COPPE
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
UFRJ

DESIGN OF EVOLUTIONARY OPTIMIZATION AND REINFORCEMENT LEARNING TECHNIQUES FOR SMART GRID SYSTEMS CONTROL

Gabriel Matos Cardoso Leite

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Carolina Gil Marcelino

Rio de Janeiro
Abril de 2024

DESIGN OF EVOLUTIONARY OPTIMIZATION AND REINFORCEMENT LEARNING TECHNIQUES FOR SMART GRID SYSTEMS CONTROL

Gabriel Matos Cardoso Leite

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Carlos Eduardo Pedreira
Carolina Gil Marcelino

Aprovada por: Prof. Carlos Eduardo Pedreira
Profa. Carolina Gil Marcelino
Profa. Laura Silvia Bahiense da Silva Leite
Prof. Antônio de Pádua Braga
Prof. Sancho Salcedo-Sanz

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2024

Leite, Gabriel Matos Cardoso

DESIGN OF EVOLUTIONARY OPTIMIZATION AND REINFORCEMENT LEARNING TECHNIQUES FOR SMART GRID SYSTEMS CONTROL/Gabriel Matos Cardoso Leite. – Rio de Janeiro: UFRJ/COPPE, 2024.

XVIII, 92 p.: il.; 29, 7cm.

Orientadores: Carlos Eduardo Pedreira

Carolina Gil Marcelino

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 78 – 92.

1. Multi-Objective Reinforcement Learning. 2. Evolutionary Algorithms. 3. Microgrids. I. Pedreira, Carlos Eduardo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Still I rise.

Acknowledgements

I believe that no one achieves great things alone. Therefore, I hereby thank those who paved the way for me to walk. All of you have helped to build the person that I am.

First and foremost, I would like to thank God for not only standing by my side on each and every moment of my life, but also for carrying me when I thought I could not go any further. I would also like to acknowledge the role my family has played throughout my career. To my parents, Patricia and Sergio, thank you for giving me everything. To my siblings, Leticia and Arthur, thank you for your endless love and support. Without you, I would not have come this far and I certainly would not have completed this Thesis.

To my advisors, I am forever grateful for all the opportunities you have given me. Professor Carlos Pedreira, thank you for accepting me as a student after all my insistence to work with you. I know that I may not be the portrait of a perfect student, nor the one that seemed most likely to succeed, but I have given my best all the time. Professor Carolina Marcelino, you entered in my life as the new postdoctoral student from Belo Horizonte, and quickly took an important role in it. I am extremely lucky to say that my friend became my advisor. Throughout the years, we have lived and accomplished many things together. Therefore, I would like to thank both Professor Carolina and my friend Carol for not giving up on me, even when the world might have been telling you to do so.

Among the many great opportunities I have had, I must highlight the one that changed my life for the better. Coming to Spain to work with the outstanding professors, researchers, and leaders Sancho Salcedo-Sanz and Silvia Jiménez-Fernández at the UAH was a dream that came true. Thank you for receiving me, for guiding me, and supporting me.

Furthermore, I would like to thank the members of the examination committee, Laura Bahiense, Antônio Braga and Sancho Salcedo-Sanz for taking the time to read the dissertation and for your helpful insights and suggestions.

I thank the UFRJ for standing still and being my second home for all these years. I also thank the PESC, CAPGOV, and the research agencies CAPES, CNPq, and FAPERJ for their support. Moreover, I thank the UAH for the infrastructure and

support, and the PROMINT and BEST-MODA projects.

I would also like to praise my colleagues from LaSI (formerly LabIA) and my colleagues from GHEODE. It has been a pleasure to share the day and the work with all of you. Moreover, I want to express my special appreciation to my colleague Vinicius Garcia, whom I am very grateful for all the insights, discussions and jokes.

Last but definitely not least, I would like to thank the woman who came into my life and has been by my side ever since. Amanda, thank you for encouraging me and being my partner and supporter in everything. Work is easier and life is brighter with you by my side.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DESIGN OF EVOLUTIONARY OPTIMIZATION AND REINFORCEMENT LEARNING TECHNIQUES FOR SMART GRID SYSTEMS CONTROL

Gabriel Matos Cardoso Leite

Abril/2024

Orientadores: Carlos Eduardo Pedreira
Carolina Gil Marcelino

Programa: Engenharia de Sistemas e Computação

A mudança para sistemas de energia mais limpos introduz desafios complexos de Gerenciamento de Recursos Energéticos (ERM), especialmente no contexto de problemas do tipo de compromisso de unidade a ser despachada (do inglês *Unit Commitment* (UC)). UC envolve a operação de uma microrrede com diversos geradores renováveis juntamente com um fornecedor externo, o que representa um problema NP difícil ao calcular o despacho econômico para cada unidade comprometida em um horizonte de planejamento. Esta tese apresenta soluções algorítmicas para problemas de ERM de objetivo único e multiobjetivo. Primeiro, novos operadores de busca local são propostos para aprimorar uma metaheurística evolutiva para resolver um problema de otimização de ERM baseado em risco com um único objetivo. Os resultados demonstram melhor desempenho em comparação com outros algoritmos baseados em inteligência de enxame, oferecendo redução de custos e proteção contra cenários extremos. Em segundo lugar, é proposto um novo modelo de problema de decisão em ERM com vários objetivos, considerando custo, emissões de CO_2 e degradação da bateria. Um agente de aprendizagem controla a profundidade de descarga de uma bateria de íons de lítio, e um novo algoritmo multiobjetivo chamado *Multi-Objective Evolutionary Policy Search* (MEPS), que utiliza o *NeuroEvolution of Augmenting Topologies*, é proposto. O MEPS desenvolve redes neurais artificiais para estimar os valores de preferência de ação. A avaliação usando o hipervolume como métrica revela a superioridade do MEPS em relação ao aprendizado por reforço profundo padrão em problemas de referência padrão e recém-propostos. Notavelmente, o MEPS encontra redes neurais com um menor número de nós e conexões, adequadas para sistemas de controle incorporados, demonstrando sua eficácia na solução do problema ERM proposto.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DESIGN OF EVOLUTIONARY OPTIMIZATION AND REINFORCEMENT LEARNING TECHNIQUES FOR SMART GRID SYSTEMS CONTROL

Gabriel Matos Cardoso Leite

April/2024

Advisors: Carlos Eduardo Pedreira
Carolina Gil Marcelino

Department: Systems Engineering and Computer Science

The shift towards cleaner energy systems introduces complex Energy Resource Management (ERM) challenges, particularly in the context of the Unit Commitment problem. This problem involves operating a microgrid with diverse renewable generators alongside an external supplier, posing an NP-hard problem when calculating the economic dispatch for each committed unit over a planning horizon. This thesis presents algorithmic solutions to single-objective and multi-objective ERM problems. First, novel local search operators are introduced to enhance an evolutionary metaheuristic addressing a single-objective risk-based ERM optimization problem. The results demonstrate improved performance compared to other swarm-intelligence-based algorithms, offering cost reduction and protection against extreme scenarios. Second, a novel multi-objective ERM decision problem model is proposed, considering cost, CO_2 emissions, and battery degradation. A learning agent controls the depth of discharge of a Lithium-Ion battery, and the new Multi-Objective Evolutionary Policy Search (MEPS) algorithm, utilizing NeuroEvolution of Augmenting Topologies (NEAT), is introduced. The MEPS evolves artificial neural networks for estimating action-preference values. Evaluation using hypervolume as a metric reveals MEPS's superiority over standard deep reinforcement learning on both standard and proposed benchmark problems. Notably, MEPS finds neural networks with minimal nodes and connections, suitable for embedded control systems, showcasing its efficacy in solving the proposed ERM problem.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network, p. 6
BW	Bonus World environment, p. 39
C-DEEPSO	Canonical Differential Evolutionary Particle Swarm Optimization, p. 11
CD	Crowding Distance, p. 28
CNN	Convolutional Neural Networks, p. 7
DDPG	Deep Deterministic Policy Gradient, p. 5
DDST	Discontinuous Deep Sea Treasure environment, p. 37
DER	Distributed Energy Resources, p. 1
DE	Differential Evolution, p. 3
DP	Dynamic Programming, p. 12
DQN	Deep Q-Network, p. 5
DST	Deep Sea Treasure environment, p. 36
DoD	Depth of Discharge, p. 56
EA	Evolutionary Algorithm, p. 2
EFC	Equivalent Full Cycle, p. 60
EMS	Energy Management System, p. 2
EPSO	Evolutionary Particle Swarm Optimization, p. 3
EP	Evolutionary Programming, p. 11
ERM	Energy Resource Management, p. 2
ESS	Energy Storage System, p. 2

ES	Evolutionary Strategy, p. 11
EV	Electric Vehicles, p. 1
FE	Function Evaluation, p. 53
GA	Genetic Algorithm, p. 3
GHG	Greenhouse gases, p. 1
GP	Genetic Programming, p. 11
HVC	Hypervolume Contribution, p. 28
HV	Hypervolume indicator, p. 39
IC	Initial Cost, p. 58
LIB	Lithium-Ion Battery, p. 60
LSTM	Long-Short Term Memory, p. 4
LS	Local Search, p. 21
LTO	spinel lithium titanate $Li4Ti5O_{12}$, p. 56
MBST	Modified Bountiful Sea Treasure environment, p. 36
MCTS	Monte Carlo Tree Search, p. 4
MDP	Markov Decision Process, p. 12
MEPS	Multi-objective Evolutionary Policy Search, p. 28
MESH	Multi-objective Evolutionary Swarm Hybrid, p. 47
MG	Microgrid, p. 1
MLP	Multi-layer Perceptron, p. 6
MOEA	Multi-Objective Evolutionary Algorithm, p. 6
MOMDP	Multi-Objective Markov Decision Process, p. 14
MOO	Multi-Objective Optimization, p. 10
MOQDN	Multi-Objective Deep Q-Network, p. 62
MORL	Multi-Objective Reinforcement Learning, p. 15

MPSAC	Multi-Policy Soft Actor Critic, p. 36
NEAT	NeuroEvolution of Augmenting Topologies, p. 17
NN	Neural Network, p. 5
NSGA-II	Nondominated Sorting Genetic Algorithm II, p. 7
PBST	Pressurized Bountiful Sea Treasure environment, p. 38
PQL	Pareto Q-Learning, p. 36
PSO	Particle Swarm Optimization, p. 3
PV	Photovoltaic Panel, p. 3
QM	Q-Managed, p. 36
RES	Renewable Energy Sources, p. 1
RL	Reinforcement Learning, p. 2
RTP	Real-time Pricing, p. 56
ReLU	Rectified Linear Unit, p. 5
SARSA	State-Action-Reward-State-Action, p. 4
SE	Space Exploration, p. 37
SGD	Stochastic Gradient Descent, p. 17
SI-MEPS	Swarm-Intelligent MEPS, p. 47
SoC	State of Charge, p. 5
TD	Temporal Difference learning, p. 14
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution, p. 67
TWEANN	Topological and Weight Evolving Artificial Neural Networks, p. 17
UC	Unit Commitment, p. 1
VC	Vapnik-Chervonenkis dimension, p. 5
WT	Wind Turbine, p. 3
aC-DEEPSO	adaptive C-DEEPSO with Local Search, p. 23

Chapter 1

Introduction

The concentration of greenhouse gases (GHG) in the atmosphere and the effects of climate change have leveraged the employment of renewable energy sources (RESs) to become one of the top priorities of humanity and industry worldwide [20, 34, 75]. During the 2010–2020 decade, solar and wind technologies have faced a cost reduction that increased their deployment in energy generation in the detriment of fossil fuel-based energy generation [55]. Furthermore, the decarbonisation of transport and mobility services is also considered a key alternative for reducing GHG emissions [33]. Accordingly, electric vehicles (EVs) are being promoted in helping to achieve a transition towards renewable energies in this sector and to reduce the corresponding environmental impacts of vehicles based on fossil fuels [9].

Towards the goal of zero carbon dioxide emissions, microgrids (MGs) have become popular candidates to integrate distributed energy resources (DERs) into public power grids. An MG is a small-scale power grid with defined electrical boundaries that can operate independently or collaboratively with both the main public grid and other small power grids. Microgrids facilitate the effective integration of distributed generators, and consist of generation sources (solar, wind, etc.), energy storage systems (typically batteries) and loads (residential neighborhoods, business centers, hospitals, etc.). Formally, the concept of microgrids comprises a low-voltage distribution system that integrates distributed energy sources (microturbines, generators, photovoltaic panels, electric vehicles, among others) with storage devices (energy capacitors and energy storage systems), and flexible loads [83]. These systems can either function autonomously or non-autonomously by interconnecting to the public grid [46, 73].

A very important task in the operation of an MG involves identifying the optimal Unit Commitment (UC), considering technical and economic constraints over a long planning horizon, up to one year. UC refers to the problem of determining the schedule of generating units within a power system, with the goal of minimizing costs while satisfying system constraints [97]. The UC module of an MG controls

not only the committed generators and the power imported from the public grid, but also the power exported to or imported from the energy storage system (ESS) units [104].

Therefore, due to the climate-dependent nature of renewable resources such as solar and wind, and the fluctuating market price, UC in MGs with ESS units is a complex constrained optimization problem. In this way, the Energy Resource Management (ERM) of an MG can be considered as a type of UC problem, where a player operates an MG with various renewable generators integrated with an external supplier [5]. An efficient ERM solution leads to not only a profitable but also a sustainable and reliable operation of the MG [46]. Thus, one of the most important optimization problems in daily operation scheduling and planning is to control the dispatch of energy-generating units. Moreover, according to the International Electrotechnical Commission in the standard IEC 61970, the computer systems that ensure the effective management operation of microgrids are called Energy Management Systems (EMSs) [23]. Consequently, the EMS of a microgrid includes both supply-side and demand-side management, while ensuring that system constraints are met.

EMSs are usually categorized as centralized or decentralized, according to their operation mode. In centralized mode, the management system is located in a central station and connected to the DERs via communication lines for control and to exchange data. On the other hand, in decentralized mode, each DER operates independently and manages itself using a local controller, thus eliminating the need for communication. Therefore, this thesis is concerned with the ERM control of centralized energy management systems.

Computational intelligence methods have been widely employed in many real-world problems. In particular, the development of algorithms to solve energy management-related problems has been an active area of research [66, 149]. Among the different classes of algorithms, it is worth noting evolutionary algorithms (EAs) and reinforcement learning (RL) techniques. The former addresses any class of optimization problem and comprises a set of metaheuristic techniques that use nature-inspired concepts to efficiently explore and exploit the search space [43]. On the other hand, the latter addresses a specific class of optimization problem that involves sequential decision making. Reinforcement learning characterizes techniques in which a learning agent interacts with its surrounding environment, without any prior knowledge, aiming at learning an ideal set of control actions [123]. Moreover, both EAs and RL techniques are considered suitable candidates for solving ERM problems because they can handle nonlinear and non-convex problems containing continuous and integer variables [95].

1.1 Evolutionary Algorithms in ERM

Evolutionary algorithms (EAs) are a class of metaheuristics, inspired by Charles Darwin's theory of natural evolution, that reflect the process of natural selection, wherein the fittest individuals are selected for reproduction to produce the offspring of the next generation. From the computational perspective, such algorithms encompass problem-independent techniques that can be applied to a wide range of problems, with purpose-appropriate values for the decision variables of an optimization problem so that the objective function is optimized [43]. Compared to classical optimization methods, such as Newton's method, linear, nonlinear, and dynamic programming, etc, EAs are not restricted to the optimization problem being linear/non-linear or convex/non-convex. They are also less computational and converge to an optimal solution in less time [15]. From the multiple applications of evolutionary algorithms present in the renewable energy sources and microgrids literature, it is worth noting Genetic Algorithms (GAs) [128], Differential Evolution (DE) [40], and an evolutionary variant of classical Particle Swarm Optimization (PSO), Evolutionary Particle Swarm Optimization (EPSO) [85].

PSO variants are often present in the ERM and UC literature. For instance, in [53] a regularized PSO algorithm is proposed for optimally charging and discharging battery energy according to electricity prices and the availability of renewable energy. Besides the cost functions, a dynamic penalty function that takes into consideration energy generation from RESs is employed. However, battery degradation cost is not taken into consideration. Similarly, a modified cost function is proposed in [54] which considers not only the electricity cost but also the cost of the ESS charging and discharging cycles in a microgrid containing photovoltaic panels (PVs) and wind turbines (WTs). Then, a PSO algorithm is applied to solve the resulting energy management system problem. The characteristics of the battery used in the microgrid are analyzed by [90], in which the depth of discharge, cycle life and investment costs are considered in minimizing the investment and ESS replacement costs. Yet, electric vehicle dynamics are not considered. Another application of a PSO-based algorithm is presented by [69], in which a quantum-based PSO algorithm is used to solve an energy management problem regarding the operation costs of day-ahead scheduling in distributed generators. Although present in the modeling, the ESS costs are not considered.

A DE algorithm is applied by [92] to solve a two-step optimization energy management problem. In this formulation, multiple microgrids containing PVs, WTs and ESSs are considered in maximizing each microgrid local RES consumption while globally maintaining stability of the management system. Similarly, DE is also employed by [40] in energy management considering PVs, WTs, ESSs, and EVs. The

goal is to evaluate EVs' potential to decrease electricity market prices and shift electricity demand during peak hours while maximizing profit. Despite promising results, neither proposal considers pollutant emissions. The ERM problem in [27] is modeled from a multi-objective perspective in which not only the electricity costs, but also the pollutant emissions were minimized, while maximizing customer satisfaction. Then, the resulting problem is solved using a multi-objective genetic algorithm, NSGA-III. A major drawback of the proposed solution is that it does not consider the stochasticity of renewable energy generation.

1.2 Reinforcement Learning in ERM

When dealing with sequential decision optimization problems, classical and heuristic optimization algorithms suffer from some drawbacks. For instance, for each iteration, the algorithm needs to be restarted and, thus, extensive computational resources are required. As a consequence, it is difficult to use such algorithms in real-time decision making [148].

Reinforcement Learning (RL) is a branch of machine learning in which a learning agent interacts with its surrounding environment, without any prior knowledge, aiming at learning an ideal behavior to solve one or more tasks by maximizing the obtained rewards [123]. This behavior, often called *policy*, denotes a set of actions that leads the agent to maximize the reward signal received [123]. To learn to solve a task, the agent evaluates state-action pairs, takes actions, and then it receives rewards that indicate how good its actions were, according to the given environment. RL has been shown to achieve compelling results, even with no prior domain knowledge [62–64, 87, 112, 135, 142, 145].

Recently, various studies have been conducted using RL as a model-free alternative to metaheuristic-based approaches in ERM problems [2, 86]. In [96], a near-optimal ESS operation strategy using state-action-reward-state-action (SARSA) algorithm is presented. The operation strategy is then employed to manage uncertainties in forecasting wind power generation. Wind power uncertainty is also tackled in [67], in which a two-fold solution is proposed. First, a long-short term memory (LSTM) model is used to predict wind power. Then, an energy management system for wind power is modeled and solved as a sequential decision problem using deep Q-learning. In the proposed model, the learning agent is responsible for finding the optimal charge/discharge decision-making strategy for the ESS present in the MG. Regarding the life span and maintenance of ESSs, a deep RL method is proposed in [140] that combines Monte Carlo tree search (MCTS) and state-action estimation using deep neural networks for preventive maintenance in a set of batteries. A combination of Q-learning and MCTS is also presented in [111], in which the learn-

ing agent handles the dispatching of the ESS in the microgrid using a multiperiod stochastic model, while also considering battery degradation cost.

Electric vehicles' popularity and availability are substantially increasing, with the objective of reducing carbon dioxide emissions. Accordingly, there are new challenges for reducing costs and increasing performance. In [65] a deep deterministic policy gradient (DDPG) is used to implement a multi-objective energy management system in electric vehicles. The reward function employed in the algorithm combines both energy loss and aging cost objectives with electrical and thermal constraints to create a scalar reward signal. In addition, deep q-networks (DQN) are applied in [124] to find an energy management strategy that minimizes both hydrogen consumption and fuel cell system degradation, while maintaining the battery state of charge (SoC) stability in fuel cell hybrid electric vehicles. Even though all the presented works performed well, the majority of RL studies focus on single-objective decision problems with scalar rewards.

1.3 Neural Networks in Reinforcement Learning

Roughly all reinforcement learning algorithms require the estimation of value functions that express the quality of being in a particular state (in terms of total expected reward in the long run) or the quality of executing a specific action in a particular state. The simplest method to construct such functions is by regularly updating a table containing a value for every state (or state-action pair). However, this approach proves impractical for large scale problems. To handle problems with a large or even infinite state space, it is imperative to utilize the generalization abilities of function approximators [51, 52].

Feedforward neural networks (NNs) are a particular case of such function approximators that have been successfully employed in combination with reinforcement learning methods [87, 112, 135]. Specifically, neural networks employing Rectified Linear Unit (ReLU) as activation function have become popular due to their practical performance [110]. Yet, the performance of these networks is subject to their complexity, that is, correctly choosing both the network's topology/architecture (number of layers, nodes and connections), and the size of the parameters. A network with a badly chosen topology may not be able to solve the task it was designed for, even with a large amount of training steps. Consequently, several works have attempted to establish a relationship between a NN complexity and its learning capability. For instance, the authors in [6, 10] have leveraged the Vapnik-Chervonenkis (VC) dimension [134] theory to show, by using the concept of fat-shattering dimension, that the NN error in learning can be bounded by constraining either the number of parameters (weights, layers, biases, and nodes) or the size of the parameters. In

this thesis, we focus on bounding the NNs error by constraining their number of parameters.

With respect to constraining the topology of a NN, Yarotsky [144] has proven that adapting the NN's topology to the function being approximated leads to a smaller upper bound for learning error, compared to fixing the topology and only adjusting the weights, for the specific case of Sobolev Spaces. Moreover, Frankle and Carbin [41] have shown that for a given dense network, there is a subnetwork with fewer nodes and connections that, when trained in isolation, achieves comparable performance as the original one. Recently, Bartlett et al. [11] have reinforced the importance of constraining the topology of a neural network by tightening the learning capability bound of ReLU-based NNs. The VC-dimension of this class of networks has been proven to have a linear dependency on the number of nodes and connections. Since this thesis deals with ReLU-based NNs for multi-objective RL policies, it can be said that the final performance of each policy directly depends not only on the adjustment of the weight values, but also on the correct choice of the number of nodes and connections in each NN [11].

1.4 Neural Networks in Multi-Objective Problems

Several real problems are inherently multi-objective. For example, a person buying a car is interested in maximizing the performance while minimizing the cost. Despite this multi-objective nature, most artificial neural network (ANN) studies dealt with such kind of problems from a single-objective standpoint, by adopting a predefined weight vector to combine different objectives into one single goal [108, 120].

The first ideas to employ ANNs in multi-objective problems were first reported in mid-1990s. Liu and Kadirkamanathan [68] treated the L_2-norm and L_∞ as objectives along with the number of nonzero elements in a polynomial basis function network of a Gaussian radial basis function network. Then, a GA was used to search for the weights of the network. However, the problem was solved using a min-max scalarization approach, which led to only one solution at the end. Afterward, Kottathra and Attikiouzel [57] modeled the multi-objective problem of minimizing both the mean squared error, and the number of nodes in the hidden layer of a Multi-layer Perceptron (MLP). The authors solved the presented problem by using an unconstrained mixed integer nonlinear multicriteria optimization technique to optimize not only the number of nodes in the hidden layer but also the set of weights in the MLP.

With the increasing popularity of multi-objective EAs (MOEAs) [37], recent studies have focused on employing ANNs as part of solutions to multi-objective problems. In [1], a multi-objective problem of minimizing the error in two disjoint

data sets is formulated and solved by optimizing the weights of the ANNs using an algorithm based on differential evolution. An attempt to optimize not only the accuracy but also the network's size is presented in [70]. The authors used a multi-objective genetic algorithm, NSGA-II, to search for the near-optimal set of hyperparameters in convolutional neural networks (CNN), considering activation functions, number and size of convolutional layers, learning rate, and so on. The hyperparameters of CNNs are also optimized in [113], in which the authors employed a multi-objective DE algorithm to optimize CNNs in classifying COVID-19-infected patients. In spite of recent works, the literature still lacks studies that target the optimization of ANN in multi-objective reinforcement learning problems. This thesis attempts to shed light on this research gap.

1.5 Reinforcement Learning in Real World

Scheduling of real-world MG systems is a fragile and expensive operation. Besides, real systems do not provide separate environments for training and evaluation [32]. Thus, learning usually is done through the use of real data along with simulations of the real system. In this regard, RL algorithms are suitable solution candidates as they can be trained offline for scheduling using different load and generation profiles, and then be applied online for other load and generation profiles [8].

However, most of the existing RL models based on ANNs are considered as black-boxes, and therefore cannot be blindly used in many cases [12]. Instead of fully trusting output actions, including the human in the process by first analyzing the action and its corresponding effects could be an alternative to increase the acceptance of RL models in energy-related problems [121, 147]. In this thesis, we are concerned with providing RL models that act as a decision-making assistance to human operator for mid-term scheduling [99].

1.6 Thesis Statement

As a result, the main research question that motivates this thesis is: "*how to perform optimal control in single and multi-objective energy resource management problems?*". Throughout this thesis we seek to provide answers to this question, addressing specific questions towards a solution for the principle question. Some specific questions addressed are:

1. Does the combination of a local search procedure with an existing EA improve its performance in solving single-objective ERM problems considering the occurrence of extreme events?

2. Is the combination of NEAT with non-dominance sorting for multi-policy search competitive against traditional multi-objective reinforcement learning approaches in a multi-objective ERM control problem?

3. Is the combination of heuristics and an existing EA with multi-objective neuroevolutionary reinforcement learning beneficial for multi-objective energy management control? How can the different concepts be combined and integrated with each other?

In this thesis, we attempt to answer these questions, with an applied research focus, by presenting original solutions to ERM-based problems.

1.7 Document Organization

This proposal is structured as follows: Chapter 2 gives an introduction to single and multi-objective optimization, evolutionary algorithms, single and multi-objective reinforcement learning, and neuroevolutionary concepts. Then, Chapter 3 details the contributions of this work along with their calibration through the use of benchmark test solutions. The statistical protocol used to assess the preliminary validation experiments is also presented in this chapter. Afterwards, Chapter 4 describes the real-world problems in which the proposals are evaluated. There is also a discussion about the experimental results and their correspondence with this thesis scope. Chapter 5 shows an exploration analysis of a hyperparameters set on the multi-objective real-world problem proposed. Finally, Chapter 6 summarizes the main contributions, conclusions, and future research directions of this thesis.

Chapter 2

Foundations

2.1 Optimization

The term optimization refers to the study of solving problems by finding one or more feasible solutions that correspond to the extreme values of one or more objectives [30]. The need for finding such optimal solutions is mostly motivated by the desire to design a solution for the lowest possible fabrication cost, the maximum reliability, or similar purposes. In order to effectively solve the problem, optimal solutions must lie within the problem's domains and respect its constraints. Because of such extreme properties of optimal solutions, the development of optimization methods are of great importance in practical engineering and business decision-making problems.

2.1.1 Single-Objective Optimization

When an optimization problem modeling a physical system involves only one objective function, the task of finding the optimal solution is called single-objective optimization. In single-objective problems, only one objective function is involved in the modeling of the physical system and, therefore, the task of finding the optimal solution is called single-objective optimization. Conceptually, a single-objective function is a mapping of type $f : \mathbb{R}^n \to \mathbb{R}$ expressed as:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}), \\ &\text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0, & \text{for } i = 1, \ldots, r \\ h_j(\mathbf{x}) = 0, & \text{for } j = 1, \ldots, l \\ \mathbf{x} \in X, \end{cases} \end{aligned} \tag{2.1}$$

in which $X \subset \mathbb{R}^n$, f denotes the objective function, and x is the vector composed of n optimization variables. Each of the functions $g_i(\mathbf{x}) \leq 0, \text{for } i = 1, \ldots, r$,

$h_j(\mathbf{x}) = 0, \text{for } j = 1, \ldots, l$ stand for inequality and equality constraints, respectively. Moreover, a solution $\mathbf{x} \in X$ that satisfies all the constraints is called a feasible solution to the problem [13].

2.1.2 Multi-Objective Optimization

The majority of real-world search and optimization problems naturally involve multiple objectives. In multi-objective problems, the optimization task involves finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements denote each of the objective functions.

The principle of searching for extremes previously mentioned cannot be applied to only one objective, as the multiple objective functions are usually conflicting. Hence, different solutions may produce trade-offs (conflicting scenarios) among different objectives. This means that one is discouraged from choosing a solution which is optimal with respect to only one objective, since a compromise with other objectives is also required [21].

A multi-objective optimization (MOO), also often referred to as vector optimization, problem has a number of objective functions which are to be minimized or maximized. Similarly to single-objective optimization, any feasible solution must satisfy a set of constraints. The MOO problem in its general form is defined as:

$$\begin{aligned} &\text{minimize/maximize } f_m(\mathbf{x}), \quad m = 1, \ldots, M, \\ &\text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0, & \text{for } i = 1, \ldots, r \\ h_j(\mathbf{x}) = 0, & \text{for } j = 1, \ldots, l \\ \mathbf{x} \in X, \end{cases} \end{aligned} \tag{2.2}$$

in which there are M objective functions, $X \subset \mathbb{R}^n$, and $\mathbf{x}$ is the vector with n optimization/decision variables. The inequality and equality constraints are $g_i(\mathbf{x}) \leq 0, \text{for } i = 1, \ldots, r$ and $h_j(\mathbf{x}) = 0, \text{for } j = 1, \ldots, l$, respectively [30].

