

Jayaraman Valadi
Krishna Pratap Singh
Muneendra Ojha
Patrick Siarry *Editors*

Advanced Machine Learning with Evolutionary and Metaheuristic Techniques



Springer

Computational Intelligence Methods and Applications

Founding Editors

Sanghamitra Bandyopadhyay, Machine Intelligence Unit
Indian Statistical Institute, Kolkata,
West Bengal, India

Ujjwal Maulik, Dept of Computer Science & Engineering
Jadavpur University, Kolkata,
West Bengal, India

Patrick Siarry, LISSI
University of Paris-Est Créteil, Créteil,
France

Series Editor

Patrick Siarry, LiSSI, E.A. 3956, Université Paris-Est Créteil,
Vitry-sur-Seine, France

The monographs and textbooks in this series explain methods developed in computational intelligence (including evolutionary computing, neural networks, and fuzzy systems), soft computing, statistics, and artificial intelligence, and their applications in domains such as heuristics and optimization; bioinformatics, computational biology, and biomedical engineering; image and signal processing, VLSI, and embedded system design; network design; process engineering; social networking; and data mining.

Jayaraman Valadi • Krishna Pratap Singh •

Muneendra Ojha • Patrick Siarry

Editors

Advanced Machine Learning with Evolutionary and Metaheuristic Techniques



Springer

Editors

Jayaraman Valadi
Computing and Data Sciences
FLAME University
Pune, India

Krishna Pratap Singh
Indian Institute of Information Technology
Allahabad
Prayagraj, Uttar Pradesh, India

Muneendra Ojha
Indian Institute of Information Technology
Allahabad
Prayagraj, Uttar Pradesh, India

Patrick Siarry
University Paris-Est Créteil
Créteil, France

ISSN 2510-1765

ISSN 2510-1773 (electronic)

Computational Intelligence Methods and Applications

ISBN 978-981-99-9717-6

ISBN 978-981-99-9718-3 (eBook)

<https://doi.org/10.1007/978-981-99-9718-3>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd. The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Paper in this product is recyclable.

Preface

In this edited book *Advanced Machine Learning with Evolutionary and Metaheuristic Techniques*, we explore the intricate synergy between the domains of optimization and machine learning, with a focus on the cutting-edge techniques rooted in evolutionary and metaheuristic approaches. In today's rapidly evolving technological landscape, the realms of machine learning and evolutionary optimization techniques have forged powerful alliances that are useful for solving real-life problems.

The opening chapters provide a thorough understanding of Evolutionary Algorithms (EA) and Machine Learning (ML), discussing the historical context and evolution of both fields to lay the groundwork for the synergistic approach explored in the following chapters. In chapters “[Evolutionary Dynamic Optimization and Machine Learning](#)”, “[Evolutionary Techniques in Making Efficient Deep-Learning Framework: A Review](#)”, “[Integrating Particle Swarm Optimization with Reinforcement Learning: A Promising Approach to Optimization](#)” and “[Synergies Between Natural Language Processing and Swarm Intelligence Optimization: A Comprehensive Overview](#)”, readers may learn how to combine EA with ML approaches to solve complicated issues creatively. While chapter “[Heuristics-Based Hyperparameter Tuning for Transfer Learning Algorithms](#)” covers how to use hyperparameter tuning to increase the efficiency of machine learning algorithms, chapter “[Machine Learning Applications of Evolutionary and Metaheuristic Algorithms](#)” offers a quick overview of how EA is used in machine learning applications. The latter part of the book is dedicated to real-world case studies.

The following is a synopsis of individual chapter's content that might be more helpful to readers.

Chapter “[From Evolution to Intelligence: Exploring the Synergy of Optimization and Machine Learning](#)” delves into the fundamental principles of machine learning and evolutionary optimization. This chapter further helps the reader discover how machine learning algorithms harness the power of data to construct predictive and descriptive models, while evolutionary optimization algorithms mimic the process of

evolution to iteratively refine potential solutions, ultimately seeking optimal or near-optimal outcomes.

Chapter “[Metaheuristic and Evolutionary Algorithms in Explainable Artificial Intelligence](#)” explores the realm of Explainable Artificial Intelligence (XAI). The transparency and interpretability of machine learning models have become critical concerns in high-stakes decision-making scenarios. This chapter reveals how the combination of evolutionary and metaheuristic techniques offer promising solutions to make AI models more understandable and transparent.

Chapter “[Evolutionary Dynamic Optimization and Machine Learning](#)” dives into the world of Evolutionary Dynamic Optimization (EDO) and its reciprocal integration with Machine Learning. EDO holds the key to addressing complex, dynamic objective functions and provides valuable insights for enhancing machine learning processes.

Chapter “[Evolutionary Techniques in Making Efficient Deep-Learning Framework: A Review](#)” brings a fresh perspective to deep learning. By incorporating evolutionary algorithms and metaheuristics, it discusses optimization of deep learning models, fine-tuning model architecture and hyperparameters to improve performance and efficiency.

Chapter “[Integrating Particle Swarm Optimization with Reinforcement Learning: A Promising Approach to Optimization](#)” illustrates the integration of Particle Swarm Optimization (PSO) with Reinforcement Learning (RL), offering a promising approach to optimization. This unique combination provides solutions to complex optimization problems and sets the stage for further advancements in both fields.

Chapter “[Synergies Between Natural Language Processing and Swarm Intelligence Optimization: A Comprehensive Overview](#)” uncovers the synergies between Natural Language Processing (NLP) and Swarm Intelligence Optimization (SI). Readers can further discover the latest developments in NLP models and SI algorithms, which are revolutionizing how machines comprehend and generate human languages.

In chapter “[Heuristics-Based Hyperparameter Tuning for Transfer Learning Algorithms](#)”, the utilization of PSO is employed for the purpose of fine-tuning hyperparameters within the transfer learning framework.

Chapter “[Machine Learning Applications of Evolutionary and Metaheuristic Algorithms](#)” provides a brief introduction to applications of evolutionary and metaheuristic algorithms to real-world applications, unveiling the symbiotic relationship between population-based metaheuristic algorithms and machine learning. Through examples and case studies, the reader will gain insight into how these algorithms solve complex engineering problems and drive advancements in various domains, from classification to clustering and beyond.

Chapter “[Machine Learning Assisted Metaheuristic Based Optimization of Mixed Suspension Mixed Product Removal Process](#)” illustrates how machine learning assists in the optimization of the Mixed-Suspension Mixed-Product Removal (MSMPR) crystallization process in the pharmaceutical industry. It discusses the methods to achieve real-time optimization through the use of data-based modeling

and surrogate techniques, reducing the computational burden of detailed models and opening the door to faster, more efficient processes.

The security of IoT networks takes center stage in chapter “[Machine Learning Based Intelligent RPL Attack Detection System for IoT Networks](#)”. This chapter addresses the vulnerability of routing protocols and describes an intelligent Intrusion Detection System (IDS) that utilizes neural networks and genetic algorithms to protect against attacks on IoT networks.

Chapter “[Shallow and Deep Evolutionary Neural Networks Applications in Solid Mechanics](#)” focuses on the application of shallow and deep evolutionary neural networks in solid mechanics. Readers can learn how data-driven models and evolutionary algorithms work together to optimize multi-objective problems in materials mechanics.

Chapter “[Polymer and Nanocomposite Informatics: Recent Applications of Artificial Intelligence and Data Repositories](#)” explores the burgeoning field of polymer and nanocomposite informatics. Here, artificial intelligence and machine learning play a pivotal role in designing and discovering polymers with desirable properties, promising advances in efficiency and the development of new materials.

Our journey concludes in chapter “[Synergistic Combination of Machine Learning and Evolutionary and Heuristic Algorithms for Handling Imbalance in Biological and Biomedical Datasets](#)” which highlights the importance of handling imbalance in biological data mining. This chapter further explores the use of Nature-Inspired Evolutionary and Metaheuristic algorithms to address data imbalance, providing clear explanations and relevant case studies.

The chapters in this book represent the collaborative efforts of experts in the fields of optimization, machine learning, and artificial intelligence. They provide a comprehensive view of how these domains come together to address real-world challenges.

We invite readers to embark on this enlightening journey, uncovering the advanced machine learning techniques empowered by evolutionary and metaheuristic approaches. The insights and knowledge contained within these pages offer a glimpse into the future of problem-solving and the remarkable potential of these collaborative methodologies.

Pune, India

Prayagraj, Uttar Pradesh, India

Prayagraj, Uttar Pradesh, India

Créteil, France

Jayaraman Valadi

Krishna Pratap Singh

Muneendra Ojha

Patrick Siarry

Contents

From Evolution to Intelligence: Exploring the Synergy of Optimization and Machine Learning	1
Kedar Nath Das and Rahul Paul	
Metaheuristic and Evolutionary Algorithms in Explainable Artificial Intelligence	33
Hardik Prabhu, Aamod Sane, Renu Dhadwal, Patrick Siarry, and Jayaraman Valadi	
Evolutionary Dynamic Optimization and Machine Learning	67
Abdenour Boulesnane	
Evolutionary Techniques in Making Efficient Deep-Learning Framework: A Review	87
Shubham Joshi, Millie Pant, and Kusum Deep	
Integrating Particle Swarm Optimization with Reinforcement Learning: A Promising Approach to Optimization	105
Arindam Ghosh, Ojaswita Tiwari, Krishna Pratap Singh, and Muneendra Ojha	
Synergies Between Natural Language Processing and Swarm Intelligence Optimization: A Comprehensive Overview	121
Ujwala Bharambe, Rekha Ramesh, Manimala Mahato, and Sangita Chaudhari	
Heuristics-Based Hyperparameter Tuning for Transfer Learning Algorithms	153
Upendra Pratap Singh, Krishna Pratap Singh, and Muneendra Ojha	
Machine Learning Applications of Evolutionary and Metaheuristic Algorithms	185
Anupam Yadav and Shrishti Chamoli	

Machine Learning Assisted Metaheuristic Based Optimization of Mixed Suspension Mixed Product Removal Process	213
Ravi Kiran Inapakurthi, Sakshi S. Naik, and Kishalay Mitra	
Machine Learning Based Intelligent RPL Attack Detection System for IoT Networks	241
A. Kannan, M. Selvi, S. V. N. Santhosh Kumar, K. Thangaramya, and S. Shalini	
Shallow and Deep Evolutionary Neural Networks Applications in Solid Mechanics	257
Anna Malá, Zdeněk Padovc, Tomáš Mareš, and Nirupam Chakraborti	
Polymer and Nanocomposite Informatics: Recent Applications of Artificial Intelligence and Data Repositories	297
Neelesh Ashok, K. P. Soman, Madhav Samanta, M. S. Sruthi, Prabaharan Poornachandran, Suja Devi V. G, and N. Sukumar	
Synergistic Combination of Machine Learning and Evolutionary and Heuristic Algorithms for Handling Imbalance in Biological and Biomedical Datasets	323
Sonal Modak, Mayur Pandya, Patrick Siarry, and Jayaraman Valadi	

From Evolution to Intelligence: Exploring the Synergy of Optimization and Machine Learning



Kedar Nath Das and Rahul Paul

Abstract The rapid advancements in machine learning (ML) and evolutionary optimization techniques (EOT) have opened up new avenues for solving complex problems across diverse scientific domains. ML algorithms utilize a provided dataset to construct a proficient predictive or descriptive model. Evolutionary optimization algorithms, on the other hand, are a class of metaheuristic optimization methods that simulate the process of evolution by refining a population of potential solutions iteratively over generations to discover optimal or nearly optimal solutions to complex problems. A multitude of proposals have been sequentially put forth to address optimization problems/methodologies within the realm of ML. It is imperative to conduct a thorough examination and implementation of evolutionary optimization methods within the context of ML in order to provide direction for the advancement of research in both optimization and ML. Hence, this chapter provides a focused examination of the intersection between ML and EOT, which have been utilized in tandem to tackle a diverse array of scientific challenges. Furthermore, the present chapter delves into the potential areas of investigation that could enhance the efficacy of ML-EOT models.

Keywords Machine learning · Optimization · Evolutionary optimization techniques · Hyperparameter · Regularization · Neural network · Gaussian process · Correlation coefficient · Bayesian technique

K. N. Das (✉) · R. Paul

Department of Mathematics, National Institute of Technology Silchar, Silchar, Assam, India
e-mail: kedarnath@math.nits.ac.in; rahul22_rs@math.nits.ac.in

1 Introduction

ML and EOT are two related fields in the domain of artificial intelligence (AI) that have been increasingly popular in recent years. ML is a field of study that utilizes algorithms to facilitate learning from data, thereby enabling machines to enhance their performance over time without the need for explicit programming. Similarly, EOT is a collection of algorithms to optimize a single/multiple objective functions under certain constraints. Evolutionary optimization techniques are essentially iterative simulations of natural processes that enable solutions to improve progressively.

The ML-EOT combination has the potential to generate highly effective and efficient learning systems. EOT has the capacity to improve the accuracy and generalization abilities of ML models by refining their parameters and structures. The application of evolutionary machine learning has proven to be successful in various domains such as image classification [1], natural language processing [2], and game playing [3].

In addition, researchers are more interested in the advancement of algorithms that possess the ability to adapt and evolve over time. This feature holds significant potential in scenarios characterized by dynamic environments, where the most favorable solution may undergo changes over time. The integration of ML and EOT presents a viable approach for developing algorithms that can dynamically adjust to varying conditions and enhance their efficacy through continuous optimization.

Furthermore, constrained optimization problems have gained significant importance in research owing to the prevalence of multiple constraints in real-world problems. The application of evolutionary algorithms to address constrained optimization problems is a widely adopted practise in the evolutionary computation field. The integration of evolutionary algorithms and learning techniques has been demonstrated to be a viable approach in attaining favorable outcomes in the context of constrained optimization problems.

Existing literature indicates that a significant quantity of research articles focusing on the utilization of EOT in the context of ML have been referenced in recent decades. Nevertheless, the works have been consistently presented in a disorganized manner over the years. In this chapter, an endeavor has been made to conduct a critical analysis of the relevant articles so that future researchers can make sense of the past knowledge. This chapter attempts to convey a comprehensive and up-to-date review of the EOT used in ML perspectives, despite the fact that there are only a handful of similar reviews published to date [4, 5]. The objective is therefore to demonstrate how EOT and ML can work in tandem to improve scientific research performance.

The chapter is organized into the following sections. Section 2 provides the fundamental concepts of EOT and ML. Section 3 provides a comprehensive analysis of how researchers effectively combined ML and EOT to achieve optimal performance in diverse scientific domains. The chapter is concluded in Sect. 4, which discusses the challenges and unresolved matters in the domain of ML and EOT.

2 Preliminary Concepts

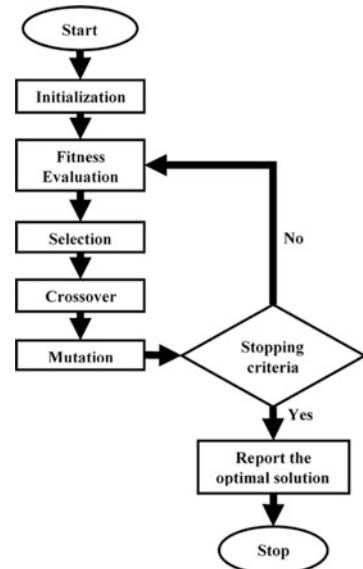
2.1 Evolutionary Optimization Techniques

EOT are a class of computational methods that draw inspiration from the natural evolutionary process to identify optimal solutions for intricate problems. The efficacy of these techniques has gained significant traction in contemporary times owing to their capacity to identify optimal solutions across diverse application domains. The prevailing EOT that are frequently employed encompass Genetic Algorithms (GA) [6], Particle Swarm Optimization (PSO) [7], Differential Evolution (DE) [8], Artificial Bee Colony (ABC) [9], Ant Colony Optimization (ACO) [10], and Evolution Strategies (ES) [11].

GA are the most popular and well-known EOT. They are based on the principle of natural selection and simulate the evolutionary process by generating a population of candidate solutions and employing selection, crossover, and mutation operators to produce new offspring. Using a fitness function that assesses how well each candidate solution solves the problem at hand, the quality of the solutions is determined. Figure 1 shows the stepwise process of the functioning of GA.

The movement of flocks of birds or schools of fish inspired the popular optimization technique PSO, whose flowchart is given in Fig. 2. It operates by generating a swarm of particles that move throughout a search space, adjusting their positions based on the positions of the finest solutions discovered to the time. PSO is a highly effective method for solving complex nonlinear optimization problems. In the flowchart given below N denotes number of particles and T denotes number of iterations. $x_i^{(t+1)}$ and $v_i^{(t+1)}$ denotes the respective updated position and velocity of

Fig. 1 Flowchart of GA



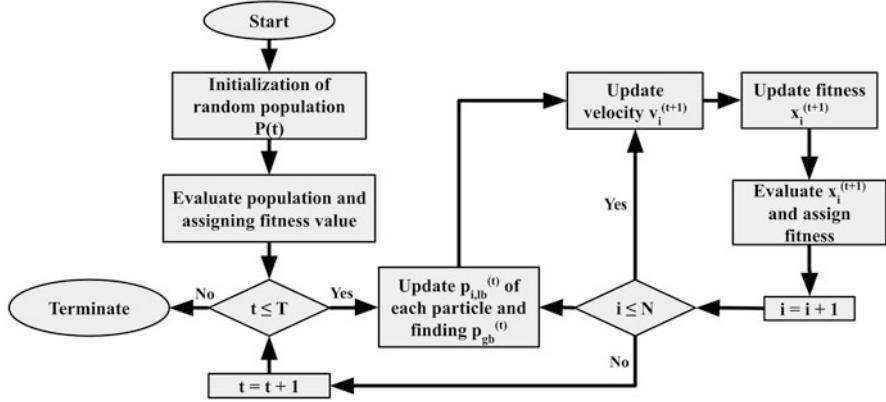
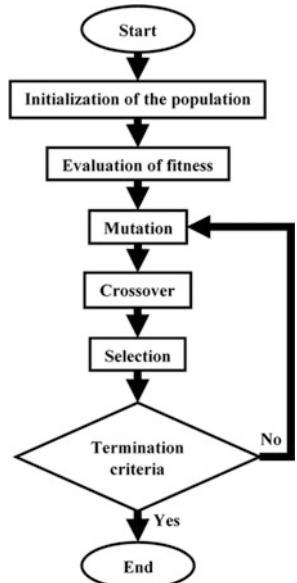


Fig. 2 Flowchart of PSO

Fig. 3 Flowchart of DE



the i -th particle in the $(t + 1)$ -th iteration. $p_{i, lb}$ and p_{gb} denotes the personal best of particle i and global best of the swarm discovered so far, respectively.

The DE optimization technique operates on the fundamental principles of mutation and selection to achieve optimization, whose working flow is given in Fig. 3. The operational principle involves the generation of a pool of potential solutions, wherein each subsequent generation is produced by introducing random perturbations to the pre-existing solutions. The perturbations are derived from the disparities among arbitrarily chosen solutions within the present population.

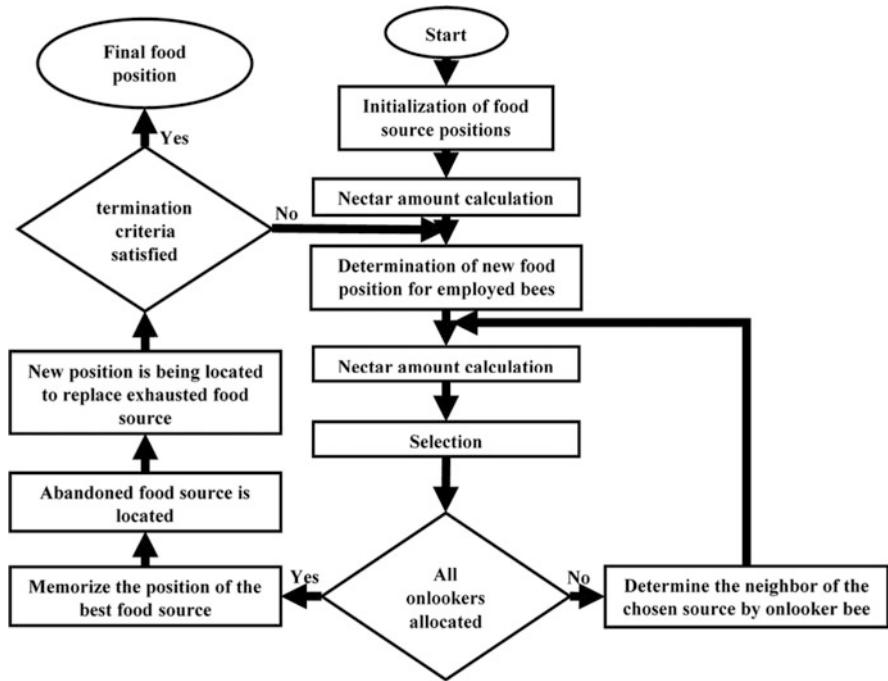


Fig. 4 Flowchart of ABC

The ABC approach is an optimization algorithm that utilizes swarm intelligence and is inspired by the foraging behavior of honey bees. The key components of ABC algorithm are initialization, employed bees, onlooker bees, scout bees, termination criteria, convergence and exploration, and parameter tuning. The detailed working process of ABC is shown in Fig. 4. Its simplicity, effectiveness, and applicability to various optimization problems make it a valuable tool in the field of computational intelligence. This method is a potent optimization approach that exhibits remarkable efficacy in addressing optimization problems that are both nonlinear and nonconvex in nature.

The ACO approach is an optimization technique that is inspired by the behavior of ant colonies. The mechanism involves the generation of a populace of synthetic ants that traverse a designated exploration area, while depositing pheromone traces that serve as a guide for subsequent ants. The reinforcement of pheromone trails occurs as superior solutions are discovered, resulting in the population's convergence towards the optimal solution. Figure 5 explains the flow of the algorithm in a detailed manner. ACO has demonstrated notable efficacy in addressing combinatorial optimization problems, including the well-known Travelling Salesman Problem [12].

ES are a category of optimization methodologies that rely on the fundamental concept of mutation and selection. The operational methodology involves generating

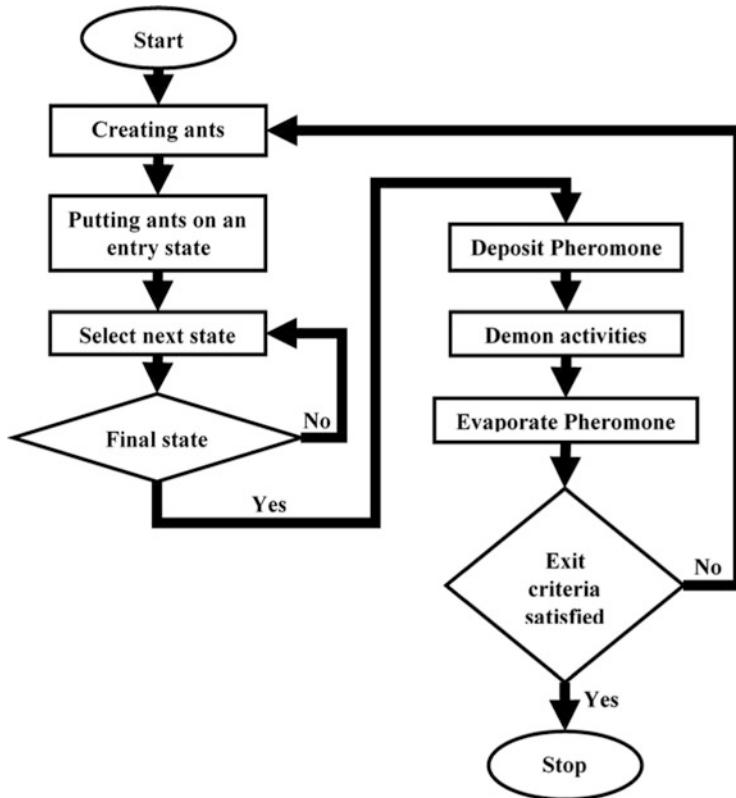


Fig. 5 Flowchart of ACO

a collection of feasible solutions and subsequently subjecting them to mutation operators to yield additional candidate solutions. The mutation operators are designed to explore the search space effectively, while the selection operator ensures that the best solutions are retained. ES is a powerful optimization technique that is particularly effective for solving large-scale and complex optimization problems.

GA and ES differ in strategy and execution. GAs employ strings of genes for solutions, while ES use vectors of real-valued parameters. GAs produce novel solutions using crossover and mutation operators, while ES mostly adapt parameter values using mutation operators. GAs are good for discrete or combinatorial optimization, while ES are good for continuous optimization. GAs use roulette wheel or tournament selection, while ES use survivor selection to keep the best performers. GAs often traverse a large search space early on, sacrificing convergence speed. They may work in harsh, multimodal environment, while ES focus on local search and distribution adaption. In complicated, multimodal environments, they may converge slower than in smooth, convex optimization problems.

2.2 Machine Learning

ML is a scientific discipline that enables computers to acquire knowledge and improve performance through self-learning, without the need for explicit programming. Some studies in ML have been done using the game of checkers. Sufficient research has been conducted to substantiate the notion that a computer can be programmed to acquire the ability to play checkers at a superior level compared to the individual who authored the program [13].

Supervised learning (SL), unsupervised learning (USL), and reinforcement learning (RL) are the three primary forms of ML based on the type of learning problem and the type of input data that is available.

SL is a prominent ML approach that has witnessed significant progress and innovation, and is widely employed in various practical domains. SL refers to algorithms that learn x (*input*) \rightarrow y (*output label*) mappings. SL is distinguished by the provision of labelled examples to the learning algorithm, consisting of input-output pairs x and y , where y represents the correct label for a given input x . Through exposure to these correct input-output pairs, the learning algorithm gradually acquires the ability to make accurate predictions based solely on the input x , without requiring the corresponding output label y .

SL is mainly classified into two main parts: regression and classification, which have been shown in Fig. 6, and also the flow of SL working is shown in Fig. 7. In the case of regression, the learning algorithm predicts a number from infinitely many possible outputs. However, in the context of classification, the goal of learning algorithm is to categorize or classify input data into one or more predefined classes or categories [14].

Supervised ML incorporates a variety of learning algorithms, including: linear regression [15], logistic regression [16], decision trees (DT) [17], random forest (RF) [18], linear discriminant analysis (LDA) [19], support vector machines (SVM) [20], k-nearest neighbors (KNN) [21], Naïve Bayes [22], and neural networks [23].

On the other hand, USL is a ML paradigm that involves the utilization of algorithms to discern patterns and relationships from unstructured or unlabeled data, without any a priori knowledge of the output or target variable. In USL, data

Fig. 6 Classification of SL

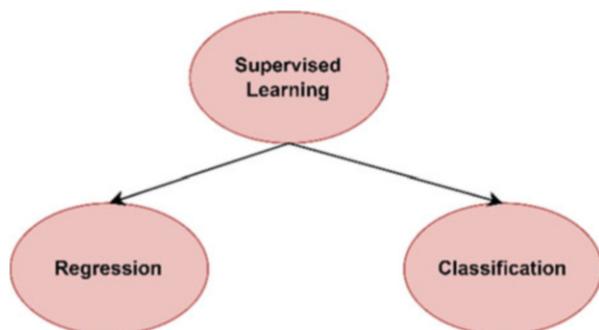
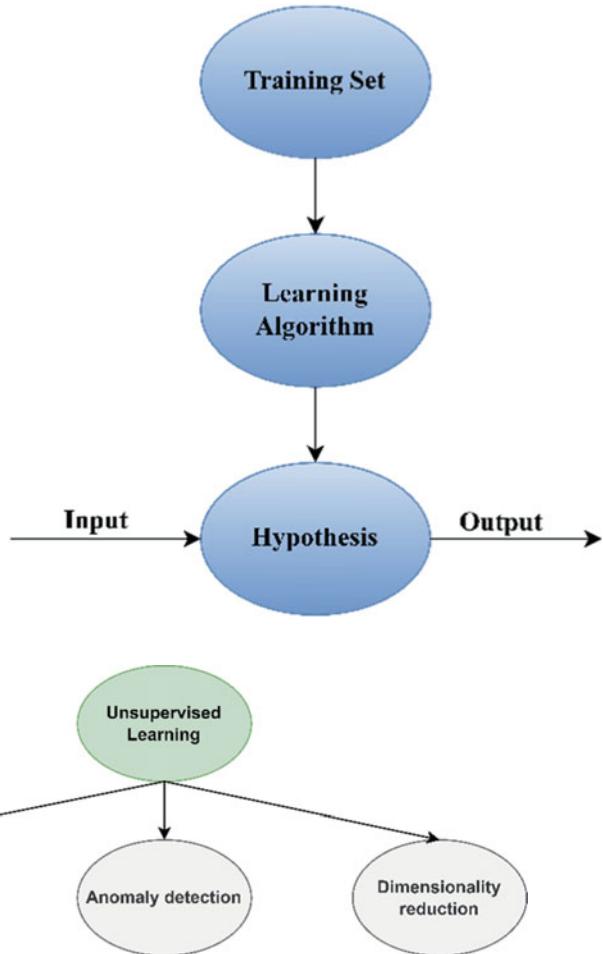


Fig. 7 Working flow of SL**Fig. 8** Popular techniques of USL

only comes with input x but not output label y , so the algorithm has to find structure in the data.

There are several techniques that fall under the umbrella of USL, including clustering, dimensionality reduction, and anomaly detection which is shown in Fig. 8. Clustering involves grouping similar data points together based on certain features or characteristics. This can be useful in identifying patterns in data and creating customer segments for targeted marketing campaigns [24–26]. Furthermore, Dimensionality reduction, involves reducing the number of variables in a dataset while preserving the key features and patterns. This can be useful in simplifying complex datasets and making them easier to work with [27, 28]. Anomaly detection entails the identification of data points that exhibit a substantial deviation from the remaining data. This can be useful in detecting fraud [29], identifying outliers, or

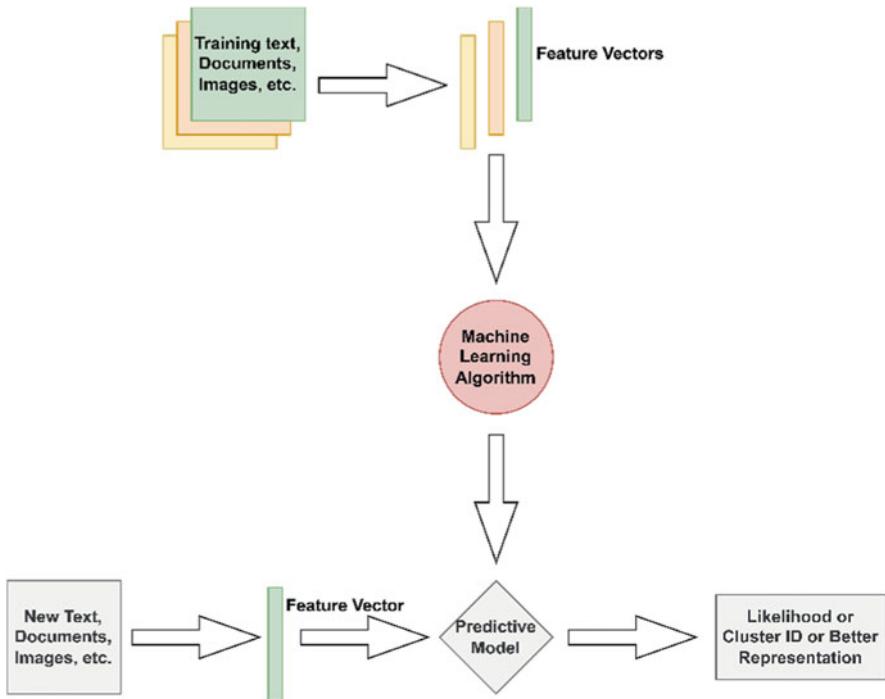


Fig. 9 Flow of USL algorithm

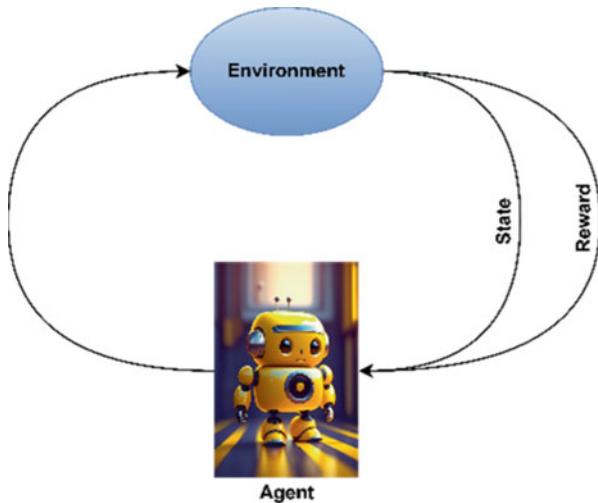
uncovering new insights in the data. Moreover, the flow of USL is also depicted in Fig. 9.

On the other hand, RL is concerned with training agents to arrive at decisions by relying on experiences and feedback obtained within a given environment. Unlike SL and USL, RL does not rely on labeled data, but rather, it learns through trial and error, using a reward signal to guide its actions towards a desired outcome. Figure 10 illustrates the process by which the agent engages with the environment to acquire optimal behavior within that particular context. The picture representing agent is collected from google creative commons licenses image search ([source](#)).

At every discrete point in time, the agent perceives the present state of the environment, selects an action to execute according to its current policy, and obtains a reward signal contingent upon the action it executed and the new state it transitioned into.

The RL algorithm typically involves an iterative process of exploration and exploitation. The objective of RL is to acquire an optimal policy that maximizes the total accumulated reward throughout the learning process. Typically, this is expressed as a Markov Decision Process (MDP), which is a mathematical framework used to explain decision-making in situations with uncertainty [30].

RL encompasses various methodologies, such as value-based, policy-based, and actor-critic techniques. Value-based techniques aim to acquire the optimal value

Fig. 10 RL process

function for each state, which quantifies the anticipated total reward starting from that state. Policy-based approaches acquire the best policy by direct learning, without explicitly calculating the value function. Actor-critic methods integrate value-based and policy-based approaches by employing distinct actor and critic networks to learn the policy and value function, respectively. Furthermore, Path Consistency Learning is a new RL method created by the authors of [31] that achieves substantial improvements over strong actor-critic and Q-learning baselines on a variety of benchmarks.

To analyze the interest of researchers in the present day in ML techniques, an advanced search has been done in Google Scholar using the search strings machine learning, supervised learning, unsupervised learning, and reinforcement learning, respectively, all in the title of the article, and the result analysis has been shown in Fig. 11.

Moreover, evolutionary optimization algorithms provide a more efficient way than the trial-and-error approach to search the space of possible policies by mimicking the process of natural selection to accomplish the goal of RL. Reinforcement Learning (RL) can employ Genetic Algorithms (GA) to efficiently optimize the parameters of a policy network or to evolve the architecture of the network itself. For example, in a policy gradient algorithm, the GA could be used to optimize the weights of the policy network, while in a genetic programming approach, the GA could be used to evolve the structure of the network by adding or removing nodes and connections.

Semi-supervised learning [32] and ensemble learning [33] are other important ML techniques that are used to improve the quality of inference.

Semi-supervised learning blends aspects of both SL and USL. Combination is done by training a model on a dataset that contains both labeled and unlabeled data.

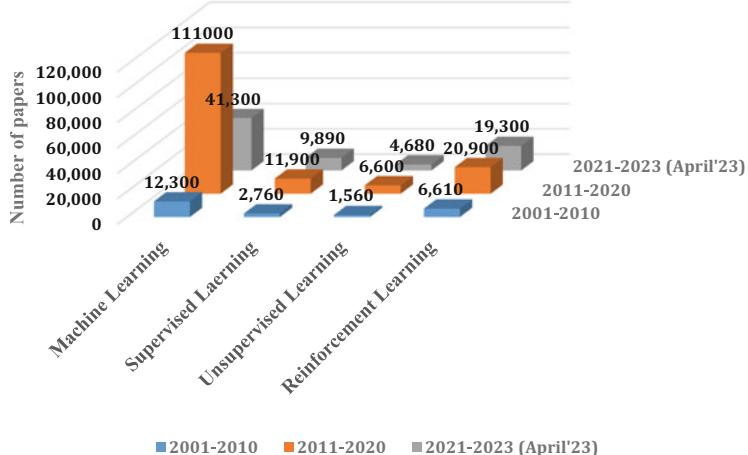


Fig. 11 Google Scholar frequency analysis of ML and its variants

This is beneficial in circumstances where acquiring labeled data is time-consuming, expensive, or simply not feasible. By leveraging the available labeled data and the substantial quantities of unlabeled data, the model can learn more about the underlying structure of the data and generalize better to new data.

Ensemble learning involves amalgamating several individual models into a unified and more robust model. The concept behind ensemble learning is to leverage the strengths of different models and reduce the limitations and biases of individual models.

3 The Synergy of Evolutionary Optimization Techniques and Machine Learning

The amalgamation of AI and EOT has surfaced as a potent amalgam in resolving intricate issues across diverse fields. The integration of evolutionary optimization methods with AI has the potential to augment the process of learning and optimization. The successful application of AI (especially ML) and EOT in a diverse array of problems is depicted in the Figs. 12 and 13, and concisely outlined in Table 1.

The author's work in [34] is the introduction of a novel approach called Multi-node Evolutionary Neural Networks for Deep Learning to automate the process of network selection on computational clusters. This is achieved through hyper-parameter optimization using GA. The article's authors note that while deep learning models acquire their parameters through data-driven methods, model selection through hyper-parameter selection remains a laborious and highly intuitive task. MENNDL is designed to resolve this issue by distributing the task of determining the optimal hyper-parameters across the nodes of a supercomputer using a large

Fig. 12 Set of AI and EOT with their intersection part I

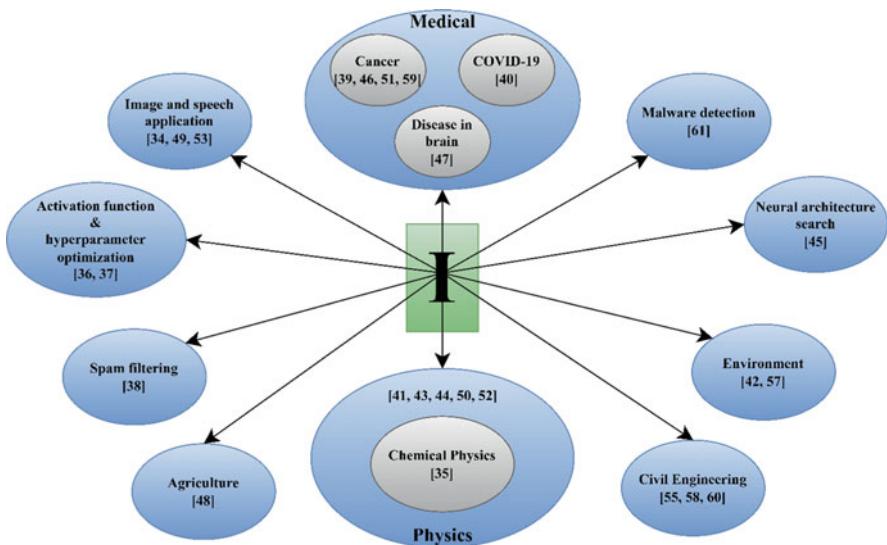
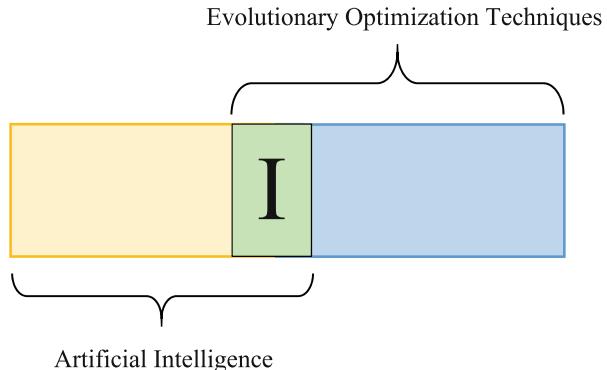


Fig. 13 Recent study on AI-EOT

number of compute nodes communicating via MPI. The software implements DL networks using the Convolutional Architecture for Fast Feature Embedding (Caffe) utility. Overall, it is anticipated that the proposed method will considerably reduce the time and effort required for deep learning hyper-parameter optimization.

The authors in [35] introduces an innovative approach by suggesting a novel algorithm for designing artificial neural network (ANN) architectures that eliminates the heuristic-based bias towards single-layer perceptron NN and instead provides a systematic approach to creating multi-layer networks. The research paper employs a multi-objective optimization methodology to concurrently build the structure of an economical neural network by taking into account many layers and determining the necessary size of sample points for training the ANN model in order to achieve the highest possible accuracy in prediction. The paper also demonstrates the

Table 1 Summary of current studies on the alliance of EOT and AI

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
“Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung-Hwan Lim, Robert M. Patton” [34]	2015	Deep learning (DL), CNN	GA	–	The proposed method for addressing the challenge of selecting hyper-parameters in DL is “Multi-node Evolutionary Neural Networks for Deep Learning (MENNDL)”. This method involves the use of GA to automate network selection on computational clusters by optimizing hyper-parameters.
“Srinivas Souniitri Miriyala, Prateek Mittal, Saptarshi Majumdar, Kishalay Mitra” [35]	2016	Artificial Neural Network (ANN)	Non-Dominated Sorting Genetic Algorithm-II (NSGA-II)	Sobol indices, Latin hypercube sampling, Mann-Whitney U test, Root Mean Square Error (RMSE)	The study describes a generic method for solving complex and time-consuming optimization problems more quickly and effectively.
“Nongnuch Aririth, Alexander Urban, Gerbrand Ceder” [36]	2018	ANN	GA	–	Using the amorphous LiSi alloy as an example, the authors demonstrated how specialized ML potentials can be used to accelerate the sampling of complex structure spaces using first principles. Proposed methodology is founded on the combination of a GA and a potential ANN.

(continued)

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
“Jin-Young Kim, Sung-Bae Cho” [37]	2019	DL, NN	Genetic programming	–	In this study, a novel approach utilizing genetic programming was suggested for the joint optimization of activation functions and optimization techniques.
“Garrett Bingham, William Macke, Risto Miikkulainen” [38]	2020	NN, DL	Evolutionary Algorithms (crossover, mutation, random search and exhaustive search)	Search strategies (Exhaustive Search, Random Search, Evolution)	<ul style="list-style-type: none"> • Several techniques were proposed and assessed to uncover innovative and effective activation functions for deep learning, aiming to improve their performance. • The utilization of loss-based fitness in the context of evolution leads to the discovery of activation functions that exhibit superior accuracy compared to conventional functions like Rectified Linear Unit (ReLU) and emerging functions like Swish. This outcome underscores the efficacy of search algorithms in navigating expansive solution spaces.

“Bilge Kagan Dedeturk, Bahriye Akay” [39]	2020	Logistic regression	ABC	—	A novel spam filtering approach has been proposed.
“Ahmed Elnakib, Hanan M. Amer, Fatma E.Z. Abou-Chadi” [40]	2020	DL architectures (Alex, VGG16, VGG19), SVM, KNN, DT, CNN	GA	Accuracy of detection, specificity, sensitivity	In order to aid in the early detection of lung nodules from low dose computed tomography (LDCT) images, this study proposes a novel strategy incorporating the use of a computer aided detection system.
“Priyanka D. Pantula, Srinivas S. Miriyala, Kishalay Mitra” [41]	2020	ANN, Clustering	GA	—	Developed a novel soft clustering algorithm Neuro-Fuzzy C-Means Clustering algorithm (NFCM).
“Seyed Mohammad Jafar Jalali, Milad Ahmadian, Sajad Ahmadian, Abbas Khosravi, Mamoun Alazab, Saeid Nahavandi” [42]	2021	CNN, Opposition-based learning (OBL), deep Q network, RL	graining-sharing knowledge (GSK) optimization algorithm	Cauchy mutation operators, accuracy, precision, F-score, sensitivity analysis, AUC	The authors of this study have introduced a new deep ensemble image classification model for the identification of COVID-19 cases through the analysis of chest X-ray images.
“Kostas Bavarios, Anastasios Dounis, Panagiotis Kofinas” [43]	2021	RL (Q-learning, SARSA)	Big Bang—Big Crunch, GA	MDP	In order to address the issue of maximum power point (MPP) tracking for photovoltaics, two universal RL approaches are taken into consideration in this study. Under a variety of environmental

(continued)

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
"Kai Qian, Jie Jiang, Yulong Ding, Shuang-Hua Yang" [44]	2021	DNN (CNN, RNN)	GA	Average topological distance (ATD), Average accumulated error (AAE)	For improving the search space of evolutionary algorithms (EAs) and increase the efficiency of searching, the authors presented the deep learning guided evolutionary algorithm (DLGEA) for contamination source identification (CSI) problems.
"Ibrahim M. Mehedi, Hussain Bassi, Muhyaddin J. Rawa, Mohammed Ajour, Abdullah Abusorrah, Mahendiran T. Vellingiri, Zainal Salam, Md. Pauzi Bin Abdullah" [45]	2021	Elman neural networks (ENN), oppositional based learning (OBL)	Oppositional Artificial Fish Swarm Optimization algorithm (OAFSA), Water wave optimization (WWO)	Wavelet Transformation, Mean Absolute Percentage Error (MAPE)	In order to anticipate the load for power systems, this article has created a novel intelligent ML with EA based Short Term load forecasting (IMLEA-STLF) model.

“Takahiro Sato, Masafumi Fujita” [46]	2021	RL	Covariance matrix adaptation evolution strategy	Gaussian process regression (GPR)	Combining RL and evolutionary optimization, this article presents an automatic design strategy for solving electric machine design problems automatically.
“Tong Zhang, Chunyu Lei, Zongyan Zhang, Xian-Bing Meng, C. L. Philip Chen” [47]	2021	Neural architecture search (NAS), RL, Neural architecture encoding strategy (NAES)	Adam (Adaptive moment estimation)	$L_{1/2}$ regularization	The non-convex nature of NAS poses a significant challenge in the realm of deep learning design. The present study proposes an adaptive scalable neural architecture search approach, namely the AS-NAS, which leverages the reinforced I-Ching divination EA (IDEA) and variable architecture encoding strategy to tackle the aforementioned issue.
“Yogendra Singh Solanki, Prasun Chakrabarti, Michal Jasinski, Zbigniew Leonowicz, Vadim Bolshev, Alexander Vinogradov, Elzbieta Jasinska, Radomir Gono, Mohammad Nami” [48]	2021	SL, Wrapper-based feature selection, SVM, J48 (C4.5 DT Algorithm), Multilayer Perceptron (MLP) (a feed-forward ANN), Naive Bayes, KNN, RF	PSO, Genetic search	RMSE, Matthew's Correlation Coefficient (MCC), Kappa Statistics, Mean absolute error (MAE), Relative absolute error (RAE), Root relative squared error, Greedy Stepwise	A novel system has been developed that integrates supervised ML techniques for the purpose of breast cancer prognosis. The present system utilizes techniques for selecting features and approaches for addressing imbalanced data.

(continued)

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
“Xinchun Cui, Ruyi Xiao, Xiaoli Liu, Hong Qiao, Xiangwei Zheng, Yiquan Zhang, Jianzong Du” [49]	2021	Adaptive LASSO logistic regression	PSO	–	A novel method is proposed for diagnosing Alzheimer’s disease (AD).
“Mohsen Yoosefzadeh-Najafabadi, Dan Tulpan, Milad Eskandari” [50]	2021	PCA, Radial Basis Function (RBF), MLP, RF, Bagging strategy (Ensemble method)	GA	Coefficient of Determination (R^2), MAE, RMSE, Best Linear Unbiased Prediction (BLUP), Nearest neighbor analysis (NNA), Pearson coefficient of correlations	The proposal entails the utilization of ML and genetic optimization algorithms to develop a model aimed at enhancing soybean yield through the optimization of its constituent traits. (data)
“Xiaohao Sun, Liang Yue, Luxia Yu, Han Shao, Xirui Peng, Kun Zhou, Frédéric Demoly, Ruike Zhao, H. Jerry Qi” [51]	2022	Recurrent neural network (RNN)	Evolutionary Algorithm (EA)	–	<ul style="list-style-type: none"> • The present study introduces a novel approach that utilizes ML and EA to facilitate the design process. • The utilization of ML in EA enables the resolution of the inverse problem, which involves the identification of the optimal design. • A novel approach is proposed that integrates ML and EA with computer vision algorithms, enabling a more efficient

“Tianxiang Wang, Youbo Liu, Gao Qiu, Lijie Ding, Wei Wei, Junyong Liu” [52]	2022	Back propagation neural network (BPNN)	Non-dominated sorting GA (NSGA-II)	–	The issue of voltage stability exhibits pronounced nonlinearity and is highly susceptible to dimensionality. Traditional methods encounter difficulties in achieving a balance between precision and efficiency in the control of voltage stability margins. In this paper, a simplified control model that is nonlinear and efficient, with a focus on voltage stability, is developed.	The present investigation employed DT, ML classifier and PSO feature selection algorithm to detect breast cancer from the UCI repository.
“Jesutofunmi Onaope Afolayan, Marion Olubummi Adebiyi, Micheal Olaoju Arowolo, Chinmay Chakraborty, Ayodele Ariyo Adebiyi” [53]	2022	DT	PSO	Accuracy of detection, sensitivity, specificity, F-score	(continued)	

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
“Yan Cao, Elham Kamrani, Saeid Mirzaei, Amith Khaundakar, Behzad Vafei” [54]	2022	ANN, Adaptive neuro-fuzzy inference systems (ANFIS), least-squares support vector regression, Cascade feedforward, Radial basis function (RBF), MLP, Generalized regression	DE algorithm	Absolute average relative deviation (AARD), MSE, R^2	The present study utilizes ML methodologies to model the electrical output of a Photovoltaic (PV)/ thermal (T) system that is cooled by Nano fluids with a water base.
“Mayuri Arul Vinayakan Rajasimman, Ranjith Kumar Manoharan, Neelakandan Subramani, Manimaran Aridoss, Mohammad Gouse Galety” [55]	2022	Deep CNN (densely connected network (DenseNet-169))	Chimp optimization algorithm (COA)	long short-term memory (LSTM), Teaching and learning-based optimization (TLBO)	The present investigation presents an innovative model, namely the Robust Facial Expression Recognition using an Evolutionary Algorithm with Deep Learning (RFER-EADL), which aims to enhance facial expression recognition.
“Zhenzhen Hu, Wenyin Gong” [56]	2022	RL (Q-learning)	DE	Friedman’s test	The authors have introduced a novel approach for addressing constrained optimization problems, which involves the integration of DE and RL techniques. This approach has been named RL-CORCO and has demonstrated promising results

				in improving the effectiveness of the optimization process.
“Hussaini Jaafaru, Bismark Agbelle” [57]	2022	Ensemble learning (Bagging, Boosting), RF, SVM, ANN, Xgboost, Stacked Model, MLP	GA, Non-dominated sorting genetic II (NSGA-II), Distributed EA in Python (DEAP)	Multi Criteria Decision Analysis (MCDA), Exponential Utility Function (EUF)
“Hassen Louati, Slim Bechikh, Ali Louati, Abdulaziz Aldaej, Lamjed Ben Said” [58]	2022	CNN, pruning, quantization, and compression approaches	GA	The proposed Compression-CNN-Xray model integrates pruning, GA, quantization, and diverse compression techniques to attain a satisfactory balance between convolutions layer pruning and the detection of potential thoracic anomalies and infections, including COVID-19 cases.
“Majid Emami Javanmard, S.F. Ghaderi” [59]	2022	Autoregressive Model (AR), ANN, Autoregressive Integrated Moving Average Model (ARIMA), Seasonal Autoregressive Integrated Moving Average Model with Exogenous factors (SARIMAX), RF, Seasonal Autoregressive Integrated	PSO, Grey Wolf Optimization (GWO)	A hybrid model has been developed through the integration of ML algorithms and an optimization model, with the aim of predicting greenhouse gas emissions based on energy market data. This approach has been the subject of research.

