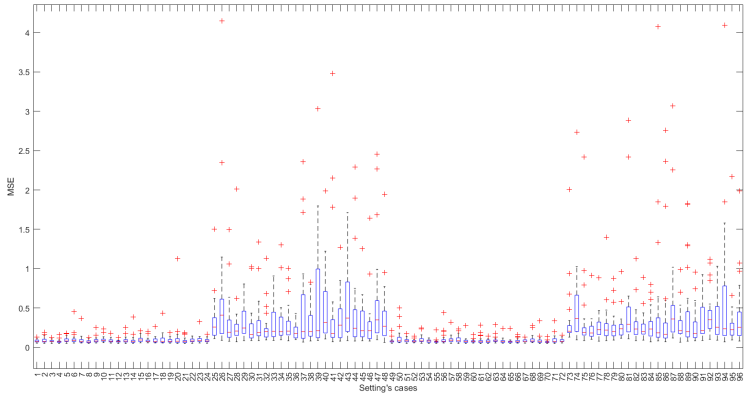
Figure A.12: Boxplots of MSE for setting 5 (ps) of bang-bang wa.



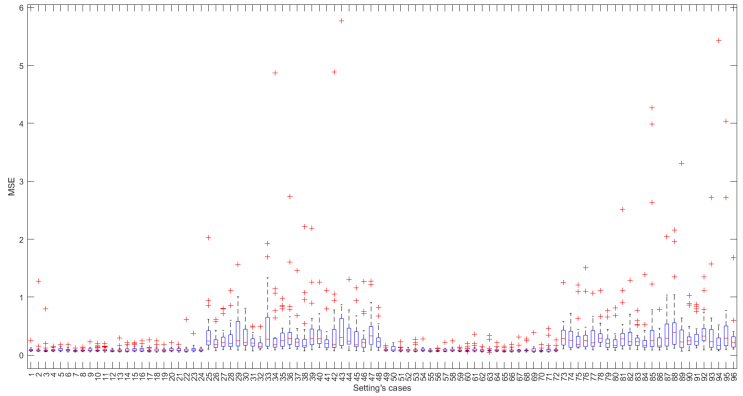
(a) Training MSE (vel=1)



(b) Training MSE (vel=2)

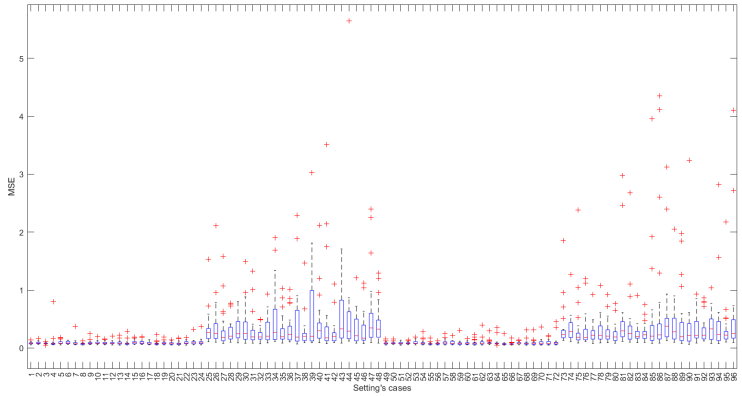


(c) Testing MSE (vel=1)

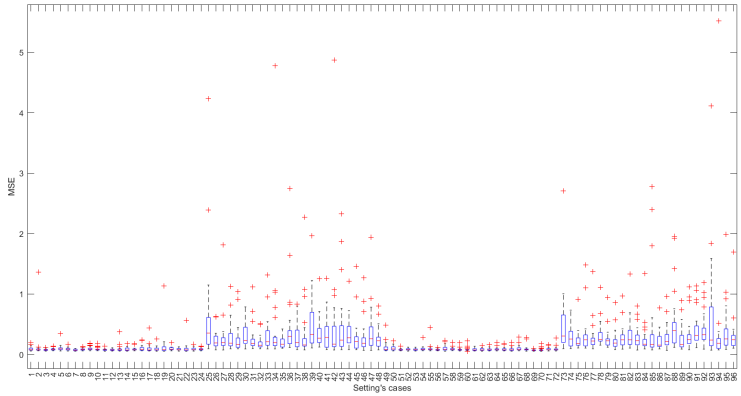


(d) Testing MSE (vel=2)

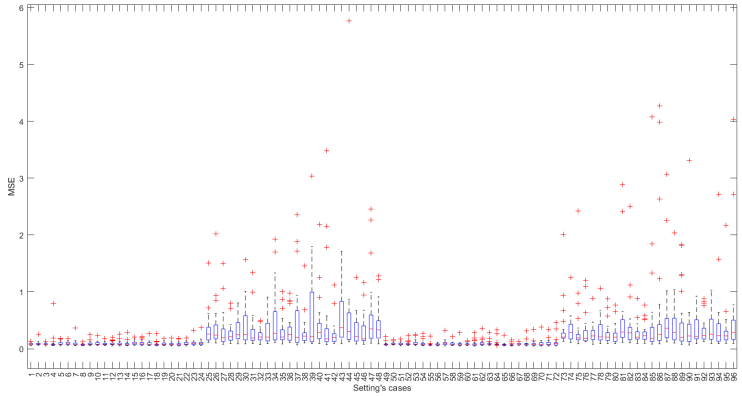
Figure A.13: Boxplots of MSE for setting 6 (vel) of bang-bang wa.



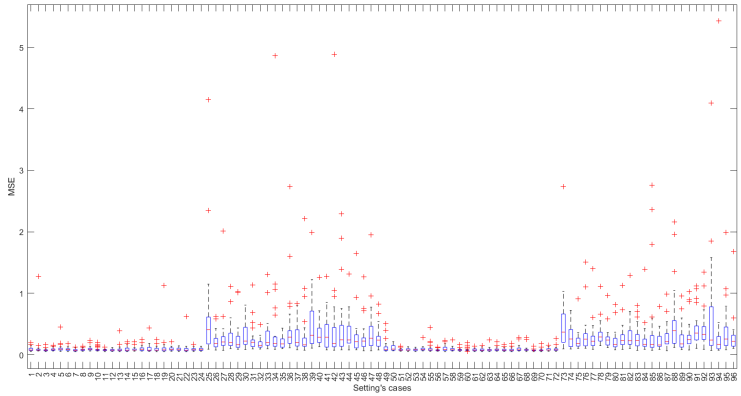
(a) Training MSE (rest=0)



(b) Training MSE (rest=1)

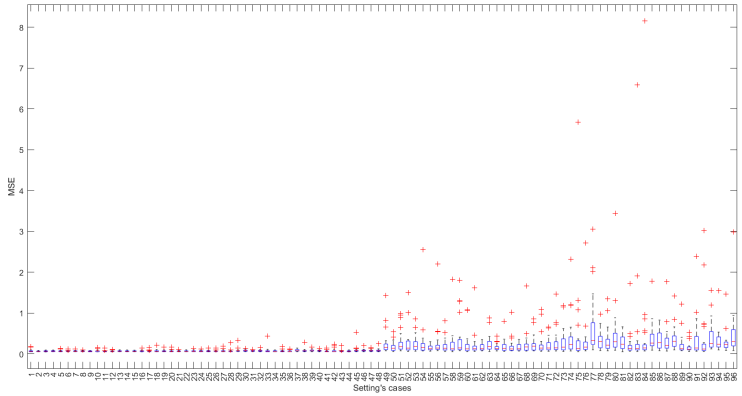


(c) Testing MSE (rest=0)

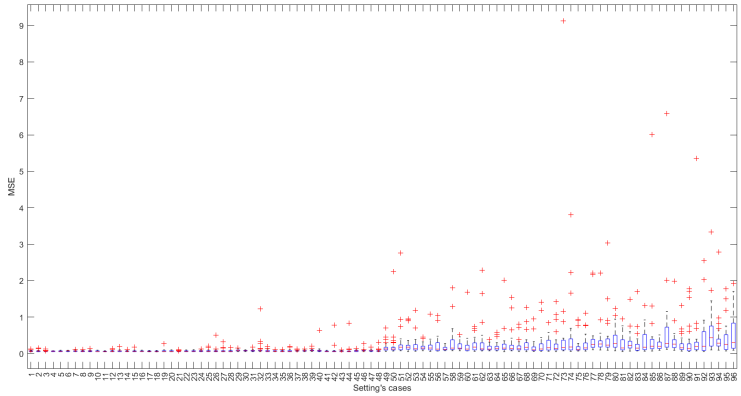


(d) Testing MSE (rest=1)

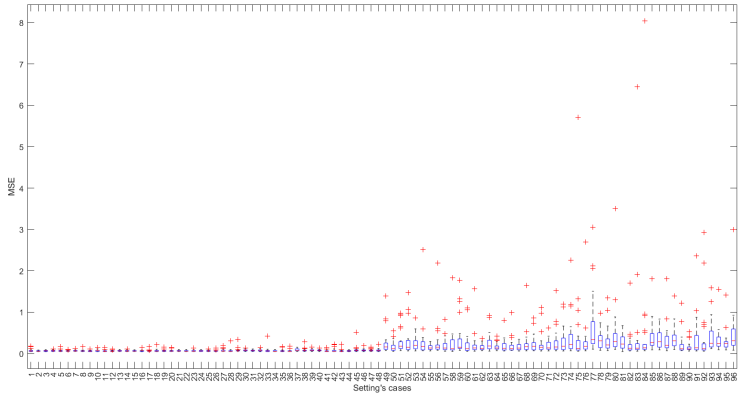
Figure A.14: Boxplots of MSE for setting 7 ($rest$) of bang-bang wa.



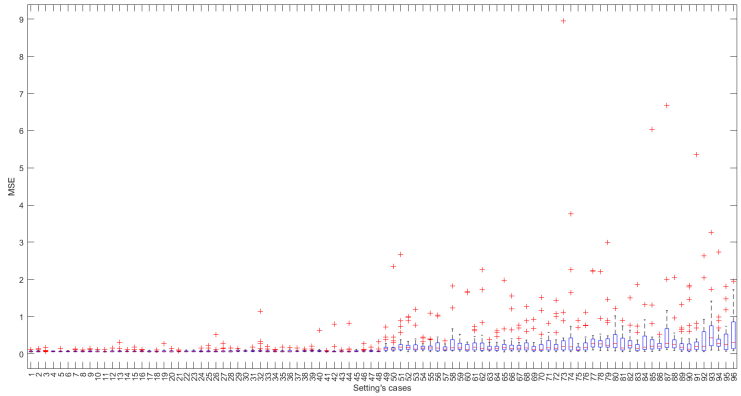
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)

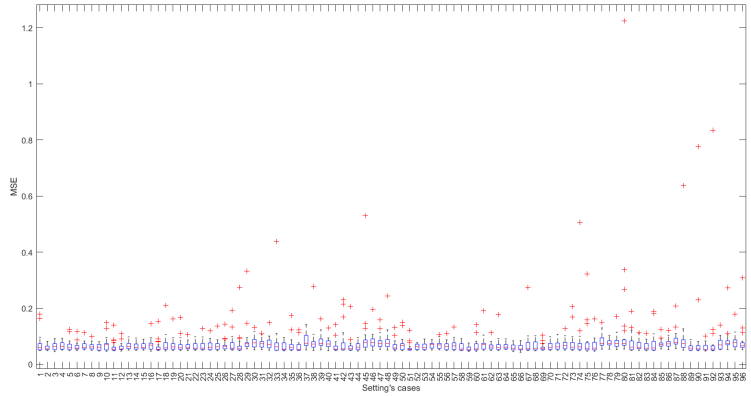


(c) Testing MSE (pd=0.6)

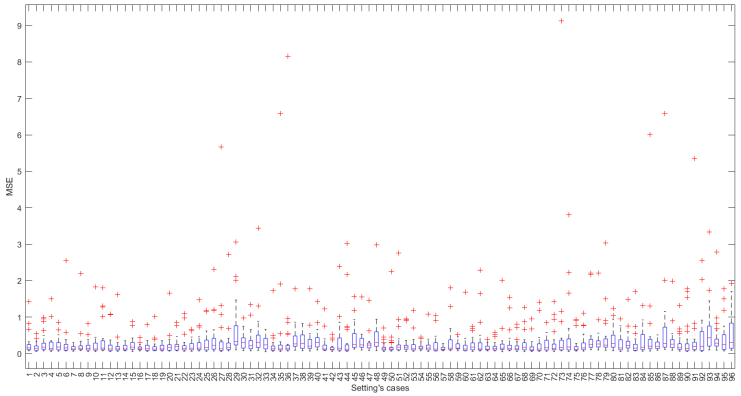


(d) Testing MSE (pd=0.8)

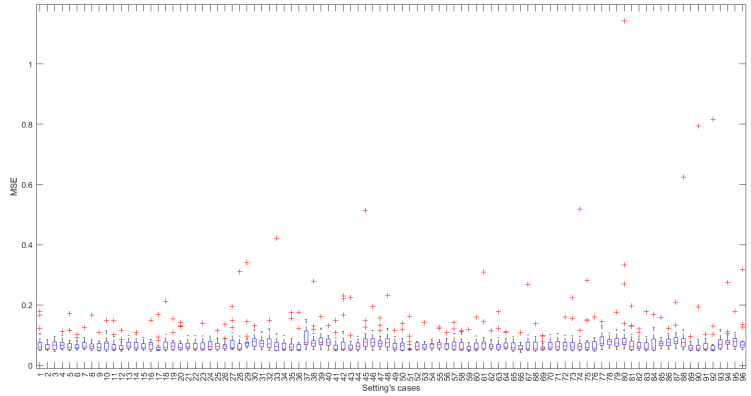
Figure A.15: Boxplots of MSE for setting 1 (pd) of dynamic wa.



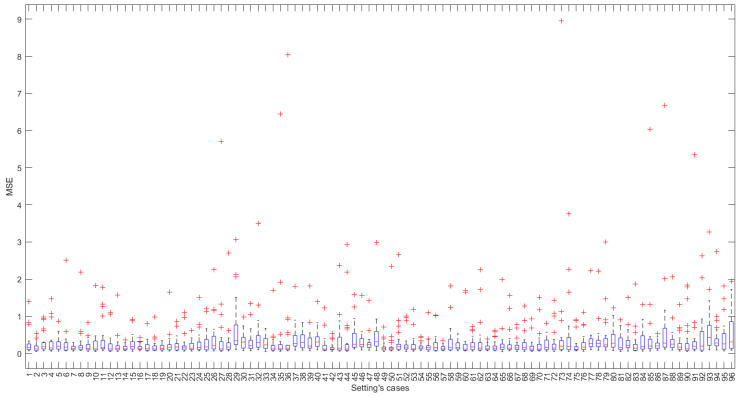
(a) Training MSE (n=5)



(b) Training MSE (n=10)

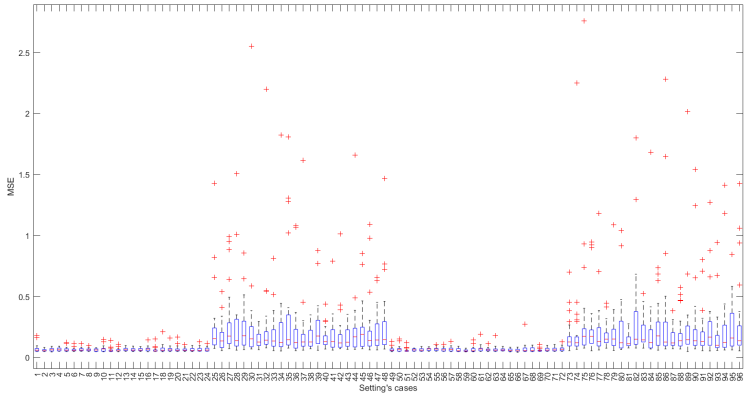


(c) Testing MSE (n=5)

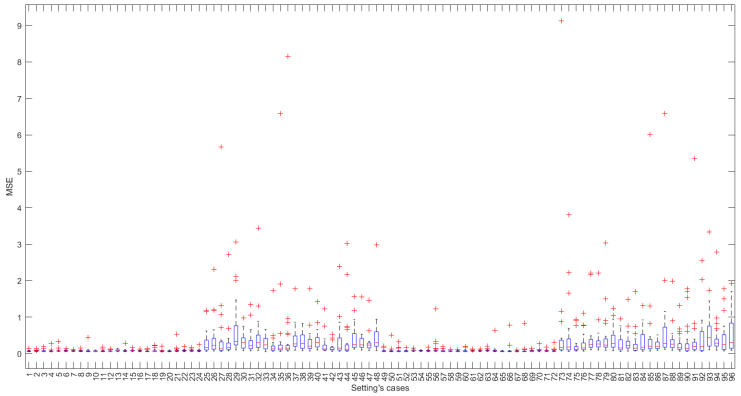


(d) Testing MSE (n=10)

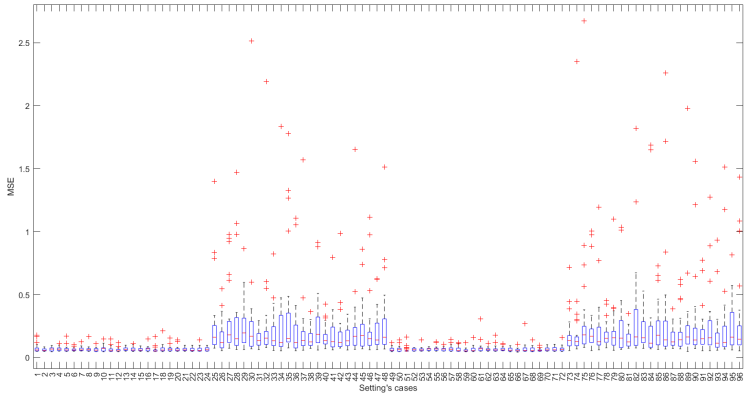
Figure A.16: Boxplots of MSE for setting 2 (n) of dynamic wa.