2.2 Evolutionary Algorithms

In natural systems, evolution produces behaviors that are optimized, yet not optimal, over the biological hierarchy of individuals and populations. As a result, evolution can be seen as an iterated, population-based process of genetic variation and selection capable of solving novel problems in new ways [38]. Historically, the biological concepts regarding evolution inspired the development of a research area called evolutionary computing. Its origin dates back to 1948 when Alan Turing pre-

sented the work "*Intelligent Machinery*" in which he states "*There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of various kinds of search.*" [26]. Afterwards, with the advance of computational resources, this area rapidly has grown as an active research field with more than 2000 publications in journals and conferences [114].

Evolutionary algorithms (EAs) are a class of generic population-based metaheuristics. Within EAs, different paradigms emerged: genetic algorithms (GAs) [49], evolution strategies (ES) [102], evolutionary programming (EP) [39], and genetic programming (GP) [58]. The various paradigms follow a general outline of maintaining a population of individuals, mating parents to generate offspring, and selecting the following generation based on a fitness evaluation. This thesis proposal focuses on evolutionary algorithms and a variant inspired by the swarm intelligence phenomenon, namely evolutionary particle swarm optimization (EPSO) [85].

2.2.1 C-DEEPSO algorithm

The Canonical Differential Evolutionary Particle Swarm Optimization (C-DEEPSO) [78] is a single-objective evolutionary algorithm based on swarm intelligence foundations from PSO [56] that merges vectorial operations from Differential Evolution (DE) [40]. C-DEEPSO has been successfully applied in solving several optimization problems such as: active power dispatch in large scale grids minimizing the costs of production [78], controlling the cascade operation of hydropower plants [80], active and reactive power dispatch in hybrid microgrid systems operation with [81] and without electric vehicles [76].

As an evolutionary algorithm, C-DEEPSO relies on mutation, recombination and selection operations. Each particle in C-DEEPSO follows a movement rule defined by Equations (2.3) and (2.4):

$$V_t = w_I^* \cdot V_{t-1} + w_A^* \cdot (X_{st} + F(X_r - X_{t-1})) \ + w_C^* \cdot C \cdot (X_{gb}^* - X_{t-1}), \tag{2.3}$$

$$X_t = X_{t-1} + V_t, \tag{2.4}$$

in which X_{st} is a different particle from X_{t-1} obtained from the *DE/current-to-best/1* operator. Besides storing each particle's best position, C-DEEPSO employs a collective memory archive containing a portion of the highest ranked solutions found in previous generations, to refine the search process by providing a wider view of the search space. The subscript t indicates the current generation, X is the particle's position, V denotes the velocity of the particle and X_{gb} is the highest

ranked solution ever found by the swarm. The weights corresponding to inertia, assimilation and communication are mutated as follows:

$$w^* = w + \tau \cdot N(0,1), \tag{2.5}$$

in which τ denotes the mutation rate parameter. Here term C in Equation (2.3) represents a $D \times D$ diagonal matrix of Bernoulli random variables that are sampled for each particle, where D stands for the decision space dimension. The intuition behind using matrix C is based on a technique named "stochastic star communication topology" (see [81]). This technique limits the amount of information allowed to be used from the global best at each iteration. This is controlled by a communication probability parameter, P. After a generation, C-DEEPSO saves a small subset of the best solutions from the swarm in a memory archive called *Memory B* [76]. Hence, with this memory mechanism, the term X_r is obtained according to the $S_g P_b - rnd$ strategy [76].

The global best solution X_{gb} is mutated to avoid getting trapped in particular regions of the search space and to attract the current individual to a promising region. The current individual is then slightly moved in the search space using a Gaussian Distribution scaled by the parameter τ. Thus, X_{gb} is mutated in accordance with Equation (2.6):

$$X^*_{gb} = X_{gb}[1 + \tau \cdot N(0,1)]. \tag{2.6}$$

2.3 Reinforcement Learning

Sequential decision making problems are ubiquitous in many scientific domains, with several application areas. In general, these problems involve a feedback loop in which a decision-maker determines a decision based on the available information at each time step. From the viewpoint of an external observer, the result of such problems is a sequence of perceived information and decisions that are performed iteratively over time.

In sequential decision making, Markov decision processes (MDP) are the formalism used to model the environment the agent is operating in. In an MDP, an environment is modelled as a set of states and actions that can be performed by an agent to control the system's state with the goal of maximizing some performance criterion. Regarding the amount of knowledge available from the environment, two paradigms can be employed: dynamic programming (DP), or reinforcement learning (RL). In the former, the full transition dynamics and reward distributions are known. The latter is associated with the more difficult setting in which there is

no prior knowledge available about the MDP [139]. In the context of ERM problems, this thesis proposal is also concerned with evolutionary methods to sequential decision ERM learning problems characterized by the RL paradigm.

2.3.1 Single-Objective Reinforcement Learning

A learning agent is a computer system capable of performing sequential autonomous actions in the environment in which it is situated. These agents are enabled to learn from their own experiences by the Reinforcement Learning framework [123]. Thus, in RL, the learning agent deals with the problem of taking decisions in unknown, possibly dynamic environments. In the standard single-objective case, the overall target of the agent is to learn a sequence of decisions that maximises the expected value of a scalar feedback signal. Essentially, these decisions relate to action selection in certain environmental states. A typical formalization of a reinforcement learning environment by means of Markov decision process (MDP) is a tuple (S, A, T, R) [123], in which:

- $S = \{s_1, \ldots, s_N\}$ denotes the state space,
- $A = \{a_1, \ldots, a_r\}$ denotes the finite set of available actions,
- $T(s'|s, a) \in [0, 1]$ is a transition function that specifies, for each state – action – next state, the probability of that next state occurring given an action at the current state, and
- $R(s, a) : S \times A \to \mathbb{R}$ is a reward function that specifies, for each state – action pair, the expected immediate reward.

The goal of an agent is to learn a policy π that maps each state to an action, so that the expected return received in the long run is maximized [101]. In this work, only deterministic transitions are considered, hence $T(s'|s, a) = 1$. The state-dependent value function of a policy π in a state s is defined as

$$V^{\pi}(s) = \mathbf{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s\right], \tag{2.7}$$

in which r_t is the reward obtained at time t and $\gamma \in [0, 1]$ is the discount factor. In a finite horizon model, the state value function becomes

$$V^{\pi}(s) = \mathbf{E}_{\pi}\left[\sum_{k=0}^{h} r_{t+k} | s_t = s\right], \tag{2.8}$$

in which h denotes the length of the horizon. The expected return from starting at state s, taking action a and following policy π is given by a $Q^\pi(s,a)$-value. For an infinite horizon model, it is expressed as:

$$Q^\pi(s,a) = \mathbf{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a \right]. \tag{2.9}$$

The optimal Q^*-values are defined as

$$Q^*(s,a) = R(s,a) + \mathbf{E}\left[\gamma \max_{a'} Q^*(s',a') \right]. \tag{2.10}$$

The Q-Learning algorithm presented in [137] provides a way to iteratively approximate Q^*. In Q-Learning, each state-action pair is stored in a Q-table and, with learning rate $\alpha \in (0,1]$, updated incrementally based on feedback values and Temporal Difference learning (TD) [122] according to the rule

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha_t \left(R(s,a) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right). \tag{2.11}$$

Assuming that all state-action pairs are visited and updated under Q-Learning, the $\hat{Q}$ estimates converge to the optimal values Q^* in either deterministic or non-deterministic MDPs [28, 125].

By selecting the first action according to the policy π, the Q-function is equivalent to the value function and can be written as

$$V^\pi(s) = \mathbf{E}_\pi \left[Q^\pi(s_t, \pi(s_t)) | s_t = s \right]. \tag{2.12}$$

Finding a policy that maximizes Equation 2.12 requires searching over a function space, which is generally an intractable problem. Therefore, policies are parameterized by some $\theta \in \Theta \subset \mathbb{R}^d$ so that search is performed in a Euclidean space of finite dimension. Consequently, the resulting problem becomes

$$\max_\theta \mathbf{E}_{\pi_\theta} \left[Q^{\pi_\theta}(s_t, \pi_\theta(s_t)) | s_t = s \right] = J(\theta). \tag{2.13}$$

In this work, policies are parameterized using neural networks.

2.3.2 Multi-Objective Reinforcement Learning

In multi-objective optimization (MOO), the objective space consists of two or more dimensions that must be optimized simultaneously [22]. Hence, a generalization of regular MDPs to multi-objective MDPs (MOMDPs) was proposed in [106, 138]. Firstly, scalar reward values in MDPs are translated into reward vectors $\mathbf{R}(s,a) \in \mathbb{R}^m$ in MOMDPs. The variable m stands for the number of objectives and the i-th

component of the reward vector denotes the reward obtained for the i-th objective. Analogously, both state and state-action value functions are vectorial:

$$\mathbf{V}^{\pi}(s) = \mathbf{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}\mathbf{r}_{t+k}|s_t = s\right], \tag{2.14}$$

$$\mathbf{Q}^{\pi}(s,a) = \mathbf{E}_{\pi}\left[\sum_{k=0}^{h}\gamma^{k}\mathbf{r}_{t+k}|s_t = s, a_t = a\right]. \tag{2.15}$$

Moreover, the multi-objective optimal state-action value is defined as

$$\mathbf{Q}^{*}(s,a) = \mathbf{R}(s,a) + \mathbf{E}\left[\gamma \max_{a'}\mathbf{Q}^{*}(s',a')\right]. \tag{2.16}$$

Considering that an agent will optimize several objectives simultaneously, there will be different optimal policies with respect to different objectives. Therefore, the optimality criteria used is, in general, the concept of Pareto dominance [30].

Generally, to establish an ordering among solutions in multi-objective reinforcement learning (MORL) problems, two solutions are compared according to Pareto dominance relation [98].

Definition 1 *Given two policies $\pi_1, \pi_2 \in \Pi$, it is said that policy π_1 strictly dominates policy π_2, denoted by $\pi_1 \succ \pi_2$, if*

$$\forall i \in \{1,\ldots,m\}, \mathbf{V}_i^{\pi_1}(s) \leq \mathbf{V}_i^{\pi_2}(s) \wedge \exists j \in \{1,\ldots,m\}, \mathbf{V}_j^{\pi_1}(s) < \mathbf{V}_j^{\pi_2}(s) \tag{2.17}$$

$\mathbf{V}^{\pi_1}(s)$ strictly improves $\mathbf{V}^{\pi_2}(s)$ in at least one objective and $\mathbf{V}^{\pi_1}(s)$ is at least equal to $\mathbf{V}^{\pi_2}(s)$ in all other objectives.

Definition 2 *Given two policies $\pi_1, \pi_2 \in \Pi$, it is said that policy π_1 is incomparable to policy π_2 if*

$$\exists i \in \{1,\ldots,m\}, \mathbf{V}_i^{\pi_1}(s) < \mathbf{V}_i^{\pi_2}(s) \wedge \exists j \in \{1,\ldots,m\}, \mathbf{V}_j^{\pi_2}(s) < \mathbf{V}_j^{\pi_1}(s). \tag{2.18}$$

$\mathbf{V}^{\pi_1}(s)$ strictly improves $\mathbf{V}^{\pi_2}(s)$ in at least one objective and $\mathbf{V}^{\pi_2}(s)$ strictly improves $\mathbf{V}^{\pi_1}(s)$ in at least one objective. Therefore, both policies are incomparable.

Definition 3 *A policy $\pi_i^{*} \in \Pi$ is said to be Pareto optimal iff it is non-dominated by any other policy π_j:*

$$\pi_i^{*} \in \Pi \leftrightarrow \nexists \pi_j \in \Pi : \pi_j \succ \pi_i^{*}. \tag{2.19}$$

The set $\Pi^{*} \subset \Pi$ with all Pareto optimal policies is named "Pareto front" [30].

Regarding the number of Pareto optimal policies a MORL approach yields after an execution, algorithms can be categorized as single-policy or multi-policy.

In single-policy approaches, the MORL problem is decomposed into several single-objective RL problems, each representing a particular pre-defined trade-off. Traditionally, a convex combination using weights $\mathbf{w} = [w_1, \ldots, w_m]$ is applied to perform a scalarization of the multiple objectives, so that $\sum_{i=0}^{m} w_i = 1$. The reformulated reward and Q-Value estimates are as follows:

$$\hat{SR}_{linear}(s,a) = \sum_{i=1}^{m} w_i \cdot R_i(s,a) \tag{2.20}$$

$$\hat{SQ}_{linear}(s,a) = \sum_{i=1}^{m} w_i \cdot Q_i(s,a). \tag{2.21}$$

Consequently, for n_p pareto optimal solutions, n_p sub-problems have to be solved.

Despite simplicity, linear scalarization functions compute a convex combination of the objectives and, therefore, are limited to only finding solutions in the convex regions of the Pareto front [129]. An alternative to this limitation is to compute a non-linear function of the objectives. Nevertheless, the algorithm's convergence cannot be guaranteed since Bellman's equation no longer holds [100, 106].

Contrary to single-policy approaches, multi-policy approaches aim at learning several optimal policies at once. To achieve this, a multi-policy DP algorithm proposed by [138] has been used as basis for several MORL algorithms. The proposed DP algorithm is able to compute a Pareto set of non-dominated policies. Its DP function is

$$\hat{Q}_{set}(s_t,a) = \mathbf{R}(s_t,a) \oplus \gamma PF\left(\cup_{a'} Q_{set}(s_{t+1},a')\right). \tag{2.22}$$

The PF operator returns only the non-dominated elements from the union of the $\hat{Q}_{set}(s_t,a)$ of all actions.

The operator $\oplus$ denotes the vector-sum between a vector $\mathbf{v}$ and a set of vectors V. This sum is performed by simply adding the components of $\mathbf{v}$ to the components of each vector $\mathbf{v}'$ in V:

$$\mathbf{v} \oplus V = \cup_{\mathbf{v}' \in V}(\mathbf{v} + \mathbf{v}'). \tag{2.23}$$

The idea in [138] is that, after propagating discounted Pareto-dominating rewards, the $\hat{Q}_{set}$s converge to a set of Pareto optimal policies. As convergence is achieved, the user can apply his/her preferences a *posteriori* to select a Pareto optimal policy.

2.4 Neuroevolution

When dealing with real world RL problems, the state space may increasingly grow. Thus, employing algorithms such as Q-Learning, that rely on tabular mapping to estimate Q-values, becomes infeasible. In this regard, artificial neural networks (ANNs) play an important role as function approximators in these problems with high-dimensional state spaces [51, 52].

To train neural networks for supervised learning, reinforcement learning, and recently, deep learning or deep reinforcement learning tasks, the dominant method is backpropagation [107]. Backpropagation is an efficient algorithm for calculating the gradient of a loss function, which, when combined with stochastic gradient descent (SGD), can modify each neural network weight to reduce loss in a greedy manner. The impressive results from Deep Q-Learning [88], Double Deep Q-Networks [132], and Dueling Deep Q-Networks [136] in game-playing have proven the effectiveness of backpropagation in RL. However, this procedure is susceptible to becoming trapped in local minima of the error function, which are numerous in the error-surface [48].

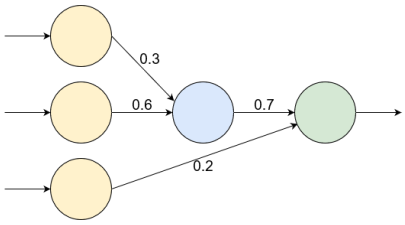
In order to overcome this limitation, by the 1980s researchers had started to employ evolutionary algorithms to train neural networks [89]. Certainly, EAs are not guaranteed to converge to the global optimum. Yet, as they evolve a diverse population of several solutions, compared to a population of only one individual like in the backpropagation approach, EAs are less susceptible to local minima [35]. Moreover, EAs are agnostic about the rate of descent with respect to the slope of the gradient. This field of research, called neuroevolution, is inspired by the biological evolution of the brain [118]. Initially, neuroevolution methods used evolutionary strategies (ES) to learn the weights of fixed topology/architecture networks [25, 44, 45, 89, 108, 120]. A potential drawback in the aforementioned approach is based on the fact that the definition of a network's topology has a major effect on its performance [6, 10]. Therefore, finding the optimal topology of an ANN requires time-consuming evaluations of potential architectures. As a result, new methods known as Topological and Weight Evolving Artificial Neural Networks (TWEANNs) have emerged as alternatives to searching not only the right ANN topology but also the right set of weights [117, 143]. This work is focused upon the latter category of methods.

2.4.1 NeuroEvolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies (NEAT) [117], is a benchmark single-objective TWEANN method. It starts with a population of simple ANNs and sequentially increases the search space dimensionality, building more complex topologies. Each individual from this population is evaluated regarding a fitness function,

and the fittest ones survive to the next generation. NEAT uses both crossover and mutation operators to perform topological and parametrical changes upon surviving networks, resulting in new complex topologies with more sophisticated behaviors.

In NEAT, an individual or genome's phenotype is represented according to Figure 2.1a and its genotype encoding is illustrated in Figure 2.1b. In order to allow comparison of network parts during recombination, NEAT employs a historical marking assigned to each connection denoted innovation. Innovation is an incremental number that starts at 1 and is increased every time a new connection/node is added to the network.



(a) Example neural network with 3 input, 1 hidden, and 1 output nodes.

start	end	i_1	h_1	i_2	h_1	i_3	o_1	h_1	o_1
Weight		0.3		0.6		0.2		0.7	
Enabled		True		True		True		True	
Innovation		1		2		3		4	

(b) Encoding for a neural network with 3 input, 1 hidden, and 1 output nodes as proposed by Stanley in [117].

Figure 2.1: Phenotype (a) and genotype (b) of a simple neural network in NEAT (adapted from [117]).

Besides the parametrical mutation applied to each connection weight, NEAT increases the search space dimensionality by using two topological mutation operators. In the first, a connection with random weight is created between two existing nodes. Note that, to preserve the feed-forward design self-recurrent links are not permitted. In the second type of mutation, a new node is created and inserted into the network in the place of an existing connection between two nodes. The procedure for inserting a new node $node_{new}$ is as follows: the selected existing connection between nodes $node_1$ and $node_2$ with weight w_{12} is disabled. Then, a new connection from $node_1$ to $node_{new}$ is created with weight equal to 1. Afterwards, a second connection is created from $node_{new}$ to $node_2$ with weight w_{12}. In this way, the complexity is increased without changing the network's behavior.

In the recombination process, the connections of two parents are aligned according to their innovation number. Despite having different weight values, connections with same innovation number are considered as matching or common. Connections that do not match are either disjoint if their innovation number is within the range of the innovation numbers of the common parent connections, or excess otherwise. Common connections are randomly selected from both parents, and transferred to offspring. Excess and disjoint connections are transferred to offspring from the fittest parent. If both parents are equally fit, excess and disjoint connections are randomly selected [117]. An exemplary illustration is presented in Figure 2.3

A problem of modifying the structure of an ANN is that a structural modification

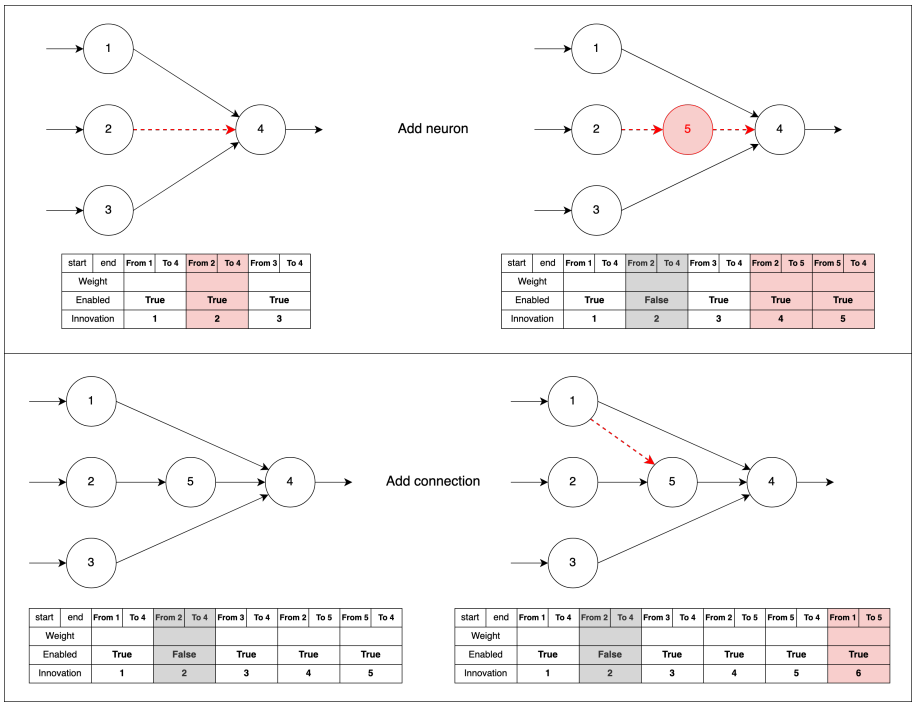


Figure 2.2: Illustration of topological mutations occurring in NEAT: Adding a new neuron (upper part) and a new connection (lower part). Genotype changes are marked in red and changes in the networks' phenotype are marked by dashed lines. Note that the innovation ids in the red boxes are updated and the add neuron mutation happens before the add link mutation w.r.t. the temporal dimension (adapted from [117]).

initially reduces the individual fitness. This would make topological innovations with unoptimized parameters be extinguished prematurely. NEAT handles this problem by speciating the population according to a topological similarity measure δ. In this way, individuals compete within their own niche, giving time for topological innovations to optimize before competing in other niches of the population. The function δ measures the distance between two genomes as a linear combination of excess (E) and disjoint (D) genes along with the average weight differences of matching genes ($\bar{W}$):

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W}. \tag{2.24}$$

The coefficients c_1, c_2 and c_3 denote the importance given to each factor. N is a normalizing constant that represents the number of genes in the largest genome and $\bar{W}$ is the difference in average connection weights. The speciation mechanism consists in iteratively comparing one genome at a time. If an individual's distance to a species' representative $\delta(X_i, X_j^s)$ is below a pre-defined speciation threshold σ^*, it is placed in that species. Otherwise, a new species is created with the current genome as its representative. Furthermore, in its initial proposal, NEAT uses an

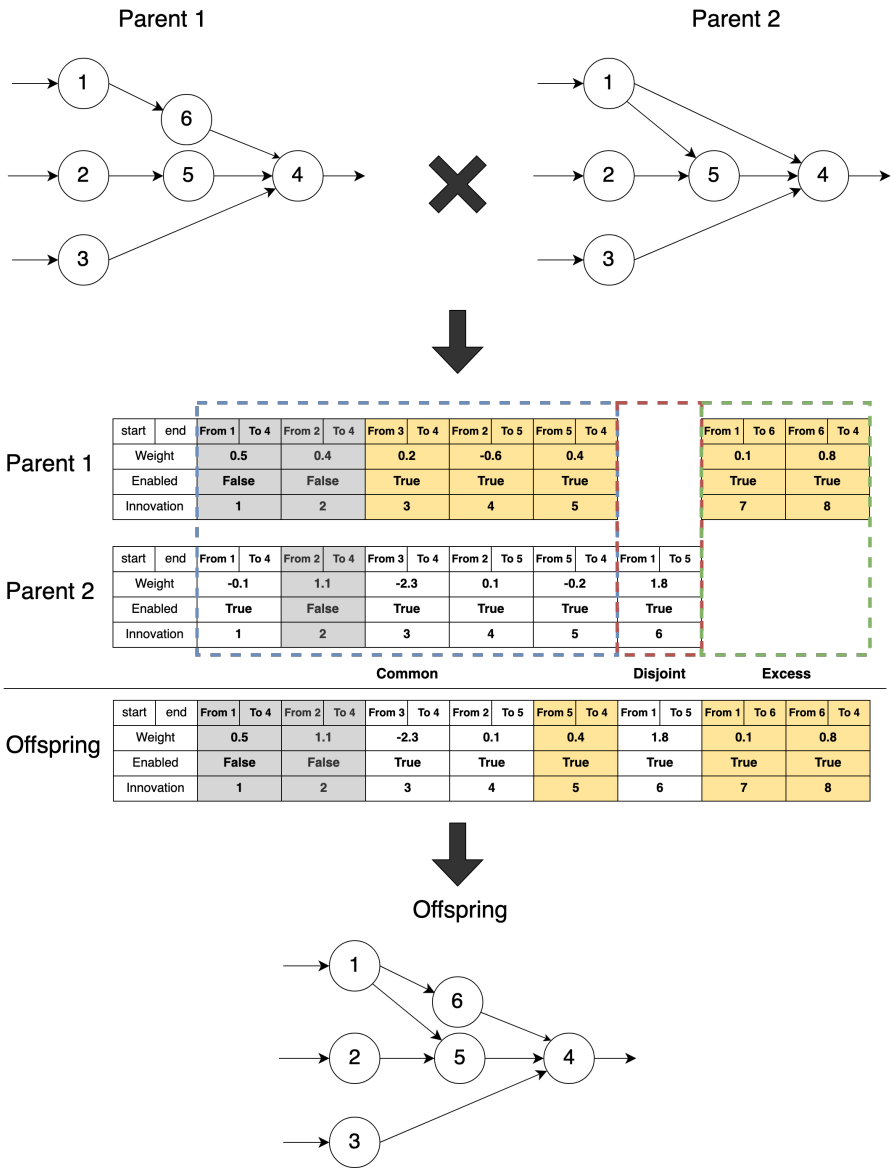


Figure 2.3: Illustration of recombination occurring in NEAT: First, connections are aligned based on innovation number and common, disjoint, and excess parts are identified. Common connections are copied randomly from parents. All the remaining connections are transferred from the most fit parent. Note that both parents are equally fit in this example (adapted from [117]).

explicit fitness sharing mechanism, in which individuals within the same species have their fitness divided by the number of individuals in that species. It prevents one species from taking over the entire population.

Chapter 3

Methods

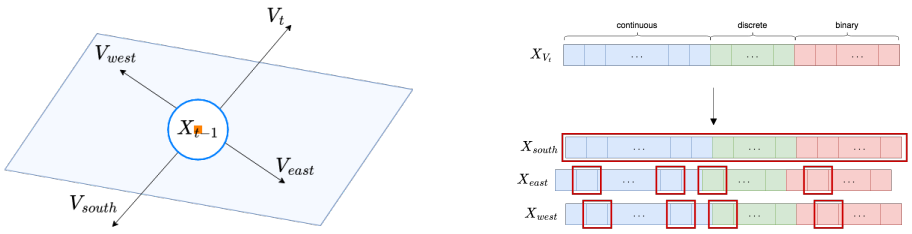
This chapter firstly addresses the proposal of two heuristics and their combination for a swarm intelligence algorithm based on a type of evolutionary swarm optimization, the aC-DEEPSO. Then, the fundamentals of the Multi-Objective Evolutionary Policy Search (MEPS) algorithm are described. The proposed MEPS algorithm integrates concepts from neuroevolution, multi-objective optimization, and policy search. Moreover, all the computational simulations were conducted using an Intel(R) Core(TM) i9-10900X CPU@3.70GHz and 64GB RAM, with Windows 10 Pro. The simulation code for aC-DEEPSO is implemented using Matlab R2020b. In addition, the code version for MEPS is based on [84] and is implemented in Python 3.10.

3.1 Adaptive C-DEEPSO with Local Search

When dealing with real-world optimization problems, algorithms have to be carefully designed to handle difficulties, such as the presence of several local minima in the search space or a high number of optimization variables, among others. In such cases, hybrid EAs - the combination of an evolutionary and a heuristic method - commonly perform better than EAs [35]. Therefore, this work shows two new mechanisms to enhance C-DEEPSO's search capability. The first is a local search (LS) operation that explores the awareness of each particle, prior to moving to the next position. The second operation changes the velocity update mechanism aiming at providing an additional exploration ability to each particle. This heuristic is inspired by the work using self-adaptive velocities in PSO for constrained optimization problems presented by [72], in which each particle considers its distance to another random particle, to reduce how far the particle's new position can deviate from the feasible region.

The first proposed procedure provides each particle in the swarm an awareness mechanism, in which they are allowed to look in three additional directions besides

the new V_t. These new directions are dubbed V_{south}, V_{east} and V_{west}. Hereof, V_{south} direction is obtained by inverting the direction of V_t. The V_{east} direction is calculated as a random vector with X_{t-1} as the origin, which lies in a randomly generated plane perpendicular to V_t passing through X_{t-1}. This plane is generated by choosing at random $d \leq D-1$ features from the original $D-$dimensional space. Similarly to V_{south} direction, V_{west} direction is generated by inverting the vector V_{east}. Figure 3.1 illustrates the local search mechanism. With the new velocities, four new positions for the particle are evaluated: $X_{south}, X_{east}, X_{west}$ and the position obtained by following V_t, namely X_{V_t}. Finally, the position and velocity that lead to the best fitness are assigned to X_t and V_t, respectively.



(a) (b)

Figure 3.1: Illustration of (a) the three search directions generated by the local search mechanism for one particle; and (b) the local search operator applied to a solution vector. In X_{south} all values are inverted. In X_{east} and X_{west}, only d values are modified plus one (see [81]).