(continued)

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
"Hoang-Le Minh, Thanh Sang-To, Magd Abdel Wahab, Thanh Cuong-Le" [60]	2022	Moving Average Model (SARIMA), Support Vector Regression (SVR), Long Short-Term Memory Model (LSTM), KNN	PSO, GWO, Whale Optimization algorithm (WOA), Slime Mould Algorithm (SMA), Arithmetic Optimization Algorithm (AOA), Reptile Search Algorithm (RSA), Aquila Optimizer (AO), Evolution Strategy (ES), Gravitational Search Algorithm (GSA), GA, Large-Scale Hybrid Adaptive Differential Evolution (LSHADE)	–	The authors have developed a novel metaheuristic optimization approach by utilizing the K-means clustering algorithm to detect structural damage. (source code)
"Mingjing Wang, Xiaoping Li, Long Chen, Huijing Chen" [61]	2023	–	Harris hawks optimization (HHO)	Penalty-based boundary intersection (PBI)	<ul style="list-style-type: none"> • The present study introduces a multi-objective evolutionary algorithm that incorporates decomposition and Harris hawks learning (MOEA/D-HHL) to facilitate medical ML.

			<ul style="list-style-type: none"> The MOEA/D-HHL algorithm has been effectively employed to optimize a ML algorithm for 35 gene expression datasets related to cancer and two clinical medical datasets with practical applications. The MOEA/D-HHL exhibits potential as a viable tool for employment in medical ML. 	
“Mousaab Zakhrouf, Bouchelka Hamid, Sungwon Kim, Stamboul Madani” [62]	2023	DL, feedforward neural network (FFNN)	PSO	RMSE, signal-to-noise ratio (SNR), Nash– Sutcliffe efficiency (NSE)
“Santosh Jhansi Kattamuri, Ravi Kiran Varma Pennmatha, Sujata Chakravarty, Venkata Sai Pavan Madabathula” [63]	2023	Gaussian Naive Bayes (GNB), SVM, DT, Wrapper-based feature selection, Nearest Centroid (NC), RF, KNN	Cuckoo Search Optimiza- tion (CSO), ACO, GWO	– The author generated a Swarm Optimization and ML Applied to Portable Executable Malware Detection dataset to explore the potential of enhancing cyber threat intelligence through improved malware detec- tion accuracy.

(continued)

Table 1 (continued)

Authors	Year	AI technique(s)	Nature-inspired/ evolutionary optimization algorithm(s)	Statistical/probabilistic tools, others	Key contribution
“Arijun Manoj, Srinivas Soumitri Miriyala, Kishalay Mitra” [64]	2023	–	Bayesian Optimization, NSGA-II	Bayesian technique	Formulated a novel Multi-Objective Optimization Problem based on first-principles-based HFM for PVAc polymerization that seeks to simultaneously maximize the quality and strength of the polymer while minimizing its production time.
“Mohammed Al-Aghbari, Ashish M. Gujarathi” [65]	2023	–	Bi-objective genetic programming (BioGP) algorithm, NSGA-II	Net flow method (NFM)	A hybrid optimization technique employing BioGP and NSGA-II algorithm is suggested to increase Pareto solution diversity and convergence speed without ignoring physical reservoir modeling. The Brugge field benchmark model and a Middle Eastern oil-field sector model apply the unique methodology.
“Saurabh Vashistha, Bashista Kumar Mahanta, Vivek Kumar Singh, Shailesh Kumar Singh” [66]	2023	ANN	Evolutionary Neural Network (EvoNN), Reference Vector Evolutionary Algorithm (RVEA)	RMSE	Al-Co-Cr-Fe-Ni HEA alloy is synthesized using vacuum induction melting, and its tribological qualities are studied in dry conditions.

effectiveness of using ANN models as surrogate models to replace original time-expensive physics-based reaction network models, which can significantly reduce the function calls required for optimization.

The paper [36] contributes to the development of a ML-assisted sampling methodology that can be used to build first-principles phase diagrams of amorphous materials. In particular, the authors integrates GA and an artificial neural network (ANN) to efficiently determine low-energy amorphous structures. The researchers showcase the effectiveness of this approach by utilizing the LiSi alloy in its amorphous state. They prove that the metastable structures produced consistently fall within a low energy range. The authors suggest that this methodology can be applied to represent amorphous and disordered materials in a broad sense, without being restricted to any particular material or amorphization mechanism. This work introduces an innovative method for modeling amorphous materials at the atomic level, which has the potential to greatly influence the field of materials research.

The article [37] proposes a novel genetic programming-based method for simultaneously determining the optimal activation functions and optimization techniques for deep learning models. This method may improve the performance of deep learning models and surmount the limitations of previous research that relied on heuristic approaches. This paper is an important contribution to the fields of AI and deep learning, as it presents a new approach for optimizing and enhancing the performance of deep learning models.

Contribution of the article [38] is the creation of an evolutionary method for optimizing activation functions in deep neural networks. The authors depict activation functions as trees and discover novel activation functions in expressive search spaces containing billions of candidate activation functions through crossover, mutation, and exhaustive search. On image classification tasks CIFAR-10 and CIFAR-100, the resultant activation functions outperform conventional activation functions, such as ReLU, and are therefore unlikely to be discovered manually. This approach provides a novel method for optimizing neural networks and can result in significant accuracy gains for neural networks. In addition, the authors demonstrate the adaptability of this method by demonstrating that the evolved activation functions can be tailored to particular objectives and architectures.

The researchers in [39] developed a new method for detecting spam that integrates the ABC algorithm with a classification model based on logistic regression. The empirical findings from the analysis of three publicly accessible datasets (Enron, CSDMC2010, and TurkishEmail) indicate that the suggested model is proficient in handling data with a large number of dimensions, owing to its very efficient local and global search capabilities. They compared the spam detection performance of the proposed model to that of SVM, logistic regression, and naive Bayes classifiers. Regarding classification precision, the authors observe that the suggested method surpasses existing spam detection techniques examined in this study.

The current work [40] introduces a new approach using a computer-aided detection (CADe) system to help identify lung nodules at an early stage from low-dose computed tomography (LDCT) pictures. Initially, the recommended system enhances the contrast of low dose images by preprocessing the raw data.

Investigating various deep learning architectures, such as Alex, VGG16, and VGG19 networks, enables the extraction of compact deep learning features. A GA is trained to choose the most pertinent features for early detection in order to optimize the extracted set of features.

The authors in [41] discusses one of the notable methods for conducting clustering, viz., Fuzzy C-means (FCM), which involves assigning each data point a fuzzy membership function value that represents its potential association with all clusters. FCM is executed through the process of minimizing an objective functional by effectively estimating the decision variables, which include the membership function values and cluster representatives. This estimation is carried out within a constrained environment. The utilization of this methodology results in a slight augmentation in the quantity of data points, which subsequently yields a substantial expansion in the magnitude of choice factors. The occurrence of this explosion hinders the utilization of evolutionary optimization solvers in FCM, thus resulting in suboptimal data clustering. This research introduces the NFCM as a solution to the aforementioned challenges, utilizing an innovative clustering approach based on ANN technology. The NFCM methodology involves the construction of a functional map that establishes a relationship between the data points and membership function values. This mapping facilitates a notable decrease in the quantity of decision variables.

The diagnosis of Alzheimer's disease (AD) and its early stage, known as moderate cognitive impairment (MCI), in computer-assisted systems depends on precise classification. There are always features that are unrelated to one another and features that are redundant, therefore not all of the AD data features will categorize properly. A particle swarm optimization-based adaptive LASSO logistic regression model (PSO-ALLR) is developed in [49] to handle this challenge. Two phases make up this algorithm. The PSO technique is utilized for global search in the first stage to remove redundant characteristics and save computational time. Local search using adaptive LASSO selects the most relevant AD classification characteristics in the second stage.

The primary contribution of the work [53] is the application of PSO to pick relevant characteristics in breast cancer data. This approach facilitates the evaluation of classification performance using the DT-ML algorithm. The paper also proposes feature selection by employing the PSO algorithm on the unprocessed breast cancer dataset to obtain a reduced, noise-free subset, which will be analyzed and evaluated using the Decision Tree machine classification algorithm.

The study [54] investigates different AI-based models to determine the most precise model for forecasting the electrical efficiency of a nanofluid-cooled PV/T system. Empirical experimentation and statistical analysis validate that the ANFIS, utilizing the subtractive clustering membership function and trained by the hybrid algorithm, yields optimal accurate predictions for the specified task. Ultimately, the introduction of DE algorithm determines the optimal PV/T electrical efficiency state.

The fundamental subject of article [64] is the utilization of Multi-objective Bayesian Optimization (MOBO) in the industrial manufacturing of Polyvinyl Acetate (PVAc). The paper proposes q-Expected Hyper-Volume Improvement (qEHVI), a new function of acquisition, to attain Pareto optimality while optimizing the

operating conditions for large-scale industrial production of PVAc. The paper also illustrates the difficulties of computationally expensive multi-objective optimization in manufacturing industries and proposes a solution based on MOBO applied to the PVAc polymerization process. The proposed method is anticipated to reduce the trade-off between cost and quality.

The tribological properties of an Al–Co–Cr–Fe–Ni high-entropy alloy (HEA) were investigated by the authors in reference [66]. The alloy was synthesized via vacuum induction melting, and its performance was examined under dry conditions. Evolutionary data-driven modeling and optimization strategies are employed to ascertain the most advantageous solutions for diverse loading conditions and frequency levels, with the goal of enhancing tribological performance. The data-driven modeling approach incorporates the use of an algorithm called EvoNN, which is based on neural networks. This algorithm is employed to develop and train models, which are subsequently utilized in the optimization process. Constraint version reference vector algorithm known as cRVEA, which utilizes a reference vector approach, is combined with EvoNN in order to assess the optimal values for wear, coefficient of friction (COF), and surface roughness in a multidimensional hyperspace.

Using the effect coefficients method, the current study [67] conducted strength analyses on the junction between the cylindrical portion and several varieties of well-known end domes. The focus of the investigation was on filament wound composite pressure vessels, which were constructed using glass-epoxy and carbon-epoxy composites. Additionally, the study also addresses the design of an appropriate end dome form, considering the junction area. The determination of the best final dome form was conducted using data-driven evolutionary algorithms. Surrogate models were developed using the Evolutionary Deep Neural Nets (EvoDN2) algorithm, utilizing an input dataset obtained from an analytical simulation. cRVEA optimization technique was utilized to conduct the multicriterial optimization procedure.

4 Conclusion

In the recent era, the synergy of ML and EOT has been proven as a potential tool due to its wide applicability in AI system. This chapter basically reviews the trends and advancements in EOT from ML perspective. It is seen that the combined effect of some/all of ML, DL, and EOT has enabled advancements in image and speech processing/recognition, chemical physics, spam filtering, lung cancer detection, breast cancer diagnosis and prognosis, Alzheimer's disease detection, COVID-19 diagnosis, electric machines, determining water contamination sources, greenhouse gas emissions, agricultural food production, bridge maintenance strategy, reducing structural damage, analyzing stream flow, and cyber threat intelligence. Moreover, to enhance the model's performance, the synergy also optimized the activation

functions and hyperparameters. However, after a thorough review of the recent notable papers, the chapter can land to the following conclusions.

- (i) Reinforcement Learning has garnered much attention of the researchers as it has been popularly applied in many areas of science and engineering. However, there is a scope too to improve the RL strategies for effective output.
- (ii) The popular hybridization of RL-EOT has been effectively used to solve the constrained optimization problems like electric machine problems, MPP tracking for photovoltaics and image classification model for the identification of COVID-19 cases through chest X-ray images.
- (iii) Recently, the blend of SL and EOT has also been vastly implemented yet in the field of medical science; more in cancer detection and less in Alzheimer's disease.

As a future Scope of research, though the review of the papers ensures the interesting applications of ML and EOT, still there are some potential fields where such combined technique can magically improve the solution quality. Apart from Cancer detection problems only, some more areas in medical sciences can be explored for the suitable applications of the discussed technique. In the other hand, in some cases, the problems can be viewed in multi-objective scenario where there are conflict of interests and a set of possible solutions (called pareto-optimal solutions) can rather be captured for an effective outcome of the framed real-world problem.

References

1. Louati H, Bechikh S, Louati A, Aldaej A, Said LB. Evolutionary optimization of convolutional neural network architecture design for thoracic x-ray image classification. In: Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part I 34. Springer International Publishing; 2021. p. 121–32.
2. Bungum L, Gamback B. Evolutionary algorithms in natural language processing. In: Norwegian Artificial Intelligence Symposium (Vol. 22). November 2010.
3. Justesen N, Bontrager P, Togelius J, Risi S. Deep learning for video game playing. IEEE Trans Games. 2019;12(1):1–20.
4. Alexandropoulos SAN, Aridas CK, Kotsiantis SB, Vrahatis MN. Multi-objective evolutionary optimization algorithms for machine learning: a recent survey. In: Demetriou I, Pardalos P, editors. Approximation and optimization: algorithms, complexity and applications. Cham: Springer; 2019. p. 35–55.
5. Telikani A, Tahmassebi A, Banzhaf W, Gandomi AH. Evolutionary machine learning: a survey. ACM Comput Surv. 2021;54(8):1–35.
6. Holland JH. Genetic algorithms. Sci Am. 1992;267(1):66–73.
7. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 4). IEEE; November 1995. p. 1942–48.
8. Das S, Suganthan PN. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput. 2010;15(1):4–31.

9. Karaboga D, Gorkemli B, Ozturk C, Karaboga N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artif Intell Rev.* 2014;42:21–57.
10. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. *IEEE Comput Intell Mag.* 2006;1(4):28–39.
11. Beyer HG, Schwefel HP. Evolution strategies—a comprehensive introduction. *Nat Comput.* 2002;1:3–52.
12. Hlaing ZCSS, Khine MA. Solving traveling salesman problem by using improved ant colony optimization algorithm. *Int J Inf Educ Technol.* 2011;1(5):404.
13. Samuel AL. Some studies in machine learning using the game of checkers. *IBM J Res Dev.* 1959;3(3):210–29.
14. Ng A. CS229 lecture notes. *CS229 Lecture Notes.* 2000;1(1):1–3.
15. Maulud D, Abdulazeez AM. A review on linear regression comprehensive in machine learning. *J Appl Sci Technol Trends.* 2020;1(4):140–7.
16. Liu L. Research on logistic regression algorithm of breast cancer diagnose data by machine learning. In: 2018 International Conference on Robots & Intelligent System (ICRIS). IEEE; May 2018. p. 157–60.
17. Quinlan JR. Induction of decision trees. *Mach Learn.* 1986;1:81–106.
18. Liu Y, Wang Y, Zhang J. New machine learning algorithm: random forest. In: Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14–16, 2012. Proceedings 3. Berlin: Springer; 2012. p. 246–52.
19. Tharwat A, Gaber T, Ibrahim A, Hassanien AE. Linear discriminant analysis: a detailed tutorial. *AI Commun.* 2017;30(2):169–90.
20. Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B. Support vector machines. *IEEE Intell Syst Their Appl.* 1998;13(4):18–28.
21. Zhang Z. Introduction to machine learning: k-nearest neighbors. *Ann Transl Med.* 2016;4(11):218.
22. Berrar D. Bayes' theorem and naive Bayes classifier. *Encyclopedia Bioinf Comput Biol.* 2018;403:412.
23. Liu Y, Liu S, Wang Y, Lombardi F, Han J. A survey of stochastic computing neural networks for machine learning applications. *IEEE Trans Neural Netw Learn Syst.* 2020;32(7):2809–24.
24. Dolnicar S. Market segmentation for e-tourism. In: Handbook of e-tourism. Cham: Springer International Publishing; 2022. p. 849–63.
25. Tabianan K, Velu S, Ravi V. K-means clustering approach for intelligent customer segmentation using customer purchase behavior data. *Sustainability.* 2022;14(12):7243.
26. Zhang X, Zheng Y, Ye X, Peng Q, Wang W, Li S. Clustering with implicit constraints: a novel approach to housing market segmentation. *Trans GIS.* 2022;26(2):585–608.
27. Kumar V, Kalitin D, Tiwari P. Unsupervised learning dimensionality reduction algorithm PCA for face recognition. In: 2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE; May 2017. p. 32–37.
28. Hurtik P, Molek V, Perfilieva I. Novel dimensionality reduction approach for unsupervised learning on small datasets. *Pattern Recogn.* 2020;103:107291.
29. Carcillo F, Le Borgne YA, Caelen O, Kessaci Y, Oblé F, Bontempi G. Combining unsupervised and supervised learning in credit card fraud detection. *Inf Sci.* 2021;557:317–31.
30. Van Otterlo M, Wiering M. Reinforcement learning and Markov decision processes. In: Wiering M, van Otterlo M, editors. Reinforcement learning: state-of-the-art. Berlin: Springer; 2012. p. 3–42.
31. Nachum O, Norouzi M, Xu K, Schuurmans D. Bridging the gap between value and policy based reinforcement learning. *Adv Neural Inf Process Syst.* 2017; <https://doi.org/10.48550/arXiv.1702.08892>.
32. Van Engelen JE, Hoos HH. A survey on semi-supervised learning. *Mach Learn.* 2020;109(2):373–440.
33. Dong X, Yu Z, Cao W, Shi Y, Ma Q. A survey on ensemble learning. *Front Comp Sci.* 2020;14:241–58.

34. Young SR, Rose DC, Karnowski TP, Lim SH, Patton RM. Optimizing deep learning hyperparameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. November 2015. p. 1–5.
35. Miriyala SS, Mittal P, Majumdar S, Mitra K. Comparative study of surrogate approaches while optimizing computationally expensive reaction networks. *Chem Eng Sci.* 2016;140:44–61.
36. Arirth N, Urban A, Ceder G. Constructing first-principles phase diagrams of amorphous Li × Si using machine-learning-assisted sampling with an evolutionary algorithm. *J Chem Phys.* 2018;148(24):241711.
37. Kim JY, Cho SB. Evolutionary optimization of hyperparameters in deep learning models. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE; June 2019. p. 831–37.
38. Bingham G, Macke W, Miikkulainen R. Evolutionary optimization of deep learning activation functions. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. June 2020. p. 289–96.
39. Dedeturk BK, Akay B. Spam filtering using a logistic regression model trained by an artificial bee colony algorithm. *Appl Soft Comput.* 2020;91:106229.
40. Elnakib A, Amer HM, Abou-Chadi FE. Early lung cancer detection using deep learning optimization. *Int J Online Biomed Eng.* 2020;16(6):82–94.
41. Pantula PD, Miriyala SS, Mitra K. An evolutionary neuro-fuzzy C-means clustering technique. *Eng Appl Artif Intell.* 2020;89:103435.
42. Jalali SMJ, Ahmadian M, Ahmadian S, Khosravi A, Alazab M, Nahavandi S. An oppositional-Cauchy based GSK evolutionary algorithm with a novel deep ensemble reinforcement learning strategy for COVID-19 diagnosis. *Appl Soft Comput.* 2021;111:107675.
43. Bavarinos K, Dounis A, Kofinas P. Maximum power point tracking based on reinforcement learning using evolutionary optimization algorithms. *Energies.* 2021;14(2):335.
44. Qian K, Jiang J, Ding Y, Yang SH. DLGEA: a deep learning guided evolutionary algorithm for water contamination source identification. *Neural Comput & Applic.* 2021;33:1889–903.
45. Mehedi IM, Bassi H, Rawa MJ, Ajour M, Abusorrah A, Vellingiri MT, et al. Intelligent machine learning with evolutionary algorithm based short term load forecasting in power systems. *IEEE Access.* 2021;9:100113–24.
46. Sato T, Fujita M. A data-driven automatic design method for electric machines based on reinforcement learning and evolutionary optimization. *IEEE Access.* 2021;9:71284–94.
47. Zhang T, Lei C, Zhang Z, Meng XB, Chen CP. AS-NAS: adaptive scalable neural architecture search with reinforced evolutionary algorithm for deep learning. *IEEE Trans Evol Comput.* 2021;25(5):830–41.
48. Solanki YS, Chakrabarti P, Jasinski M, Leonowicz Z, Bolshev V, Vinogradov A, et al. A hybrid supervised machine learning classifier system for breast cancer prognosis using feature selection and data imbalance handling approaches. *Electronics.* 2021;10(6):699.
49. Cui X, Xiao R, Liu X, Qiao H, Zheng X, Zhang Y, Du J. Adaptive LASSO logistic regression based on particle swarm optimization for Alzheimer's disease early diagnosis. *Chemom Intell Lab Syst.* 2021;215:104316.
50. Yoosefzadeh-Najafabadi M, Tulpan D, Eskandari M. Application of machine learning and genetic optimization algorithms for modeling and optimizing soybean yield using its component traits. *PLoS One.* 2021;16(4):e0250665.
51. Sun X, Yue L, Yu L, Shao H, Peng X, Zhou K, et al. Machine learning-evolutionary algorithm enabled design for 4D-printed active composite structures. *Adv Funct Mater.* 2022;32(10): 2109805.
52. Wang T, Liu Y, Qiu G, Ding L, Wei W, Liu J. Deep learning-driven evolutionary algorithm for power system voltage stability control. *Energy Rep.* 2022;8:319–24.
53. Afolayan JO, Adebiyi MO, Arowolo MO, Chakraborty C, Adebiyi AA. Breast cancer detection using particle swarm optimization and decision tree machine learning technique. In: Intelligent healthcare: infrastructure, algorithms and management. Singapore: Springer Nature; 2022. p. 61–83.

54. Cao Y, Kamrani E, Mirzaei S, Khandakar A, Vaferi B. Electrical efficiency of the photovoltaic/thermal collectors cooled by nanofluids: machine learning simulation and optimization by evolutionary algorithm. *Energy Rep.* 2022;8:24–36.
55. Arul Vinayakam Rajasimman M, Manoharan RK, Subramani N, Aridoss M, Galety MG. Robust facial expression recognition using an evolutionary algorithm with a deep learning model. *Appl Sci.* 2022;13(1):468.
56. Hu Z, Gong W. Constrained evolutionary optimization based on reinforcement learning using the objective function and constraints. *Knowl-Based Syst.* 2022;237:107731.
57. Jaafar H, Agbelie B. Bridge maintenance planning framework using machine learning, multi-attribute utility theory and evolutionary optimization models. *Autom Constr.* 2022;141:104460.
58. Louati H, Bechikh S, Louati A, AlDaej A, Said LB. Evolutionary optimization for CNN compression using thoracic X-ray image classification. In *Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings*. Cham: Springer International Publishing; August 2022. p. 112–23.
59. Javannard ME, Ghaderi SF. A hybrid model with applying machine learning algorithms and optimization model to forecast greenhouse gas emissions with energy market data. *Sustain Cities Soc.* 2022;82:103886.
60. Minh HL, Sang-To, T, Wahab MA, Cuong-Le T. A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification. *Knowl-Based Syst.* 2022;251:109189.
61. Wang M, Li X, Chen L, Chen H. Medical machine learning based on multiobjective evolutionary algorithm using learning decomposition. *Expert Syst Appl.* 2023;216:119450.
62. Zakhrouf M, Hamid B, Kim S, Madani S. Novel insights for streamflow forecasting based on deep learning models combined the evolutionary optimization algorithm. *Phys Geogr.* 2023;44(1):31–54.
63. Kattamuri SJ, Penmatsa RKV, Chakravarty S, Madabathula VSP. Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence. *Electronics.* 2023;12(2):342.
64. Manoj A, Miriyala SS, Mitra K. Multi-objective optimization through a novel Bayesian approach for industrial manufacturing of Polyvinyl Acetate. *Mater Manuf Process.* 2023;38: 1–9.
65. Al-Aghbari M, Gujarathi AM. Hybrid approach of using bi-objective genetic programming in well control optimization of waterflood management. *Geoenergy Sci Eng.* 2023;228:211967.
66. Vashistha S, Mahanta BK, Singh VK, Singh SK. Machine learning assisted optimization of tribological parameters of Al–Co–Cr–Fe–Ni high-entropy alloy. *Mater Manuf Process.* 2023;38(16):2093–106.
67. Vondráček D, Padovec Z, Mareš T, Chakraborti N. Analysis and optimization of junction between cylindrical part and end dome of filament wound pressure vessels using data driven evolutionary algorithms. *Proc Inst Mech Eng Part C.* 2023; <https://doi.org/10.1177/09544062231191319>.

Metaheuristic and Evolutionary Algorithms in Explainable Artificial Intelligence



Hardik Prabhu, Aamod Sane, Renu Dhadwal, Patrick Siarry,
and Jayaraman Valadi

Abstract There has been an increasing interest in Explainable Artificial Intelligence (XAI) in recent years. Complex machine learning algorithms, such as deep neural networks, can accurately predict outcomes, but provide little insight into how the decision was made or what factors influenced the outcome. This lack of transparency can be a major issue in high-stakes decision-making scenarios, where understanding the reasoning behind a decision is crucial. XAI aims to address the problem of the “black box” in machine learning models, where the AI’s decision-making process is not transparent, and humans cannot understand how the AI arrived at a particular decision or prediction. Evolutionary and metaheuristic techniques offer promising avenues for achieving explainability in AI systems, and there is a lot of ongoing research in this area to further explore their potential. Our work is a concise literature review that explores the potential adoption of these techniques to facilitate the attainment of explainability in AI systems. We have highlighted some of the contributions of evolutionary and metaheuristic techniques in different approaches to achieving explainability, such as counterfactual explanations, local surrogate modelling, and the development of transparent models.

Keywords Explainable AI · Metaheuristic and evolutionary algorithms · Counterfactuals · Local surrogate models · Transparent models

H. Prabhu

Indian Institute of Science, Bengaluru, India
e-mail: hardikprabhu@iisc.ac.in

A. Sane · R. Dhadwal (✉)

FLAME University, Pune, Maharashtra, India
e-mail: aamod.sane@flame.edu.in; renu.dhadwal@flame.edu.in; jayaraman.vk@flame.edu.in

P. Siarry

University Paris-Est Créteil, Créteil, France
e-mail: siarry@u-pec.fr

1 Explainable Artificial Intelligence (XAI)

Artificial intelligence (AI) has witnessed enormous progress and advancement in recent years. With the advent of deep learning and other sophisticated techniques, AI has surpassed human-level performance in many complex tasks. AI systems have found their application in diverse fields, including financial modelling, natural language processing, medical diagnosis, self-driving cars, and various other domains. However, the rise of AI has also raised concerns about the need for explainability [1].

There are ethical and societal implications associated with the use of AI. Incorporating AI in decision-making processes can perpetuate existing biases and discrimination [2]. Explainability can help to identify and address these biases, help to build trust in AI systems, ensure fair and unbiased decision-making, and enable accountability. As AI continues to advance and become more ubiquitous, the need for explainability will only continue to grow.

XAI in ML could be broadly classified into two categories: (i) Transparent Models and (ii) Post-hoc Explainability [3]. Transparent Models are those which are self-explainable, and they do not require any external tools in order to explain their predictions. Building Transparent models is ideal for explainability. However, it is very challenging to interpret many complex models such as deep neural networks which consist of thousands of parameters and are practically black boxes. Another alternative is to build post-hoc tools which do not interfere with the model training but could be used to explain its prediction post-training. The post-hoc approaches are further divided into two; (i) Model specific and (ii) Model agnostic. The model-specific methods such as backwards propagation in neural networks necessitate awareness of the internal design of the model to facilitate the flow of information in the reverse direction [4]. Model agnostic methods do not make any assumptions regarding the internal workings of a black-box model and rely only on querying the black-box model to make predictions whenever required.

There is an inverse relation between model complexity and interpretability. From an interpretability standpoint, it is desired to have models which are simple and yet perform well on a given task. As the task gets increasingly complex, the simple models may not be flexible enough and may perform poorly. Opting for more sophisticated models may boost the performance provided the training data is sufficiently large. It is important to note that complex models are prone to overfitting and may not generalise well if sufficient data is not available for training. If the task to learn is fairly complex such that a simple model may not perform well, and the training data is also sufficiently large such that a complex model may generalize well, then there exists a trade-off between accuracy and interpretability as shown in Fig. 1.

The remaining sections of this chapter are organized as follows: Section 2 presents a comprehensive overview of several common metaheuristics and evolutionary algorithms. Following this, Sect. 3 examines the application of these techniques in attaining explainability within AI systems.

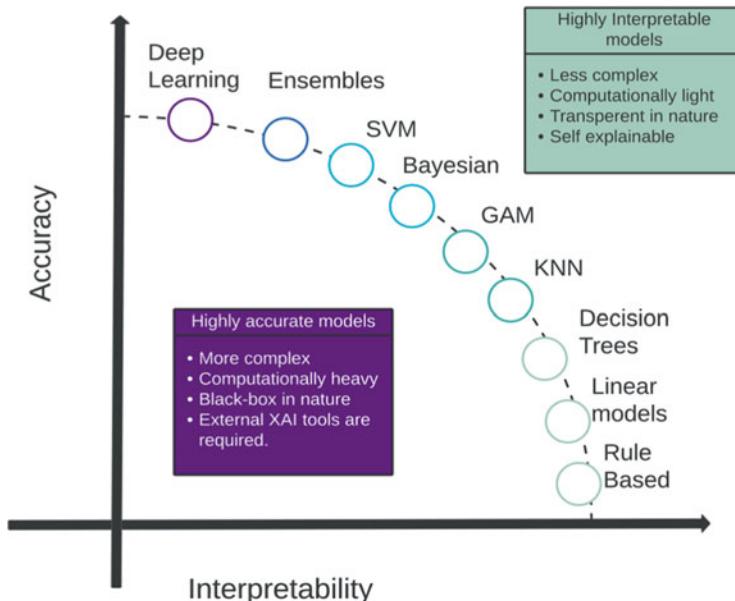


Fig. 1 The trade-off between machine learning model interpretability and accuracy

2 Evolutionary and Metaheuristic Algorithms

Evolutionary algorithms draw inspiration from the natural selection process, where a population of candidate solutions is evolved through selection, crossover, and mutation operations, leading to the discovery of better solutions over time. Common examples of evolutionary algorithms include genetic algorithms, evolutionary strategies, and genetic programming.

Metaheuristic algorithms are a family of optimization techniques which, unlike traditional optimization methods, do not rely on mathematical models and explicit problem-specific information. Metaheuristics are problem-independent and do not require a precise problem formulation. Instead, they use a set of heuristic rules and strategies to explore the search space and gradually improve the solution quality. These algorithms are intended to efficiently explore the search space by balancing exploration and exploitation of the solution space. Common examples of metaheuristic algorithms include simulated annealing, particle swarm optimization, differential evolution and ant colony optimization.

Evolutionary algorithms and metaheuristics are powerful tools that can be used out of the box to address a vast array of problems, including XAI [5]. This section provides detailed descriptions of some of these algorithms, which are employed in the XAI techniques discussed in the following section.

2.1 Genetic Algorithms

Genetic algorithms (GAs) [6] are a class of optimization algorithms that are inspired by the principles of natural selection. These algorithms work by imitating the process of evolution. These algorithms work by iteratively improving a population of potential solutions to a problem through operations like selection, crossover, and mutation. GA has been extensively researched and implemented in a variety of disciplines, such as engineering, computer science, economics, and biology, to solve complex optimization problems that are difficult or impossible to solve with traditional methods. Genetic algorithms are popular because of their ability to find optimal or near-optimal solutions in large, complex search spaces and their ability to handle non-linear and non-continuous objective functions.

To implement GA, one must first devise a way to represent the solution space. Encoding in genetic algorithms entails representing each potential solution to a problem as a chromosome which is a set or a vector of genes. This process is also known as chromosome encoding. In earlier works, binary encoding was popularized. Real value representation is another popular encoding mechanism. A more detailed review of various types of encoding is done in [7]. Value encoding is a good choice for encoding real-valued variables. For example, if the objective is to maximize $f(x_1, x_2) = \sin(x_1) + x_2$ then it may seem logical to encode the inputs as $x = [x_1, x_2]^T$, a real valued chromosome. Main Components of the GA are:

- Fitness Function: Each solution in the population is assigned a fitness value by the fitness function, which assesses the quality of a solution based on how well it solves the problem.
- Selection: The selection operation chooses a subset of solutions from the existing population, based on their fitness values, where solutions with better fitness values have a greater probability of being chosen.
- Crossover: The crossover operation combines the genetic material (parts of the chromosome) of the two parent solutions to generate new offspring solutions.
- Mutation: The mutation operation incorporates small random changes to an individual solution's chromosome. It aids in introducing new genetic material into the population, preventing the algorithm from being stuck in local optima.

There are various versions of implementing genetic algorithms. We present the pseudocode of the classical method [8]. It is the easiest in terms of implementation and it also contains all the main components of GA.

1. Initialize population: $P = \{x_1, x_2, \dots, x_n\}$, where x_i is a potential solution.
2. Evaluate fitness: Calculate the fitness value $f(x_i)$ for each x_i in P .
3. Repeat until a termination condition is met:
 - (a) Select parents: Choose two individuals from P with a probability proportional to their fitness. Let p_a and p_b be the selected parents.
 - (b) Crossover: Create a new individual p_o by recombining the genes of p_a and p_b .
 - (c) Mutate: With a small probability, mutate one or more genes in p_o .

- (d) Evaluate fitness: Calculate the fitness value $f(p_o)$ for p_o .
 - (e) Replace: Replace one of the individuals in P with p_o based on a fitness.
4. Return the best solution(s) found during the process.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [9] is a metaheuristic algorithm that draws inspiration from the social behaviour of fish schools and flocks of birds. Its appeal stems from its ease of use and effectiveness in addressing complicated optimization problems in continuous space. In PSO, a swarm of particles travel iteratively across a search space while employing social interaction and self-experience to navigate towards the best solution. Each particle's position represents a potential solution to the problem, and its movement is governed by both its best solution as well as the best solution discovered until now by the swarm.

At iteration t , the particle i moves in the search space by updating its position x_i^t by adding current velocity v_i^t to its previous position.

$$x_i^t = x_i^{t-1} + v_i^t \quad (1)$$

$$v_i^t = wv_i^{t-1} + c_1r_1(pbest_i^{t-1} - x_i^{t-1}) + c_2r_2(gbest^{t-1} - x_i^{t-1}) \quad (2)$$

Where $pbest_i^{t-1}$ is the personal best position (solution) achieved so far by the particle i . It is calculated using a fitness function. And $gbest^{t-1}$ is the best position for the entire swarm so far. c_1 and c_2 are the cognitive and social coefficients which are set by the user. r_1 and r_2 are random values generated uniformly between 0 and 1.

There are three components to the velocity of a particle; (i) Momentum, (ii) Cognitive, and (iii) Social.

- Momentum (wv_i^{t-1}): Weighted previous velocity vector. The weight is also referred to as inertia.
- Cognitive ($c_1r_1(pbest_i^{t-1} - x_i^{t-1})$): It is a component in the direction of the current personal best position for the particle from its previous position.
- Social ($c_2r_2(gbest^{t-1} - x_i^{t-1})$): It is a component in the direction of the current global best position for the particle from its previous position.

The PSO Pseudocode is as follows.

1. Initialize swarm size, the maximum number of iterations, and other parameters (w, c_1, c_2).
2. Randomly initialize each particle's position (x^0) and velocity (v^0).
3. Evaluate the fitness of each particle.
4. Set the personal best position ($pbest^0$) and personal best fitness of each particle to its current position and fitness.
5. Determine the global best position ($gbest^0$) based on the personal best position and fitness of all particles.

6. For each iteration (t):
 - (a) Update each particle's velocity (v^t) and position (x^t) using the Eqs. (1) and (2).
 - (b) Evaluate the fitness of each particle.
 - (c) For each particle, update its personal best position and its personal best fitness. Additionally, update the global best position.
7. Return the best solution(s) found during the process.

2.3 Differential Evolution

Differential evolution (DE) [10] is another population-based metaheuristic optimization approach. Similar to GA, DE is also inspired by natural selection, but they differ in the way they mimic it. In DE, an initial population of possible solutions is produced at random. Each individual in the population is represented by a vector of real values, hence individuals are also referred to as vectors. The elements of the vector are often called genes. DE uses the difference between the vectors of two individuals in the population to generate a new candidate solution. It involves the selection, crossover, and mutation of individuals to create new individuals for the next generation. It is important to note that GA and DE have distinct characteristics and applications. DE is a specialized algorithm that uses a specific mechanism to create new individuals, while GA is a more general approach that encompasses a range of methods. DE is often used for continuous optimization problems with few constraints.

In DE, a vector in the current population, also known as the target vector, might get substituted with a trial vector. To create the trial vector corresponding to the target vector, first, a mutant vector is created, which is a linear combination of some randomly picked vectors from the population. The trial vector is then created by performing a crossover operation on the mutant vector and the target vector. The target vector and trial vector are compared and based on the fitness only one of them is passed to the next generation.

2.3.1 Types of Vectors

- Target Vector (x_i): A vector in the population, which may get replaced.
- Mutant Vector (v_i): A vector which is generated by adding the difference between two randomly chosen vectors in the population to a third vector. All the chosen vectors are not the target vector.
- Trial vector (u_i): A vector which is generated by performing a crossover operation between the target vector and the mutant vector.

Let the population at iteration t be $P(t) = \{x_1^t, x_2^t, x_3^t \dots x_n^t\}$. Vector x_i^{t+1} for $P(t+1)$ is obtained by first setting x_i^t as the target vector. Then r_1, r_2 and r_3 are three distinct

vectors obtained from $P(t)$ which are not the same as the target vector. They are used to create a mutant vector v_i^{t+1} (Eq. 3). F is an user-defined constant between $[0, 2]$.

$$v_i^{t+1} = r_1 + F(r_2 - r_3) \quad (3)$$

Trial vector u_i^t is obtained by performing a crossover operation between the mutant and the target vectors. The j th element of u_i^{t+1} is obtained by first generating a random value r_j using uniform distribution between 0 and 1. p_c is the pre-defined crossover probability. Either the j th element of the target vector or the mutant vector is selected according to Eq. (4).

$$u_{ij}^{t+1} = \begin{cases} x_{ij}^t & \text{if } r_j > p_c \\ v_{ij}^{t+1} & \text{otherwise} \end{cases} \quad (4)$$

Ultimately, depending on fitness, either the target vector is retained or the trial vector replaces the target vector in $P(t + 1)$. The DE Pseudocode is as follows.

1. Initialize population $P(0)$ with random vectors (solutions) and evaluate their fitness
2. While the stopping criterion is not met, for each iteration (t):
 - (a) For each individual x_i^t in the population, $P(t)$ do:
 - i. Select three distinct individuals r_1 , r_2 , and r_3 from $P(t)$, not including x_i^t
 - ii. Generate a mutant vector v_i^{t+1}
 - iii. Generate a trial vector u_i^{t+1} by applying crossover between v_i^{t+1} and x_i^t
 - iv. Evaluate the fitness of u_i^{t+1}
 - v. If the fitness of u_i^{t+1} is better than x_i^t , then $x_i^{t+1} = u_i^{t+1}$ else $x_i^{t+1} = x_i^t$
 3. Return the best solution(s) from $P(T)$, T is the total number of iterations.

2.4 Multi-objective Optimization Using Non-dominated Sorting Genetic Algorithm II

The practice of simultaneously optimising many objectives is known as multi-objective optimisation. Unlike traditional optimization problems that focus on optimizing a single objective function, real-world situations often entail the need to optimize multiple objectives concurrently. For instance, minimizing costs while maximizing performance is a common example of conflicting objectives that need to be optimized together.

Multi-objective optimization techniques aim to find a set of solutions that represent a trade-off between conflicting objectives. These solutions are called Pareto-optimal solutions and represent the best possible compromise between the

objectives. The Pareto front is the set of all Pareto-optimal solutions. Pareto optimal solutions are solutions which cannot be dominated by any other solution. A solution is said to be dominating any other solution if, for all the objectives, the dominant solution is performing better or equal to the other solution. Refer to Fig. 2 for an illustration.

One way to simplify a multi-objective optimization task is to transform it into a single-objective optimization problem by computing a weighted sum of all the objectives. The problem is that it may not be straightforward to assign weights as choosing weights for the different objectives is often subjective and arbitrary. It is difficult to justify why one weight should be given more importance than another, and different weights can lead to different optimal solutions. Moreover, multi-objective optimization is often concerned with finding Pareto-optimal solutions which are as diverse as possible. Evolutionary algorithms are well-suited for multi-objective optimization tasks because they are population-based and can explore multiple regions in the solution space simultaneously.

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a popular evolutionary multi-objective optimization algorithm. The NSGA-II algorithm is based on the genetic algorithm framework and employs several novel strategies to address some of the limitations of its predecessor NSGA. NSGA-II is able to efficiently handle multiple objectives simultaneously. The algorithm operates by generating and

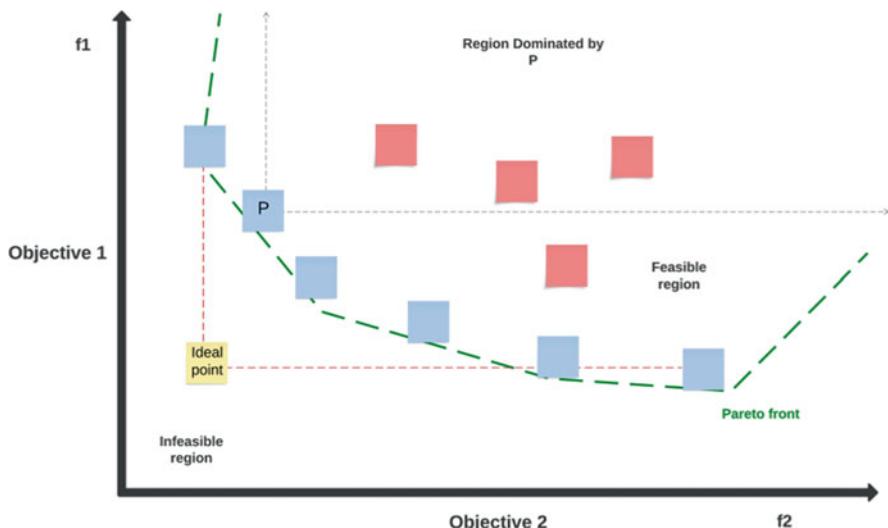


Fig. 2 Pareto optimal front for two objective functions. The region above the green dotted line is the feasible region. The goal is to minimize both objectives. Blue points indicate the non-dominated solutions which are also known as Pareto-optimal solutions. The yellow point indicates an ideal point which minimizes both objectives but lies in the infeasible region. Red points indicate sub-optimal solutions. Note that the points in the figure are in the “objective space” and don’t represent their spatial arrangement. An individual is mapped to a vector consisting of the values of the objectives $x \rightarrow (f_1(x), f_2(x))$

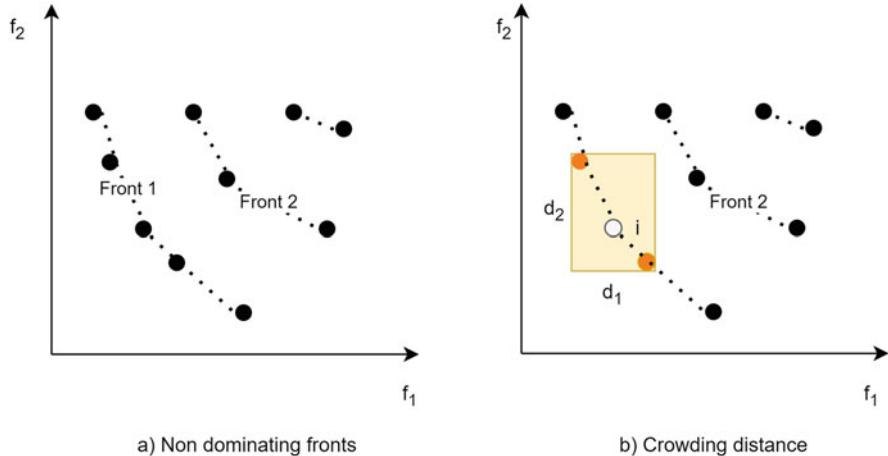


Fig. 3 For the multiobjective task involving minimization of both the objectives, f_1 and f_2 , the figures illustrate the concept of non-dominated fronts (**a**) and crowding distance (**b**). Crowding distance depends on the sides of the hyper-cuboid formed by the nearest solutions belonging to the same front in the objective space

evolving a population of candidate solutions (chromosomes), where each chromosome represents a potential solution to the optimization problem. NSGA-II uses a combination of non-dominated sorting and crowding distance measurement to rank the chromosomes based on their fitness values. The non-dominated sorting method is used to identify the non-dominated solutions, while the crowding distance measure is utilized to maintain diversity within the population.

Non-dominated sorting involves categorizing solutions into various levels or fronts, where each level contains solutions that are dominated by the solutions in the previous levels. The first front contains solutions that are not dominated by any other solutions in the population, while the second front contains solutions that are not dominated by the solutions not in the first front, and so on. The algorithm continues to sort the solutions until all the solutions are classified into different fronts. Crowding distance measures the degree of crowding around each solution in the population. Solutions with higher crowding distances are preferred over those with lower crowding distances. The solutions are first ranked based on their non-dominated front. A solution in the 1st front is superior to a solution in the 2nd front. For ranking solutions within the same front, crowding distance is used. Refer to Fig. 3 for an illustration of these concepts.

$$cdist(i) = \frac{d1}{f_1^{\max} - f_1^{\min}} + \frac{d2}{f_2^{\max} - f_2^{\min}}.$$

NSGA-II is a genetic algorithm which specializes in solving a multi-objective optimization task. The algorithm starts by generating a random population P_0 of

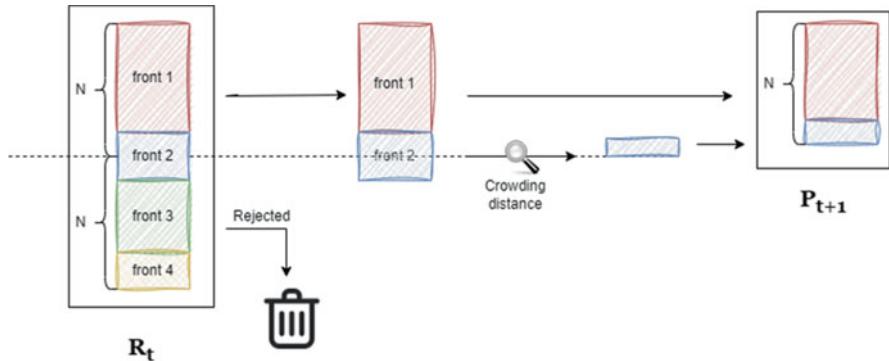


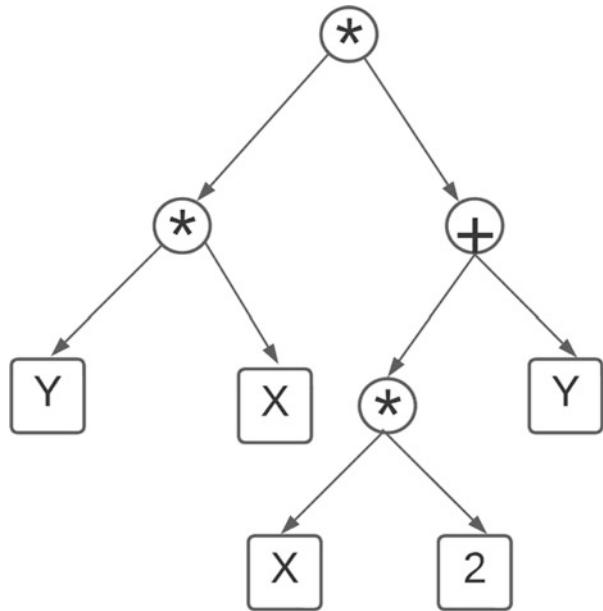
Fig. 4 Elitism in NSGA-II. The combined population R_t is first sorted into non-dominating fronts. Starting from the lower fronts, if the size of a front is less than the available size, it is added to P_{t+1} . After adding some fronts in order, if the size of the latest front to be added is less than the available size, then the front is sorted with the help of the crowding distance. The sorted front is clipped so that it could be fitted in the available space

size N . Non-dominant sorting and crowding distance are used to rank all solutions within the population. The fitness assigned to the solutions is equivalent to the level of the non-dominating front to which they belong. Hence, fitness is to be minimised. Binary tournament selection method is employed for the selection process. After performing crossover and mutation, the offspring population Q_0 is generated. At every iteration t , by incorporating an elitism mechanism S , the combined population $R_t = Q_t \cup P_t$ of size $2N$ is reduced to size N which is the population at iteration $t + 1$, $P_{t+1} = S(R_t, N)$. Figure 4 illustrates the elitism mechanism.

2.5 Genetic Programming

Genetic Programming (GP) [11] uses a technique inspired by natural evolution to create computer programs by automating the program development process, reducing the need for human intervention. It uses a population of computer programs, represented as a string of symbols, a tree or code, and evolves them over time through the application of genetic operators such as mutation and crossover. The fitness of each program is evaluated based on how well it performs on a set of test cases. GP has the potential to be highly beneficial in creating models for tasks involving regression or classification. For example, the function $f(x, y) = yx(2x + y)$ is represented as a binary tree in Fig. 5. The goal of genetic programming is to start with a random population of trees and then through the application of genetic operators evolve the population of trees such that their fitness is improved. The fitness of a tree measures the accuracy of the function represented by it, for example, the mean square error over a test dataset.

Fig. 5 Binary tree representing the expression: $f(x, y) = yx(2x + y)$. The terminal nodes (square-shaped) contain the model inputs and constants. The internal node represents functions and operations



A simple GP has the following basic components.

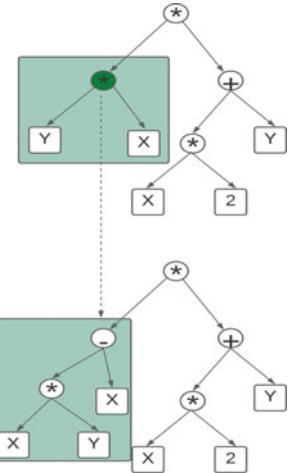
1. A mechanism to convert the population individual to a corresponding executable code.
2. A mechanism to measure the fitness of an individual.
3. Terminal set T containing all the input variables and constants.
4. Function set F containing all the simple functions and operations applicable on an internal node. For example, $F = \{+, -, *, /, max\}$
5. Control parameters similar to genetic algorithms which include population size, mutation rate, and crossover rate, among others.