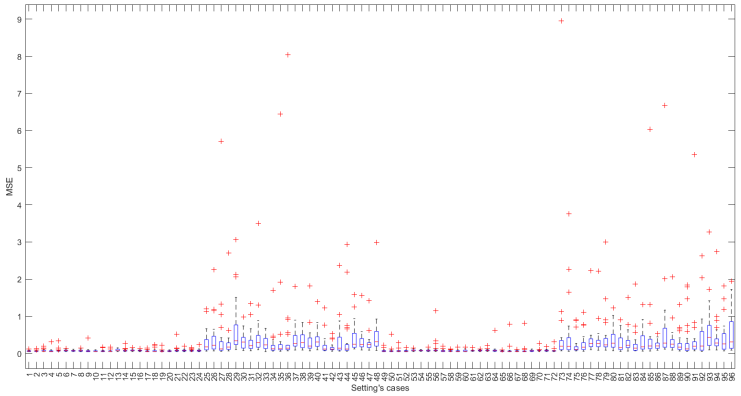
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)

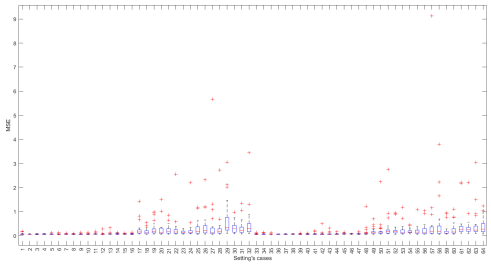


(c) Testing MSE (vcb=10)

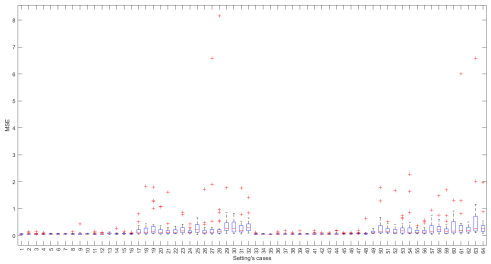


(d) Testing MSE (vcb=20)

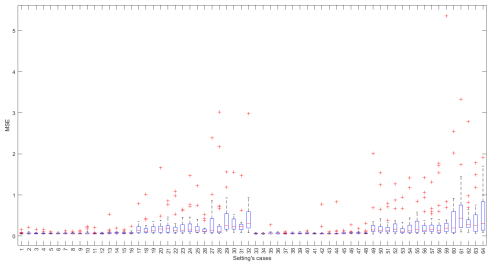
Figure A.17: Boxplots of MSE for setting 3 (vcb) of dynamic wa.



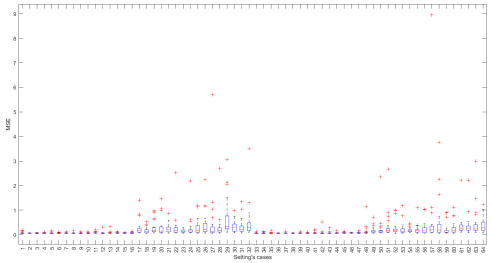
(a) Training MSE (str=1)



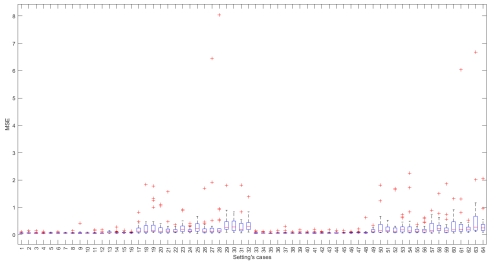
(b) Training MSE (str=2)



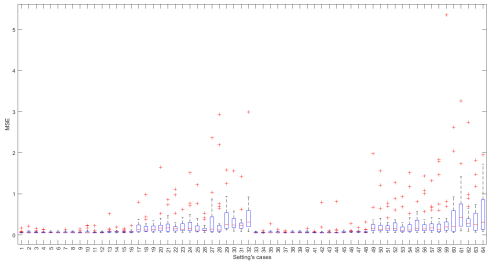
(c) Training MSE (str=3)



(d) Testing MSE (str=1)

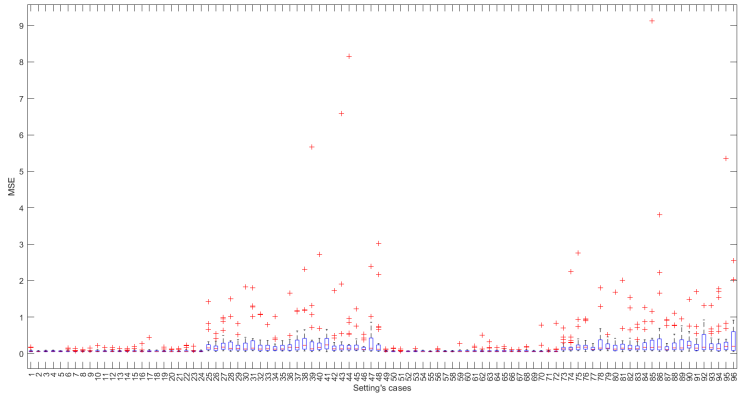


(e) Testing MSE (str=2)

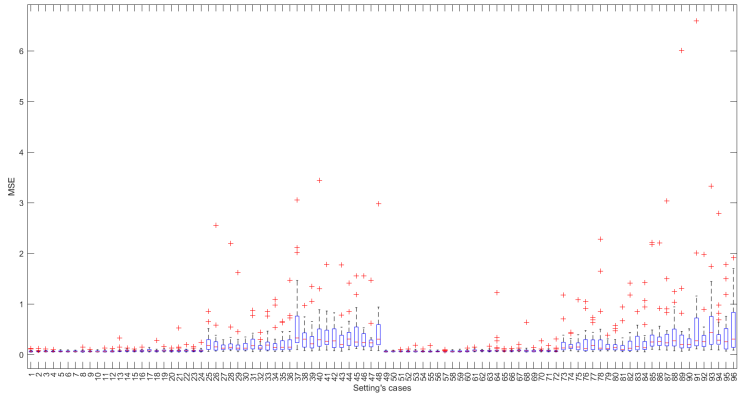


(f) Testing MSE (str=3)

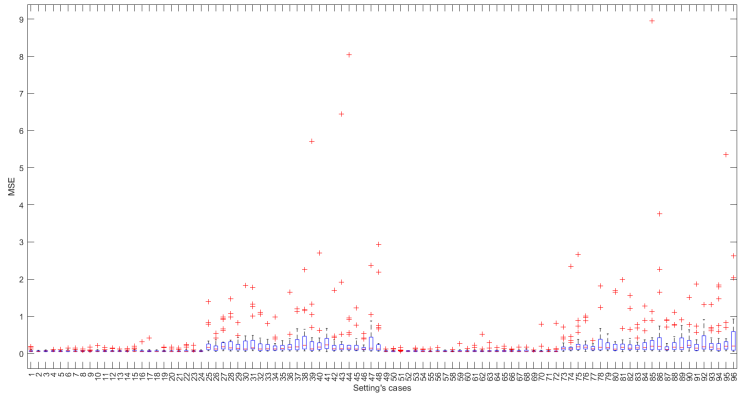
Figure A.18: Boxplots of MSE for setting 4 (str) of dynamic wa.



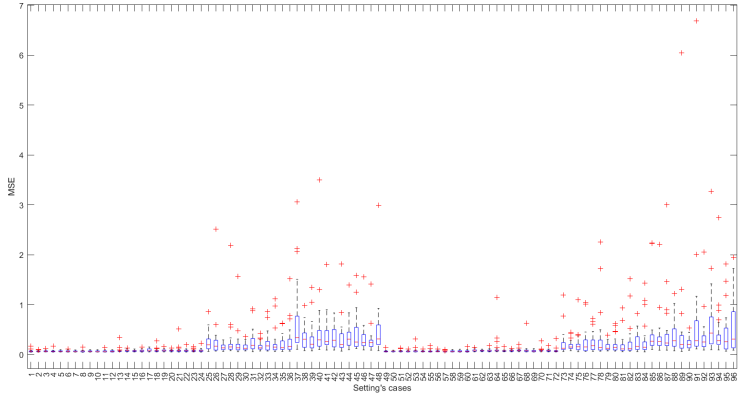
(a) Training MSE (ps=20)



(b) Training MSE (ps=40)

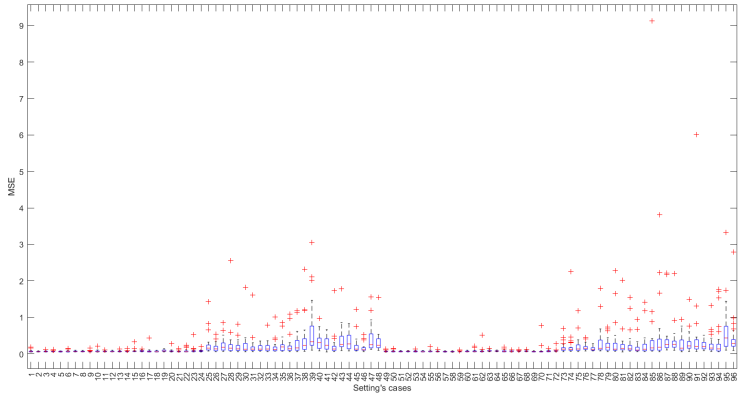


(c) Testing MSE (ps=20)

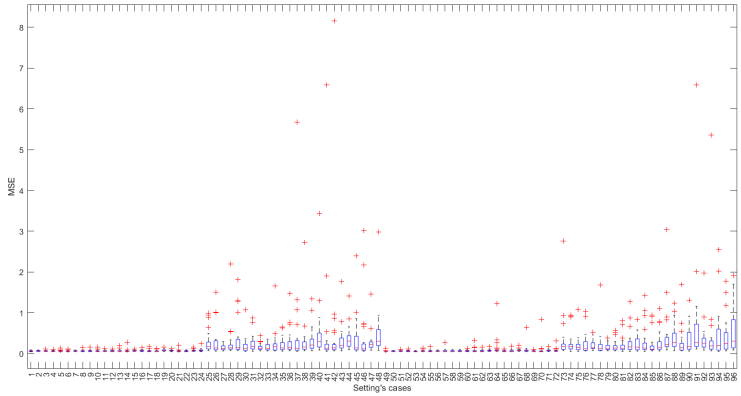


(d) Testing MSE (ps=40)

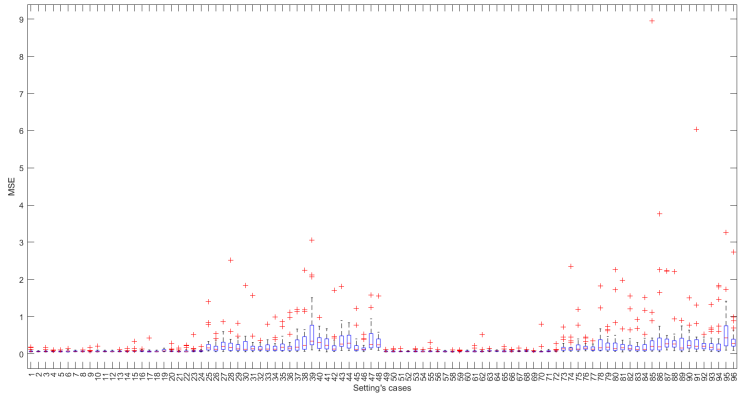
Figure A.19: Boxplots of MSE for setting 5 (ps) of dynamic wa.



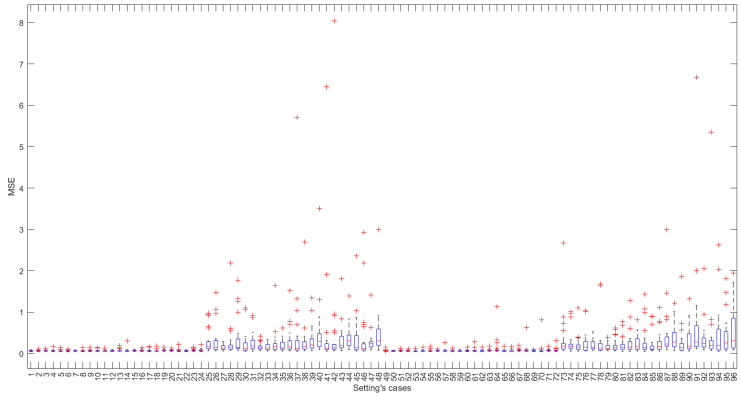
(a) Training MSE (vel=1)



(b) Training MSE (vel=2)

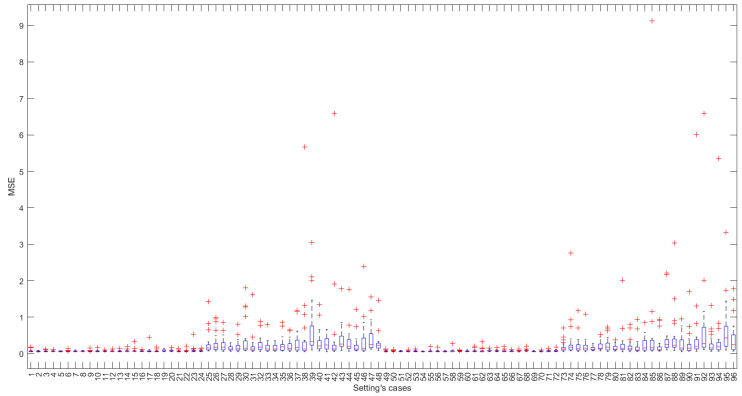


(c) Testing MSE (vel=1)

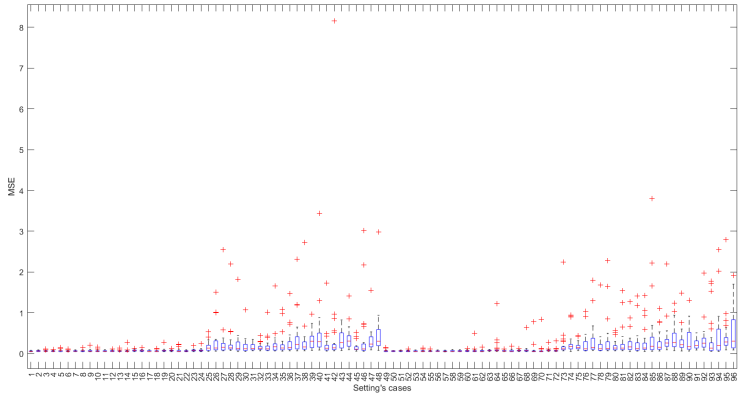


(d) Testing MSE (vel=2)

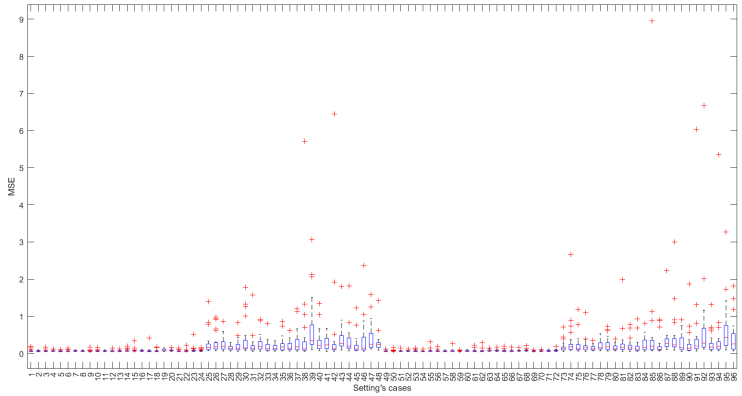
Figure A.20: Boxplots of MSE for setting 6 (vel) of dynamic wa.