This mechanism helps particles to search the whereabouts of their current position in the search space for better movements than the one calculated by the movement rule. By choosing a small d, each particle can search for a better movement in fewer dimensions than the original decision space. However, similarly to most local search methods, there is a trade-off between computational time and solution quality. As this local search mechanism requires four times more function evaluations, it can only be applied a predefined number of times and, preferably, at random iterations. The Algorithm 1 shows the local search mechanism. Note that the $\odot$ operator denotes an element-wise multiplication between vectors.

The second heuristic is an adaptive velocity heuristic based on the work presented in [72]. In this proposed heuristic, instead of adjusting velocity based on the distance to a random particle, it takes advantage of C-DEEPSO's memory mechanism. By doing this, a particle adjusts its velocity according to how far it is to the swarm of best solutions ever found. Thus, this heuristic modifies Equation (2.3) as follows,

$$V_t = w_I^* V' + w_A^* \cdot (X_{st} + F(X_r - X_{t-1})) + w_C^* \cdot C \cdot (X_{gb}^* - X_{t-1}), \qquad (3.1)$$

in which V' is calculated by means of Equation (3.2),

Algorithm 1: Local Search Mechanism

Input: $X_{t-1}, X_{lb}, X_{ub}, d$

Compute velocity V_t;
$V_{south} \leftarrow -1 \cdot V_t$;
$X_{plane} \leftarrow \{0\}^D$;
Sample $y_1, \ldots, y_d, y_{d+1} \sim \mathcal{U}(1, D)$;
For each $i \in (1, d)$ **do**
  $X_{plane}[y_i] \leftarrow \mathcal{U}(X_{lb}[y_i], X_{ub}[y_i])$;
end
$X_{rand} \leftarrow (X_{rand} - X_{t-1}) \odot V_t$;
if $V_t[y_{d+1}] > 0$ **then**
  $X_{plane}[y_{d+1}] \leftarrow X_{t-1}[y_{d+1}] - 1/V_t[y_{d+1}] \times$;
  $\sum_{y_i \neq y_{d+1}} X_{plane}[y_i]$;
end
else
  $X_{plane}[y_{d+1}] \leftarrow X_{t-1}[y_{d+1}] - \sum_{y_i \neq y_{d+1}} X_{plane}[y_i]$;
end
$V_t \leftarrow argmin_{V \in \{V_t, V_{south}, V_{east}, V_{west}\}} f(X_{t-1} + V)$;
$X_t \leftarrow X_{t-1} + V_t$;

$$V' = \begin{cases} V_{t-1}, & U(0,1) > \gamma \\ (X_m - X_{gb}) \odot sign(V_{t-1}), & \text{otherwise.} \end{cases} \tag{3.2}$$

In Equation (3.2), X_m is randomly sampled from *memory B*, X_{gb} is the particle with best fitness in memory and the operator $\odot$, as previously stated, denotes an element by element multiplication. The function $sign(\cdot)$ is a vector containing -1 and 1 in the components below zero and greater or equal to zero, respectively. Figure 3.2 illustrates the movement following C-DEEPSO's standard velocity, for $U(0,1) > \gamma$, and the movement following the proposed heuristic, for $U(0,1) \leq \gamma$. This modification enhances each particle with the ability to, with a probability given by γ, use its V_{t-1} direction with the magnitude given by the distance between the best particle in memory and a particle randomly sampled from memory. Associated with $S_g P_b - rnd$ memory strategy [76], particles are able to increase their velocity based on how far the best particle is from a random position sampled from both memory and population or, reversely, reduce velocity if both population and memory converged to a region in the search space, preventing the particle from jumping from the region and increasing exploitation. Thus, this newly proposed version is dubbed adaptive C-DEEPSO with Local Search (aC-DEEPSO). The resulting algorithm for aC-DEEPSO is presented in Algorithm 2.

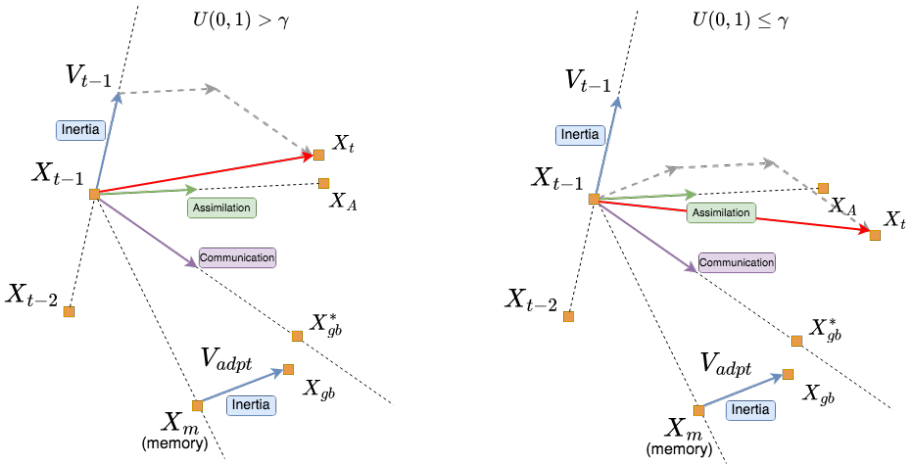


Figure 3.2: Illustration of the movement rule in calculating new particle position X_t using C-DEEPSO's standard movement rule (left) and the proposed adaptive velocity heuristic (right). The particle X_m is randomly sampled from *memory B*

.

Algorithm 2: Adaptive C-DEEPSO with Local search

Input: population size (NP), mutation rate τ, communication rate (P), memory size (MB), dimension (D), dimension (d), number of local search operator calls (n_{ls}), lower bounds (X_{lb}) and upper bounds (X_{ub}), adaptive velocity probability (γ)

Set the generation number $t = 0$;
Initialize the NP individuals in the population at random according to $\mathcal{U}(X_{lb}, X_{ub})$;
Evaluate the current population;
Update the global best X_{gb};
Sample n_{ls} generation numbers to apply local search operator and store in N_{ls};
while *stopping criterion is not satisfied* **do**
    For each *individual i in the population* NP **do**
        Calculate X_r using the strategy $S_g P_B - rnd$;
        Copy current individual X_{t-1};
        Mutate strategy parameters w_I, w_A, w_C and X^*_{gb};
        Apply movement rule in current individual X_{t-1}, according to Equations (3.1) and (3.2);
        if $t \in N_{ls}$ **then**
            $V_t \leftarrow LocalSearch(X_{t-1}, X_{lb}, X_{ub}, d)$;
        end
        Evaluate current individual X_t and its copy;
        Select the fittest individual to proceed to next generation;
        Update personal best individual;
    end
    Update memory MB;
    $t = t + 1$;
end

3.1.1 Computational Complexity

Let the population size, number of decision variables (dimension), number of decision variables used in the LS procedure, number of times the LS procedure is run, number of fitness evaluations, time complexity of fitness function be NP, D, d, N_{ls}, N_f, and $O(fitness)$, respectively. The time complexity of the algorithm (see Algorithm 2) is divided in two parts: whether the LS procedure is run or not.

To provide a clearer notation, let $N_{it} = \lceil N_f/NP \rceil$ denote the number of iterations the algorithm runs without the LS procedure and $O(recombination) = D \cdot NP$

denote the time complexity of the recombination operator. Thus, the time complexity of iterations running LS is $O(d \cdot NP \cdot 4 \cdot N_{ls} \cdot O(fitness) \cdot P \cdot O(recombination))$. The time complexity of iterations without LS is $O(D \cdot N_{it} \cdot NP \cdot O(fitness) \cdot P \cdot O(recombination))$. Finally, after ommiting the low-order terms, the total time complexity of aC-DEEPSO is $O(P \cdot O(recombination) \cdot NP \cdot O(fitness) \cdot [d \cdot N_{ls} + D \cdot N_{it}])$, which is polynomial in NP and D.

3.1.2 Preliminary Validation Simulations for Adaptive C-DEEPSO with LS

In order to validate the effectiveness of the proposed heuristics, a preliminary study is carried out on two multimodal benchmark functions from the literature: Rosenbrock and Schaffer. A multimodal function presents several local minima or maxima into which an EA can get trapped. The goal of this experiment is to verify the results obtained by C-DEEPSO with adaptive velocity and local search compared to the standard versions of both PSO and C-DEEPSO. After fine-tuning parameters using irace [71], both versions of the C-DEEPSO were initialized using 0.6 and 0.9 for communication and mutation rates, respectively. The local search is empirically set to execute at 7 random iterations with $\gamma = 0.7$. Moreover, to evaluate different levels of difficulty, each algorithm was run 30 times for a total of 10^5 fitness evaluations with a population of 50 particles in dimensions 30, 50, and 100. The following equations (3.3) and (3.4) show the benchmark functions:

- Rosenbrock function - Multimodal (D $>$ 2) - Goal = 0,

$$f(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]. \tag{3.3}$$

- Schaffer function - Multimodal - Goal = 0,

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 + x_2^2) - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2}. \tag{3.4}$$

Table 3.2 shows the results of standard PSO, standard C-DEEPSO and the proposed adaptive C-DEEPSO with local search in the Schaffer function. The results indicate that although standard C-DEEPSO obtained a competitive performance compared to PSO, neither algorithm consistently achieved the global minimum value. On the other hand, adaptive C-DEEPSO with LS was able to find, in all the evaluated dimensions, the global optima more often, which led to obtaining a lower mean objective function value. Regarding the median objective function value, adaptive C-DEEPSO with LS obtained the global minimum value in all dimensions.

Rosenbrock is considered a difficult non-convex function. Table 3.1 shows that the Rosenbrock function was the most difficult benchmark problem for PSO. Standard C-DEEPSO nonetheless achieved better results for both mean and median objective function values than PSO. These good results are improved by employing the proposed operators in adaptive C-DEEPSO with LS, lowering both mean and median values in all cases.

Table 3.1: Summary of the obtained results on Rosenbrock benchmark function.

		Rosenbrock				
D	Algorithm	Mean	Std.	Best	Median	Worst
30	PSO	82.020	17.378	51.879	79.651	130.280
	C-DEEPSO	25.385	1.214	24.212	25.072	28.727
	aC-DEEPSO	23.553	1.300	22.509	23.131	28.789
50	PSO	162.225	27.537	122.430	156.555	234.040
	C-DEEPSO	46.191	1.357	44.601	45.636	48.638
	aC-DEEPSO	44.421	1.609	42.908	43.953	48.515
100	PSO	398.752	50.450	286.640	398.765	534.840
	C-DEEPSO	97.484	1.145	95.003	98.180	98.439
	aC-DEEPSO	96.365	1.629	92.990	96.554	98.276

Table 3.2: Summary of the obtained results on Schaffer benchmark function.

		Schaffer				
D	Algorithm	Mean	Std.	Best	Median	Worst
30	PSO	11.115	0.786	9.600	11.073	12.698
	C-DEEPSO	1.566	1.379	0.000	1.542	5.290
	aC-DEEPSO	0.216	0.510	0.000	0.000	2.293
50	PSO	20.368	0.978	18.523	20.284	22.051
	C-DEEPSO	8.003	4.160	0.000	9.383	14.172
	aC-DEEPSO	1.796	3.370	0.000	0.000	11.687
100	PSO	42.881	1.111	39.413	43.121	44.639
	C-DEEPSO	19.708	13.646	0.000	27.437	35.968
	aC-DEEPSO	2.564	6.712	0.000	0.000	27.563

The mean and standard deviation values are preliminary measures that are often not sufficient to provide an effective analysis of the obtained results. Hence, a statistical protocol based on [78] and [82] is employed. The Kruskal-Wallis test [59] is performed, aiming to find possible differences among the mean objective function values. Using a significance value of 5%, a p-value below 0.05 is found indicating that there is a difference among the means. Accordingly, the Wilcoxon signed-rank test [24] with Holm-Bonferroni correction [50] is carried out to perform a pairwise analysis to identify the differences between the samples analyzed. For a null hypothesis of equality of means, a p-value less than 0.05 indicates that the null hypothesis can be rejected with 5% significance. Table 3.3 provides the results of the statistical test indicating whether aC-DEEPSO is different or not when compared

to standard PSO and C-DEEPSO. According to the results, aC-DEEPSO presents the lowest values among the compared algorithms in both benchmark functions.

Table 3.3: Wilcoxon signed rank test results for both benchmark functions.

	P-value 0.05		Winner
	PSO x aC-DEEPSO	C-DEEPSO x aC-DEEPSO	
Schaffer 30D	True	True	aC-DEEPSO
Schaffer 50D	True	True	aC-DEEPSO
Schaffer 100D	True	True	aC-DEEPSO
Rosenbrock 30D	True	True	aC-DEEPSO
Rosenbrock 50D	True	True	aC-DEEPSO
Rosenbrock 100D	True	True	aC-DEEPSO

3.2 The Multi-Objective Evolutionary Policy Search

The Multi-objective Evolutionary Policy Search (MEPS) algorithm proposed in this study is a model-free algorithm that estimates action-preference values in MORL problems. MEPS falls in the RL "actor-only" family of algorithms and inherits NEAT structure to evolve artificial neural networks (ANNs) that implement deterministic policies in MORL environments. First, an initial random population P_t (for time $t = 0$) of n_p ANNs with one output node for each possible action is created. At each generation, individuals are evaluated according to a multi-objective reward function over h episodes, with $\mathbf{r}_h$ indicating the accumulated reward of each individual. Thereafter, the accumulated reward is used to sort the networks in population P_t by means of non-dominated sorting and a density measure.

Specifically, the ANNs employed in MEPS are designed to output action-preference values $p(s, a)$ for each available action a, given a state s as input. Moreover, to ensure that the agent deterministically follows the policy, the actions are selected in a greedy manner. Thus, Figure 3.3 shows an example of an agent at state s_t selecting an action, and transitioning to state s_{t+1}.

The density measures considered are crowding distance (CD) [31] and hypervolume contribution (HVC) [36]. Crowding distance is defined as infinity for extremal solutions, and as the sum of side lengths of the cuboid that touches adjacent solutions in the case of a non-extremal solution on the Pareto front. It is meant to distribute solution points uniformly on the Pareto front. In contrast to this, the hypervolume contribution measure assigns a value to each solution according to its contribution to the hypervolume of the Pareto front. Consequently, it is meant to distribute them in a way that maximizes the covered hypervolume, focusing on knee-points without losing extremal points of the Pareto front.

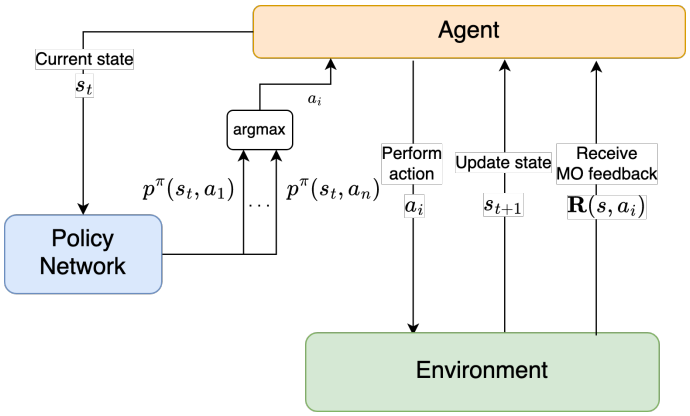


Figure 3.3: Example of the agent's interaction with the environment, transitioning from state s_t to state s_{t+1} after selecting the action associated with the highest preference value $p(s, a)$.

Afterwards, a K_t set of n_p parents are randomly selected from the population P_t using a binary crowded tournament selection [31]. Note that if the density measure used is the hypervolume contribution, the density value of an individual x_i used in the tournament selection is $1/HVC(x_i)$. Thereafter, an offspring population Λ_t is generated by cloning selected parents and applying two types of mutation: structural and parametrical mutations. Structural mutations occur with a pre-defined probability and comprise (1) adding a new connection to previously unconnected nodes, and (2) adding a new hidden node.

It is important to note that MEPS provides ReLU-based-feedforward ANNs, hence, no recurrent connections are allowed. Parametrical mutation encompasses updating connection weights and biases by adding a Gaussian noise with zero mean and standard deviation given by a parameter σ. Despite the performance of the crossover operator in the ablation studies presented in [117], it is not guaranteed to generate a chromosome that preserves the good characteristics of the parents regarding the quality of the solution. As a result, the crossover operator disturbs NEAT's search ability, as attested in [109]. Therefore, the crossover operator has not been employed in MEPS.

The next generation population P_{t+1} is composed of the survivors selected from population $R_t = P_t \cup \Lambda_t$ of size $2n_p$. Differently from NEAT original proposal, MEPS does not utilize any speciation mechanism. Therefore, there are two possible approaches to perform survival selection. The first approach is similar to the NSGA-II [31] survival selection mechanism, in which the population R_t is sorted by means of non-dominated sorting in fronts or ranks, and then selected to the next generation iteratively by front. If the size of a front is bigger than the available slots in the next generation, the front is sorted in descending order according to the selected density measure, and the individuals from less dense regions are selected. Figure 3.4

illustrates this approach.

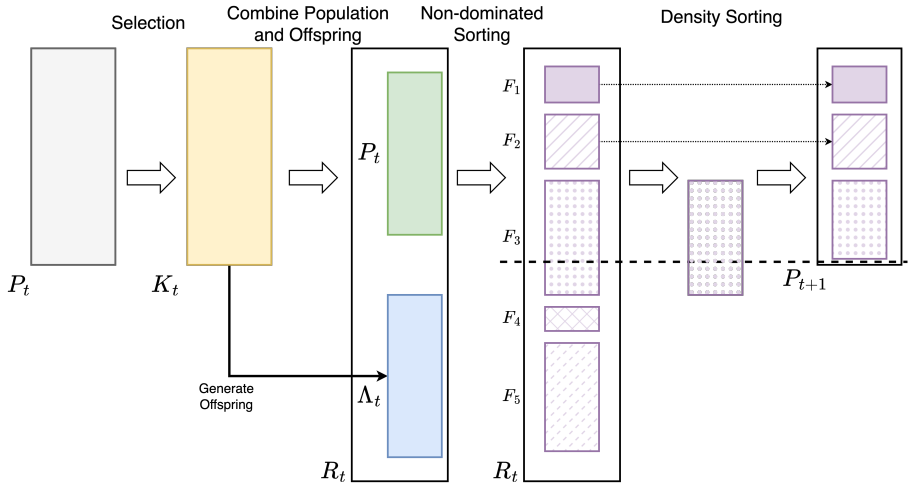


Figure 3.4: Standard MEPS flowchart of the evolution from generation t to generation $t+1$

The second approach comprises adaptively restricting the number of selected survivors from each front. From a multi-objective optimization perspective, in early generations, some undesired non-dominated individuals are selected over individuals from other fronts [146]. Figure 3.5 shows a situation in which some ignored dominated solutions are potential candidates to improve the Pareto front. Likewise, from a neuroevolutionary perspective, discarded potential solutions may indicate newly introduced topologies, which are prematurely extincted. Thus, this work proposes a novel selection method that allows some individuals from higher ranks to survive. Specifically, a heavy-tailed Pareto distribution [7] is used for this purpose. The maximum number of survivors for each i-th front is given by

$$n_{F_i} = \begin{cases} \frac{\alpha/i^{\alpha+1}}{C} \cdot (n_p - \lceil n_p \cdot ratio \rceil), & i > 1 \\ \lceil n_p \cdot ratio \rceil, & i = 1, \end{cases} \tag{3.5}$$

in which $C = \sum_{i>1}^{K} \alpha/i^{\alpha+1}$ for K non-dominated fronts, and $ratio$ denotes the fraction of individuals selected from the first front. The parameter α determines how heavy the distribution's tail is, as presented in Figure 3.6. To avoid both premature convergence to an incomplete Pareto front and extinction of potential topology innovations, the $ratio$ is set to increase with the number of generations in a relationship defined by Equation 3.6

$$ratio = \begin{cases} 1, & t_{max} > t > t_r \\ \psi + t_r \cdot \frac{(1-\psi)}{t}, & t \leq t_r, \end{cases} \tag{3.6}$$

where $\psi \in (0, 1)$ stands for the initial fraction of non-dominated individuals selected,

which is increased over generations. This definition of *ratio* allows the algorithm to gradually decrease the exploration of solutions from all non-dominated fronts. Moreover, after the predefined t_r generations, the heavy tail selection mechanism is replaced by the approach shown in Figure 3.4.

Despite the fact that Equation 3.5 denotes the maximum number n_{F_i} of allowed survivors in each front i, there may not exist all the n_{F_i} individuals in the i-th front. To handle this problem, the procedure is started from the first front and, if $|front_i| < n_{F_i}$, the remaining $n_{F_i} - |front_i|$ slots are added to the $n_{f_{i+1}}$ allowed survivors of front $i + 1$. This procedure is repeated until all fronts are processed. Although unlikely, there could be situations in which there are still some slots left at the end. In such cases, the procedure is repeated from the beginning with the remaining individuals from each front until filling the remaining slots.

MEPS uses a memory of the same size as the population to store the best individuals ever found throughout generations. The memory is updated after the next generation population P_{t+1} selection. Let M be the memory population, the memory update mechanism, inspired by [79], can be summarized as follows:

(1) **Generate** a temporary population $M_{temp} = M \cup P_{t+1}$.

(2) **Clear** memory M.

(3) **Sort** the temporary population M_{temp} into non-dominated fronts.

(4) **If** the first front size is bigger than n_p, sort based on the density measure employed.

(5) **Assign** individuals from the sorted front to M until memory is full.

Algorithm 3: Memory update mechanism

Input: Population (P_{t+1}), Memory (M), density measure S
Output: Updated memory M

$M_{temp} \leftarrow M \cup P_{t+1}$;
$M \leftarrow \emptyset$;
Sort population in non-dominated fronts $F_1, \ldots, F_k$ according to Pareto Dominance;
if $|F_1| > n_p$ **then**
    Sort F_1 based on the density measure S;
end
$i \leftarrow 0$;
while $|M| < n_p$ **do**
    Assign the i-th individual from F_1 to M;
    $i \leftarrow i + 1$;
end
return M

Finally, MEPS avoids performing gradient updates and computing value function or q-value function estimators as it evolves network policies in an evolutionary

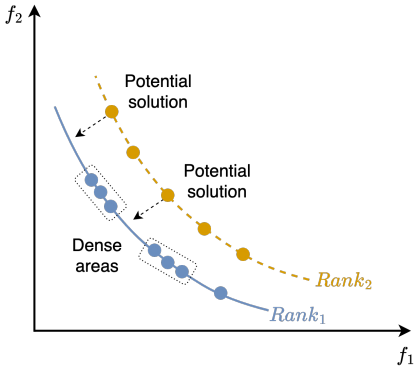


Figure 3.5: Discarded potential solutions due to dense regions of non-dominated solutions.

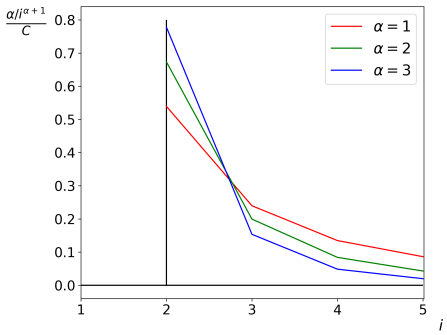


Figure 3.6: Pareto distribution illustration for different values of α and four non-dominated fronts. X-axis and Y-axis denote the non-dominated fronts and the proportion of the population which will be selected from each front, respectively.

manner. Additionally, MEPS is classified as a multi-policy algorithm, as it leverages population-based techniques and produces a set of Pareto-optimal policies [47]. Algorithm 4 shows the pseudocode for the proposed method. In addition, Tables 3.4 and 3.5 summarizes a description of the MEPS hyperparameters and problem-specific parameters used, respectively.

In order to provide a clear distinction among MEPS versions, the different configurations of MEPS are identified based on both the survivor selection method and the density measure used. The survivor selection method can be either based on heavy tail selection (H1) or based on non-dominance sorting (H0). The density measure can be crowding distance (S0) or hypervolume contribution (S1). In this way, $4(2 \cdot 2)$ versions of MEPS are proposed and analyzed. As an example, one possible setting of MEPS is H1/S1, in which the proposed heavy tail survivor selection mechanism is used along with hypervolume contribution as the density measure.

Table 3.4: MEPS hyperparameter description

Parameter	Description
n_p	Population size
ψ	Initial fraction selected from first front
t_{max}	Total generations
t_r	Final generation of the heavy tail survivor selection
α	Heavy tail selection parameter
n_h	Number of initial hidden nodes
p_{ac}	"Add connection" mutation probability
p_{an}	"Add node" mutation probability
σ	Parametrical mutation standard deviation

Table 3.5: Problem-specific parameter description

Parameter	Description
S	Density measure function
n_i	Number of input nodes
n_o	Number of output nodes
HT	Indicate the use or not of the heavy tail survivor selection
h	Length of the episode to evaluate the agent

Algorithm 4: The Multi-Objective Evolutionary Policy Search algorithm (MEPS)

Input: n_p, ψ, t_{max}, t_r, α, σ, p_{ac}, p_{an}, n_i, n_o, n_h, S, HT, h
Output: Memory M

$t \leftarrow 0$;
Initialize population P_t with fully connected ANNs containing n_i input nodes, n_h hidden nodes and n_o output nodes;
Evaluate each individual of P_t for an episode of length h;
while $t < t_{max}$ **do**
    Generate a population K_t of n_p parents selected from the population P_t using binary crowded tournament selection and the selected density measure S;
    Generate Λ_t offspring population;
    Evaluate each individual of Λ_t for an episode of length h;
    $R_t \leftarrow P_t \cup \Lambda_t$;
    Sort population R_t in non-dominated fronts $F_1, \ldots, F_k$;
    $P_{t+1} \leftarrow \emptyset$;
    if $t \leq t_r$ *OR* $HT > 0$ **then**
        `/* runs heavy tail selection */`
        Calculate *ratio* according to Equation 3.6;
        Calculate maximum number of survivors $n_{F_1}, \ldots, n_{F_k}$ for each front using parameters α and ψ;
        $remaining \leftarrow 0$;
        For each *Front* F_i **do**
            $n_{F_i} \leftarrow n_{F_i} + remaining$;
            if $n_{F_i} < |F_i|$ *OR* $\min(|F_i|, n_{F_i}) > n_p - |P_{t+1}|$ **then**
                Sort front based on density measure function S;
                Add the $\min(|F_i|, n_{F_i}, n_p - |P_{t+1}|)$ fittest individuals from F_i to P_{t+1};
            end
            else
                $P_{t+1} \leftarrow P_{t+1} \cup F_i$;
            end
            if $n_p = |P_{t+1}|$ **then**
                break;
            end
            $remaining \leftarrow \max(n_{F_i} - \min(|F_i|, n_p - |P_{t+1}|), 0)$;
        end
    end
    else
        For each *Front* F_i **do**
            if $|F_i| > n_p - |P_{t+1}|$ **then**
                Sort front based on density measure function S;
                Add the $n_p - |P_{t+1}|$ fittest individuals from F_i to P_{t+1};
            end
            else
                $P_{t+1} \leftarrow P_{t+1} \cup F_i$;
            end
            if $n_p = |P_{t+1}|$ **then**
                break;
            end
        end
    end
    Update memory M following memory update procedure;
    $t \leftarrow t + 1$;
end

3.2.1 Computational Complexity

We then evaluate the time complexity of MEPS as shown in Algorithm 4. At each generation, MEPS relies on executing the non-dominated sorting and, either the binary crowded tournament selection operator or the hypervolume contribution operator. Let m be the number of objectives, the time complexity for the non-dominated sorting and the crowding distance selection operator are $O(m \cdot n_p^2)$ and $O(m \cdot n_p \log n_p)$, respectively. The time complexity of the hypervolume contribution operator is determined by the calculation of the hypervolume for each individual from the population. This procedure is executed with a time complexity of $O(n_p^3 \cdot m^2)$, as stated by Emmerich et al. [36].

Thus, the time complexity of MEPS varies according to the density measure employed. When using the crowding distance, the binary crowded tournament selection operator in Step (5) involves running the non-dominated sorting for all fronts and sorting based on the density measure once. Thus, this step is governed by the non-dominated sorting, which is performed with a time complexity $O(m \cdot n_p^2)$. On the other hand, this Step is executed with a time complexity of $O(n_p^3 \cdot m^2)$ in the MEPS counterpart that employs the hypervolume contribution.

In Step (6), the offspring generation is independent of the chosen density measure. For each individual in the population, its nodes and connections are iterated for the mutation procedure. Therefore, it is executed with time complexity $O(n_p \cdot (TC + TN))$, in which TC and TN are the total maximum number of connections and nodes, respectively. Additionally, the fitness evaluation, which is also independent of the density measure, is performed in Steps (3) and (7) with time complexity $O(h \cdot |A|)$. Then the sorting is done in Step (9) with time complexity $O(m \cdot (2n_p)^2)$.