A set of Genetic operations include,

1. Selection: After the fitness of the population has been evaluated, the algorithm selects fitter individuals as parents with probability proportional to the fitness.
2. Reproduction: Some of the fittest individuals are simply carried on to the next generation.
3. Crossover: It involves randomly selecting nodes for a crossover between the two parent trees. The subtrees rooted at the selected nodes (below) are exchanged between the two parents and hence two offspring are created.

Mutation: It involves randomly selecting a node for mutation. A randomly generated tree replaces the subtree rooted at the selected node. See Fig. 6 for an illustration. It is mutated to a tree given just below. It is done by first selecting a node in the tree

Fig. 6 A binary tree representing the expression:
 $f(x, y) = yx(2x + y)$



(green coloured). Then the subtree rooted at the selected node is replaced by a randomly generated tree. Essentially replacing yx in the expression by some $g(x, y)$.

Operationally GP is the same as GA, it begins with a random population of trees and applies genetic operations at each iteration to improve the population's fitness through incremental updates. The final population generated after meeting certain termination criteria is used to determine the fittest program tree(s).

3 Application of Metaheuristic and Evolutionary Algorithms in XAI

Explainability could be viewed through the lens of optimization. For example, balancing the accuracy and interpretability of AI systems is itself a multi-objective optimization task, as there is a trade-off between these two factors. Achieving high accuracy often requires more complex models that can be difficult to interpret, while achieving high interpretability may require sacrificing some level of accuracy.

In addition to balancing accuracy and interpretability, post-hoc techniques such as searching for counterfactual explanations can be viewed as a constrained optimization task. These techniques aim to find a set of inputs that, when modified, would result in a different output from the AI system. This requires searching for input values that satisfy certain constraints, such as similarity to the original input, belonging to the data distribution and giving desired outputs from the model.

Another popular XAI technique of fitting local surrogate models involves generating samples that are close to the point that needs to be explained and belong to the data distribution. These objectives can be formulated as optimization problems.

Evolutionary and metaheuristic algorithms are well-suited to solving optimization problems in XAI because they are flexible in terms of problem definition. These algorithms can be applied to a wide range of optimization tasks, including multi-objective optimization and constrained optimization. Furthermore, they can be used to optimize complex, non-linear functions that are difficult to solve using traditional optimization techniques.

Overall, viewing XAI as an optimization task provides a useful framework for developing new methods and algorithms for achieving better interpretability in AI systems. By applying evolutionary and metaheuristic algorithms to these optimization problems, it is possible to develop more effective and efficient techniques for achieving explainability in AI systems. In this section, we will look at some of the existing implementations.

3.1 Counterfactual Explanations

The first formalization of counterfactual explanations in machine learning can be traced back to the work of Wachter et al. [12]. In this paper, the authors proposed using counterfactual explanations as a way to comply with the transparency and accountability requirements of the European Union's General Data Protection Regulation (GDPR).

A counterfactual explanation provides an explanation by showing how changing a particular set of input features would have resulted in a different prediction by a complex machine learning model. Counterfactual explanations are beneficial in situations where the model's output needs to be changed or improved. By generating a counterfactual explanation, users can see how performing minimal changes to specific input features can lead to a more desirable outcome. For example, after getting a loan application rejected by a bank, one might wonder what went wrong, more specifically why the application fell short. In this regard, the bank could return a counterfactual, which could be in the form of "if the personal income had been more by x amount and the current debt was lower by y amount the loan may have been approved". By doing so, it not only properly justifies its decision-making but also provides an actionable recourse. It is to be noted that for some features, the change suggested may not be actionable. For example, someone can't age in the reverse direction nor can change their race. One way to deal with it is to limit the search space of the counterfactual to what changes of feature values are realistically possible. Nevertheless, such explanations may help uncover hidden biases in the system.

Evolutionary and metaheuristic algorithms are advantageous for finding counterfactuals because they are highly effective at navigating and searching through large solution spaces, this helps in producing accurate and diversified counterfactual explanations. Additionally, they can be used to optimize multiple objectives simultaneously [13]. However, It should be noted that an algorithm's effectiveness in finding optimal counterfactuals depends heavily on the reliability of the objective

function. If the objective function is not well-defined or does not reflect the desired outcome precisely, the algorithm may converge on suboptimal solutions. For a classification model, the objective function could be as simple as Eq. (5).

$$\begin{aligned} & \min_c d(x, c) \\ & s.t. f(x) \neq f(c) \end{aligned} \quad (5)$$

Where x is the original instance, $d(x, c)$ is the distance between the counterfactual c and x , and f is the black-box model.

These methods are also model agnostic, which means they can work on a wide range of model types without the requirement of model parameters as inputs. This is particularly useful when a company is reluctant to share the model's internal processes due to privacy concerns but still needs to comply with regulations.

Before discussing the implementation details of different counterfactual generation algorithms, it is important to understand the concept of distance, which plays a crucial role in counterfactual generation.

3.1.1 Data Types and Distance Measures

A key feature of a counterfactual is that it should be as close to the input instance as possible. A proper distance measure is crucial for generating counterfactuals because it determines how close the generated counterfactuals are to the original instance. The distance measure is used to define the closeness between the original instance and a counterfactual instance in the feature space. Defining a good distance measure may not be straightforward as real-world data comes in different forms.

First, let's examine the scenario where all the characteristics are numeric. In this case, each data point is situated in an R^n space, where n is the total number of features. As distance measures, two extremely popular options are the L_1 (mean absolute error or MAE) distance and L_2 (mean squared error or MSE) distance.

$$MAE(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (6)$$

$$MSE(x, y) = \sum_{i=1}^n (x_i - y_i)^2 \quad (7)$$

Using these distances may not be appropriate as the features may not be of equal scale, and this may lead to some features dominating others. The right approach is to take a weighted distance.

One possible definition for the distance function d could be the weighted L_1 distance, weighted by the inverse of the Median Absolute Deviation (MAD) of each feature taken over some data points P , typically the training dataset.

$$d(x, y) = \sum_{i=1}^n \frac{|x_i - y_i|}{w_i} \quad (8)$$

Where

$$w_i = median_{j \in P}(x_{j,i}) - median_{l \in P}(x_{l,i}) \quad (9)$$

In many real-world scenarios, data is not purely numeric, but rather a mix of different data types. This is particularly true in the banking industry, where data can include not only numeric values such as account balances, transaction amounts, and interest rates, but also non-numeric data such as customer names, addresses, and account types. A proper distance measure should be defined in order to deal with mixed data types.

One such distance measure that can accommodate both numerical and categorical features is the Gower distance [13]. The Gower distance between x and y with mixed attributes is the following.

$$d(x, y) = \frac{1}{n} \sum_{i=1}^n \delta(x_i, y_i) \quad (10)$$

Where

$$\delta(x_i, y_i) = \begin{cases} \frac{1}{R_i} |x_i - y_i| & \text{if numerical} \\ \mathbf{1}_{x_i \neq y_i} & \text{if categorical} \end{cases} \quad (11)$$

R_i is the observed range for numerical feature i . Typically calculated on the training dataset.

3.1.2 Counterfactuals Using Genetic Algorithms

Implementing a genetic algorithm to produce counterfactuals includes randomly picking values for each variable to build an initial population of potential counterfactuals. The fitness function would then evaluate each solution in the population in terms of how similar it is to the original input, and whether it is delivering the intended prediction from the black-box model. The algorithm would then employ genetic operations such as selection, crossover, and mutation to generate a new population of potential counterfactuals. This process is repeated until a condition for termination is reached or a predefined number of iterations is reached. Figure 7 gives an illustration for a binary classification model. The green and the brown regions are the regions for the two classes. Counterfactual search could simply be viewed as the search for the nearest points of the opposite class. The difficulty arises when dealing

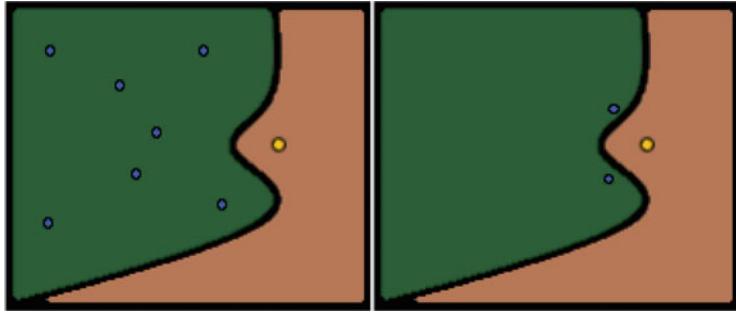


Fig. 7 Any population-based algorithm begins with an initial population of sub-optimal solutions (left). The population is improved over several generations and finally, the solutions which have high fitness are returned as counterfactuals (right)

with complex models, where we don't know the precise location of the decision boundary (curve separating the classes).

Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence (CERTIFAI) [14] uses a custom genetic algorithm to generate counterfactuals for a given instance. The black-box model is used for a classification task. The genetic algorithm is used to get a counterfactual belonging to a different class than the original instance. An open-source implementation is present at <https://github.com/Ighina/CERTIFAI>. The algorithm's key components are outlined below.

The authors have defined two distance measures, one for mixed attributes tabular data (Eq. 12) and another for image data (Eq. 13). The *NormAbs* is the weighted L_1 distance defined in Eq. (8). *SimpMat*, a simple matching distance is selected to measure the distance between categorical attributes.

$$d(x, c) = \text{NormAbs}(x, c) + \text{SimpMat}(x, c) \quad (12)$$

$$d(x, c) = \frac{1}{\text{SSIM}(x, c)} \quad (13)$$

For images, *SSIM* (Structural Similarity Index Measure) is used to define the distance. The optimization objective used is the simple one defined earlier (Eq. 5). The fitness function is defined as the reciprocal of the distance from the selected instance x . The closer the point to x , the higher its fitness (Eq. 14).

$$\text{fitness}(c) = \frac{1}{d(x, c)} \quad (14)$$

Since we are interested in the points which are close but also belong to a different class after every iteration the population is filtered such that only solutions that satisfy the belonging to the different class constraint remain and then fitness is calculated. The black-box model is used to determine the class of a solution. The

initial population is generated such that all the solutions belong to a class which isn't the same as the class of the instance x . Further constraints could be imposed on the search space to make the counterfactual realistically attainable, for example, the race of a person cannot be changed. This is done by passing bounds for the features as parameters. Infeasible solutions are removed after every iteration. The mutation operation is done by slightly perturbing a few of the feature values. And crossover is done by simply interchanging some feature values between two solutions.

CERTIFAI introduces a more generalized, flexible approach to counterfactual generation that works for any model and data type. Most prior counterfactual methods like Russell et al. [15] and Ustun et al. [16] are limited to only generating counterfactuals for linear models. In contrast, CERTIFAI uses a genetic algorithm that is model-agnostic and can generate counterfactuals for any black-box model, including complex nonlinear models like neural networks. Some existing methods like Wachter et al. [12] cannot handle mixed data types or categorical features. CERTIFAI is flexible to generate counterfactuals directly from any combination of raw categorical, discrete, and continuous features. Some techniques [15] generate a single counterfactual explanation for a given instance. CERTIFAI can easily produce multiple diverse counterfactuals by taking the top individuals from the genetic algorithm.

CERTIFAI also introduces the idea of using counterfactuals for evaluating model robustness and fairness. This provides additional use cases beyond just explainability. The genetic algorithm approach also allows CERTIFAI to incorporate user-defined constraints on feature ranges to produce more feasible and realistic counterfactuals.

In CERTIFAI, the authors showcase multiple use cases of counterfactual generation. The experiments are done on the UCI adult dataset and the Pima Indian diabetes dataset, along with some standard datasets such as iris and breast cancer. Counterfactuals generated are used for the following.

1. Robustness: Measuring the robustness of the machine learning model against any adversarial attack. This is done by treating counterfactuals as adversarial examples and measuring the expected distance between input instances and their corresponding counterfactuals.
2. Fairness: The concept of burden is defined as the expected distance between input instances belonging to a particular group having an undesired outcome, and their corresponding counterfactuals. It reflects the degree of change required for instances belonging to the different partitions of the population, to change the undesired outcome. The burden is calculated for different ethnic groups present in the UCI adult dataset and based on the values, it can be seen that the black-box model is biased towards whites and unfair towards the black ethnic group.
3. Explainability: For explaining instances from both the UCI adult and the Pima Indian datasets, multiple counterfactuals are generated with or without constraints. The counterfactuals are also used for identifying the importance assigned to the features by the black-box model. The importance of a feature depends upon

the number of times that feature is changed in order to generate counterfactuals across the dataset.

3.1.3 Counterfactuals Using Particle Swarm Optimization and Differential Evolution

In the work of Anderson et al. [17], DE and PSO alternatives are suggested as a replacement for simple GA. For an optimization task (counterfactual generations) in a continuous space, a simple GA algorithm can struggle to converge [18, 19]. On the other hand, PSO and DE are often used in optimization tasks in continuous spaces. An open-source implementation is present at <https://github.com/HaydenAndersen/ECCOUNTERFACTUALS>. The key components of both algorithms are outlined below.

The algorithms are proposed to work on continuous spaces, which means all features are numerical. That's why the L_1 norm (Eq. 6) is preferred as the choice of the distance measure. However, in their work, the data is assumed to be scaled beforehand. If not, it may be preferable to use weighted distance instead (Eq. 8). The authors generated the starting population using a $U(0, 1)$ uniform distribution. This is because the data is assumed to be scaled, and all the features are lying between 0 and 1.

The choice of fitness function for both algorithms is the same, it is exactly the distance from the selected instance x . Since the distance is directly taken as the fitness function, the lower the fitness, the better the generated candidate. If the prediction made by the black-box model for a candidate is not a desired one, for example, in classification, if the candidate is predicted to belong to the same class as x . Then the fitness assigned is ∞ .

The algorithm is mostly the same as the one discussed in Sect. 2. The initial population is generated randomly. Each particle is assigned fitness as ∞ , as lower fitness is better. After every iteration, the personal best position for a particle is the one with the lowest fitness attained by it so far. The parameters take the standard values, these are $w = 0.7298$, $c_1 = 1.4962$, and $c_2 = 1.4962$ [20].

Similar to PSO, the DE algorithm is almost the same as discussed in Sect. 2. Just as in PSO, the initial population is generated randomly and each vector is assigned fitness as ∞ . Instead of taking a fixed value of $F \in [0, 2]$, it is drawn uniformly at every mutation from $U(F_{min}, F_{max})$, the range (F_{min}, F_{max}) is user-defined. The default parameter values are used for DE as given in the SciPy Python library.

The authors have performed a qualitative study as well as a comparative study. The datasets used for the studies have all continuous attributes. The proposed algorithms are compared with CERTIFAI. The comparison is done by measuring the mean number of features modified while generating counterfactuals, and also the mean of L1 and L2 distances from the original points to the counterfactuals. The datasets used for the study have all continuous attributes. The idea is that if the compared metrics are lower, then the quality of the counterfactuals generated is higher. It is to be noted that this won't account for the diversity observed in the

generated counterfactuals. Qualitative analysis is done on the Penguins dataset to verify if the changes suggested by the counterfactuals are reasonable.

3.1.4 Multi Objective Counterfactuals Using NSGA-II

Dandl et al. [13] propose the search of counterfactuals as a multi-objective search. Their work formalizes the concept of using multi-objective optimization for counterfactual generation (MOC). For the instance to be explained, MOC returns a diverse collection of counterfactuals, which corresponds to the Pareto front of a four-objective optimization problem.

Let $f : \mathcal{X} \rightarrow R$ be the black-box model mapping the feature space \mathcal{X} to a real-valued output and x is the selected instance for which counterfactuals are to be generated. $Y \subset R$ is the subset of the desired outcome. It could be a range of values such that $f(x) \notin Y$. For the classification task, $f(x)$ could be the probability returned by the black-box model for a particular class. A counterfactual c has certain desired properties.

1. $f(c)$ should be as close as possible to the desired set Y
2. c is very close to x in \mathcal{X} space
3. c differs from x in only few features
4. c is lying within the data distribution in \mathcal{X}

These properties are expressed in terms of a multi-objective optimization task.

$$\min_c O(c) := \min_c (O_1(f(c), Y), O_2(c, x), O_3(c, x), O_4(c, X^{obs})) \quad (15)$$

Where

$$O_1(f(c), Y) = \begin{cases} 0 & \text{if } f(c) \in Y \\ \inf_{y \in Y} |f(c) - y| & \text{otherwise} \end{cases} \quad (16)$$

$$O_2(c, x) = \text{Gower distance}(x, c) \quad (17)$$

$$O_3(c, x) = \|c - x\|_0 \quad (18)$$

$$O_4(c, X^{obs}) = \sum_{j=1}^k w^{[j]} \text{Gower distance}(x^{[j]}, c) \quad (19)$$

The objective functions O_1 , O_2 , O_3 and O_4 reflect the desired properties of a counterfactual. Gower distance (Eq. 10) measures the distance between a solution c and x . The objective O_3 counts the number of features for which x and c don't have the same values. It is done by counting the number of non-zero elements of $x - c$. The objective O_4 involves a collection of observations X^{obs} , and the training data

used to train the black-box model is the preferred choice $X^{train} = X^{obs}$. It measures the weighted distance between c and its first k neighbours in X^{obs} .

For searching counterfactuals with multiple objectives, a slightly modified version of NSGA-II is implemented. Since the solution could be a mixture of continuous and discrete attributes, a mixed integer strategy [21] is used for the genetic algorithm. The crowding distance is also modified to take into account not only the distance in the objective space but also the distance in the solution space. An open-source implementation is present at <https://github.com/dandls/moc>.

The authors demonstrate the viability of developing a diverse range of counterfactuals for cases in the German credit dataset. A comparative study is also done with state-of-the-art counterfactual generation tools such as DiCE [22], Recourse [16] and Tweaking [23]. This is done on the datasets accessed from the OpenML platform. The metric used for comparison is the coverage rate [24], which is the measure of the relative frequency of the counterfactuals generated by a particular method dominated by MOC. If the coverage rate is 1 then there exists at least one counterfactual generated by MOC which dominates it.

3.1.5 Additional Works

It can be observed from Table 1 that further research has been conducted on generating counterfactuals [33] utilizing evolutionary and metaheuristic algorithms.

The table also reflects the significance of regarding counterfactual generation as a multi-objective optimization problem. The objectives are associated with some of the desirable qualities of counterfactuals such as (i) correctness in outcome (validity), (ii) closeness to the original instance (proximity), (iii) less number of features changed (sparsity), (iv) within feasibility constraints (actionability), (v) several alternatives for recourse (diversity) and (vi) belonging to the data distribution.

Table 1 Counterfactuals using evolutionary and metaheuristic algorithms

Paper	Technique used	Open source implementation
[25]	Custom genetic algorithm	https://github.com/mjschleich/GeCo.jl
[26]	Genetic algorithm with local search	https://github.com/michael-lash/Inverse_Classification
[27]	Speed-constrained multi-objective particle swarm optimization	None found
[28]	Multi-objective optimization with NSGA-III	https://github.com/peymanrasouli/CARE
[29]	Custom genetic algorithm	None found
[30]	Multi-objective optimization with U-NSGA-III	https://github.com/wmonteiro-ai/xmoai
[31]	Multi objective optimization with NSGA-II	https://github.com/tridungduong16/multiobj-scm-cf
[32]	Custom genetic algorithm	https://github.com/masoudhashemi/PermuteAttack

3.2 Local Surrogate Modelling

Local surrogate modelling involves creating an interpretable model that approximates the behaviour of the black-box model in a specific region of input space. The resulting surrogate model can be used to provide insights into how the black-box model works in that region of input space. Exists an inherent compromise between the interpretability and the performance of a model. A simple interpretable model may not perform well on the entire test dataset when compared to the more intricate black-box model. Therefore, the objective is not to mimic the complex black-box model entirely, instead, the idea is to use a simple interpretable model to mimic the black-box model in a locality. This is analogous to the idea of Taylor series expansion in mathematics which suggests that a real-valued function could be approximated to a lower-order polynomial around a small neighbourhood. Local surrogate modelling has gained a lot of popularity in XAI. LIME [34] is a very popular technique which falls under this category. All these techniques have a generic framework which is the following.

1. Select an instance of interest from the dataset for which the prediction has to be explained.
2. Generate a neighbourhood of local data points around the instance of interest by perturbing the features of the instance within a certain range.
3. Train a simpler, interpretable model, such as a decision tree or linear regression, on the local data points and their corresponding black-box model predictions as targets.
4. Examine the surrogate model to get a more understandable view of the black-box model's prediction mechanism in the vicinity of the instance of interest.

The quality of explanations produced by local surrogate models is highly influenced by the quality of the local neighbourhood. If the generated data is biased or not representative of the population, the explanations may not accurately reflect the behaviour of the black-box model on the population instances. LIME uses independent Gaussian distributions to generate a sample. It has been shown that explanations using such a simple mechanism of generating a local sample are prone to adversarial attacks [35].

Another challenge with local datasets is that they may not contain enough information to capture the complexity of the black-box model accurately. This can result in the surrogate model being overly simplistic, and the explanations generated by the model may not provide a complete understanding of the behaviour of the black-box model.

3.2.1 Local Rule-Based Explanations

The quality of a local explanation relies heavily on the quality of the local neighbourhood generated around the instance of interest. LORE [36] uses a genetic

algorithm for generating a good neighbourhood around the instance to be explained for a binary classification task. A decision tree model is fitted and a single rule is obtained along with some counterfactual rules extracted from the tree structure which are returned as an explanation. The LORE algorithm investigates the black-box model's decision boundaries around the instance for which the prediction has to be explained. The idea is the decision boundary is best captured by not only the region near the point of interest but also the region around the nearest counterfactual and by sampling points in these regions, the generated neighbourhood has points belonging to both classes. The genetic algorithm is used only for the neighbourhood generation. An open-source implementation is present at <https://github.com/riccotti/LORE>. The algorithm's key components are outlined below. The genetic algorithm uses two custom fitness measures $\text{fitness}_=^x$ and fitness_\neq^x , for generating the set of points belonging to the same and the different class denoted by $Z_$ and Z_\neq respectively. $Z = Z_ \cup Z_\neq$ is the generated neighbourhood on which a decision tree model is fitted.

$$\text{fitness}_=^x(z) = \mathbf{1}_{f(x)=f(y)} + (1 - d_x(z)) - \mathbf{1}_{x=y} \quad (20)$$

$$\text{fitness}_\neq^x(z) = \mathbf{1}_{f(x)\neq f(y)} + (1 - d_x(z)) - \mathbf{1}_{x=y} \quad (21)$$

Where $x \in \mathcal{X}$ is the instance of interest. $d : \mathcal{X} \rightarrow [0, 1]$ is a function which measures the distance from x . $f(z)$ is the black-box model prediction. Assume that there are m features overall, and h of which are categorical features. Then the distance is a weighted sum of the distance between the categorical and the numerical features.

$$d_x(z) = \frac{h}{m} \text{SimpMat}(x, z) + \frac{m-h}{m} \text{NormEuclid}(x, z) \quad (22)$$

The genetic algorithm is used twice, for generating $Z_$ and Z_\neq . The parameters of the algorithm are x , fitness , N , p_c and p_m . First, the population of size N is initialized $P_0 = \{x_i | x_i = x; \forall i = 1, 2, \dots, n\}$. All the values are set to x . The generation counter i is set to 0. While $i < G$, P_{i+1} is updated from P_i in the following way.

1. Perform the selection operation and select a portion of the fittest sub-population of P_i as parents.
2. Perform a 2-point crossover operation to generate offsprings using crossover probability p_c . Update the population as the union of the parents and the offsprings.
3. Perform mutation on a portion of the population using probability p_m . Update the population as the union of the mutated and unmutated individuals.
4. Based on fitness, select the top N individuals as the new population P_{i+1} .

After the neighbourhood generation using GA, a decision tree model is trained using the neighbourhood data Z and the target labels $f(Z)$ as the black-box model predictions on it. LORE returns an explanation $e = \{r, \Phi\}$. Where r is a single rule and Φ is a collection of counterfactual rules. The conjunction of split conditions along a

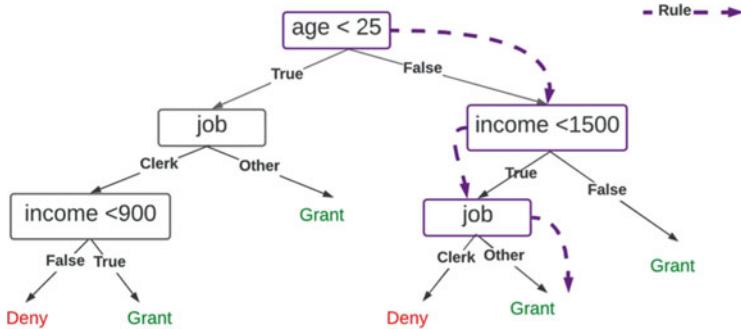


Fig. 8 For $x = \{\text{age: 27, income: 1200, job: other}\}$, the purple path shows the single rule for the explanation of the instance, $r = \{\text{age} \geq 25, \text{income} \leq 1500, \text{job: other}\} \rightarrow \text{"Grant"}$. All the paths leading to the class “Deny” are candidates for the counterfactual rules

path of the local decision tree which is satisfied by x forms r . See Fig. 8 for an illustration.

The number of split conditions that are not fulfilled by x is tallied for each possible counterfactual rule. For a counterfactual rule, if it is below a certain threshold, then the rule is added to Φ .

The authors have performed experiments on three datasets containing both categorical and numerical attributes. These are adult, compas and german. All the datasets represent attributes of a person. The datasets are used to perform binary classification. After performing a train-test split, a black-box model (b) is fitted and the quality of the mimicking the black-box model locally of LORE is assessed.

In order to fit a decision tree (c) around an instance x , a train set Z is generated. This set Z is used to measure five evaluation metrics. These are (i) fidelity: measured by comparing the predictions of b and c over Z. (ii) l-fidelity: The subset of Z which satisfy the single rule generated using LORE is used for calculating fidelity. (iii) cl-fidelity: The subset of Z which satisfies the counterfactual rules generated using LORE is used for calculating fidelity. (iv) hit: It compares $b(x)$ and $c(x)$ directly. If $b(x) = c(x)$, then 1 else 0. (v) c-hit: It measures the hit of one instance which is derived from the counterfactual rules. The average value is measured for all the evaluation metrics over the test dataset.

A comparison study is also done by using these metrics. LIME and ANCHOR [37] are the two state-of-the-art surrogate modelling techniques used for the study.

3.2.2 Genetic Programming Explainer

LIME operates under the assumption that for a complex black-box model, a linear model serves as a reliable local approximation. To explain a model prediction for an instance, a linear model is fitted on a weighted neighbourhood around the instance and the coefficients of the model are returned as the feature importance values. In

some cases, this assumption could lead to a considerable reduction in the precision of the local model's ability to replicate the complex black-box model's behaviour in the surrounding area of the instance of interest.

Genetic Programming Explainer (GPX) [38] uses genetic programming to generate a more accurate interpretable mimicking model. GPX assumes that all the features are numeric. A candidate model is represented as a binary tree. For more details regarding GP refer to Sect. 2. The model generated is not bounded by linearity but at the same time has a closed mathematical expression. If the function set of GP contains only simple arithmetic operations and differentiable functions, then it allows the local surrogate model to be differentiable, and the partial derivative of a feature could be calculated to study the sensitivity of the black-box model with respect to the feature. Let $f_b : R^n \rightarrow R$ be the black-box model, and x is the instance for which the prediction $f_b(x)$ has to be explained. GPX returns a local surrogate model f^* which is easy to interpret and also mimics the model f_b in the neighbourhood around x . An open-source implementation is present at <https://github.com/leauferreira/GpX>. The algorithm's key components are outlined below.

Since all the features are numerical, the local neighbourhood is generated by using a multivariate Gaussian distribution centred at x . It assumes all the features are independent. The covariance matrix is $\Sigma = \sigma I_n$, where σ is calculated over the training data.

The objective of GPX is to generate a function f^* such that the distance between $f^*(N(x))$ and $f_b(N(x))$ is minimum.

$$f^* = \arg \min_f d(f(N(x)), f_b(N(x))) \quad (23)$$

Where d could be L_2 norm (Eq. 7). $N(x)$ is the generated neighbourhood of size H around the point x . $f(N(x)) = [f(s_1), f(s_2) \dots f(s_H)]^\top$ such that $s_i \in N(x) \forall i \in \{1, 2, 3 \dots H\}$.

Given a neighbourhood around an instance, GPX creates a local surrogate model which mimics the black-box model in the neighbourhood. The surrogate model is a simple non-linear expression of features. The complexity could be controlled by limiting the depth of the generated tree. The interpretability comes from the fact that the local surrogate model is a simple non-linear, differentiable mathematical expression.

The authors have used twenty different datasets for performing experiments. These datasets are extracted from popular repositories such as the UCI Machine Learning repository, OpenML, Kaggle and scikit-learn. The experiments are done in two parts, (i) A comparative study is done with LIME and Locally Generated Decision Trees to measure the mimicking capability of the surrogate models. (ii) A case study is done to interpret a Random Forest Regressor on the Boston and Diabetes dataset using GPX.

For the comparative study, the accuracy (classification task) or the mean square error (regression task) is used. For a particular instance x , a neighbourhood is generated around x then the local model is trained. The accuracy or the mean square

error of the local model on the generated neighbourhood is evaluated by taking the black-box model predictions as the target values. This is performed on all the instances in the test dataset and the average over all the test instances is returned.

3.3 *Transparent Models*

Transparent models are those that are easy to understand and interpret, and their decision-making process is clear and explicit, making them useful in contexts where understanding the model's decision-making process is important.

3.3.1 Decision Tree

Decision trees [39] are constructed using a set of simple rules that help to partition the data into smaller subsets. At each step of the decision tree construction, the algorithm selects the best feature or attribute that divides the data into subsets. This is done by evaluating various criteria such as information gain. Once the best feature is selected, a value is chosen to split the data into two or more subsets. The process is repeated recursively until a stopping criterion is satisfied, which could be a maximum tree depth or a minimum number of training examples in each leaf node. The leaves are assigned a class label or a regression value, depending on the task.

Because decision trees are constructed using a series of simple rules that are based on the values of the input features, they are easy to interpret and explain to humans. It is possible to visualize a decision tree and see the decisions made at each node, which makes it clear how the model is making its predictions.

A lot of work has been done on applying evolutionary algorithms for decision tree generation. While any optimization technique could be implemented to tune the hyper-parameters of the tree, our interest lies in algorithms that are specifically used in the tree-building process. The primary motivation for adopting such algorithms is that the traditional tree-building algorithms are greedy. Once a node is created there is no way of backtracking and changing the attribute split condition later in the tree-building process. In contrast, evolutionary algorithms provide a global optimization approach for tree construction that the traditional methods lack.

Evo-Tree [40] is an example where an evolutionary algorithm (genetic algorithm) is used in the tree-building process. The trees start with a random population and evolve using a multi-objective GA. To be accurate, it works with a single objective which is a weighted aggregate of two objectives. The two objectives are validation set accuracy and tree size. It is to be noted that balancing complexity and accuracy as competing objectives and employing multi-objective optimization to solve is a powerful idea. And despite its potential benefits, a single objective optimization is performed. Components of the Evo-Tree GA are,

1. Fitness function: The fitness function is a combination of both objectives mentioned earlier, with weights assigned to each

$$\text{fitness} = \alpha_1 f_1 + \alpha_2 f_2 \quad (24)$$

$$f_1 = 1 - \text{acc} \quad (25)$$

$$f_2 = \frac{\text{Tree current depth}}{\text{Tree target depth}} \quad (26)$$

Where f_1 is the measure of inaccuracy on a validation set, and f_2 , is the measure of tree complexity in terms of tree depth.

2. Genetic Representation: A Decision tree is represented as a chromosome. There are several ways to encode the tree into a chromosome, the easiest and the most natural way to do it is by using tree encoding. A decision tree is represented by a binary tree, where each node has an attribute and also a threshold value. It represents the splitting condition based on an attribute value threshold at the corresponding node of the decision tree. It works under the assumption that all attributes are numeric in nature, and the categorical features are to be converted into numeric beforehand.
3. Crossover: A child is created by performing a crossover between two parent trees. A node is randomly selected for both trees as the crossover point. The subtree rooted at the crossover point for the first parent tree is replaced by the subtree rooted at the crossover point of the second parent tree, creating a new child tree.
4. Mutation: A node condition mutation is implemented. First, a node is randomly selected as the mutation point. The attribute and the threshold value present at the node are replaced by some arbitrary attribute and value.

Figure 9 illustrates the crossover and mutation operations performed on parent tree(s). There are a lot of various implementations of evolutionary algorithms in building decision trees. Tree-GA hybrid [41, 42] has a two-stage training scheme. First, a D-tree is created by using a traditional algorithm, and then a population of all the paths from the root to the distinct leaves are treated as tree rules. To be more specific, the conjunction of the split conditions satisfied along a particular path makes a rule. Then the population of rules is evolved using GA. Genetic programming could also be used for creating decision trees [43]. In the work of Evans et al. [44], a multi-objective genetic programming approach is suggested. The objectives are improving accuracy and reducing tree complexity. The implementation is a hybrid between NSGA-II and GP.

3.3.2 Learning Classifier System

A Learning Classifier System (LCS) [45] is a rule-based approach which maintains a population of rules or classifiers. These rules are evaluated on the basis of their fitness in solving a particular problem and the population evolves over time through

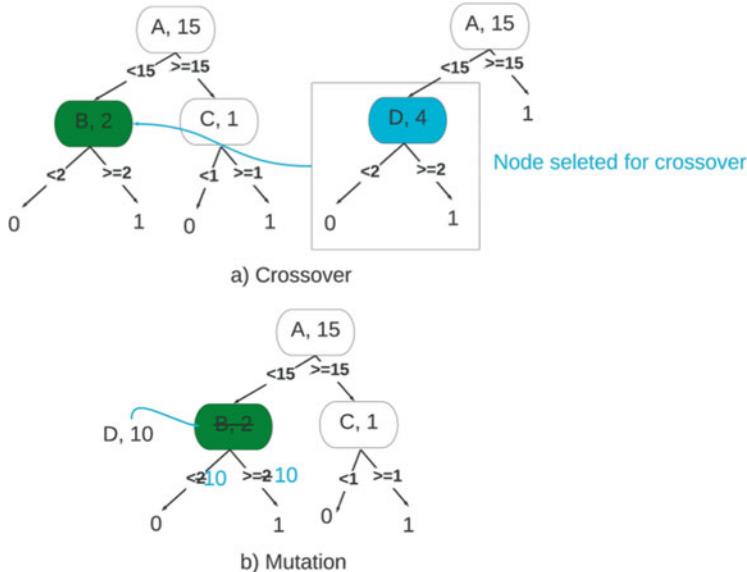


Fig. 9 Genetic operations in Evo-Tree algorithm

a process of selection, recombination, and mutation. It is important to note that LCS is more of a programming paradigm rather than a specific method. It is made up of several components which could be modified as per the task. LCS has been widely used in both supervised as well as in reinforcement learning.

The components may vary based on the specification of the task. The basic components of a generic LCS are listed below.

1. Current Knowledge: The finite population of the classifier or rules that represents the system's overall body of collective knowledge.
2. Performance Component: It involves the interplay between the environment and the classifier population.
3. Credit Assignment (Reinforcement): It distributes the reward (updates fitness) to the classifiers after interaction with the environment.
4. Discovery: Discovers or modifies the population of classifiers using Genetic Algorithm.

The elements may vary based on the task specification. To be more specific, we are considering the Michigan-style implementation for a binary classification task [46].

- Environment: The environment is a source of experience. It is simply the training data for the classification task. In the Michigan-style system, where the learning is incremental, only one training instance is used for one learning cycle.
- Classifier: A classifier or a rule is an “if condition then action” statement. In the context of classification, it could be viewed as the condition being a conjunction of conditions on feature values and action being the class label. Note that a

classifier itself is not a complete ML model. For a given test example, it may be possible that a particular rule is not applicable. It is more appropriate to treat it as a local model while the entire population of classifiers collectively forms the ML model.

- Matching set: While training or predicting a single instance is queried. The matching set is a subset of the population of classifiers whose conditions are satisfied (matched) by the instance.
- Covering mechanism: In the training cycle, if no classifier belongs to the matching set, then new classifiers are introduced which match the instance by a covering mechanism.

A classifier or rule has a condition which is a conjunction of conditions on the individual feature values. For continuous features, it could be an interval of values and for discrete features, it could be a particular value that it could take. Wildcard “#” is the condition on a feature which allows it to take any possible value. One instance from the training set is chosen for a learning cycle. If the feature values of the selected instance satisfy all the feature conditions of a rule, then the rule is said to “match” and is moved to the matching set. See Fig. 10.

A covering mechanism is triggered if none of the population’s rules matches the training instance. New rules are derived based on the feature values of the training instance. For example, if all the features are discrete, some of the features are

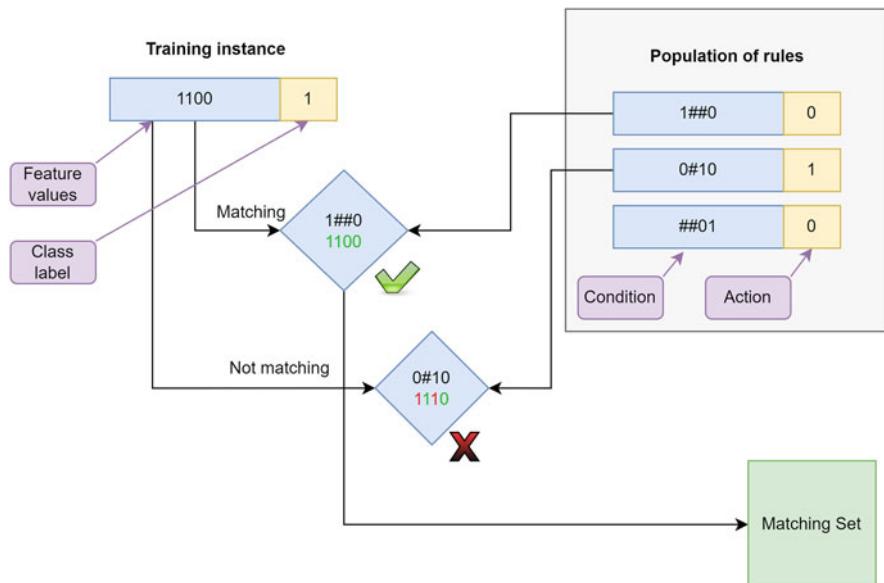


Fig. 10 The matching process in LCS. The training instance is shown on the left with its attributes taking boolean values, belonging to class 1. The classifiers in the population set are compared one by one. If the non-wildcard attributes of the classifier and the instance are identical then the classifier is included in the matching set

arbitrarily selected and assigned the feature condition “#” wildcard. For the rest of the features, the condition allows them to only take the same value as the training instance (Eq. 27). If the population is already at capacity, the new rules replace some of the rules present in the population.

$$\text{Instance} : [1, 2, 2, 0, 1] \rightarrow 1; \text{New Rule} : [\#, 2, \#, 0, 1] \rightarrow 1 \quad (27)$$

UCS [47, 48] is an LCS framework, which is used specifically in the supervised learning setting. Following is the description of a learning cycle for UCS for the binary classification task.

There are certain parameters that are associated with a classifier. These parameters are updated in every training cycle. In UCS, the main parameters are (i) accuracy (acc), (ii) fitness (F), (iii) numerosity (num) and (iv) experience (exp).

$$acc = \frac{\text{number of correct classifications}}{\text{experience}} \quad (28)$$

$$F = (acc)^c \quad (29)$$

Where c is a user-defined constant. The experience is incremented whenever the classifier participates in a Matching Set. The numerosity is the count of the number of replicas of the classifier in the classifier population. Fitness is used in the rule discovery which is done by the GA. It is also used to regulate the population size of the classifier population.

First a training instance x is selected from the training set. LCS begins with an empty population [P] of classifiers, in contrast to other population-based evolutionary algorithms. In the first cycle, the population is automatically filled by the covering mechanism. In a usual cycle, the training instance x is compared one by one with the condition of the classifiers in [P] to create a matching set [M]. The set [M] is further divided into two subsets, correct [C] and incorrect classifiers [IC]. It is done by comparing the action of the classifiers with the class label associated with x . At this point, if [C] is empty, then the new rules, created using the covering mechanism, replace some of the rules in [P]. So far, the performance and the credit assignment components of the LCS are only involved. Now new rules are generated by the discovery component by using the genetic algorithm. GA selects two classifiers from the population as parents based on fitness values. These parents create two offspring by applying recombination and mutation operations. The current population is updated by inserting these new classifiers, while some classifiers are removed to preserve the population size.

Additionally, a subsumption mechanism could be applied in every cycle. It removes the rules which are overly specific and redundant, in presence of a more general rule which has similar accuracy. In this process, the numerosity of the general rule, which covers the feature space of the redundant rule, is incremented.

At the time of prediction, a matching set [M] of rules is created. Then based on fitness and numerosity, a weighted vote of all the classifiers in [M] is used in order to

make a prediction. The list of the rules used to generate a prediction is a list of simple “IF:THEN” statements which are human readable. Investigating too many rules at a time in a complex system may become incomprehensible for a human reader. In these cases, the transparency of the model may be reduced, making it more challenging to comprehend the decisions being made by the system. To tackle this, several visualization techniques have also been implemented [49, 50].

X’Overall, whether LCS can be considered as a transparent model depends on the specific implementation, and the complexity of the rules learned. But LCS can be considered to be more interpretable than some other machine learning approaches. Additionally, because LCS uses an evolutionary process to learn the rules, it is possible to trace the evolution of the rules over time and understand how they have changed and adapted to different situations. This can provide insights into the decision-making process and make the model more transparent.

4 Concluding Remarks

The importance of explainability in artificial intelligence (AI) systems has been highlighted in this book chapter, and the use of evolutionary and metaheuristic algorithms to achieve it has been discussed. Various research efforts in this area have been outlined, including counterfactual explanations, local surrogate modeling, and transparent models. Counterfactual explanations involve generating alternative scenarios that could have led to a different outcome, while local surrogate modeling aims to simplify complex black-box models. Transparent models, in contrast, are inherently interpretable and can provide insights into how they reached a specific decision or prediction.

Given the increasing role of AI in our lives, it is crucial that we prioritize transparency and accountability in these systems. To achieve this, evolutionary and metaheuristic algorithms are well-suited and flexible for some of the optimization tasks involved in explainability. These algorithms can handle high-dimensional and nonlinear search spaces effectively and can handle multiple competing objectives and constraints. Furthermore, their ability to explore a wide range of potential solutions makes them useful. We conclude that evolutionary and metaheuristic algorithms are valuable tools for developing trustworthy and responsible AI systems.

References

1. Preece A, Harborne D, Braines D, Tomsett R, Chakraborty S. Stakeholders in explainable AI. 2018;arXiv preprint arXiv:181000184.
2. Mehrabi N, Morstatter F, Saxena N, Lerman K, Galstyan A. A survey on bias and fairness in machine learning. ACM Comput Surv. 2021;54(6):1–35.
3. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, García S, Gil-López S, Molina D, Benjamins R, Chatila R. Explainable Artificial Intelligence (XAI):

- concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf Fusion.* 2020;58: 82–115.
- 4. Montavon G, Samek W, Müller KR. Methods for interpreting and understanding deep neural networks. *Digit Signal Process.* 2018;73:1–15.
 - 5. Bacardit J, Brownlee AE, Cagnoni S, Iacca G, McCall J, Walker D. The intersection of evolutionary computation and explainable AI. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. July 2022. p. 1757–62.
 - 6. Holland JH. Genetic algorithms. *Sci Am.* 1992;267(1):66–73.
 - 7. Kumar A. Encoding schemes in genetic algorithm. *Int J Adv Res IT Eng.* 2013;2(3):1–7.
 - 8. Katouch S, Chauhan SS, Kumar V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl.* 2021;80:8091–126.
 - 9. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks (Vol. 4). IEEE; 1995. p. 1942–48.
 - 10. Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim.* 1997;11(4):341–459.
 - 11. Koza J. Genetic programming, on programming of computer by natural selection. *Stat Comput.* 1994;4:87–112.
 - 12. Wachter S, Mittelstadt B, Russell C. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harv JL Tech.* 2017;31:841.
 - 13. Dandl S, Molnar C, Binder M, Bischi B. Multi-objective counterfactual explanations. In: Parallel Problem Solving from Nature—PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5–9, 2020. Proceedings, Part I, Springer; 2020. p. 448–69.
 - 14. Sharma S, Henderson J, Ghosh J. Certifai: a common framework to provide explanations and analyse the fairness and robustness of black-box models. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. 2020. p. 166–72.
 - 15. Russell C. Efficient search for diverse coherent explanations. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. 2019. p. 20–28.
 - 16. Ustun B, Spangher A, Liu Y. Actionable recourse in linear classification. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. 2019. p. 10–19.
 - 17. Andersen H, Lensen A, Browne WN, Mei Y. Evolving counterfactual explanations with particle swarm optimization and differential evolution. In: 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2022. p. 1–8.
 - 18. Rasheed K, Hirsh H, Gelsey A. A genetic algorithm for continuous design space search. *Artif Intell Eng.* 1997;11(3):295–305.
 - 19. Hassan R, Cohenim B, De Weck O, Venter G. A comparison of particle swarm optimization and the genetic algorithm. In: 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. 2005. p. 1897.
 - 20. Van den Bergh F. An analysis of particle swarm optimizers. PhD diss., Department of Computer Science, University of Pretoria, Pretoria, South Africa. 2002.
 - 21. Li R, Emmerich MT, Eggermont J, Bäck T, Schütz M, Dijkstra J, Reiber JH. Mixed integer evolution strategies for parameter optimization. *Evol Comput.* 2013;21(1):29–64.
 - 22. Mothilal RK, Sharma A, Tan C. Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. 2020. p. 607–17.
 - 23. Tolomei G, Silvestri F, Haines A, Lalmas M. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017. p. 465–74.
 - 24. Zitzler E, Thiele L. Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Parallel Problem Solving from Nature—PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998. Proceedings 5, Springer; 1998. p. 292–301.

25. Schleich M, Geng Z, Zhang Y, Suciu D. Geco: quality counterfactual explanations in real time. 2021; arXiv preprint arXiv:210101292.
26. Lash MT, Lin Q, Street N, Robinson JG, Ohlmann J. Generalized inverse classification. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM. 2017. p. 162–70.
27. Barredo-Arrieta A, Del Ser J. Plausible counterfactuals: auditing deep learning classifiers with realistic adversarial examples. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE; July 2020. p. 1–7.
28. Rasouli P, Chieh Yu I. Care: coherent actionable recourse based on sound counterfactual explanations. *Int J Data Sci Anal*. 2022; <https://doi.org/10.1007/s41060-022-00365-6>.
29. Dastile X, Celik T, Vandierendonck H. Model-agnostic counterfactual explanations in credit scoring. *IEEE Access*. 2022;10:69543–54.
30. Monteiro WR, Reynoso-Meza G. Counterfactual generation through multi-objective constrained optimisation. 2022.
31. Duong TD, Li Q, Xu G. Prototype-based counterfactual explanation for causal classification 2105.00703. 2021.
32. Hashemi M, Fathi A. Permuteattack: counterfactual explanation of machine learning credit scorecards. 2020;arXiv preprint arXiv:200810138.
33. Navas-Palencia G. Optimal counterfactual explanations for scorecard modelling. 2021; arXiv preprint arXiv:210408619.
34. Ribeiro MT, Singh S, Guestrin C. “Why should I trust you?” Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. p. 1135–44.
35. Slack D, Hilgard S, Jia E, Singh S, Lakkaraju H. Fooling lime and shap: adversarial attacks on post hoc explanation methods. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. 2020. p. 180–86.
36. Guidotti R, Monreale A, Ruggieri S, Pedreschi D, Turini F, Giannotti F. Local rule-based explanations of black box decision systems. 2018;arXiv preprint arXiv:180510820.
37. Ribeiro MT, Singh S, Guestrin C. Anchors: high-precision model-agnostic explanations. In: AAAI Conference on Artificial Intelligence (AAAI). 2018.
38. Ferreira LA, Guimaraes FG, Silva R. Applying genetic programming to improve interpretability in machine learning models. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2020. p. 1–8.
39. Quinlan JR. Simplifying decision trees. *Int J Man Mach Stud*. 1987;27(3):221–34.
40. Jankowski D, Jackowski K. Evolutionary algorithm for decision tree induction. In: Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5–7, 2014. Proceedings 14, Springer; 2014. p. 23–32.
41. Ds L, Sj F. A modified decision tree algorithm based on genetic algorithm for mobile user classification problem. *Sci World J*. 2014;2014:468324.
42. Carvalho DR, Freitas AA. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. In: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. 2000. p. 1061–68.
43. Espejo PG, Ventura S, Herrera F. A survey on the application of genetic programming to classification. *IEEE Trans Syst Man Cybern Part C Appl Rev*. 2009;40(2):121–44.
44. Evans BP, Xue B, Zhang M. What’s inside the black-box? A genetic programming method for interpreting complex machine learning models. In: Proceedings of the Genetic and Evolutionary Computation Conference. 2019. p. 1012–20.
45. Urbanowicz RJ, Moore JH. Learning classifier systems: a complete introduction, review, and roadmap. *J Artif Evol Appl*. 2009;2009:736398.
46. Urbanowicz R, Browne W. Introducing rule-based machine learning: a practical guide. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. 2015. p. 263–92.

47. Bernadó-Mansilla E, Garrell-Guiu JM. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evol Comput*. 2003;11(3):209–38.
48. Orriols-Puig A, Bernadó-Mansilla E. A further look at UCS classifier system. GECCO'06; 2006. p. 8–12.
49. Liu Y, Browne WN, Xue B. Visualizations for rule-based machine learning. *Nat Comput*. 2022;21:243–64.
50. Urbanowicz RJ, Granizo-Mackenzie A, Moore JH. An analysis pipeline with statistical and visualization-guided knowledge discovery for Michigan-style learning classifier systems. *IEEE Comput Intell Mag*. 2012;7(4):35–45.

Evolutionary Dynamic Optimization and Machine Learning



Abdennour Boulesnane

Abstract Evolutionary Computation (EC) has emerged as a powerful field of Artificial Intelligence, inspired by nature's mechanisms of gradual development. However, EC approaches often face challenges such as stagnation, diversity loss, computational complexity, population initialization, and premature convergence. To overcome these limitations, researchers have integrated learning algorithms with evolutionary techniques. This integration harnesses the valuable data generated by EC algorithms during iterative searches, providing insights into the search space and population dynamics. Similarly, the relationship between evolutionary algorithms and Machine Learning (ML) is reciprocal, as EC methods offer exceptional opportunities for optimizing complex ML tasks characterized by noisy, inaccurate, and dynamic objective functions. These hybrid techniques, known as Evolutionary Machine Learning (EML), have been applied at various stages of the ML process. EC techniques play a vital role in tasks such as data balancing, feature selection, and model training optimization. Moreover, ML tasks often require dynamic optimization, for which Evolutionary Dynamic Optimization (EDO) is valuable. This paper presents the first comprehensive exploration of reciprocal integration between EDO and ML. The study aims to stimulate interest in the evolutionary learning community and inspire innovative contributions in this domain.