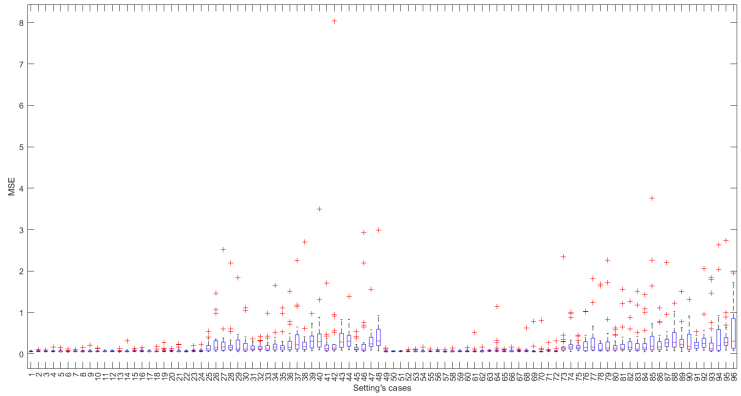
(a) Training MSE (rest=0)



(b) Training MSE (rest=1)



(c) Testing MSE (rest=0)



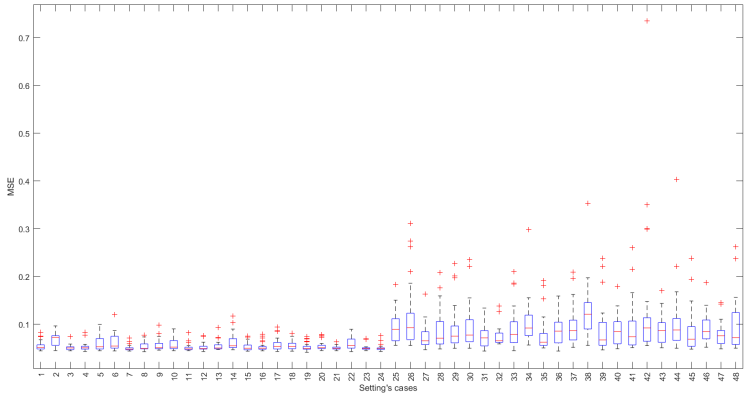
(d) Testing MSE (rest=1)

Figure A.21: Boxplots of MSE for setting 7 ($rest$) of dynamic wa.

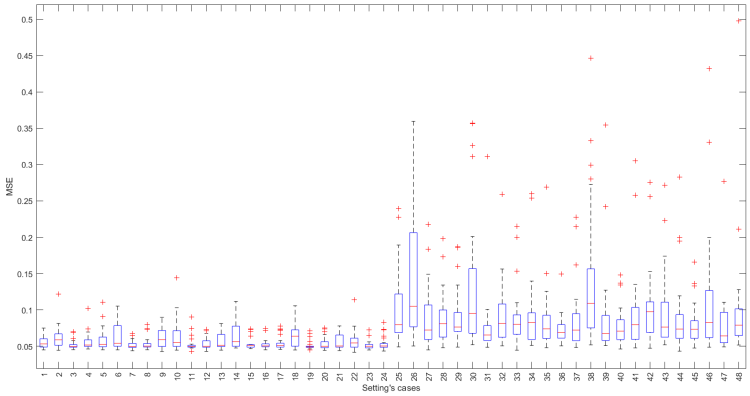
A.2 Examined cases for the non-sorting genetic algorithm

Table A.8: Examined cases for setting 1 (pd) of NSGA-II.

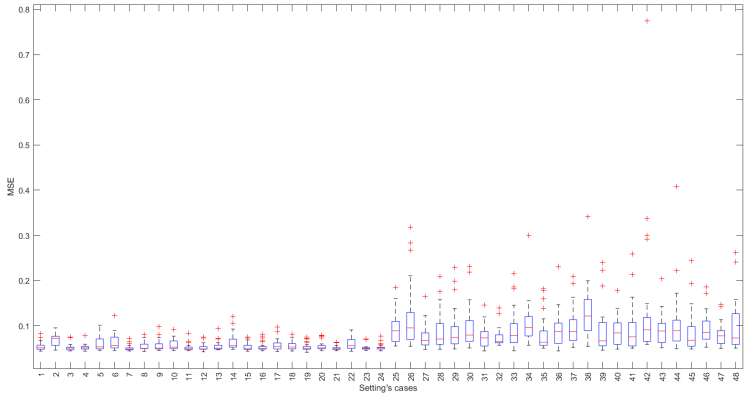
ID	Examined cases	ID	Examined cases
1	n=5 vcb=10 str=1 ps=20 mut=1	25	n=10 vcb=10 str=1 ps=20 mut=1
2	n=5 vcb=10 str=1 ps=20 mut=2	26	n=10 vcb=10 str=1 ps=20 mut=2
3	n=5 vcb=10 str=1 ps=40 mut=1	27	n=10 vcb=10 str=1 ps=40 mut=1
4	n=5 vcb=10 str=1 ps=40 mut=2	28	n=10 vcb=10 str=1 ps=40 mut=2
5	n=5 vcb=10 str=2 ps=20 mut=1	29	n=10 vcb=10 str=2 ps=20 mut=1
6	n=5 vcb=10 str=2 ps=20 mut=2	30	n=10 vcb=10 str=2 ps=20 mut=2
7	n=5 vcb=10 str=2 ps=40 mut=1	31	n=10 vcb=10 str=2 ps=40 mut=1
8	n=5 vcb=10 str=2 ps=40 mut=2	32	n=10 vcb=10 str=2 ps=40 mut=2
9	n=5 vcb=10 str=3 ps=20 mut=1	33	n=10 vcb=10 str=3 ps=20 mut=1
10	n=5 vcb=10 str=3 ps=20 mut=2	34	n=10 vcb=10 str=3 ps=20 mut=2
11	n=5 vcb=10 str=3 ps=40 mut=1	35	n=10 vcb=10 str=3 ps=40 mut=1
12	n=5 vcb=10 str=3 ps=40 mut=2	36	n=10 vcb=10 str=3 ps=40 mut=2
13	n=5 vcb=20 str=1 ps=20 mut=1	37	n=10 vcb=20 str=1 ps=20 mut=1
14	n=5 vcb=20 str=1 ps=20 mut=2	38	n=10 vcb=20 str=1 ps=20 mut=2
15	n=5 vcb=20 str=1 ps=40 mut=1	39	n=10 vcb=20 str=1 ps=40 mut=1
16	n=5 vcb=20 str=1 ps=40 mut=2	40	n=10 vcb=20 str=1 ps=40 mut=2
17	n=5 vcb=20 str=2 ps=20 mut=1	41	n=10 vcb=20 str=2 ps=20 mut=1
18	n=5 vcb=20 str=2 ps=20 mut=2	42	n=10 vcb=20 str=2 ps=20 mut=2
19	n=5 vcb=20 str=2 ps=40 mut=1	43	n=10 vcb=20 str=2 ps=40 mut=1
20	n=5 vcb=20 str=2 ps=40 mut=2	44	n=10 vcb=20 str=2 ps=40 mut=2
21	n=5 vcb=20 str=3 ps=20 mut=1	45	n=10 vcb=20 str=3 ps=20 mut=1
22	n=5 vcb=20 str=3 ps=20 mut=2	46	n=10 vcb=20 str=3 ps=20 mut=2
23	n=5 vcb=20 str=3 ps=40 mut=1	47	n=10 vcb=20 str=3 ps=40 mut=1
24	n=5 vcb=20 str=3 ps=40 mut=2	48	n=10 vcb=20 str=3 ps=40 mut=2



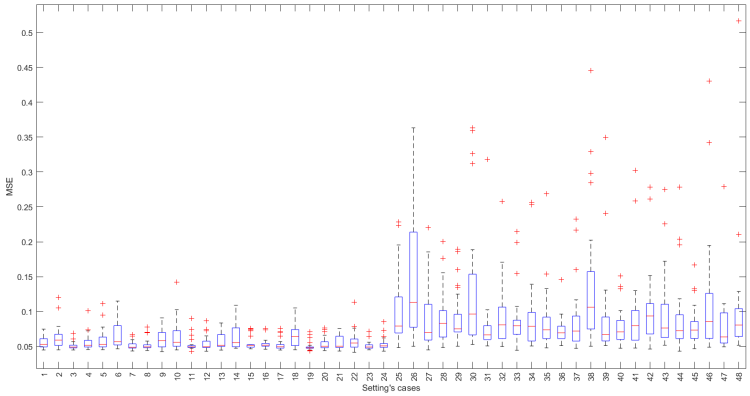
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)



(c) Testing MSE (pd=0.6)

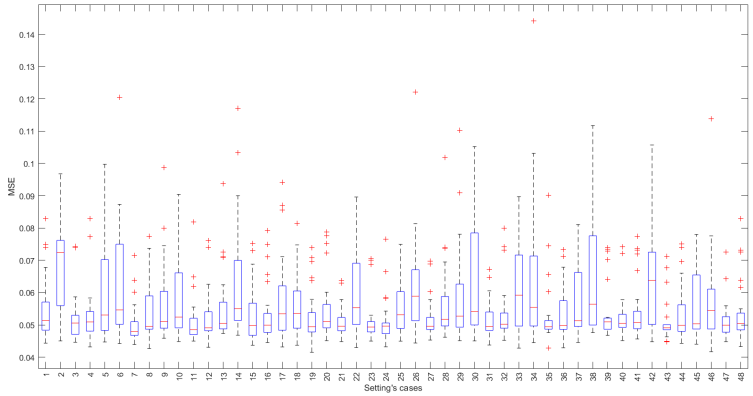


(d) Testing MSE (pd=0.8)

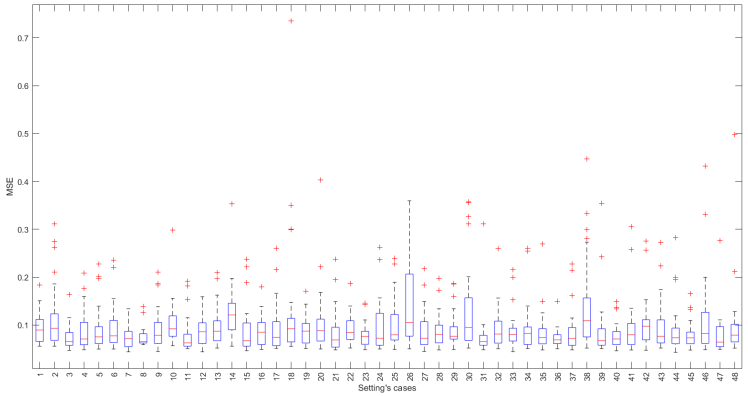
Figure A.22: Boxplots of MSE for setting 1 (pd) of NSGA-II.

Table A.9: Examined cases for setting 2 (n) of NSGA-II.

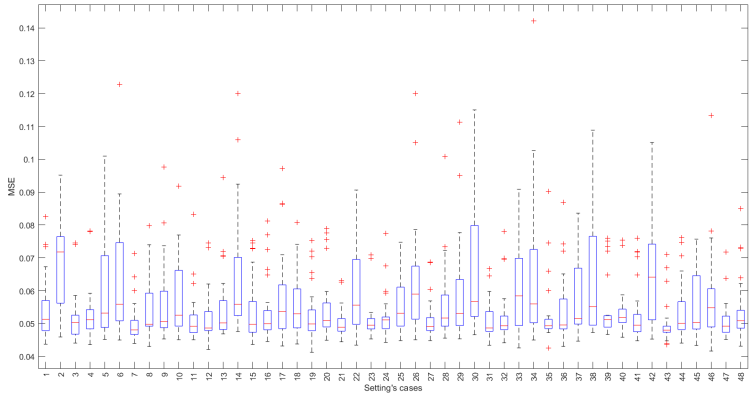
ID	Examined cases	ID	Examined cases
1	pd=0.6 vcb=10 str=1 ps=20 mut=1	25	pd=0.8 vcb=10 str=1 ps=20 mut=1
2	pd=0.6 vcb=10 str=1 ps=20 mut=2	26	pd=0.8 vcb=10 str=1 ps=20 mut=2
3	pd=0.6 vcb=10 str=1 ps=40 mut=1	27	pd=0.8 vcb=10 str=1 ps=40 mut=1
4	pd=0.6 vcb=10 str=1 ps=40 mut=2	28	pd=0.8 vcb=10 str=1 ps=40 mut=2
5	pd=0.6 vcb=10 str=2 ps=20 mut=1	29	pd=0.8 vcb=10 str=2 ps=20 mut=1
6	pd=0.6 vcb=10 str=2 ps=20 mut=2	30	pd=0.8 vcb=10 str=2 ps=20 mut=2
7	pd=0.6 vcb=10 str=2 ps=40 mut=1	31	pd=0.8 vcb=10 str=2 ps=40 mut=1
8	pd=0.6 vcb=10 str=2 ps=40 mut=2	32	pd=0.8 vcb=10 str=2 ps=40 mut=2
9	pd=0.6 vcb=10 str=3 ps=20 mut=1	33	pd=0.8 vcb=10 str=3 ps=20 mut=1
10	pd=0.6 vcb=10 str=3 ps=20 mut=2	34	pd=0.8 vcb=10 str=3 ps=20 mut=2
11	pd=0.6 vcb=10 str=3 ps=40 mut=1	35	pd=0.8 vcb=10 str=3 ps=40 mut=1
12	pd=0.6 vcb=10 str=3 ps=40 mut=2	36	pd=0.8 vcb=10 str=3 ps=40 mut=2
13	pd=0.6 vcb=20 str=1 ps=20 mut=1	37	pd=0.8 vcb=20 str=1 ps=20 mut=1
14	pd=0.6 vcb=20 str=1 ps=20 mut=2	38	pd=0.8 vcb=20 str=1 ps=20 mut=2
15	pd=0.6 vcb=20 str=1 ps=40 mut=1	39	pd=0.8 vcb=20 str=1 ps=40 mut=1
16	pd=0.6 vcb=20 str=1 ps=40 mut=2	40	pd=0.8 vcb=20 str=1 ps=40 mut=2
17	pd=0.6 vcb=20 str=2 ps=20 mut=1	41	pd=0.8 vcb=20 str=2 ps=20 mut=1
18	pd=0.6 vcb=20 str=2 ps=20 mut=2	42	pd=0.8 vcb=20 str=2 ps=20 mut=2
19	pd=0.6 vcb=20 str=2 ps=40 mut=1	43	pd=0.8 vcb=20 str=2 ps=40 mut=1
20	pd=0.6 vcb=20 str=2 ps=40 mut=2	44	pd=0.8 vcb=20 str=2 ps=40 mut=2
21	pd=0.6 vcb=20 str=3 ps=20 mut=1	45	pd=0.8 vcb=20 str=3 ps=20 mut=1
22	pd=0.6 vcb=20 str=3 ps=20 mut=2	46	pd=0.8 vcb=20 str=3 ps=20 mut=2
23	pd=0.6 vcb=20 str=3 ps=40 mut=1	47	pd=0.8 vcb=20 str=3 ps=40 mut=1
24	pd=0.6 vcb=20 str=3 ps=40 mut=2	48	pd=0.8 vcb=20 str=3 ps=40 mut=2



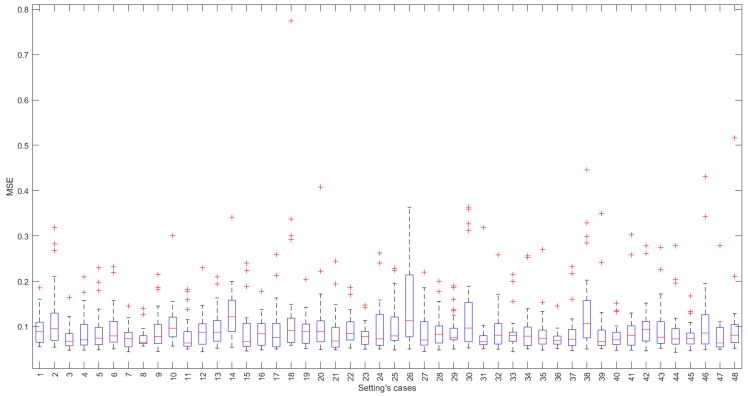
(a) Training MSE (n=5)



(b) Training MSE (n=10)



(c) Testing MSE (n=5)

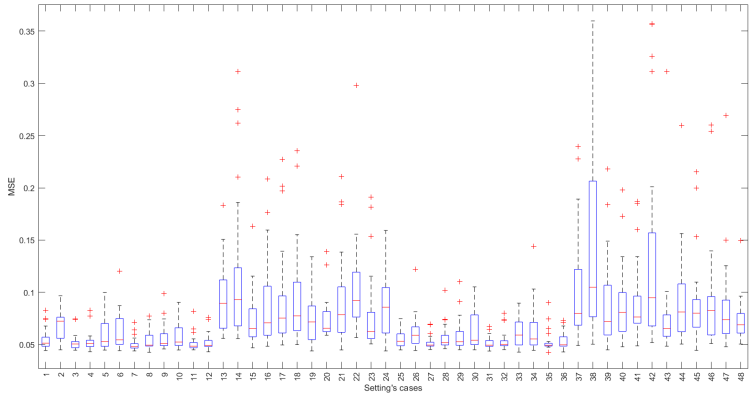


(d) Testing MSE (n=10)

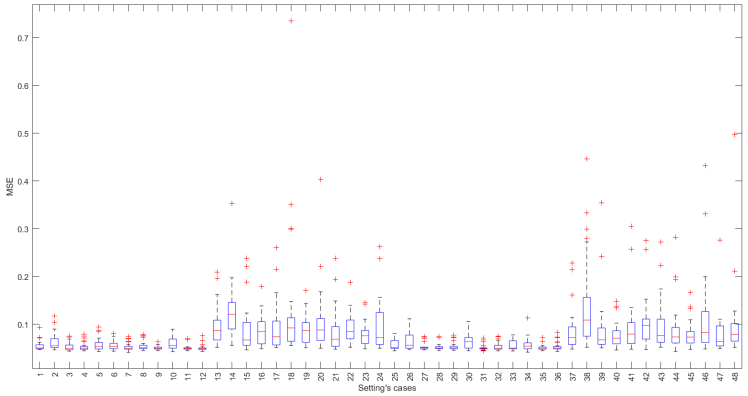
Figure A.23: Boxplots of MSE for setting 2 (n) of NSGA-II.