In Steps (10)-(29) and (31)-(42), the worst-case scenario comprises sorting only one front with size $2n_p$. Hence, the time complexity for these Steps is $O(m \cdot (2n_p) \log(2n_p))$ and $O(2n_p^3 \cdot m^2)$ for the versions using crowding distance and hypervolume contribution, respectively. Later in Step (44), the memory update mechanism, presented in Algorithm 3, initially involves performing the non-dominated sorting for all fronts. In the worst-case scenario that the current population and memory are non-dominated, sorting is performed based on the selected density measure. Hence, the time complexity of this Step depends directly on the density measure. If the selected measure is the crowding distance, this Step is similar to Step (5), with a time complexity of $O(m \cdot n_p^2)$. In the MEPS version with hypervolume contribution, this Step is governed by the hypervolume contribution calculation, which runs with time complexity $O(n_p^3 \cdot m^2)$.

Finally, after omitting low-order terms, the MEPS version that employs crowding distance is governed by the non-dominated sorting component of the algorithm

and runs with time complexity $O(t_{max} \cdot m \cdot n_p^2)$. The MEPS counterpart using the hypervolume contribution is governed by hypervolume computation, which is executed with time complexity $O(n_p^3 \cdot m^2)$ [36]. As a result, both MEPS versions are polynomial in the population size n_p. A full version of the MEPS code is available at `https://github.com/gabrielmatos26/MEPS-Paper`.

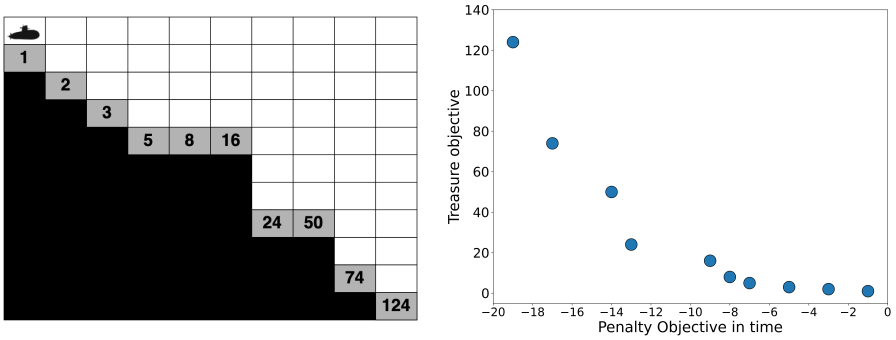
3.2.2 Preliminary Validation Simulations for MEPS

The proposed versions of MEPS are compared with three state-of-the-art MORL algorithms, Pareto Q-Learning (PQL) [133], Q-Managed (QM) [29] and Multi-Policy Soft Actor Critic (MPSAC) [19]. For a fair comparison, as Q-Managed was initially proposed only for 2-objective problems, we did not analyze its performance on environments with 3 objectives. Five MORL benchmark environments with 2 and 3 objectives have been selected to evaluate the proposal for an MORL algorithm, namely Deep Sea Treasure [130], Pressurized Bountiful Sea Treasure [133], Modified Bountiful Sea Treasure [29], Space Exploration [131], and Bonus World [131]. Besides, a sixth novel MORL environment is proposed and used to validate the MEPS proposal.

Deep Sea Treasure (DST) is an episodic deterministic environment in which an agent controls a submarine searching for an undersea treasure. The environment is a rectangular grid with 11 rows and 10 columns (Figure 3.7a), containing 10 treasures of varying values at different locations. Each episode starts with the submarine at the top left corner and ends if either a treasure is found or h actions were taken. The treasures are arranged to have the lowest treasure value at the closest location to the starting point and the highest value at the furthest location, which means the treasure value is inversely proportional to its distance from the source. The agent can move around the environment by performing four available actions, representing the four cardinal directions - (1) right, (2) left, (3) down, or (4) up. Each action taken by the agent incurs a time penalty[1], which is a -1 decrease applied to the time penalty objective. Note that, this time penalty objective is not a time unit, but is the sum of penalties the agent receives for the time it takes in interacting with the environment. Additionally, the agent's goals are to minimize the time penalty received in finding a treasure while maximizing its value. The reward vector $\mathbf{r}_h \in \mathbb{R}^2$ consists in the time penalty as first element and the treasure value as second element. The true Pareto front (Figure 3.7b) is globally concave with some local concavities in the second, fourth and sixth points from left to right.

A variation of the DST is the Modified Bountiful Sea Treasure (MBST) environment. This is a 2-objective problem in which not only the treasure values but

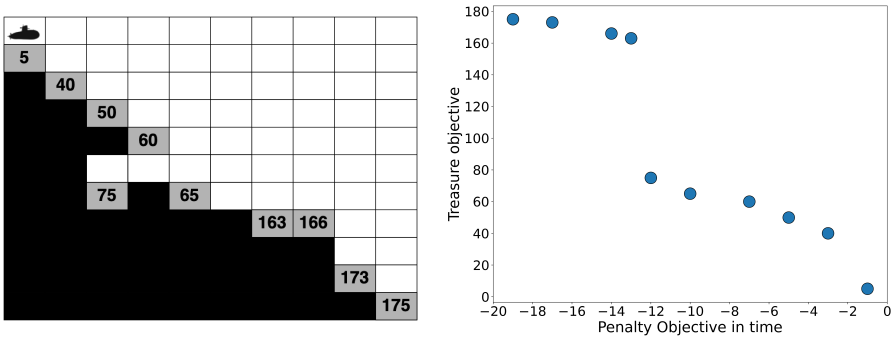
[1]In order to maintain the same description for penalty objective in time from [29], it is presented as a negative time penalty in the Pareto fronts.



(a) Environment (b) Pareto Front

Figure 3.7: Deep Sea Treasure (DST) benchmark

also their location are modified (Figure 3.8a). To find all the Pareto optimal policies, the agent must learn how to enter in a tunnel between two locations, which makes learning more challenging. With this change, the Pareto front (Figure 3.8b) is divided in one convex part and one non-convex part.

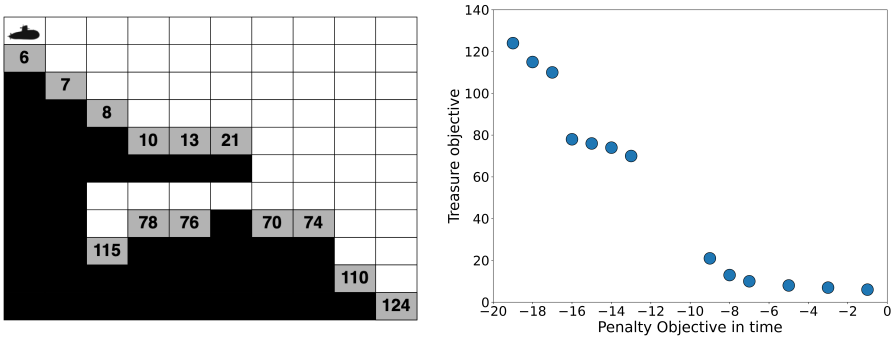


(a) Environment (b) Pareto Front

Figure 3.8: Modified Bountiful Sea Treasure (MBST) benchmark

Afterwards, a new variation of the DST is proposed, dubbed Discontinuous DST (DDST). Inspired by the ZDT set of MOO benchmark functions [150], specifically ZDT3, both treasure values and treasure locations are modified to create two discontinuities in the Pareto front. Furthermore, to increase difficulty in learning, DDST includes a tunnel containing three treasures (Figure 3.9a). To discover all the Pareto optimal policies, the agent needs to decide not to collect the closest reward to the tunnel entrance but, instead, to enter the tunnel. The true Pareto front (Figure 3.9b) is also divided into three parts, mixing convex and non-convex parts.

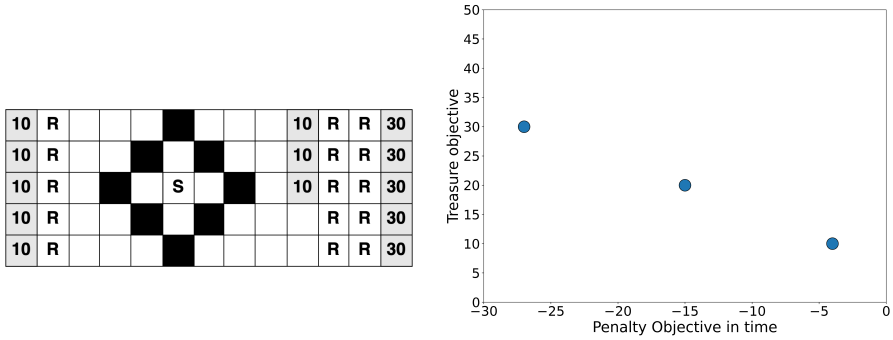
Another 2-objective benchmark environment is the Space Exploration (SE) (Figure 3.10a), in which the agent controls a spaceship that starts the episode in the location marked 'S' with the goal of discovering a habitable planet while minimising the radiation it is exposed to during the search. The first objective is the radiation penalty. After every action, the radiation objective is penalized by -1, or -11 if the next state is marked 'R'. The second objective is the mission level of success, which



(a) Environment (b) Pareto Front

Figure 3.9: Discontinuous Deep Sea Treasure (DDST) benchmark

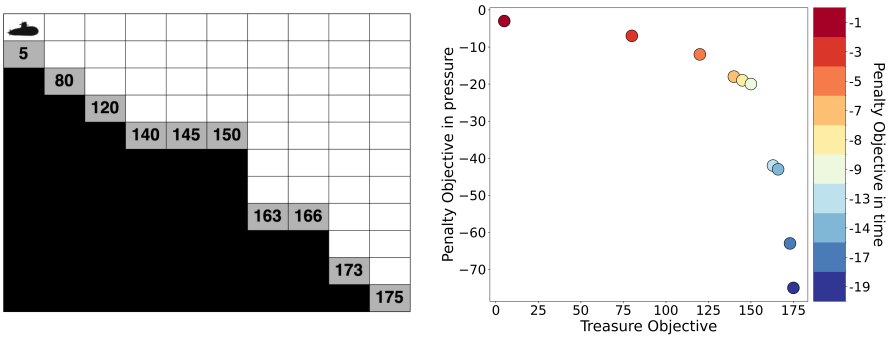
denotes the habitability of the planet, or -100 if the agent moves to a black cell, representing an asteroid. The episode ends after the agent performs h actions, or reaches a planet, or collides with an asteroid. Unlike the previous environments, in SE the agent is also allowed to move diagonally, totaling eight directions. Additionally, a movement that would lead the agent out of the grid, takes it to the opposite edge of the grid. For example, if the agent moves downwards from the bottom row of the grid, the next state will be the top row of the grid, maintaining the column. Thus, in this environment, every action leads to a state change. The Pareto front is illustrated in Figure 3.10b.



(a) Environment (b) Pareto Front

Figure 3.10: Space Exploration (SE) benchmark

Pressurized Bountiful Sea Treasure (PBST) Environment (Figure 3.11a) is a variation of DST in which a third objective representing the pressure penalty is included. Similarly to penalty in time, this objective denotes the pressure penalty received by the agent for staying underwater at the depth indicated by the row, with initial value equal to -1. For example, if the agent moves down from starting point, it receives a -2 pressure penalty for row 2 and, therefore the total pressure penalty is -3. The first and second objectives remain the same as in the DST environment. Moreover, the treasure values are modified to create a Pareto front that is globally convex. The set of Pareto optimal policies is presented in Figure 3.11b.

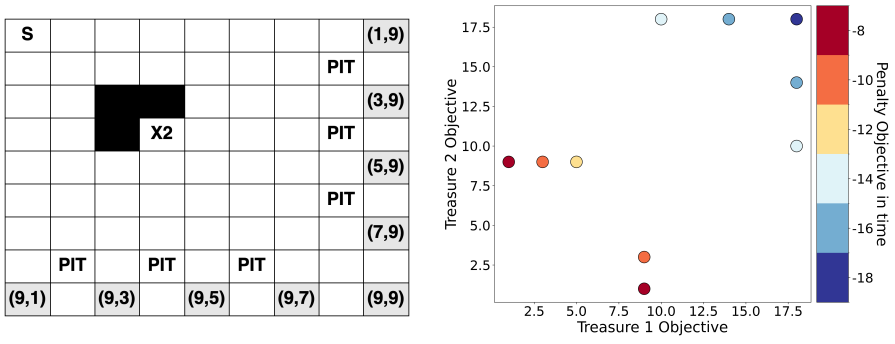


(a) Environment

(b) Two-dimensional projection of Pareto Front with colors indicating Pressure objective

Figure 3.11: Pressurized Bountiful Sea Treasure (PBST) benchmark

Bonus world (BW, Figure 3.12a) is a 3-objective environment similar to DST, in which the agent starts at cell marked 'S' and is allowed to move in the four cardinal directions with black cells indicating walls the agent cannot pass through. At every step, a time penalty of -1 is applied to the first objective. The gray cells correspond to terminal states that reward the agent with values corresponding to treasure 1 and treasure 2. If the agent reaches a cell marked 'X2', a bonus is activated and the values for treasure 1 and treasure 2 are doubled. Contrarily, if the agent enters a cell marked 'PIT', it not only loses the bonus but is also moved to the start state. Although not all the Pareto optimal policies require the agent to activate the bonus, some Pareto optimal policies are only reachable with the bonus activated. Therefore, this is a difficult environment because the agent must learn to both avoid penalties and activate the bonus before moving to some terminal states. The set of Pareto optimal policies is show in Figure 3.12b.



(a) Environment

(b) Two-dimensional projection of Pareto Front with colors indicating Time objective

Figure 3.12: Bonus World (BW) benchmark

The metric employed to evaluate the performance of each algorithm is provided by the hypervolume indicator (HV) on the multiple policies' accumulated rewards obtained by the learning agents at the end of the pre-defined number of generations

[130]. Accordingly, the obtained HVs are compared to the HV of the true Pareto front in each benchmark environment. For the Bonus World and Space Exploration environments, the choice of reference point was based on choosing a point that is worse than the nadir point. In the Deep Sea Treasure and its variations, the reference points used are the same as in [133]. Moreover, as a measure of network complexity, MEPS and MPSAC are compared in terms of their average number of network nodes and connections.

In order to guarantee a fair comparison, each algorithm was executed independently 20 times in every problem. Since the main goal of this experiment is to validate the proposed approach, no fine-tuning of the parameters has been done. Therefore, PQL and QM were run with the hyperparameter values found in the literature. Also, MPSAC execution was divided in two stages. The hyperparameters used throughout executions are presented in Table 3.6 and initial topology used in MEPS is presented in Figure 3.13. The output layer for both MEPS and MPSAC contains as many nodes as the number of available actions in the evaluated benchmark. Moreover, the different configurations of MEPS are identified based on both the survivors selection method and the density measure used. The survivors selection method can be based on either heavy tail selection (H1) or non-dominance sorting (H0). The density measure can be crowding distance (S0) or hypervolume contribution (S1). In this way, $4(2 \cdot 2)$ versions of MEPS have been analyzed. As an example, one possible setting of MEPS is H1/S1, in which the proposed heavy tail survivor selection mechanism is used along with hypervolume contribution as the density measure.

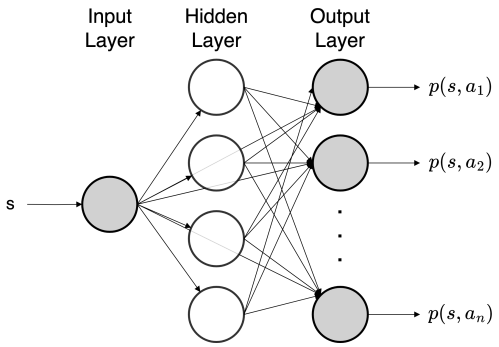


Figure 3.13: MEPS initial topology configuration for benchmark tests.

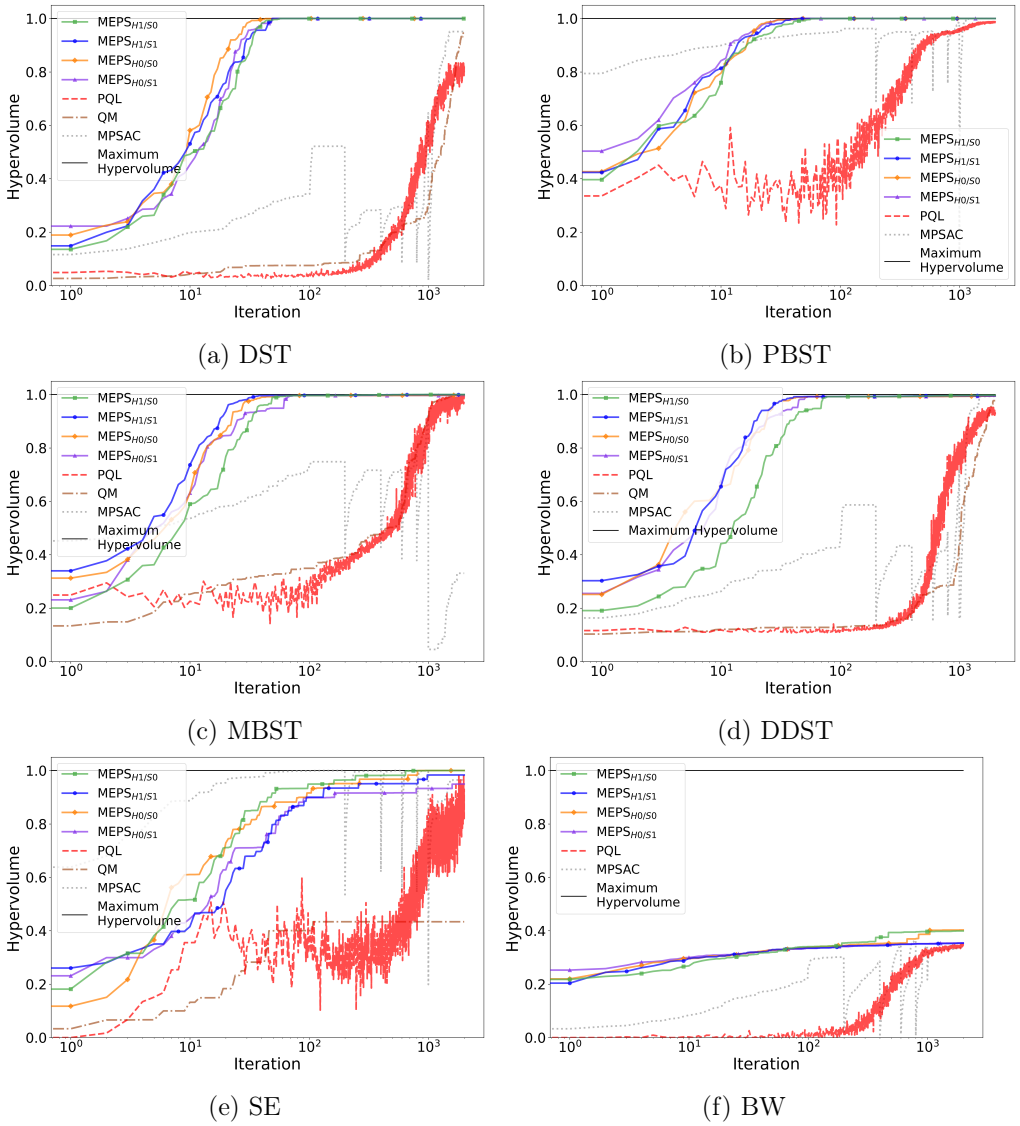
Table 3.6: Parameter initialization values used by the algorithms in benchmark tests

Description	MEPS	PQL	QM	MPSAC
Population size	50	-	50	50
Activation function	relu	-	-	relu
Initial fraction selected from first front	0.5	-	-	-
End generation number of heavy tail survivor selection	1000	-	-	-
Heavy tail selection parameter	1.0	-	-	-
Number of input nodes	1	-	-	1
Number of initial hidden nodes	4	-	-	64
Number of output nodes	4 or 8	-	-	4 or 8
Add connection mutation probability	0.2	-	-	-
Add node mutation probability	0.2	-	-	-
Parametrical mutation standard deviation	0.5	-	-	1.0
Learning rate	-	-	-	0.001
Gamma	-	-	-	0.99
Generations	2000	2000	2000	1000 + 1000
Episode length			20	

Detailed benchmark results based on hypervolume indicator are shown in Table 3.7 and the algorithm with superior performance is indicated in bold for each problem. The normalized average hypervolume obtained by each algorithm in each benchmark environment is shown in Figure 3.14. In both the DST (Figure 3.14a) and PBST (Figure 3.14b) problems, all the four versions of MEPS are able to find the entire Pareto front in less than half the number of maximum generations with similar topologies. The final networks evolved using MEPS are composed by an average of 25 connections and 10 (1 input, 5 hidden, and 4 output) nodes. PQL, QM and MPSAC are not able to consistently find all the non-dominated policies in DST, only achieving the maximum hypervolume value in some executions. However, MPSAC is able to find all the non-dominated policies in all the 20 runs in PBST. It is worth noting that, due to the simpler Pareto front of these benchmarks compared to the other benchmarks, the use of the proposed heavy tail survivor selection operator (H1/S0 and H1/S1) results in a delay in the search. Hence, both H0/S0 and H0/S1 achieve the maximum hypervolume in fewer generations.

In the MBST problem (Figure 3.14c), with the exception of MPSAC, all the algorithms obtained all the Pareto optimal solutions in at least one execution. The effectiveness of the proposed survivor selection method is demonstrated with H1/S0 and H1/S1 achieving the highest mean hypervolume values. Specifically, H1/S0 can find all the Pareto optimal solutions in all the executions. Regarding the topology of the final networks, H1/S0 networks provide an average of 26 connections and 10 (1 input, 5 hidden, and 4 output) nodes. By increasing the difficulty, as in the DDST problem (Figure 3.14d), the MEPS stands out as an advantageous algorithm, as PQL, QM and MPSAC cannot find all the Pareto optimal solutions in any execution. Among the MEPS configurations, H1/S0 achieves the highest mean

hypervolume with less complex networks, with an average of 28 connections and 12 nodes. Therefore, it can be said that H1/S0 achieved the highest mean hypervolumes in the DST problem and its variations, providing a population of solutions comprising less complex networks.



(a) DST (b) PBST

(c) MBST (d) DDST

(e) SE (f) BW

Figure 3.14: Normalized average hypervolumes obtained by PQL, QM, MPSAC, and MEPS versions in all the MORL benchmark environments.

Furthermore, in both the SE (Figure 3.14e) and BW (Figure 3.14f) problems, the MEPS obtained the highest average hypervolumes, which indicates a better approximation of the true Pareto front. Similarly to the DST and PBST results, the use of the heavy tail survivor selection operator in a problem with a Pareto front without any discontinuities, such as SE, worsened the results. The configuration H0/S0 consistently found the entire Pareto front, while the use of hypervolume contribution as the density measure achieved the worst results among MEPS versions. Moreover, MEPS versions using crowding distance, H0/S0 and H1/S0, obtained networks with

a smaller average number of connections and nodes, 52 and 18, respectively. It is worth mentioning that, despite the simple Pareto front (only three solutions) in the SE problem, the increased action space poses a more difficult problem in searching for the optimal policies. This is indicated by results showing that six out of seven algorithms struggled to obtain all the Pareto optimal solutions across runs. Unlike previous benchmark results, BW stands as the most difficult benchmark problem, in which none of the algorithms were able to approximate the Pareto front. H1/S0 nonetheless achieved the highest average hypervolume among the algorithms, with an average of only 7 hidden nodes (12 nodes in total) and 28 connections.

Table 3.7: Hypervolume analysis for each MORL benchmark environment (labeled as Env). The hypervolume for the true Pareto front (PF) is calculated with the reference points given in parenthesis.

Env	PF	Algorithm	Mean	Std	Min	Median	Max
DST	1155.00 (25, 0)	**$\text{MEPS}_{\text{H1/S0}}$**	**1155.00**	**0.00**	**1155.00**	**1155.00**	**1155.00**
		$\text{MEPS}_{\text{H1/S1}}$	**1155.00**	**0.00**	**1155.00**	**1155.00**	**1155.00**
		$\text{MEPS}_{\text{H0/S0}}$	**1155.00**	**0.00**	**1155.00**	**1155.00**	**1155.00**
		$\text{MEPS}_{\text{H0/S1}}$	**1155.00**	**0.00**	**1155.00**	**1155.00**	**1155.00**
		PQL	947.40	223.66	663.00	1005.00	1155.00
		QM	1091.80	120.22	759.00	1155.00	1155.00
		MPSAC	1098.95	250.66	34.00	1155.00	1155.00
PBST	358636.00 (25, 0)	**$\text{MEPS}_{\text{H1/S0}}$**	**358636.00**	**0.00**	**358636.00**	**358636.00**	**358636.00**
		$\text{MEPS}_{\text{H1/S1}}$	**358636.00**	**0.00**	**358636.00**	**358636.00**	**358636.00**
		$\text{MEPS}_{\text{H0/S0}}$	**358636.00**	**0.00**	**358636.00**	**358636.00**	**358636.00**
		$\text{MEPS}_{\text{H0/S1}}$	**358636.00**	**0.00**	**358636.00**	**358636.00**	**358636.00**
		PQL	353983.20	1250.58	351582.00	353790.00	356175.00
		QM	-	-	-	-	-
		MPSAC	**358636.00**	**0.00**	**358636.00**	**358636.00**	**358636.00**
MBST	2632.00 (25, 0)	**$\text{MEPS}_{\text{H1/S0}}$**	**2632.00**	**0.00**	**2632.00**	**2632.00**	**2632.00**
		$\text{MEPS}_{\text{H1/S1}}$	2630.50	3.66	2622.00	2632.00	2632.00
		$\text{MEPS}_{\text{H0/S0}}$	2626.00	5.03	2622.00	2622.00	2632.00
		$\text{MEPS}_{\text{H0/S1}}$	2623.00	3.08	2622.00	2622.00	2632.00
		PQL	2608.20	26.97	2564.00	2620.00	2632.00
		QM	2629.50	11.18	2582.00	2632.00	2632.00
		MPSAC	870.60	1176.35	120.00	120.00	2622.00
DDST	1416.00 (25, 0)	**$\text{MEPS}_{\text{H1/S0}}$**	**1412.15**	**2.08**	**1411.00**	**1411.00**	**1416.00**
		$\text{MEPS}_{\text{H1/S1}}$	1409.95	2.82	1405.00	1411.00	1416.00
		$\text{MEPS}_{\text{H0/S0}}$	1408.95	3.05	1405.00	1409.00	1416.00
		$\text{MEPS}_{\text{H0/S1}}$	1407.55	3.42	1405.00	1405.00	1416.00
		PQL	1310.15	114.26	1033.00	1330.00	1411.00
		QM	1383.40	38.57	1299.00	1405.00	1405.00
		MPSAC	1406.60	2.01	1405.00	1405.00	1409.00
SE	11540.00 (400, 0)	$\text{MEPS}_{\text{H1/S0}}$	11539.00	4.47	11520.00	11540.00	11540.00
		$\text{MEPS}_{\text{H1/S1}}$	11347.00	832.95	7810.00	11540.00	11540.00
		$\text{MEPS}_{\text{H0/S0}}$	**11540.00**	**0.00**	**11540.00**	**11540.00**	**11540.00**
		$\text{MEPS}_{\text{H0/S1}}$	10956.50	1388.37	7700.00	11540.00	11540.00
		PQL	10740.00	1954.72	3960.00	11420.00	11540.00
		QM	5006.00	3313.47	0.00	7570.00	7790.00
		MPSAC	11131.00	1139.12	7810.00	11530.00	11540.00
BW	2038.00 (20, 0, 0)	**$\text{MEPS}_{\text{H1/S0}}$**	**799.12**	**166.09**	**690.46**	**724.86**	**1210.58**
		$\text{MEPS}_{\text{H1/S1}}$	709.32	6.46	693.90	709.12	720.66
		$\text{MEPS}_{\text{H0/S0}}$	772.10	127.18	708.68	722.19	1121.11
		$\text{MEPS}_{\text{H0/S1}}$	711.55	9.35	691.83	711.53	728.89
		PQL	704.00	29.42	608.00	708.00	732.00
		QM	-	-	-	-	-
		MPSAC	684.40	20.18	660.00	692.00	716.00

Similarly to Section 3.1.2, the mean and standard deviation hypervolume values are not sufficient to provide an effective analysis of the obtained results. Hence, the same statistical protocol is employed with a 5% significance value. Table 3.8

provides the results of the statistical test. The test results indicated that the results of algorithms under the column with a (+) differ with 95% confidence from those under the column with a (-). According to the test results, MEPS versions obtained competitive results in all the benchmark problems evaluated. In both DST and PBST, MEPS was as efficient as MPSAC. Moreover, MPSAC's networks contained 320 $(64 + 64 + (64 \cdot 7))$ connections and 69 (1 input, 64 hidden, 4 output) nodes, whereas MEPS output networks with an average of 25 connections and 10 nodes. Despite obtaining similar results as the QM algorithm in the MBST problem, MEPS versions were the best algorithms in DDST, SE and BW. Besides, H1/S0 was the best algorithm for the DDST problem, indicating that the proposal for a survivor selection mechanism that attempts to search through all the non-dominated fronts leads to competitive results in a problem with discontinuities and non-convex regions in the Pareto front. It is worth noting that no fine-tuning was performed and the parameters employed were selected empirically.

Table 3.8: Wilcoxon signed-rank test using hypervolume analysis of each MORL problem. Algorithms in the (+) column are statistically significant compared to algorithms in the column (-).