Keywords Evolutionary computation · Evolutionary dynamic optimization · Metaheuristics · Dynamic optimization problems · Machine learning

A. Boulesnane (✉)

BIOSTIM Laboratory, Medicine Faculty, Salah Boubnider University Constantine 03, Constantine, Algeria

e-mail: aboulesnane@univ-constantine3.dz

1 Introduction

Evolutionary Computation (EC) is an extraordinary field of Artificial Intelligence that draws inspiration from nature's mechanisms responsible for the gradual development of intelligent organisms throughout millennia [1]. EC techniques have emerged as highly efficient and effective problem-solving methods by emulating these natural processes. These algorithms employ individuals' populations, each striving to find optimal solutions for specific challenges [2].

However, EC approaches face challenges that hinder their optimal performance. These obstacles often manifest as a tendency to get stuck in suboptimal solutions, diversity loss, computational complexity, population initialization, and premature convergence. Researchers have sought to integrate learning algorithms with evolutionary techniques to overcome these limitations [1]. This integration aims to address the aforementioned challenges and enhance the overall performance of EC methods. The fundamental idea behind this approach is to harness the wealth of data generated by the EC algorithm during its iterative search. This data contains valuable insights into the search space, problem characteristics, and population dynamics. By incorporating learning techniques, this data can be thoroughly analyzed and exploited to significantly improve the effectiveness of the search process [3].

Interestingly, the relationship between evolutionary algorithms and Machine Learning (ML) goes both ways. ML tasks often involve intricate optimization problems characterized by noisy, non-continuous, non-unique, inaccurate, dynamic, and multi-optimal objective functions [4]. In such complex scenarios, evolutionary computing algorithms, renowned for their versatility and stochastic search methods, offer exceptional opportunities for optimization. Consequently, several EC approaches have emerged in recent years at various stages of the ML process, ranging from pre-processing and learning to post-processing, to overcome traditional methods' limitations. These innovative hybrid techniques are collectively known as Evolutionary Machine Learning (EML) [5].

During the pre-processing stage, utilizing EC techniques becomes valuable for tasks such as data balancing and feature selection. As we delve into the ML learning phase, EC approaches autonomously play a vital role in determining essential components of ML models, encompassing hyperparameters and architecture. Furthermore, in the post-processing stage, EC can be applied, for instance, to optimize rules or facilitate ensemble learning. Therefore, the integration of EC techniques into ML signifies a profound paradigm shift, empowering the development of ML systems that are both robust and adaptable [6].

ML tasks are typically carried out in a real-world setting that undergoes dynamic changes, leading to the need for dynamic optimization. In such scenarios, the optimization problem's objective, constraint, or solution space can undergo variations over time. Evolutionary Dynamic Optimization (EDO) is a widely used approach that can swiftly adapt to these dynamic changes, making it valuable for practical applications in dynamic optimization. In this context, this paper introduces an innovative investigation into the convergence of EDO and ML techniques. To the

best of our knowledge, this study represents a thorough investigation into the reciprocal relationship between EDO and ML. By offering up-to-date insights on using EDO approaches in ML and the integration of ML into EDO, our primary objective is to stimulate interest and inspire the evolutionary learning community, fostering the development of innovative contributions in this domain.

In the remainder of our paper, we introduce EDO in Sect. 2, discussing its principles for solving optimization problems in dynamic environments. Section 3 provides an overview of ML. Section 4 focuses on applying ML to resolve dynamic optimization problems. Section 5 delves into utilizing EDO in ML, showcasing the synergy between these two domains. Finally, in Sect. 6, we conclude by summarizing our investigation and suggesting future research directions.

2 Evolutionary Dynamic Optimization

Dynamic optimization, also known as optimization in dynamic environments, is a highly active and extensively researched field due to its direct applicability to real-world problems. The inherent challenge lies in addressing the dynamic nature of these problems, which require finding optimal solutions within specific time constraints. These types of problems are called Dynamic Optimization Problems (DOPs) [7].

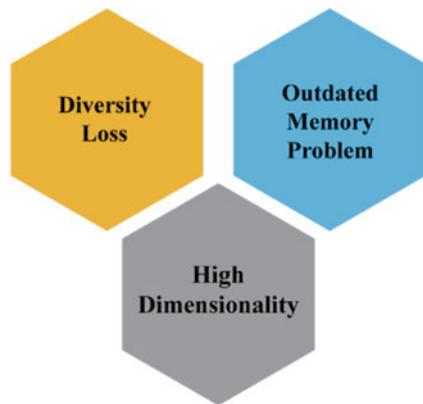
Formally, a DOP can be defined as the task of finding the optimal solutions $(x_1^*, x_2^*, \dots, x_n^*)$ that optimize the time-dependent objective function $f(x, t)$, as follows:

$$\begin{aligned} & \text{Optimize } (\max / \min) f(x, t) \text{ with } x \in R^n \\ & \text{subject to } h_j(x, t) = 0 \text{ for } j = 1, 2, \dots, u. \\ & g_k(x, t) \leq 0 \text{ for } k = 1, 2, \dots, v. \end{aligned} \quad (1)$$

Where $h_j(x, t)$ represents the j^{th} equality constraint and $g_k(x, t)$ represents the k^{th} inequality constraint.

In the literature, Dynamic Multiobjective Optimization (DMO) [8], Dynamic Constrained Optimization (DCO) [9], Robust Optimization over Time (ROOT) [10], and Dynamic Time-Linkage Optimization (DTO) [11] are closely related to DOPs. This latter revolves around optimizing systems that change over time, and these specific classes of problems delve into various aspects within such dynamic contexts. DMO focuses on simultaneously optimizing multiple conflicting objectives as the system evolves. DCO deals with optimization problems with constraints that must be satisfied throughout the dynamic process. ROOT emphasizes finding resilient and robust solutions against uncertainties and changes over time. DTO tackles the optimization of interdependencies and linkages between different time steps or stages of a dynamic problem. Together, these classes of problems

Fig. 1 Complex challenges in the realm of evolutionary dynamic optimization



encompass a comprehensive range of challenges and considerations when addressing dynamic optimization scenarios.

Solving DOPs requires an algorithm to find the best solution and adapt to environmental changes efficiently. However, DOPs present complex challenges, including issues like diversity loss, the persistence of outdated memory, and coping with large-scale dimensions, as depicted in Fig. 1. Diversity loss happens when all potential solutions converge to a single area in the search space, posing difficulties in exploring new optima after environmental changes. At the same time, the outdated memory problem arises when the information accumulated during the search process becomes invalid or irrelevant after a dynamic change occurs. The integration of obsolete knowledge during the search process can yield misleading outcomes and impede the dynamic optimizer's aptitude to locate the global optimum amidst a dynamic environment [12]. Furthermore, with the emergence of big data, another recent challenge is sometimes the high dimensionality of DOPs. Dealing with large-scale dimensions is a significant challenge when facing DOPs [13]. Exploring all potential solutions becomes challenging as the problem's complexity and scale escalate, leading to an exponential growth of the search space. Large-scale dimensions often result in longer computation times and increased memory requirements, making the optimization process computationally expensive.

Effectively solving DOPs requires addressing these critical challenges. In this regard, researchers have drawn inspiration from biological evolution and natural self-organized systems in the last two decades, leading to the widespread use of Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) methods [7]. These techniques offer inherent adaptability and resilience to handle environmental changes, making them well-suited for optimizing DOPs. The field dedicated to applying EAs and similar approaches to address DOPs is known as Evolutionary Dynamic Optimization (EDO). By using the principles of EAs and SI, EDO aims to tackle the complexities of DOPs and provide efficient solutions that can adapt to dynamic environments.

3 Machine Learning

Artificial intelligence encompasses a wide range of techniques, and machine learning (ML) is one specific subset. ML empowers computer systems to learn and improve from data without explicit instructions [14]. It encompasses various techniques designed to enable computers to analyze and interpret vast amounts of information effectively. As shown in Fig. 2, three primary types of ML techniques exist: Supervised, Unsupervised, and Reinforcement learning [15].

- **Supervised learning** involves training algorithms with labeled examples where the expected outputs are already known. The algorithm discerns underlying patterns and relationships by examining these labeled instances, effectively learning a function that maps inputs to outputs. This technique enables predictions and classifications for new, unseen data points. Examples of supervised learning algorithms include decision trees, random forests, support vector machines, and neural networks [16].
- **Unsupervised learning** allows algorithms to learn from unlabeled data without predefined outputs. In this scenario, the algorithms independently discover hidden patterns and structures within the data without explicit guidance. Unsupervised learning utilizes clusterings and dimensionality reduction algorithms, like K-means clustering and principal component analysis (PCA), to detect patterns or resemblances within the data, enabling the exploration and extraction of insights and knowledge [17].
- **Reinforcement learning** focuses on training software agents to learn through trial and error while interacting with an environment. The agents are given feedback in the form of rewards or penalties based on their actions, and they make choices with the goal of maximizing the total rewards they accumulate over a period of time. This technique is particularly valuable in game playing, robotics, and autonomous systems, where agents can continuously learn and adapt their behavior to achieve desired goals [18].

Apart from the above paradigms, there are other valuable ML techniques. Semisupervised learning [19] combines labeled and unlabeled data to improve

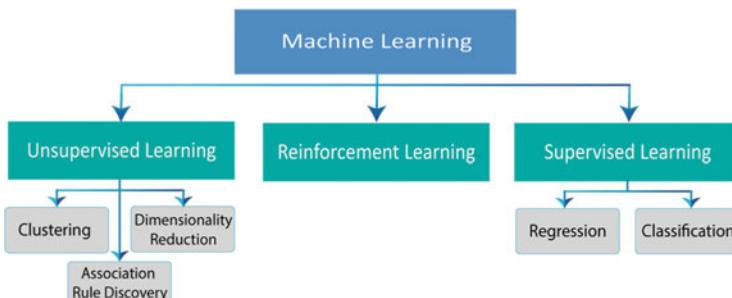


Fig. 2 Taxonomy of machine learning

training efficiency. Transfer learning [20] utilizes knowledge from a related problem or domain with abundant labeled data to enhance performance on a target problem with limited training data. Multitask learning [21] trains models to tackle multiple related tasks simultaneously, exploiting shared information and differences to enhance learning efficiency and prediction accuracy. These techniques broaden the scope of ML, allowing for more effective use of data and knowledge to achieve better results in various applications. Furthermore, Deep learning is another significant ML subfield that has recently gained immense popularity [22]. It has impressively advanced the field of ML, pushing the boundaries of what computers can achieve in tasks such as image classification, object detection, machine translation, and more [23].

4 Machine Learning for Resolving Dynamic Optimization Problems

Integrating ML in optimization enables more efficient and accurate problem-solving, enhancing decision-making processes in diverse fields such as logistics, finance, healthcare, and manufacturing [24]. ML algorithms can effectively learn patterns, relationships, and trends to find optimal solutions by using the accumulated data during optimization. Furthermore, ML can be combined with traditional optimization methods to create hybrid approaches that embrace the strengths of both paradigms [25].

On the other hand, DOPs have attracted significant attention due to their capacity to capture the nonstationary nature inherent in real-world problems. These problems require robust algorithms to discover optimal solutions in ever-changing or uncertain environments. Given these challenges, researchers have increasingly embraced the application of ML paradigms as powerful tools to address DOPs, supplementing traditional EC approaches. By incorporating ML paradigms like Transfer Learning, Supervised Learning, Reinforcement Learning, and others (see Table 1), the goal is

Table 1 ML techniques used for resolving DOPs and corresponding references

ML technique	DOP type	References
Transfer learning	DMOPs	[8, 26–34]
Supervised learning	DMOPs	[35–43]
	DOPs	[44–50]
Reinforcement learning	DMOPs	[51]
	DOPs	[52, 53]
	DTPs	[11]
Unsupervised learning	DMOPs	[54]
	DOPs	[55–60]
Deep learning	DMOPs	[61]
Ensemble learning	DMOPs	[62]
Online learning	DMOPs	[63]
Dual learning	DMOPs	[64]

to enhance the adaptability and efficiency of algorithms. This enables them to dynamically adjust, learn, and evolve their strategies in response to changing problem landscapes.

4.1 Transfer Learning-Based

In the literature realm, many approaches and algorithms have been put forth to tackle the complexities of dynamic multi-objective optimization problems (DMOPs) by harnessing the power of transfer learning techniques [26]. DMOPs encompass optimization problems with multiple conflicting objectives that evolve over time, posing a significant hurdle in effectively tracking the ever-changing set of Pareto-optimal solutions [8].

Transfer Learning, a method that involves leveraging past experiences and knowledge gained from previous computational processes, has garnered considerable attention due to its capacity to adapt to environmental changes and tap into valuable knowledge acquired in the past [26]. By employing transfer learning in DMOPs, the task of efficiently and accurately tracing the evolving Pareto-optimal fronts is greatly facilitated.

Various methods have been proposed to exploit past experiences and enhance the optimization process's performance, including manifold transfer learning [27, 28], individual transfer learning [29], regression transfer learning [30], and clustering difference-based transfer learning [31].

Nonetheless, some common issues have been identified, such as the loss of population diversity and high computational consumption. To overcome these challenges, hybrid methods have emerged as a solution. These methods often integrate transfer learning with other strategies, such as elitism-based mechanisms [32], surrogate models [33], manifold learning [28], or Bayesian classification [34]. The goal of combining these techniques is to improve the quality of the initial population, accelerate convergence, maintain diversity, improve computational efficiency, and achieve robust prediction, as shown in Fig. 3.

4.2 Supervised Learning-Based

Recently, supervised learning methods have played a crucial role in addressing DOPs by exploiting historical data and employing predictive models like artificial neural networks (ANN) [35, 44–49], kernel ridge regression (KRR) [40], and support vector machines (SVM) [36, 37, 43, 50]. As depicted in Fig. 4, these models enable the estimation and prediction of the behaviour of optimization problems in changing environments.

ANNS are trained using historical data to forecast future optimal solutions or estimate the position of the next optimum. They can generate initial solutions or

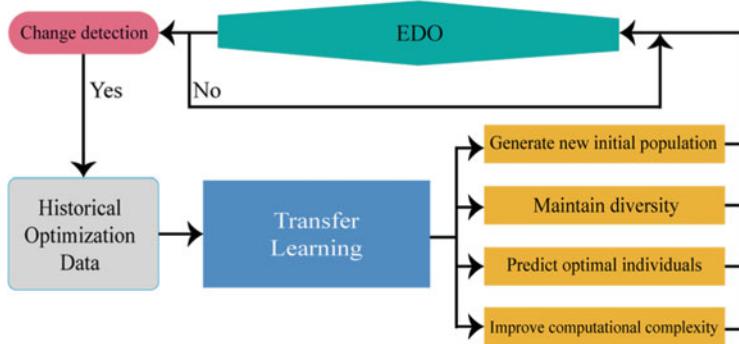


Fig. 3 Synergizing transfer learning and EDO

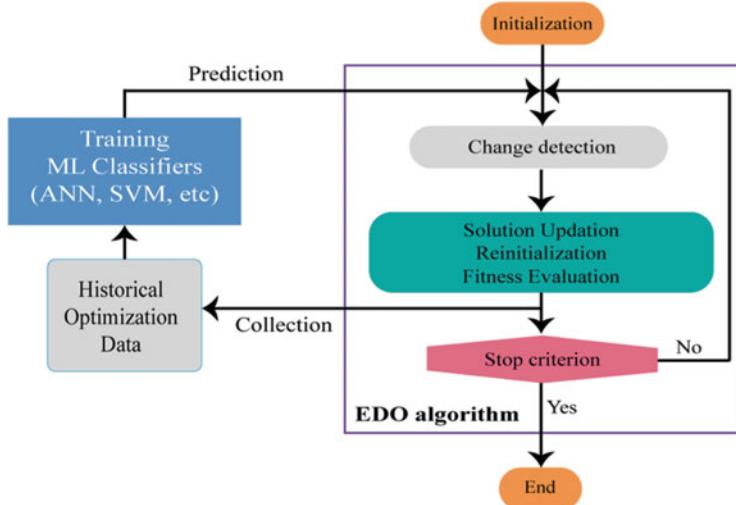


Fig. 4 Employing supervised learning with EDO to address DOPs

guide the evolutionary process towards promising regions of the solution space. By accelerating convergence and improving accuracy in tracking the optimum, ANNs contribute to enhanced performance [39].

Additionally, researchers have developed prediction mechanisms that quickly adapt to changes in the problem environment. Strategies based on KRR, Gaussian kernel functions [40], or Inverse Gaussian process [38] are employed to anticipate future changes and optimize evolutionary algorithms accordingly. Effective predictions play a vital role in maintaining solution quality and enhancing the search performance of the population.

SVMs are extensively utilized to tackle the challenges of DOPs. These models are trained using historical data to accurately model and predict solutions or information

in dynamic scenarios. SVMs can capture linear and nonlinear correlations between past and present solutions, making them well-suited for this task. Notably, SVM-based DOPs often employ incremental learning [41, 42], continuously updating the SVM model with the latest optimal solutions obtained from previous periods. This iterative approach effectively incorporates knowledge from all historical optimal solutions. As a result, incremental learning facilitates real-time exploration of nonlinear correlations between solutions, enhancing the model's predictive capabilities.

4.3 Reinforcement Learning-Based

The utilization of reinforcement learning (RL) methods to tackle DOPs is an emerging and promising research field that warrants heightened attention. While conventional approaches in dynamic optimization have predominantly focused on EDO techniques, limited studies have explored the application of RL techniques in resolving DOPs.

Among the few studies conducted, Zou et al. [51] introduced a groundbreaking algorithm known as RL-DMOEA (see Fig. 5). This reinforcement learning-based dynamic multi-objective evolutionary algorithm effectively tracks the movements of Pareto fronts over time in DMOPs. It adapts to varying degrees of environmental changes by incorporating change response mechanisms. The algorithm's effectiveness is demonstrated on CEC 2015 benchmark problems.

Additionally, in [11], the authors address dynamic time-linkage optimization problems (DTPs) using a dynamic evolutionary optimization algorithm named

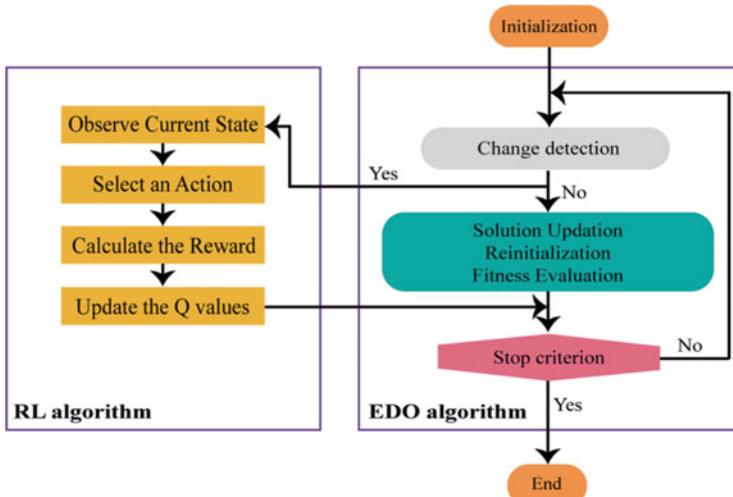


Fig. 5 Illustration of the RL-DMOEA algorithm [51]

SQLEDO. This innovative approach combines surrogate-assisted Q-learning with evolutionary optimization to handle continuous black-box DTPs. It achieves this by extracting and predicting states, employing surrogate models for evaluating Q-values, and integrating evolutionary optimization and RL techniques for long-term decision-making. The proposed algorithm outperforms comparable algorithms and demonstrates its capability to handle different dynamic changes.

Furthermore, researchers in [52] propose an innovative Q-learning RL algorithm for DOPs, drawing inspiration from EDO techniques. This algorithm draws inspiration from EDO techniques, leading to a novel perspective on defining states, actions, and reward functions. The performance of this RL model is thoroughly assessed using a customized variant of the Moving Peaks Benchmark problem, yielding compelling results that position it competitively against state-of-the-art DOPs. To provide a clearer understanding of this algorithm, we include a pseudocode representation in Algorithm 1.

Algorithm 1 Pseudo Code for the Q-learning RL Algorithm in [52]

Initialization: Set initial values for the Q-values, $Q(s, a)$, for all states and actions;
for each state s in the state space S **do**
 Generate a random solution s ;
 Evaluate the objective function $f(s, t)$;
end
repeat
 for each state s in the state space S **do**
 if $f(s, t) > f(\text{best}_\text{state}, t)$ **then**
 Update the best state: $\text{best}_\text{state} = s$;
 Update the best objective value: $f(\text{best}_\text{state}, t) = f(s, t)$;
 end
 end
 for each state s in the state space S **do**
 if $\text{rand}() < \epsilon$ **then**
 Randomly select an action a from the action space A ;
 else
 Select the action a that maximizes $Q(s, a)$: $a = \arg \max_a(Q(s, a))$;
 end
 Execute action a and observe the resulting reward R ;
 Compute the maximum Q-value for the next state: $a' = \max(Q(s, A))$;
 Calculate the TD target: $\text{TD target} = R + \gamma * Q(s, a')$;
 Update the Q-value: $Q(s, a) = Q(s, a) + \alpha * (\text{TD target} - Q(s, a))$;
 end
until Stopping criteria;

Moreover, in a separate study referenced as [53], the same algorithm is successfully applied to hyperparameter optimization for Convolutional Neural Networks (CNNs), demonstrating its promising capabilities in this context as well.

By harnessing RL's capacity to learn from the environment and make decisions based on accumulated knowledge, these studies provide valuable insights into the application of RL in the realm of DOPs. Furthermore, they demonstrate promising outcomes in comparison to traditional dynamic optimization algorithms.

4.4 *Unsupervised Learning-Based*

Numerous scientific studies have employed the power of unsupervised learning techniques to tackle the intricate challenges posed by DOPs. Among the prominent methodologies employed are Clustering, Gaussian Mixture Model (GMM), and mean shift algorithms, each offering unique insights and contributions to the field. Clustering strategies have been utilized effectively to navigate dynamic fitness landscapes with multiple peaks. For instance, in the paper by Halder et al. [55], the cluster-based dynamic differential evolution with an external archive algorithm is introduced. This algorithm incorporates adaptive clustering within a multi-population framework, enabling periodic information sharing among clusters. Similarly, studies in [56, 57] present clustering particle swarm optimizers that incorporate hierarchical clustering and nearest neighbor search strategies to locate and track peaks, complemented by fast local search methods for refinement. Furthermore, Li et al. [58] explore hierarchical clustering in dynamic optimization and introduce a random immigrants method to reduce redundancy without relying on change detection.

Another unsupervised learning technique is the GMM, a statistical model known for its prowess in representing intricate data patterns. GMMs play a pivotal role in enhancing the performance and effectiveness of solving DOPs. The study in [54] introduces MOEA/D-GMM (see Fig. 6), a prediction method that integrates GMMs into the framework to accurately track the changing Pareto set in DMOPs. Work in [59] presents GMM-DPSO, a memory-based approach that efficiently optimizes streams of recurrent problems in recurrent DOPs using GMMs.

Furthermore, Cuevas et al. [60] have made significant advancements by modifying the mean shift algorithm to effectively detect global and local optima in DOPs. The mean shift algorithm is an iterative and non-parametric process used to identify local maxima in a density function based on a set of samples. The authors enhance the algorithm by incorporating the density and fitness value of candidate solutions. This modification allows the algorithm to prioritize regions with higher fitness, which is particularly advantageous in optimization problems aiming to find the best solution.

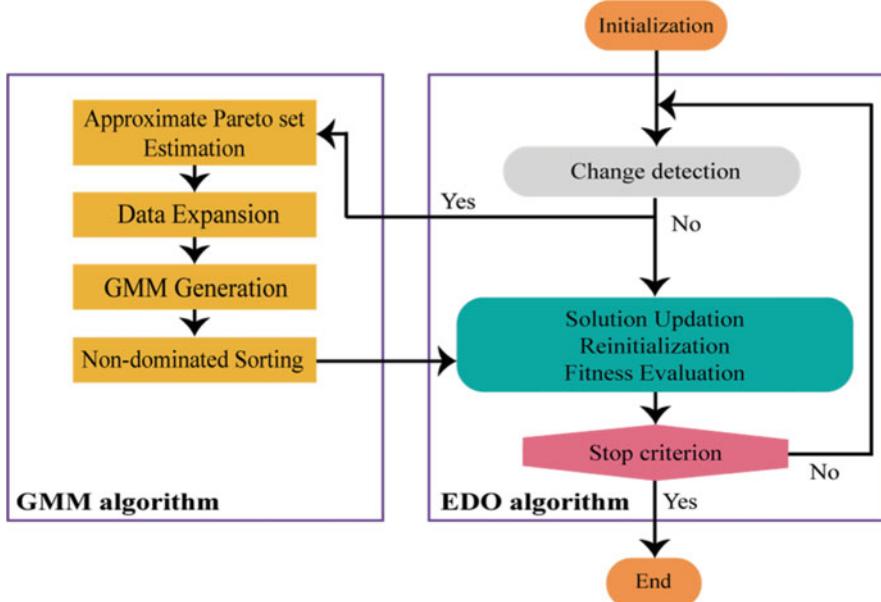


Fig. 6 Flowchart representation of the MOEA/D-GMM algorithm [54]

4.5 Other Learning-Based Models

In addition to previous research, several alternative ML paradigms have been investigated for tackling DOPs. Although not yet widely adopted, these approaches have shown significant effectiveness and promising outcomes, highlighting the need for further exploration.

In a study conducted by Zhu et al. [61], a method centered around deep multi-layer perceptrons was introduced. This method employed historical optimal solutions to generate an initial population for the algorithm when faced with novel environments. Another approach introduced an ensemble learning-based prediction strategy [62] that integrated multiple prediction models to adapt to environmental changes more effectively. An online learning-based strategy employing Passive-Aggressive Regression was also integrated into an evolutionary algorithm to predict the Pareto optimal solution set in dynamic environments [63]. Lastly, in [64], Yan et al. proposed a novel approach in their paper, which integrated decomposition-based inter-individual correlation transfer learning and dimension-wise learning. This combined method aimed to improve adaptability and expedite convergence in DMOPs.

These studies collectively demonstrate the potential of diverse ML techniques in addressing the challenges presented by dynamic environments, warranting further investigation and attention.

5 Using Evolutionary Dynamic Optimization in Machine Learning

Optimization techniques are pivotal in ML, significantly impacting performance and efficiency. By minimizing loss functions during model training, algorithms like gradient descent accelerate convergence and improve accuracy. Hyperparameter tuning methods such as grid search and Bayesian optimization find optimal parameter combinations for enhanced model performance [65]. Optimization approaches assist in feature selection, identifying relevant subsets for improved results [66]. Additionally, optimization facilitates automated model architecture search, enabling the discovery of high-performing architectures [67].

On the other hand, the utilization of EDO techniques in the realm of ML remains relatively scarce within the existing literature. However, a handful of notable works have emerged, delving into the vast potential of EDO across multiple facets of ML. These endeavours delve into the exploration and implementation of EDO methodologies in diverse ML domains, encompassing dynamic feature selection, hyperparameter tuning, model training optimization, and reinforcement learning.

The application of dynamic optimization in ML has been introduced for the first time in the literature by Boulesnane et al. [68]. The authors present a novel method that employs dynamic optimization to address the dynamic characteristics of streaming feature selection. The paper proposes an efficient approach for identifying relevant feature sets by combining the dynamic optimization algorithm WD2O and the Online Streaming Feature Selection (OSFS) algorithm within a hybrid model (see Fig. 7). The goal is to find an optimal subset of attributes that enables better classification of unclassified data, considering the evolving nature of the online feature selection problem.

In the proposed framework, the OSFS algorithm aims to retain relevant features for classification, incorporating them with existing Best Candidate Features (BCF). Ensuring non-redundancy in BCF is vital, with irrelevant attributes being removed. However, a discarded redundant attribute could become important when interacting with new attributes. To address this, the Best Redundant Candidate Feature (BRCF) set is introduced, exclusively containing redundant attributes. The EDO algorithm then determines an optimal feature sequence using the BRCF set, independently of OSFS. Eventually, the results from both OSFS and EDO are combined to establish the final set of selected attributes.

In another similar study in [69], a novel approach has been proposed to address the challenges of classifying nonstationary data streams. The intention of the authors is to combine nonstationary stream classification and EDO by adapting the Genetic Algorithm (GA) to optimize the configuration of a streaming multi-layer ensemble. The main contribution is the introduction of SMiLE (Streaming Multi-layer Ensemble), a novel classification algorithm explicitly created for nonstationary data streams. SMiLE comprises multiple layers of diverse classifiers, and the authors additionally devise an ensemble selection approach to determine the best

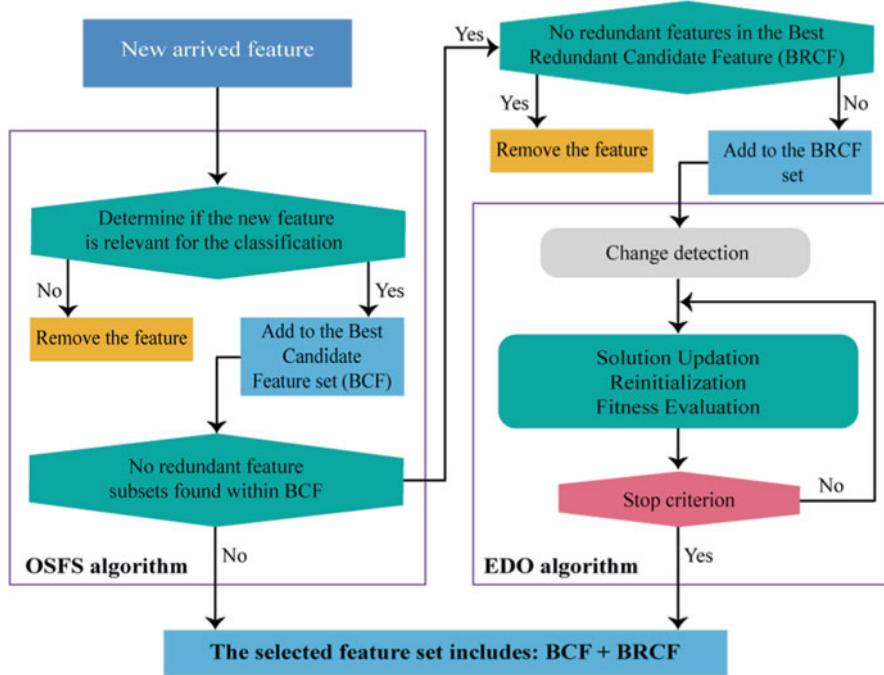


Fig. 7 Streaming feature selection using EDO

combination of classifiers for each layer in SMiLE. They formulate the selection process as a DOP and solve it using an adapted dynamic GA tailored for the streaming context.

The article [70] introduces an advanced approach known as the Arithmetic Optimization Algorithm (AOA) to enhance the training of ANNs in dynamic environments. Although metaheuristic techniques have demonstrated effectiveness in training ANNs, their underlying assumption of static environments may not accurately capture the dynamics of real-world processes. To address this limitation, the authors of this study approach the training of ANNs as a dynamic optimization issue and introduce the AOA as a potential solution for efficiently optimizing the connection weights and biases of the ANN while accounting for concept drift. This novel method specifically focuses on improving classification tasks.

Based on EDO, these studies aim to unlock new avenues of advancement and optimization within the realm of ML, paving the way for enhanced performance and greater adaptability in the face of complex and evolving challenges.

Table 2 provides a concise overview of the previously mentioned studies that employ EDO in ML tasks, highlighting both their strengths and obstacles.

Table 2 Advantages and challenges of integrating EDO methods in ML tasks

EDO technique	ML tasks	Advantages	Challenges	References
Dynamic WD2O	Streaming feature selection and classification	– Enhanced Classification by selecting the best features, improving accuracy	– Data drift problem	[68]
		– Considering feature interactions, leading to more robust models	– Resource constraints	
		– Complement online feature selection algorithms		
Dynamic GA	Streaming classification	– Enhance prediction accuracy	– Data drift problem	[69]
		– Optimize ensemble selection by dynamically choosing the best classifiers for each layer	– Model drift – Resource constraints	
Dynamic AOA	Training ANNs for classification	– Enable the training of dynamic ANNs	– Scalability challenges	[70]
		– Optimize neural network parameters in real-time	– Resource constraints	
		– Enhance classification tasks		

6 Conclusion

In this chapter, we comprehensively explore the convergence of EDO and ML techniques. EDO and ML have shown a reciprocal relationship, with each field offering valuable insights and techniques to enhance the other. We intend to inspire the evolutionary learning community to further investigate and contribute to this emerging field by highlighting the potential benefits and showcasing the promising outcomes of integrating these two domains, positing that this integration can revolutionize optimization in dynamic and complex environments.

Throughout this study, we have observed a significant interest in using ML, particularly transfer and supervised learning, in conjunction with dynamic multi-objective optimization compared to other approaches and problem types. However, the integration of EDO into the domain of ML remains somewhat limited, even though promising results have been achieved in prior studies. This implies that while there has been some exploration in this area, there remains significant untapped potential for further research to fully capitalize on their complementary strengths.

Future research in this domain should focus on developing novel EDO algorithms, refining existing ML techniques, and exploring other learning-based models to tackle the challenges of DOPs.

References

1. Jiang Y, Zhan ZH, Tan KC, Zhang J. Knowledge learning for evolutionary computation. *IEEE Trans Evol Comput.* 2023; <https://doi.org/10.1109/tevc.2023.3278132>.
2. Miikkulainen R, Forrest S. A biological perspective on evolutionary computation. *Nat Mach Intell.* 2021;3(1):9–15. <https://doi.org/10.1038/s42256-020-00278-8>.
3. Zhang J, Hui Zhan Z, Lin Y, Chen N, Jiao Gong Y, Hui Zhong J, Chung HS, Li Y, Hui Shi Y. Evolutionary computation meets machine learning: a survey. *IEEE Comput Intell Mag.* 2011;6(4):68–75. <https://doi.org/10.1109/mci.2011.942584>.
4. Qian C. Towards theoretically grounded evolutionary learning. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization. 2022. <https://doi.org/10.24963/ijcai.2022/819>.
5. Al-Sahaf H, Bi Y, Chen Q, Lensen A, Mei Y, Sun Y, Tran B, Xue B, Zhang M. A survey on evolutionary machine learning. *J R Soc N Z.* 2019;49(2):205–28. <https://doi.org/10.1080/03036758.2019.1609052>.
6. Telikani A, Tahmassebi A, Banzhaf W, Gandomi AH. Evolutionary machine learning: a survey. *ACM Comput Surv.* 2021;54(8):1–35. <https://doi.org/10.1145/3467477>.
7. Yazdani D, Cheng R, Yazdani D, Branke J, Jin Y, Yao X. A survey of evolutionary continuous dynamic optimization over two decades—part A. *IEEE Trans Evol Comput.* 2021;25(4):609–29. <https://doi.org/10.1109/tevc.2021.3060014>.
8. Wang P, Ma Y. A dynamic multiobjective evolutionary algorithm based on fine prediction strategy and nondominated solutions-guided evolution. *Appl Intell.* 2023;53:18398–419. <https://doi.org/10.1007/s10489-022-04429-9>.
9. Hamza N, Elsayed S, Sarker R, Essam D. Evolutionary constrained optimization with dynamic changes and uncertainty in the objective function. In: 2022 14th International Conference on Software, Knowledge, Information Management and Applications (SKIMA). IEEE; 2022. <https://doi.org/10.1109/skima57145.2022.10029469>.
10. Yazdani D, Yazdani D, Branke J, Omidvar MN, Gandomi AH, Yao X. Robust optimization over time by estimating robustness of promising regions. *IEEE Trans Evol Comput.* 2023;27:657–70. <https://doi.org/10.1109/tevc.2022.3180590>.
11. Zhang T, Wang H, Yuan B, Jin Y, Yao X. Surrogate-assisted evolutionary q-learning for black-box dynamic time-linkage optimization problems. *IEEE Trans Evol Comput.* 2023;27(5):1162–76. <https://doi.org/10.1109/tevc.2022.3179256>.
12. Boulesnane A, Meshoul S. Do we need change detection for dynamic optimization problems? A survey. In: Artificial intelligence and its applications. Cham: Springer International Publishing; 2022. p. 132–42. https://doi.org/10.1007/978-3-030-96311-8_13.
13. Yazdani D, Omidvar MN, Branke J, Nguyen TT, Yao X. Scaling up dynamic optimization problems: a divide-and-conquer approach. *IEEE Trans Evol Comput.* 2020;24(1):1–15. <https://doi.org/10.1109/tevc.2019.2902626>.
14. Zhou ZH. Machine learning. Singapore: Springer; 2021. <https://doi.org/10.1007/978-981-15-1967-3>.
15. Zhang Y. New advances in machine learning. Rijeka: IntechOpen; 2010. <https://doi.org/10.5772/225>.
16. Hastie T, Tibshirani R, Friedman J. Overview of supervised learning. In: The elements of statistical learning. New York: Springer; 2008. p. 9–41. https://doi.org/10.1007/978-0-387-84858-7_2.
17. Alloghani M, Al-Jumeily D, Mustafina J, Hussain A, Aljaaf AJ. A systematic review on supervised and unsupervised machine learning algorithms for data science. In: Berry M, Mohamed A, Yap B, editors. Unsupervised and semi-supervised learning. Cham: Springer International Publishing; 2019. p. 3–21. https://doi.org/10.1007/978-3-030-22475-2_1.
18. Gosavi A. Reinforcement learning: a tutorial survey and recent advances. *Informs J Comput.* 2009;21(2):178–92. <https://doi.org/10.1287/ijoc.1080.0305>.

19. Hady MFA, Schwenker F. Semi-supervised learning. In: Kacprzyk J, Jain LC, editors. Intelligent systems reference library. Berlin: Springer; 2013. p. 215–39. https://doi.org/10.1007/978-3-642-36657-4_7.
20. Weiss K, Khoshgoftaar TM, Wang D. A survey of transfer learning. *J Big Data*. 2016;3(1):9. <https://doi.org/10.1186/s40537-016-0043-6>.
21. Zhang Y, Yang Q. An overview of multi-task learning. *Natl Sci Rev*. 2017;5(1):30–43. <https://doi.org/10.1093/nsr/nwx105>.
22. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436–44. <https://doi.org/10.1038/nature14539>.
23. Balas VE, Roy SS, Sharma D, Samui P, editors. Handbook of deep learning applications. Cham: Springer International Publishing; 2019. <https://doi.org/10.1007/978-3-030-11479-4>.
24. Chelouah R, Siarry P. Optimization and machine learning: optimization for machine learning and machine learning for optimization. London: Wiley; 2022.
25. Calvet L, Jd A, Masip D, Juan AA. Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Math*. 2017;15(1):261–80. <https://doi.org/10.1515/math-2017-0029>.
26. Jiang M, Huang Z, Qiu L, Huang W, Yen GG. Transfer learning-based dynamic multiobjective optimization algorithms. *IEEE Trans Evol Comput*. 2018;22(4):501–14. <https://doi.org/10.1109/tevc.2017.2771451>.
27. Jiang M, Wang Z, Qiu L, Guo S, Gao X, Tan KC. A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning. *IEEE Trans Cybernet*. 2021;51(7):3417–28. <https://doi.org/10.1109/tcyb.2020.2989465>.
28. Zhang X, Yu G, Jin Y, Qian F. An adaptive Gaussian process based manifold transfer learning to expensive dynamic multi-objective optimization. *Neurocomputing*. 2023;538:126212. <https://doi.org/10.1016/j.neucom.2023.03.073>.
29. Jiang M, Wang Z, Guo S, Gao X, Tan KC. Individual-based transfer learning for dynamic multiobjective optimization. *IEEE Trans Cybernet*. 2021;51(10):4968–81. <https://doi.org/10.1109/tcyb.2020.3017049>.
30. Wang Z, Jiang M, Gao X, Feng L, Hu W, Tan KC. Evolutionary dynamic multi-objective optimization via regression transfer learning. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE; 2019. <https://doi.org/10.1109/ssci44817.2019.9002942>.
31. Yao F, Wang GG. Transfer learning based on clustering difference for dynamic multi-objective optimization. *Appl Sci*. 2023;13(8):4795. <https://doi.org/10.3390/app13084795>.
32. Zhang X, Yu G, Jin Y, Qian F. Elitism-based transfer learning and diversity maintenance for dynamic multi-objective optimization. *Inf Sci*. 2023;636:118927. <https://doi.org/10.1016/j.ins.2023.04.006>.
33. Fan X, Li K, Tan KC. Surrogate assisted evolutionary algorithm based on transfer learning for dynamic expensive multi-objective optimization problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2020. <https://doi.org/10.1109/cec48606.2020.9185522>.
34. Ye Y, Li L, Lin Q, Wong KC, Li J, Ming Z. Knowledge guided Bayesian classification for dynamic multi-objective optimization. *Knowl-Based Syst*. 2022;250:109173. <https://doi.org/10.1016/j.knosys.2022.109173>.
35. Li S, Yang S, Wang Y, Yue W, Qiao J. A modular neural network-based population prediction strategy for evolutionary dynamic multi-objective optimization. *Swarm Evol Comput*. 2021;62: 100829. <https://doi.org/10.1016/j.swevo.2020.100829>.
36. Cao L, Xu L, Goodman ED, Bao C, Zhu S. Evolutionary dynamic multiobjective optimization assisted by a support vector regression predictor. *IEEE Trans Evol Comput*. 2020;24(2): 305–19. <https://doi.org/10.1109/tevc.2019.2925722>.
37. Jiang M, Hu W, Qiu L, Shi M, Tan KC. Solving dynamic multi-objective optimization problems via support vector machine. In: 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI). IEEE; 2018. <https://doi.org/10.1109/icaci.2018.8377567>.

38. Zhang H, Ding J, Jiang M, Tan KC, Chai T. Inverse gaussian process modeling for evolutionary dynamic multiobjective optimization. *IEEE Trans Cybernet.* 2022;52(10):11240–53. <https://doi.org/10.1109/tcyb.2021.3070434>.
39. Han H, Liu Y, Zhang L, Liu H, Yang H, Qiao J. Knowledge reconstruction for dynamic multi-objective particle swarm optimization using fuzzy neural network. *Int J Fuzzy Syst.* 2023;25: 1853–68. <https://doi.org/10.1007/s40815-023-01477-2>.
40. Liu M, Chen D, Zhang Q, Liu Y, Zhao Y. A dynamic multi-objective evolutionary algorithm assisted by kernel ridge regression. In: Liu Y, Wang L, Zhao L, Yu Z, editors. *Advances in natural computation, fuzzy systems and knowledge discovery.* Cham: Springer International Publishing; 2022. p. 128–36. https://doi.org/10.1007/978-3-030-89698-0_14.
41. Hu W, Jiang M, Gao X, Tan KC, Ming Cheung Y. Solving dynamic multi-objective optimization problems using incremental support vector machine. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2019. <https://doi.org/10.1109/cec.2019.8790005>.
42. Xu D, Jiang M, Hu W, Li S, Pan R, Yen GG. An online prediction approach based on incremental support vector machine for dynamic multiobjective optimization. *IEEE Trans Evol Comput.* 2022;26(4):690–703. <https://doi.org/10.1109/tevc.2021.3115036>.
43. Wu X, Lin Q, Lin W, Ye Y, Zhu Q, Leung VC. A kriging model-based evolutionary algorithm with support vector machine for dynamic multimodal optimization. *Eng Appl Artif Intell.* 2023;122:106039. <https://doi.org/10.1016/j.engappai.2023.106039>.
44. Meier A, Kramer O. Prediction with recurrent neural networks in evolutionary dynamic optimization. In: Sim K, Kaufmann P, editors. *Applications of evolutionary computation.* Cham: Springer International Publishing; 2018. p. 848–63. https://doi.org/10.1007/978-3-319-77538-8_56.
45. Meier A, Kramer O. Recurrent neural network-predictions for PSO in dynamic optimization. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* ACM; 2018. <https://doi.org/10.1145/3205455.3205527>.
46. Liu XF, Zhan ZH, Zhang J. Neural network for change direction prediction in dynamic optimization. *IEEE Access.* 2018;6:72649–62. <https://doi.org/10.1109/access.2018.2881538>.
47. Meier A, Kramer O. Predictive uncertainty estimation with temporal convolutional networks for dynamic evolutionary optimization. In: Tetko I, Kůrková V, Karpov P, Theis F, editors. *Artificial neural networks and machine learning—ICANN 2019: deep learning, Lecture notes in computer science.* Cham: Springer International Publishing; 2019. p. 409–21. https://doi.org/10.1007/978-3-030-30484-3_34.
48. Liu XF, Zhan ZH, Gu TL, Kwong S, Lu Z, Duh HBL, Zhang J. Neural network-based information transfer for dynamic optimization. *IEEE Trans Neural Netw Learn Syst.* 2020;31(5):1557–70. <https://doi.org/10.1109/tnnls.2019.2920887>.
49. Shoreh MH, Aragones RH, Neumann F. Using neural networks and diversifying differential evolution for dynamic optimization. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE; 2020. <https://doi.org/10.1109/ssci47803.2020.9308154>.
50. Kalita DJ, Singh S. SVM hyper-parameters optimization using quantized multi-PSO in dynamic environment. *Soft Comput.* 2019;24(2):1225–41. <https://doi.org/10.1007/s00500-019-03957-w>.
51. Zou F, Yen GG, Tang L, Wang C. A reinforcement learning approach for dynamic multi-objective optimization. *Inf Sci.* 2021;546:815–34. <https://doi.org/10.1016/j.ins.2020.08.101>.
52. Boulesnane A, Meshoul S. Reinforcement learning for dynamic optimization problems. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* ACM; 2021. <https://doi.org/10.1145/3449726.3459543>.
53. Talaat FM, Gamel SA. RL based hyper-parameters optimization algorithm (ROA) for convolutional neural network. *J Ambient Intell Human Comput.* 2023;14:13349–59. <https://doi.org/10.1007/s12652-022-03788-y>.
54. Wang F, Liao F, Li Y, Wang H. A new prediction strategy for dynamic multi-objective optimization using gaussian mixture model. *Inf Sci.* 2021;580:331–51. <https://doi.org/10.1016/j.ins.2021.08.065>.

55. Halder U, Das S, Maity D. A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments. *IEEE Trans Cybernet.* 2013;43(3):881–97. <https://doi.org/10.1109/tsmc.2012.2217491>.
56. Li C, Yang S. A clustering particle swarm optimizer for dynamic optimization. In: 2009 IEEE Congress on Evolutionary Computation. IEEE; 2009. <https://doi.org/10.1109/cec.2009.4982979>.
57. Yang S, Li C. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Trans Evol Comput.* 2010;14(6):959–74. <https://doi.org/10.1109/tevc.2010.2046667>.
58. Li C, Yang S. A general framework of multipopulation methods with clustering in undetectable dynamic environments. *IEEE Trans Evol Comput.* 2012;16(4):556–77. <https://doi.org/10.1109/tevc.2011.2169966>.
59. Vellasques E, Sabourin R, Granger E. A dual-purpose memory approach for dynamic particle swarm optimization of recurrent problems. In: Abielmona R, Falcon R, Zincir-Heywood N, Abbass HA, editors. Recent advances in computational intelligence in defense and security. Cham: Springer International Publishing; 2015. p. 367–89. https://doi.org/10.1007/978-3-319-26450-9_14.
60. Cuevas E, Galvez J, Toski M, Avila K. Evolutionary-mean shift algorithm for dynamic multimodal function optimization. *Appl Soft Comput.* 2021;113:107880. <https://doi.org/10.1016/j.asoc.2021.107880>.
61. Zhu Z, Yang Y, Wang D, Tian X, Chen L, Sun X, Cai Y. Deep multi-layer perceptron-based evolutionary algorithm for dynamic multiobjective optimization. *Complex Intell Syst.* 2022;8(6):5249–64. <https://doi.org/10.1007/s40747-022-00745-2>.
62. Wang F, Li Y, Liao F, Yan H. An ensemble learning-based prediction strategy for dynamic multi-objective optimization. *Appl Soft Comput.* 2020;96:106592. <https://doi.org/10.1016/j.asoc.2020.106592>.
63. Liu M, Chen D, Zhang Q, Jiang L. An online machine learning-based prediction strategy for dynamic evolutionary multi-objective optimization. In: Ishibuchi H, et al., editors. Evolutionary multi-criterion optimization, Lecture notes in computer science. Cham: Springer International Publishing; 2021. p. 193–204. https://doi.org/10.1007/978-3-030-72062-9_16.
64. Yan L, Qi W, Liang J, Qu B, Yu K, Yue C, Chai X. Inter-individual correlation and dimension-based dual learning for dynamic multi-objective optimization. *IEEE Trans Evol Comput.* 2023;27(6):1780–93. <https://doi.org/10.1109/tevc.2023.3235196>.
65. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J Electron Sci Technol.* 2019;17(1):26–40.
66. Ghamisi P, Benediktsson JA. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geosci Remote Sens Lett.* 2015;12(2):309–13. <https://doi.org/10.1109/lgrs.2014.2337320>.
67. Liu Y, Sun Y, Xue B, Zhang M, Yen GG, Tan KC. A survey on evolutionary neural architecture search. *IEEE Trans Neural Netw Learn Syst.* 2023;34(2):550–70. <https://doi.org/10.1109/tnnls.2021.3100554>.
68. Boulesnane A, Meshoul S. Effective streaming evolutionary feature selection using dynamic optimization. In: Amine A, Mouhoub M, Ait Mohamed O, Djebbar B, editors. Computational intelligence and its applications. Cham: Springer International Publishing; 2018. p. 329–40. https://doi.org/10.1007/978-3-319-89743-1_29.
69. Luong AV, Nguyen TT, Liew AWC. Streaming multi-layer ensemble selection using dynamic genetic algorithm. In: 2021 Digital Image Computing: Techniques and Applications (DICTA). IEEE; 2021. <https://doi.org/10.1109/dicta52665.2021.9647220>.
70. Golcuk I, Ozsoydan FB, Durmaz ED. An improved arithmetic optimization algorithm for training feedforward neural networks under dynamic environments. *Knowl-Based Syst.* 2023;263:110274. <https://doi.org/10.1016/j.knosys.2023.110274>.