Table A.10: Examined cases for setting 3 (vcb) of NSGA-II.

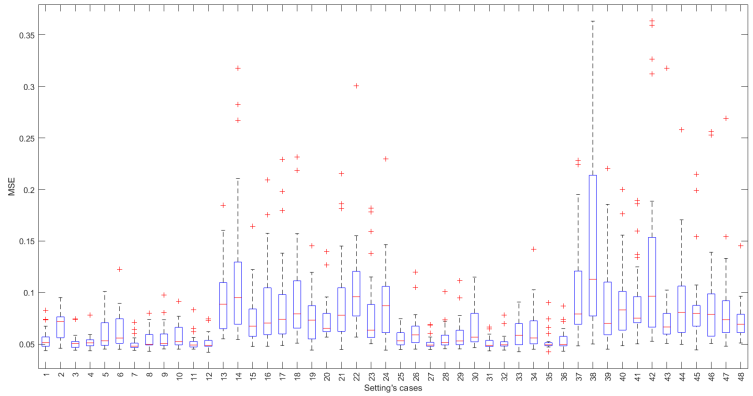
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 str=1 ps=20 mut=1	25	pd=0.8 n=5 str=1 ps=20 mut=1
2	pd=0.6 n=5 str=1 ps=20 mut=2	26	pd=0.8 n=5 str=1 ps=20 mut=2
3	pd=0.6 n=5 str=1 ps=40 mut=1	27	pd=0.8 n=5 str=1 ps=40 mut=1
4	pd=0.6 n=5 str=1 ps=40 mut=2	28	pd=0.8 n=5 str=1 ps=40 mut=2
5	pd=0.6 n=5 str=2 ps=20 mut=1	29	pd=0.8 n=5 str=2 ps=20 mut=1
6	pd=0.6 n=5 str=2 ps=20 mut=2	30	pd=0.8 n=5 str=2 ps=20 mut=2
7	pd=0.6 n=5 str=2 ps=40 mut=1	31	pd=0.8 n=5 str=2 ps=40 mut=1
8	pd=0.6 n=5 str=2 ps=40 mut=2	32	pd=0.8 n=5 str=2 ps=40 mut=2
9	pd=0.6 n=5 str=3 ps=20 mut=1	33	pd=0.8 n=5 str=3 ps=20 mut=1
10	pd=0.6 n=5 str=3 ps=20 mut=2	34	pd=0.8 n=5 str=3 ps=20 mut=2
11	pd=0.6 n=5 str=3 ps=40 mut=1	35	pd=0.8 n=5 str=3 ps=40 mut=1
12	pd=0.6 n=5 str=3 ps=40 mut=2	36	pd=0.8 n=5 str=3 ps=40 mut=2
13	pd=0.6 n=10 str=1 ps=20 mut=1	37	pd=0.8 n=10 str=1 ps=20 mut=1
14	pd=0.6 n=10 str=1 ps=20 mut=2	38	pd=0.8 n=10 str=1 ps=20 mut=2
15	pd=0.6 n=10 str=1 ps=40 mut=1	39	pd=0.8 n=10 str=1 ps=40 mut=1
16	pd=0.6 n=10 str=1 ps=40 mut=2	40	pd=0.8 n=10 str=1 ps=40 mut=2
17	pd=0.6 n=10 str=2 ps=20 mut=1	41	pd=0.8 n=10 str=2 ps=20 mut=1
18	pd=0.6 n=10 str=2 ps=20 mut=2	42	pd=0.8 n=10 str=2 ps=20 mut=2
19	pd=0.6 n=10 str=2 ps=40 mut=1	43	pd=0.8 n=10 str=2 ps=40 mut=1
20	pd=0.6 n=10 str=2 ps=40 mut=2	44	pd=0.8 n=10 str=2 ps=40 mut=2
21	pd=0.6 n=10 str=3 ps=20 mut=1	45	pd=0.8 n=10 str=3 ps=20 mut=1
22	pd=0.6 n=10 str=3 ps=20 mut=2	46	pd=0.8 n=10 str=3 ps=20 mut=2
23	pd=0.6 n=10 str=3 ps=40 mut=1	47	pd=0.8 n=10 str=3 ps=40 mut=1
24	pd=0.6 n=10 str=3 ps=40 mut=2	48	pd=0.8 n=10 str=3 ps=40 mut=2



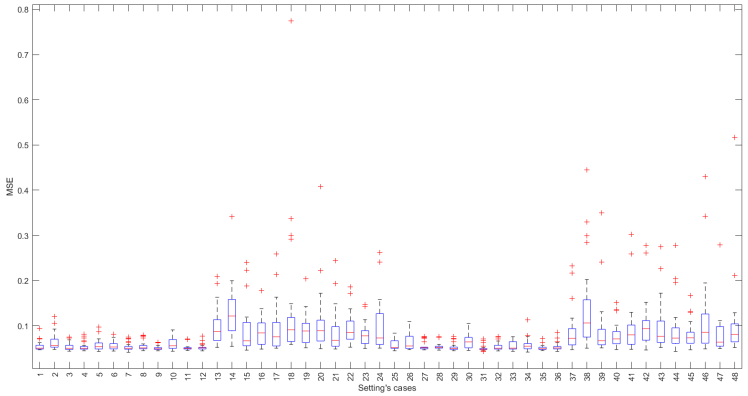
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)



(c) Testing MSE (vcb=10)

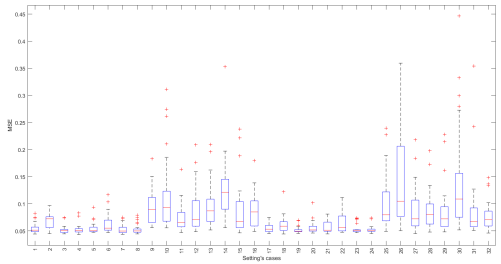


(d) Testing MSE (vcb=20)

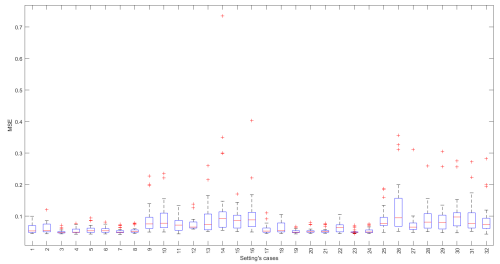
Figure A.24: Boxplots of MSE for setting 3 (vcb) of NSGA-II.

Table A.11: Examined cases for setting 4 (str) of NSGA-II.

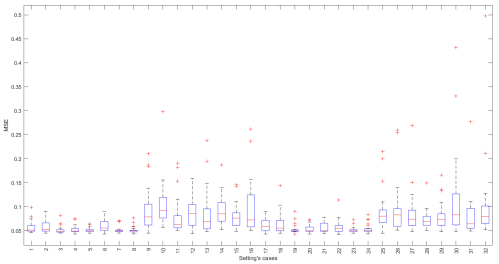
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 ps=20 mut=1	17	pd=0.8 n=5 vcb=10 ps=20 mut=1
2	pd=0.6 n=5 vcb=10 ps=20 mut=2	18	pd=0.8 n=5 vcb=10 ps=20 mut=2
3	pd=0.6 n=5 vcb=10 ps=40 mut=1	19	pd=0.8 n=5 vcb=10 ps=40 mut=1
4	pd=0.6 n=5 vcb=10 ps=40 mut=2	20	pd=0.8 n=5 vcb=10 ps=40 mut=2
5	pd=0.6 n=5 vcb=20 ps=20 mut=1	21	pd=0.8 n=5 vcb=20 ps=20 mut=1
6	pd=0.6 n=5 vcb=20 ps=20 mut=2	22	pd=0.8 n=5 vcb=20 ps=20 mut=2
7	pd=0.6 n=5 vcb=20 ps=40 mut=1	23	pd=0.8 n=5 vcb=20 ps=40 mut=1
8	pd=0.6 n=5 vcb=20 ps=40 mut=2	24	pd=0.8 n=5 vcb=20 ps=40 mut=2
9	pd=0.6 n=5 vcb=20 ps=40 mut=2	25	pd=0.8 n=5 vcb=20 ps=40 mut=2
10	pd=0.6 n=5 vcb=20 ps=40 mut=2	26	pd=0.8 n=5 vcb=20 ps=40 mut=2
11	pd=0.6 n=10 vcb=10 ps=40 mut=1	27	pd=0.8 n=10 vcb=10 ps=40 mut=1
12	pd=0.6 n=10 vcb=10 ps=40 mut=2	28	pd=0.8 n=10 vcb=10 ps=40 mut=2
13	pd=0.6 n=10 vcb=20 ps=20 mut=1	29	pd=0.8 n=10 vcb=20 ps=20 mut=1
14	pd=0.6 n=10 vcb=20 ps=20 mut=2	30	pd=0.8 n=10 vcb=20 ps=20 mut=2
15	pd=0.6 n=10 vcb=20 ps=40 mut=1	31	pd=0.8 n=10 vcb=20 ps=40 mut=1
16	pd=0.6 n=10 vcb=20 ps=40 mut=2	32	pd=0.8 n=10 vcb=20 ps=40 mut=2



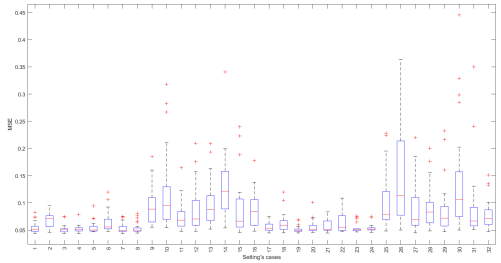
(a) Training MSE (str=1)



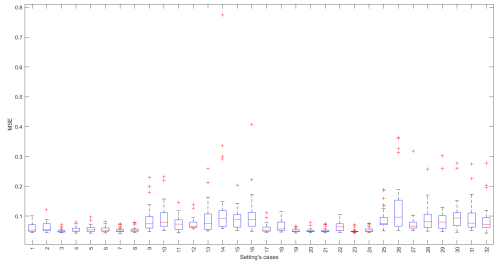
(b) Training MSE (str=2)



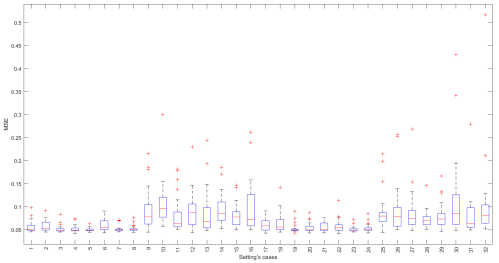
(c) Training MSE (str=3)



(d) Testing MSE (str=1)



(e) Testing MSE (str=2)

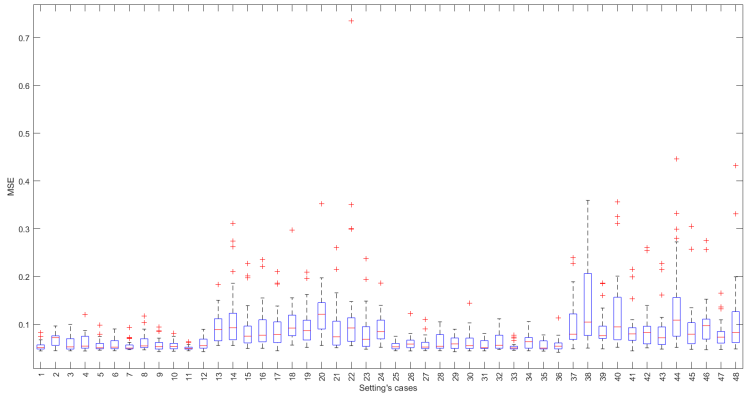


(f) Testing MSE (str=3)

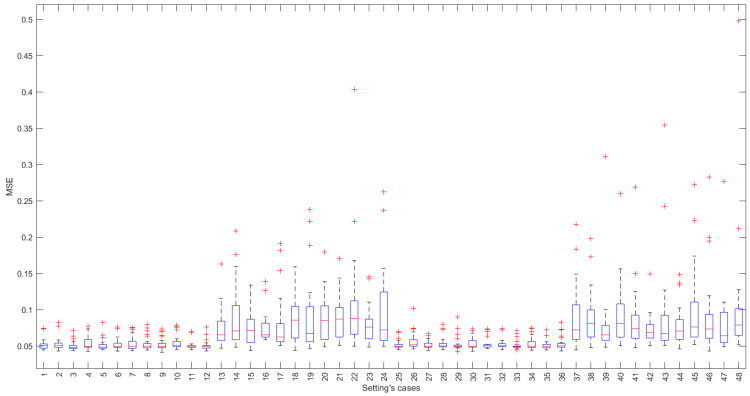
Figure A.25: Boxplots of MSE for setting 4 (*str*) of NSGA-II.

Table A.12: Examined cases for setting 5 (ps) of NSGA-II.