Env	Statistically Significant	
	(+)	(-)
DST	MEPS (all versions), MPSAC	PQL, QM
PBST	MEPS (all versions), MPSAC	PQL
MBST	$\text{MEPS}_{\text{H1/S0}}$, $\text{MEPS}_{\text{H1/S1}}$, QM	$\text{MEPS}_{\text{H0/S0}}$, $\text{MEPS}_{\text{H0/S1}}$ PQL, MPSAC
DDST	$\text{MEPS}_{\text{H1/S0}}$	$\text{MEPS}_{\text{H1/S1}}$, $\text{MEPS}_{\text{H0/S0}}$, $\text{MEPS}_{\text{H0/S1}}$ PQL, QM, MPSAC
SE	MEPS (all versions)	PQL, QM, MPSAC
BW	$\text{MEPS}_{\text{H1/S0}}$, $\text{MEPS}_{\text{H0/S0}}$	$\text{MEPS}_{\text{H1/S1}}$, $\text{MEPS}_{\text{H0/S1}}$ PQL, QM, MPSAC

3.2.3 Ablation Study for MEPS

After evaluating the different versions of MEPS in multiple benchmark environments, we performed an ablation study to investigate the effects of both parametrical and structural mutation operations. Ablations can significantly harm performance. Accordingly, we selected the DDST environment for the ablation study. This environment is complex enough even for unablated MEPS versions. Thus, we believe that it is suitable to compare ablated MEPS to its unablated counterpart.

According to the previously presented results, the MEPS version using crowding

distance and the heavy tail survivor selection operator, namely H1/S0, obtained the most competitive performance among the compared versions. Thus, we selected H1/S0 as the unablated MEPS version and, consequently, ablations were performed on this MEPS version.

In order to assess the hypothesis that H1/S0, using both structural mutations in combination with parametrical mutation, is the most competitive option for the DDST problem, we performed six ablations as follows:

- Ablation 1: allowed only the structural mutation of adding a new connection to previously unconnected nodes;
- Ablation 2: allowed only the structural mutation of adding a new hidden node;
- Ablation 3: allowed both structural mutations;
- Ablation 4: allowed only parametrical mutation;
- Ablation 5: allowed both the structural mutation of adding a new connection to previously unconnected nodes and parametrical mutation;
- Ablation 6: allowed both the structural mutation of adding a new hidden node and parametrical mutation.

Each ablated version was initialized using the parameters presented in Table 3.6, zeroing each respective restricted mutation probability. For example, Ablation 1 was initialized with zero probability for the "add node" mutation and zero standard deviation for parametrical mutation. With respect to the initial topology, ablated versions 2, 3, 4, and 6 were initialized as presented in Figure 3.13. However, if ablated versions 1 and 5 were initialized containing all the possible connections, the "add connection" mutation would never be executed and, therefore, we would not be able to assess its effect on the final performance. As a result, networks in ablated versions 1 and 5 were initialized by randomly connecting 50% of the possible connections.

Subsequently, 20 independent executions were performed for each ablated version. Figure 3.15 shows the normalized average hypervolume obtained by the unablated version and each of the ablations. Detailed results based on the hypervolume indicator are presented in Table 3.9. Ablations 1 and 2, which constrained the mutations to either add a new hidden node or a new connection, presented the worst average hypervolume values. Although Ablation 3, which employs both structural mutations, improved its performance when compared to Ablations 1 and 2, its performance is still far from those that employ parametrical mutation. This highlights the importance of the parametrical mutation in MEPS.

Among the ablated versions that employed parametrical mutation, Ablation 4, which does not use any structural mutation, achieved the highest mean hypervolume value. This indicates that the combination of parametrical mutation with either the "add connection" (Ablation 5) or "add node" (Ablation 6) mutations is still worse than employing all operators together as in the unablated version. Finally, we can see that only the unablated version of MEPS was able to find all the Pareto optimal solutions.

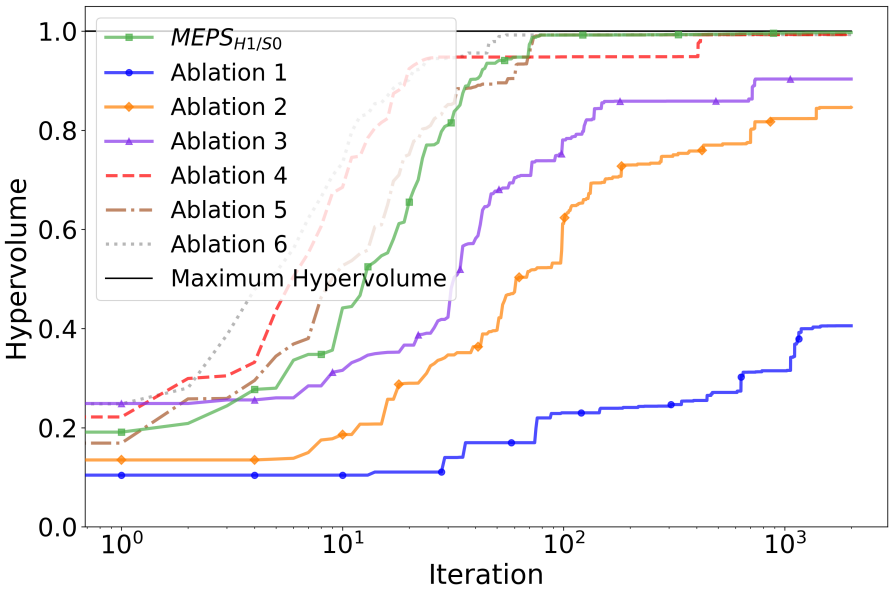


Figure 3.15: Average normalized hypervolumes obtained by the unablated H1/S0 version and its six ablated versions in the DDST MORL benchmark environment.

Table 3.9: Hypervolume analysis of H1/S0 unablated version and its six ablated versions in the DDST MORL benchmark environment. The hypervolume for the true Pareto front (PF) is calculated with reference points given in parenthesis.

PF	Algorithm	Mean	Std	Min	Median	Max
1416.00 (25, 0)	**MEPS$_{\mathbf{H1/S0}}$**	**1412.15**	**2.08**	**1411.00**	**1411.00**	**1416.00**
	Ablation 1	574.30	602.43	144.00	144.00	1405.00
	Ablation 2	1198.05	461.82	144.00	1405.00	1411.00
	Ablation 3	1279.20	388.23	144.00	1405.00	1411.00
	Ablation 4	1407.60	2.76	1405.00	1407.00	1411.00
	Ablation 5	1405.90	2.20	1405.00	1405.00	1411.00
	Ablation 6	1406.20	2.46	1405.00	1405.00	1411.00

3.3 Swarm Intelligence in Multi-Objective Evolutionary Policy Search

This thesis evaluates whether the combination of the Multi-objective Evolutionary Swarm Hybrid (MESH) [79] and MEPS improves the performance of standard MEPS. The motivation for addressing this challenge is based on the promising results in single-objective policy search achieved through a two-step evolution proposed by Stein et. al.[119]. This approach utilizes NEAT to optimize the topology of an ANN and PSO to optimize its corresponding weights. Therefore, this study proposes evaluating the use of MESH coupled with the aC-DEEPSO's adaptive velocity operator to perform an initial depth search in the weights space. Subsequently, MEPS will be utilized to evolve the ANN's topology along with weights to estimate action-preference values in MORL. Figure 3.16 illustrates the flowchart of the proposed approach, dubbed Swarm-Intelligent MEPS (SI-MEPS).

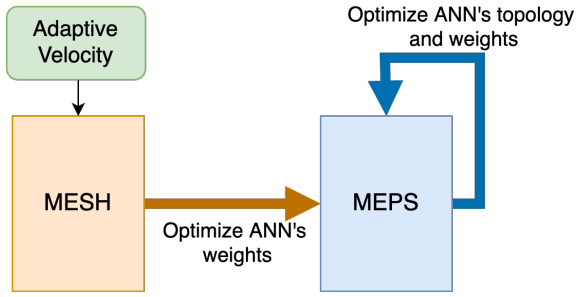


Figure 3.16: Illustration of the proposed integration between MESH and MEPS.

In order to run MESH, each individual in the MEPS population must be encoded as a real-valued vector. This encoding is done by creating a vector with a dimension equal to the number of connections and biases present in the initial population. Figure 3.17 illustrates this encoding for a population of ANNs containing three input nodes, one hidden node, and one output node. Finally, Algorithm 5 shows the pseudocode for the proposed coupling.

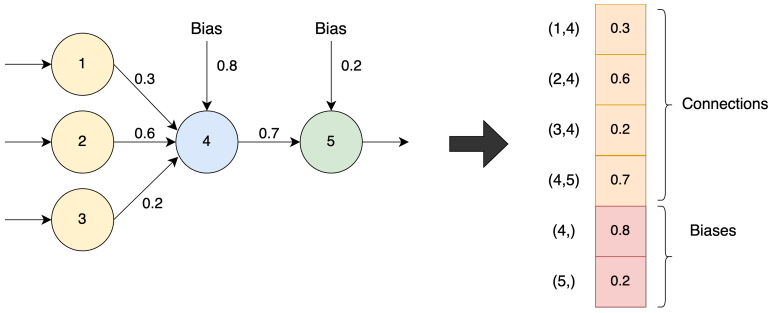


Figure 3.17: Illustration of the encoding of MEPS individuals into real-valued vectors.

Algorithm 5: Swarm-Intelligent Multi-Objective Evolutionary Policy Search (SI-MEPS)

Input: n_p, ψ, t_{max}, t_r, t_{mesh}, α, σ, p_{ac}, p_{an}, n_i, n_o, n_h, S, HT, h
Output: Memory M

Initialize population P_t with fully connected ANNs containing n_i input nodes, n_h hidden nodes and n_o output nodes;
$t \leftarrow 0$;
Evaluate each individual of P_t for an episode of length h;
Encode each individual of P_t as real-valued vector;
while $t < t_{mesh}$ **do**
    Run MESH[a] evaluating each individual of P_t for an episode of length h;
    $t \leftarrow t + 1$;
end
Update population P_t with the individuals output from MESH;
while $t < t_{max}$ **do**
    Run MEPS main loop, evaluating each individual of P_t for an episode of length h;
    Update memory M following memory update procedure;
    $t \leftarrow t + 1$;
end

[a]Both the pseudocode and the Python implementation of MESH are available in [79].

3.3.1 Preliminary Validation Simulations for SI-MEPS

The proposed coupling is compared to standard MEPS in three MORL benchmark environments, namely Modified Bountiful Sea Treasure [29], Discontinuous Deep Sea Treasure [60], and Space Exploration [131]. These environments were selected because not all versions of MEPS were able to find the entire set of solutions. Additionally, both versions were compared not only using the hypervolume measure, but also considering the complexity (number of connections + number of nodes) of the final solutions.

Similarly to Section 3.2.2, the SI-MEPS was executed independently 20 times in every problem. Since the main goal of this experiment is to validate the proposed approach, no fine-tuning of the parameters has been done. Therefore, SI-MEPS was executed with the same hyperparameters and initial topology as presented in Table 3.6 and Figure 3.13, respectively. Furthermore, the hyperparameters for the MESH counterpart of SI-MEPS were selected based on the results presented in [79]. As a result, the employed MESH version was E1/V1/D1 with 0.3 for both mutation and communication rates, guide size of 3, and memory size equal to the population size (for further details see [79]). In addition, the adaptive velocity operator of aC-DEEPSO presented in Section 3.1 is included with $\gamma = 0.2$. With respect to the number of generations, MESH was run for 200 generations and, subsequently, MEPS was run for 1800 generations.

Detailed benchmark results based on hypervolume indicator are shown in Table 3.10 and the algorithm with superior performance is indicated in bold for each problem. The average complexity of the networks obtained by each algorithm in

each benchmark environment is shown in Figure 3.18.

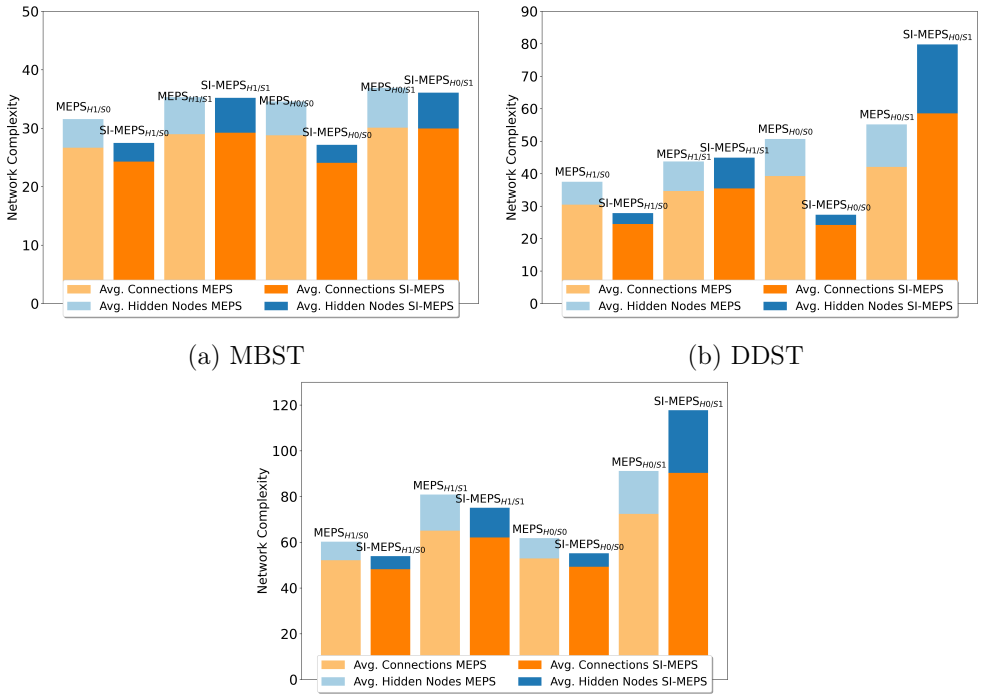
Table 3.10: Hypervolume analysis for each MORL benchmark environment (labeled as Env). The hypervolume for the true Pareto front (PF) is calculated with the reference points given in parenthesis.

Env	PF	Algorithm	Mean	Std	Min	Median	Max
MBST	2632.00 (25, 0)	**MEPS$_{\mathbf{H1/S0}}$**	**2632.00**	**0.00**	**2632.00**	**2632.00**	**2632.00**
		SI-MEPS$_{\mathbf{H1/S0}}$	**2632.00**	**0.00**	**2632.00**	**2632.00**	**2632.00**
		MEPS$_{\mathrm{H1/S1}}$	2630.50	3.66	2622.00	2632.00	2632.00
		SI-MEPS$_{\mathbf{H1/S1}}$	**2632.00**	**0.00**	**2632.00**	**2632.00**	**2632.00**
		MEPS$_{\mathrm{H0/S0}}$	2626.00	5.03	2622.00	2622.00	2632.00
		SI-MEPS$_{\mathrm{H0/S0}}$	2624.50	4.44	2622.00	2622.00	2632.00
		MEPS$_{\mathrm{H0/S1}}$	2623.00	3.08	2622.00	2622.00	2632.00
		SI-MEPS$_{\mathrm{H0/S1}}$	2624.50	4.44	2622.00	2622.00	2632.00
DDST	1416.00 (25, 0)	MEPS$_{\mathrm{H1/S0}}$	1412.15	2.08	1411.00	1411.00	1416.00
		SI-MEPS$_{\mathrm{H1/S0}}$	1411.25	1.12	1411.00	1411.00	1416.00
		MEPS$_{\mathrm{H1/S1}}$	1409.95	2.82	1405.00	1411.00	1416.00
		SI-MEPS$_{\mathbf{H1/S1}}$	**1414.25**	**2.45**	**1411.00**	**1416.00**	**1416.00**
		MEPS$_{\mathrm{H0/S0}}$	1408.95	3.05	1405.00	1409.00	1416.00
		SI-MEPS$_{\mathrm{H0/S0}}$	1410.05	2.58	1405.00	1411.00	1416.00
		MEPS$_{\mathrm{H0/S1}}$	1407.55	3.42	1405.00	1405.00	1416.00
		SI-MEPS$_{\mathrm{H0/S1}}$	1412.05	3.47	1405.00	1411.00	1416.00
SE	11540.00 (400, 0)	MEPS$_{\mathrm{H1/S0}}$	11539.00	4.47	11520.00	11540.00	11540.00
		SI-MEPS$_{\mathrm{H1/S0}}$	11537.00	4.70	11530.00	11540.00	11540.00
		MEPS$_{\mathrm{H1/S1}}$	11347.00	832.95	7810.00	11540.00	11540.00
		SI-MEPS$_{\mathbf{H1/S1}}$	**11540.00**	**0.00**	**11540.00**	**11540.00**	**11540.00**
		MEPS$_{\mathbf{H0/S0}}$	**11540.00**	**0.00**	**11540.00**	**11540.00**	**11540.00**
		SI-MEPS$_{\mathrm{H0/S0}}$	11537.50	4.44	11530.00	11540.00	11540.00
		MEPS$_{\mathrm{H0/S1}}$	10956.50	1388.37	7700.00	11540.00	11540.00
		SI-MEPS$_{\mathbf{H0/S1}}$	**11540.00**	**0.00**	**11540.00**	**11540.00**	**11540.00**

In MBST environment, the coupling of MESH as an initial depth search was able to improve the standard MEPS performance for the H1/S1 version, which consistently found all the Pareto solutions across executions. Despite this improvement, neither H0/S0 nor H1/S0 average hypervolumes were improved in SI-MEPS. The performance of the former version was slightly worse compared to standard MEPS. There was a marginal improvement in H0/S1, which was not enough to obtain the maximum hypervolume in every run. Concerning the network's complexity, SI-MEPS solutions consisted of networks at least as complex as standard MEPS solutions, indicating that, the proposed approach efficiently found less complex solutions for the H1/S0 configuration and improved H1/S1 performance without increasing its network complexity.

Except H1/S0, SI-MEPS improved the performance of all versions in the DDST environment. Moreover, the average hypervolume of H1/S1 not only increased but also outperformed the highest standard MEPS hypervolume. In terms of network complexity, SI-MEPS solutions were at least as complex as standard MEPS solutions, apart from the H0/S1 configuration. Similarly to the MBST environment, the SI-H1/S1 version improved the H1/S1 performance without increasing its network complexity. Furthermore, in SE environment, SI-MEPS had also enhanced standard MEPS performance performance. Contrarily to standard MEPS, in which only the H0/S0 version found all the solutions across runs, SI-MEPS enhanced the

performance of both H1/S1 and H0/S1 versions, at the expense of worsening the performances of H1/S0 and H0/S0. Although SI-H1/S1 solutions were less complex than standard H1/S1, SI-MEPS increased the network complexity of H0/S1 solutions.



(a) MBST

(b) DDST

(c) SE

Figure 3.18: Complexity of the networks obtained by each algorithm in each benchmark environment. Darker colors are used to indicate the complexities for SI-MEPS, while lighter colors indicate the complexities for standard MEPS.

Similarly to previous Sections, the performances are analyzed employing the same statistical protocol with a 5% significance value. Table 3.11 provides the results of the statistical test. The test results indicated that the results of algorithms under the column with a (+) differ with 95% confidence from those under the column with a (-). According to test results, the coupling between MESH and MEPS was most successful for H1/S1 version, which had its performance improved in every benchmark environment.

Table 3.11: Wilcoxon signed-rank test using hypervolume analysis of each MORL problem. Algorithms in the (+) column are statistically significant compared to algorithms in the column (-).

Env	Statistically Significant	
	(+)	(-)
MBST	$\text{MEPS}_{\text{H1/S0}}$, $\text{MEPS}_{\text{H1/S1}}$, $\text{SI-MEPS}_{\text{H1/S0}}$, $\text{SI-MEPS}_{\text{H1/S1}}$	$\text{MEPS}_{\text{H0/S0}}$, $\text{MEPS}_{\text{H0/S1}}$ $\text{SI-MEPS}_{\text{H0/S0}}$, $\text{SI-MEPS}_{\text{H0/S1}}$
DDST	$\text{SI-MEPS}_{\text{H1/S1}}$	$\text{MEPS}_{\text{H1/S1}}$, $\text{MEPS}_{\text{H0/S0}}$, $\text{MEPS}_{\text{H0/S1}}$ $\text{MEPS}_{\text{H1/S0}}$, $\text{SI-MEPS}_{\text{H0/S0}}$, $\text{SI-MEPS}_{\text{H0/S1}}$ $\text{SI-MEPS}_{\text{H1/S0}}$
SE	$\text{SI-MEPS}_{\text{H1/S1}}$, $\text{MEPS}_{\text{H0/S0}}$, $\text{SI-MEPS}_{\text{H0/S1}}$	$\text{MEPS}_{\text{H1/S1}}$, $\text{MEPS}_{\text{H1/S0}}$, $\text{MEPS}_{\text{H0/S1}}$ $\text{SI-MEPS}_{\text{H1/S0}}$, $\text{SI-MEPS}_{\text{H0/S0}}$

Chapter 4

Results and Discussion on the ERM problem

4.1 Single-Objective Energy Resource Management Problem

Additionally to the preliminary validation tests for aC-DEEPSO, the framework of an energy resource management problem (ERM) [4, 17] is utilized to assess the performance of the proposed algorithm in a large-scale problem. This framework, which was proposed in the scope of the *GECAD Smart Grid Competition 2022* [42, 115], simulates a large smart grid or smart city mockup located in BISITE laboratory[1] [17]. This simulated smart grid is built on a 13-bus medium voltage distribution network as shown in Figure 4.1. It is composed of one substation, two wind farms, 13 photovoltaic (PV) parks, and 2 energy storage systems (ESSs). Moreover, a fleet of 500 electric vehicles (EVs) and five charging stations are also included [4].

Regarding load demand, 25 distinct loads are considered, including residential (1375 houses) and office buildings (seven buildings), a hospital, a fire station, and a shopping mall [17]. Thus, each particle is encoded as the hourly state of this distribution network in a 24 hour period. For each hour, the simulation follows these optimization characteristics: (1) 21 continuous variables are used to represent the active power in each generator; (2) 21 binary variables are used to indicate whether each generator is active or not; (3) 500 continuous variables are used to describe the amount of power being charged or discharged in each EV; (4) 25 continuous variables are employed to indicate the reduction in each of the 25 loads; (5) 2 continuous variables are used to represent the charge/discharge state of each ESS;

[1] This laboratory is located in the University of Salamanca, but the network does not correspond to a physical electrical network located in the city of Salamanca.

and (6) 1 continuous variable is used to describe the behavior of the aggregator in the energy market. Moreover, a maximum limit of 5000 function evaluations is allowed in each execution [4, 17, 42, 115].

Hence, each particle lies in $\mathbb{R}^{24 \cdot 570}$ totaling 13680 optimization variables. In addition to this, the objective function is calculated through the evaluation of 150 randomly generated scenarios. For further details on scenario generation, please refer to [4].

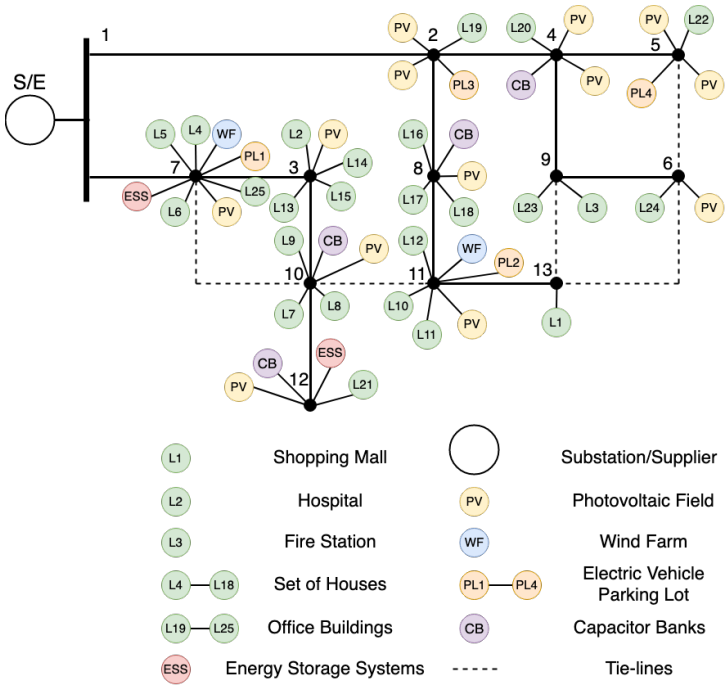


Figure 4.1: Line diagram of the 13-bus distribution network. Adapted from [17].

Accordingly, the objective function (OF) employed in the management problem is the following [4]:

$$OF = Z^{Ex} + \beta \cdot CVaR_{\alpha}(Z_s^{tot}), \tag{4.1}$$

in which $\beta \in [0, 1]$ represents the level of aversion to risk for CVaR in the worst case scenario s, and Z^{Ex} is the expected cost accross the set of scenarios. Moreover, in this experiment, population size was set to 50 particles and C-DEEPSO's parameters to 0.6 and 0.9 for communication and mutation rates, respectively. Each run was evaluated over 150 different randomly-generated scenarios that may include, or not, an extreme event. For the addressed ERM problem, each algorithm was allowed to execute for a maximum of 5000 function evaluations (FEs). Due to the stochastic nature of the evaluated methods, each algorithm was run 20 times. Moreover, the average computational time of seven minutes for each execution to find an optimized solution is viable for day-ahead operational planning in a high-dimensional complex ERM problem.

Figure 4.2 shows a radar plot containing the average performance in 150 scenarios of each of the evaluated algorithms for each run. The goal was to minimize the objective function of the ERM problem described by Equation (4.1), with $\beta = 1.0$. Thus, the smaller the area covered by the algorithm, the better its performance is. Regarding standard PSO and EPSO, neither version of either algorithm was able to escape from a local minima in any of the 20 runs, obtaining a mininum cost of around 18500 monetary units (m.u.).

C-DEEPSO version with local search, depicted as the orange area, presented some improvements. It was able to escape the aforementioned local minima in eight out of the 20 runs, outperforming not only PSO and EPSO but also their counterparts with local search in runs 4, 6, 10, 11, 12, 13, 14 and 19. After introducing the proposed adaptive velocity operator to C-DEEPSO with local search it is possible to see a reduction in the area depicted in blue. This smaller area indicates that combining adaptive velocity and local search (LS) with C-DEEPSO leads to finding smaller costs across runs. For instance, Adaptive C-DEEPSO with LS was able to escape from the previously mentioned local minima in 14 out of 20 runs, outperforming C-DEEPSO with LS in runs 2, 3, 4, 5, 8, 9, 13, 15, 16, 17, 19 and 20.

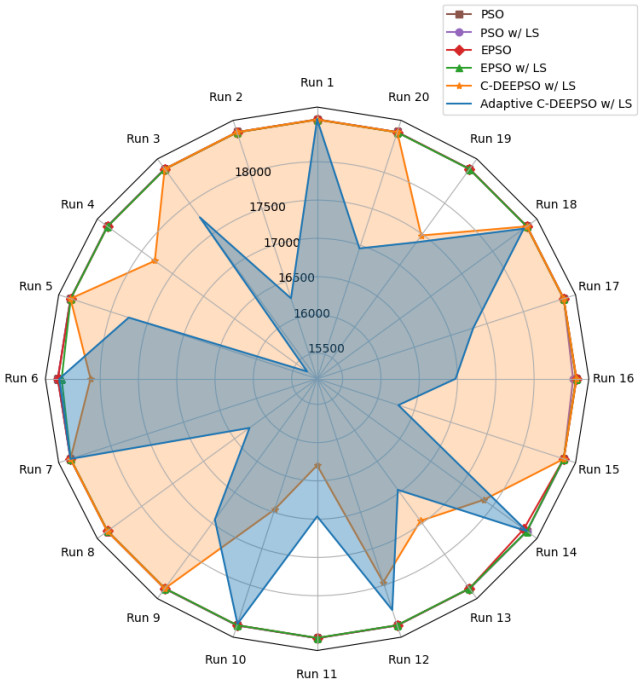


Figure 4.2: Radar plot containing the average cost of each algorithm in 150 random scenarios for 20 runs. The smaller the area covered by the algorithm, the better it performed. Extracted from [61]

.

Table 4.1 shows the average results for the 20 runs of each algorithm, separated into the mean expected cost and $CVaR_{\alpha=95\%}$. It can be seen that aC-DEEPSO achieves the smallest value of 17460.838 m.u.. It is worth noting that there is a trade-

off between mean expected cost and $CVaR_{\alpha=95\%}$. While PSO and EPSO algorithms obtained a lower mean expected cost, the risk cost was increased. Reversely, both C-DEEPSO versions obtained results with smaller $CVaR_{\alpha=95\%}$ cost at the expense of a small increase in the mean expected cost. This trade-off is illustrated by Figure 4.3, in which the risk factor β is varied from 0 (risk-neutral) to 1 (risk-averse).

For a value of $\beta < 0.1$, the increase in the mean expected cost leads to a higher average cost for both C-DEEPSO with LS and Adaptive C-DEEPSO with LS. However, for $\beta \geq 0.1$, the reduction in risk cost compensates the higher mean expected cost. With respect to the costs of the worst scenarios, Adaptive C-DEEPSO with LS also achieved the lowest average cost. Moreover, Adaptive C-DEEPSO with LS produced less constraint violations, indicated by a penalty value of 385 against 411.667 for C-DEEPSO with LS and 433.333 for the other algorithms.

Table 4.1: Summary of the obtained average results in the ERM problem.