Evolutionary Techniques in Making Efficient Deep-Learning Framework: A Review



Shubham Joshi, Millie Pant, and Kusum Deep

Abstract Deep learning (DL) which is an important part of artificial intelligence has become an advanced tool for learning processes viz. classification, regression, image processing and natural language processing. While DL has demonstrated notable successes in addressing various real-world challenges in recent years, its performance can be significantly impacted by several factors, including the chosen model architecture and hyperparameters. Evolutionary algorithms can be employed at various stages of the DL framework to enhance the model performance and optimize the hyperparameters. Evolutionary algorithms and metaheuristics are the search techniques inspired by the natural phenomenon and biological evolution which are easy to use because of their non-dependency to mathematical constraints. The present study provides the brief survey of the various evolutionary algorithms employed to the DL framework at different stages to make the model efficient.

Keywords Evolutionary deep learning · Metaheuristics · Artificial intelligence

S. Joshi

Department of Applied Mathematics and Scientific Computing, IIT Roorkee, Roorkee, Uttarakhand, India

e-mail: shubham_j@amsc.iitr.ac.in

M. Pant (✉)

Department of Applied Mathematics and Scientific Computing, IIT Roorkee, Roorkee, Uttarakhand, India

Mehta Family School of Data Science and Artificial Intelligence, Roorkee, Uttarakhand, India
e-mail: pant.milli@as.iitr.ac.in

K. Deep

Mehta Family School of Data Science and Artificial Intelligence, Roorkee, Uttarakhand, India

Department of Mathematics, IIT Roorkee, Roorkee, Uttarakhand, India
e-mail: kusum.deep@ma.iitr.ac.in

1 Introduction

The field of artificial intelligence has experienced a revolution and deep learning (DL) has gained popularity in recent years due to its capability to tackle real-world problems. Unlike other machine learning techniques, deep learning commonly utilizes the Deep Neural Network (DNN) model for tackling tasks such as classification and regression. A typical deep learning framework consists of four essential components: data preprocessing, model searching, model training, and model evaluation.

However, while working with enormous datasets and intricate architectures creating effective DL frameworks continues to be difficult. In this context, there is a need for developing efficient DL frameworks that can improve training times and performance. The use of evolutionary methods has been investigated as a potential fix for this issue. Evolutionary techniques have been widely used in optimization problems for many years. These methods draw their inspiration from genetics and natural selection including genetic algorithms (GA) and biological evolution. These algorithms are recognized for their capacity to search through a large search space and find the most appropriate answers to challenging issues. Researchers have created more effective DL frameworks by utilizing the evolutionary methods to find the best network architectures and optimize hyperparameters.

The present study aims to review the role of evolutionary techniques in making efficient DL framework. It is important to note that a number of other authors have recently carried out comparable reviews. As an illustration Liu et al. [1] carried out a review in 2021, with focus on evolutionary computation for neural architecture search. Zhou et al. [2] wrote a review in 2021 with a focus on evolutionary computation for the construction of DNN and Zhou et al. [3] included this subject in their review with focus on evolutionary computation for advances of evolutionary neural architecture search, which was also published in 2021. Zhan et al. [4] conducted a review in the year 2022 for the evolutionary computation in whole DL procedures. Despite their comparable scopes each of these studies presents a distinctive viewpoint on the subject and offers insightful information about the topic.

The objective of this chapter is to provide a comprehensive overview of the latest research conducted in the field of evolutionary DL from 2018 to 2023. The time frame of selected papers was chosen to ensure the high-quality studies and emerging trends, methodologies and technologies that have gained prominence in the last 5 years. Specifically, we will focus on four key areas, which include data preprocessing techniques like missing value imputation and data generation, model architecture search for finding optimal neural network structures and hyperparameters, model training to identify the most effective weight parameters, and model evaluation including pruning and ensemble of different models.

The present study is divided into five Parts. Apart from the introduction the remaining chapter is formatted as follows. Section 2 presents a theoretical background. Section 3 presents a detailed literature survey of the various evolutionary techniques used to make a more efficient DL framework. In Sect. 4 we summarize

the key findings and in Sect. 5 we highlight potential future directions for research in this area.

2 Theoretical Background

In this section, we'll explore the basic ideas behind two important concepts that are really important in today's computer intelligence world: DL models and evolutionary algorithms. Even though they work in different ways, both of these ideas play a big role in making computers smarter and better at solving problems.

2.1 Deep Learning Models

DL which is a subset of machine learning is simply a neural network with more than two layers also known as DNN. DL models consist of multiple layers of interconnected neurons which enable them to automatically extract useful features from raw data and learn complex patterns and relationships between inputs and outputs. Deep belief network (DBN), Convolutional neural network (CNN), recurrent neural network (RNN), stacked auto-encoder (SAE), and generative adversarial network (GAN) are the five general categories that can be used to classify widely-used DL models. A CNN uses a process of convolution, pooling, and activation to automatically extract relevant features from images and classify them into different categories. In a DBN each layer of hidden units is trained to reconstruct the input data at that level. DBN is used for supervised learning problems like classification and regression as well as unsupervised learning activities like feature extraction and dimensionality reduction.

A SAE is made up of many layers of autoencoders. It is employed for dimensionality reduction and unsupervised feature learning. The network is trained layer by layer via backpropagation with the output of each layer serving as the input to the following layer. A RNN uses feedback connections to handle sequential data and long-term dependencies as well as input sequence context can be captured by it. RNNs are frequently employed in applications where the input data is a sequence, such as speech recognition and natural language processing. A Generative Adversarial Network (GAN) comprises two distinct models: a generator and a discriminator. The generator is responsible for acquiring the ability to generate new data resembling the training data, while the discriminator is tasked with learning to differentiate between real and counterfeit data. In a competitive training procedure the two models are put together to provide high-quality synthetic data that is comparable to the real data. Figure 1 represent different types of DL models.

Since the network structure of DL greatly affects its performance, many academics have worked hard to create cutting-edge DL structures including Lenet, AlexNet, VGG, ResNet, and DenseNet, among others.

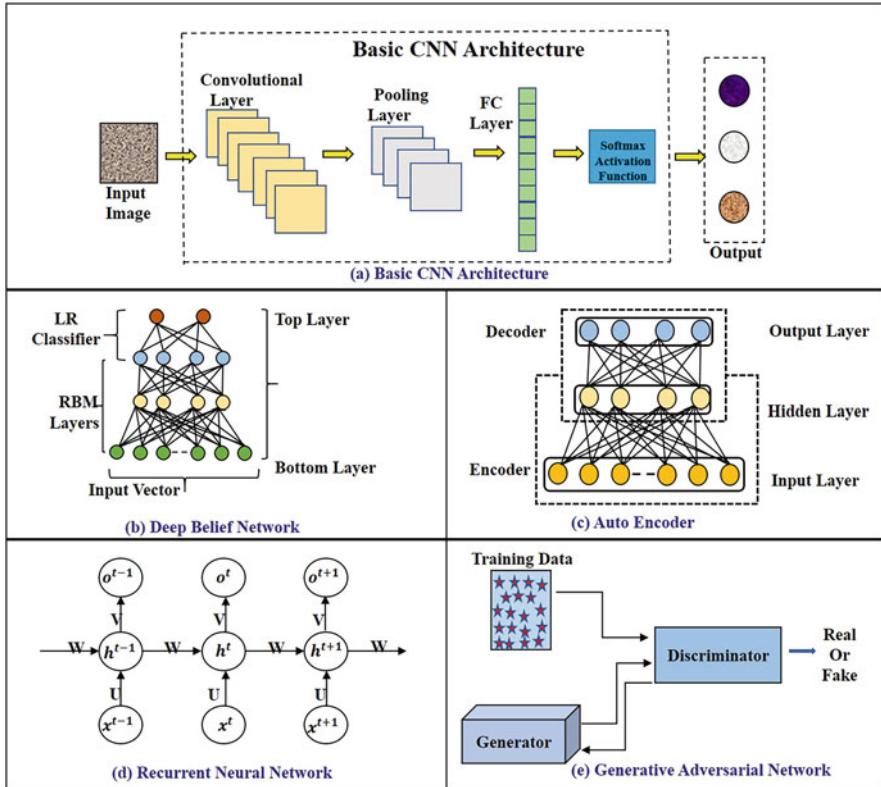


Fig. 1 Types of DL models. (a) Basic CNN architecture. (b) Deep belief network. (c) Auto encoder. (d) Recurrent neural network. (e) Generative adversarial network

2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) maintain a population and determines each individual's fitness using the theory of biological evolution. By crossover and mutation, the fittest species endures and passes on its offspring. The population fits the environment well after several generations. EAs are the methodologies used to tackle optimization issues. DL models include finding an optimal neural architecture search, finding optimum hyperparameters which can be considered to be as an optimization problem.

2.3 Taxonomy

Data, model choice, model parameters, and evaluation methodology should all be taken into account while aiming to maximize DL performance. Therefore data processing, model search, model training, and model evaluation and utilization are the four main methods used in DL model to solve learning challenges. By improving the outcomes of one or more of these methods better DL can be obtained. Furthermore Fig. 2 illustrates the four major topics where evolutionary techniques can be applied for improving the results of the four procedures of DL.

3 Literature Survey

In this section a selected literature review of the use of evolutionary computing in the different frameworks of DL has performed.

3.1 Data Preprocessing

A typical first step in DL workflow is preprocessing data to ensure that it is in a format that the network can understand. Before making any DL model on our dataset we need to ensure that the data provided is clean, balanced and complete. Missing value imputation, data generation and processing the imbalanced data are the main parts of data preprocessing. In the recent years it has been seen that evolutionary techniques can be used to fill the missing values in incomplete dataset, generate better data, achieve better data balance and in the task specific data preprocessing to meet the requirements of the DL model.

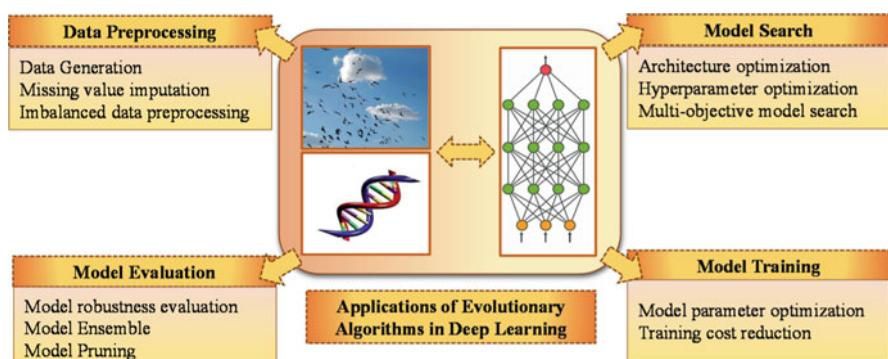


Fig. 2 Applications of evolutionary algorithms in DL

Wang et al. [5] proposed a novel evolutionary-based GAN framework called 44E-GAN. The proposed technique viewed discriminators as an environment and producers as an evolutionary population. The approach enables the algorithm to incorporate the strengths of various adversarial objectives, resulting in the production of a highly competitive solution. Jhang et al. [6] have proposed Evolutionary Cost Sensitive Deep Belief Network (ECS-DBN) for classification problem in imbalance data. The suggested approach integrates a cost-sensitive function directly into the classification framework, aiming to enhance the optimization of misclassification costs using adaptive differential evolution. This novel method has been tested on a comprehensive set of 58 benchmark datasets, in addition to a real-world dataset. Awawdeh et al. [7] have considered the missing data problem and proposed a GA based imputation technique EvoImputer and simultaneously perform the feature selection to enhance the model performance. The proposed algorithm was tested on ten different benchmark datasets and compared the performance with classical imputation techniques. The findings demonstrated that using the suggested methodology produced improved results in terms of precision, sensitivity, specificity, G-mean, and area under curve. Liu et al. [8] have proposed an evolutionary technique assisted GAN, EvoGAN to generate facial expressions. Lin et al. [9] proposed a GA based technique evolutionary architecture search GAN(EAS-GAN) to automatically search the architecture of a GAN. The architecture searched by the proposed method was compared with other state-of-the-art GAN and found to outperform others. In order to impute many missing observations in multivariate data, Garcia et al. [10] suggested a GA that minimises a novel multi-objective (fitness) function based on the Minkowski distance which considers the means, variances, covariances, and skewness between available or completed data. Table 1 represents the literature survey of Evolutionary techniques used in Data Preprocessing.

3.2 Model Search

After the data has been preprocessed the next important step in DL procedure is the selection of model. There may not be sufficient past information and experience to direct the construction and selection of DL models when tackling learning problems. Therefore, there is a need of the method which helps in the automatic selection of the suitable DL model. It is possible to think of choosing an appropriate model as an optimization problem. In the recent years many researchers have attempted to find the suitable DL models with the help of evolutionary techniques. Martin et al. [11] proposed novel EA, EvoDeep to find the best parameters and architecture by evolution in a DNN. Each individual in this method represents a certain network architecture, and the fitness value of each individual is determined by its classification accuracy. The MNIST dataset is used to test the suggested algorithm's performance.

Table 1 Literature survey of EAs for data preprocessing

Year	Objective	Approach	Evolutionary technique	References
2019	Generation of synthetic data	Development of Evolutionary-GAN (E-GAN)	EA	[5]
2019	Imbalanced data classification	Evolutionary cost sensitive deep belief network (ECS_DBN)	Adaptive differential evolution	[6]
2022	Missing value imputation and feature extraction	EvoImputer, an evolutionary approach for optimizing the best subset of incomplete feature	GA	[7]
2022	Generation of human facial images using a pretrained GAN	Combination of EA and GAN (EvoGAN) to generate face images with compound expression	EvoGAN	[8]
2022	Automation of entire design process of GAN	Development of Evolutionary architecture search (EAS)	GA	[9]
2023	Missing value imputation in multi variate data	GA based minimization based on Minkowski distances of means, variance and covariance	Multiple Imputation GA (MIGA)	[10]

Wen et al. [12] proposed an encoding strategy to map fixed length encoding variable to DL structure and used GA to find optimal DL architecture for image classification. Xie et al. [13] used the genetic encoding strategy and environmental selection for finding the optimal CNN architecture where the fitness of each possible individual architecture is calculated based on classification accuracies. Shi et al. [14] have proposed a GA based approach Genetic-GNN to find the optimal architecture of a GNN by optimizing both GNN structure and the hyperparameters. Mahdaddi et al. [15] have proposed a hybrid DL model for predicting the drug target binding affinities and to find the best hyperparameters they have used Differential Evolution algorithm. Samaranda Belciug [16] proposed a potential method based on differential evolution to automatically learn a deep neural architecture. The authors have created a set of potential CNN architectures, represented by a fixed-length integer array, and utilized Differential Evolution to identify the best-suited hyperparameters for these structures. The proposed model has been tested on three cancer datasets and found to outperform benchmark DL models. Rajashree et al. [17] have proposed a hybrid DL framework for privacy preservation in edge computing. In this work authors have used evolutionary algorithm to search and provide an optimized auto encoder. Qu et al. [18] have proposed a two staged coevolution method for deep CNN. In the first stage hyperparameters were self evolved and in the second stage the hyperparameters were evolved through the combination of particle swarm optimization and GA. Table 2 represents the selected literature survey of EAs used in model search.

Table 2 Literature survey of EAs for model search

Year	Objective	Approach	Hyperparameters optimized	Evolutionary technique	References
2018	Evolution of hyperparameters & architecture	EvoDeep, an evolutionary algorithm to find best parameters by evolution	Optimizer, Batch size, Activation function, Pooling size	EA	[11]
2022	Neural network architecture search for image classification	New encoding strategy maps fixed-length encoding to variable depth DL structure using RepVGG nodes. GA used to find optimal individual and corresponding DL model	–	For the purpose of finding the optimal individual and its related DL model, the GA is used.	[12]
2022	Automated design of CNN architecture	Building blocks based on CNN with a triplet attention mechanism, with evolutionary operators crafted to explore the search space.	–	Genetic encoding strategy with environmental selection	[13]
2022	Neural Architecture search (NAS), Hyperparameter optimization	Proposes a Genetic Graph Neural Network (Genetic-GNN) NAS framework to evolve the GNN architectures and optimize hyperparameters	Dropout rate, Weight decay rate, Learning rate.	GA	[14]
2022	Hyperparameter optimization in DL models for effective drug-target interaction prediction	CNN-AbiLSTM	Number of filters, Filter sizes, Number of FC layer, Dropout rate of FC layer	Differential evolution based hyperparameter optimization framework	[15]
2022	CNN architecture search	Establish a population of candidate CNN structures that can be represented in a fixed-length integer array	Number of layers, units, height and length of filters	Differential evolution	[16]

2023	Develop privacy preserving framework to protect data shared in IOT	Used evolutionary algorithm to search and provide an optimized auto encoder	–	GA	[17]
2023	Hyperparameter optimization	Two-stage coevolution method (TSC)	Loss function, activation function, optimization function	GA	[18]

3.3 Model Training

Model training involves solving an optimization problem to identify the most suitable global model parameters using training data. The objective of model parameter optimization is to discover the best values for parameters such as connection weights and biases within a pre-established model architecture. This optimization process entails modifying a set of decision variables that symbolize the model's parameters, all with the aim of enhancing its performance for a particular task. Numerous research in EA have been carried out to optimize model parameters during the training phase, demonstrating it as an efficient and economical method for optimizing complex problems.

Harris et al. [19] used an EA with encoding based on direct acyclic graph to optimize the CNN. The proposed algorithm is applied on 200 randomly generated CNN architectures and sampled only 10% of the training data which reduced the time required for training. Mariya et al. [20] have optimized DNN using a multiobjective covariance matrix adaptive EA. The tournament selection method is used to extract a subset of hypothesis from a total number of genes. The model is simultaneously trained using cloud computing resources and distributed technology. The proposed algorithm is tested in Kaldi speech recognition test. Gong et al. [21] have proposed a hybrid backpropagation and cooperative coevolution framework which decomposes a large scale problem to smaller subcomponents and then these subcomponents are evolved separately using evolutionary algorithms. Authors have used this technique to optimize the weight parameters of the DL model. Li et al. [22] proposed Evolutionary distribution algorithm, SHEDA that can be used to optimize hyperparameters in the mixed variable problem.

In this study encoding of the hyperparameters have been used by the mixed variable encoding scheme. The proposed method have been found effective in computational cost reduction. Sun et al. [23] has improvised evolutionary DL by improving its fitness by employing random forest as the fitness predictor. The authors found that the proposed algorithm is efficient in reducing the computational cost.

Table 3 represents the selected literature survey of Evolutionary techniques used in model Training.

3.4 Model Evaluation and Utilization

Evaluating and effectively deploying deep learning models is essential for their performance, as a model can excel in certain situations but underperform in others. To enhance the robustness of a deep learning model, techniques like ensemble learning and model pruning are employed. Ensemble learning involves combining multiple models, which can result in improved performance, such as higher accuracy, greater resilience, and reduced uncertainty, compared to relying on a single

Table 3 Literature survey of EAs for model training

Year	Objective	Approach	Parameters optimized	Evolutionary technique	References
2019	Training time reduction, optimizing CNN	Evolutionary algorithm with graph based encoding strategy	–	EA	[19]
2019	Tuning of meta parameters of speech recognition system	Evolutionary algorithm with a multi objective pareto optimization	–	GA	[20]
2021	Model Parameter optimization, computational cost reduction	Cooperative coevolution with backpropagation	Weight parameters	Differential Evolution	[21]
2021	Hyperparameter optimization, computational cost reduction	Surrogate-assisted multi-level evaluation method	Number of Kernels and Neurons, Kernel Size, Activation function	Estimation of distribution algorithm (EDA)	[22]
2020	Computational cost reduction	To speed up the fitness assessment in EDL, a random forest-based end-to-end offline performance predictor is developed	–	EA	[23]

model. Because of this, some EDL methods suggested using evolutionary computation to enhance the robustness of DL models.

Su et al. [24] proposed a black box DNN model in case of attack scenario to check the robustness of model where only one pixel is perturbated using Differential Evolution. Perturbation used in this study is a five element x-y coordinate and RGB value of a perturbation. In their work, Gong et al. [25] introduced an efficient niching algorithm based on neighborhood principles for learning multimodal parameters within deep learning models. This algorithm leverages a neighborhood strategy and a diversity-preserving operator to facilitate multimodal optimization. The proposed algorithm used local Gaussian reproduction in neighborhood to locate multiple optima. This algorithm contributed to the simultaneous optimization of parameters across multiple networks. The efficacy of this method was assessed through evaluations conducted on the CIFAR-10 and Imagenet datasets. Zhou et al. [26] introduced an evolutionary approach known as ECDNN, designed to effectively prune and compress deep neural networks specifically for biomedical image segmentation tasks. Convolutional filters from layers used in feature map concatenation are intended to be removed by the pruning operator. Vidnerova et al. [27] proposed a method to check the vulnerability of different classifiers to adversarial

examples generated by the evolutionary algorithms without access to model weights. In their study authors found that the majority of classifiers are somewhat vulnerable to adversarial samples. Salehinejad et al. [28] have Proposed a Differential Evolution approach to reduce various Deep CNN's fully connected and convolutional phases. In their study authors have proposed an algorithm EDropout to select the best pruning state that can reduce the number of parameters without much affecting the classification accuracy. Cruz et al. [29] proposed a CNN ensemble method based on evolutionary algorithm for the image classification. The proposed technique is implemented in a case study that focuses on identifying metal sheet misalignment prior to joining them using submerged arc welding. Poyatos et al. [30] reduced the complexity of a transfer learning based DNN by an evolutionary pruning method, EvoPruneDeepTL. In their work authors have replaced the fully connected layer by the sparse layers optimized by a GA. The proposed method was successful in reducing the network architecture while maintaining the accuracy. Pietroñ et al. [31] have taken the advantage of efficient unstructured pruning and bit width reduction to speedup DL models on GPU using the evolutionary algorithms. Table 4 represents the selected literature survey of evolutionary techniques used in ensemble, pruning and to increase model robustness.

4 Results and Discussion

Evolutionary algorithms aims to construct the DL models to obtain the high accuracy and less computational time by improving the four phases of any DL framework. It has been seen that Evolutionary algorithms have been applied in designing all kind of networks including DBN, RNN, SAE, CNN etc. Particularly in this study we have seen the role of EAs in four procedures of DL, as illustrated in Fig. 3.

From the papers reviewed it has been observed that in the current years researchers have given a lot of attention in the use of evolutionary algorithms in the data preprocessing for the missing value imputations, data balancing and Data generations. Generative adversarial networks (GAN) have been assisted by the Evolutionary algorithms for the new data generation. In recent years Evolutionary algorithms were applied to automatically search the best neural network architecture and the optimum hyperparameters which significantly improve the accuracy of any DL model and reduces the computational cost and time. The most frequent hyperparameters optimized were number of layers, filters and number of kernels. GA is the most commonly used evolutionary technique in improving the DL performance. The fundamental building blocks of DL models are the weight parameters. They are essential in avoiding overfitting since they enable the model to learn from the data and produce precise predictions. In the recent years researchers have applied evolutionary algorithms such as Cooperative coevolution, Cooperative dual-evolution for optimizing the weight parameters in the DL models. Model pruning is the process of removing extra weights, connections, or neurons from a DL model in order to reduce its size and complexity. In the recent years researchers have used

Table 4 Literature survey of EC for model evaluation and utilization

Year	Objective	Approach	Model targeted	Evolutionary technique	References
2019	Low cost adversarial attack in DL models in limited scenario	Robustness evaluation by modifying only one pixel by differential evolution	DNN	Differential Evolution	[24]
2019	Learning multi-modal parameters	Neighbourhood based niching algorithm	DNN	Differential Evolution	[25]
2020	Model utilization by pruning	Proposed Evolutionary Compression of Deep Neural Networks (ECDNN)	CNN	EA	[26]
2020	Adversarial sample optimization to evaluate model robustness and vulnerability	Proposed an evolutionary method that, in the case of a black-box attack, may produce adversarial samples for any machine-learning model	CNN	GA	[27]
2021	Model pruning	Proposed a Differential Evolution approach to reduce various Deep CNN's fully connected and convolutional phases	CNN	Differential Evolution	[28]
2021	Ensemble of CNN using Evolutionary algorithm	Proposed method for producing a close to ideal ensemble of CNNs using an effective search technique built on an evolutionary algorithm	CNN	EA	[29]
2022	Model pruning for transfer learning based networks	Proposed an evolutionary pruning model EvoPruneDeepTL which substitutes sparse layers that have been genetically optimised for performance for the final fully connected layers	CNN	GA	[30]
2023	Efficient unstructured pruning and quantization of neural network	proposed evolutionary technique ensures that weights left unremoved retain the same values they possessed after the previous training phase	CNN	GA	[31]

evolutionary algorithms such as differential evolution and GA to automatically prune the model, to find out the best connection which can be removed to make the model performance better.

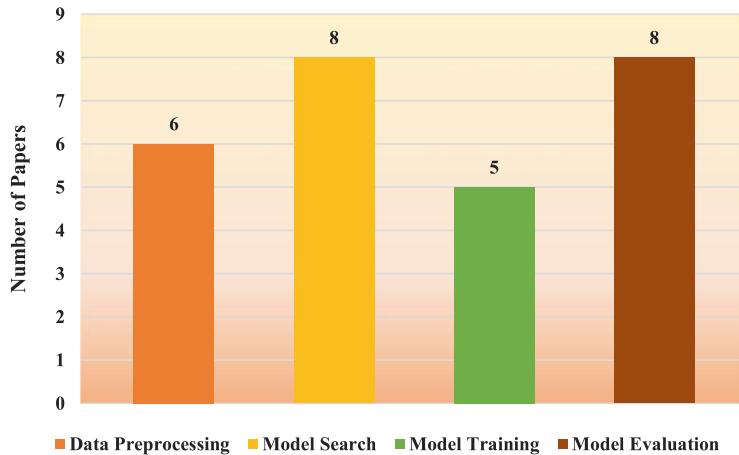


Fig. 3 Category wise distribution of papers reviewed

Table 5 Datasets used in the reviewed papers

Dataset	Description	References
CIFAR 10	The CIFAR-10 dataset is composed of 10 classes, each containing 6000 images, resulting in a total of 60,000 32×32 color images	[9, 12, 13, 21–23, 31]
CIFAR 100	The CIFAR-100 dataset comprises 100 classes, with each class containing 600 images, resulting in a total of 60,000 32×32 color images	[12, 13, 21–23, 31]
MNIST	The MNIST dataset consists of handwritten digits, comprising a total of 60,000 training images and 10,000 testing images	[17, 21, 27]
Miscellaneous (NEU-CLS, Iris, STL-10, Kiba, Davis, CATARACT, Lung cancer, colon cancer etc.)	Surface detection dataset, Flower images, image recognition datasets, Drug interaction dataset, medical images etc.	[7, 9, 10, 14–16, 26, 28]

Table 5 shows the different datasets used in the research papers which are reviewed in this chapter. The benchmark datasets chosen in most of the studies are CIFAR10 and CIFAR100. The fundamental reason why these benchmark datasets were chosen is that the most recent CNN architecture design techniques utilise them extensively. Even while evolutionary approaches have made great progress in recent years, the large compute resource required remains a challenge. Therefore, when considering evolutionary strategies in the present, researchers should balance off the time cost and performance of the designed model architecture.

5 Conclusion and Future Scope

This book chapter provides an overview of advances in DL by the use of the Evolutionary techniques. This chapter has examined how evolutionary strategies can be used to create effective DL frameworks. The chapter has covered the optimisation of deep neural networks using evolutionary algorithms such as GAs, and differential evolution. The chapter has illustrated the value of these strategies in enhancing the precision, speed, and generalisation capacities of DL models through a thorough literature analysis. Further study in this area has a lot of potential. As we look to the future evolutionary approaches will play a bigger role in model optimisation as DL's popularity and complexity continue to rise. Even though evolutionary techniques have been applied to construct efficient DL frameworks in different procedures, there are still open problems such as the use of evolutionary techniques in weight parameter optimization by quantization and weight clustering. The potential contributions of evolutionary techniques within the domain of Explainable Artificial Intelligence have not yet been thoroughly investigated and their exploration in the context of Explainable Artificial Intelligence will be a subject of interest in future research endeavors.

References

1. Liu Y, Sun Y, Xue B, Zhang M, Yen GG, Tan KC. A Survey on Evolutionary Neural Architecture Search, in IEEE Transactions on Neural Networks and Learning Systems. 2023;34(5):550–570. <https://doi.org/10.1109/TNNLS.2021.3100554>.
2. Zhou X, Qin AK, Gong M, Tan KC. A survey on evolutionary construction of deep neural networks. IEEE Trans Evol Comput. 2021;25(5):894–912. <https://doi.org/10.1109/TEVC.2021.3079985>.
3. Zhou X, Qin AK, Sun Y, Tan KC. A survey of advances in evolutionary neural architecture search. In: 2021 IEEE Congress on Evolutionary Computation (CEC)—2021 Proceedings. 2021. p. 950–57. <https://doi.org/10.1109/CEC45853.2021.9504890>.
4. Zhan ZH, Li JY, Zhang J. Evolutionary deep learning: a survey. Neurocomputing. 2022;483: 42–58. <https://doi.org/10.1016/j.neucom.2022.01.099>.
5. Wang C, Xu C, Yao X, Tao D. Evolutionary generative adversarial networks. IEEE Trans Evol Comput. 2019;23(6):921–34. <https://doi.org/10.1109/TEVC.2019.2895748>.
6. Zhang C, Tan KC, Li H, Hong GS. A cost-sensitive deep belief network for imbalanced classification. IEEE Trans Neural Netw Learn Syst. 2019;30(1):109–22. <https://doi.org/10.1109/TNNLS.2018.2832648>.
7. Awaddeh S, Faris H, Hiary H. EvoImputer: an evolutionary approach for missing data imputation and feature selection in the context of supervised learning. Knowl-Based Syst. 2022;236:107734. <https://doi.org/10.1016/j.knosys.2021.107734>.
8. Liu F, et al. EvoGAN: an evolutionary computation assisted GAN. Neurocomputing. 2022;469: 81–90. <https://doi.org/10.1016/j.neucom.2021.10.060>.
9. Lin Q, Fang Z, Chen Y, Tan KC, Li Y. Evolutionary architectural search for generative adversarial networks. IEEE Trans Emerg Top Comput Intell. 2022;6(4):783–94. <https://doi.org/10.1142/S0129065723500260>.

10. Figueroa-García JC, Neruda R, Hernandez-Pérez G. A genetic algorithm for multivariate missing data imputation. *Inf Sci (NY)*. 2023;619:947–67. <https://doi.org/10.1016/j.ins.2022.11.037>.
11. Martín A, Lara-Cabrera R, Fuentes-Hurtado F, Naranjo V, Camacho D. EvoDeep: a new evolutionary approach for automatic deep neural networks parametrisation. *J Parallel Distrib Comput*. 2018;117:180–91. <https://doi.org/10.1016/j.jpdc.2017.09.006>.
12. Wen L, Gao L, Li X, Li H. A new genetic algorithm based evolutionary neural architecture search for image classification. *Swarm Evol Comput*. 2022;75:101191. <https://doi.org/10.1016/j.swevo.2022.101191>.
13. Xie Y, Chen H, Ma Y, Xu Y. Automated design of CNN architecture based on efficient evolutionary search. *Neurocomputing*. 2022;491:160–71. <https://doi.org/10.1016/j.neucom.2022.03.046>.
14. Shi M, et al. Genetic-GNN: evolutionary architecture search for graph neural networks. *Knowl-Based Syst*. 2022;247:108752. <https://doi.org/10.1016/j.knosys.2022.108752>.
15. Mahdaddi A, Meshoul S, Belguidoum M. EA-based hyperparameter optimization of hybrid deep learning models for effective drug-target interactions prediction. *Expert Syst Appl*. 2021;185:115525. <https://doi.org/10.1016/j.eswa.2021.115525>.
16. Belciug S. Learning deep neural networks' architectures using differential evolution. Case study: medical imaging processing. *Comput Biol Med*. 2022;146:105623. <https://doi.org/10.1016/j.combiomed.2022.105623>.
17. Harine Rajashree R, Sundarakantham K, Sivasankar E, Mercy Shalinie S. A hybrid deep learning framework for privacy preservation in edge computing. *Comput Secur*. 2023;129:103209. <https://doi.org/10.1016/j.cose.2023.103209>.
18. Qu Y, Ma Y, Ming X, Wang Y, Cheng S, Chu X. Two-stage coevolution method for deep CNN: a case study in smart manufacturing. *Appl Soft Comput*. 2023;135:110026. <https://doi.org/10.1016/j.asoc.2023.110026>.
19. Irwin-Harris W, Sun Y, Xue B, Zhang M. A graph-based encoding for evolutionary convolutional neural network architecture design. In: 2019 IEEE Congress on Evolutionary Computation (CEC)—2019 Proceedings. 2019. p. 546–53. <https://doi.org/10.1109/CEC.2019.8790093>.
20. Moriya T, Tanaka T, Shinozaki T, Watanabe S, Duh K. Evolution-strategy-based automation of system development for high-performance speech recognition. *IEEE/ACM Trans Audio Speech Lang Process*. 2019;27(1):77–88. <https://doi.org/10.1109/TASLP.2018.2871755>.
21. Gong M, Liu J, Qin AK, Zhao K, Tan KC. Evolving deep neural networks via cooperative coevolution with backpropagation. *IEEE Trans Neural Netw Learn Syst*. 2021;32(1):420–34. <https://doi.org/10.1109/TNNLS.2020.2978857>.
22. Li JY, Zhan ZH, Xu J, Kwong S, Zhang J. Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks. *IEEE Trans Neural Netw Learn Syst*. 2021;34:2338–52. <https://doi.org/10.1109/TNNLS.2021.3106399>.
23. Sun Y, Wang H, Xue B, Jin Y, Yen GG, Zhang M. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Trans Evol Comput*. 2020;24(2):350–64. <https://doi.org/10.1109/TEVC.2019.2924461>.
24. Su J, Vargas DV, Sakurai K. One pixel attack for fooling deep neural networks. *IEEE Trans Evol Comput*. 2019;23(5):828–41. <https://doi.org/10.1109/TEVC.2019.2890858>.
25. Gong YJ, Zhang J, Zhou Y. Learning multimodal parameters: a bare-bones niching differential evolution approach. *IEEE Trans Neural Netw Learn Syst*. 2018;29(7):2944–59. <https://doi.org/10.1109/TNNLS.2017.2708712>.
26. Zhou Y, Yen GG, Yi Z. Evolutionary compression of deep neural networks for biomedical image segmentation. *IEEE Trans Neural Netw Learn Syst*. 2020;31(8):2916–29. <https://doi.org/10.1109/TNNLS.2019.2933879>.
27. Vidnerová P, Neruda R. Vulnerability of classifiers to evolutionary generated adversarial examples. *Neural Netw*. 2020;127:168–81. <https://doi.org/10.1016/j.neunet.2020.04.015>.

28. Salehinejad H, Valae S. EDropout: energy-based dropout and pruning of deep neural networks. *IEEE Trans Neural Netw Learn Syst.* 2022;33(10):5279–92. <https://doi.org/10.1109/TNNLS.2021.3069970>.
29. Cruz YJ, Rivas M, Quiza R, Villalonga A, Haber RE, Beruvides G. Ensemble of convolutional neural networks based on an evolutionary algorithm applied to an industrial welding process. *Comput Ind.* 2021;133:103530. <https://doi.org/10.1016/j.compind.2021.103530>.
30. Poyatos J, Molina D, Martinez AD, Del Ser J, Herrera F. EvoPruneDeepTL: an evolutionary pruning model for transfer learning based deep neural networks. *Neural Netw.* 2023;158:59–82. <https://doi.org/10.1016/j.neunet.2022.10.011>.
31. Pietroń M, Żurek D, Śnieżyński B. Speedup deep learning models on GPU by taking advantage of efficient unstructured pruning and bit-width reduction. *J Comput Sci.* 2023;67:101971. <https://doi.org/10.1016/j.jocs.2023.101971>.

Integrating Particle Swarm Optimization with Reinforcement Learning: A Promising Approach to Optimization



Arindam Ghosh, Ojaswita Tiwari, Krishna Pratap Singh,
and Muneendra Ojha

Abstract The popular optimization method, Particle Swarm Optimization (PSO), drew inspiration from the collective behavior of fish schools and bird flocks, leading to its development. PSO has some drawbacks as it exhibits slow convergence for complex optimization problems and tends to converge prematurely in local optima. PSO was primarily designed for single-objective optimization problems. Nevertheless, numerous real-life challenges encompass multiple objectives. In contrast, reinforcement learning (RL) is a semi-supervised learning framework where an intelligent agent interacts with its environment, making rational decisions to maximize the associated reward. RL, however, can be computationally intensive. In this chapter, we investigate the recent research work to overcome the aforementioned issues by integrating RL with PSO. This approach has garnered substantial attention in recent years. In this approach, PSO is used to explore the search space and find potential solutions, while RL is used to refine and optimize the solutions found by PSO. We also discuss the methodology used in various research works in hybrid RL and identify directions for future research. Our analysis underscores RL's potential as a valuable instrument for enhancing the efficiency and effectiveness of PSO algorithms.

Keywords Particle swarm optimization · Reinforcement learning · Swarm intelligence · Multi-agent systems · Metaheuristics · Policy optimization

A. Ghosh (✉) · K. P. Singh · M. Ojha

Department of Information Technology, Indian Institute of Information Technology Allahabad,
Prayagraj, Uttar Pradesh, India

e-mail: rsi2021001@iiita.ac.in; kpsingh@iiita.ac.in; muneendra@iiita.ac.in

O. Tiwari

Mahindra and Mahindra, Mumbai, Maharashtra, India

e-mail: tiwari.ojaswita1@mahindra.com

1 Introduction

Many machine learning algorithms are inspired by nature. The Genetic Algorithm, for instance, is inspired by the natural evaluation process that occurs in living organisms. Reinforcement learning is modeled on behavioral psychology, whereas the swarm algorithm takes cues from wild swarms of animals, such as bird flocks and fish schools [1]. One of the ways to learn the policy through machine learning is by using Reinforcement Learning. This technique involves interacting with the environment to understand the agent's behavior, which ultimately helps the agent to achieve its goal. Maximizing the expected cumulative benefit is the agent's objective. A policy, or a map that creates an action for any given state, is the solution to an RL problem [2].

Particle Swarm Optimization (PSO) and Reinforcement Learning (RL) are popular optimization techniques in various fields, such as engineering, finance, and robotics. PSO is an algorithm that generates a collection of prospective solutions. It performs stochastic optimization that uses this collection of prospective solutions to find the optimal solution. The combination of PSO and RL has recently gained attention as a promising approach to optimization. This integration enables the optimization algorithm to leverage the advantages of both techniques, such as PSO's global search capability and RL's ability to learn from experience. The population-based approaches are used in optimization problems, such as genetic algorithms and particle swarm optimization, to quickly find optimal global solutions for multi-modal functions with broad solution spaces. This approach can be applied to reinforcement learning algorithms to help them find ideal policies rapidly. The resulting learning process is known as multi-agent reinforcement learning or Swarm Reinforcement Learning, which uses PSO to enhance learning in an optimized way. In this process, each particle is considered an agent, and each agent perceives the information of other agents as a part of the environment in their respective ways [3]. Swarm robotics, which uses intelligent swarm systems to accomplish various tasks, such as exploration or search-and-rescue missions, has been explored in [4]. Encompassing a multitude of affordable, easily produced robots capable of mimicking natural swarm tendencies such as foraging, formation control, collective manipulation, and locating a shared source of "food", robot swarms are a functional solution to accomplish these tasks. Employing either a model of the agents or a graph abstraction of the swarm, optimization-based techniques are frequently utilized in order to develop control strategies for swarm systems [5, 6]. These approaches can effectively describe tasks such as rendezvous or consensus problems [7] and formation control [8], enabling the computation of optimal control policies and the learning of pursuit tactics to apprehend evaders [9]. However, there are limitations to these techniques, such as the tendency to get stuck in local optima in Swarm Robotics and the high computational resources required for reinforcement learning to search for optimal policies in a dynamic environment. Despite these limitations, applying population-based approaches to optimization problems in multi-agent

systems has great potential in various fields, including robotics and artificial intelligence.

This chapter explores the challenges faced in optimizing complex problems in real-world applications and using hybrid techniques to overcome them. However, traditional reinforcement learning algorithms are computationally expensive and may not be feasible in large-scale problems. Researchers have proposed incorporating RL with PSO to address this issue. Particle swarm optimization is employed to optimize the behavior of the reinforcement learning agent. The process can include optimizing the parameters of the agent or optimizing the robot's path to minimize the use of resources. Particle swarm optimization in conjunction with reinforcement learning has been shown to reduce the computational demand and optimize the robot's path, making it more feasible for real-world applications. Additionally, this chapter notes the recent advancements in deep reinforcement learning (DRL), which has shown great promise in solving complex problems in discrete decision-making [10].

The paper includes two sections, “Overview of Methodologies in Literature”, which describes the methods used in the literature, and “PSO-RL Implementation for Optimization”, discussing the experiments’ findings. The final section contains the chapter’s conclusion and highlights potential avenues for future research. This chapter outlines a specific approach to solving optimization problems using RL and PSO and provides experimental results to support the approach’s effectiveness.

2 Overview of Methodologies in Literature

Optimization is crucial to several real-world applications, including engineering design, robotics, machine learning, and economics. Particle swarm optimization (PSO) and reinforcement learning (RL) are popular optimization techniques widely used in various fields. PSO is a meta-heuristic optimization algorithm introduced by James Kennedy and Russell Eberhart [11]. A swarm of particles are simulated to move in a search space and locate the best optimal solution. RL is a learning paradigm, an intelligent policy is supposed to be learned to make optimal decisions in the given environment by maximizing the reward signal. PSO and RL have their strengths and weaknesses, and researchers have been exploring ways to integrate them to leverage their complementary features. This literature review discusses recent research on integrating PSO with RL and its promising approach to optimization.

The combination of reinforcement learning (RL) and particle swarm optimization (PSO) has become increasingly popular in recent years. Hybrid RL with PSO has been used in various applications, from robotics to financing. Several methodologies have been proposed in the literature to integrate these two techniques. Two approaches for hybrid RL with PSO include using PSO to optimize exploration in RL by adjusting exploration rate and strategy and using RL to optimize PSO parameters, such as swarm size and maximum velocity, dynamically during the

optimization process. This section elaborates on a brief study of these two techniques.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a popular machine learning technique used for decision-making in sequential decision-making problems. RL algorithms enable agents to learn how to interact with an environment by taking actions that maximize a reward signal. The reward signal provides feedback to the agent, guiding it towards better decisions [2]. Markov decision process (MDP) forms the basis of the mathematical framework for reinforcement learning, which provides a formal way to model decision-making problems in uncertain environments.

An MDP is a tuple (S, A, T, R, γ) , where:

S : is a set of states that the agent can be in

A : is a set of potential actions for the agent

$T: (S \times A \rightarrow S)$ is the transition probability function, providing the probability of transitioning from one state to another while taking a specific action

$R: (S \times A \times S \rightarrow R)$ is the reward function, which gives the reward that the agent receives for being in a particular state and taking a particular action

$\gamma \in [0, 1]$ is the discount factor, which determines the relative importance of future rewards

Learning a policy $\pi(a|s)$ is the primary objective of the agent. The policy defines the likelihood of choosing a particular action a based on the current state s . This policy is learned through trial-and-error interactions with the environment, where the agent takes actions and receives rewards based on the current state and the chosen action. After receiving rewards, the agent adjusts its policy to optimize its cumulative reward in the long run. There are several algorithms for solving MDPs, including value iteration, policy iteration, and Q-learning. These algorithms differ in how they update the value functions or policies based on the observed rewards, and in the assumptions they make about the underlying structure of the MDP [12]. While Q-learning is the most popular technique, RL has been incorporated into various metaheuristic optimization algorithms, including PSO [13].

2.2 Deep Learning Approach for RL

Deep reinforcement learning (DRL) combines reinforcement learning (RL) and deep learning (DL) to approximate the value function or policy in an RL problem via deep neural networks. DRL has successfully solved complex decision-making problems in various domains, including robotics, games, and natural language processing.

This approach has enabled agents to learn directly from high-dimensional sensory inputs like images without relying on hand-engineered features.

The leading architecture in DRL is the deep Q-network (DQN), which utilizes a deep neural network to estimate the action-value function. RL and deep learning are combined in the DQN algorithm, where a neural network is used to learn the optimal action-value function by predicting it from a state observation. In the training process, the mean-squared error between the predicted and observed action value obtained from a replay buffer of experiences is minimized. This enables the DQN to learn from past experiences and avoid catastrophic forgetting, where the agent forgets previously known information when learning new information [14].

Another popular architecture used in DRL is the policy gradient method, which directly optimizes the agent's policy. In this approach, the neural network outputs the probability distribution over actions given a state observation, and the objective function is the expected reward under this policy. The REINFORCE algorithm is commonly used to calculate the policy gradient and updates neural network weights using the expected reward gradient with policy parameters [15].

One of the main challenges in DRL is the instability and slow convergence of the training process. This is due to the non-stationarity of the environment, the correlation between samples in the replay buffer, and the high variance of the gradient estimates in policy gradient methods. Several techniques have been proposed to address these challenges, including:

- **Experience replay:** The process involves retaining the agent's experiences in a replay buffer. During the training process, random sampling from the buffer takes place, which helps to prevent correlation between samples and leads to improved stability in the learning process [16].
- **Target network:** This involves using a separate target network with fixed parameters to compute the Q-function's target values, which helps stabilize the training process and avoid overestimating the Q-values [17, 18].
- **Batch normalization:** This involves normalizing the inputs to the neural network to improve the stability of the gradient estimates and accelerate the training process [19].
- **Actor-critic methods:** It requires the utilization of two neural networks: one for the policy and another for the value function. This approach allows the agent to acquire both the policy and the value function at the same time [19].

Multi-objective optimization within Deep Reinforcement Learning presents significant research hurdles. This emerging domain focuses on concurrently optimizing multiple conflicting objectives using deep reinforcement learning (DRL). This complex issue has garnered attention due to its relevance in real-world applications like portfolio optimization and resource allocation, where objectives may clash.

One approach for multi-objective optimization with DRL is to use a multi-objective optimization algorithm that incorporates DRL, such as the multi-objective evolutionary algorithms (MOEAs) or the multi-objective grey wolf optimizer (MOGWO). These algorithms use a population-based search strategy to explore

the decision space and maintain diverse Pareto-optimal solutions. They are then refined through DRL by training a policy to maximize the reward function [20, 21].

There are several advantages to using DRL for multi-objective optimization, including the ability to handle high-dimensional and complex decision spaces, learn from experience and adapt to changing environments, and balance exploration and exploitation. However, there are also several challenges in applying DRL to multi-objective optimization problems, including the curse of dimensionality, non-stationarity, efficient exploration and exploitation, and interpretation and visualization of the solutions.

2.3 Particle Swarm Optimization

PSO algorithm is a powerful optimization methodology mimicking the natural behavior where a bunch of animals do something collectively such as a flock of birds or fish school. Optimization problems in various domains, such as engineering, finance, and machine learning, can be solved using population-based optimization algorithms [31–33]. These algorithms employ a swarm of particles navigating the search space, aiming for the optimal solution. The algorithm adjusts each particle's position and velocity using its past position, the best swarm position, and the most recent best position discovered. The PSO algorithm is highly effective at finding near-optimal solutions for various problems.

The movement of each particle within the swarm is influenced by its own experience, as well as the collective experience of its fellow particles, all of which represent potential optimization solutions. PSO can be formalized mathematically as:

$$v_i^{t+1} = \omega v_i^t + C_1 r_1 (P_{best} - P_i^t) + C_2 r_2 (G_{best} - P_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

where:

- $i = 1, 2, \dots, N$: N is the number of particle
- ω : termed as inertia weight, determines impact of old velocity on particle's new velocity
- C_1 : determines the weight of the particle's own best position, termed as cognitive learning factor
- C_2 : determines the weight of the swarm's best position, termed as social learning factor.
- r_1 and r_2 : random numbers $\in [0, 1]$
- v_i : is the velocity of the i^{th} particle
- x_i : is the position of the i^{th} particle
- P_{best} : local best position of the particle

- G_{best} : Global best position of the swarm
- P_i^t : position of the i^{th} particle

Algorithm starts with assigning initial values to particles, then evaluates the objective function, computes every particle's velocity and position, determines the personal best and global best and ascertains the stopping criterion. The algorithm updates the velocity along with position of all particles using a combination of its own best position and the best position of the swarm, as well as a random component. The algorithm repeats these steps until the acceptance threshold is achieved [22]. Particle Swarm Optimization has been efficiently utilized to tackle various optimization problems, such as feature selection, function optimization, and neural network training [34, 35]. It has also been extended and combined with other optimization algorithms to improve performance.

Multi-objective Optimization with PSO Multi-objective optimization (MOO) requires optimizing numerous conflicting objectives concurrently, and PSO is acknowledged as a potent algorithm for this task. Within this framework, PSO can yield a set of Pareto-optimal solutions. Here, enhancing one objective without compromising another proves unattainable. Diverse PSO variants have been suggested for MOO challenges, integrating various methods to sustain swarm diversity and balance convergence toward the Pareto front.

The PSO method can handle multi-objective optimization (MOO) by establishing several fitness functions and preserving a collection of non-dominated solutions, referred to as the Pareto front. It iteratively adjusts particle velocity and position using individual best and swarm's best positions in the objective space. In MOO, the swarm's optimal position is identified through the Pareto dominance relation among particles. Several PSO variations, such as MOPSO, MO-CoPSO, and MO-CPSO, have been suggested for MOO. In [23], an approach named MOPSO, utilizing a particle swarm optimization (PSO) algorithm, has been introduced. This MOPSO algorithm combines external archive and crowding distance techniques to preserve a varied set of non-dominated solutions. Its efficacy in attaining a balanced trade-off between solution convergence and diversity is assessed on diverse benchmark problems. The evaluation involved comparing its performance with other cutting-edge multi-objective algorithms. The findings indicated MOPSO's superiority over numerous algorithms, showcasing its potential for addressing intricate multi-objective optimization challenges. To address MOPSO's stagnation issue, [24] introduced MOPSO-CA, a hybrid method blending multi-objective particle swarm optimization (MOPSO) and cooperative agents. This approach employs sub-populations, with each governed by a single agent, and agents are dynamically adjusted via Pareto ranking, resolving the stagnation problem. Automated negotiation (AN) enables knowledge sharing while agents handle exploitation and exploration within and across sub-populations. During optimization, AN facilitates particles sharing information and adjusting positions, seeking compromises among objectives. Experimental results prove MOPSO-CA's superiority over comparable algorithms, achieving an optimal balance between exploration and exploitation, thus enhancing optimization efficiency and effectiveness. The MO-CPSO method, as

introduced in [25], prevents being stuck in local optima by utilizing a disturbance operation on non-dominated particles. It also employs a correlated processing strategy to maintain population diversity. Experiments on standard benchmark problems show that MO-CPSO surpasses three comparable algorithms—NSGA II, MOIPSO, and MOPSO—in terms of both convergence and diversity metrics. This establishes MO-CPSO as a more practical solution for addressing multi-objective optimization problems.