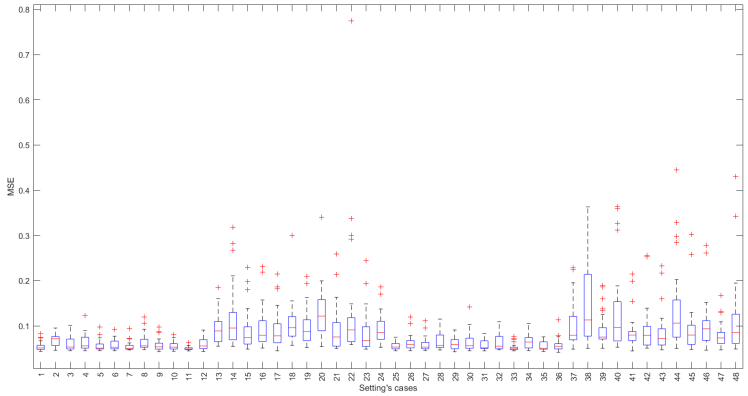
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 str=1 mut=1	25	pd=0.8 n=5 vcb=10 str=1 mut=1
2	pd=0.6 n=5 vcb=10 str=1 mut=2	26	pd=0.8 n=5 vcb=10 str=1 mut=2
3	pd=0.6 n=5 vcb=10 str=2 mut=1	27	pd=0.8 n=5 vcb=10 str=2 mut=1
4	pd=0.6 n=5 vcb=10 str=2 mut=2	28	pd=0.8 n=5 vcb=10 str=2 mut=2
5	pd=0.6 n=5 vcb=10 str=3 mut=1	29	pd=0.8 n=5 vcb=10 str=3 mut=1
6	pd=0.6 n=5 vcb=10 str=3 mut=2	30	pd=0.8 n=5 vcb=10 str=3 mut=2
7	pd=0.6 n=5 vcb=20 str=1 mut=1	31	pd=0.8 n=5 vcb=20 str=1 mut=1
8	pd=0.6 n=5 vcb=20 str=1 mut=2	32	pd=0.8 n=5 vcb=20 str=1 mut=2
9	pd=0.6 n=5 vcb=20 str=2 mut=1	33	pd=0.8 n=5 vcb=20 str=2 mut=1
10	pd=0.6 n=5 vcb=20 str=2 mut=2	34	pd=0.8 n=5 vcb=20 str=2 mut=2
11	pd=0.6 n=5 vcb=20 str=3 mut=1	35	pd=0.8 n=5 vcb=20 str=3 mut=1
12	pd=0.6 n=5 vcb=20 str=3 mut=2	36	pd=0.8 n=5 vcb=20 str=3 mut=2
13	pd=0.6 n=10 vcb=10 str=1 mut=1	37	pd=0.8 n=10 vcb=10 str=1 mut=1
14	pd=0.6 n=10 vcb=10 str=1 mut=2	38	pd=0.8 n=10 vcb=10 str=1 mut=2
15	pd=0.6 n=10 vcb=10 str=2 mut=1	39	pd=0.8 n=10 vcb=10 str=2 mut=1
16	pd=0.6 n=10 vcb=10 str=2 mut=2	40	pd=0.8 n=10 vcb=10 str=2 mut=2
17	pd=0.6 n=10 vcb=10 str=3 mut=1	41	pd=0.8 n=10 vcb=10 str=3 mut=1
18	pd=0.6 n=10 vcb=10 str=3 mut=2	42	pd=0.8 n=10 vcb=10 str=3 mut=2
19	pd=0.6 n=10 vcb=20 str=1 mut=1	43	pd=0.8 n=10 vcb=20 str=1 mut=1
20	pd=0.6 n=10 vcb=20 str=1 mut=2	44	pd=0.8 n=10 vcb=20 str=1 mut=2
21	pd=0.6 n=10 vcb=20 str=2 mut=1	45	pd=0.8 n=10 vcb=20 str=2 mut=1
22	pd=0.6 n=10 vcb=20 str=2 mut=2	46	pd=0.8 n=10 vcb=20 str=2 mut=2
23	pd=0.6 n=10 vcb=20 str=3 mut=1	47	pd=0.8 n=10 vcb=20 str=3 mut=1
24	pd=0.6 n=10 vcb=20 str=3 mut=2	48	pd=0.8 n=10 vcb=20 str=3 mut=2



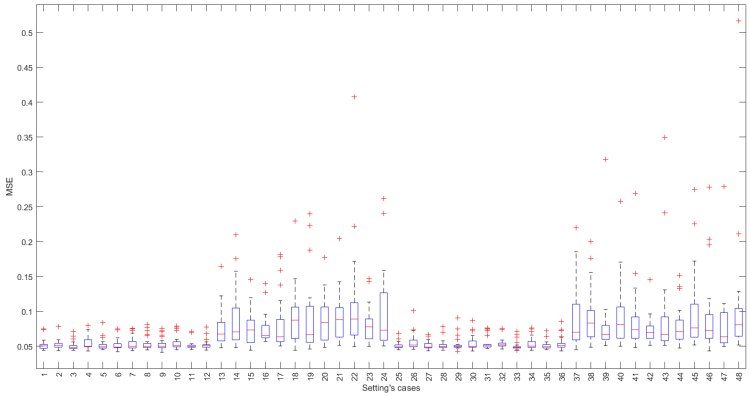
(a) Training MSE (ps=20)



(b) Training MSE (ps=40)



(c) Testing MSE (ps=20)

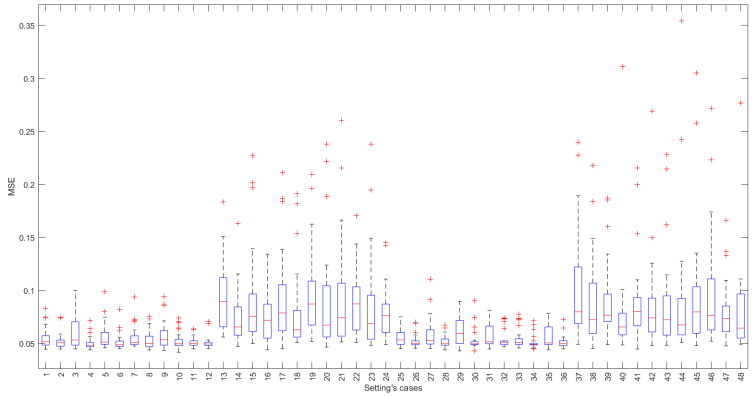


(d) Testing MSE (ps=40)

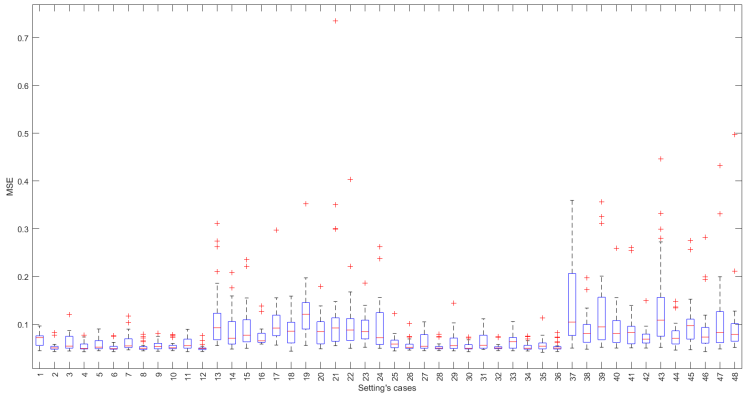
Figure A.26: Boxplots of MSE for setting 5 (ps) of NSGA-II.

Table A.13: Examined cases for setting 6 (mut) of NSGA-II.

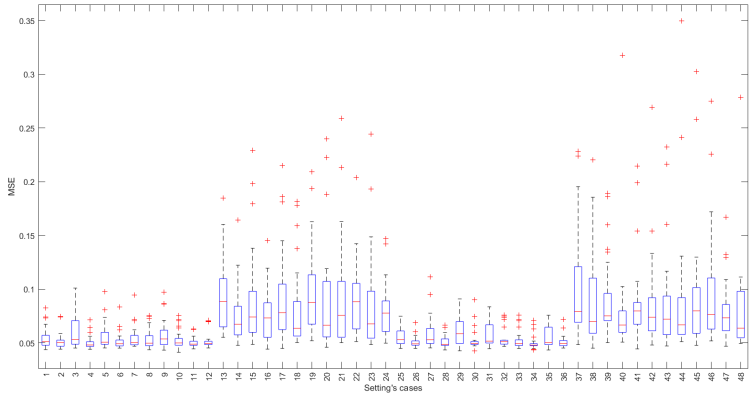
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 str=1 ps=20	25	pd=0.8 n=5 vcb=10 str=1 ps=20
2	pd=0.6 n=5 vcb=10 str=1 ps=40	26	pd=0.8 n=5 vcb=10 str=1 ps=40
3	pd=0.6 n=5 vcb=10 str=2 ps=20	27	pd=0.8 n=5 vcb=10 str=2 ps=20
4	pd=0.6 n=5 vcb=10 str=2 ps=40	28	pd=0.8 n=5 vcb=10 str=2 ps=40
5	pd=0.6 n=5 vcb=10 str=3 ps=20	29	pd=0.8 n=5 vcb=10 str=3 ps=20
6	pd=0.6 n=5 vcb=10 str=3 ps=40	30	pd=0.8 n=5 vcb=10 str=3 ps=40
7	pd=0.6 n=5 vcb=20 str=1 ps=20	31	pd=0.8 n=5 vcb=20 str=1 ps=20
8	pd=0.6 n=5 vcb=20 str=1 ps=40	32	pd=0.8 n=5 vcb=20 str=1 ps=40
9	pd=0.6 n=5 vcb=20 str=2 ps=20	33	pd=0.8 n=5 vcb=20 str=2 ps=20
10	pd=0.6 n=5 vcb=20 str=2 ps=40	34	pd=0.8 n=5 vcb=20 str=2 ps=40
11	pd=0.6 n=5 vcb=20 str=3 ps=20	35	pd=0.8 n=5 vcb=20 str=3 ps=20
12	pd=0.6 n=5 vcb=20 str=3 ps=40	36	pd=0.8 n=5 vcb=20 str=3 ps=40
13	pd=0.6 n=10 vcb=10 str=1 ps=20	37	pd=0.8 n=10 vcb=10 str=1 ps=20
14	pd=0.6 n=10 vcb=10 str=1 ps=40	38	pd=0.8 n=10 vcb=10 str=1 ps=40
15	pd=0.6 n=10 vcb=10 str=2 ps=20	39	pd=0.8 n=10 vcb=10 str=2 ps=20
16	pd=0.6 n=10 vcb=10 str=2 ps=40	40	pd=0.8 n=10 vcb=10 str=2 ps=40
17	pd=0.6 n=10 vcb=10 str=3 ps=20	41	pd=0.8 n=10 vcb=10 str=3 ps=20
18	pd=0.6 n=10 vcb=10 str=3 ps=40	42	pd=0.8 n=10 vcb=10 str=3 ps=40
19	pd=0.6 n=10 vcb=20 str=1 ps=20	43	pd=0.8 n=10 vcb=20 str=1 ps=20
20	pd=0.6 n=10 vcb=20 str=1 ps=40	44	pd=0.8 n=10 vcb=20 str=1 ps=40
21	pd=0.6 n=10 vcb=20 str=2 ps=20	45	pd=0.8 n=10 vcb=20 str=2 ps=20
22	pd=0.6 n=10 vcb=20 str=2 ps=40	46	pd=0.8 n=10 vcb=20 str=2 ps=40
23	pd=0.6 n=10 vcb=20 str=3 ps=20	47	pd=0.8 n=10 vcb=20 str=3 ps=20
24	pd=0.6 n=10 vcb=20 str=3 ps=40	48	pd=0.8 n=10 vcb=20 str=3 ps=40



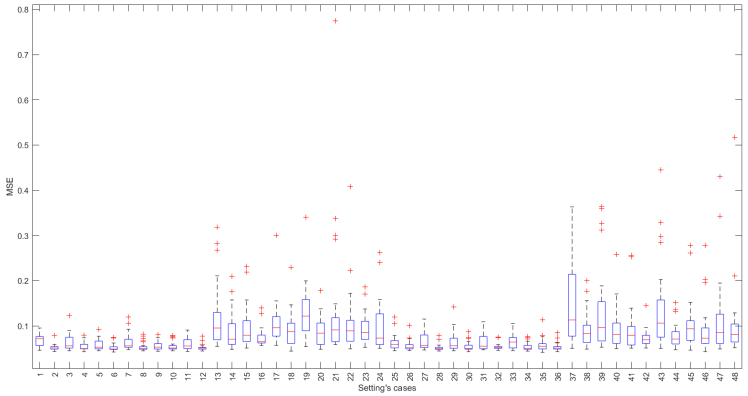
(a) Training MSE (mut=1)



(b) Training MSE (mut=2)



(c) Testing MSE (mut=1)



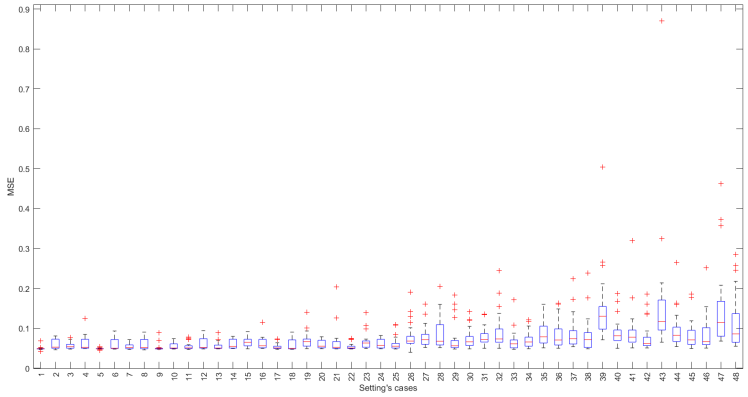
(d) Testing MSE (mut=2)

Figure A.27: Boxplots of MSE for setting 6 (*mut*) of NSGA-II.

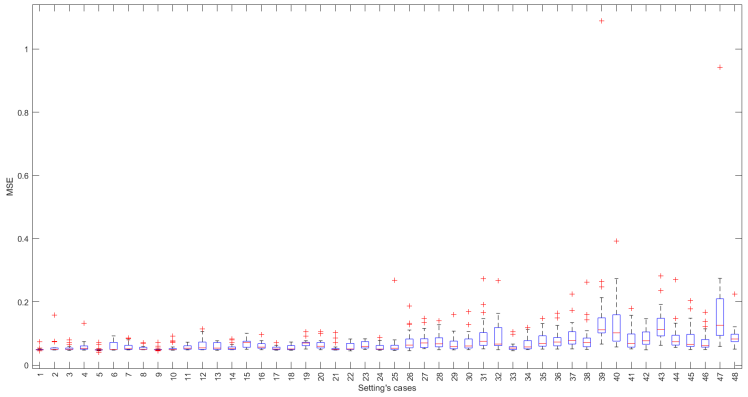
A.3 Examined cases for the multi-objective particle swarm optimization algorithm

Table A.14: Examined cases for setting 1 (pd) of MOPSO.

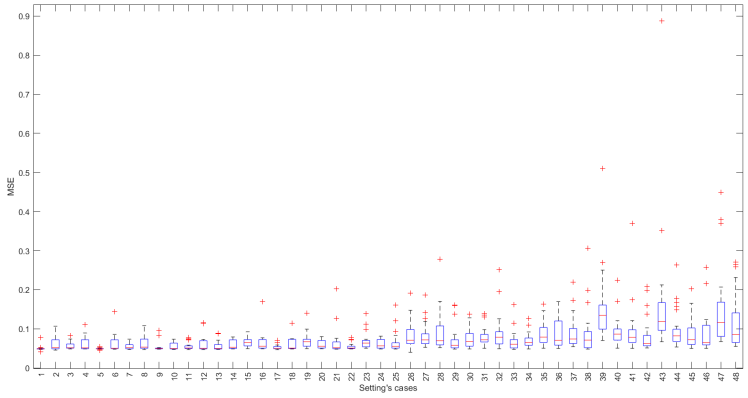
ID	Examined cases	ID	Examined cases
1	n=5 vcb=10 str=1 ps=20 vel=1	25	n=10 vcb=10 str=1 ps=20 vel=1
2	n=5 vcb=10 str=1 ps=20 vel=2	26	n=10 vcb=10 str=1 ps=20 vel=2
3	n=5 vcb=10 str=1 ps=40 vel=1	27	n=10 vcb=10 str=1 ps=40 vel=1
4	n=5 vcb=10 str=1 ps=40 vel=2	28	n=10 vcb=10 str=1 ps=40 vel=2
5	n=5 vcb=10 str=2 ps=20 vel=1	29	n=10 vcb=10 str=2 ps=20 vel=1
6	n=5 vcb=10 str=2 ps=20 vel=2	30	n=10 vcb=10 str=2 ps=20 vel=2
7	n=5 vcb=10 str=2 ps=40 vel=1	31	n=10 vcb=10 str=2 ps=40 vel=1
8	n=5 vcb=10 str=2 ps=40 vel=2	32	n=10 vcb=10 str=2 ps=40 vel=2
9	n=5 vcb=10 str=3 ps=20 vel=1	33	n=10 vcb=10 str=3 ps=20 vel=1
10	n=5 vcb=10 str=3 ps=20 vel=2	34	n=10 vcb=10 str=3 ps=20 vel=2
11	n=5 vcb=10 str=3 ps=40 vel=1	35	n=10 vcb=10 str=3 ps=40 vel=1
12	n=5 vcb=10 str=3 ps=40 vel=2	36	n=10 vcb=10 str=3 ps=40 vel=2
13	n=5 vcb=20 str=1 ps=20 vel=1	37	n=10 vcb=20 str=1 ps=20 vel=1
14	n=5 vcb=20 str=1 ps=20 vel=2	38	n=10 vcb=20 str=1 ps=20 vel=2
15	n=5 vcb=20 str=1 ps=40 vel=1	39	n=10 vcb=20 str=1 ps=40 vel=1
16	n=5 vcb=20 str=1 ps=40 vel=2	40	n=10 vcb=20 str=1 ps=40 vel=2
17	n=5 vcb=20 str=2 ps=20 vel=1	41	n=10 vcb=20 str=2 ps=20 vel=1
18	n=5 vcb=20 str=2 ps=20 vel=2	42	n=10 vcb=20 str=2 ps=20 vel=2
19	n=5 vcb=20 str=2 ps=40 vel=1	43	n=10 vcb=20 str=2 ps=40 vel=1
20	n=5 vcb=20 str=2 ps=40 vel=2	44	n=10 vcb=20 str=2 ps=40 vel=2
21	n=5 vcb=20 str=3 ps=20 vel=1	45	n=10 vcb=20 str=3 ps=20 vel=1
22	n=5 vcb=20 str=3 ps=20 vel=2	46	n=10 vcb=20 str=3 ps=20 vel=2
23	n=5 vcb=20 str=3 ps=40 vel=1	47	n=10 vcb=20 str=3 ps=40 vel=1
24	n=5 vcb=20 str=3 ps=40 vel=2	48	n=10 vcb=20 str=3 ps=40 vel=2