Algorithm	OF	Fex	CVaR	Worst Scenario	Violations
PSO	18554.270	8508.164	10046.106	20709.064	433.333
PSO w/ LS	18551.606	8508.476	10043.129	20706.777	433.333
EPSO	18549.344	8507.215	10042.129	20706.786	433.333
EPSO w/ LS	18548.565	8506.886	10041.679	20706.770	433.333
C-DEEPSO w/ LS	18128.825	8600.884	9527.942	20160.072	411.667
Adaptive C-DEEPSO w/ LS	**17460.838**	**8628.736**	**8832.103**	**19309.028**	**385.000**

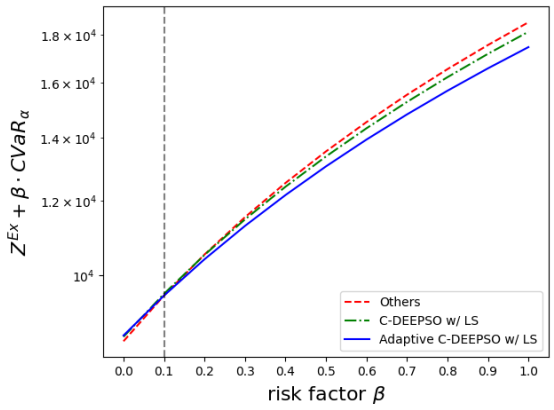


Figure 4.3: Average cost for different values of risk factor β

.

To provide a robust evaluation of the results, the null hypothesis is defined as the equality of the mean results obtained after 20 runs of the experiment. After that, the Conover non-parametric statistical test [24] is applied to do a pairwise comparison and assess the pairs in which the null hypothesis can be rejected, for a given significance level α. The test results indicated that, with 99% confidence,

Adaptive C-DEEPSO with LS achieved lower costs compared to PSO, PSO with LS, EPSO and C-DEEPSO with LS. Furthermore, test results also indicated, with 95% confidence, that Adaptive C-DEEPSO with LS achieved lower costs than EPSO with LS.

Finally, in a monthly projection Adaptive C-DEEPSO with LS is able to save more than 20 thousand m.u. and more than 32 thousand m.u. when compared to C-DEEPSO with LS and standard PSO, respectively. Additionally, from a carbon dioxide (CO_2) emission standpoint, an estimation of the daily environmental impact of the ERM solutions obtained using our proposal compared to standard PSO, in terms of the amount of CO_2 generated per kWh consumed, indicated that our proposal saves up to 4.497 tons of CO_2eq./kWh per day (see [61]). Furthermore, it can be said that Adaptive C-DEEPSO with LS is suitable for a risk-based energy management system.

4.2 Multi-Objective Energy Resource Management Problem

4.2.1 Problem Formulation

In this section the Energy Resource Management problem for a planning horizon of one year is formally defined. First, the microgrid configuration followed by the objective functions and constraints definitions is presented. Then, the ERM control problem is modeled as a Markov Decision Process (MDP) and treated from a multi-objective perspective regarding operating cost, greenhouse gas emissions, and battery degradation. In this resulting control problem, the learning agent plays the role of an EMS that is responsible for managing the maximum allowed monthly ESS' depth of discharge (DoD), and the energy imported from public grid.

Microgrid System model

The structure of the simulated solar wind power microgrid system is based on [83]. It is composed by six Norvento nED 100-22 wind turbines [94], 5000 HiKu 450W-CS3W-450MS photovoltaic panels [16], an ESS, DC/AC converter, electrical load that comprises both residences and industrial buildings, main grid connection (the main grid price mechanism employed is real-time pricing (RTP)), and EMS. Figure 4.4 presents the system structure. Moreover, the ESS consists in a 1000kW capacity spinel lithium titanate ($Li4Ti5O_{12}$ (LTO) [141]) battery. Table 4.2 details the configurations values for the MG project with lifetime of 24 years.

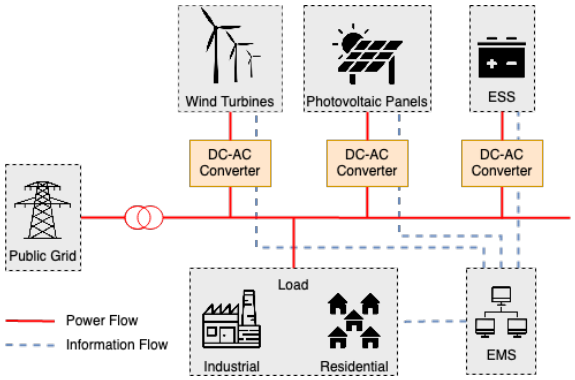


Figure 4.4: Solar wind power microgrid system structure. Based on [67].

Table 4.2: General information about the microgrid.

	Unit	INV	PV	WT	LTO Battery
Life Time (years)	years	15	24	24	17.5
Efficiency (%)	%	96	20.4	95	90
Rated Power	kW	-	0.45	100	-
Capacity	kW	-	-	-	1000
Cycles	un	-	-	-	8000
Initial Cost	$	-	500.00	1800.00	-
Cost	$/kW	700.00	-	-	1143.00
Operational Cost	$/kW	-	18	0.36	-

Regarding the data used, a generic region from Cadiz, in Southern Spain, was used as case study of this work. In this regard, an annual load profile (measured in 8640h) comprising both a residential community and an industrial consumption are analyzed. Besides, hourly data from 2021 for dynamic energy price along with solar radiation, wind speed, and ambient temperature from Solcast [116] serve as input to the MG system. Next, the objective functions and the different constraints involved in the problem are introduced.

Operational Costs objective and ESS constraints

The charge/discharge control of the ESS represents the use or storage of energy from the ESS. At each time step t, the ESS can be either charging or discharging. The minimum state-of-charge (SoC) value is defined as the $1 - DoD(t)$ of ESS. Furthermore, the SoC at time step t is given by [67]:

$$SoC(t) = \begin{cases} SoC(t-1) + \frac{E_{bat}(t) \cdot \eta_c}{E_{rated}}, & E_{bat}(t) \geq 0 \\ SoC(t-1) + \frac{E_{bat}(t)}{E_{rated} \cdot \eta_d}, & E_{bat}(t) < 0, \end{cases} \tag{4.2}$$

in which $\eta_c = 0.90$ and $\eta_d = 1.0$ are the charging and discharging efficiencies, respectively. E_{rated} denotes the rated capacity of the ESS. The amount of used or stored energy from the ESS, $E_{bat}(t)$, is calculated as

$$E_{bat}(t) = \begin{cases} E_{dch}(t), & \text{if discharging} \\ E_{ch}(t), & \text{if charging,} \end{cases} \tag{4.3}$$

in which $E_{dch}(t)$ and $E_{ch}(t)$ are the discharging and charging requested energy amount given by [83]:

$$E_{dch}(t) = \begin{cases} \max\{(1 - DoD(t) - SoC(t-1)) \cdot E_{rated}, E_{dch}(t)\}, & SoC(t-1) > 1 - DoD(t) \\ 0, & SoC(t-1) \leq 1 - DoD(t), \end{cases} \tag{4.4}$$

$$E_{ch}(t) = \min\{(SoC_{max} - SoC(t-1)) \cdot E_{rated}, E_{ch}(t)\}. \tag{4.5}$$

In addition, the use and degradation costs of ESS are considered in the operation. The ESS use cost coefficient is given by [77, 83]:

$$c_d(t) = \frac{C_i}{L_c \cdot E_{rated} \cdot DoD(t)}, \tag{4.6}$$

in which L_c is the available cycle lifetime and C_i is the initial investment of ESS. Therefore, in this modeling the total cost of an operation is obtained by [77]:

$$C_{total}(t_{start}, t_{end}) = IC + PW_p + PW_{np} + \sum_{t=t_{start}}^{t_{end}} c_d(t), \tag{4.7}$$

in which the value $C_{total}(t_{start}, t_{end})$ represents the sum of the system costs of operating from time t_{start} to t_{end}. The initial cost (IC) refers to the 20% cost for operation & maintenance, 6% discount rate, 1.4% inflation rate, personnel cost, installation and connections. It is also included both the periodic costs PW_p of components maintenance, such as PV panels and wind generators, and the non-recurrent costs PW_{np} of components replacement, such as ESS [77]. Moreover, the charging/discharging power constraints are given by

$$\begin{cases} 0 \leq E_{ch} \leq E_{ch}^{max}, \\ 0 \leq E_{dch} \leq E_{dch}^{max}, \\ E_{ch} \cdot E_{dch} = 0, \end{cases} \tag{4.8}$$

in which E_{ch}^{max} and E_{dch}^{max} are the maximum charge and discharge energy, respectively. Additionally, the ESS' SoC should be maintained at a suitable range at each time

step

$$SoC_{min}(t) \leq SoC(t) \leq SoC_{max}, \tag{4.9}$$

with $SoC_{min}(t)$ denoting the minimum SoC value at time step t and SoC_{max} as the maximum allowed SoC. Finally, the resulting objective function for cost minimization modeled by [83] is

$$\min Cost(t_{start}, t_{end}) = C_{total}(t_{start}, t_{end}) + \sum_{t=t_{start}}^{t_{end}} P_t^{buy} \cdot Pr_t, \tag{4.10}$$

where the operational cost is combined with the cost of the energy bought from the public grid P_t^{buy} at price Pr_t to supply the insufficient microgrid power.

Microgrid CO_2 emissions objective

In order to take into consideration an estimation of the amount of CO_2 emissions for public grid energy bought and renewable generation, each energy source is quantified in terms of grams of CO_2eq./KWh. The greenhouse gas emission values used are the average between the minimum and maximum values from [127]. Hence, the following values have been used:

- Solar Photovoltaic: 44.15 g CO_2eq./KWh;
- Wind Power: 11.90 g CO_2eq./KWh;
- Nuclear: 5.75 g CO_2eq./KWh;
- Hydro: 76.50 g CO_2eq./KWh;
- Cogeneration and Combined cycle: 156.00 g CO_2eq./KWh;

Moreover, with respect to energy bought from public grid, the CO_2eq./KWh quantity is calculated using the dispatchable energy composition in Spain as reported by [103]:

- Solar Photovoltaic: 9%;
- Wind Power: 26%;
- Nuclear: 24%;
- Hydro: 13%;
- Cogeneration: 17%:
- Combined cycle: 11%;

Thus, the total emissions in CO_2eq./KWh [61] for wind/solar power generation and energy imported from the public grid are each:

$$Emission_{WT}(t_{start}, t_{end}) = 11.90 \cdot \sum_{t=t_{start}}^{t_{end}} P_t^{WT} \tag{4.11}$$

$$Emission_{PV}(t_{start}, t_{end}) = 44.15 \cdot \sum_{t=t_{start}}^{t_{end}} P_t^{PV} \tag{4.12}$$

$$\begin{aligned} Emission_{Buy}(t_{start}, t_{end}) = (44.15 \cdot 0.09 + 11.90 \cdot 0.26 + 5.75 \cdot 0.24 + \\ 75.60 \cdot 0.13 + 156.00 \cdot 0.28) \cdot \sum_{t=t_{start}}^{t_{end}} P_t^{Buy}, \end{aligned} \tag{4.13}$$

in which P_t^{WT} and P_t^{PV} stand for the energy generate from wind turbines and photovoltaic panels at time t, respectively. From that, Leite et al. [61] modeled an objective function for minimizing the amount of CO_2 emissions, according to:

$$\begin{aligned} \min Emission(t_{start}, t_{end}) = Emission_{WT}(t_{start}, t_{end}) + \\ Emission_{PV}(t_{start}, t_{end}) + Emission_{Buy}(t_{start}, t_{end}). \end{aligned} \tag{4.14}$$

ESS degradation objective

This modeling approach includes a quantification of the Lithium-Ion battery (LIB) capacity degradation as a combination from calendar and cycle ageing, given by [105]:

$$\begin{aligned} \Delta C = C_0 \cdot \Bigg(0.75\tau \cdot \sum_t \alpha_{cap_t} \cdot d^{-0.25} + \\ \frac{1}{\sqrt{EFC}} \left(\sum_{w=1}^{W} \beta_{cap}(DoD_w) \cdot DoD_w + \frac{1}{2} \sum_{h=1}^{H} \beta_{cap}(DoD_h) \cdot DoD_h \right) \Bigg), \end{aligned} \tag{4.15}$$

in which C_0 is the initial battery capacity, d are the total days, t is the model period in hours. W and H are the number of whole and half equivalent full cycles (EFC), obtained after applying rainflow cycle counting algorithm to the ESS state of charge's (SoC) profile [91]. DoD_w and DoD_h are the depths of discharge (DoD) associated to each whole and half EFCs, respectively. The term α_{cap} denotes the calendar ageing factor, given by [105]

$$\alpha_{cap} = (7.543V - 23.75) \cdot 10^6 \cdot e^{6976/T}, \tag{4.16}$$

in which V and T are the battery's cell voltage and temperature (in K), respectively. It is assumed that the cell voltage is constant and equals to 3.7 [74] and T is equal to the ambient temperature T_{amb} converted from Celsius to Kelvin. In addition, the term β_{cap} describes the battery ageing factor in terms of equivalent full cycles [105],

$$\beta_{cap}(DoD) = 7.348 \cdot 10^{-3} \cdot (\bar{V} - 3.667)^2 + 7.6 \cdot 10^{-4} + 4.081 \cdot 10^{-3} \cdot DoD. \quad (4.17)$$

The term $\bar{V}$ is the average cell voltage, which, in this work, is equal to the cell voltage $V = 3.7$. Therefore, the objective function for minimizing the accumulated ESS' degradation according to [105] is

$$\min Degradation(t_{start}, t_{end}) = \sum_{t=t_{start}}^{t_{end}} \Delta C(DoD(t)). \quad (4.18)$$

ERM Markov Decision Process

In the presented ERM problem, the energy management is performed through both the monthly control of the ESS's depth of discharge (DoD), and the imported energy from public grid. By setting a DoD value, the manager restricts the amount of energy that is available to be used from the ESS. The insufficient microgrid power is supplied by importing from the public grid. Thus, the system needs information from the environment to set different DoD values under different states. As a result, the system dynamics of the ERM can be formalized as a Markov decision process (MDP) characterized by a state space S, an action space A, and a reward R evaluated every month over a finite time horizon of a year. The state space S is characterized by a $\mathbb{R}^3$ vector that contains the state information at each month m:

$$s_m \in S = \left\{ SoC(t), \frac{T_m^{buy}}{T_m^{load}}, \frac{T_m^{dch}}{T_m^{load}} \right\}, \quad (4.19)$$

in which T_m^{buy} denotes the total energy bought in month m, T_m^{dch} denotes the total energy discharged from the ESS in month m, and T_m^{load} is the accumulated load demand over the month m. Note that, since data time steps are defined in hours, each month consists in a batch of 720 hours.

At each month, the agent decides the DoD that will be used. Thus, the action space contains eight available actions and is defined as

$$a_m \in A = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}. \quad (4.20)$$

After an action is performed, the ESS dynamics is updated as follows:

$$DoD(m) = a_m, \quad (4.21)$$

in which $DoD(m)$ indicates the DoD used for every hour corresponding to month m.

Finally, the multi-objective reward function is a $\mathbb{R}^3$ vector as follows:

$$\mathbf{R}(s_m, a_m) = [Cost(t^m_{start}, t^m_{end}), Emission(t^m_{start}, t^m_{end}), Degradation(t^m_{start}, t^m_{end})]\,, \tag{4.22}$$

where $t^m_{start} = 720 \cdot (m-1)$ and $t^m_{end} = 720 \cdot m$. The components of the reward vector indicate a summation of the *Cost*, CO_2 *Emission* and ESS *Degradation* over the hours within the current month m.

Once the MDP is defined, reinforcement learning is used for solving the resulting control problem. The learning agent dynamics of the proposed control problem can be summarized as follows:

(1) **Choose** an action a_m.

(2) **Run** the microgrid for 720 hours (30 days).

(3) **Update** the current state s_m.

(4) **Update** Costs.

(5) **Update** Emission.

(6) **Update** Degradation.

(7) **Update** battery capacity based on its degradation from the previous month.

(8) **Repeat** steps (1)-(7).

4.2.2 Solving the ERM problem

This section presents an analysis of the MEPS results in the proposed multi-objective ERM problem of controlling the DoD of an ESS in a microgrid, with both solar and wind generation. In addition, two standard MORL algorithms are used for comparison, namely the Multi-Policy Soft Actor Critic and the Multi-Objective Deep Q Networks (MODQN). The former is based on ANNs and multi-objective CMA-ES [19], while the latter is based on the MORL framework for Deep RL proposed in [93].

In terms of configurations, MODQN used two 64-neuron fully connected layers. MPSAC used ANNs with a hidden layer of size 64, and MEPS initialized the ANN population without any hidden layers. For all algorithms, ReLU function activation was used in the neurons. In addition, the configuration of the output layer was the same for all three algorithms. It contained 8 neurons for the 8 actions as given in

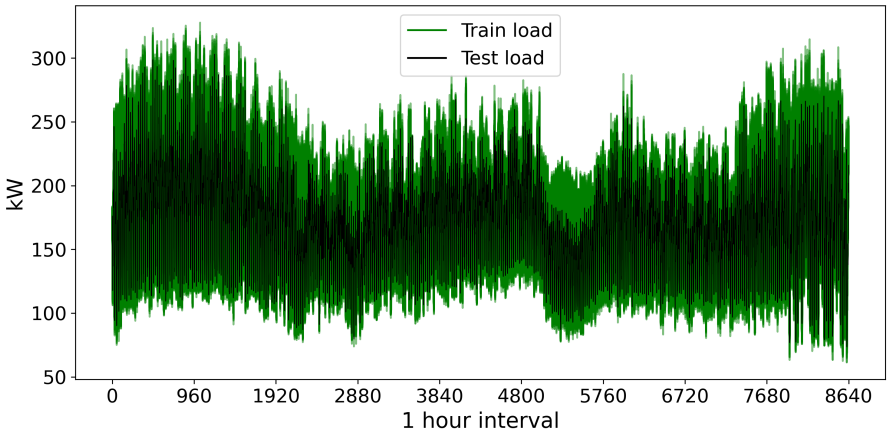
equation 4.20. After fine-tuning the Heavy tail selection parameter, parametrical mutation standard deviation, "Add connection" and "Add node" mutation probabilities, Table 4.3 lists the parameters used for each version of MEPS as well as MPSAC and MODQN. Further details of the performed fine-tuning are presented in Chapter 5.

Table 4.3: Parameter initialization values used by the algorithms in the microgrid environment.

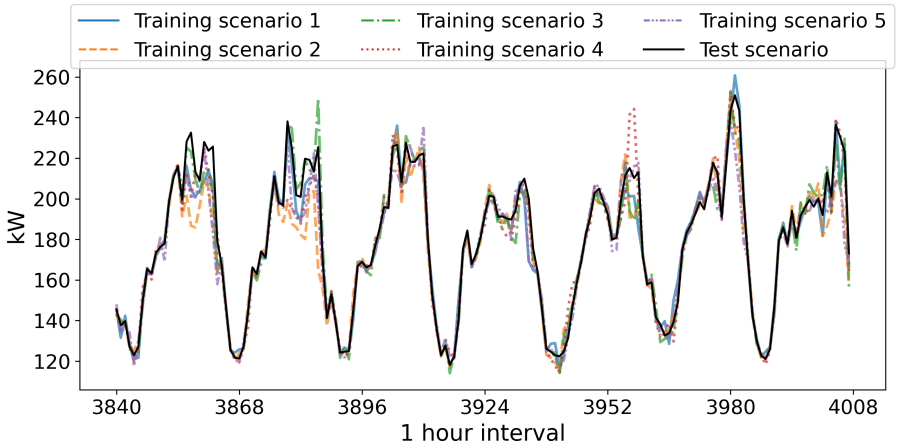
Description	MEPS				MPSAC	MODQN
	H1/S0	H1/S1	H0/S0	H0/S1		
Population size	20					
Initial fraction selected from first front	0.5				-	-
End generation number of heavy tail survivor selection	250				-	-
Heavy tail selection parameter	1.0	0.3	-	-	-	-
"Add connection" mutation probability	0.6	0.3	0.6	0.4	-	-
"Add node" mutation probability	0.6	0.5	0.5	0.3	-	-
Parametrical mutation standard deviation	1.1	1.2	1.3	0.9	1.0	-
Learning rate	-				0.001	0.001
Gamma	-				0.99	0.99
Initial epsilion	-				-	0.1
Epsilon decay	-				-	$3.7 \cdot 10^{-5}$
Generations	500				250 + 250	500
Episode length	12					

In order to assess the generalization ability of each algorithm, different load scenarios were used in training and testing. The test scenario is depicted in Figure 4.5a as a black line, and the green area shows the range of the noise added to the test scenario at each hour when generating training scenarios. During training, five different load scenarios are randomly sampled from the green area as shown in Figure 4.5b. The training rewards were the average of the rewards obtained from each one of the five scenarios. Moreover, each algorithm was run 20 times with the same initialization parameters as presented in Table 4.3 and started with ESS at 20% SoC, in state $s_1 = \{0.2, 0.0, 0.0\}$. The reference point used for hypervolume calculation was $(600000, 65, 0.085)$.

The average hypervolume for the 20 runs of each algorithm's training rewards is presented in Figure 4.6. During training, HV values from MPSAC oscillated in the first 250 generations and improved in the last 250, when the MO-CMA-ES counterpart started. On the other hand, although the multi-objective DQNs increased accross generations, it only presented a HV value higher than MPSAC in the first 250 generations. After 250 generations, MODQN performance was below the performances of all the other algorithms. MEPS shows the best performance regarding HV, with H1/S1 and H0/S1 achieving the highest HV values at the end of training.



(a) Load demand scenarios. Black line represents the test scenario, and the green area represent the range of the noise added to the test scenario when generating training scenarios.



(b) Example of a week in autumn from five sampled training scenarios.

Figure 4.5: Train and test load scenarios generation.

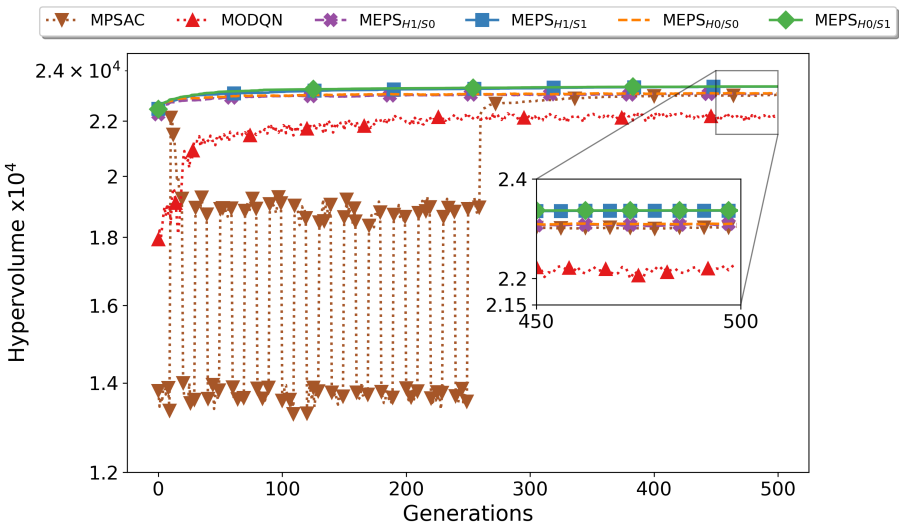


Figure 4.6: Hypervolume average values of 20 executions of each algorithm during training.

Afterwards, each algorithm was evaluated on the test scenario to assess its generalization ability on a scenario never seen before. Table 4.4 details the test results obtained. It can be seen that, although able to generalize to an unseen scenario, MODQN mean results are not only worse than the results from MEPS and MPSAC, but also present a higher standard deviation. Comparing MPSAC with MEPS versions, MPSAC results are very similar to H1/S0 results and very close to H0/S0 results. However, with respect to H1/S1 and H0/S1, MPSAC's performance is worse not only in terms of mean HV but also in terms of standard deviation. Moreover, MPSAC's best HV value is lower than worst HV values in both H1/S1 and H1/S0. Among the MEPS versions, H1/S1 and H1/S0 show very similar performances and present the best values with the smallest variation when compared to MODQN, MPSAC and other MEPS versions.

Table 4.4: Performance of each algorithm regarding hypervolume when evaluated in the test scenario.

	Mean	Std.	Worst	Median	Best
$\text{MEPS}_{\text{H1/S0}}$	23073.24	94.59	22896.73	23053.54	23295.62
$\text{MEPS}_{\text{H1/S1}}$	23346.12	57.66	23249.00	23336.72	23470.91
$\text{MEPS}_{\text{H0/S0}}$	23077.56	82.06	22935.55	23077.32	23241.86
$\text{MEPS}_{\text{H0/S1}}$	**23351.87**	**51.16**	**23265.61**	**23347.58**	**23440.45**
MPSAC	23018.29	122.85	22735.94	23062.46	23202.51
MODQN	22311.92	59.24	22225.02	22339.06	22368.22

Nevertheless, two non-parametric tests were conducted to provide a robust evaluation of the results obtained. First, the boxplot behavior was analyzed, and then, similar to Section 3.2.2, a two-fold analysis was carried out. This analysis consisted of a Kruskal-Wallis test, followed by a Wilcoxon signed-rank posthoc test with the Holm-Bonferroni correction. Boxplots are a useful tool for analyzing the range and distribution of the data, and sometimes, obtain information about the true difference among the means. If the notches in the boxplots do not overlap, it can be concluded, with 95% confidence, that the true means do differ [82]. With this in mind, and by analyzing Figure 4.7, it is possible to conclude that there are differences among the means of the algorithms.

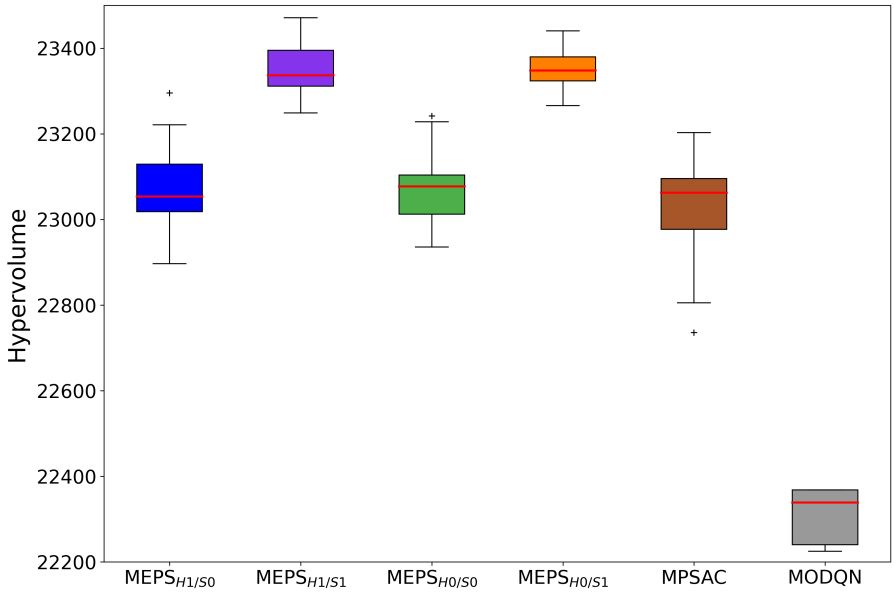


Figure 4.7: Boxplot results of the average hypervolumes on test scenario for 20 executions of each algorithm.

To statistically determine the difference in performance of the evaluated algorithms, a Kruskal-Wallis test with 1% significance level was applied. Thereafter, a Wilcoxon signed-rank test with Holm-Bonferroni correction was applied to find out, by pairwise comparisons, which specific group's means were different. The results of the Kruskal-Wallis test corroborated the boxplot analysis, and attested that, with a higher confidence level of 99%, there are differences among the mean hypervolume values. From the results of the posthoc tests, it is possible to rank the algorithms as shown in Table 4.5.

Table 4.5: Ranking of algorithms based on Wilcoxon signed-rank test results using mean hypervolumes in test scenario.

Rank		
1	2	3
MEPS$_{H1/S1}$	-	-
MEPS$_{H0/S1}$	-	-
-	MEPS$_{H1/S0}$	-
-	MEPS$_{H0/S0}$	-
-	MPSAC	-
-	-	MODQN

Therefore, the experimental results have shown that MEPS networks are able to provide feasible solutions to the proposed ESS control problem with performance comparable to state-of-the-art MORL techniques. In particular, H1/S1 and H0/S1 outperformed Deep Q Networks as well as the combination of MO-CMA-ES and Soft Actor-Critic in the proposed multi-objective ERM problem. Furthermore, both H1/S1 and H0/S1 solutions are composed of ANNs with an average of 30 connections and 15 nodes. Compared to MPSAC networks (75 nodes and 715 connections) and

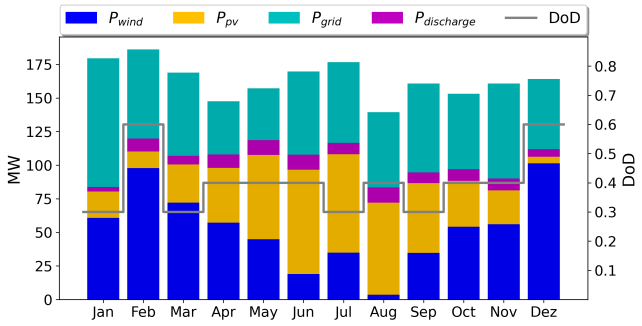
MODQN networks (131 nodes and 4936 connections), H1/S1 and H0/S1 solutions consist of much lighter networks. These light networks are a suitable solution to microgrids because they can be deployed as controllers in devices with low computing power.