PSO exhibits potential in solving MOO challenges by generating a collection of Pareto-optimal solutions through a relatively straightforward implementation. Nevertheless, the algorithm's efficacy hinges significantly on parameter configurations and variant selection. Enhancing PSO's variants for greater efficiency and effectiveness can address practical MOO challenges successfully.

2.4 *Hybrid RL with PSO*

Hybrid reinforcement learning (RL) with particle swarm optimization (PSO) is a promising area of research that aims to combine the strengths of both techniques to improve the learning process and enhance the performance of RL algorithms. Reinforcement Learning, a subfield of machine learning, deals with the development of algorithms that learn through interactions with the environment to achieve specific objectives. However, RL often suffers from slow convergence and can get stuck in local optima while PSO can explore and exploit the search space. Combining these two techniques leads to improved convergence rates, higher quality solutions, and reduced computational time.

The concept of hybrid RL with PSO involves employing PSO as a meta-algorithm for optimizing RL algorithm hyperparameters. PSO optimizes diverse RL parameters like learning rate, exploration rate, and discount factor. It can also optimize the neural network structure in deep RL. Moreover, PSO optimizes the reward function in RL, achieving an optimal balance between exploration and exploitation.

In [26], a new method utilizing particle swarm optimization (PSO) and deep Q-network (DQN) has enhanced tunneling-induced settlement prediction accuracy. This innovative DQN-PSO optimizer refines the extreme learning machine (ELM) model by assessing rewards linked to states, actions, rules, and objective functions. Tested on a real tunnel project, the hybrid model outperforms traditional metaheuristic algorithms in accuracy and computational efficiency. It discerns connections between influential factors and ground responses, enabling rapid and precise tunneling-induced ground response forecasts. This research underscores the potential of combining RL and PSO algorithms to create improved predictive models for engineering applications. [27] presents a model-based reinforcement learning approach that uses PSO to optimize control action sequences online. The proposed particle swarm optimization policy (PSO-P) method doesn't need initial policy representation while being effective for continuous state and action spaces. It

treats RL as an optimization problem opening the door to the applicability of multitudes of optimization methods available in literature. The findings suggest that PSO-P achieves superior performance compared to other RL techniques regarding convergence speed, computational efficiency, and stability. The proposed method is expected to be useful in real-world applications where the optimization of complex control problems is required. The HRLPSO algorithm, described in [28], aims to solve the issue of PSO algorithm's local optima trap. Grounded in reinforcement learning theory, it employs adaptive weights, learning factors, and dimension learning. This enhances individual outcomes, balancing global exploration and local development. Furthermore, improved reinforcement learning and mutation strategies are used to improve the quality of individual and globally optimal solutions. The results showed its usefulness in balancing individual learning capability and social learning capability [28].

Role of Hybrid RL in Multi-Objective Optimization Hybrid RL is a combination of RL and other optimization techniques, such as evolutionary algorithms or swarm intelligence algorithms, to improve the performance of the optimization process. However, Multi-objective optimization (MOO) issues pose the difficulty of optimizing numerous conflicting objectives simultaneously. It is necessary to address all the objectives simultaneously to achieve the most optimal outcome. Balancing exploration and exploitation in MOO for achieving convergence towards the Pareto front, signifying the optimal objectives trade-off is a major challenge. Hybrid RL exhibits potential in MOO by effectively managing exploration and exploitation in the search space, leading to promising outcomes.

The Q-Learning-based Particle Swarm Optimization (QLPSO) [29] approach integrates PSO and RL to enhance traditional PSO performance. It employs RL to optimize PSO parameters through learning from past experiences, enhancing decision-making abilities. A method described in [30] develops a MOPSO algorithm for collaborative path planning of multiple UAVs in complex environments with constraints. The MOPSO algorithm, as proposed, integrates RL to generate paths within the specified constraints, employing a multi-input collaboration strategy. The algorithm, through RL, chooses the optimal position update mode, ensuring high performance. Results in the paper show efficiency and robustness in solving path planning for multiple UAVs.

3 PSO Implementation for Optimization

In this section, the implementation of PSO for solving a specific optimization problem has been described. We first propose a model for the problem in the following subsection and then describe the experimental setup used to evaluate the performance of the PSO algorithm.

3.1 Proposed Model

Our proposed model is to search a surface for the optimum global point with minimum traveling cost and guide the agent to reach the corresponding optimum point or target point in the simulation space. For searching a surface, we need to identify the objective function we want to optimize. Our experiment uses the Sphere function Eq. (3), a well-known test function for optimization problems.

$$f(x) = \sum_{i=1}^n x_i^2 \quad (3)$$

Where x_i is the position of the particles in a 2D search space, and $f(x)$ is the fitness value of the particles. The particle must find the coordinate of the position with a minimum fitness value using the Particle Swarm Optimization algorithm. The algorithm has been illustrated as follows:

Algorithm: Particle Swarm Optimization

```

FOR each i
    FOR each d
        Initialize x(i)
        Initialize v(i)
    END FOR
END FOR
DO
    FOR each i
        Calculate fitness
        IF fitness > P_best
            SET P_best = fitness
        END IF
    END FOR
    G_best = particle with best fitness
    FOR each i
        FOR each d
            Update x(i)
            Update v(i)
        END FOR
    END FOR
    k = k+1
WHILE maximum iterations or minimum convergence criteria are met

```

Each particle has a corresponding agent in the simulation space. A scaling factor has been used in the simulation space to move the agent according to the particle's movement. The scaling factor has been shown in Eqs. (4) and (5).

$$x' = \frac{x + k}{2k} W \quad (4)$$

$$y' = \frac{y + k}{2k} H \quad (5)$$

Where k is the magnitude of the range boundary of the search space. Since our search space is in the range of -10 to $+10$, the value of the k is 10 . H and W represent the height and the width of the simulation space, respectively.

3.2 Experimental Setup

We will discuss the experimental setup of PSO implementation to solve the proposed test optimization function over a surface in this subsection. Two spaces: search space and simulation space, have been used for the experiment. Particles move in the search space, and agents move in the simulation space. Each particle has a corresponding agent in the simulation space, and the agent's movement completely depends on the direction of particles in the search space.

Search space spreads over $[-10, +10]$, and simulation space is a graphical window with a height of 800 pixels and a width of 1000 pixels. There are five particles, and the number of agents equals the number of particles. The particles are generated in a specific search space region $[-10, -5]$; hence, agents start their movement from the equivalent region of the simulation space. Figure 1 shows the

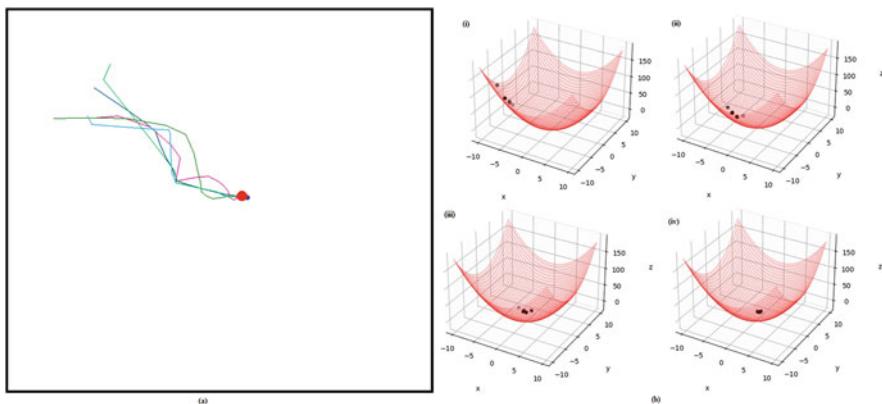


Fig. 1 (a) Simulation space with agent trajectories. (b) Movement of particles in search space

movement of the particle as well as the agents. Agents are denoted as the solid red circle, and the target point is represented as the solid blue circle. The trajectory of each agent has been shown in a different color. We assume the value of inertia is 0.5, cognitive weight (C_1) is 0.1 and social weight (C_2) is 1.0. The experiment has been performed in 30 epochs.

4 Result and Discussion

In this subsection, we elaborate on the results obtained from our experiments on PSO implementation. The investigation of the PSO algorithm runs until the convergence criteria have been met, or the maximum epoch has been reached. The convergence criteria are determined by computing the difference between the global best fitness value of the swarm and the fitness value of the optimum point. If this difference is below a predefined threshold value ($\epsilon = 0.05$), we consider the algorithm to have converged. If the max. epochs are reached before convergence, the experiment is stopped. Figure 2 in our study represents the convergence of the PSO algorithm towards the optimal fitness value. As the algorithm progresses through its iterations, the global best value of the swarm steadily decreases towards the optimum fitness value. The graph shows a clear trend towards convergence, with a decreasing global best value over time.

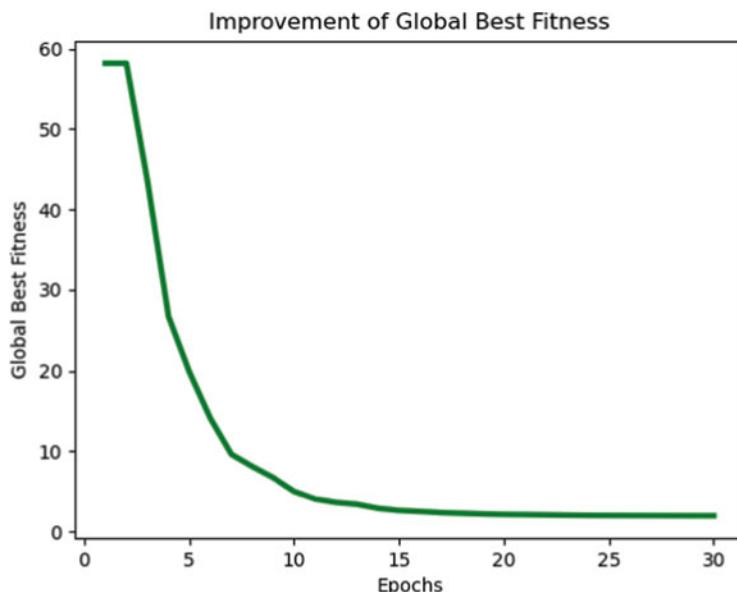


Fig. 2 Global best fitness value of the swarm

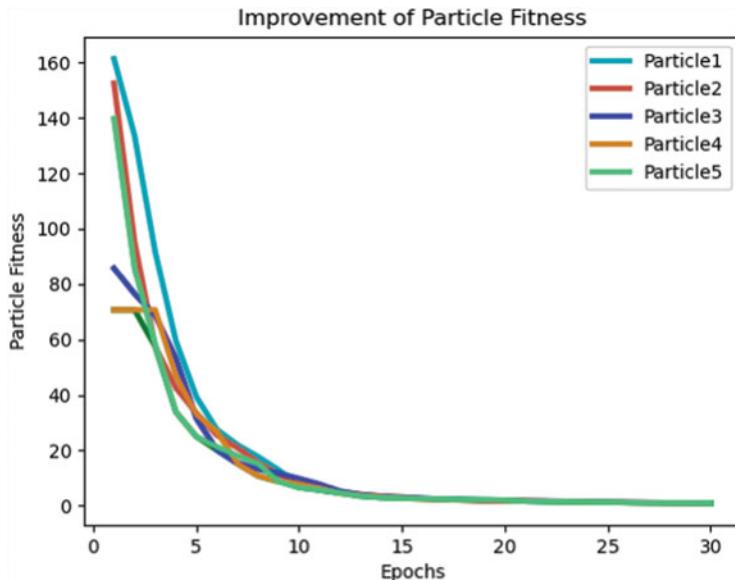


Fig. 3 Fitness value of the particles

The performance of particles in a given system over multiple epochs or time intervals is improving. In this experiment, the fitness value of particles is an important metric to evaluate their effectiveness. Figure 3 is a graph that represents this improvement in fitness value. The particles are becoming more effective in their tasks as the fitness value of particles is improving over the progress of the epochs. The graph from Fig. 3 shows that when the execution of the experiment begins, the particles have a higher fitness value. However, as time progresses, this fitness value gradually decays. This decay is likely since the particles constantly adapt and explore new possibilities to improve their fitness value. Eventually, the fitness value of the particles stops decaying and stabilizes close to the optimum fitness value. The particles have reached a point where they can no longer significantly improve their fitness value and have reached a state of optimal performance.

It is important to note that the convergence of a metaheuristic algorithm does not guarantee the global optimum solution. It only indicates that the algorithm has converged to a locally optimal solution. On the other hand, it can be the best solution the algorithm has found so far. Our experiment shows that the PSO algorithm effectively finds reasonable solutions for the test function. The convergence criteria ensure that the algorithm did not run indefinitely, and the results indicated a clear trend towards convergence.

5 Conclusion and Future Direction

Integrating Particle Swarm Optimization (PSO) with Reinforcement Learning (RL) is a promising approach that can enhance the performance of RL algorithms in solving complex real-world problems. PSO can be used to optimize the exploration process of RL agents, by efficiently searching for the optimal policy in the action space. By combining PSO and RL, the convergence speed of RL algorithms can be improved as well as reducing the exploration time, and increasing the stability of the learning process.

Nevertheless, combining PSO and RL demands meticulous attention to algorithmic aspects, including fitness function design, exploration-exploitation balance, and optimization of PSO algorithm parameters. Furthermore, adapting PSO with RL must be customized to the particular problem in question, as problem characteristics can influence the integration's efficacy.

One potential future work on integrating Particle Swarm Optimization (PSO) with Reinforcement Learning (RL) could be to explore its effectiveness in solving complex real-world problems. To accomplish this, researchers could begin by selecting a challenging control task, such as a swarm control that needs to learn to maintain the togetherness and reach the target in a cluttered environment. Next, an RL algorithm can be developed that can learn the optimal policy through trial and error. Once the RL algorithm is in place, researchers could integrate PSO into the RL framework. PSO could be used to optimize the hyperparameters of the RL algorithm. Alternatively, PSO could be used to optimize the RL policy directly, by exploring for the optimal set of weights that govern the agent's behavior. Future work could explore different ways of integrating PSO into RL frameworks, as well as the generalization and transferability of PSO-optimized RL policies across different tasks and environments.

Overall, integrating PSO with RL has a lot of potential as a research field that could lead to the creation of better learning algorithms that are more proficient and productive. This method has the ability to make substantial impacts on the areas of artificial intelligence and machine learning through further investigation and improvement.

References

1. Meerza SIA, Islam M, Uzzal MM. Q-learning based particle swarm optimization algorithm for optimal path planning of swarm of mobile robots. In: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). IEEE; May 2019. p. 1–5.
2. Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge, MA: MIT Press; 2018.
3. Lima H, Kuroe Y. Swarm reinforcement learning algorithms based on particle swarm optimization. In: 2008 IEEE International Conference on Systems, Man and Cybernetics. IEEE; 2008.

4. Hottenrauch M, Adrian S, Neumann G. Deep reinforcement learning for swarm systems. *J Mach Learn Res.* 2019;20(54):1–31.
5. Jadbabaie A, Lin J, Stephen Morse A. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans Autom Control.* 2003;48(6):988–1001.
6. Lin Z, Broucke M, Francis B. Local control strategies for groups of mobile autonomous agents. *IEEE Trans Autom Control.* 2004;49(4):622–9.
7. Lin J, Stephen Morse A, Anderson BDO. The multi-agent rendezvous problem. Part 2: the asynchronous case. *SIAM J Control Optim.* 2007;46(6):2120–47.
8. Ranjbar-Sahraei B, et al. A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems. *IEEE Trans Ind Electron.* 2012;59(8):3124–34.
9. Zhou Z, et al. Cooperative pursuit with Voronoi partitions. *Automatica.* 2016;72:64–72.
10. Li K, et al. Deep reinforcement learning for combinatorial optimization: covering salesman problems. *IEEE Trans Cybernet.* 2021;52(12):13142–55.
11. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 4. IEEE. 1995.
12. Yang Y, Wang J. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583.* 2020.
13. Wauters T, et al. Boosting metaheuristic search using reinforcement learning. In: Talbi EG, editor. *Hybrid metaheuristics.* Berlin: Springer; 2013. p. 433–52.
14. Mnih V, et al. Human-level control through deep reinforcement learning. *Nature.* 2015;518 (7540):529–33.
15. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn.* 1992;8(3–4):229–56.
16. Schaul T, et al. Prioritized experience replay. *arXiv preprint arXiv: 1511.05952.* 2015.
17. Hasselt H. Double Q-learning. In: *Advances in Neural Information Processing Systems 23.* 2010.
18. Lillicrap TP, et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971.* 2015.
19. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: *International Conference on Machine Learning.* PMLR; 2015. p. 1889–97.
20. Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput.* 1999;3(4):257–71.
21. Mirjalili S, et al. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Syst Appl.* 2016;47:106–19.
22. Shi Y. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546).* Vol. 1. IEEE; 2001.
23. Coello CA, Lechuga MS. MOPSO: a proposal for multiple objective particle swarm optimization. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600).* Vol. 2. IEEE; 2002.
24. Kouka N, Fdhila R, Alimi AM. Multi objective particle swarm optimization based cooperative agents with automated negotiation. In: *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part IV 24.* Cham: Springer International Publishing; 2017.
25. Shen, Y, Wang G, Liu Q. Correlative particle swarm optimization for multi-objective problems. In: *Advances in Swarm Intelligence: Second International Conference, ICSI 2011, Chongqing, China, June 12–15, 2011, Proceedings, Part II 2.* Berlin: Springer; 2011.
26. Zhang P, Li H, Ha QP, Yin Z-Y, Chen R-P. Reinforcement learning based optimizer for improvement of predicting tunneling-induced ground responses. *Adv Eng Inform.* 2020;45: 101097.
27. Hein D, Hentschel A, Runkler TA, Udluft S. Reinforcement learning with particle swarm optimization policy (PSO-P) in continuous state and action spaces. *Int J Swarm Intell Res.* 2016;7(3):23–42.

28. Huang W, Liu Y, Zhang X. Hybrid particle swarm optimization algorithm based on the theory of reinforcement learning in psychology. *Systems*. 2023;11(2):83.
29. Liu Y, Lu H, Cheng S, Shi Y. An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2019. p. 815–22.
30. Zhang X, et al. Multi-objective particle swarm optimization with multimode collaboration based on reinforcement learning for path planning of unmanned air vehicles. *Knowl Based Syst*. 2022;250:109075.
31. Xie R. A flock-of-starling optimization algorithm with reinforcement learning capability. In: MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. 2020.
32. Xu Y, Pi D. A reinforcement learning-based communication topology in particle swarm optimization. *Neural Comput & Applic*. 2020;32:10007–32.
33. Iima H, Kuroe Y. Swarm reinforcement learning algorithm based on particle swarm optimization whose personal bests have lifespans. In: Neural Information Processing: 16th International Conference, ICONIP 2009, Bangkok, Thailand, December 1–5, 2009, Proceedings, Part II 16. Berlin: Springer; 2009.
34. Parsopoulos KE, Vrahatis MN. Particle swarm optimizer in noisy and continuously changing environments. *Methods*. 2001;5(6):23.
35. Pan H, Wang L, Liu B. Particle swarm optimization for function optimization in noisy environment. *Appl Math Comput*. 2006;181(2):908–19.

Synergies Between Natural Language Processing and Swarm Intelligence Optimization: A Comprehensive Overview



Ujwala Bharambe, Rekha Ramesh Manimala Mahato, and Sangita Chaudhari

Abstract Natural Language Processing (NLP) constitutes a vital facet of artificial intelligence, focusing on the intricate interplay between human language and computing systems. This rapidly advancing field explores how machines can comprehend, interpret, and produce human language with precision and naturalness. Concurrently, Swarm Intelligence Optimization (SI) emerges as a metaheuristic approach inspired by the collaborative behaviours observed in social animals. SI leverages these principles to tackle complex optimization challenges, efficiently discovering robust solutions within limited time frames.

This chapter discusses natural language processing and Swarm Intelligence Optimization and the potential applications these technologies can have in various fields, including social media analytics, recommender systems, chatbots, and virtual assistants. Additionally, the chapter presents recent developments in these areas, including deep learning-based NLP models, such as transformer-based models, and new Swarm Intelligence Optimization algorithms, such as the bat algorithm and ant colony optimization. These new methods are shown to outperform existing approaches in terms of efficiency and accuracy, and have become widely used in practice. They have enabled the development of more complex and powerful NLP

U. Bharambe ()

Thadomal Shahni Engineering College, Mumbai, Maharashtra, India

e-mail: ujwala.bharambe@thadomal.org

R. Ramesh

Shah & Kutchhi Engineering College, Mumbai, Maharashtra, India

Indian Institute of Technology, Mumbai, India

e-mail: rekha.ramesh@iitb.ac.in

M. Mahato

Shah & Kutchhi Engineering College, Mumbai, Maharashtra, India

e-mail: manimala.mahato@sakec.ac.in

S. Chaudhari

Ramrao Adik Institute of Technology, D. Y. Patil University, Nerul, Navi Mumbai, Maharashtra, India

e-mail: sangita.chaudhari@rait.ac.in

models, which have been used for a variety of tasks, such as Neural Machine Translation systems and Sentiment Analysis Classification, Question Answering, Topic Modeling, Sentiment analysis and machine translation. The chapter presents a thorough analysis of challenges and potential future pathways in the field of NLP and SIO.

Keywords Natural language processing · Swarm intelligence · Metaheuristic optimization · Evolutionary computing

1 Introduction

Natural Language Processing and Swarm Intelligence Optimization (SIO) are two rapidly advancing technologies that have the potential to revolutionize a wide range of applications [1]. NLP encompass utilization of algorithms and computational models to analyze, comprehend, and produce human language, while SIO is an optimization technique inspired by the collective behavior of social animals and insects. NLP plays a vital role in the contemporary digital era of increasing text and image. It enables machines to understand and generate human language, and has application in various fields such as Speech Recognition, Sentiment Analysis, Machine Translation and Chatbots [2, 3]. NLP can help organizations extract meaningful insights from unstructured data and make data-driven decisions, leading to increased efficiency and productivity. However, SIO is a powerful optimization method that can be applied to resolve challenging issues in various domains such as logistics, engineering and finance. SIO uses algorithms to simulate the behavior of swarms in nature. This allows SIO to find optimal solutions to complex problems more efficiently than traditional optimization techniques [4].

In recent years, the combined application of NLP and SIO has gained significant attention, leading to innovative approaches that leverage the collective intelligence of swarms to enhance various aspects of language processing. This chapter in the book offers an extensive exploration of the synergies between these two fields, emphasizing their capacity to optimize NLP tasks and models. The swarm intelligence algorithms, inspired by the collaborative patterns observed in social insects can be seamlessly incorporated into NLP workflow. This is covered in a pipeline for Swarm Intelligence-based optimization in NLP. The pipeline covers preprocessing, feature extraction, model training, and evaluation, showcasing how swarm intelligence optimization techniques can enhance each stage and improve overall performance.

One crucial aspect of NLP is hyperparameter tuning, which significantly affects the performance of models [5]. The chapter delves into the application of swarm optimization algorithms for hyperparameter tuning. Ant Colony Optimization and Particle Swarm Optimization are used to efficiently search the hyperparameter space and find optimal configurations, enabling better model performance and generalization [1].

Moreover, the chapter explores the application of swarm intelligence in pretraining language models. Pretrained language models like BERT and GPT have transformed NLP tasks drastically. Swarm intelligence algorithms can be leveraged to optimize the pretraining process, enabling more effective learning and knowledge representation in language models. The chapter discusses how swarm intelligence can enhance pre-training methodologies, improving generation capabilities and language understanding.

Transfer learning has also been a significant advancement in NLP, allowing models trained on one task or domain to be adapted to another with minimal data [6]. The chapter investigates how swarm intelligence can be utilized in transfer learning-based language processing. Swarm optimization algorithms aid in fine-tuning pretrained models, optimizing the transfer learning process, and improve models' adaptation to specific tasks or domains.

Furthermore, the chapter explores the application of swarm intelligence in large-scale language models. With the advent of enormous language models like GPT-3, handling and training such models efficiently becomes challenging. Swarm intelligence offers scalable and distributed optimization approaches that can be applied to large-scale language models. The chapter discusses how swarm optimization algorithms can accelerate training, improve convergence, and enhance the performance of these models on various NLP tasks.

Through a comprehensive review of recent advances, this chapter aims to offer a more profound insight into the potential of NLP and SIO in driving innovation and solving complex problems. We will conclude the chapter with a thorough analysis of the challenges and future directions for research in this exciting and rapidly evolving field.

2 Natural Language Processing Techniques

With the goal of enabling machines to understand, interpret, and generate human-like text, NLP employs a range of techniques and algorithms to analyze and process natural language. There are several phases in NLP, including lexical analysis (tokenization and removing unnecessary elements), syntactic analysis (understanding grammatical structure), semantic analysis (extracting meaning and context), and discourse analysis (comprehending larger text units and sentiment). In addition to chatbots, language translation, and sentiment analysis, NLP applies these phases to enable machines to process and interpret language. Using NLP techniques, text analysis can extract meaning, classify documents, identify sentiment, and more. Below, you will find a description of natural language processing techniques, examples of natural language processing applications, a commentary about technological advances in NLP applications, and a list of challenges facing NLP applications.

2.1 *Overview of Natural Language Processing Techniques for Text Analysis*

NLP has become an important area of research and application in text analysis. Text analysis involves processing large volumes of unstructured text data to extract insights and information from it. It involves uncovering and leveraging interesting, nonnumerical knowledge within unstructured or unformatted textual content. All types of information retrieval, entity relations, event extraction, or text classification and grouping, are included in this.

The primary goal of NLP is to achieve a holistic comprehension of unrestricted text by discerning the “who did what to whom, when, where, how, and why” within it. This process commonly relies on linguistic principles, including part-of-speech and grammatical structure. NLP tackles the challenge of extracting deeper meaning from free text, which involves resolving ambiguities and anaphora. NLP relies on different resources, including lexicons, grammar rules, and knowledge representations like ontologies or thesauri to achieve this. Table 1 shows some commonly used NLP techniques and their meaning.

In NLP, several deep neural network architectures have been commonly used due to their effectiveness in various NLP tasks [7]. Some of the commonly used architectures are:

- (a) **Recurrent Neural Networks (RNNs):** RNNs are specifically engineered to handle sequential data, rendering them well-suited for NLP tasks that require sequence modeling, such as sentiment analysis, machine translation and language modeling. The challenge in capturing long-range dependencies arises from the vanishing gradient issue inherent to the fundamental RNN architecture. This challenge can be tackled using Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), that are more advanced variants of basic RNN.

Table 1 NLP techniques and their purpose

NLP techniques	Purpose of NLP techniques
Tokenization	Breaking down a sentence or a paragraph into individual words or tokens
Part-of-speech (POS) tagging	Assigning grammatical role (noun, verb, adjective, etc.) to each word in a sentence
Named entity recognition (NER)	Recognizing and retrieving entities, including individuals, locations, corporations, and other named elements, from textual content
Topic modeling	A technique for a method for locating themes or patterns in a collection of texts or documents
Text classification	Categorizing or labeling a piece of text based on predefined criteria
Text summarization	Automatically generating a summary of a longer text document
Word embedding	A method for encoding words in a high-dimensional vector space, enabling the capture of the semantic meaning of words and their relationships with other words

There have been improvements in the performance of LSTM and GRU networks when it comes to capturing long-term dependencies in text.

- (b) **Convolutional Neural Networks (CNNs):** CNNs, initially designed for image processing, have found utility in NLP tasks like sentiment analysis and text classification. In NLP, CNNs operate on 1D input, treating text as a sequence of words or characters. By capturing local features or patterns in the input, they are effective in tasks such as document classification and text categorization [1].
- (c) **Transformer:** The Transformer architecture has garnered notable interest in recent years, mainly due to its impressive performance in tasks like machine translation and language comprehension [8]. Transformers use self-attention mechanisms to ensure that dependencies between words in the input text are captured. This enhances the modeling of long-range dependencies. The Transformer architecture has seen widespread adoption in state-of-the-art models, such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers)

2.2 *Applications of Natural Language Processing*

NLP finds extensive applications across diverse industries, including healthcare, education, finance, customer service, and entertainment. The adoption of NLP techniques has transformed human-machine interactions, enabling machines to comprehend human language and provide relevant responses. Some of the widely used applications of NLP are:

- (a) **Sentiment Analysis:** Sentiment analysis is used to identify the emotional tone within a text, like a tweet or a customer review [9]. Common NLP techniques used for sentiment analysis include Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).
- (b) **Text Classification:** Text classification entails assigning a piece of text to one or x predefined categories, such as categorizing it as spam or non-spam. The most common methods of text classification used by NLP include Decision Trees, Naive Bayes and Maximum Entropy models [10].
- (c) **Named Entity Recognition (NER):** NER encompasses the recognition and extraction of entities like names, organizations, and locations from a given document. Common NLP techniques used for NER include Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), and Neural Networks [5].
- (d) **Machine Translation:** Machine translation refers to the process of converting a text or document from one language into another. Neural Machine Translation (NMT), Statistical Machine Translation (SMT), and Rule-based Machine Translation (RMT) are types of NLP techniques used for machine translation. [11].
- (e) **Speech Recognition:** Speech recognition involves converting spoken language into text. Standard NLP techniques used for speech recognition include Artificial

Neural Networks (ANNs), Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs).

- (f) **Text Summarization:** Text summarization is an NLP technique to create a concise summary of long documents and represent them in small, simpler sentences. Traditional methods used for text summarization include Latent Semantic Analysis (LSA), LexRank and Deep Learning-based methods [12].
- (g) **Question Answering:** Question answering involves finding answers to questions asked in natural language. NLP techniques used for question answering include Knowledge Graphs, Neural Networks, Information Retrieval (IR) models [13].
- (h) **Text Generation:** Text generation involves generating natural language text from a given input. There are several NLP techniques that can be used to generate text, including generative adversarial networks (GANs), Recurrent Neural Networks (RNNs) and transformer-based models.
- (i) **Text Mining:** The process of extracting useful information from a large text corpus is called Text mining. Standard NLP techniques used for text mining include Latent Dirichlet Allocation (LDA), Topic Modeling, and Non-negative Matrix Factorization (NMF) [14].
- (j) **Chatbots and Virtual Assistants:** Chatbots and virtual assistants use NLP techniques to interpret user input and provide appropriate responses. NLP techniques used for chatbots and virtual assistants include Intent Recognition, Named Entity Recognition (NER), and Dialog Management.
- (k) **Information Extraction:** The process of extracting structured information from unstructured text is called information extraction. Among the NLP techniques used to extract information are Pattern Matching, Rule-based Systems, and Machine Learning.
- (l) **Text-to-Speech (TTS):** TTS involves converting written text into spoken language. Common NLP techniques used for TTS include Speech Synthesis Markup Language (SSML), Neural TTS, and Concatenative TTS.

2.3 NLP Technological Evolution

The field of NLP has been the subject of ongoing research for several decades, with its origins tracing back to the late 1940s. Figure 1 shows the technological evolution in the field of NLP. The first NLP system, the Georgetown-IBM experiment, was the first public demonstration of a machine translation system developed in 1949 to translate Russian sentences into English using machine translation techniques [15]. In the 1950s–1960s Rule-based systems were developed for applications such as parsing and text generation, but they are limited by the complexity of the rules required. During same period, other prominent NLP application areas such as speech processing started emerging. The theoretical framework of generative grammar prevails in the language processing community, whereas statistical information theory takes precedence in the speech community [16]. Statistical techniques, such

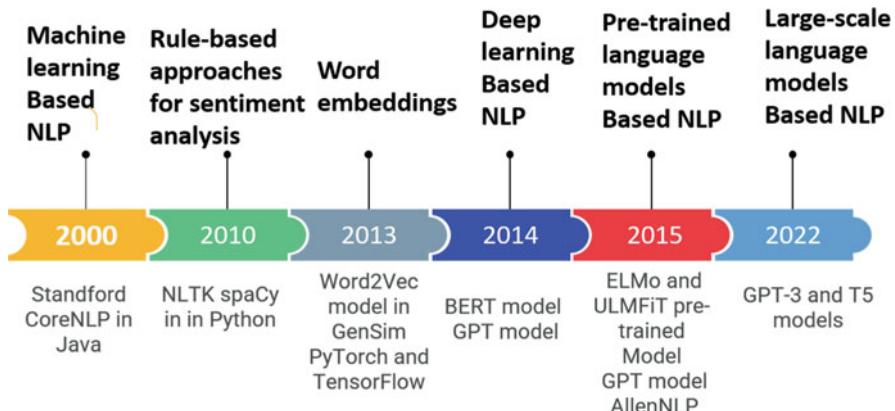


Fig. 1 Evolution of natural language processing

as Hidden Markov Models and n-grams were introduced during the 1970s–1980s for tasks such as speech recognition and language modeling.

In the 1990s, the advancement of machine learning algorithms, such as Support Vector Machines and decision trees, brought about substantial improvements in applications like named entity recognition and sentiment analysis.

The language modelling, machine translation, and text classification techniques improved and gained importance with the introduction of neural network-based models, like RNN and CNN in early 2000. The researchers started adopting deep learning techniques and word embeddings, which have since become essential components of contemporary natural language models around the year 2010.

In recent years, a noteworthy advancement in NLP has been the introduction of extensive pre-trained language models, including BERT (Bidirectional Encoder Representations from Transformers), GPT-3 (Generative Pre-trained Transformer 3), and T5. These models have undergone training on massive volumes of text data, enabling them to generate text resembling human language and perform various NLP tasks with few examples. As research in this field continues, we can expect to see even more powerful language models that are capable of understanding natural language at an even deeper level.

2.4 Challenges in NLP Applications

Natural language is often ambiguous, meaning that a single word or phrase can have multiple meanings depending on the context. This can make it difficult for NLP systems to interpret and understand text accurately. Further, NLP applications may be specific to a particular domain or subject matter, making it challenging to develop accurate models that can handle a wide range of topics and contexts. Large amounts of labelled data to train machine learning models are required by most NLP systems

require and, in many cases, such data may not be available. They may need to handle multiple languages, which can pose significant challenges in language identification, translation, and interpretation. Text data may contain errors, typos, and inconsistencies, which can affect the accuracy of NLP systems. It may also contain irrelevant or misleading information, making it difficult for NLP systems to interpret the data accurately. In order to accurately understand and interpret natural language, NLP systems must be able to take into account the context in which the language is being used. This can be particularly challenging when dealing with sarcasm, irony, or other forms of figurative language.

Overall, addressing these challenges requires the development of robust NLP models that can handle ambiguity, data sparsity, domain specificity, multilingualism, noise, errors, and contextual understanding. Swarm optimization algorithms, such as particle swarm optimization (PSO) and ant colony optimization (ACO), can be used to optimize machine learning models used for NLP tasks that can handle the challenges mentioned above [17]. These are further described in detail in later sections.

3 Introduction to Swarm Intelligence

Optimization is a fundamental concept in artificial intelligence (AI) that aims to find the best solution from a set of possible options, considering specific criteria or objectives. It plays a crucial role in various AI applications. Swarm optimization, metaheuristics, and population-based optimization are alternative approaches that offer flexibility, adaptability, and efficiency in searching for solutions (Refer Fig. 2).

Swarm intelligence is a field of study that seeks to understand the collective behavior of groups of individuals, such as social insect colonies, flocks of birds, and schools of fish. The term “swarm intelligence” was coined by Benoît Mandelbrot in the 1980s, and Eric Bonabeau further developed it in the 1990s. Eric Bonabeau has defined swarm intelligence as “any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies

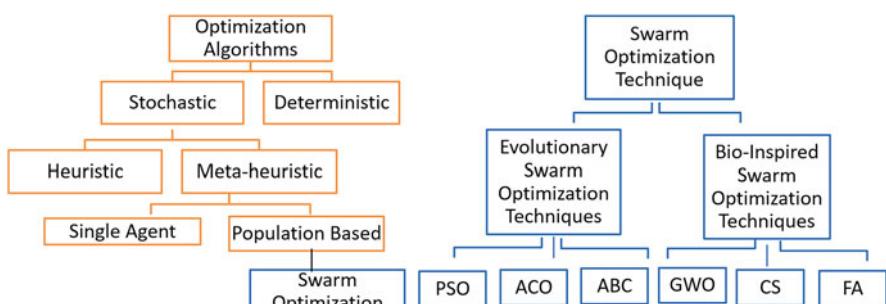


Fig. 2 Classification of optimization algorithm

and other animal societies". This definition emphasizes using the collective behavior of groups of individuals as inspiration for designing algorithms and problem-solving devices.

The collective behavior of social insect colonies, such as ants, bees, and termites, is characterized by self-organization and division of labour [18]. Self-organization is a process in which individual agents in a system interact with each other and the environment to create emergent behavior patterns at the group level. The interactions between individuals are based on simple rules or heuristics and do not require global knowledge or communication. Instead, the behavior of each individual is influenced by its local environment and the behavior of its neighbours. Division of labor is a mechanism for allocating tasks among the individuals in a group based on their abilities and the group's needs. This mechanism allows the group to perform complex tasks impossible for any individual to perform alone. The allocation of tasks can be based on individual preferences, skills, or physical characteristics.

Swarm intelligence algorithms, such as ant colony optimization (ACO) and particle swarm optimization (PSO), are effective in solving optimization problems that involve a large number of variables, nonlinear relationships, and multiple optima. In such problems, traditional optimization algorithms may struggle to find the global optimum due to the high-dimensionality and complexity of the search space (Sahoo).

Swarm optimization techniques are a family of metaheuristic optimization algorithms that are inspired by the collective behavior of social animals such as birds, bees, ants, and fish. The idea is to mimic these groups' collective intelligence and cooperation to solve optimization problems. Swarm optimization techniques typically involve a set of candidate solutions, called particles, agents, or individuals, that move in a search space to find the optimal solution. Each particle has a position and a velocity that are updated iteratively based on some optimization criteria, such as fitness, objective function value, or error, using a set of rules that reflect the behavior of the social group [19].

Particles agents, or individuals, communicate and exchange information to reach a global optimization solution. As in the case of social insects, the communication can be direct, as with pheromones or visual cues, or indirect, as with swarm algorithms that use global or local best positions. Swarm optimization techniques typically include parameters that control the swarm's behavior, such as the number of particles, the velocity update rule, the learning factors, and the termination criteria. Algorithm performance depends greatly on the selection of these parameters, which often requires some tuning.

The evolution of swarm-based optimization can be traced back to the 1960s, when researchers began studying the behavior of social insects and birds developed by Marco Dorigo in 1991 [20]. Ant Colony Optimization was the first swarm intelligence algorithm. Kennedy and Eberhart proposed Particle Swarm Optimization (PSO) in 1995, and it is now one of the most commonly used algorithms for swarm-based optimization [21]. PSO was inspired by the behavior of bird flocks and fish schools, which exhibit coordinated movement without any central control. Over time, researchers have developed several variants of PSO and other swarm-based

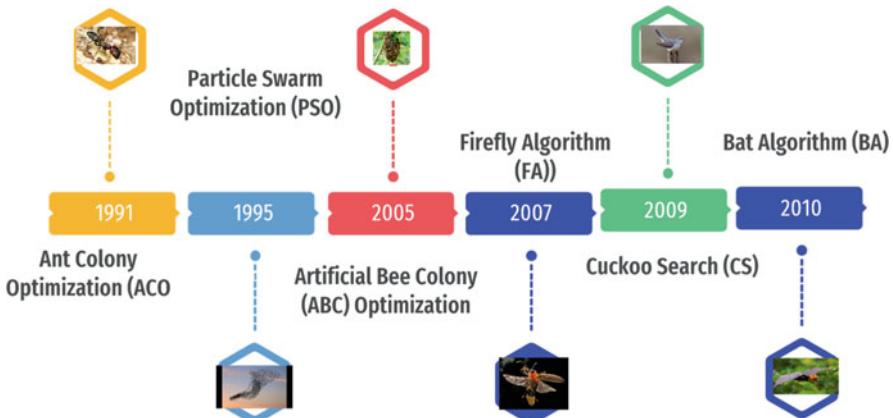


Fig. 3 Timeline of swarm intelligence algorithms

optimization algorithms, including Artificial Bee Colony (ABC) optimization, and Firefly Algorithm (FA). In Fig. 3, the timeline shows the algorithms' application to a variety of optimization problems, including feature selection, parameter optimization, and clustering data, and in Table 2, SI algorithms and their tools are compared.

3.1 ACO

Ant colony optimization (ACO) is a metaheuristic algorithm that draws inspiration from the foraging behavior of ant colonies. Its purpose is to address optimization problems by imitating the actions of ants, which utilize pheromone trails for communication and efficient discovery of the shortest path to a food source [22].

ACO uses a colony of artificial ants to find the optimal solution to a given problem. The ants explore the solution space by constructing solutions incrementally and laying pheromone trails to mark the quality of their solutions. The pheromone trails evaporate over time, and the ants preferentially follow trails with higher pheromone concentrations, reinforcing the suitable solutions and avoiding the bad ones. Consider the scenario with an ant colony and food source for simplicity. There are two paths of different length for possible traversal between the food source and the ant colony. The shorter path will have a higher concentration of pheromone, leading the ants to take the shorter route and eventually optimize the route for the colony [23]. Here are the steps for the Ant Colony Optimization algorithm: (1) *Initialization*: The ACO algorithm begins by randomly generating a set of artificial ants. Each ant is given a starting position and a goal position. The ants then begin to explore the search space, moving from one node to another. (2) *Choose a path*: When an ant reaches a node, it must choose a path to the next node. The ant makes this choice based on the following factors: The amount of pheromone on the path, The distance between the current node and the next node, a random number.

Table 2 Swarm Intelligence algorithms and its tools with programming languages

Algorithm	Advantages	Disadvantages	Types of algorithms	Libraries \ tools	Programming language
PSO	Easy to implement and tune	May get stuck in local optima	Global Best PSO, Local Best PSO, Hybrid PSO	pyswarm, PSO-Lib, PySwarms, Optunity	Python, MATLAB, R
ACO	Can find optimal solutions in complex search spaces	Slow convergence, difficulty in tuning parameters	Ant Colony Optimization, Max-Min Ant System, Ant-Q	Ant Colony Optimization Library, JCOLIB	Java, Python
ABC	Handles noisy and dynamic optimization problems	Easily trapped in local optima, requires careful tuning of parameters	Standard ABC, Modified ABC, Multi-objective ABC	ABC-Py, ABC Toolbox, ABC Clustering	Python, MATLAB, R
FA	Efficiently explores search space, flexible algorithm	May converge to suboptimal solutions	Standard FA, Enhanced FA, Hybrid FA	FALCON, FA-Matlab, JFA	Python, MATLAB, Java
GWO	Effective at global exploration and convergence	Limited research and applications compared to other algorithms	Grey Wolf Optimizer, Modified GWO, Hybrid GWO	GWO Toolbox, PyGWO, GWO-Java	Python, MATLAB, Java

The ant is more likely to choose a path with a higher pheromone trail, but it will also consider the distance between the nodes and a random number. This helps to ensure that the ants explore the entire search space and do not get stuck in local minima.

(3) *Pheromone evaporation:* Over time, the pheromone trails on the graph evaporate. This means that the pheromone trails become weaker. The evaporation rate is a parameter of the ACO algorithm. A higher evaporation rate means that the pheromone trails will evaporate more quickly. The evaporation of the pheromone trails helps to prevent the ants from getting stuck in a local minimum. As the pheromone trails evaporate, the ants are more likely to explore new paths. This helps the algorithm to find better solutions. (4) **Termination:** The ACO algorithm terminates when one of the following conditions is met:

- A certain number of iterations has been completed.
- The ants have found a solution that is within a certain distance of the optimal solution.
- The ants have converged on a single path.

An optimization algorithm called ACO can be used to solve a variety of problems. In addition to being easy to implement and understand, it can solve problems involving many variables. There are, however, some drawbacks, including low convergence rates and parameter sensitivity. The detail algorithm is given below

1. Initialize pheromone levels on all edges to a small constant value
2. Repeat for a specified number of iterations or until a termination criterion is met:
 - Create a set of m ants, each starting at a random node
 - Repeat until all ants have completed their tours:
 - For each ant:
 - Calculate the probability of moving from the current node to a neighboring node using:
$$P(i,j) = (T(i,j)^\alpha \times (\eta(i,j)^\beta)) / \sum((T(i,k)^\alpha \times (\eta(i,k)^\beta))) \text{ for all neighboring nodes } k$$
 - Select the next node to visit based on the calculated probabilities
 - Move the ant to the selected node
 - Evaluate the quality of each ant's solution
 - Update pheromone levels on each edge based on the quality of the solutions found by the ants
 - Evaporate a fraction ρ of pheromone on all edges
3. Return the best solution found

In above algorithm, $T(i,j)$ represents the pheromone level on the edge from node i to node j , and $\eta(i,j)$ is a heuristic value that represents information about the desirability of moving from i to j . The parameters α , β , and ρ control the relative importance of pheromone and heuristic information, as well as the rate of pheromone evaporation.

3.2 PSO

Developed by Kennedy and Eberhart in [24], PSO has become a widely used SI-based algorithm. The PSO algorithm searches the space of an objective function by quasi-stochastically adjusting the trajectories of individual agents. Quasi-stochastically means that the search process is not completely random, but also not completely deterministic. Instead, the search process is guided by a set of rules that determine how the trajectories of individual agents are adjusted. As agents explore the objective space, they share their experience with other agents and the best agents move towards the optimum. This allows the whole population to converge faster to the global best solution [25].

In PSO, each particle represents a possible solution to the problem, and the particles move through the search space in a way that is guided by their own best-known position (p_{best}) and the best-known position of the entire swarm (g_{best}). The PSO algorithm works as follows [26]:

1. Initialization: The first step is to initialize the population of particles. This can be done randomly or by using some other method, such as a genetic algorithm.
2. Velocity and position updating: In each iteration, the velocity and position of each particle is updated according to the following equations:

For each particle i:

Let x_i be the current position of particle i.

Let v_i be the current velocity of particle i.

Let p_i be the best position found by particle i so far.

Let p_g be the best position found by the entire swarm so far.

Let w , c_1 , and c_2 be constants representing the inertia weight, cognitive weight, and social weight, respectively. Then, the velocity and position of particle i are updated as follows:

$$v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (p_i - x_i(t)) + c_2 \times r_2 \times (p_g - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where r_1 and r_2 are random numbers between 0 and 1.

3. Termination: The algorithm terminates when a stopping criterion is met, such as a maximum number of iterations or a certain level of fitness

PSO is a powerful optimization algorithm that can be used to solve a variety of problems. It is simple to implement and understand, and it can be used to solve problems with a large number of variables. However, it can be slow to converge to a solution and it can be sensitive to the choice of parameters [25].

3.3 Artificial Bee Colony

The Artificial Bee Colony (ABC) algorithm, introduced by Drvis Karaboga in 2005 [27], is a metaheuristic optimization technique. This algorithm consists in three groups of artificial bees which, inspired by the foraging behaviour of honeybees, are scouts, observers and employed bees. Scouts engage in a random search for food sources, while employed bees revisit previously discovered food sources. Onlooker bees observe the “dance area” where employed bees perform a “waggle dance,” communicating the location and quality of the food source they have found. In the context of optimization, the number of food sources in the ABC algorithm corresponds to the population size of solutions. A favorable food source’s position indicates a promising solution to the optimization problem, while the nectar quality reflects the fitness cost associated with that solution.

The ABC algorithm can be split into four main steps: Initialization, Population updating, Bee source selection and Population elimination [27]. During initialization, the initial population size (SN) is set, and the first N food sources are randomly generated to populate the population. In the population update phase, employed bees

search for neighboring food sources to potentially improve their currently assigned source. The bee source selection phase involves the movement of employed bees based on the income rate of their associated food sources. Food sources with higher income rates, calculated according to their fitness values, are more likely to be selected by the bees. In the population elimination phase, solutions that do not show significant improvement after continuous updates are assumed to be caught in a local optimum and are eliminated. The corresponding onlooker bees then become scout bees and randomly produce new solutions to replace the eliminated ones [27].

3.4 Firefly (FA)

The basic idea of the FA is that fireflies are attracted to each other's brightness. The brighter a firefly is, the more attractive it is to other fireflies. This means that fireflies will tend to move towards the brightest fireflies in their vicinity [28]. This phenomenon is used by scientists to simulate various optimization problems. By having each firefly represent a potential solution and their brightness representing the fitness of the solution, the FA can be used to find the best solution to a given problem.

In the FA, each firefly represents a possible solution to an optimization problem. The brightness of a firefly is a measure of how good the solution is. The algorithm starts with a population of randomly generated fireflies. The fireflies then move towards the brightest fireflies in their vicinity. This process is repeated until the algorithm converges to a good enough solution. Fireflies of opposite sexes flash to attract each other, but females flaunt to attract males of other populations is a firefly and can flash to attract other fireflies. At each iteration, fireflies are attracted to more appealing fireflies, which they then consume. The movement of a firefly (firefly a) towards another firefly (firefly b) with higher attraction is determined by the following equation:

$$a = a + \beta I(kb - ak)(b - a) + \alpha q \quad (3)$$

where β denotes the source intensity and I represent intensity of the attraction. A random walk component is included as well, where q is drawn from a multivariate Gaussian distribution with zero mean and unit covariance. The step size is scaled by α . Consequently, the update process involves a random walk that is biased towards brighter fireflies. ' k ' is a user-defined constant or parameter that controls the effect of the difference in attractiveness ($kb - ka$) between firefly 'b' and firefly 'a' on the movement of firefly 'a' towards 'b'.

3.5 Grey Wolf Optimizations (GWO)

Grey wolf optimization (GWO) is a metaheuristic optimization algorithm inspired by the leadership hierarchy and hunting mechanism of grey wolves in nature. It was proposed by Seyedali Mirjalili [29]. The basic idea of GWO is that grey wolves are organized in a social hierarchy with four types of wolves: alpha, beta, delta, and omega. The alpha wolf is the leader of the pack, followed by the beta wolf, delta wolf, and omega wolf. The alpha wolf is the strongest and most experienced wolf, and it is responsible for leading the pack in hunting and protecting its territory. The beta wolf is the second-in-command, and it helps the alpha wolf in leading the pack. The delta wolf is the third-in-command, and it helps the alpha and beta wolves in hunting and protecting the pack's territory. The omega wolf is the weakest and least experienced wolf, and it is responsible for tasks such as gathering food and taking care of the young wolves.

In GWO, each wolf represents a possible solution to an optimization problem. The alpha wolf represents the best solution found so far, the beta wolf represents the second-best solution, the delta wolf represents the third-best solution, and the omega wolf represents the fourth-best solution. The algorithm starts with a population of randomly generated wolves. The wolves then move towards the alpha wolf, beta wolf, delta wolf, and omega wolf according to their social hierarchy. This process is repeated until the algorithm converges to a good enough solution.

The grey wolf optimization algorithm consists of the following steps: (1) *Initialize the population*. The population is a set of wolves, each of which represents a possible solution to the optimization problem. The wolves can be initialized randomly or using some other method. (2) *Calculate the fitness of each wolf*. The fitness of a wolf is a measure of how good the solution it represents is. The fitness can be calculated using a fitness function. (3) *Update the positions of the wolves*. The wolves move towards the alpha wolf, beta wolf, delta wolf, and omega wolf according to their social hierarchy. The amount that a wolf moves is determined by a number of factors, including the fitness of the wolf and the distance between the wolf and the alpha wolf, beta wolf, delta wolf, and omega wolf.