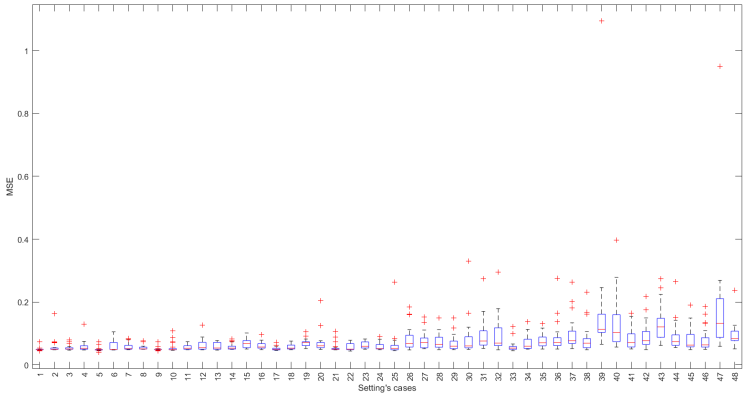
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)



(c) Testing MSE (pd=0.6)

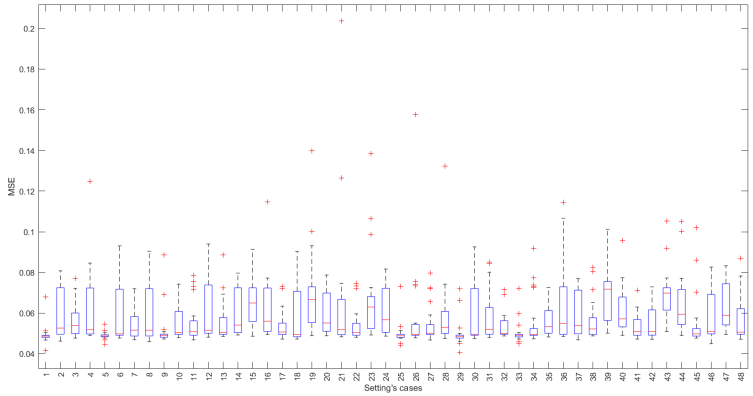


(d) Testing MSE (pd=0.8)

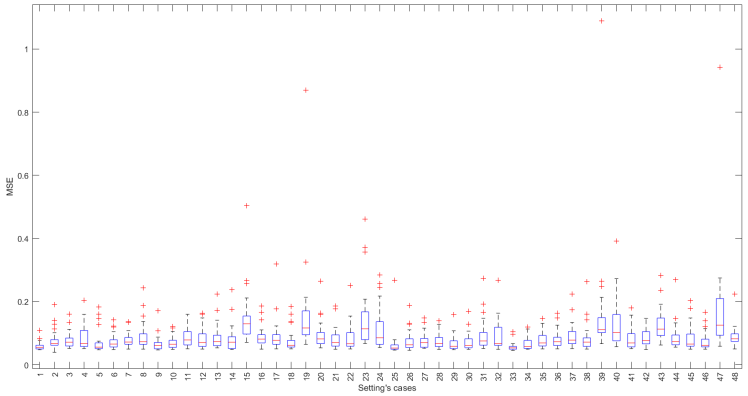
Figure A.28: Boxplots of MSE for setting 1 (pd) of MOPSO.

Table A.15: Examined cases for setting 2 (n) of MOPSO.

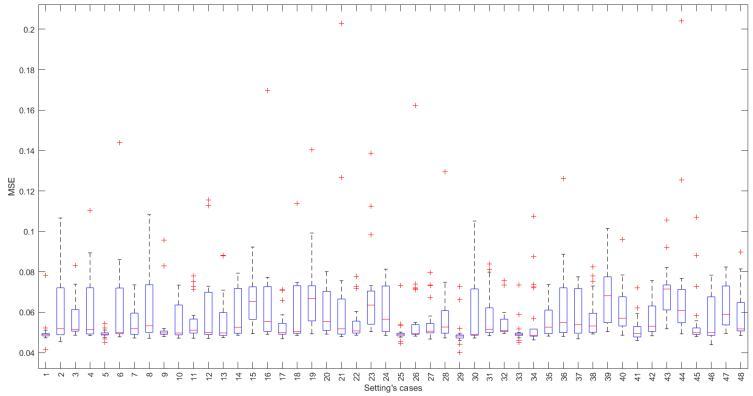
ID	Examined cases	ID	Examined cases
1	pd=0.6 vcb=10 str=1 ps=20 vel=1	25	pd=0.8 vcb=10 str=1 ps=20 vel=1
2	pd=0.6 vcb=10 str=1 ps=20 vel=2	26	pd=0.8 vcb=10 str=1 ps=20 vel=2
3	pd=0.6 vcb=10 str=1 ps=40 vel=1	27	pd=0.8 vcb=10 str=1 ps=40 vel=1
4	pd=0.6 vcb=10 str=1 ps=40 vel=2	28	pd=0.8 vcb=10 str=1 ps=40 vel=2
5	pd=0.6 vcb=10 str=2 ps=20 vel=1	29	pd=0.8 vcb=10 str=2 ps=20 vel=1
6	pd=0.6 vcb=10 str=2 ps=20 vel=2	30	pd=0.8 vcb=10 str=2 ps=20 vel=2
7	pd=0.6 vcb=10 str=2 ps=40 vel=1	31	pd=0.8 vcb=10 str=2 ps=40 vel=1
8	pd=0.6 vcb=10 str=2 ps=40 vel=2	32	pd=0.8 vcb=10 str=2 ps=40 vel=2
9	pd=0.6 vcb=10 str=3 ps=20 vel=1	33	pd=0.8 vcb=10 str=3 ps=20 vel=1
10	pd=0.6 vcb=10 str=3 ps=20 vel=2	34	pd=0.8 vcb=10 str=3 ps=20 vel=2
11	pd=0.6 vcb=10 str=3 ps=40 vel=1	35	pd=0.8 vcb=10 str=3 ps=40 vel=1
12	pd=0.6 vcb=10 str=3 ps=40 vel=2	36	pd=0.8 vcb=10 str=3 ps=40 vel=2
13	pd=0.6 vcb=20 str=1 ps=20 vel=1	37	pd=0.8 vcb=20 str=1 ps=20 vel=1
14	pd=0.6 vcb=20 str=1 ps=20 vel=2	38	pd=0.8 vcb=20 str=1 ps=20 vel=2
15	pd=0.6 vcb=20 str=1 ps=40 vel=1	39	pd=0.8 vcb=20 str=1 ps=40 vel=1
16	pd=0.6 vcb=20 str=1 ps=40 vel=2	40	pd=0.8 vcb=20 str=1 ps=40 vel=2
17	pd=0.6 vcb=20 str=2 ps=20 vel=1	41	pd=0.8 vcb=20 str=2 ps=20 vel=1
18	pd=0.6 vcb=20 str=2 ps=20 vel=2	42	pd=0.8 vcb=20 str=2 ps=20 vel=2
19	pd=0.6 vcb=20 str=2 ps=40 vel=1	43	pd=0.8 vcb=20 str=2 ps=40 vel=1
20	pd=0.6 vcb=20 str=2 ps=40 vel=2	44	pd=0.8 vcb=20 str=2 ps=40 vel=2
21	pd=0.6 vcb=20 str=3 ps=20 vel=1	45	pd=0.8 vcb=20 str=3 ps=20 vel=1
22	pd=0.6 vcb=20 str=3 ps=20 vel=2	46	pd=0.8 vcb=20 str=3 ps=20 vel=2
23	pd=0.6 vcb=20 str=3 ps=40 vel=1	47	pd=0.8 vcb=20 str=3 ps=40 vel=1
24	pd=0.6 vcb=20 str=3 ps=40 vel=2	48	pd=0.8 vcb=20 str=3 ps=40 vel=2



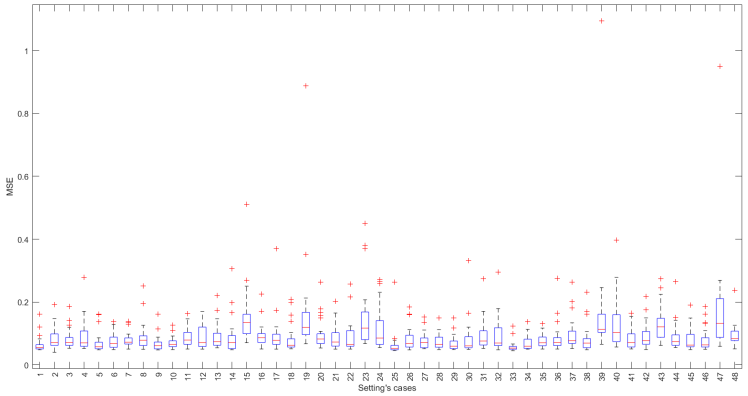
(a) Training MSE (n=5)



(b) Training MSE (n=10)



(c) Testing MSE (n=5)

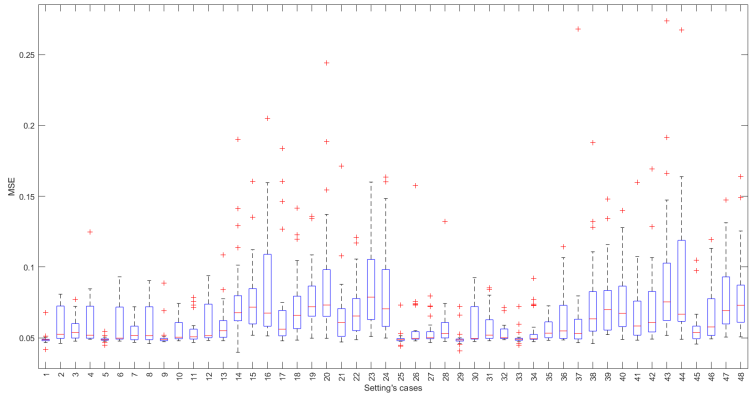


(d) Testing MSE (n=10)

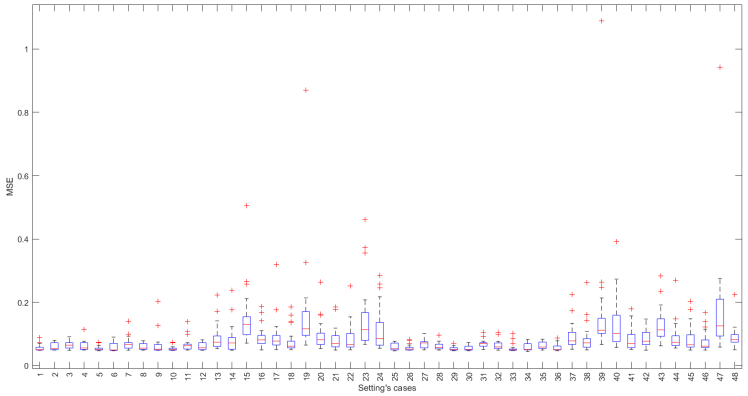
Figure A.29: Boxplots of MSE for setting 2 (n) of MOPSO.

Table A.16: Examined cases for setting 3 (vcb) of MOPSO.

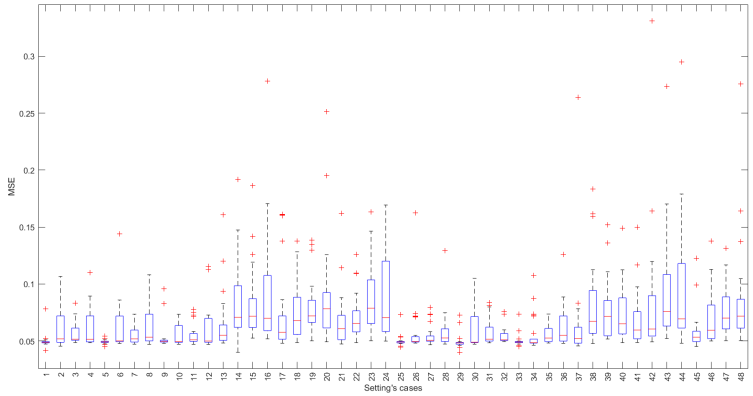
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 str=1 ps=20 vel=1	25	pd=0.8 n=5 str=1 ps=20 vel=1
2	pd=0.6 n=5 str=1 ps=20 vel=2	26	pd=0.8 n=5 str=1 ps=20 vel=2
3	pd=0.6 n=5 str=1 ps=40 vel=1	27	pd=0.8 n=5 str=1 ps=40 vel=1
4	pd=0.6 n=5 str=1 ps=40 vel=2	28	pd=0.8 n=5 str=1 ps=40 vel=2
5	pd=0.6 n=5 str=2 ps=20 vel=1	29	pd=0.8 n=5 str=2 ps=20 vel=1
6	pd=0.6 n=5 str=2 ps=20 vel=2	30	pd=0.8 n=5 str=2 ps=20 vel=2
7	pd=0.6 n=5 str=2 ps=40 vel=1	31	pd=0.8 n=5 str=2 ps=40 vel=1
8	pd=0.6 n=5 str=2 ps=40 vel=2	32	pd=0.8 n=5 str=2 ps=40 vel=2
9	pd=0.6 n=5 str=3 ps=20 vel=1	33	pd=0.8 n=5 str=3 ps=20 vel=1
10	pd=0.6 n=5 str=3 ps=20 vel=2	34	pd=0.8 n=5 str=3 ps=20 vel=2
11	pd=0.6 n=5 str=3 ps=40 vel=1	35	pd=0.8 n=5 str=3 ps=40 vel=1
12	pd=0.6 n=5 str=3 ps=40 vel=2	36	pd=0.8 n=5 str=3 ps=40 vel=2
13	pd=0.6 n=10 str=1 ps=20 vel=1	37	pd=0.8 n=10 str=1 ps=20 vel=1
14	pd=0.6 n=10 str=1 ps=20 vel=2	38	pd=0.8 n=10 str=1 ps=20 vel=2
15	pd=0.6 n=10 str=1 ps=40 vel=1	39	pd=0.8 n=10 str=1 ps=40 vel=1
16	pd=0.6 n=10 str=1 ps=40 vel=2	40	pd=0.8 n=10 str=1 ps=40 vel=2
17	pd=0.6 n=10 str=2 ps=20 vel=1	41	pd=0.8 n=10 str=2 ps=20 vel=1
18	pd=0.6 n=10 str=2 ps=20 vel=2	42	pd=0.8 n=10 str=2 ps=20 vel=2
19	pd=0.6 n=10 str=2 ps=40 vel=1	43	pd=0.8 n=10 str=2 ps=40 vel=1
20	pd=0.6 n=10 str=2 ps=40 vel=2	44	pd=0.8 n=10 str=2 ps=40 vel=2
21	pd=0.6 n=10 str=3 ps=20 vel=1	45	pd=0.8 n=10 str=3 ps=20 vel=1
22	pd=0.6 n=10 str=3 ps=20 vel=2	46	pd=0.8 n=10 str=3 ps=20 vel=2
23	pd=0.6 n=10 str=3 ps=40 vel=1	47	pd=0.8 n=10 str=3 ps=40 vel=1
24	pd=0.6 n=10 str=3 ps=40 vel=2	48	pd=0.8 n=10 str=3 ps=40 vel=2



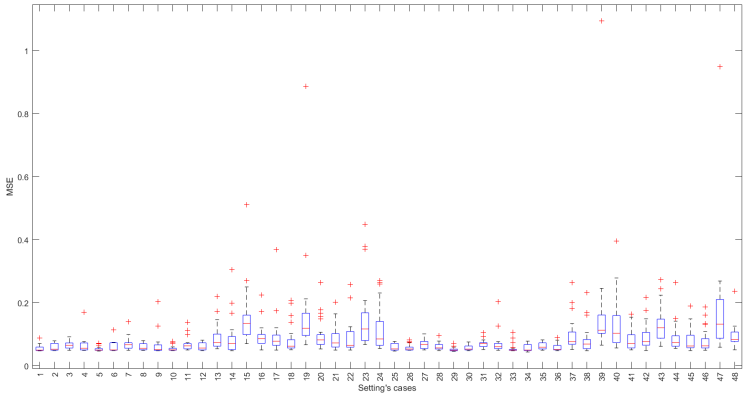
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)



(c) Testing MSE (vcb=10)

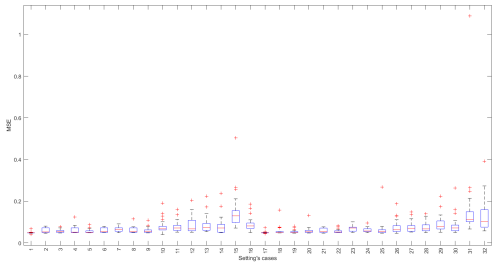


(d) Testing MSE (vcb=20)

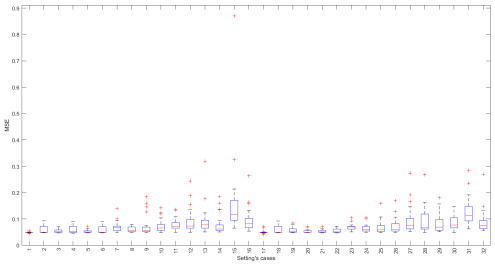
Figure A.30: Boxplots of MSE for setting 3 (vcb) of MOPSO.