An important observation regarding MEPS configuration is that, contrary to the preliminary validation results, the use of hypervolume contribution as the density measure (S1) achieved higher hypervolume values than MEPS counterparts using crowding distance (S0). This indicates that, although benchmarking algorithms using test environments are necessary to validate new techniques, test environments are not globally sufficient to attest a model's performance. In fact, when it comes to real-world problems such as energy management-based problems, some models are better suited than others.

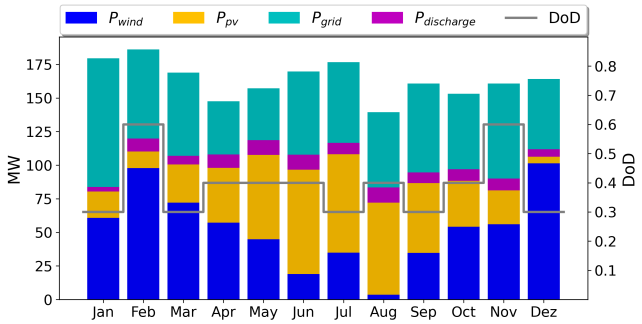
Next, an analysis of the microgrid functioning in both the H1/S1 and H0/S1 solutions is carried out. As the multi-objective approach provides alternatives for a decision maker to select knowledge-based solutions, two multi-criteria decision technique, namely Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) and Modified TOPSIS[18, 126], are employed to select a solution that satisfies an equal preference among the three objectives. In standard TOPSIS and its modified variant incorporate, the decision maker's preferences are considered to find the solution closest to the positive ideal solution and furthest from the negative ideal solution. In this analysis, each objective was given an equal preference weight value of 33.3%.

Due to the use of equal preferences for each objective, the solutions obtained by both standard and modified TOPSIS were the same. Figure 4.8 illustrates the MG behavior of the obtained solution in a year of operation using H1/S1 and H0/S1. The total amount of wind and solar energy consumed per month is indicated by P_{wind} and P_{pv}, respectively. The amount of energy purchased from the public grid is indicated by P_{grid}, and the total energy discharged from the battery per month is denoted by $P_{discharge}$.

Both solutions are nearly identical, with the only difference being in November and December. Yet, this difference results in a significant change in terms of cost and emissions. In November when there is less wind and solar radiation compared to December, the H0/S1 solution uses a higher DoD. This leads to a cost of \$518366.63, CO_2 emissions of 60.62 tons of CO_2eq./kWh and an ESS degradation of 3.47% per year. In contrast, the H1/S1 solution adopts a lower DoD in November and a higher DoD value in December. This behavior change leads to a higher cost of \$521748.77, but also leads to a decrease in ESS degradation and CO_2 emissions by 3.43% and 60.55 tons of CO_2eq./kWh, respectively.



(a) MG behavior of H1/S1 solution.



(b) MG behavior of H0/S1 solution.

Figure 4.8: MG behavior analysis of H1/S1 and H0/S1 solutions selected from TOPSIS with equal preference over the three objectives.

4.2.3 Comparison between MEPS and SI-MEPS in the ERM problem

This section presents an analysis comparing the results of standard MEPS and SI-MEPS in the proposed multi-objective ERM problem of controlling the DoD of an ESS in a microgrid, with both solar and wind generation. In terms of configurations, both MEPS and SI-MEPS initialized the ANN population without any hidden layers with 8 output neurons for the 8 actions as given in equation 4.20. Besides, SI-MEPS employed the E1/V1/D1 version with a memory size equal to the population size. For a fair comparison, SI-MEPS was also fine-tuned. This fine-tuning followed the protocol described in Chapter 5, with the inclusion of MESH's mutation and recombination rates within the domains of $[0.1, 0.9]$ and $[0.1, 0.9]$, respectively. Finally, Table 4.6 lists the parameters used for both algorithms.

Table 4.6: Parameter initialization values used by the algorithms in the microgrid environment.

Description	MEPS				SI-MEPS			
	H1/S0	H1/S1	H0/S0	H0/S1	H1/S0	H1/S1	H0/S0	H0/S1
Population size	20				20			
Initial fraction selected from first front	0.5				0.5			
End generation number of heavy tail survivor selection	250				250			
Heavy tail selection parameter	1.0	0.3	-	-	0.8	0.5	-	-
"Add connection" mutation probability	0.6	0.3	0.5	0.4	0.5	0.3	0.6	0.4
"Add node" mutation probability	0.6	0.5	0.6	0.3	0.5	0.4	0.6	0.2
Parametrical mutation standard deviation	1.1	1.2	0.7	0.9	0.4	0.6	1.1	0.7
Mesh mutation rate	-				0.3	0.4	0.8	0.9
Mesh recombination rate	-				0.7	0.2	0.6	0.2
Personal guide size	-				3			
Adaptive Velocity γ	-				0.2			
Generations	500				100 + 400			
Episode length	12				12			

Similarly to Section 4.2.2, each algorithm was run 20 times with the same initialization parameters as presented in Table 4.6 and started with ESS at 20% SoC, in state $s_1 = \{0.2, 0.0, 0.0\}$. The reference point used for hypervolume calculation was $(600000, 65, 0.085)$. Table 4.7 details the results obtained on the test scenario and Figure 4.9 shows the complexity of the solutions from each algorithm in terms of number of nodes and number of connections.

Table 4.7: Performance of each algorithm regarding hypervolume when evaluated in the test scenario.

	Mean	Std.	Worst	Median	Best
$\text{MEPS}_{\text{H1/S0}}$	23073.24	94.59	22896.73	23053.54	23295.62
$\text{SI-MEPS}_{\text{H1/S0}}$	23037.79	58.43	22903.59	23032.61	23143.11
$\text{MEPS}_{\text{H1/S1}}$	23346.12	57.66	23249.00	23336.72	23470.91
$\text{SI-MEPS}_{\text{H1/S1}}$	23269.54	84.18	23076.46	23267.03	23417.45
$\text{MEPS}_{\text{H0/S0}}$	23077.56	82.06	22935.55	23077.32	23241.86
$\text{SI-MEPS}_{\text{H0/S0}}$	23079.68	92.60	22874.58	23075.48	23240.54
$\text{MEPS}_{\text{H0/S1}}$	**23351.87**	**51.16**	**23265.61**	**23347.58**	**23440.45**
$\text{SI-MEPS}_{\text{H0/S1}}$	23308.52	96.26	23098.38	23318.50	23439.75

Moreover, as done in the previous Section, a Kruskal-Wallis test with 1% significance level was applied to assess if there are any statistical differences among versions. Thereafter, a Wilcoxon signed-rank test with Holm-Bonferroni correction was applied to find out, by pairwise comparisons, which specific group's means were different. The results of the Kruskal-Wallis test attested that, with a higher confidence level of 99%, there are differences among the mean hypervolume values. From the results of the posthoc tests, it is possible to rank the algorithms as shown in Table 4.8.

Table 4.8: Ranking of algorithms based on Wilcoxon signed-rank test results using mean hypervolumes in the test scenario.

Rank	
1	2
$\text{MEPS}_{\text{H1/S1}}$	-
$\text{SI-MEPS}_{\text{H1/S1}}$	-
$\text{MEPS}_{\text{H0/S1}}$	-
$\text{SI-MEPS}_{\text{H0/S1}}$	-
-	$\text{MEPS}_{\text{H1/S0}}$
-	$\text{MEPS}_{\text{H0/S0}}$
-	$\text{SI-MEPS}_{\text{H1/S0}}$
-	$\text{SI-MEPS}_{\text{H0/S0}}$

It can be seen that, despite the improvements of SI-MEPS in the benchmark environments, only the H0/S0 SI-MEPS version was able to outperform the HV value of its standard counterpart in this MG environment. Furthermore, Figure 4.9 shows that, even though both H1/S1 and H0/S1 versions of MEPS versions are not statistically different, SI-MEPS solutions were composed by ANNs with more nodes and connections than the ones in standard MEPS. However, these results do not invalidate the proposed approach. By including MESH in the coupled approach, several new hyperparameters are included, resulting in another layer of complexity being added on top of MEPS. Moreover, different from the benchmark tests, there were fewer available generations to run each algorithm of the coupled approach in this MG environment. With respect to the parameters, SI-MEPS was fine-tuned as a coupled algorithm considering both MESH and MEPS parameters. As a result, future work may include a parameters analysis in which MEPS parameters are fixed and MESH parameters are fine-tuned. Finally, despite SI-MEPS presenting a promising approach, future research is necessary to both investigate the effects of the new hyperparameters, and evaluate its performance in different environments.

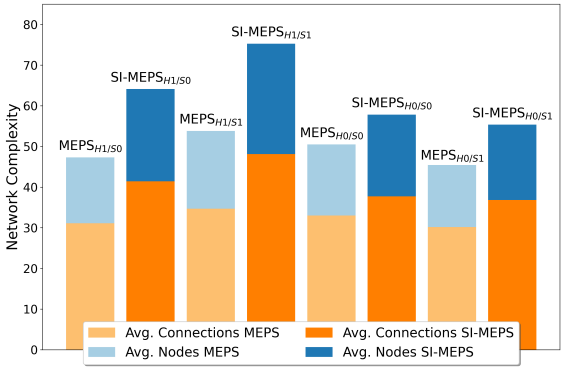


Figure 4.9: Complexity of the networks obtained by both MEPS versions in the proposed MG environment. Darker colors are used to indicate the complexities for SI-MEPS, while lighter colors indicate the complexities for standard MEPS.

Chapter 5

Hyperparameters Analysis of MEPS

Several machine learning algorithms are highly configurable through hyperparameters. These parameters often significantly affect the the complexity, behavior, speed, and most importantly, performance. Therefore, it is essential to carefully select an algorithm's hyperparameter values [14]. This chapter describes the analysis of hyperparameters performed on MEPS in the proposed MG environment.

Since MEPS features a set of hyperparameters that must be determined before applying the algorithm to any kind of problem, a hyperparameter analysis has been carried on using the MG environment presented in Section 4.2. In this analysis, a uniform random search for a specific set of hyperparameters within a given domain has been performed. The selected set of MEPS' hyperparameters, along with their domains, are the following:

- Heavy tail selection parameter $\alpha \in [0.1, 1.0]$;
- Parametrical mutation standard deviation $\sigma \in [0.1, 1.5]$;
- "Add connection" mutation probability $p_{ac} \in [0.1, 0.6]$;
- "Add node" mutation probability $p_{an} \in [0.1, 0.6]$;

The simulations have been conducted using the Optuna tool [3] in a computer with an Intel(R) Core(TM) i9-10900X CPU@3.70GHz, 64GB RAM, and Windows 10 Pro. Moreover, a total of 50 search steps have been performed, in which each step involves sampling a set of hyperparameters, and executing MEPS with the sampled set of hyperparameters for 20 independent runs. Finally, each set of hyperparameters was assigned a mean hypervolume value as well as a standard deviation value. Figures 5.1, 5.2, 5.3, 5.4 show both the mean and standard deviation values for the hypervolume obtained for each hyperparameter value using H1/S0, H1/S1, H0/S0, and H0/S1, respectively.

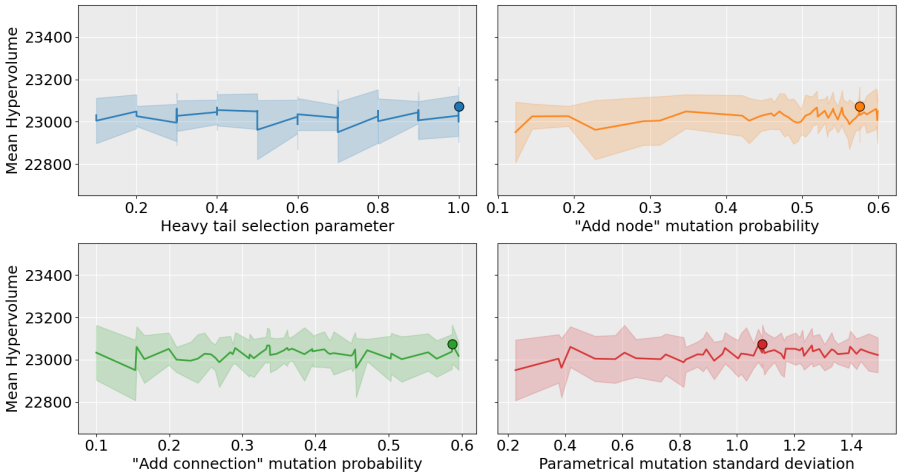


Figure 5.1: Performance associated with each hyperparameter value using H1/S0. Solid lines indicate the mean hypervolume, light areas indicate the standard deviation, and a circle marks the value that led to the highest mean hypervolume value.

Upon comparing the two versions that utilize the heavy tail selection parameter, it can be noted that higher values for the "Add node" parametrical mutation have resulted in higher mean HV values. However, the two versions differ in terms of the heavy tail and "Add connection" parameters. In H1/S1, lower values for both parameters are linked to higher HV values, whereas in H1/S0, the best performance is associated with higher values for these two parameters. This behavior change demonstrates how the values for the heavy tail selection parameter vary depending on the density measure used.

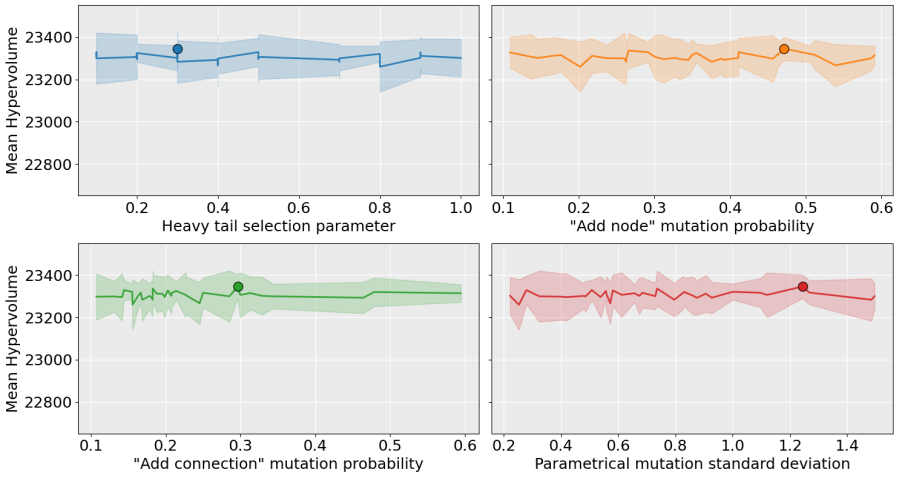


Figure 5.2: Performance associated with each hyperparameter value using H1/S1. Solid lines indicate the mean hypervolume, light areas indicate the standard deviation, and a circle marks the value that led to the highest mean hypervolume value.

Then, the versions without the heavy tail parameter, H0/S0 and H0/S1, are compared. Removing the heavy tail operator does not result in significant differences

for the "Add connection" mutation parameter between these versions. However, these versions differ in terms of the values of "Add node" and parametrical mutation parameters. When using crowding distance in H0/S0, both a high value for "Add node" mutation and a low value for parametrical mutation have led to the highest mean HV value. In contrast, the H0/S1 version that employs the hypervolume contribution as density measure has achieved the highest mean HV values with a low value for the "Add node" mutation and a high value for the parametrical mutation.

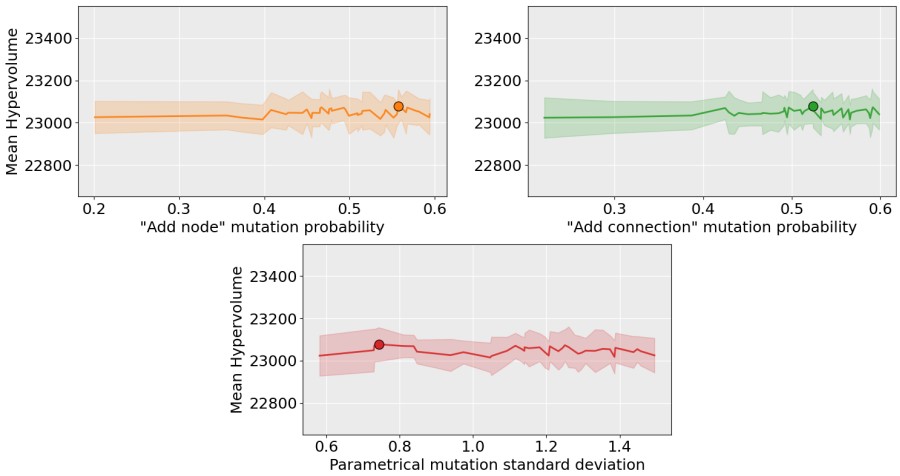


Figure 5.3: Performance associated with each hyperparameter value using H0/S0. Solid lines indicate the mean hypervolume, light areas indicate the standard deviation, and a circle marks the value that led to the highest mean hypervolume value.

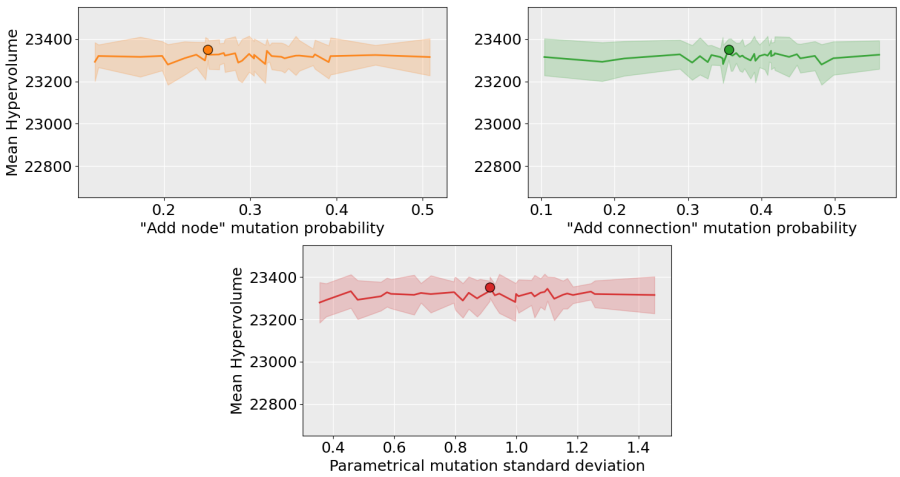


Figure 5.4: Performance associated with each hyperparameter value using H0/S1. Solid lines indicate the mean hypervolume, light areas indicate the standard deviation, and a circle marks the value that led to the highest mean hypervolume value.

To recommend a MEPS version and its corresponding hyperparameters, we analyzed only the best-performing sets. For each MEPS version, we calculated the

95th percentile of the mean HV value and filtered out hyperparameter sets with a mean HV value below this threshold. Table 5.1 details the hyperparameter values and their corresponding mean HV values for each version.

Table 5.1: Hyperparameter values associated with the mean hypervolume values above the 95th percentile per MEPS version. The values for the highest mean HV are in bold.

Version	Avg. HV	α	p_{ac}	p_{an}	σ
H1/S0	23073.24	1.0	0.6	0.6	1.1
	23067.15	0.4	0.3	0.5	1.1
	23067.13	0.7	0.3	0.6	1.3
H1/S1	23346.12	0.3	0.3	0.5	1.2
	23336.76	0.3	0.2	0.3	0.7
H0/S0	23077.56	-	0.5	0.6	0.7
	23073.28	-	0.6	0.5	1.3
H0/S1	**23351.87**	-	**0.4**	**0.3**	**0.9**
	23344.99	-	0.4	0.3	1.1

Among the different versions, it is worth noting the two versions that obtained the highest HV values, H1/S1 and H0/S1. In H1/S1, it was possible to determine a fixed value of 0.3 for the α parameter and ensure a performance superior to 95% of the executions. Moreover, the attained bound for the other parameters is tighter than the initial search bound. Although H0/S1 has one less parameter than its H1/S1 counterpart, it was still possible to set both the p_{ac} and p_{an} values to 0.4 and 0.3, respectively. As a result, the user is only required to define one parameter from the evaluated set. Furthermore, a user that chooses to use H0/S1 in this problem would need to determine a value for σ within the [0.9, 1.1] interval. Even though the two versions are not statistically different from each other (as presented in Section 4.2.2), there are significant differences in the number of hyperparameters that need to be determined by the user. In addition to not requiring the α parameter in H0/S1, the analysis determined optimal values for two additional hyperparameters. Due to requiring fewer hyperparameters to be set, we recommend using the H0/S1 version to solve the proposed problem.

Chapter 6

Conclusions

This thesis presented algorithmic solutions to energy resource management (ERM) problems modeled as single-objective and multi-objective problems. First, an evolutionary metaheuristic is enhanced with the addition of novel local search operators. This improved algorithm, namely aC-DEEPSO was employed in solving a single-objective risk-based ERM optimization problem in microgrids. Although presenting a suitable solution for the problem, aC-DEEPSO uses more function evaluations per generation than the other algorithms, due to the use of local search. This reduces the number of available evolutionary cycles compared to standard C-DEEPSO.

Afterwards, a novel multi-objective ERM decision problem model was proposed, which consider cost, CO2 emissions, and battery degradation. In this framework, a learning agent controls the depth of discharge of a Lithium-Ion battery. Then, a new multi-objective reinforcement learning algorithm was proposed to address this decision problem. The proposed algorithm, dubbed Multi-Objective Evolutionary Policy Search (MEPS), uses the NeuroEvolution of Augmenting Topologies structure to evolve artificial neural networks for estimating action-preference values considering multi-objective reward signals. The experimental results showed that MEPS is competitive against standard deep reinforcement learning techniques in both benchmark environments, and the proposed multi-objective ERM problem.

Lastly, this thesis evaluated the combination of MEPS with the Multi-objective Swarm Evolutionary Hybrid (MESH), and the proposed adaptive velocity operator from aC-DEEPSO. In this coupled approach, MESH acts as a depth initialization strategy that performs an initial search in the weights space to initialize the population. The MEPS then initializes using the obtained set of weights. The proposed approach was validated on three different benchmarks environments, in which not all MEPS versions achieved the maximum performance in terms of the hypervolume of the solution set. The proposed coupled algorithm, namely Swarm-Intelligent MEPS (SI-MEPS), not only improved the performance of MEPS but also achieved solutions with fewer nodes and connections. In spite of the competitive results in

the benchmark environments, SI-MEPS was not able to outperform standard MEPS in the proposed multi-objective ERM problem.

Overall, the current results obtained in this thesis have indicated a potential in providing algorithmic solutions to the problem of providing optimal control of microgrid systems with respect to either one or multiple objectives. Providing solutions that ensure a reliable and profitable operation of a microgrid system contributes towards society's decabornization goal. Moreover, it also helps to popularize the implementation of intelligent solutions in managing complex systems.

Since many real-world problems are inherently multi-objective, developing reinforcement learning algorithmic solutions for such problems is a current challenge. In addition, evolving neural networks is a highly active and important research topic that offers advantages over gradient-based optimization algorithms. Therefore, efficient neuroevolutionary methods can lead to an automation in the process of designing neural networks.

6.1 Contributions

This thesis demonstrated that coupling local search operators to an existing metaheuristic enhanced its performance in high-dimensional optimization problems. Furthermore, this study showcased the efficacy of evolving the topology and weights of artificial neural networks to address multi-objective reinforcement learning problems, regardless of the dimensionality of the state space. As a result of this research, the following works were published:

- **Leite, G. M. C.**, Marcelino, C. G., Pedreira, C. E., Jiménez-Fernández, S., & Salcedo-Sanz, S.. Evaluating the risk of uncertainty in smart grids with electric vehicles using an evolutionary swarm-intelligent algorithm. **Journal of Cleaner Production, 401, 136775**, 2023. doi: https://doi.org/10.1016/j.knosys.2023.111027;

- **Leite, G. M. C.**, Jiménez-Fernández, S., Salcedo-Sanz, S., Marcelino, C. G., & Pedreira, C. E.. Solving an energy resource management problem with a novel multi-objective evolutionary reinforcement learning method. **Knowledge-Based Systems, 280, 111027**, 2023. doi: https://doi.org/10.1016/j.jclepro.2023.136775;

6.2 Future Work

MEPS is designed to evolve neural networks with one output node for each available action. As the action space size increases, the network size also increases. However,

for the extremal case of an infinite action space size, the current version of MEPS cannot be used. To avoid discretization of continuous action spaces, a future research may investigate how to update MEPS to enable its application to problems with continuous action spaces.

The coupling of MESH and MEPS introduces several new hyperparameters and adds another layer of complexity on top of MEPS. Furthermore, there were fewer available generations to run each algorithm of the coupled approach in this MG environment, in contrast to the benchmark tests. Therefore, although SI-MEPS presents a promising approach, further research is necessary to investigate the effects of the new hyperparameters and evaluate their performance in different environments.

Regarding the crossover operator in neuroevolution, MESH addresses this issue by encoding weights and biases into vectors and performing crossover on them. However, the initial population is initialized with the same topology, meaning that every weight-encoded vector in MESH lies in the same dimension. A future research should investigate how to employ MESH for chromosomes with different lengths.

Finally, SI-MEPS was fine-tuned as a coupled algorithm considering both MESH and MEPS parameters. However, further studies are necessary to assess whether fine-tuning each part of SI-MEPS individually would lead to improved performance. Thus, a future work should examine the performance impacts of fixing SI-MEPS parameters to standard MEPS fine-tuned values, and uniformly searching MESH parameters.

References

[1] ABBASS, H. A., 2003, "Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization". In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, v. 3, pp. 2074–2080. IEEE.

[2] AHMAD, S., SHAFIULLAH, M., AHMED, C. B., et al., 2023, "A Review of Microgrid Energy Management and Control Strategies", *IEEE Access*.

[3] AKIBA, T., SANO, S., YANASE, T., et al., 2019, "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631.

[4] ALMEIDA, J., SOARES, J., LEZAMA, F., et al., 2022, "Robust Energy Resource Management Incorporating Risk Analysis Using Conditional Value-at-Risk", *IEEE Access*, v. 10, pp. 16063–16077. ISSN: 2169-3536. doi: 10.1109/ACCESS.2022.3147501.

[5] ALVARADO-BARRIOS, L., DEL NOZAL, A. R., VALERINO, J. B., et al., 2020, "Stochastic unit commitment in microgrids: Influence of the load forecasting error and the availability of energy storage", *Renewable Energy*, v. 146, pp. 2060–2069.

[6] ANTHONY, M., BARTLETT, P. L., BARTLETT, P. L., et al., 1999, *Neural network learning: Theoretical foundations*, v. 9. cambridge university press Cambridge.

[7] ARNOLD, B. C., 2014, "Pareto distribution", *Wiley StatsRef: Statistics Reference Online*, pp. 1–10. doi: https://doi.org/10.1002/9781118445112.stat01100.pub2.

[8] ARWA, E. O., FOLLY, K. A., 2020, "Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review", *Ieee Access*, v. 8, pp. 208992–209007.

[9] ASSOCIATION INTERNATIONALE POUR L'ÉVALUATION DU RENDEMENT SCOLAIRE, 2016, "Global EV outlook 2016: beyond one million electric cars". IEA.

[10] BARTLETT, P. L., 1998, "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network", *IEEE TRANSACTIONS ON INFORMATION THEORY*, v. 44, n. 2.

[11] BARTLETT, P. L., HARVEY, N., LIAW, C., et al., 2019, "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks", *The Journal of Machine Learning Research*, v. 20, n. 1, pp. 2285–2301.

[12] BAVARESCO, M. V., D'OCA, S., GHISI, E., et al., 2019, "Technological innovations to assess and include the human dimension in the building-performance loop: A review", *Energy and Buildings*, v. 202, pp. 109365.

[13] BAZARAA, M. S., SHERALI, H. D., SHETTY, C. M., 2013, *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

[14] BISCHL, B., BINDER, M., LANG, M., et al., 2023, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 13, n. 2, pp. e1484.

[15] BOZORG-HADDAD, O., SOLGI, M., LOÁICIGA, H. A., 2017, *Meta-heuristic and evolutionary algorithms for engineering optimization*. John Wiley & Sons.

[16] CANADIAN SOLAR, 2023. "HiKu". `https://www.csisolar.com/au/hiku/` Available in 11th july 2023.

[17] CANIZES, B., SOARES, J., VALE, Z., et al., 2019, "Optimal Distribution Grid Operation Using DLMP-Based Pricing for Electric Vehicle Charging Infrastructure in a Smart City", *Energies*, v. 12, n. 4, pp. 686. ISSN: 1996-1073. doi: 10.3390/en12040686.

[18] CHAKRABORTY, S., 2022, "TOPSIS and Modified TOPSIS: A comparative analysis", *Decision Analytics Journal*, v. 2, pp. 100021.

[19] CHEN, D., WANG, Y., GAO, W., 2020, "Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning", *Ap-*

plied Intelligence, v. 50, n. 10, pp. 3301–3317. ISSN: 1573-7497. doi: https://doi.org/10.1007/s10489-020-01702-7.

[20] CHOUDHURY, S., 2022, "Review of energy storage system technologies integration to microgrid: Types, control strategies, issues, and future prospects", *Journal of Energy Storage*, v. 48, pp. 103966.