1. Initialize a population of N wolves randomly within the search space
2. Calculate the fitness for each wolf in the population
3. Repeat for a specified number of iterations or until a termination criterion is met:
 - Identify the alpha, beta, and delta wolves with the best, second-best, and third-best fitness values
 - For each wolf i in the population:
 - Calculate the distances A, B, and D from wolf i to the alpha, beta, and delta wolves, respectively
 - For each dimension j (1 to D):
 - Generate random numbers r1, r2, and r3 between 0 and 1
 - Update the position of wolf i as follows:

$$X1[i][j] = (X[i][j] + X[\alpha][j])/2 - A * r1$$

$$X2[i][j] = (X[i][j] + X[\beta][j])/2 - B * r2$$

$$X3[i][j] = (X[i][j] + X[\delta][j])/2 - D * r3$$

Ensure that the new positions are within the search space boundaries

Recalculate the fitness for wolf i's new position

Update the alpha, beta, and delta wolves based on their new fitness values

4. Return the best solution found

The above swarm optimization algorithms are compared and tabulated in Table 3.

4 Combining Natural Language Processing and Swarm Intelligence Optimization

NLP is a branch of artificial intelligence emphasizing human computer interface with the help of natural language. It involves major steps such as text classification, sentiment analysis, and machine translation. On the other hand, SIO is a meta-heuristic optimization technique that mimics the collective behavior of social animals to solve complex optimization problems [30]. This study describes a relationship that can be seen in Fig. 4. Table 4 summarizes SWO algorithms and their suitability for handling NLP tasks.

4.1 Pipeline for Swarm Intelligence-Based Optimization Based Natural Language Processing

Combining NLP and SIO can improve NLP applications' performance. As an example, text classification involves categorizing text. To improve classification accuracy, SIO helps identify the optimal set of features. In sentiment analysis, it can also be determined which features are optimal to improve sentiment classification accuracy. Moreover, SIO can be used to extract structured data from text, which can then be used as input to NLP models. This can result in improved accuracy and faster performance. Additionally, SIO can be used to automate text annotation for training NLP models. Figure 5 represented the pipeline for ML/DL based NLP application which consist of following steps:

1. **Corpus preprocessing:** The first step is to pre-process the text corpus by removing stop words, stemming, and converting the text into numerical representation using term frequency-inverse document frequency (TF-IDF) techniques. After preprocessing, the text corpus is ready to be used as input for the machine learning algorithm. The cleaned corpus can then be used to build models that can be used for NLP applications.

Table 3 Comparison of popular swarm optimization algorithm

Algorithm	Key characteristics	Individual representation	Updation policy	Fitness function	Stopping criteria
Ant colony optimization (ACO)	Inspired by the foraging behavior of ants, uses a population of artificial ants to explore the search space	Each ant represents a path through the search space, with each path consisting of a sequence of moves	Each ant updates its path based on a combination of local information and global information	Used to evaluate the quality of each ant's path through the search space	Based on a maximum number of iterations or a minimum level of fitness
Particle swarm optimization (PSO)	Uses a population of particles to explore the search space	Each particle is represented by a position vector in the search space	Each particle updates its position based on its current position, its best position, and the best position of the swarm	Used to evaluate the quality of each particle's position in the search space	Based on a maximum number of iterations or a minimum level of fitness
Artificial bee colony (ABC)	Inspired by the foraging behavior of bees, uses a population of artificial bees to explore the search space	Each bee represents a potential solution to the optimization problem	Each bee updates its position based on local search and global search	Used to evaluate the quality of each bee's position in the search space	Based on a maximum number of iterations or a minimum level of fitness
Grey wolf optimization (GWO)	Inspired by the social hierarchy and hunting behavior of grey wolves, uses a population of wolves to explore the search space	Each wolf represents a potential solution to the optimization problem	Each wolf updates its position based on its own position, the positions of the alpha, beta, and delta wolves (the highest-ranking wolves in the pack), and a random factor	Used to evaluate the quality of each wolf's position in the search space	Based on a maximum number of iterations or a minimum level of fitness
Firefly algorithm (FA)	Inspired by the flashing behavior of fireflies, uses a population of artificial fireflies to explore the search space	Each firefly represents a potential solution to the optimization problem	Each firefly updates its position based on its own position, the positions of other fireflies, and the brightness of the other fireflies	Used to evaluate the quality of each firefly's position in the search space	Based on a maximum number of iterations or a minimum level of fitness

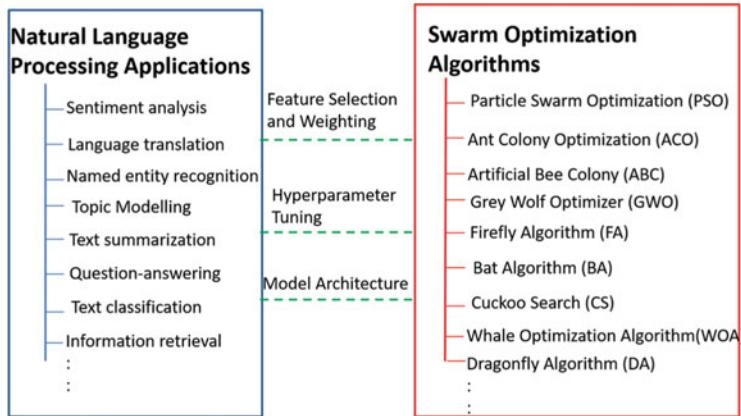


Fig. 4 Natural language processing and swarm optimization algorithm

Table 4 Suitability of SWO algorithms for NLP applications

Swarm optimization algorithm	NLP tasks
Ant Colony Optimization (ACO) [31]	Text Clustering, Keyword Extraction, Information Retrieval
Particle Swarm Optimization (PSO) [24]	Feature Selection, Document classification, Text Summarization
Artificial Bee Colony (ABC) [27]	Sentiment analysis, Text categorization, Feature selection.
Grey Wolf Optimization (GWO) [32]	Word disambiguation, Text Summarization, Topic Modeling
Firefly Algorithm (FA) [33]	Document clustering, Feature selection, Sentiment analysis

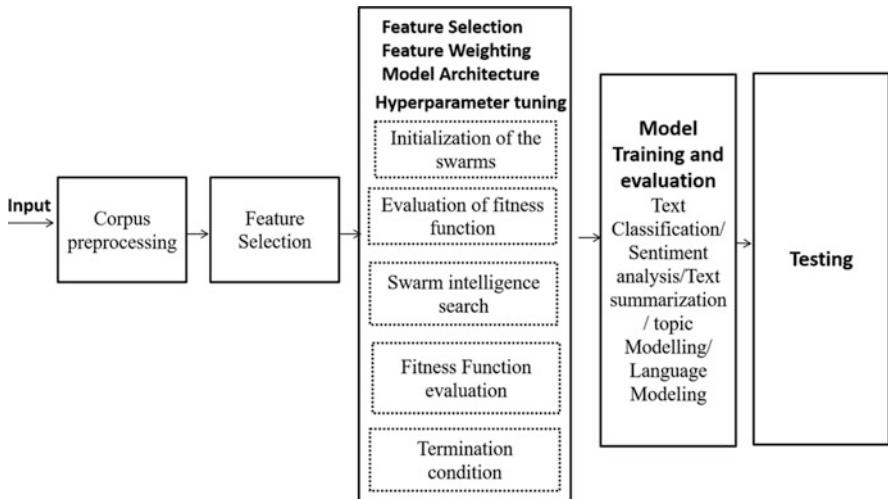


Fig. 5 NLP application pipeline based on swarm optimization

2. **Feature representation:** In this step, the pre-processed text corpus is converted into a matrix of features where each row represents a document and each column represents a feature. The features can be unigrams, bigrams, or n-grams. Then these feature matrices can then be used as input for topic modelling, sentiment analysis, and text classification [34].
 - (a) **Initialization of the swarm:** The swarm is initialized with a set of candidate feature subsets, each consisting of a random subset of the features. The swarm is then optimized to find the best feature subset that yields the best performance for the NLP applications. The optimization process is based on a cost function, which is defined by the performance of the NLP application. The optimization process is repeated until the best feature subset is found.
 - (b) **Evaluation of fitness function:** The fitness function is evaluated for each candidate feature subset. The fitness function typically measures the classification performance of a machine learning algorithm on a subset of the features. The best performing subset is selected as the output, and the process is repeated with the new set of features. The process is repeated until the desired number of features is reached or the performance does not improve any further.
 - (c) **Swarm intelligence search:** The swarm intelligence search algorithm is executed to explore the space of possible feature subsets. The search algorithm typically involves iteratively updating the candidate feature subsets using a combination of local search and global search techniques.
 - (d) **Fitness function evaluation:** After each iteration of the search algorithm, the fitness function is evaluated for the updated candidate feature subsets.
 - (e) **Termination condition:** The search algorithm concludes upon fulfilling a stopping criterion, which can involve reaching a maximum number of iterations or ceasing when there is no further enhancement in the fitness score.
3. **Final feature subset selection:** The feature subset with the highest fitness score is selected as the final feature subset.
4. **Hyperparameter tuning:** Swarm optimization can also be used for hyperparameter tuning, where the hyperparameters of a machine learning algorithm are optimized to achieve the best possible performance. The swarm can be initialized with a set of candidate hyperparameters, and the optimization process can be based on a cost function that measures the performance of the machine learning algorithm on a validation set.
5. **Model architecture:** Swarm optimization can also be used for optimizing the architecture of a machine learning model, such as the number of layers in a neural network or the number of trees in a random forest. The swarm can be initialized with a set of candidate model architectures, and the optimization process can be based on a cost function that measures the performance of the machine learning algorithm on a validation set
6. Swarm optimization algorithms such as Ant Colony Optimization, Particle Swarm Optimization, and Bee Colony Optimization can be used for NLP tasks

such as feature selection [35] such as ACO [36], FA [37] and weighting, hyperparameter tuning [38], and model architecture. The choice of algorithm will depend on the specific problem being solved and the characteristics of the data.

4.2 *Hyperparameter Tuning with Swarm Optimization*

Swarm optimization techniques, such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), can be used to tune NLP applications. These techniques can be used to improve the performance of NLP tasks such as text classification, sentiment analysis, and machine translation. Table 5 shows a comparison of PSO-based NLP tasks.

In NLP hyperparameter tuning, the search space would consist of all possible combinations of hyperparameters. Based on the performance of the NLP model with the corresponding hyperparameters, the swarm optimization algorithm updates the position of the agents iteratively in the search space according to their fitness. As the algorithm runs, a stopping criterion, such as a maximum number of iterations or a satisfactory level of performance, would be met.

Table 5 Comparison of PSO based NLP task

NLP task	Study	Swarm optimization technique	Advantages	Disadvantages
Sentiment classification and QA tasks	Aghaebrahimian and Cieliebak [39]	Particle swarm optimization	Can effectively search high-dimensional and non-convex search spaces, can handle noisy and incomplete data, easily parallelizable	Can get stuck in local optima, sensitive to initialization, requires careful parameter tuning
Multi-label text classification	Reimers and Gurevych [40]	Particle swarm optimization	Can handle high-dimensional and non-convex search spaces, can handle noisy and incomplete data, easily parallelizable	Can get stuck in local optima, sensitive to initialization, requires careful parameter tuning
Machine translation	Vaswani et al. [41]	Particle Swarm optimization	Can effectively search high-dimensional and non-convex search spaces, can handle noisy and incomplete data, easily parallelizable	Can get stuck in local optima, sensitive to initialization, requires careful parameter tuning

4.3 Swarm Intelligence Based Pretrain Language Model

A swarm-based optimization technique could be employed to improve the performance of pre-trained language models in NLP. Among the models that have been trained on large amounts of text data are BERT (Bidirectional Encoder Representations from Transformers), GPT-3 (Generative Pre-trained Transformer 3) RoBERTa (Robustly Optimized BERT pretraining approach) and ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately). The models are trained unsupervised to predict missing words or produce text based on a prompt. Once trained, these models can be fine-tuned to perform specific NLP tasks, such as text classification, question answering, or language translation. Nevertheless, these models can be fine-tuned or optimized to improve their performance on specific tasks.

Particle Swarm Optimization (PSO) can be effectively applied in NLP applications. For instance, an approach called hybrid swarm intelligence linguistic feature selection (HSI-LFS) has been proposed. This method utilizes PSO, along with Dragonfly Optimization (DO) and Grey Wolf Optimization (GWO) algorithms, to extract a combined feature set. The HSI-LFS approach, when combined with BERT, has shown promise in predicting Alzheimer's disease using audio transcript data [42]. Swarm-based optimization techniques can also be employed to optimize the hyperparameters of pretrained language models. This includes fine-tuning parameters such as the learning rate, batch size, and number of training epochs. Using these techniques makes it possible to identify optimal hyperparameter values that lead to enhanced performance on specific NLP tasks.

Another potential application of swarm-based optimization techniques in NLP is to improve the interpretability of pre-trained language models. Pre-trained language models are often criticized for their lack of interpretability, which can make it difficult to understand how the models are making predictions. Swarm-based optimization techniques can be used to identify important features or substructures in the input data that are relevant to the model's predictions. This can help improve the model's interpretability and provide insights into how the model is making predictions (Table 6).

4.4 Swarm Intelligence in Transfer Learning-Based Language Processing

Transfer learning is the process of using knowledge acquired from one task to improve performance on another. In NLP, transfer learning involves training a model on a large corpus of text data and then using the learned representations to improve performance on a downstream NLP task. It can be accomplished by fine-tuning the pre-trained model on the target task or by using the learned representations as features for a separate classifier. Transfer learning can be applied to a wide

Table 6 Comparison of pretrained language model and transfer learning based model

Criteria	Pre-trained language models	Transfer learning-based models
Input Data	Large corpus of raw text data	Smaller corpus of annotated text data
Training	Unsupervised learning	Supervised learning
Model Complexity	Very large and complex	Smaller and simpler than pre-trained models
Fine-tuning	Additional training on task-specific data	Training on task-specific data
Performance	State-of-the-art on a wide range of NLP tasks	Performance depends on the quality and size of the annotated data
Examples	GPT-3, BERT, RoBERTa, ELECTRA, GPT-2	ULMFiT, ELMo, OpenAI GPT, BERT, RoBERTa
Libraries	Hugging Face, TensorFlow, PyTorch	Keras, TensorFlow, PyTorch
Use cases	General NLP tasks, text generation, summarization, language translation, etc.	Specific NLP tasks, such as sentiment analysis, named entity recognition, text classification, etc.

range of NLP tasks, including text classification, question answering, language translation, and more [6].

Swarm optimization techniques can be combined with transfer learning to improve the performance of NLP models. Better performance on downstream tasks can be achieved by using SI algorithms to optimize hyperparameters or fine-tune pre-trained models on specific tasks. Additionally, SI algorithms can identify important features or patterns in the input data, improving the interpretability of pre-trained models.

In NLP applications, swarm optimization techniques can be used to fine-tune pre-trained language models like BERT or GPT to improve their performance on specific tasks or datasets. Fine-tuning involves adjusting hyperparameters of the pre-trained model to optimize its performance on the new task or dataset. Swarm optimization techniques, such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), can be used to find an optimal set of hyperparameters, leading to improved performance of the pre-trained language model. Advantages of using swarm optimization for NLP include the ability to optimize complex models with large hyperparameter spaces, the potential for improved interpretability of models, and the ability to improve the performance of pre-trained models on new tasks or datasets. However, swarm optimization can also be computationally expensive and time-consuming, and may require significant computational resources.

4.5 *Swarm Intelligence and Large-Scale Language Model*

Swarm intelligence (SI) techniques can be useful for large-scale language models (LSLMs) in several ways:

- Optimization of hyperparameters: LSLMs often have a large number of hyperparameters that need to be optimized to achieve optimal performance. SI algorithms, such as particle swarm optimization (PSO) and ant colony optimization (ACO), can efficiently search the hyperparameter space and identify the optimal set of hyperparameters.
- Model compression: LSLMs can be very large and computationally expensive, making them difficult to deploy on resource-constrained devices. SI algorithms can compress the model by identifying the most critical parameters and removing the less important ones, while preserving the model's overall performance.
- Distributed training: Training LSLMs on large datasets can be time-consuming and computationally expensive. SI algorithms, such as swarm-based distributed optimization, can speed up the training process by distributing the computation across multiple nodes or GPUs.
- Data selection: LSLMs are typically trained on large, diverse datasets. However, not all the data may be relevant or useful for a particular task. SI algorithms can be used to select the most informative and representative subset of the data for training, which can help improve the model's performance.

Interpretability: LSLMs can be difficult to interpret because of their complexity and the large number of parameters. SI algorithms can be used to identify the most important features or patterns in the input data.

5 Use Case or Application Where Integration of SI and NLP Can Be Beneficial

SI and NLP technologies can be integrated to benefit a wide range of industries and domains, automating processes, extracting insights from unstructured data, and improving user experience. For example, Integrating SI and NLP technologies can enable healthcare providers to extract valuable insights from unstructured medical records, clinical notes, and other healthcare data. These documents can be analysed with NLP techniques, while SI can be used to integrate this information with electronic health records (EHRs) and other healthcare systems. Fraud detection is another example of NLP using topic modeling. Fraud detection can use topic modeling to identify topics related to fraudulent activities, such as money laundering, identity theft, or credit card fraud. By analyzing text data associated with financial transactions, topic modeling algorithms can identify patterns of suspicious behavior that may be indicative of fraud. By analyzing financial transaction data, SI algorithms can identify patterns of behavior that might indicate fraud. Many industries, including finance, insurance, and e-commerce, require fraud detection. It can be improved using natural language processing (NLP) and swarm intelligence (SI). The following sections describe a basic NLP application that can be combined with swarm intelligence.

5.1 Sentiment Analysis with SI

The sentiment analysis literature uses swarm optimization to select and optimize features [43]. A study proposes a Grid-based Adaptive Many-Objective Grey Wolf Optimizer (GAM-GWO) for sentiment analysis based on the Grey Wolf Optimizer algorithm. With the aim of improving sentiment classification accuracy on the sentiment labeled sentences benchmark dataset [2], another study proposes a metaheuristic-based approach to selecting a feature subset through the binary particle swarm optimization (BPSO) metaheuristic algorithm.

In sentiment analysis, swarm optimization has been used to select features. As an example, one study [44] presents a metaheuristic-based approach to selecting feature subsets with the binary particle swarm optimization (BPSO) metaheuristic algorithm. The aim is to improve sentiment classification accuracy on the sentiment labelled sentences benchmark dataset. In another study, support vector machine (SVM) is used as the classifier along with particle swarm optimization (PSO) and Information Gain [2].

The method proposed by Akhtar et al. [45] uses particle swarm optimization (PSO) to select features and construct ensembles. As part of their proposed work, three basic classifiers are used to determine the best fitting feature sets for aspect term extraction and sentiment classification, including Conditional Random Fields (CRF), Support Vector Machines (SVMs), and Maximum Entropy (MEs). PSO-based feature selection is used to select a set of solutions for each classifier. In the final step, the ensemble selection algorithm determines the best set of classifiers to improve classification performance by combining the outputs of the classifiers using PSO (Fig. 6).

5.2 Social Media Analytics with SI

Social media analytics in the context of natural language processing (NLP) refers to the process of extracting insights and information from social media data using NLP techniques. NLP can be used to analyse conversations, sentiment, and intent and identify trends, topics, and keywords in social media data. This allows businesses to gain insights into their customers' preferences and behavior, allowing them to tailor their offerings accordingly. One potential use case for swarm intelligence in this

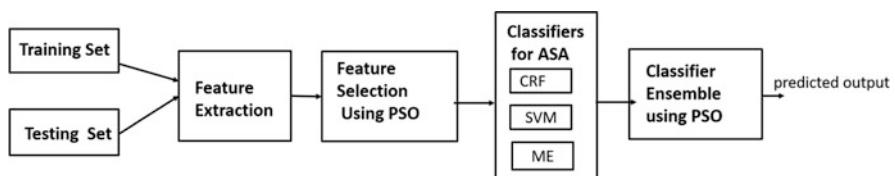


Fig. 6 Aspect based sentiment analysis using particle swarm optimization

context is to identify clusters of social media activity related to the conflict. This could involve using an algorithm like particle swarm optimization or ant colony optimization to identify groups of related posts based on various criteria, such as their proximity in time or space, the language used in the posts, or the topics being discussed.

Swarm optimization algorithms, such as particle swarm optimization (PSO), can be used for social media text analytics. For example, a study introduced a sigmoidal particle swarm optimization (SPSO) algorithm for analyzing individual tweets about COVID-19 vaccines. The SPSO algorithm was first tested on a set of benchmark problems and then used for selecting optimal text features and categorizing sentiment [46]. Within the domain of social network analysis, community detection involves partitioning a network into distinct clusters. The objective is to create clusters so that individuals within each cluster engage in more frequent interactions with one another compared to interactions across different clusters [47]. One study used an ACO-based approach to detect communities in social networks. Comparing their ACO-based approach to existing heuristics in the literature, the authors demonstrated significant improvements in modularity values [48].

5.3 Topic Modeling with SI

The topic modeling approach is used to identify latent topics within a collection of documents, where each topic corresponds to a probability distribution over the words in the vocabulary. A “topic” is viewed as a probability distribution over a fixed vocabulary. For instance, in a corpus of documents related to sports, a topic might be associated with words such as “team”, “goal”, and “player” with higher probabilities than words such as “book”, “read”, and “theater”. There are several popular algorithms for topic modeling in natural language processing, including:

1. Latent Dirichlet Allocation (LDA): LDA is a generative probabilistic model representing documents as mixtures of latent topics.
2. Non-negative Matrix Factorization (NMF): NMF is a matrix decomposition technique representing documents as a linear combination of non-negative basis vectors, which can be interpreted as latent topics.
3. Hierarchical Dirichlet Process (HDP): HDP is a Bayesian nonparametric model that extends LDA by allowing for an unbounded number of topics.
4. Probabilistic Latent Semantic Analysis (PLSA): PLSA is a generative probabilistic model that is similar to LDA but does not include a prior distribution over the topics.
5. Correlated Topic Model (CTM): CTM is an extension of LDA that allows for correlations between the topics.

Swarm optimization can be used to optimize the parameters of topic models, such as Latent Dirichlet Allocation (LDA), a popular probabilistic model used in NLP. In addition to optimizing the number of topics, the hyperparameters of the prior

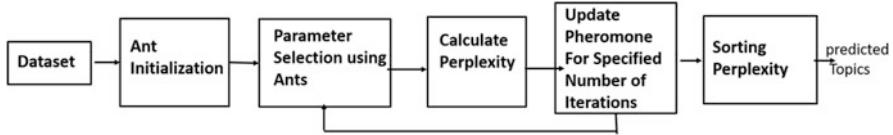


Fig. 7 Topic modeling using ACO

distributions, and the posterior distribution, swarm optimization can also optimize the hyperparameters of the prior distributions.

Optimization techniques, such as PSO and ACO are particularly effective in optimizing the number of topics in LDA. The objective function measuring the quality of the topic model, such as the coherence between words in the topic model, has been optimized using PSO. Based on the objective function, ACO was used to choose the best set of topics for the topic model. Generally, as a probabilistic model representing documents as mixtures of latent topics, the Latent Dirichlet Allocation (LDA) topic model can be optimized using ACO. As a result of using ACO for topic modeling, the task of choosing the best set of topics is treated as a combinatorial optimization problem, where the goal is to find the set of topics that maximize a given objective function. Depending on the topic model, an objective function can be used to measure the coherence or quality of the topics, or it can be used to measure the semantic similarity between the topics and words. For Instance, Yarnguy et al. [49] propose a method for tuning the parameters of Latent Dirichlet Allocation (LDA) using Ant Colony Optimization (ACO). LDA is a technique that represents documents as mixtures of latent topics (The architecture is depicted in Fig. 7. It requires the specification of several hyperparameters, such as the number of topics and the concentration parameters of the prior distributions. ACO optimizes the hyperparameters of LDA based on the perplexity, which is a standard for measuring topic model performance. Using the ACO algorithm, candidate solutions are constructed by iteratively adding or removing topics from the model and adjusting concentration parameters, guided by a pheromone trail indicating the solutions' quality.

6 Challenges and Future Directions

The future direction of Swarm Intelligence (SI) for Natural Language Processing (NLP) holds significant potential for advancing both fields. Yang et al. [50] extensively survey SI optimization techniques applied to NLP. These techniques are poised for further development and specialization to address specific NLP tasks [51].

Developing SI algorithms customized to specific NLP tasks is a promising avenue. While current SI techniques have shown success, there remains room for optimization. For instance, researchers can create novel SI algorithms specialized in text classification, machine translation, question answering, and text generation [51]. Such specialization can lead to more efficient and effective solutions.

As SI and established machine learning methods, such as deep learning, converge, there is potential for enhancement [50]. SI's capacity for exploration and exploitation can be combined with deep learning's representation learning to design new algorithms that improve NLP system performance. For example, researchers could combine SI with deep learning to develop new algorithms for text classification, machine translation, question answering, and text generation.

As discussed in Zhang, Chen, and Liu's review, SI can be extended to novel NLP domains and applications [17]. This includes areas like dialogue systems, sentiment analysis, and natural language inference. For instance, researchers could leverage SI to construct dialogue systems that comprehend and respond to human language more effectively, analyze sentiment in text comprehensively, and deduce natural language inferences with greater accuracy.

The scalability of SI algorithms is crucial, especially for large-scale NLP tasks such as machine translation on extensive text corpora or question answering over vast knowledge bases. Research efforts can be directed towards creating SI algorithms capable of training and optimizing large language models efficiently.

Real-time NLP applications, like translating live audio or summarizing dynamic news feeds, present a unique set of challenges [52]. Future research can focus on developing SI algorithms adept at handling real-time data streams, including translating video content and summarizing social media posts in real-time. Integration of linguistic context effectively into the optimization process is a key challenge. Handling large-scale text data efficiently is a pressing concern. Researchers should explore techniques such as parallelization, distributed computing, and hybrid approaches to improve the scalability and efficiency of SI for NLP.

Many real-world scenarios involve multimodal data, including images, audio, and video. Integrating NLP and SI can extend to multimodal optimization problems. This provides comprehensive optimization capabilities across different modalities, aligning with the need to address multimodal data. There is potential for adapting SI to different domains with different linguistic characteristics [17]. Future research should focus on domain-specific language models and adapted SI algorithms for effective optimization across various domains.

Meeting the demands of dynamic environments and real-time data streams is crucial [52]. Future research should explore incremental optimization algorithms, adaptive learning strategies, and online learning frameworks in the context of NLP and SI integration. To drive progress, standardized benchmark datasets and evaluation metrics are essential. Future research should prioritize the creation of representative datasets and evaluation metrics that assess linguistic quality and optimization performance.

7 Conclusion and Summary

Integrating Natural Language Processing (NLP) and Swarm Intelligence Optimization (SIO) techniques offers immense potential for addressing complex problems in various fields. Through a comprehensive examination of the theoretical foundations,

methodologies, and practical applications, this book chapter has shed light on the rich possibilities that how NLP and SIO can be synergistically combined to enhance the accuracy, efficiency, and scalability of various applications such as sentiment analysis, social media analytics, topic modeling, document classification, machine translation, and information retrieval. This book chapter explores the transformative impact of Natural Language Processing (NLP) techniques in understanding and processing human language. It discusses various NLP techniques highlighting their applications in diverse industries and the potential of ongoing advancements in NLP techniques. This book chapter introduces swarm intelligence (SI), a collective behavior inspired by swarms found in nature. It provides an overview of SI and its applications in diverse fields. The chapter classifies optimization algorithms, explores a timeline of SI algorithms, and offers comparisons between different SI algorithms. It highlights the potential of SI in solving complex problems and inspiring further research.

Further, the integration of swarm intelligence (SI) and natural language processing (NLP) is explored in various applications, highlighting their benefits. It presents a pipeline for SI-based optimization in NLP tasks, including hyperparameter tuning to enhance NLP models. It also discusses the application of SI in developing pretrain language models, leveraging the collective behavior of swarms to improve language understanding and generation. The integration of SI in transfer learning for language processing, enabling knowledge transfer and improving performance across different NLP tasks is also explored. The chapter delves into the potential of SI in developing large-scale language models, highlighting how swarm intelligence can address scalability challenges and enhance the efficiency of processing vast amounts of textual data.

The chapter discusses the use of SI in sentiment analysis, where the collective behavior of swarms can enhance the accuracy and efficiency of sentiment classification tasks. Additionally, the chapter explores how SI can be applied in social media analytics, enabling the extraction and analysis of valuable insights from large volumes of social media data. Furthermore, the chapter delves into topic modeling with SI, leveraging swarm intelligence to discover meaningful topics and improve the accuracy of topic modeling algorithms.

Overall, the chapter emphasized that the collaboration between NLP and SIO opens up a vast spectrum of possibilities for advancing both fields. The seamless integration of language understanding and optimization capabilities can lead to innovative solutions in various domains, including artificial intelligence, robotics, data analysis, and decision-making systems. In conclusion, the comprehensive overview presented in this book chapter underscores the transformative power that arises from synergizing Natural Language Processing and Swarm Intelligence Optimization. By leveraging the strengths of each domain and exploring their interplay, researchers and practitioners can unlock new frontiers in intelligent systems and pave the way for future breakthroughs in artificial intelligence.

References

1. Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. *IEEE Comput Intell Mag.* 2018;13(3):55–75.
2. Botchway RK, Yadav V, Komíková ZO, Senkerik R. Text-based feature selection using binary particle swarm optimization for sentiment analysis. In: 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET). 20 Jul 2022. p. 1–4.
3. Yildirim G. A novel grid-based many-objective swarm intelligence approach for sentiment analysis in social media. *Neurocomputing.* 2022;503:173–88.
4. Yang XS. Swarm intelligence-based algorithms: a critical analysis. *Evol Intel.* 2014;7:17–28.
5. Ali YA, Awwad EM, Al-Razgan M, Maarouf A. Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *PRO.* 2023;11(2):349.
6. Malte A, Ratadiya P. Evolution of transfer learning in natural language processing. 2019; arXiv preprint arXiv:1910.07370.
7. Yin W, Kann K, Yu M, Schütze H. Comparative study of CNN and RNN for natural language processing. 2017; arXiv preprint arXiv:1702.01923.
8. Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv 2018:1810.04805.
9. Kumar Gupta D, Srikanth Reddy K, Ekbal A. Pso-asent: feature selection using particle swarm optimization for aspect based sentiment analysis. In: Natural Language Processing and Information Systems: 20th International Conference on Applications of Natural Language to Information Systems, NLDB 2015, Passau, Germany, June 17–19, 2015, Proceedings 20. Springer International Publishing; 2015. p. 220–233.
10. Janani R, Vijayarani S. Automatic text classification using machine learning and optimization algorithms. *Soft Comput.* 2021;25:1129–45.
11. Neubig G, Watanabe T. Optimization for statistical machine translation: a survey. *Comput Linguist.* 2016;42(1):1–54.
12. Nazari N, Mahdavi MA. A survey on automatic text summarization. *J AI Data Mining.* 2019;7 (1):121–35.
13. Sarmah DK. A survey on the latest development of machine learning in genetic algorithm and particle swarm optimization. In: Kulkarni A, Satapathy S, editors. Optimization in machine learning and applications. Singapore: Springer; 2020. p. 91–112.
14. Bai X, Gao X, Xue B. Particle swarm optimization based two-stage feature selection in text mining. In: 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2018. p. 1–8.
15. Hutchins WJ, Dostert L, Garvin P. The Georgetown-IBM experiment. *Mach Transl Lang.* 1955;124–35.
16. Beattie JD. Natural language processing by computer. *Int J Man-Mach Stud.* 1969;1(3):311–29.
17. Zhang L, Yang Y. Towards sustainable energy systems considering unexpected sports event management: integrating machine learning and optimization algorithms. *Sustainability.* 2023;15(9):7186.
18. Seeley TD. The wisdom of the hive: the social physiology of honey bee colonies. *Perspect Biol Med.* 1997;40(2):303.
19. Engelbrecht AP. Fundamentals of computational swarm intelligence. Chichester: Wiley; 2005.
20. Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comput.* 1997;1(1):53–66.
21. Kennedy J, Mendes R. Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 2). 12 May 2002. p. 1671–76.
22. Dorigo M, Maniezzo V, Colomi A. Ant system: optimization by a colony of cooperating agents. *IEEE Trans Syst Man Cybernet Part B Cybernet.* 1996;26(1):29–41.
23. Flórez E, Gómez W, Bautista L. An ant colony optimization algorithm for job shop scheduling problem. 2013 Sep 19; arXiv preprint arXiv:1309.5110.

24. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 4). 27 Nov 1995. p. 1942–48.
25. Wang D, Tan D, Liu L. Particle swarm optimization algorithm: an overview. *Soft Comput.* 2018;22:387–408.
26. Agarwal K, Kumar T. Email spam detection using integrated approach of Naïve Bayes and particle swarm optimization. In: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). 14 Jun 2018. p. 685–90.
27. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J Glob Optim.* 2007;39:459–71.
28. Yang XS, He X. Firefly algorithm: recent advances and applications. *Int J Swarm Intell.* 2013;1(1):36–50.
29. Mirjalili S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput Applic.* 2016;27:1053–73.
30. Bonabeau E. Agent-based modeling: methods and techniques for simulating human systems. *Proc Natl Acad Sci.* 2002;99(Suppl_3):7280–7.
31. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. *IEEE Comput Intell Mag.* 2006;1(4):28–39.
32. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Adv Eng Softw.* 2014;69:46–61.
33. Yang XS. Firefly algorithm, stochastic test functions and design optimization. *Int J Bio-inspired Comput.* 2010;2(2):78–84.
34. Nguyen BH, Xue B, Zhang M. A survey on swarm intelligence approaches to feature selection in data mining. *Swarm Evolut Comput.* 2020;54:100663.
35. Sharma M, Kaur P. A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. *Arch Comput Methods Eng.* 2021;28:1103–27.
36. Aghdam MH, Ghasem-Aghaee N, Basiri ME. Application of ant colony optimization for feature selection in text categorization. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). 1 Jun 2008. p. 2867–73.
37. Banati H, Bajaj M. Fire fly-based feature selection approach. *Int J Comput Sci Issues.* 2011;8(4):473.
38. Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization. *Inf Sci.* 2012;192:120–42.
39. Aghaebrahimian A, Cieliebak M. Hyperparameter tuning for deep learning in natural language processing. In: 4th Swiss Text Analytics Conference (Swisstext 2019), Winterthur, June 18–19 2019. 2019. Swisstext.
40. Reimers N, Gurevych I. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. 2017 Jul 21; arXiv preprint arXiv:1707.06799.
41. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. *Adv Neural Inf Proces Syst.* 2017;30.
42. Khan YF, Kaushik B, Rahmani MK, Ahmed ME. HSI-LFS-BERT: novel hybrid swarm intelligence based linguistics feature selection and computational intelligent model for Alzheimer's prediction using audio transcript. *IEEE Access.* 2022;10:126990–7004.
43. Sankar S. Sentiment analysis and deep learning based chatbot for user feedback. In: Intelligent communication technologies and virtual mobile networks: ICICV 2019. Cham: Springer International Publishing; 2020. p. 231–7.
44. Kurniawati I, Pardede HF. Hybrid method of information gain and particle swarm optimization for selection of features of SVM-based sentiment analysis. In: 2018 International Conference on Information Technology Systems and Innovation (ICITSI). 22 Oct 2018. p. 1–5.
45. Akhtar MS, Gupta D, Ekbal A, Bhattacharyya P. Feature selection and ensemble construction: a two-step method for aspect-based sentiment analysis. *Knowl-Based Syst.* 2017;125:116–35.
46. Kumar S, Khan MB, Hasanat MH, Saudagar AK, AlTameem A, AlKhathami M. Sigmoidal particle swarm optimization for Twitter sentiment analysis. *Comput Mater Continua.* 2023;1:897–914.

47. Javadi SH, Khadivi S, Shiri ME, Xu J. An ant colony optimization method to detect communities in social networks. In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014). 17 Aug 2014. p. 200–203.
48. Mandala SR, Kumara SR, Rao CR, Albert R. Clustering social networks using ant colony optimization. *Oper Res.* 2013;13:47–65.
49. Yarnguy T, Kanarkard W. Tuning latent Dirichlet allocation parameters using ant colony optimization. *J Telecommun Electron Comput Eng.* 2018;10(1–9):21–4.
50. Yang X, Xiao Y, Zhang Y. Swarm intelligence optimization for natural language processing: a survey. *Neurocomputing.* 2023;542:184–200.
51. Li X, Wang X, Gao Y. Swarm intelligence optimization for text classification: a comparative study. *IEEE Trans Syst Man Cybernet Syst.* 2022;52(8):8714–27.
52. Wu Y, Wang X, Zhang Y. Swarm intelligence optimization for question answering: a reinforcement learning approach. *IEEE Trans Neural Netw Learn Syst.* 2020;31(11):4355–67.
53. Balakumar J, Mohan SV. Artificial bee colony algorithm for feature selection and improved support vector machine for text classification. *Inform Discov Deliv.* 2019;47(3):154–70.
54. Karaboga D, Gorkemli B, Ozturk C, Karaboga N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artif Intell Rev.* 2014;42:21–57.
55. Peška L, Tashu TM, Horváth T. Swarm intelligence techniques in recommender systems—a review of recent research. *Swarm Evolut Comput.* 2019;48:201–19.
56. Wang Y, Li X, Zhang Y. Swarm intelligence optimization for text summarization: a hybrid approach. *Int J Comput Intell Syst.* 2020;13(1):1035–44.
57. Yu T, Zhu H. Hyper-parameter optimization: a review of algorithms and applications. *arXiv preprint arXiv* 2020:2003.05689.

Heuristics-Based Hyperparameter Tuning for Transfer Learning Algorithms



Upendra Pratap Singh, Krishna Pratap Singh, and Muneendra Ojha

Abstract Hyperparameters play a crucial role in controlling the learning process, consequently impacting the model performance significantly. In most machine and deep learning paradigms, these hyperparameters must be set explicitly with limited heuristics making the hyperparameter fine-tuning process computationally expensive. This computational complexity gets further aggravated in the transfer learning paradigm involving complex source domain models. This work uses particle swarm optimization for hyperparameter fine-tuning within the transfer learning framework. Specifically, dictionary learning-based self-taught and zero-shot classifiers are proposed for classifying lung diseases and recognizing novel concepts. The hyperparameters in these classifiers are obtained using the particle swarm optimizer by equating the corresponding loss functions to the optimizer's fitness function; model parameters are obtained as a corollary to this optimization. The proposed self-taught learner with parameters and hyperparameters obtained by particle swarm optimizer achieves a classification accuracy of up to 95.27% on the Covid-19 Lungs CT Scan dataset. The recognition results with the proposed zero-shot learners are 14.5% and 8.9% in the conventional and generalized settings, respectively. Therefore, these results confirm the usefulness of particle swarm optimization in obtaining optimal model parameters and hyperparameters efficiently.

Keywords Zero-shot learning · Self-taught learning · Transfer learning · Hyperparameter optimization · Evolutionary algorithms · Particle swarm optimization · Meta-heuristics

U. P. Singh

Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur, Rajasthan, India

e-mail: lalupendrapratap.singh@lnmiit.ac.in

K. P. Singh (✉) · M. Ojha

Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj, Uttar Pradesh, India

e-mail: kpsingh@iiita.ac.in; muneendra@iiita.ac.in

1 Introduction

State of the art machine and deep learning-based approaches surpass human capabilities in domains like bio-informatics, natural language processing, computer vision and speech technology [1–11]. These methods thrive on discovering useful patterns in a large corpus of labelled and unlabelled datasets and using these patterns to solve tasks like image recognition, object detection, named-entity recognition, sentiment analysis, drug discovery and speech recognition [1–11].

However, these approaches remain extremely data-hungry, and their effectiveness relies heavily on the dataset's quantity and quality. Moreover, supervised learning-based models require accurate annotation of the instances; since an expert manually annotates the instances, deployment of these models has a bottleneck regarding the time required for this manual annotation [1]. On the other hand, application domains like remote sensing, military and healthcare are low-resource environments characterized by limited training corpus [12]. Specifically, these domains may have small datasets or be unwilling to use the data for training state-of-the-art models. Consequently, even state-of-the-art machine and deep learning models need more usability in these domains.

The limited performance of these models in low-resource environments can be improved using transfer learning-based approaches [12, 13]. These approaches gain knowledge in a related domain by solving tasks in this domain and leverage this knowledge to build generalizable models in low-resource environments. The domain used to obtain knowledge is the source domain, and the one in which this knowledge is eventually used, i.e., the domain featuring this low-resource environment, is the target domain. In other words, the knowledge obtained in the source domain is utilized to compensate for the limited knowledge in the target domain. The relative similarity of source and target domain governs the applicability and usability of transfer learning-based approaches [12]. Moreover, existing transfer learning algorithms are classified into different settings depending on the size of labelled information in the target domain. Specifically, if the target domain has numerous labelled instances, the setting corresponds to self-taught learning [12]; zero-shot learning (ZSL) involves training a model to recognize novel concepts in the target domain [12]. Both self-taught and zero-shot models are increasingly used in applications like computer vision, healthcare and remote sensing. However, training these models is computationally intensive as these models are parameterized by millions of parameters; the computational complexity of these models is further aggravated by the need to fine-tune numerous hyperparameters. Consequently, increasing attention is being given to training linear transfer learning-based models [13]; these linear models have lesser trainable parameters and the transformations involved from input to the output space are also straightforward.

Evolutionary algorithms are nature-inspired algorithms designed to solve an optimization problem by mimicking the behaviour of different natural entities [14]. Consequently, they are increasingly popular in optimization, control processes and machine learning; these algorithms do not rely on gradient information to

perform optimization [14–16]. Therefore, these approaches are favoured for minimizing loss functions that are discrete and non-differentiable; additionally, these approaches can obtain model hyperparameters during the optimization process. However, limited efforts have been made towards fine-tuning hyperparameters while learning the model parameters using evolutionary algorithms, particularly in the transfer learning setting [15–17].

This work proposes transfer learning-based approaches in the healthcare and computer vision domain; specifically, a dictionary learning-based self-taught learning model is proposed for classification of lung diseases using radiographs. Unsupervised feature learning proceeds using source domain data and useful feature representations are learned as a source dictionary; thereafter, the transformation (or the source dictionary) modifies the labelled target instances. These transformed target domain instance-label pairs are then used for the supervised classification. A dictionary learning-based zero-shot learner is also proposed to recognize novel image concepts. Specifically, dictionaries are learned in the two domains separately, and the dictionary learned in the source domain maps the instances from their feature space to their embedding space. The learning of source dictionaries is regularized via a nuclear norm term to yield a low-ranked and robust dictionary. Further, a combination of the target dictionary and target embedding matrix is obtained using target domain data; the low-ranked source dictionary obtained using source domain data regularizes the dictionary obtained in the target domain so that the two dictionaries remain close to each other and capture similar mapping from feature to embedding space. Lastly, the classification performance of the proposed learner is reported in the conventional and generalized setting. The proposed self-taught and zero-shot models also have different regularization hyperparameters that need to be explicitly set; in the proposed work, the parameter learning and fine-tuning of model hyperparameters occur simultaneously using particle swarm optimization (PSO). Specifically, the learnable parameters and hyperparameters are vectorized into a single solution vector, and the optimization algorithm computes improved estimates of these trainables and hyperparameters with each iteration. Once the optimization algorithm terminates, the trainables and hyperparameters can be segregated from each other. To summarize, this research contributes in the following ways:

- (i) A dictionary learning-based self-taught model is proposed for recognizing lung diseases using patients' radiographs.
- (ii) A dictionary learning-based robust zero-shot learner is proposed for recognizing novel image concepts. The performance of this robust classifier is reported in the generalized setting as well.
- (iii) The different trainable parameters and hyperparameters in the proposed self-taught and zero-shot models are estimated using PSO.

The rest of the chapter is organized as follows: Sect. 2 compiles the related works in context to self-taught, zero-shot and evolutionary computation and a general introduction to evolutionary algorithms is provided in Sect. 3. The mathematical background of the PSO algorithm is provided in Sect. 4, and Sect. 5 covers the fundamental aspects and categorization of transfer learning algorithms. The results

obtained with the self-taught learner are reported in Sect. 6; the corresponding results with the zero-shot learner are put in Sect. 7. Finally, some takeaways from the experiments have been provided in Sect. 8.

2 Related Works

Self-taught learning models learn rich feature representations using ample unlabelled instances in the source domain; the feature transformation in the source domain then transforms the limited labelled target instances in the target domain [13]. The knowledge contained in the limited labelled instances in the target domain gets compensated by rich feature representations obtained in the source domain. Consequently, self-taught models are increasingly used in domains like natural language processing, computer vision, remote sensing, healthcare and social network analytics as it is feasible to collect ample unlabelled instances in the source domain [18–26]. However, training self-taught learning-based models has its challenges; specifically, there are fewer restrictions on the source domain data distribution and consequently, outliers and noise can creep in very easily. Further, the non-availability of target domain instances in the source domain restricts regularization in the latter domain. Although self-taught models are extensively reported in the literature, limited attention has been given to exploiting dictionary-based loss functions; these methods are simple to interpret, implement and exhibit nice convergence properties [25, 26]. Moreover, since the source domain is massive in size, efforts are required to speed up the unsupervised feature learning in the source domain [13].

Zero-shot learners are broadly categorized into instance-based and classifier-based methods depending on the entity being learned [27]. Out of these class of methods, instance-based methods are more popular compared to the latter and are further categorized into projection-based, instance-borrowing based and synthesizing methods [27]. Projection-based methods are extremely popular in the literature and involve learning projections in both the domains and using specific regularizations to capture specific structures in the data manifold [12]. Consequently, these methods are extremely popular and are primarily based on semantic autoencoders, using embeddings in the transductive setting, joint learning of source and target parameters and exploiting intermediate manifold structures [27–34]. The methods based on learning projections attempt to minimize the projection-domain shift between the two set of classes so that unbiased projections are learned. However, despite the popularity of projection-based methods, limited attention has been given towards minimizing the projection domain shift using dictionary learning-based methods [35]; further, out of efforts made in this direction, little attention has been given towards exploiting low-rank dictionaries [35]. Moreover, only a few projection function learning-based methods have gone into extending the model in the conventional setting into the generalized setting, particularly for the fine-grained classification task [27–34]. Lastly, very few experiments are dedicated

to finding an alternative of the nearest neighbour algorithm for the instance-label assignment in the target domain [27, 35].

Evolutionary algorithms are increasingly used in control systems, machine learning and engineering processes for single and multi-objective optimization [14–17]. These methods are relatively simple to interpret and involve linear operations, making them computationally inexpensive [17]. Therefore, in the context of machine learning, these algorithms are a good alternative to conventional machine and deep learning methods, particularly for hyperparameter tuning tasks [15–17]. Several methods have been proposed for parameter learning and hyperparameter tuning in the machine and deep learning applications. However, limited attention has been given towards parameter learning and hyperparameter fine-tuning in the transfer learning paradigm [36, 37]. Therefore, dedicated efforts are required for learning parameters and setting hyperparameters while using transfer learning algorithms primarily because these algorithms exploit large and complex source domains, and the conventional parameter learning and hyperparameter fine-tuning need to be computationally efficient.

3 Evolutionary Algorithms

Evolutionary algorithms have gained immense popularity in machine learning and optimization; the algorithms in this class are inspired by natural evolution processes like reproduction, mutation and natural selection to obtain solutions to optimization problems, particularly in scenarios where the exact methodology for getting the optimal solution may not be available [16]. These algorithms are computationally simple and able to self-adapt themselves in the search space; moreover, in machine learning, these approaches are useful as they do not rely on gradient information and can be used to minimize loss functions that are non-differentiable [36–38].

In optimization, meta-heuristics refers to a high-level procedure/methodology to find, generate, and tune a search algorithm to obtain the optimal solution efficiently. In this way, good meta-heuristics can help understand the problem and design efficient search methods to accelerate the entire optimization process. The class of evolutionary algorithms is a subset of meta-heuristics called populational as they rely on mimicking population dynamics to develop an efficient search procedure [14]. In the following section, the categorization of different evolutionary algorithms is provided.

3.1 *Categorization of Evolutionary Algorithms*

Based on how a population is created and maintained and the way natural selection happens, the existing evolutionary algorithms are classified into the following categories:

- (i) **Genetic Algorithms:** Genetic algorithm refers to a class of adaptive search-based methods designed for solving optimization problems; Darwin's theory of evolution inspires the algorithms in this class [14]. Specifically, a population containing individuals/solutions is initialized, and a fitness is assigned to each population member; the fitness measures how likely the individual is to survive across generations. Thereafter, new offspring are created using these solutions, and some diversity is added to them. Specifically, a mutation occurs when the offspring are modified only slightly, and crossover happens when significant diversity is added to the offspring. Further, the fitness of these diverse offspring is computed, and all the individuals are ranked according to their fitness. Lastly, a selection mechanism filters most fit individuals and some relatively less fit individuals into the next generation. Multiple generations keep emerging until an optimal solution is found in any given generation; with increasing generations, the estimate of solutions/fitness of individuals continues to improve. These algorithms are useful in machine learning as they can optimize loss functions that are discrete, continuous, differentiable and non-differentiable. However, the optimality of the solutions is not guaranteed.
- (ii) **Genetic Programming:** The algorithms based on genetic programming are a subset of genetic algorithms; specifically, both these classes of algorithms are based on maintaining and evolving a population across multiple generations so that with increased generations, the estimate of individuals and their fitness/survivability improves [14]. However, the fundamental difference between these classes of algorithms is that while genetic algorithms produce an optimal solution, algorithms based on genetic programming yield an optimal program, i.e., the two classes of algorithms differ in terms of what the individuals mean.
- (iii) **Evolutionary Programming:** Evolutionary programming relates very closely to genetic programming and is popular when the program's structure is fixed [14]. Specifically, evolutionary programming searches for the optimal program following evolutionary strategy, i.e., by updating the program parameters across generations; however, in contrast to genetic programming, the general structure of the program across generations is fixed.
- (iv) **Evolution Strategies:** Algorithms based on evolution strategy directly use the decision variables to represent the potential solutions/individuals; moreover, instead of using the fitness function, the problem functions are considered and manipulated directly [14]. In this way, evolution strategies are problem-dependent instances of evolutionary algorithms and use the decision variables and functions to create and evolve the solutions in any generation.

In this study, PSO algorithm is used for tuning the hyperparameters of different transfer learning-based models. PSO is a popular optimization technique inspired by the social behavior of birds and fish, particularly their flocking and swarming behaviors. The motivation stems from its ability to efficiently search for optimal solutions in complex, high-dimensional search spaces. Specifically, the algorithm is simple to implement, strikes a balance between exploration and exploitation

capabilities of the optimizer and remains highly parallelizable. The mathematical formulation of this algorithm is detailed in the following section.