Table A.17: Examined cases for setting 4 (str) of MOPSO.

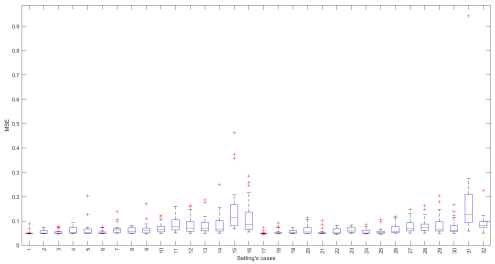
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 ps=20 vel=1	17	pd=0.8 n=5 vcb=10 ps=20 vel=1
2	pd=0.6 n=5 vcb=10 ps=20 vel=2	18	pd=0.8 n=5 vcb=10 ps=20 vel=2
3	pd=0.6 n=5 vcb=10 ps=40 vel=1	19	pd=0.8 n=5 vcb=10 ps=40 vel=1
4	pd=0.6 n=5 vcb=10 ps=40 vel=2	20	pd=0.8 n=5 vcb=10 ps=40 vel=2
5	pd=0.6 n=5 vcb=20 ps=20 vel=1	21	pd=0.8 n=5 vcb=20 ps=20 vel=1
6	pd=0.6 n=5 vcb=20 ps=20 vel=2	22	pd=0.8 n=5 vcb=20 ps=20 vel=2
7	pd=0.6 n=5 vcb=20 ps=40 vel=1	23	pd=0.8 n=5 vcb=20 ps=40 vel=1
8	pd=0.6 n=5 vcb=20 ps=40 vel=2	24	pd=0.8 n=5 vcb=20 ps=40 vel=2
9	pd=0.6 n=5 vcb=20 ps=40 vel=2	25	pd=0.8 n=5 vcb=20 ps=40 vel=2
10	pd=0.6 n=5 vcb=20 ps=40 vel=2	26	pd=0.8 n=5 vcb=20 ps=40 vel=2
11	pd=0.6 n=10 vcb=10 ps=40 vel=1	27	pd=0.8 n=10 vcb=10 ps=40 vel=1
12	pd=0.6 n=10 vcb=10 ps=40 vel=2	28	pd=0.8 n=10 vcb=10 ps=40 vel=2
13	pd=0.6 n=10 vcb=20 ps=20 vel=1	29	pd=0.8 n=10 vcb=20 ps=20 vel=1
14	pd=0.6 n=10 vcb=20 ps=20 vel=2	30	pd=0.8 n=10 vcb=20 ps=20 vel=2
15	pd=0.6 n=10 vcb=20 ps=40 vel=1	31	pd=0.8 n=10 vcb=20 ps=40 vel=1
16	pd=0.6 n=10 vcb=20 ps=40 vel=2	32	pd=0.8 n=10 vcb=20 ps=40 vel=2



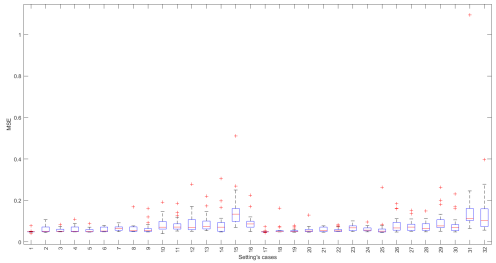
(a) Training MSE (str=1)



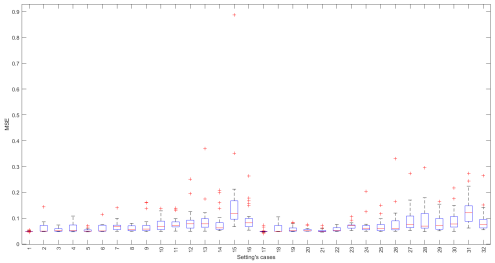
(b) Training MSE (str=2)



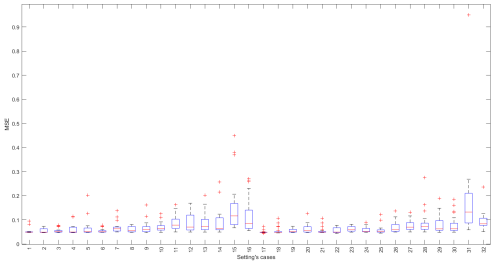
(c) Training MSE (str=3)



(d) Testing MSE (str=1)



(e) Testing MSE (str=2)

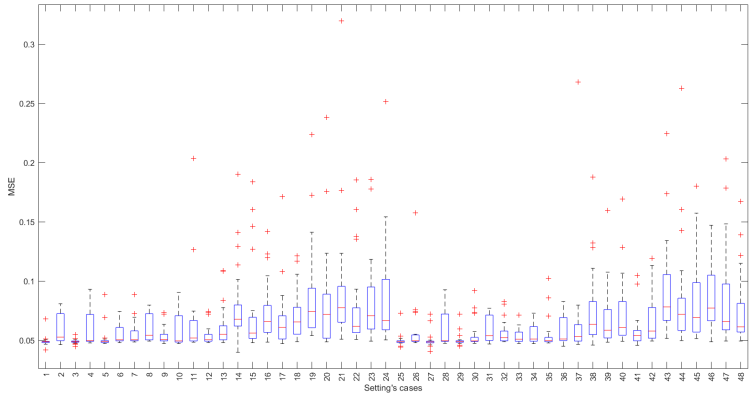


(f) Testing MSE (str=3)

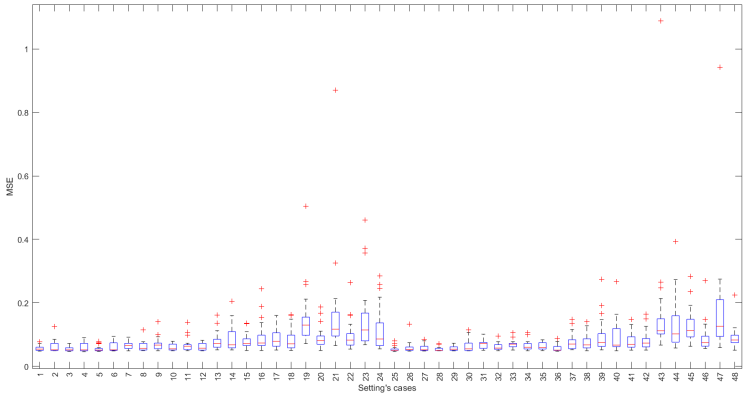
Figure A.31: Boxplots of MSE for setting 4 (str) of MOPSO.

Table A.18: Examined cases for setting 5 (ps) of MOPSO.

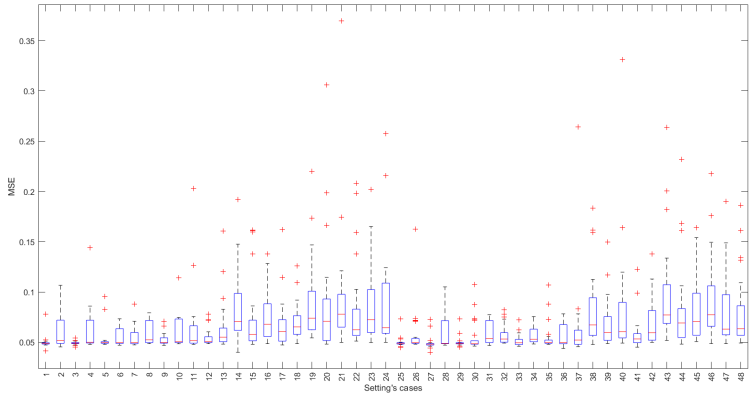
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 str=1 vel=1	25	pd=0.8 n=5 vcb=10 str=1 vel=1
2	pd=0.6 n=5 vcb=10 str=1 vel=2	26	pd=0.8 n=5 vcb=10 str=1 vel=2
3	pd=0.6 n=5 vcb=10 str=2 vel=1	27	pd=0.8 n=5 vcb=10 str=2 vel=1
4	pd=0.6 n=5 vcb=10 str=2 vel=2	28	pd=0.8 n=5 vcb=10 str=2 vel=2
5	pd=0.6 n=5 vcb=10 str=3 vel=1	29	pd=0.8 n=5 vcb=10 str=3 vel=1
6	pd=0.6 n=5 vcb=10 str=3 vel=2	30	pd=0.8 n=5 vcb=10 str=3 vel=2
7	pd=0.6 n=5 vcb=20 str=1 vel=1	31	pd=0.8 n=5 vcb=20 str=1 vel=1
8	pd=0.6 n=5 vcb=20 str=1 vel=2	32	pd=0.8 n=5 vcb=20 str=1 vel=2
9	pd=0.6 n=5 vcb=20 str=2 vel=1	33	pd=0.8 n=5 vcb=20 str=2 vel=1
10	pd=0.6 n=5 vcb=20 str=2 vel=2	34	pd=0.8 n=5 vcb=20 str=2 vel=2
11	pd=0.6 n=5 vcb=20 str=3 vel=1	35	pd=0.8 n=5 vcb=20 str=3 vel=1
12	pd=0.6 n=5 vcb=20 str=3 vel=2	36	pd=0.8 n=5 vcb=20 str=3 vel=2
13	pd=0.6 n=10 vcb=10 str=1 vel=1	37	pd=0.8 n=10 vcb=10 str=1 vel=1
14	pd=0.6 n=10 vcb=10 str=1 vel=2	38	pd=0.8 n=10 vcb=10 str=1 vel=2
15	pd=0.6 n=10 vcb=10 str=2 vel=1	39	pd=0.8 n=10 vcb=10 str=2 vel=1
16	pd=0.6 n=10 vcb=10 str=2 vel=2	40	pd=0.8 n=10 vcb=10 str=2 vel=2
17	pd=0.6 n=10 vcb=10 str=3 vel=1	41	pd=0.8 n=10 vcb=10 str=3 vel=1
18	pd=0.6 n=10 vcb=10 str=3 vel=2	42	pd=0.8 n=10 vcb=10 str=3 vel=2
19	pd=0.6 n=10 vcb=20 str=1 vel=1	43	pd=0.8 n=10 vcb=20 str=1 vel=1
20	pd=0.6 n=10 vcb=20 str=1 vel=2	44	pd=0.8 n=10 vcb=20 str=1 vel=2
21	pd=0.6 n=10 vcb=20 str=2 vel=1	45	pd=0.8 n=10 vcb=20 str=2 vel=1
22	pd=0.6 n=10 vcb=20 str=2 vel=2	46	pd=0.8 n=10 vcb=20 str=2 vel=2
23	pd=0.6 n=10 vcb=20 str=3 vel=1	47	pd=0.8 n=10 vcb=20 str=3 vel=1
24	pd=0.6 n=10 vcb=20 str=3 vel=2	48	pd=0.8 n=10 vcb=20 str=3 vel=2



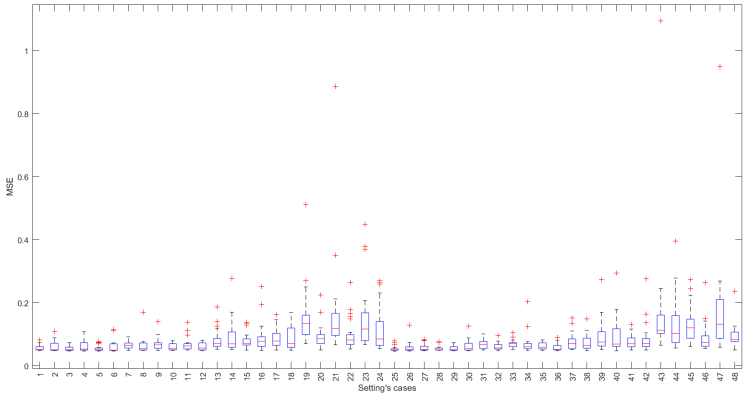
(a) Training MSE (ps=20)



(b) Training MSE (ps=40)



(c) Testing MSE (ps=20)

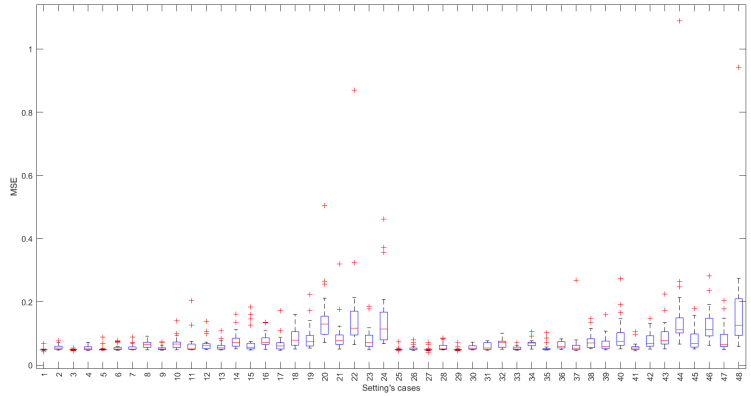


(d) Testing MSE (ps=40)

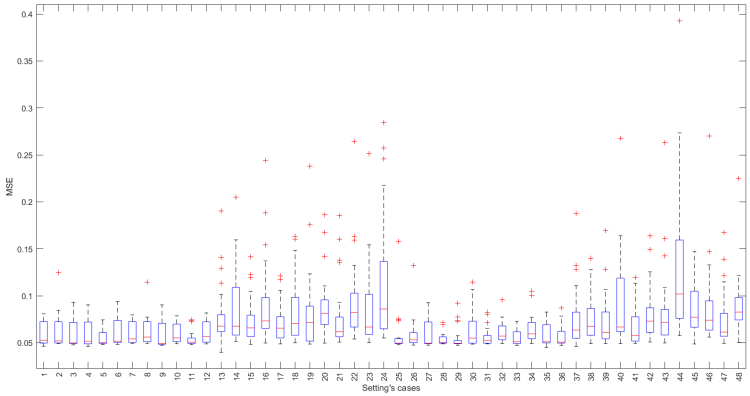
Figure A.32: Boxplots of MSE for setting 5 (ps) of MOPSO.

Table A.19: Examined cases for setting 6 (vel) of MOPSO.