[21] COELLO, C. A. C., LAMONT, G. B., VAN VELDHUIZEN, D. A., et al., 2007, *Evolutionary algorithms for solving multi-objective problems*, v. 5. Springer.

[22] COLLETTE, Y., SIARRY, P., 2004, *Multiobjective optimization: principles and case studies*. Springer Science & Business Media.

[23] COMMISSION, I. E., OTHERS, 2004, "IEC 61970: Energy management system application program interface (EMS-API)", *International Electrotechnical Commission (IEC): Geneva, Switzerland*.

[24] CONOVER, W. J., 1999, *Practical nonparametric statistics*, v. 350. John Wiley & Sons.

[25] CONTI, E., MADHAVAN, V., SUCH, F. P., et al., 2018, "Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents". In: *Advances in Neural Information Processing Systems*, v. 31, p. 5032–5043.

[26] COPELAND, B. J., 2004, *The essential turing*. Clarendon Press.

[27] DA SILVA, I. R., DE AL RABÊLO, R., RODRIGUES, J. J., et al., 2020, "A preference-based demand response mechanism for energy management in a microgrid", *Journal of Cleaner Production*, v. 255, pp. 120034.

[28] DAYAN, P., WATKINS, C., 1992, "Q-learning", *Machine learning*, v. 8, n. 3, pp. 279–292. ISSN: 1573-0565. doi: https://doi.org/10.1007/BF00992698.

[29] DE OLIVEIRA, T. H. F., DE SOUZA MEDEIROS, L. P., NETO, A. D. D., et al., 2021, "Q-Managed: A new algorithm for a multiobjective reinforcement learning", *Expert Systems with Applications*, v. 168, pp. 114228. ISSN: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.114228.

[30] DEB, K., 2001, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc.

[31] DEB, K., PRATAP, A., AGARWAL, S., et al., 2002, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, pp. 182–197. ISSN: 1941-0026. doi: https://doi.org/10.1109/4235.996017.

[32] DULAC-ARNOLD, G., LEVINE, N., MANKOWITZ, D. J., et al., 2021, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis", *Machine Learning*, v. 110, n. 9, pp. 2419–2468.

[33] EC, TRANSPORT, 2011, "Commission outlines ambitious plan to increase mobility and reduce emissions, European Commission". Brussels, Press Release.

[34] EGHBALI, N., HAKIMI, S. M., HASANKHANI, A., et al., 2022, "Stochastic energy management for a renewable energy based microgrid considering battery, hydrogen storage, and demand response", *Sustainable Energy, Grids and Networks*, v. 30, pp. 100652.

[35] EIBEN, A. E., SMITH, J. E., 2015, *Introduction to evolutionary computing*. Springer.

[36] EMMERICH, M., BEUME, N., NAUJOKS, B., 2005, "An EMO algorithm using the hypervolume measure as selection criterion". In: *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 62–76. ISBN: 9783540318804. doi: https://doi.org/10.1007/978-3-540-31880-4_5.

[37] FALCÓN-CARDONA, J. G., GÓMEZ, R. H., COELLO, C. A. C., et al., 2021, "Parallel multi-objective evolutionary algorithms: A comprehensive survey", *Swarm and Evolutionary Computation*, v. 67, pp. 100960.

[38] FOGEL, G. B., CORNE, D. W., 2003, *Evolutionary computation in bioinformatics*. Morgan Kaufmann.

[39] FOGEL, L. J., 1965, "Artificial intelligence through a simulation of evolution". In: *Proc. of the 2nd Cybernetics Science Symp., 1965*.

[40] FOULADFAR, M. H., SAEED, N., MARZBAND, M., et al., 2021, "Home-Microgrid Energy Management Strategy Considering EV's Participation in DR", *Energies*, v. 14, n. 18. ISSN: 1996-1073. doi: 10.3390/en14185971.

[41] FRANKLE, J., CARBIN, M., 2018, "The lottery ticket hypothesis: Finding sparse, trainable neural networks", *arXiv preprint arXiv:1803.03635*.

[42] GECAD, 2022. "GECAD Smart Grid Competition 2022". http://www.gecad.isep.ipp.pt/ERM-competitions/2022-2/ Available in 8th may 2024.

[43] GENDREAU, M., POTVIN, J.-Y., OTHERS, 2010, *Handbook of metaheuristics*, v. 2. Springer.

[44] GOMEZ, F., MIIKKULAINEN, R., 1997, "Incremental evolution of complex general behavior", *Adaptive Behavior*, v. 5, n. 3-4, pp. 317–342.

[45] GOMEZ, F., MIIKKULAINEN, R., 1998, "2-d pole balancing with recurrent evolutionary networks". In: *ICANN 98: Proceedings of the 8th International Conference on Artificial Neural Networks, Skövde, Sweden, 2–4 September 1998 8*, pp. 425–430. Springer.

[46] HATZIARGYRIOU, N., ASANO, H., IRAVANI, R., et al., 2007, "Microgrids", *IEEE power and energy magazine*, v. 5, n. 4, pp. 78–94.

[47] HAYES, C. F., RĂDULESCU, R., BARGIACCHI, E., et al., 2022, "A practical guide to multi-objective reinforcement learning and planning", *Autonomous Agents and Multi-Agent Systems*, v. 36, n. 1, pp. 1–59. ISSN: 1573-7454. doi: https://doi.org/10.1007/s10458-022-09552-y.

[48] HAYKIN, S., 2009, *Neural networks and learning machines, 3/E*. Pearson Education India.

[49] HOLLAND, J. H., 1973, "Genetic algorithms and the optimal allocation of trials", *SIAM journal on computing*, v. 2, n. 2, pp. 88–105.

[50] HOLM, S., 1979, "A Simple Sequentially Rejective Multiple Test Procedure", *Scandinavian Journal of Statistics*, v. 6, pp. 65–70. ISSN: 03036898, 14679469. Disponível em: <http://www.jstor.org/stable/4615733>.

[51] HORNIK, K., STINCHCOMBE, M., WHITE, H., 1989, "Multilayer feedforward networks are universal approximators", *Neural networks*, v. 2, n. 5, pp. 359–366. ISSN: 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8.

[52] HORNIK, K., STINCHCOMBE, M., WHITE, H., 1990, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks", *Neural networks*, v. 3, n. 5, pp. 551–560. ISSN: 0893-6080. doi: https://doi.org/10.1016/0893-6080(90)90005-6.

[53] HOSSAIN, M. A., POTA, H. R., SQUARTINI, S., et al., 2019, "Modified PSO algorithm for real-time energy management in grid-connected microgrids", *Renewable energy*, v. 136, pp. 746–757.

[54] HOSSAIN, M. A., POTA, H. R., SQUARTINI, S., et al., 2019, "Energy management of community microgrids considering degradation cost of battery", *Journal of Energy Storage*, v. 22, pp. 257–269.

[55] IRENA, RENEWABLE ENERGY STATISTICS, 2020, "The international renewable energy agency, abu dhabi", *Renewable Power Generation Costs in 2019*.

[56] KENNEDY, J., EBERHART, R., 1995, "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*, v. 4, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[57] KOTTATHRA, K., ATTIKIOUZEL, Y., 1996, "A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks", *Journal of Network and Computer Applications*, v. 19, n. 2, pp. 135–147.

[58] KOZA, J. R., 1994, "Genetic programming as a means for programming computers by natural selection", *Statistics and computing*, v. 4, pp. 87–112.

[59] KRUSKAL, W. H., WALLIS, W. A., 1952, "Use of ranks in one-criterion variance analysis", *Journal of the American statistical Association*, v. 47, n. 260, pp. 583–621.

[60] LEITE, G., JIMÉNEZ-FERNÁNDEZ, S., SALCEDO-SANZ, S., et al., 2023, "Solving an energy resource management problem with a novel multi-objective evolutionary reinforcement learning method", *Knowledge-Based Systems*, v. 280, pp. 111027.

[61] LEITE, G., MARCELINO, C., PEDREIRA, C., et al., 2023, "Evaluating the risk of uncertainty in smart grids with electric vehicles using an evolutionary swarm-intelligent algorithm", *Journal of Cleaner Production*, v. 401, pp. 136775. ISSN: 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2023.136775.

[62] LI, C., ZHENG, P., YIN, Y., et al., 2023, "An AR-assisted Deep Reinforcement Learning-based approach towards mutual-cognitive safe human-robot interaction", *Robotics and Computer-Integrated Manufacturing*, v. 80, pp. 102471. ISSN: 0736-5845. doi: https://doi.org/10.1016/j.rcim.2022.102471.

[63] LI, J., YAO, L., XU, X., et al., 2020, "Deep reinforcement learning for pedestrian collision avoidance and human-machine cooperative driving", *Information*

Sciences, v. 532, pp. 110–124. ISSN: 0020-0255. doi: https://doi.org/10.1016/j.ins.2020.03.105.

[64] LI, T., YANG, D., XIE, X., 2023, "Prioritized experience replay based reinforcement learning for adaptive tracking control of autonomous underwater vehicle", *Applied Mathematics and Computation*, v. 443, pp. 127734. ISSN: 0096-3003. doi: https://doi.org/10.1016/j.amc.2022.127734.

[65] LI, W., CUI, H., NEMETH, T., et al., 2021, "Cloud-based health-conscious energy management of hybrid battery systems in electric vehicles with deep reinforcement learning", *Applied Energy*, v. 293, pp. 116977.

[66] LIPU, M. H., ANSARI, S., MIAH, M. S., et al., 2022, "A review of controllers and optimizations based scheduling operation for battery energy storage system towards decarbonization in microgrid: Challenges and future directions", *Journal of Cleaner Production*, p. 132188.

[67] LIU, F., LIU, Q., TAO, Q., et al., 2023, "Deep reinforcement learning based energy storage management strategy considering prediction intervals of wind power", *International Journal of Electrical Power & Energy Systems*, v. 145, pp. 108608.

[68] LIU, G.-P., KADIRKAMANATHAN, V., 1995, "Learning with multi-objective criteria", .

[69] LOLLA, P. R., RANGU, S. K., DHENUVAKONDA, K. R., et al., 2022, "Optimal day ahead energy consumption management in grid-connected microgrids", *International Journal of Energy Research*, v. 46, n. 2, pp. 1864–1881. ISSN: 1099-114X. doi: 10.1002/er.7303.

[70] LONI, M., SINAEI, S., ZOLJODI, A., et al., 2020, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems", *Microprocessors and Microsystems*, v. 73, pp. 102989.

[71] LÓPEZ-IBÁÑEZ, M., DUBOIS-LACOSTE, J., CÁCERES, L. P., et al., 2016, "The irace package: Iterated racing for automatic algorithm configuration", *Operations Research Perspectives*, v. 3, pp. 43–58.

[72] LU, H., CHEN, W., 2008, "Self-adaptive velocity particle swarm optimization for solving constrained optimization problems", *Journal of Global Optimization*, v. 41, n. 3, pp. 427–445. ISSN: 1573-2916. doi: 10.1007/s10898-007-9255-9.

[73] LUO, L., ABDULKAREEM, S. S., REZVANI, A., et al., 2020, "Optimal scheduling of a renewable based microgrid considering photovoltaic system and battery energy storage under uncertainty", *Journal of Energy Storage*, v. 28, pp. 101306.

[74] MAHESHWARI, A., PATERAKIS, N. G., SANTARELLI, M., et al., 2020, "Optimizing the operation of energy storage using a non-linear lithium-ion battery degradation model", *Applied Energy*, v. 261, pp. 114360. ISSN: 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2019.114360.

[75] MAHMOUDAN, A., SAMADOF, P., HOSSEINZADEH, S., et al., 2021, "A multigeneration cascade system using ground-source energy with cold recovery: 3E analyses and multi-objective optimization", *Energy*, v. 233, pp. 121185.

[76] MARCELINO, C., BAUMANN, M., CARVALHO, L., et al., 2020, "A combined Optimization and Decision-Making approach for Battery-Supported HMGS", *Journal of the Operational Research Society*, v. 71, n. 5, pp. 762–774. ISSN: 1476-9360. doi: 10.1080/01605682.2019.1582590.

[77] MARCELINO, C., BAUMANN, M., CARVALHO, L., et al., 2020, "A combined optimisation and decision-making approach for battery-supported HMGS", *Journal of the Operational Research Society*, v. 71, n. 5, pp. 762–774. ISSN: 0160-5682. doi: https://doi.org/10.1080/01605682.2019.1582590.

[78] MARCELINO, C. G., ALMEIDA, P. E. M., WANNER, E. F., et al., 2018, "Solving security constrained optimal power flow problems: a hybrid evolutionary approach", *Applied Intelligence*, v. 48, pp. 3672–3690. ISSN: 1573-7497. doi: 10.1007/s10489-018-1167-5.

[79] MARCELINO, C. G., LEITE, G. M. C., DELGADO, C. A. D. M., et al., 2021, "An efficient multi-objective evolutionary approach for solving the operation of multi-reservoir system scheduling in hydro-power plants", *Expert Systems with Applications*, v. 185, pp. 115638. ISSN: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115638.

[80] MARCELINO, C. G., LEITE, G. M. C., DELGADO, C. A. D. M., et al., 2021, "An efficient multi-objective evolutionary approach for solving the operation of multi-reservoir system scheduling in hydro-power plants", *Expert Systems with Applications*, v. 185, pp. 115638. ISSN: 0957-4174. doi: 10.1016/j.eswa.2021.115638.

[81] MARCELINO, C. G., LEITE, G. M. C., JIMÉNEZ-FERNÁNDEZ, S., et al., 2022, "An improved C-DEEPSO algorithm for optimal active-reactive power dispatch in microgrids with electric vehicles", *IEEE Access*, pp. 1–1. ISSN: 2169-3536. doi: 10.1109/ACCESS.2022.3203728.

[82] MARCELINO, C. G., LEITE, G. M. C., JIMÉNEZ-FERNÁNDEZ, S., et al., 2022, "An improved C-DEEPSO algorithm for optimal active-reactive power dispatch in microgrids with electric vehicles", *IEEE Access*, pp. 1–1. ISSN: 2169-3536. doi: 10.1109/ACCESS.2022.3203728.

[83] MARCELINO, C. G., LEITE, G. M. C., WANNER, E. F., et al., 2023, "Evaluating the use of a Net-Metering mechanism in microgrids to reduce power generation costs with a swarm-intelligent algorithm", *Energy*, v. 266, pp. 126317. ISSN: 0360-5442. doi: https://doi.org/10.1016/j.energy.2022.126317.

[84] MCINTYRE, A., KALLADA, M., MIGUEL, C. G., et al., 2019. "neat-python". `https://github.com/CodeReclaimers/neat-python`.

[85] MIRANDA, V., FONSECA, N., 2002, "EPSO-evolutionary particle swarm optimization, a new algorithm with applications in power systems". In: *IEEE/PES Transmission and Distribution Conference and Exhibition*, v. 2, pp. 745–750. IEEE.

[86] MISCHOS, S., DALAGDI, E., VRAKAS, D., 2023, "Intelligent energy management systems: a review", *Artificial Intelligence Review*, pp. 1–40.

[87] MNIH, V., KAVUKCUOGLU, K., SILVER, D., et al., 2013, "Playing atari with deep reinforcement learning", *arXiv preprint arXiv:1312.5602*. doi: https://doi.org/10.48550/arXiv.1312.5602.

[88] MNIH, V., KAVUKCUOGLU, K., SILVER, D., et al., 2015, "Human-level control through deep reinforcement learning", *nature*, v. 518, n. 7540, pp. 529–533. ISSN: 1476-4687. doi: https://doi.org/10.1038/nature14236.

[89] MONTANA, D. J., DAVIS, L., OTHERS, 1989, "Training feedforward neural networks using genetic algorithms." In: *IJCAI*, v. 89, pp. 762–767.

[90] MOSTAFA, M. H., ALEEM, S. H. E. A., ALI, S. G., et al., 2020, "Robust energy management and economic analysis of microgrids considering different battery characteristics", *IEEE Access*, v. 8, pp. 54751–54775.

[91] MUSALLAM, M., JOHNSON, C. M., 2012, "An efficient implementation of the rainflow counting algorithm for life consumption estimation", *IEEE*

Transactions on reliability, v. 61, n. 4, pp. 978–986. ISSN: 1558-1721. doi: https://doi.org/10.1109/TR.2012.2221040.

[92] NAZ, K., ZAINAB, F., MEHMOOD, K. K., et al., 2021, "An Optimized Framework for Energy Management of Multi-Microgrid Systems", *Energies*, v. 14, n. 19. ISSN: 1996-1073. doi: 10.3390/en14196012.

[93] NGUYEN, T. T., NGUYEN, N. D., VAMPLEW, P., et al., 2020, "A multi-objective deep reinforcement learning framework", *Engineering Applications of Artificial Intelligence*, v. 96, pp. 103915. ISSN: 0952-1976. doi: https://doi.org/10.1016/j.engappai.2020.103915.

[94] NORVENTO, 2023. `https://www.norvento.com/productos/aerogeneradores-de-media-potencia/` Available in 11th july 2023.

[95] NUTAKKI, M., MANDAVA, S., 2023, "Review on optimization techniques and role of Artificial Intelligence in home energy management systems", *Engineering Applications of Artificial Intelligence*, v. 119, pp. 105721.

[96] OH, E., WANG, H., 2020, "Reinforcement-learning-based energy storage system operation strategies to manage wind power forecast uncertainty", *IEEE Access*, v. 8, pp. 20965–20976.

[97] PADHY, N. P., 2004, "Unit commitment-a bibliographical survey", *IEEE Transactions on power systems*, v. 19, n. 2, pp. 1196–1205.

[98] PARETO, V., 1964, *Cours d'économie politique*, v. 1. Librairie Droz.

[99] PEREIRA, M., 1985, "Optimal scheduling of hydrothermal systems-an overview", *IFAC Proceedings Volumes*, v. 18, n. 7, pp. 1–9.

[100] PERNY, P., WENG, P., 2010, "On finding compromise solutions in multiobjective Markov decision processes". In: *ECAI 2010*, IOS Press, pp. 969–970.

[101] PUTERMAN, M. L., 2014, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[102] RECHENBERG, I., 1965, "Cybernetic solution path of an experimental problem", *Roy. Aircr. Establ., Libr. transl.*, v. 1122.

[103] RED ELÉCTRICA, 2022. "The Spanish Electricity System. Preliminary report 2021". `https://www.ree.es/en/datos/publications/annual-system-report/the-spanish-electricity-system-preliminary-report-2021` Available in 11th January 2022.

[104] REZAEE JORDEHI, A., 2021, "An improved particle swarm optimisation for unit commitment in microgrids with battery energy storage systems considering battery degradation and uncertainties", *International Journal of Energy Research*, v. 45, n. 1, pp. 727–744.

[105] ROBERTS, D., BROWN, S., 2022, "The economics of firm solar power from Li-ion and vanadium flow batteries in California", *MRS Energy & Sustainability*, v. 9, pp. 129–141. ISSN: 2329-2237. doi: https://doi.org/10.1557/s43581-022-00028-w.

[106] ROIJERS, D. M., VAMPLEW, P., WHITESON, S., et al., 2013, "A survey of multi-objective sequential decision-making", *Journal of Artificial Intelligence Research*, v. 48, pp. 67–113. ISSN: 1076-9757. doi: https://doi.org/10.1613/jair.3987.

[107] RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J., 1986, "Learning representations by back-propagating errors", *nature*, v. 323, n. 6088, pp. 533–536.

[108] SALIMANS, T., HO, J., CHEN, X., et al., 2017, "Evolution strategies as a scalable alternative to reinforcement learning", *arXiv preprint arXiv:1703.03864*.

[109] SARTI, S., OCHOA, G., 2021, "A NEAT visualisation of neuroevolution trajectories". In: *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pp. 714–728. ISBN: 9783030726997. doi: https://doi.org/10.1007/978-3-030-72699-7_45.

[110] SCHMIDT-HIEBER, JOHANNES, 2020, "NONPARAMETRIC REGRESSION USING DEEP NEURAL NETWORKS WITH RELU ACTIVATION FUNCTION", *The Annals of Statistics*, v. 48, n. 4, pp. 1875–1897.

[111] SHANG, Y., WU, W., GUO, J., et al., 2020, "Stochastic dispatch of energy storage in microgrids: An augmented reinforcement learning approach", *Applied Energy*, v. 261, pp. 114423.

[112] SILVER, D., SCHRITTWIESER, J., SIMONYAN, K., et al., 2017, "Mastering the game of go without human knowledge", *nature*, v. 550, n. 7676, pp. 354–359. ISSN: 1476-4687. doi: https://doi.org/10.1038/nature24270.

[113] SINGH, D., KUMAR, V., VAISHALI, et al., 2020, "Classification of COVID-19 patients from chest CT images using multi-objective differential evolution–based convolutional neural networks", *European Journal of Clinical Microbiology & Infectious Diseases*, v. 39, pp. 1379–1389.

[114] SLOWIK, A., KWASNICKA, H., 2020, "Evolutionary algorithms and their applications to engineering problems", *Neural Computing and Applications*, v. 32, pp. 12363–12379.

[115] SOARES, J., LEZAMA, F., ALEMEIDA, J., et al., 2022, "Guidelines for WCCI (CEC)/GECCO 2022 Competition Evolutionary Computation in the Energy Domain: Risk-based Energy Scheduling", .

[116] SOLCAST, 2019. "Global solar irradiance data and PV system power output data". `https://solcast.com/` Available in 5th january 2024.

[117] STANLEY, K. O., MIIKKULAINEN, R., 2002, "Evolving neural networks through augmenting topologies", *Evolutionary computation*, v. 10, n. 2, pp. 99–127. ISSN: 1063-6560. doi: https://doi.org/10.1162/106365602320169811.

[118] STANLEY, K. O., CLUNE, J., LEHMAN, J., et al., 2019, "Designing neural networks through neuroevolution", *Nature Machine Intelligence*, v. 1, n. 1, pp. 24–35. ISSN: 2522-5839. doi: https://doi.org/10.1038/s42256-018-0006-z.

[119] STEIN, G., GONZALEZ, A. J., BARHAM, C., 2015, "Combining NEAT and PSO for learning tactical human behavior", *Neural Computing and Applications*, v. 26, pp. 747–764.

[120] SUCH, F. P., MADHAVAN, V., CONTI, E., et al., 2017, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning", *arXiv preprint arXiv:1712.06567*.

[121] SUN, X., XU, Z., QIU, J., et al., 2023, "Optimal Volt/Var Control for Unbalanced Distribution Networks With Human-in-the-Loop Deep Reinforcement Learning", *IEEE Transactions on Smart Grid*.

[122] SUTTON, R. S., 1988, "Learning to predict by the methods of temporal differences", *Machine learning*, v. 3, n. 1, pp. 9–44. ISSN: 1573-0565. doi: https://doi.org/10.1007/BF00115009.

[123] SUTTON, R. S., BARTO, A. G., 2018, *Reinforcement learning: An introduction*. MIT press.

[124] TANG, X., ZHOU, H., WANG, F., et al., 2022, "Longevity-conscious energy management strategy of fuel cell hybrid electric Vehicle Based on deep reinforcement learning", *Energy*, v. 238, pp. 121593.

[125] TSITSIKLIS, J. N., 1994, "Asynchronous stochastic approximation and Q-learning", *Machine learning*, v. 16, n. 3, pp. 185–202. ISSN: 1573-0565. doi: https://doi.org/10.1023/A:1022689125041.

[126] TZENG, G.-H., HUANG, J.-J., 2011, *Multiple attribute decision making: methods and applications*. CRC press.

[127] UNECE, 2022. "Life Cycle Assessment of Electricity Generation Options". `https://unece.org/sed/documents/2021/10/reports/life-cycle-assessment-electricity-generation-options` Available in 16th march 2022.

[128] VALENCIA-RIVERA, G. H., AMAYA, I., CRUZ-DUARTE, J. M., et al., 2021, "Hybrid Controller Based on LQR Applied to Interleaved Boost Converter and Microgrids under Power Quality Events", *Energies*, v. 14, n. 21. ISSN: 1996-1073. doi: 10.3390/en14216909.

[129] VAMPLEW, P., YEARWOOD, J., DAZELEY, R., et al., 2008, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts". In: *Australasian joint conference on artificial intelligence*, pp. 372–378. ISBN: 9783540893783. doi: https://doi.org/10.1007/978-3-540-89378-3_37.

[130] VAMPLEW, P., DAZELEY, R., BERRY, A., et al., 2011, "Empirical evaluation methods for multiobjective reinforcement learning algorithms", *Machine learning*, v. 84, n. 1, pp. 51–80. ISSN: 1573-0565. doi: https://doi.org/10.1007/s10994-010-5232-5.

[131] VAMPLEW, P., DAZELEY, R., FOALE, C., 2017, "Softmax exploration strategies for multiobjective reinforcement learning", *Neurocomputing*, v. 263, pp. 74–86. ISSN: 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.09.141.

[132] VAN HASSELT, H., GUEZ, A., SILVER, D., 2016, "Deep reinforcement learning with double q-learning". In: *Proceedings of the AAAI conference on artificial intelligence*, v. 30, p. 2094–2100. doi: https://doi.org/10.1609/aaai.v30i1.10295.

[133] VAN MOFFAERT, K., NOWÉ, A., 2014, "Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies", *Journal of Machine Learning Research*, v. 15, n. 107, pp. 3663–3692.

[134] VAPNIK, V. N., CHERVONENKIS, A. Y., 2015, "On the uniform convergence of relative frequencies of events to their probabilities". In: *Measures of complexity: festschrift for alexey chervonenkis*, Springer, pp. 11–30.

[135] VINYALS, O., BABUSCHKIN, I., CZARNECKI, W. M., et al., 2019, "Grandmaster level in StarCraft II using multi-agent reinforcement learning", *Nature*, v. 575, n. 7782, pp. 350–354. ISSN: 1476-4687. doi: https://doi.org/10.1038/s41586-019-1724-z.

[136] WANG, Z., SCHAUL, T., HESSEL, M., et al., 2016, "Dueling network architectures for deep reinforcement learning". In: *International conference on machine learning*, v. 48, pp. 1995–2003.

[137] WATKINS, C. J. C. H., 1989. "Learning from delayed rewards". .

[138] WHITE, D. J., 1982, "Multi-objective infinite-horizon discounted Markov decision processes", *Journal of Mathematical Analysis and Applications*, v. 89, n. 2, pp. 639–647. ISSN: 0022-247X. doi: https://doi.org/10.1016/0022-247X(82)90122-6.

[139] WIERING, M. A., VAN OTTERLO, M., 2012, "Reinforcement learning", *Adaptation, learning, and optimization*, v. 12, n. 3, pp. 729.

[140] WU, Q., FENG, Q., REN, Y., et al., 2021, "An intelligent preventive maintenance method based on reinforcement learning for battery energy storage systems", *IEEE Transactions on Industrial Informatics*, v. 17, n. 12, pp. 8254–8264.

[141] YAN, H., ZHANG, D., QILU, et al., 2021, "A review of spinel lithium titanate (Li4Ti5O12) as electrode material for advanced energy storage devices", *Ceramics International*, v. 47, n. 5, pp. 5870–5895. ISSN: 0272-8842. doi: https://doi.org/10.1016/j.ceramint.2020.10.241.

[142] YAN, S., WU, Z., WANG, J., et al., 2023, "Real-World Learning Control for Autonomous Exploration of a Biomimetic Robotic Shark", *IEEE Transactions on Industrial Electronics*, v. 70, n. 4, pp. 3966–3974. ISSN: 1557-9948. doi: https://doi.org/10.1109/TIE.2022.3174306.

[143] YAO, X., LIU, Y., 1998, "Towards designing artificial neural networks by evolution", *Applied Mathematics and Computation*, v. 91, n. 1, pp. 83–90.

[144] YAROTSKY, D., 2017, "Error bounds for approximations with deep ReLU networks", *Neural Networks*, v. 94, pp. 103–114.

[145] YOO, S., KIM, C., CHOI, J., et al., 2023, "GIN: Graph-Based Interaction-Aware Constraint Policy Optimization for Autonomous Driving", *IEEE Robotics and Automation Letters*, v. 8, n. 2, pp. 464–471. ISSN: 2377-3766. doi: https://doi.org/10.1109/LRA.2022.3227862.

[146] YUE, C., SUGANTHAN, P. N., LIANG, J., et al., 2021, "Differential evolution using improved crowding distance for multimodal multiobjective optimization", *Swarm and Evolutionary Computation*, v. 62, pp. 100849. ISSN: 2210-6502. doi: https://doi.org/10.1016/j.swevo.2021.100849.

[147] ZHANG, K., ZHANG, J., XU, P.-D., et al., 2021, "Explainable AI in deep reinforcement learning models for power system emergency control", *IEEE Transactions on Computational Social Systems*, v. 9, n. 2, pp. 419–427.

[148] ZHANG, Z., ZHANG, D., QIU, R. C., 2019, "Deep reinforcement learning for power system applications: An overview", *CSEE Journal of Power and Energy Systems*, v. 6, n. 1, pp. 213–225.

[149] ZIA, M. F., ELBOUCHIKHI, E., BENBOUZID, M., 2018, "Microgrids energy management systems: A critical review on methods, solutions, and prospects", *Applied energy*, v. 222, pp. 1033–1055.

[150] ZITZLER, E., DEB, K., THIELE, L., 2000, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary computation*, v. 8, n. 2, pp. 173–195. ISSN: 1063-6560. doi: https://doi.org/10.1162/106365600568202.