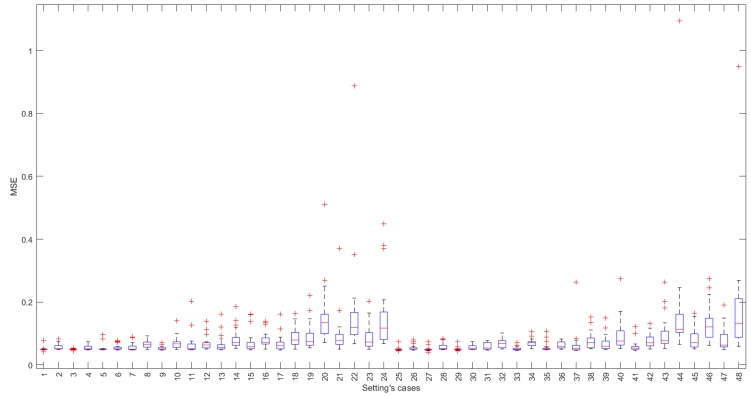
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5 vcb=10 str=1 ps=20	25	pd=0.8 n=5 vcb=10 str=1 ps=20
2	pd=0.6 n=5 vcb=10 str=1 ps=40	26	pd=0.8 n=5 vcb=10 str=1 ps=40
3	pd=0.6 n=5 vcb=10 str=2 ps=20	27	pd=0.8 n=5 vcb=10 str=2 ps=20
4	pd=0.6 n=5 vcb=10 str=2 ps=40	28	pd=0.8 n=5 vcb=10 str=2 ps=40
5	pd=0.6 n=5 vcb=10 str=3 ps=20	29	pd=0.8 n=5 vcb=10 str=3 ps=20
6	pd=0.6 n=5 vcb=10 str=3 ps=40	30	pd=0.8 n=5 vcb=10 str=3 ps=40
7	pd=0.6 n=5 vcb=20 str=1 ps=20	31	pd=0.8 n=5 vcb=20 str=1 ps=20
8	pd=0.6 n=5 vcb=20 str=1 ps=40	32	pd=0.8 n=5 vcb=20 str=1 ps=40
9	pd=0.6 n=5 vcb=20 str=2 ps=20	33	pd=0.8 n=5 vcb=20 str=2 ps=20
10	pd=0.6 n=5 vcb=20 str=2 ps=40	34	pd=0.8 n=5 vcb=20 str=2 ps=40
11	pd=0.6 n=5 vcb=20 str=3 ps=20	35	pd=0.8 n=5 vcb=20 str=3 ps=20
12	pd=0.6 n=5 vcb=20 str=3 ps=40	36	pd=0.8 n=5 vcb=20 str=3 ps=40
13	pd=0.6 n=10 vcb=10 str=1 ps=20	37	pd=0.8 n=10 vcb=10 str=1 ps=20
14	pd=0.6 n=10 vcb=10 str=1 ps=40	38	pd=0.8 n=10 vcb=10 str=1 ps=40
15	pd=0.6 n=10 vcb=10 str=2 ps=20	39	pd=0.8 n=10 vcb=10 str=2 ps=20
16	pd=0.6 n=10 vcb=10 str=2 ps=40	40	pd=0.8 n=10 vcb=10 str=2 ps=40
17	pd=0.6 n=10 vcb=10 str=3 ps=20	41	pd=0.8 n=10 vcb=10 str=3 ps=20
18	pd=0.6 n=10 vcb=10 str=3 ps=40	42	pd=0.8 n=10 vcb=10 str=3 ps=40
19	pd=0.6 n=10 vcb=20 str=1 ps=20	43	pd=0.8 n=10 vcb=20 str=1 ps=20
20	pd=0.6 n=10 vcb=20 str=1 ps=40	44	pd=0.8 n=10 vcb=20 str=1 ps=40
21	pd=0.6 n=10 vcb=20 str=2 ps=20	45	pd=0.8 n=10 vcb=20 str=2 ps=20
22	pd=0.6 n=10 vcb=20 str=2 ps=40	46	pd=0.8 n=10 vcb=20 str=2 ps=40
23	pd=0.6 n=10 vcb=20 str=3 ps=20	47	pd=0.8 n=10 vcb=20 str=3 ps=20
24	pd=0.6 n=10 vcb=20 str=3 ps=40	48	pd=0.8 n=10 vcb=20 str=3 ps=40



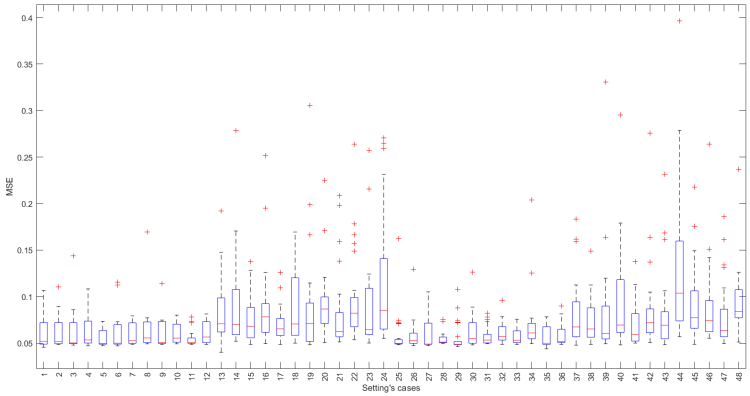
(a) Training MSE (vel=1)



(b) Training MSE (vel=2)



(c) Testing MSE (vel=1)



(d) Testing MSE (vel=2)

Figure A.33: Boxplots of MSE for setting 6 (vel) of MOPSO.

A.4 Examined cases for the BFGS method

Table A.20: Examined cases for setting 1 (pd) of BFGS.

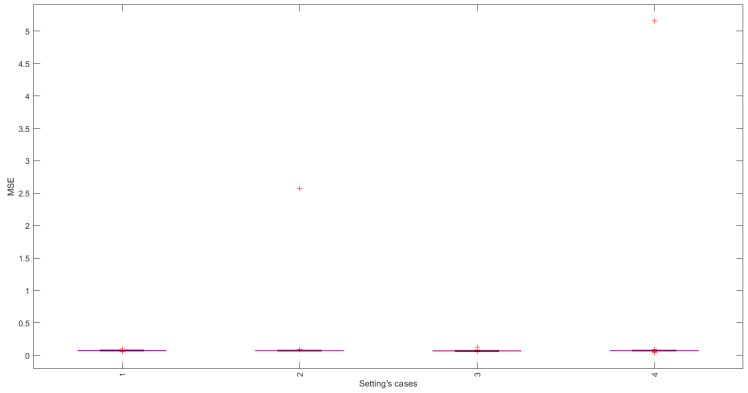
ID	Examined cases	ID	Examined cases
1	n=5 vcb=10	3	n=10 vcb=10
2	n=5 vcb=20	4	n=10 vcb=20

Table A.21: Examined cases for setting 2 (n) of BFGS.

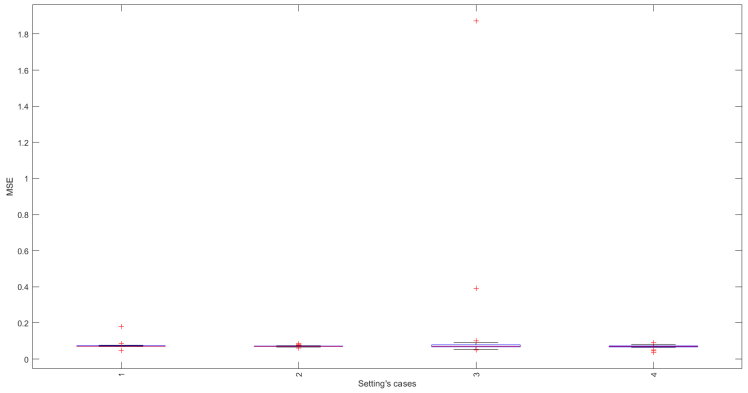
ID	Examined cases	ID	Examined cases
1	pd=0.6 vcb=10	3	pd=0.8 vcb=10
2	pd=0.6 vcb=20	4	pd=0.8 vcb=20

Table A.22: Examined cases for setting 3 (vcb) of BFGS.

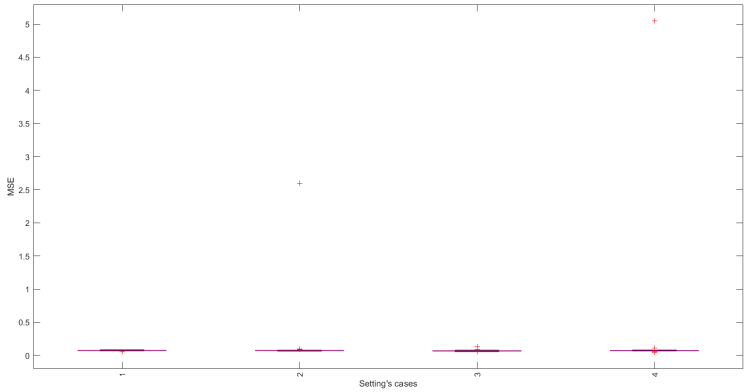
ID	Examined cases	ID	Examined cases
1	pd=0.6 n=5	3	pd=0.8 n=5
2	pd=0.6 n=10	4	pd=0.8 n=10



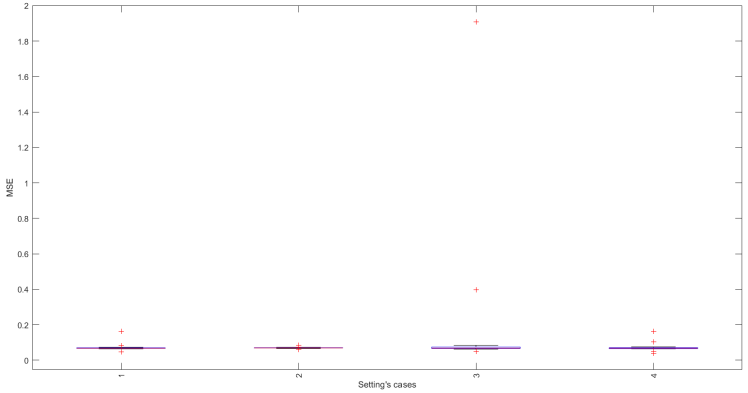
(a) Training MSE (pd=0.6)



(b) Training MSE (pd=0.8)

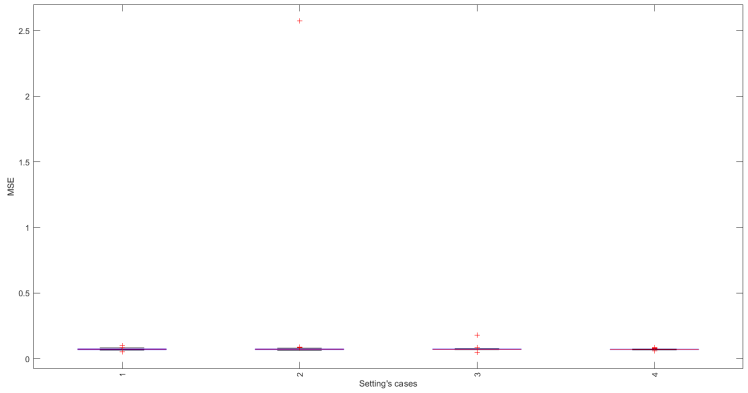


(c) Testing MSE (pd=0.6)

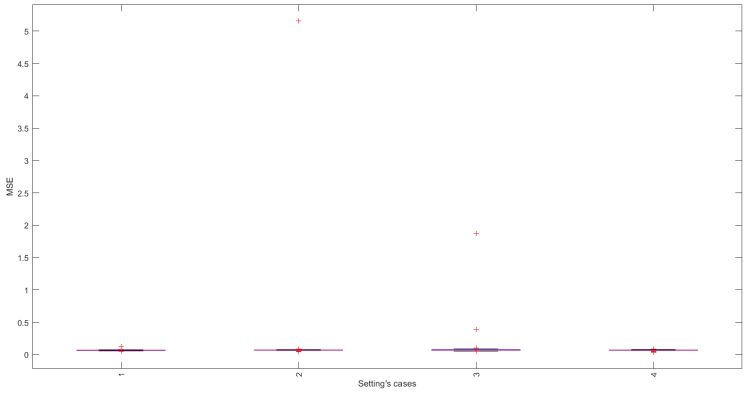


(d) Testing MSE (pd=0.8)

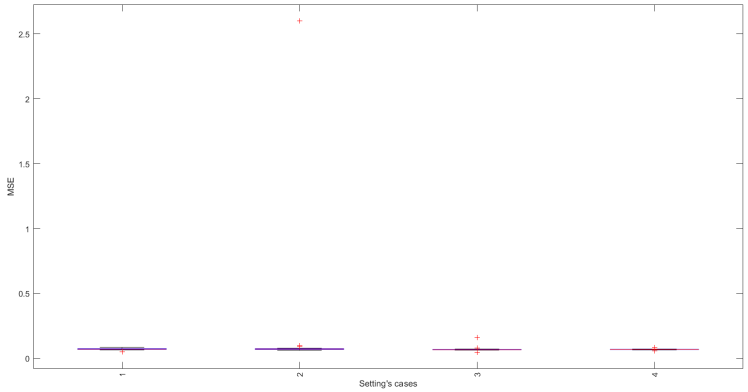
Figure A.34: Boxplots of MSE for setting 1 (pd) of BFGS.



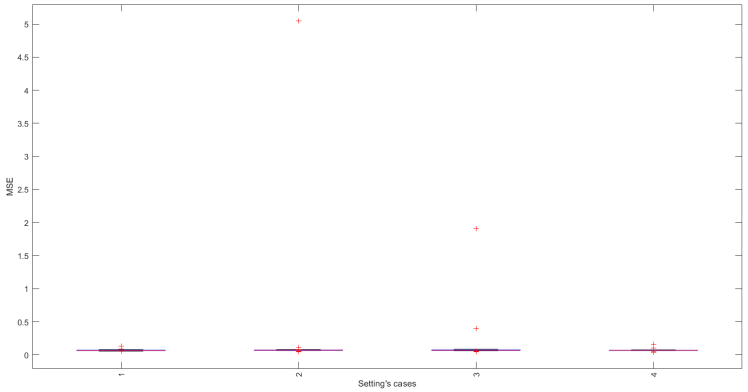
(a) Training MSE (n=5)



(b) Training MSE (n=10)

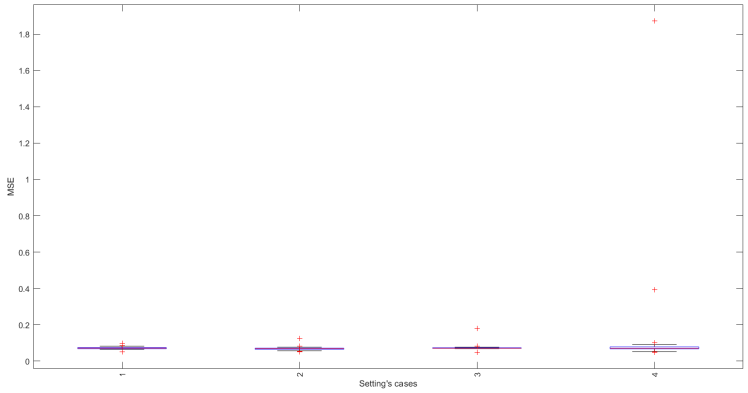


(c) Testing MSE (n=5)

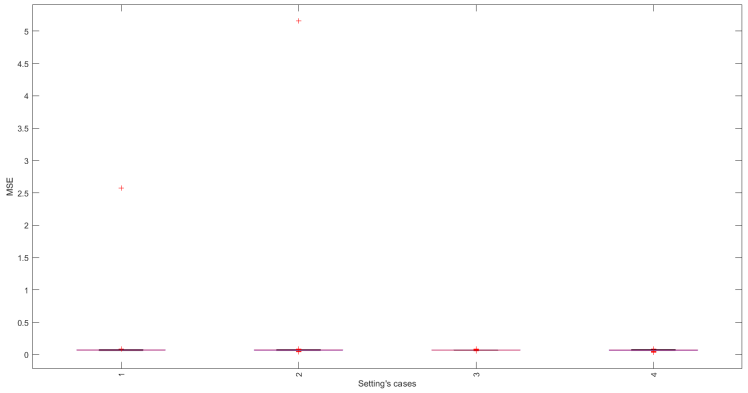


(d) Testing MSE (n=5)

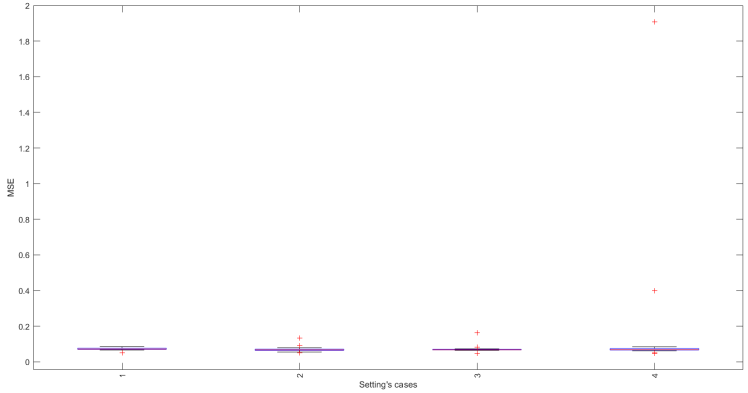
Figure A.35: Boxplots of MSE for setting 2 (n) of BFGS.



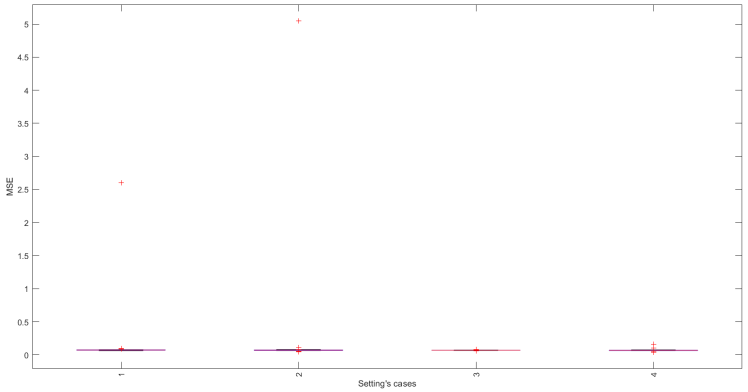
(a) Training MSE (vcb=10)



(b) Training MSE (vcb=20)



(c) Testing MSE (vcb=10)



(d) Testing MSE (vcb=20)

Figure A.36: Boxplots of MSE for setting 3 (vcb) of BFGS.