4 Particle Swarm Optimization

PSO belongs to the class of evolutionary algorithms and mimics different real-world phenomena, including the foraging and social behaviour of the swarm and school of fish [39]. Specifically, the cooperative behaviour of the swarm and school of fish in their search for food is expressed by this algorithm. The different members in the swarm/fish learn from their experience and other members' experience to effectively locate their food. In this way, PSO expresses the principles of artificial life and evolutionary computing. Additionally, the algorithm is one of the direct search methods meaning that no gradient information is required during the optimization, and the optimization remains computationally inexpensive in both memory and time; as a result, the algorithm is extensively applied to functions that are not continuous. The following section presents the mathematical formalization of the PSO algorithm.

4.1 Mathematical Formulation

Similar to other evolutionary algorithms like genetic algorithms, PSO proceeds with random initialization of the population; the initialized population contains solutions called particles. However, unlike other evolutionary algorithms like genetic algorithms, no crossover or mutation occurs, and the particle is assigned a randomized velocity component. Specifically, instead of performing crossover or mutation, the current particle's velocity is perturbed relative to its previous position. Further, an instance of PSO algorithm has the following distinct features:

- (i) Best fitness of each particle i known as *local best*, $p_{i, lb}$.
- (ii) Best fitness of the swarm known as *global best*, $p_{gb, best}$
- (iii) Velocity and position update of each particle, $v_i^{(t+1)}$ and $x_i^{(t+1)}$, respectively

The personal/local best solution (in terms of fitness function) $p_{i, lb}$ of the i -th particle is the best solution achieved by that particle so far; global best p_{gb} is the best solution discovered by any particle in the swarm so far.

Specifically, the position of the i -th particle is adjusted as:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (1)$$

where $x_i^{(t+1)}$ is the updated position of the i -th particle in the $(t + 1)$ -th iteration; $x_i^{(t)}$ and $v_i^{(t)}$ are the position and the velocity components in the (t) -th iteration. Further, the velocity of the i -th particle is updated as:

$$v_i^{(t+1)} = \Omega v_i^{(t)} + c_1 r_1 \left(p_{i,lb}^{(t)} - x_i^{(t)} \right) + c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)} \right) \quad (2)$$

In Eq. (2), the initial velocity component $v_i^{(0)}$ is randomly initialized; Ω is the inertia component that controls the contribution of previous velocity component $v_i^{(t)}$ in the updated component $v_i^{(t+1)}$. The local best of the i -th particle is $p_{i,lb}^{(t)}$ and $p_{gb}^{(t)}$ is the global best among all the particles achieved till the (t) -th iteration. The velocity update rule in Eq. (2) has the following components:

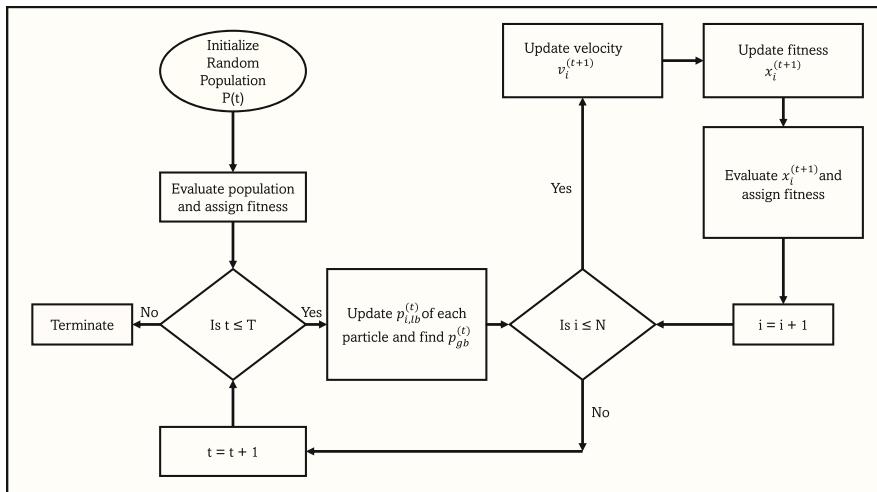
- (i) **Momentum:** The term $\Omega v_i^{(t)}$ is the momentum or the inertia component as some fraction of the velocity in the previous iteration (t) is considered in the $(t + 1)$ -th iteration, i.e., this component stores a memory of the previous velocity/flight direction and hence, prevents the particle from changing its direction drastically.
- (ii) **Cognitive/Nostalgia:** The term $c_1 r_1 \left(p_{i,lb}^{(t)} - x_i^{(t)} \right)$ is the cognitive component as each particle i is aware of its position relative to its best performance $\left(p_{i,lb}^{(t)} - x_i^{(t)} \right)$ achieved till the (t) -th iteration.
- (iii) **Social/Envy:** The term $\left(c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)} \right) \right)$ captures the performance of the particle i relative to its neighbours/swarm members and is the social/envy component.

In Eq. (2), c_1 , c_2 and Ω are the hyperparameters that need to be set before running the algorithm; r_1 and r_2 are the random numbers generated in $[0, 1]$. The algorithmic steps in the PSO are given in Algorithm 1.

Further, the details of the different algorithmic steps in the PSO are shown graphically as Fig. 1.

Algorithm 1 Pseudocode: Particle Swarm Optimization

1: **Input:** Number of particles N , hyperparameters c_1 , c_2 and Ω , random numbers r_1 and r_2 and maximum number of iterations T .
 2: **Output:** Optimal solution $x_{gb}^{(T)}$
 3: **Initialization Step**
 4: **for** particle $i = 1$ to N **do**
 5. Initialize particle and compute its $fitness_i$
 6. Set local best $p_{i,lb} = fitness_i$
 7. Set $v_i^{(0)}$ randomly
 8: **end for**
 9: **Algorithm Steps**
 10: **while** $t \leq T$ **do**
 11. **for** particle $i = 1$ to N **do**
 12. Compute its fitness as $fitness_i^{(t)}$
 13. **if** $fitness_i^{(t)} > p_{i,lb}^{(t)}$ **then** set $p_{i,lb}^{(t)} = fitness_i^{(t)}$
 14. **end if**
 15. **end for**
 16. Set the particle with the best fitness $p_{gb}^{(t)}$ as $x_{gb}^{(t)}$
 17. **for** particle $i = 1$ to N **do**
 18. Update velocity component using Equation 2
 19. Update position component using Equation 1
 20: **end for**
 21: **end while**
 22: Report the optimal solution as $x_{gb}^{(T)}$

**Fig. 1** Flowchart describing the different algorithmic steps involved in the PSO algorithm

5 Transfer Learning

Transfer learning algorithms allow the training of the machine and deep learning models in low-resource environments characterized by limited training exemplars. However, to train these models in low-resource environments, the limited information in these domains must be complemented by side information from a related source domain. These algorithms are primarily motivated by the following psychological theories [12]:

- (i) *Theory of identical elements*: E.L. Thorndike developed this theory and related knowledge transfer between two circumstances; the extent of this transfer is proportional to the similarity between these circumstances.
- (ii) *Theory of generalization of experience*: Charles Judd proposed this theory and suggested that when a learner learns a task A, he has essentially developed a general understanding of the task and this understanding gets carried forward while solving a related task B.

5.1 Mathematical Formulation

A typical transfer learning setting comprises a source and a target domain, and the learner capitalizes on knowledge obtained in the source domain to further improve its performance in the latter domain.

Mathematically, a domain \mathfrak{D} comprises of marginal probability $P(X)$ and a feature space X where $X \equiv \{x_1, x_2, \dots, x_n\}$ and x_i 's are the different features in the feature space. Further, any domain has a corresponding task \mathcal{T} that comprises of the label space \mathcal{Y} and a predictive function $f(\cdot) : X \rightarrow \mathcal{Y}$; this predictive function maps instances in X to corresponding labels in \mathcal{Y} .

With the above setting, the objective of any transfer learning algorithm is to learn a task \mathcal{T}_S in \mathcal{D}_S and use the knowledge so obtained in \mathcal{D}_T to improve the learning of target task \mathcal{T}_T . In this setup, it is assumed that the tasks \mathcal{T}_S and \mathcal{T}_T are relatable; subscripts S and T denote that the entities belong to the source and target domains in the same order. The improved target domain predictive function is denoted as $f_{ST}(\cdot)$ to underscore that the function learned in \mathcal{D}_S , $f_S(\cdot)$ improves the function obtained in the target domain $f_T(\cdot)$. To summarize, a task \mathcal{T}_S is learned in \mathcal{D}_S and the knowledge gained in learning \mathcal{T}_S is transferred to the target domain to improve the learning of task \mathcal{T}_T . The following section details the different transferable entities in a typical transfer learning framework.

5.2 What Can Be Transferred?

In this section, the different transferable entities in a transfer learning-based framework are described; these entities are transferable as they encode the different forms of knowledge obtained in the source domain while solving the source task \mathcal{T}_S . Precisely, the following entities can be transferred to achieve knowledge transfer across the two domains [12]:

- (i) *Instances*: Instances may be transferred partly or wholly from the source to the target domain to compensate for the limited training exemplars in the target domain; the popular methods to transfer the instances include re-weighting and importance sampling.
- (ii) *Features*: Features may be transferred to the target domain in scenarios where the features in the source domain are rich and hence, discriminatory; in the transfer learning nomenclature, this process is also called as feature sharing. Feature sharing assumes that the source and target domain instances share a common feature space, and the relative similarity/dissimilarity of these instances could be preserved in this common feature space.
- (iii) *Model Parameters*: Model parameters and hyperparameters can also be transferred to the target domain if the tasks \mathcal{T}_S and \mathcal{T}_T have common parameters/hyperparameters.
- (iv) *Relational Knowledge*: Relational knowledge is vital in applications like probabilistic graphical models, genome sequencing, pattern matching and graph-based neural networks where different entities are connected, and their association needs to be learned. Moreover, extracting relational knowledge is challenging and intricate, particularly for large-sized and complex datasets.
- (v) *Model Architecture*: Techniques in this category involve the transfer of the source domain model architecture to the target domain; the architecture may be transferred completely or in parts. These techniques are primarily used in applications like speech recognition, computer vision and natural language processing where a complex and state-of-the-art model is already trained in the source domain and needs to be reused in the target domain for solving a similar task. In cases where the source and the target tasks differ significantly, the last few layers are trimmed before the model architecture is transferred.

The following section highlights the categorization of transfer learning algorithms/settings based on the outcome achieved as a result of knowledge transfer and the data available in both domains.

5.3 Categorization of Transfer Learning Algorithms

Transfer learning algorithms/settings can be classified into different groups based on what outcome is achieved due to knowledge transfer and the kind and volume of data available in the participating domains.

For instance, based on whether the knowledge transfer taking place from the source to the target domain assisted in learning the target task, knowledge transfer may be classified as:

5.3.1 On the Basis of Effect

- (i) If the knowledge transfer improves the estimate of the predictive function in the target domain, i.e., if the performance of predictive function $f_{ST}(.)$ is better than $f_T(.)$, the transfer is *positive* in nature.
- (ii) The transfer becomes negative if the knowledge transfer degrades the predictive function $f_T(.)$ obtained in the target domain, i.e., if the target model performs better without the knowledge transfer, *negative* transfer ensues.
- (iii) If the performance of target predictive function $f_T(.)$ does not change after the transfer, the transfer remains *neutral*.

5.3.2 On the Basis of Availability of Label Information for Transfer

Transfer Learning settings can also be categorized into inductive, transductive and unsupervised settings based on label information in the source and the target domain [12]. Specifically, the setting is inductive if the target domain instances are labelled, and label information in the source domain is immaterial. In the transductive setting, instances in the source domain are labelled; however, the instances in the target domain remain unlabelled. The unsupervised transfer learning setting corresponds to the non-availability of label information in both domains and is the most challenging of the three settings. The categorization of the transfer learning setting is tabulated in Table 1.

Table 1 Classification of existing transfer learning algorithms on the basis of availability of label information

Labels in \mathcal{D}_S	Labels in \mathcal{D}_T	Variant
Yes/No	Yes	Inductive
Yes	No	Transductive
No	No	Unsupervised

5.3.3 Inductive Transfer Learning

Transfer learning algorithms in the inductive setup improve the estimates of target domain predictive function $f_T(\cdot)$ by leveraging the knowledge in \mathcal{D}_S while solving for the source task \mathcal{T}_T given that the tasks in the two domains are not same. In this setting, features, instances and parameters are usually transferred. Further, based on the availability of labels in the source domain, approaches in the inductive setting are classified into self-taught and multitask learning approaches. Specifically, label information in the source domain corresponds to multitask learning; self-taught learning proceeds without the label information in \mathcal{D}_S . The categorization of transfer learning algorithms in the inductive setup into self-taught and multitask learning is given in Table 2.

5.3.4 Transductive Transfer Learning

Transfer learning approaches in the transductive setting improve the estimates of target domain predictive function $f_T(\cdot)$ by using the knowledge derived in the source domain \mathcal{D}_S while solving the source domain task \mathcal{D}_S where the domains are dissimilar, i.e., $\mathcal{D}_S \neq \mathcal{D}_T$; however, in contrast to the inductive setup, the tasks in the two domains are same, i.e., $\mathcal{T}_S = \mathcal{T}_T$. The underlying rationale behind these approaches is to adapt the predictive function $f_S(\cdot)$ obtained in the source domain to better the target domain predictive function $f_T(\cdot)$; Approaches in this setting proceed by the transfer of instances and features only.

5.3.5 Unsupervised Transfer Learning

Unsupervised transfer learning-based approaches are most challenging and are prevalent in scenarios where the source and target tasks are different, i.e., $\mathcal{T}_S \neq \mathcal{T}_T$ and the corresponding labels spaces \mathcal{Y}_S and \mathcal{Y}_T are not provided.

5.3.6 Multitask Learning

Multitask learning attempts to enhance the model for the task \mathcal{T}_i by utilizing the information contained in the ‘ m ’ related tasks $\mathcal{T}_{i=1}^m$ [13]. Precisely, the learner uses *shared* information contained across these ‘ m ’ tasks to maximize its learning.

Table 2 Classification of transfer learning approaches in the inductive setting based on the label information in the source domain

Availability of labels in \mathcal{D}_S	Availability of labels in \mathcal{D}_T	Approach
No	Yes	Self-taught learning
Yes	Yes	Multitask learning

Multitask learning-based approaches use the knowledge obtained in solving multiple related tasks to leverage the performance of the target task. Specifically, a given a set of ' m ' related tasks $\mathcal{T}_{i=1}^m$ where all tasks or a subset of them are similar, a multitask learner \mathcal{L} captures the shared information contained across these ' m ' tasks to learn the target task \mathcal{T}_T ; In most of the applications, the ' m ' different tasks $\mathcal{T}_{i=1}^m$ are supervised and reside in the source domain itself.

5.3.7 On the Basis of Size of Target Domain

Transfer learning algorithms/settings can also be classified based on the size of the target domain, i.e., based on the number of instances present in the target domain. Specifically, when there are numerous instances in the target domain, it is a self-taught learning setting; in the few-shots setting, there are only a few samples for each participating class in the target domain. One-shot learning is the most constrained setting of few-shot learning and is characterized by the availability of only one sample per class. Typically, the number of participating classes is denoted by N-way and the number of samples in that class is marked as C-shots; therefore, a few-shots setting where there are only five participating classes and only three samples are there for each of these five classes, the setting is called 5-way, 3-shot setting. Zero-shot learning corresponds to learning novel concepts in the target domain, i.e., in the zero-shot setting, no samples are available in the target domain for the classes that need to be recognized in this domain. The different transfer learning settings based on target domain size are tabulated in Table 3.

Different transfer learning settings are considered based on the target domain size for the experiments, and models specific to these settings are proposed; specifically, dictionary learning-based approaches to self-taught and zero-shot learning tasks are proposed. The proposed models are based on learning dictionaries in both the domains separately; further, these proposed models also have some hyperparameters that need to be estimated. In this work, trainable parameters and hyperparameters are estimated using a PSO algorithm.

Table 3 Classification of transfer learning algorithm given the size of target data

Serial no.	Number of instances	Approach
1	Numerous	Self-taught
2	Few	Few shots
3	One	One shot
4	No samples	Zero-shot

6 Self-Taught Learning

In a self-taught learning [18] setup, supervised classifiers are trained in the target domain using transformed target domain instances. The unsupervised feature learner learned in the source domain then transforms the original target domain instances. Subsequently, a weak supervised classifier is trained on these transformed instances to map these transformed instances to their corresponding labels. The steps involved in the self-taught learning setup are detailed in the following subsections.

6.1 Mathematical Formulation

Self-taught learning considers a source domain S containing ample unlabelled instances, i.e., $S = \{(x_s)\}_{i=1}^{i=M}$ and these M unlabelled instances participate in the unsupervised feature learning task. The feature learner or the feature transformation function obtained at the end of the unsupervised feature learning step is denoted as $f_W(\cdot)$; in this notation, W is the set of trainable parameters learned. Further, the target domain $T = \{(x_t, y_t)\}_{i=1}^{i=N}$ contains N labelled instances where $N \ll M$; these N labelled instances are split into train and test sets represented by pairs $(x_t, \text{train}, y_t, \text{train})$ and $(x_t, \text{test}, y_t, \text{test})$ respectively. The unsupervised feature learner $f_W(\cdot)$ parameterized by W transforms the instances in the target domain from $(x_t, \text{train}, x_t, \text{test})$ to $(f_W(x_t, \text{train}), f_W(x_t, \text{test}))$. Unsupervised feature learning and the feature transformation steps are illustrated in Fig. 2a, b. After that, a supervised feature learner $g_\theta : X_1 \rightarrow \mathcal{Y}$ characterized by a of trainables θ is trained to map these transformed instances to their corresponding labels, i.e., $g_\theta : f_W(x_t, \text{train}) \rightarrow y_t, \text{train}$. The performance of this supervised classifier g_θ is then estimated on the transformed test set $(f_W(x_t, \text{test}), y_t, \text{test})$. In the above notations, the subscripts ‘ s ’ and ‘ t ’ stand for the two domains, respectively, and the subscripts train and test denote whether the data is used for training or testing, respectively.

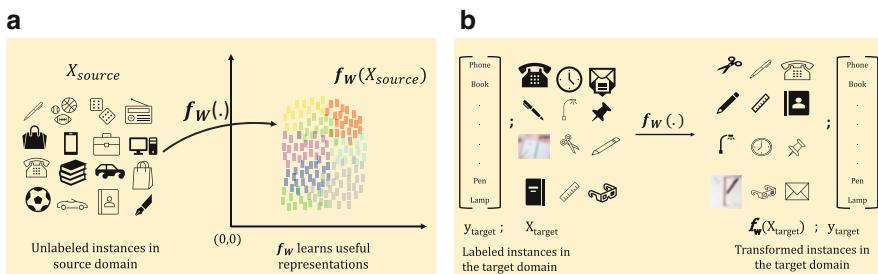


Fig. 2 (a) Illustration of feature learning that proceeds in the source domain. (b) Feature transformation of target domain instances through the feature learner derived in the source domain

6.2 Proposed Methodology

Self-taught learning is a two-step process—firstly, unsupervised feature learning takes place using ample unlabelled instances present in the source domain to learn an efficient feature transformation function $f(.)$. Thereafter, the feature transformation function $f(.)$ obtained in the source domain is used to transform target domain instances. Lastly, weakly supervised classifiers are trained on these transformed target domain instances to learn a mapping from transformed features to their corresponding labels.

Unsupervised feature learning in \mathcal{D}_S proceeds using a vanilla autoencoder; specifically, training a vanilla autoencoder involves minimizing a loss function based on learning a dictionary. The encoder weights obtained by training the autoencoder capture the feature transformation function in the source domain; further, support vector machine (*SVM*) and logistic regressor (*LR*) are used as weak classifiers in the target domain. The different steps involved in training a self-taught learner are described below:

- 1. Unsupervised feature learning in the source domain:** Vanilla autoencoder with a dictionary learning-based loss function is trained to learn feature representation using unlabelled instances in the source domain. The loss function of this autoencoder is

$$W^* = \arg \min_W \| X_S - X_S W W^T \|_F^2 + \lambda (\| W \|_F^2) \quad (3)$$

Where X_S contains instances in the source domain and $\hat{X} = X_S W W^T$, is its reconstructed version. The entities in the encoder matrix W are the trainable parameters; λ is the regularization hyperparameter and ensures that the encoder matrix W has a smaller Frobenius norm. The different entities in the encoder matrix W and the regularization hyperparameter λ are concatenated to form a particle; specifically, the loss function in Eq. (3) becomes the fitness function for the PSO.

- 2. Feature transformation and supervised classification in the target domain:** The encoder matrix W obtained by minimizing Eq. (3) transforms the instances in the target domain and then these transformed target domain instances along with their labels participate in the supervised classification in the target domain. In our experiments, a support vector machine and a logistic regression classifier are used for the supervised learning task in the target domain. These weak classifiers are purposely chosen so that any improvements in the generalization capability of the classifiers could be attributed to rich feature representations obtained via the encoder matrix W and not to the target classifier itself. The loss function used for the support vector machine and logistic regressor is hinge loss and log loss, respectively.

6.3 Experiments

The experimental setup followed by dataset description and relevant data preprocessing steps is detailed in the following subsections.

Experiment Setup: The experiments are performed on a system having two x64-based Intel(R) i5 processors supported by 8 GB of RAM and 4 GB of NVIDIA GeForce GTX 1650Ti GPU; the programs are run in Python using basic libraries like Numpy and Pandas.

Dataset Description and Preprocessing: The Covid-19 Lungs CT Scan dataset [40] is an openly accessible dataset created to detect Covid-19 using a patient's lung radiographs. These radiographs are grey-scale images sized 512×512 and have been collected from patients admitted to different hospitals in Iran. In totality, the dataset consists of 8439 radiographs; among these, 7496 radiographs show lungs infected with Covid-19, and the remaining 943 radiographs represent healthy lungs. The illustration of radiographs contained in this dataset is shown in Fig. 3.

As a part of data preprocessing, these radiographs are re-scaled from 512×512 to 64×64 and then represented as a 4096-dimensional feature vector. Further, the

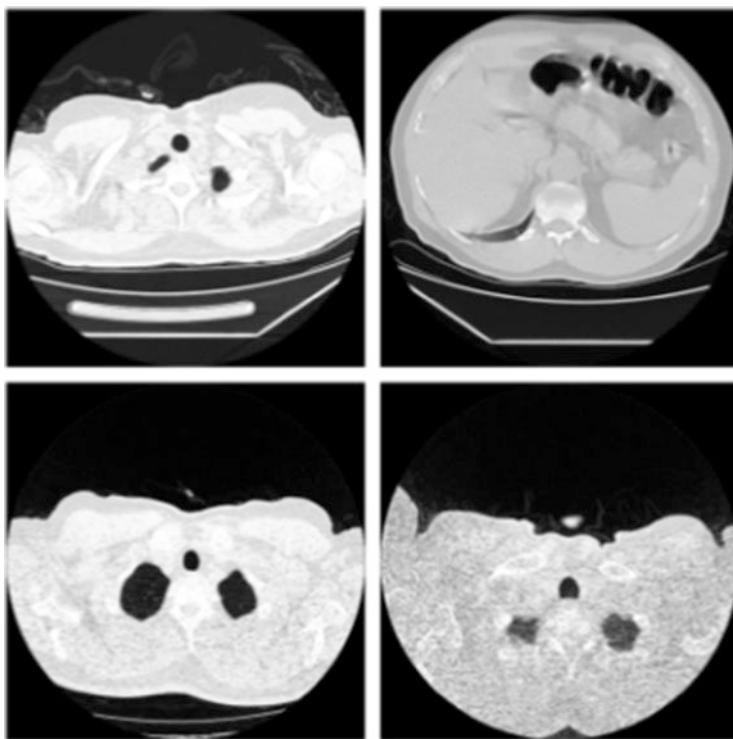


Fig. 3 Illustration of radiographs contained in the Covid-19 Lungs dataset. The top row consists of radiographs of patients infected by Covid; the bottom row shows radio graphs of healthy lungs

values in these features are normalized to [0, 1.0]; the data pool containing 8439 samples is split between the two domains so that 7173 samples are in source, and the remaining 1266 samples are in the target domain. The samples in the source domain participate in the unsupervised feature learning task, while target domain samples are used to train and evaluate the target domain classifiers. Specifically, 1012 out of 1216 target domain samples train the classifiers, while the remaining 254 samples are used to evaluate the classifiers. The results obtained with the proposed self-taught learners are reported in Sect. 6.4.

6.4 Results and Discussion

This section reports the performance of supervised classifiers trained on the target domain data; specifically, the performance of logistic regression and support vector classifier is reported. These classifiers have been chosen to ensure that the target classifiers remain weak so that any performance improvements obtained with the self-taught features could be accredited to rich self-taught features and not to the classifiers themselves. The classifiers trained on original features are taken as a baseline for comparison with those trained on self-taught features.

The comparative performance of target classifiers on original and self-taught features for the Covid-19 Lungs dataset is tabulated in Table 4. The performance of the logistic regressor trained on self-taught features comes at par with the classifier trained on original features. Specifically, a classification accuracy of 95.27%, precision, recall and F1 score of 0.95 are obtained with both self-taught and original features. These results show that self-taught features can capture the same discriminating information in lower dimensions as the original features. However, with the support vector machine classifier, the performance with self-taught features is marginally lesser than that obtained with the original features. Specifically, the performance with the self-taught features is only $90.94 - 87.79 = 3.15\%$ lesser compared with the performance obtained with the original features; the

Table 4 Comparative analysis of the target classifier's performance with the original and learned features

Serial no.	Performance metrics	Logistic regression		Support vector machines (SVM)	
		Original features	Self-taught features	Original features	Self-taught features
1	Classification accuracy	95.27	95.27	90.94	87.79
2	Precision	0.95	0.95	0.91	0.87
3	Recall	0.95	0.95	0.91	0.88
4	F1 score	0.95	0.95	0.89	0.84

The results are reported for logistic regressor and support vector machine; the classifiers are trained on the Covid-19 Lungs dataset to detect Covid-19 infection

corresponding decline in precision, recall and F1 score is 0.04, 0.03 and 0.84, respectively. The results obtained with both the classifiers suggest that the self-taught features contained almost the same amount of discriminating information as the original features and hence, can be used in place of the original features.

In the above experiments, unsupervised feature learning proceeds with 7173 unlabelled samples in the source domain, and the optimal code dimensions, i.e., the dimensionality of the self-taught features, is empirically obtained as 1536. In this way, the number of trainable parameters in the source domain is $4096 * 1536 = 6,291,456$; together with the one hyperparameter λ , the dimensions of the particle are 6,291,457. For obtaining the feature transformation function $f(\cdot)$ and the feature learner hyperparameter λ , the PSO runs with four particles for 20 generations; the bound on the solution is $[-3, +3]$ and values of acceleration coefficients c_1, c_2 and inertia parameter Ω is 0.5 each. Lastly, the feature learner hyperparameter λ obtained by the optimization algorithm is -0.89 . The classifiers in the target domain are trained on 1012 labelled samples and the performance of these classifiers is detailed on a test set containing 254 samples.

7 Zero-Shot Learning

The research community has recently witnessed a rising interest in how people recognize novel concepts and have discovered that to recognize novel concepts, some description of these novel concepts is required. Zero-shot learning attempts to learn classifiers that can recognize novel concepts and is inspired by the way humans relate known concepts to recognize novel concepts [12].

7.1 Mathematical Formulation

A zero-shot learning setup considers a source and a target domain given by S and T , respectively. Source domain S contains N_S labelled instances spanning across C_S different classes. We define by X_S and z_S the source domain feature matrix and the corresponding label vector in the same order; further, the class-level semantic representations/embeddings of labels contained in z_S are contained in the source domain embedding matrix Y_S . The instances in X_S and their corresponding labels z_S participate in training the classifier; therefore, these instances are called as seen class instances, and the C_S classes contained in z_S are the seen classes. In this way, the source domain S is represented as a triplet containing $S = \{X_S, z_S, Y_S\}$ as known quantities. Further, target domain T contains N_t unlabelled instances from among the C_t classes such that the set of classes C_S and C_t are disjoint, i.e., $C_S \cap C_t = \emptyset$. We denote by X_t the feature matrix in the target domain; further, as target domain instances are unlabelled, the label vector z_t and hence, the target embedding matrix Y_t are unknown. Specifically, we only know the set of classes that each of the

instances in X_t may belong to without knowing the label of target instances. The target instances in X_t do not participate in the training of the classifier, and therefore, these instances are called unseen class instances, and the different classes in z_t are unseen classes. To summarize, the target domain T is a triplet $T = \{X_t, z_t, Y_t\}$ where the label vector z_t and the target embedding matrix Y_t are unknown. The instances reside in d -dimensional space, i.e., $X_S \in \mathbb{R}^{N_s \times d}$ and $X_t \in \mathbb{R}^{N_t \times d}$. Further, the semantic embeddings come from a secondary source of side information and are placed in an m -dimensional feature space, i.e., $Y_S \in \mathbb{R}^{n_s \times m}$, $Y_t \in \mathbb{R}^{n_t \times m}$. These semantic representations relate seen and unseen classes and can be binary or real-valued; in the proposed experiments, the embeddings are expert-annotated binary vectors.

With the above setup in place, zero-shot learner recognizes novel concepts in the target domain using partly or wholly the feature-label associations obtained in the source domain and the side information when the seen and unseen class sets are disjoint, i.e., when $C_S \cap C_t = \emptyset$. Further, the zero-shot learning setup can be conventional or generalized depending upon the participation of instances during testing; specifically, if the test instances can come from only the unseen classes, the setup is conventional. In the generalized setup, both seen and unseen classes participate in the testing.

In general, ZSL proceeds by learning a projection $f(\cdot) : X_S \rightarrow \mathcal{Y}_S$ using source domain data. The projection $f(\cdot)$ obtained is then used in \mathcal{D}_T to project target instances (contained in X_t) to the embedding space. The projections of target instances in the semantic space Y_t are then fed to the nearest neighbour classifier to determine the unseen class embedding nearest to these projections. Once the nearest unseen class embedding is determined, the corresponding unseen class label is assigned to the projected target instance. The detailed steps in training a zero-shot classifier are described in the following section.

7.2 Side Information: Role and the Different Sources

The classifier trained using seen class instances cannot recognize novel class instances. Therefore, for zero-shot recognition to occur, some source of side information must establish the relatedness between the two class sets. Since these sources relate the seen and unseen classes in terms of how similar/dissimilar they are to each other, the information in these sources is called semantic information, and the space in which this information lies in the semantic space. Moreover, these sources can come from several sources and exist in multiple modalities. This information generally comes as expert annotated attribute vectors or lexical embeddings obtained from an already trained classifier trained to solve a natural language processing task.

7.3 Proposed Methodology

As described in Sect. 7.1, the projection learning-based approaches learn a projection function $f(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$ from \mathcal{X} to \mathcal{Y} using source-domain data. We follow [31] to learn this projection in a dictionary learning-based framework; specifically, the learning of $f(\cdot)$ is transformed into a sparse-coding problem. Moreover, the learning of projection function in the form of dictionary occurs in both the domains; the implementation details are described in the following section.

Dictionary Learning: Source Domain We build on [31] to model the dictionary learning in the source domain and represent it as:

$$D_S^* = \arg \min_{D_S} \| X_S - D_S Y_S \|_F^2 \text{ such that } \| d_i \|_F^2 \leq 1 \quad (4)$$

where the source domain feature matrix, X_S , and the corresponding semantic embedding matrix, Y_S are known and the D_S needs to be computed; The subscript S in the variables in Eq. (4) corresponds to the source domain and $\|\cdot\|_F$ denotes the Frobenius norm. The above Equation corresponds to a least-squares problem that has a closed-form solution. The loss in Eq. (4) is regularized to get solutions with a smaller Frobenius norm. The regularization step is modelled as follows:

$$D_{s,fro} = \arg \min_{D_s} \| X_s - D_s Y_s \|_F^2 + \lambda_1 \| D_s \|_F^2 \text{ such that } \| d_i \|_2^2 \leq 1 \quad (5)$$

where λ_1 is a regularization hyperparameter; subscripts ‘ s ’ and ‘ fro ’ indicate the domain and the regularization put on the dictionary learned in Eq. (5). The loss given in Eq. (5) represents a ridge-regression loss with a closed form solution given by $D_{s,fro} = (X_s^T X_s + \lambda_1 \mathbb{I})^{-1} X_s^T Y_s$ [41].

Further, no structural constraint is put on D_s to capture specific patterns in the data manifold. In this direction, the loss given in Eq. (5) is regularized further to obtain a low-ranked solution; low-ranked solutions are favoured in dictionary learning-based frameworks as these solutions remain robust in the feature space [35, 42], i.e., the low-ranked and hence robust source dictionary D_s can accurately map instances from feature to the embedding space even when the former space gets corrupted by noise. In this direction, a nuclear norm regularization is added to regularize the loss function given in Eq. (5); this regularization is chosen to provide theoretical guarantees in obtaining low-ranked solutions [42]. The updated loss function is written as:

$$D_{s,fro+nuc} = \arg \min_{D_s} \| X_s - D_s Y_s \|_F^2 + \lambda_1 \| D_s \|_F^2 + \lambda_2 \| D_s \|_F^2 \text{ s.t. } \| d_i \|_2^2 \leq 1 \quad (6)$$

In Eq. (6), λ_1 and λ_2 are the model hyperparameters and control the strength of the two norms; the subscripts ‘ s ’ and ‘ $fro + nuc$ ’ correspond to the domain and the regularization added in the same order. Nuclear norm given by $\|\cdot\|_F^2$ is computed as the summation of singular values; mathematically, $\|D_s\|_*$ is computed by finding the

sum of singular values of the source dictionary D_s where the singular values of D_s are obtained via its *SVD* decomposition, i.e., $D_s = U \Sigma V^T$. By circularity of trace and *SVD* decomposition the nuclear norm of the source dictionary D_s is [35].

$$\| D_s \|_* = \text{tr} \left(\sqrt{V^T V \Sigma^2} \right) = \text{tr} \left(\sqrt{\Sigma^2} \right) = \text{tr}(\Sigma) \quad (7)$$

It turns out that while the Frobenius norm terms are differentiable, the nuclear norm term in Eq. (6) is not; consequently, gradient-based algorithms cannot be used to minimize the loss in Eq. (6). Therefore, to minimize the loss in Eq. (6), the subgradient value for the nuclear norm is computed as $\frac{\partial \|D_s\|_*}{\partial D_s} = UV^T$, and this subgradient is added to the gradient computed for all the differentiable Frobenius norm. Then the sum of gradients and subgradients is taken as the effective gradient with any gradient-based optimizer; the details of the optimization procedure are provided in Sect. 7.4. Lastly, for a fair comparison, the regularization hyperparameters λ_1 and λ_2 take equal values, i.e., $\lambda_1 = \lambda_2$. Relevant steps involved in learning dictionaries in the target domain are detailed in the following section.

Dictionary Learning: Target Domain Learning of dictionary in the target domain is different from the learning in the source domain; specifically, in the source domain, X_s and Y_s were known, and the only unknown was D_s . However, in the target domain, as the instances in X_t are unlabelled, their label vector z_t and hence, Y_t is unknown. Thus, to summarize, dictionary learning in \mathcal{D}_S involves learning only the source dictionary D_s ; in the latter domain, both D_t and Y_t need to be learned. This learning is given by:

$$\{D_t^*, Y_t^*\} = \arg \min_{D_t, Y_t} \|X_t - D_t Y_t\|_F^2 + \lambda_3 \|Y_t\|_1 \text{ s.t. } \|d_{ij}\|_2^2 \leq 1 \quad (8)$$

In the above Equation, the subscript ‘ t ’ in the symbols indicates that the entities belong to the target domain; as the seen and unseen class embeddings are usually sparse, the loss function described above is regularized with an L_1 regularization to yield target embeddings Y_t that are sparse. Mathematically, this regularization is given by $\|Y_t\|_1 = \sum_{i=1}^{n_t} \|y_i\|_1$ where y_i is the target embedding obtained for the i -th target domain instance x_i . However, it turns out that the solutions D_t and Y_t obtained by minimizing the loss in Eq. (8) may not be relevant in context to zero-shot learning task as confirmed by authors in [31].

Therefore, to ensure that solutions D_t and Y_t remain relevant to the zero-shot learning task, the target dictionary D_t is regularized to remain close to the source dictionary D_s . The underlying rationale is that since the domains are similar, the parameters obtained in each cannot be distant apart. Specifically, the source domain dictionary $D_{s, \text{fro} + \text{nuc}}$ regularizes the target dictionary D_t and an additional regularization term is added to loss described in Eq. (8) and the updated loss function is represented as:

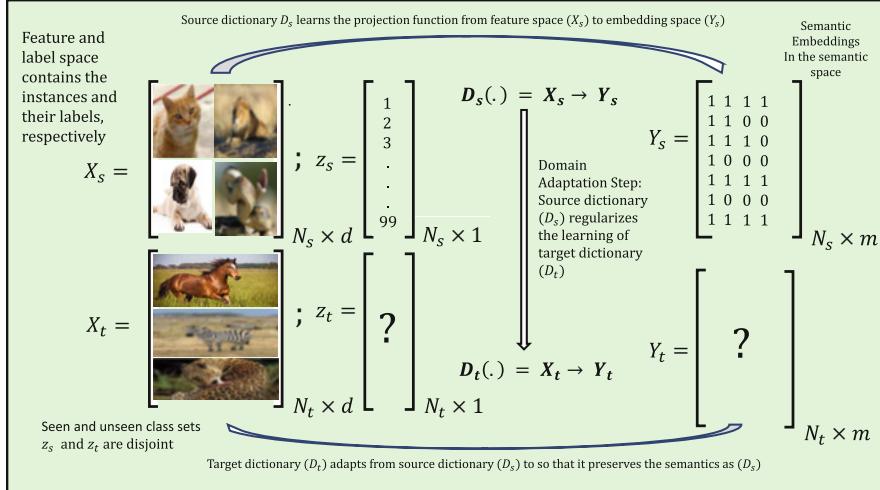


Fig. 4 Projection function learning approach to zero-shot learning involves learning projection functions from the feature to the semantic space; in the proposed approach, these functions are learned as different dictionaries with relevant regularization

$$\left\{ D_t^*, Y_t^* \right\} = \arg \min_{D_t, Y_t} \| X_t - D_t Y_t \|_F^2 + \lambda_3 \| Y_t \|_1 + \lambda_4 \| D_t - D_{s,fro+nuc} \|_F^2 \text{ s.t. } \| d_i \|_2^2 \leq 1 \quad (9)$$

The additional regularization term $\| D_t - D_{s,fro+nuc} \|_F^2$ is referred to as an adaptation regularization term, and this term ensures that the target dictionary D_t does not deviate much from the low-ranked source dictionary $D_{s,fro+nuc}$ and is, therefore, able to benefit from the knowledge contained in $D_{s,fro+nuc}$. In Eq. (9), the regularization hyperparameters λ_3 and λ_4 regularize the sparsity and domain adaptation terms, respectively. The above mentioned steps involved in training a zero-shot classifier are illustrated in Fig. 4. The optimization procedure for minimizing the loss functions given in Eqs. (4), (5), (6), (8) and (9) are provided in the following section.

7.4 Optimization Process

This section details the optimization procedure for minimizing the different loss functions described in Sect. 7.3.

To begin with, the loss in Eq. (4) represents a least-squares problem and has a closed-form solution; this loss function is provided for completeness and has not been optimized. Further, the loss given in Eq. (5) is a regularized form of loss given in Eq. (4) and represents a standard ridge regressor with a solution given by $D_{s,fro+nuc} = (X_s^T X_s + \lambda_1 \mathbb{I})^{-1} X_s^T Y_s$.

The loss in Eq. (6) is the proposed model and regularizes the loss in Eq. (5) with a nuclear norm constraint on D_s to get solutions that are low-ranked and hence, robust. The nuclear norm regularization in Eq. (6) makes it to become non-differentiable; therefore, gradient-based optimization techniques cannot be applied to minimize it. Consequently, to minimize the loss in the Eq. (6).

The method of subgradients is followed; specifically, gradients are computed for the terms that are differentiable, and subgradient is obtained for the non-differentiable nuclear norm. As detailed in [35], the subgradients for the nuclear norm regularization is computed as:

$$\frac{\partial \parallel D_s \parallel_*}{\partial D_s} = \frac{\text{tr}(\partial \Sigma)}{\partial D_s} = (VU^T)^T \quad (10)$$

In the above Equation, the matrices U and V come from the *SVD* decomposition of the source dictionary D_s . Thereafter, a gradient descent algorithm minimizes the loss given in Eq. (6); the effective gradient value used to update the parameters is the sum of gradients and subgradients computed above; the hyperparameter values for this mini-batch gradient descent algorithm are provided along with the recognition results.

The loss in Eq. (8) is for representation purposes and has not been minimized in the experiments; Eq. (9) involves learning of target dictionary D_t and the embedding matrix Y_t in the target domain. The minimization of this loss proceeds using the alternative minimization technique, where the loss function is minimized with respect to one variable at any given time, keeping the other variable fixed. Specifically, in one step, we solve for the target dictionary D_t keeping the target embedding matrix Y_t as fixed; the resulting loss becomes a least squares loss having a well defined solution. After that, in the alternate step, D_t is fixed and an optimal value for the embedding matrix Y_t is computed; in this setting, the resultant loss function represents a Lasso regressor and is solved using gradient descent algorithm. The above two steps continue until the optimal solutions are obtained, or the stopping criterion is reached. The detailed procedure is presented in Algorithm 2 while the procedure for quantifying the recognition results is described in Sect. 7.5.

Algorithm 2 Alternate Minimization Technique to Minimize the Loss Function Described in Eq. (9)

- 1: **Given:** Target domain feature matrix X_t , hyper-parameters λ_3 and λ_4
- 2: **Require:** Optimal estimates of the target dictionary D_t , and target embedding matrix Y_t
- 3: **while** still not-converged **do**
4. Solve $D_t^* = \arg \min_{D_t} \parallel X_t - D_t Y_t \parallel_F^2 + \lambda_4 \parallel D_t - D_{s,\text{fro+nuc}} \parallel_F^2$
5. Solve $Y_t^* = \arg \min_{Y_t} \parallel X_t - D_t Y_t \parallel_F^2 + \lambda_3 \parallel Y_t \parallel_1$
- 6: **end while**
- 7: Report the optimal values D_t^* and Y_t^*

7.5 Model Testing

In this section, the procedure for quantifying the recognition performance of the proposed zero-shot learner is detailed.

Specifically, once the target dictionary D_t and the embedding matrix Y_t are computed after minimizing the loss in Eq. (9), class assignment of target instances contained in X_t occurs using the nearest neighbour classifier or its alternative. For this, the learned embeddings Y_t are considered where $y_i \in Y_t$ represents the semantic representation for the i -th test sample $x_i \in X_t$. The label given to i -th target instance $x_i \in X_t$ corresponds to the nearest unseen class embedding in the vicinity of its learned embedding $y_i \in Y_t$; in the experiments, the number of neighbours considered for the instance-label assignment is 5. In the generalized zero-shot setting, both seen and unseen class embeddings and their distances with $y_i \in Y_t$ are considered for assigning the label to the target instance $x_i \in X_t$. Label propagation algorithm is also considered in addition to nearest neighbour algorithm for instance-label assignment in the target domain; the corresponding details are described in Sect. 7.6.

7.6 Label Propagation

Label propagation [43] is popularly used as an alternative to the nearest neighbour classifier in zero-shot learning tasks and represents a semi-supervised learning setup where the class assignment occurs by label propagation through a dataset containing labelled and unlabelled instances. The propagation happens through a graph-like structure where the instances are represented as nodes, and their similarity is modelled as edges. In our experiments, the similarity between different nodes is quantified using cosine similarity defined as $\text{cosine similarity } (y_i, y_j) = \frac{y_i \cdot y_j}{\|y_i\| \times \|y_j\|}$ where y_i and y_j are semantic embeddings and the quantification happens in the embedding space.

A generalized zero-shot learning setting is a more practical scenario characterized by the participation of all class instances in the testing phase, and in this setting, the challenge is to determine the label for these test instances accurately. Moreover, given the seen-unseen class imbalance, the mean of the two class accuracies do not correctly represent the performance of this generalized model. To this effect, researchers in [27] have proposed harmonic mean of unseen and seen class accuracy to quantify the generalized model's performance. Mathematically, this harmonic mean is given by:

$$H_{accuracy} = \frac{2 \times accuracy_{seen} \times accuracy_{unseen}}{accuracy_{seen} + accuracy_{unseen}} \quad (11)$$

In the above Equation, $accuracy_{unseen}$ and $accuracy_{seen}$ are accuracy estimates for the two classes. Dataset description and the relevant preprocessing steps are detailed in the following section.

7.7 Experiments

The experimental setup followed by dataset description and relevant data preprocessing steps is detailed in the following subsections.

Experiment Setup The experiments are performed on a system having two x64-based Intel(R) i5 processors supported by 8 GB of RAM and 4 GB of NVIDIA GeForce GTX 1650Ti GPU; the programs are run in Python using basic libraries like Numpy and Pandas.

Dataset Description and Preprocessing Caltech-UCSD Birds 200 (CUB) [35] dataset is used extensively to assess the performance of zero-shot learners, particularly for the fine-grained classification task. The original dataset contains 11,788 pictures of birds spanning across 200 different categories; the side information relating the seen and unseen classes is in the form of expert annotated real-valued vectors lying in \Re^{312} . In experiments with this dataset, samples from 150 seen classes participate in the training process and the samples coming from the remaining 50 unseen classes are used to test the model. Further, as a part of data preprocessing, an already pre-trained VGG-19 network is used to obtain the feature representations of the samples; these representations are taken out of the *fc2* layer of the network and lie in a 4096-dimensional feature space. The performance of the proposed zero-shot model is reported in the following section along with the hyperparameter values obtained.

7.8 Results and Discussion

This section details the recognition results obtained with the proposed zero-shot learner on the Caltech-UCSD Birds 200 (CUB) dataset; the results are reported in both the settings, namely conventional and generalized. Additionally, two different classifiers, namely, nearest neighbour and label propagation algorithms, are considered in the experiments for label assignment; the detailed results are tabulated in Table 5.

The recognition results (in terms of Top-1 accuracy) obtained on the Caltech-UCSD Birds 200 (CUB) are reported in Table 5; for the instance-label assignment, nearest neighbour and label propagation algorithms are used, and the performance of the classifier is reported in both conventional and the generalized settings. Specifically, the performance of the proposed zero-shot learner is 13.5% with the nearest neighbour-based classifier; however, with the label propagation classifier, the performance is 14.5%, thereby registering an increase in the performance by 14.5 –

Table 5 Recognition results obtained with the proposed zero-shot learner in the conventional and the generalized setting on the Caltech-UCSD Birds 200 (CU B) dataset

Serial no.	Dataset	Recognition accuracy (in %)			
		Conventional setting		Generalized setting	
		Nearest neighbor classifier	Label propagation classifier	Nearest neighbor classifier	Label propagation classifier
1	Caltech-UCSD (CUB)	13.5	14.5	8.9	8.2

The different columns correspond to the different settings and label assignment algorithms used. Top-1 class accuracy

$13.5 = 1.0\%$. These results suggest that in the conventional setting, the label propagation approach propagates the labels more accurately than the nearest neighbour classifier. In the generalized setting, the classifier's performance with the nearest neighbour classifier is better than the one reported with the label propagation method. Specifically, the nearest neighbour classifier reports a recognition accuracy of 8.9% that is $8.9 - 8.2 = 0.7\%$ better than the label propagation algorithm. Moreover, for a given instance-label assignment method, the classifier's performance in the conventional setting is better than the performance reported in the generalized setting. For instance, with the nearest neighbour classifier, the classifier's performance in the conventional setting is 13.5%; this performance drops to 8.9% in the generalized setting. Similarly, with label propagation, the performance in the conventional setting is 14.5%, and for the generalized setting, it is only 8.2%. The above results are expected as instances can come from both the class sets with significant similarity among them. In these scenarios, even a small deviation in the value of the target embedding $y_i \in Y_t$ can lead to incorrect labelling of the target instance $x_t \in X_t$.

To obtain dictionaries in the two domains along with the different hyperparameters, the trainables and the hyperparameters in Eqs. (6) and (9) are vectorized. Thereafter, these vectors are passed to different particle swarm optimizers, and the optimizers consider the negative of the loss functions given in Eqs. (6) and (9) as their fitness function. Specifically, in the source domain, source dictionary D_s needs to be learned; as the class embeddings are represented as 312-dimensional vectors, and the instances are represented using 4096 features, the total trainable parameters in D_s is $312 \times 4096 = 1,277,952$. Along with the one source domain hyperparameter λ_1 , the solution of the PSO algorithm is represented as a 1,277,953-dimensional vector. Further, in the target domain, there are 2976 instances in Y_t each represented using 312-dimensional; therefore, for estimating the target embedding matrix Y_t and the regularization hyperparameter λ_3 , the solution vector for the PSO needs to be initialized as a $(312 \times 2976) + 1 = 928,513$ -dimensional vector. Similarly, for estimating target dictionary D_t and hyperparameter λ_4 , a 1,277,953-dimensional solution needs to be initialized.

The PSO proceeds with four particles that evolve across 20 generations; for simplicity, the bound on the solution is fixed at $[-3, +3]$. The acceleration coefficients c_1 , c_2 and the inertia parameter Ω are each set to 0.5. The estimates for hyperparameters λ_1 , λ_2 and λ_3 are 1.23, 1.65 and 1.42, respectively. Lastly, the performance of the zero-shot learner is estimated on 2976 unlabelled instances in the target domain.

8 Key Takeaways

In this section, the key takeaways of using an evolutionary algorithm for training a self-taught and zero-shot learner is provided with a focus on the interdependence among model parameters, model hyperparameters and evolutionary algorithm hyperparameters.

8.1 *Role of Hyperparameters*

Typical machine and deep learning algorithms proceed by minimising a loss function; this function usually consists of different terms that capture specific properties/structures in the data manifold or the projection function. Hyperparameters control the weightage that needs to be assigned to each term in the loss function. In contrast to trainable parameters, which are learned using the algorithm, hyperparameters are set explicitly before the training ensues. In other words, hyperparameters remain external to the model and cannot be determined by the data.

Hyperparameters can influence the behaviour of the training algorithm, and consequently, this influence can significantly impact the model's performance. The values of these hyperparameters can be explicitly set by the practitioner and fine-tuned to suit a specific predictive problem. Generally, some rules of thumb and heuristics are used to set the hyperparameters. Additionally, these hyperparameters' values can be obtained via trial and error or by borrowing them from a similar task.

8.2 *Obtaining Trainables as a Byproduct of Hyperparameter Tuning Process*

In machine and deep learning, model parameters (also referred to as trainables) and hyperparameters have different roles to play; specifically, while trainables are obtained by training the classifier on the data, hyperparameters are set by the practitioner before the training starts. However, in the context of evolutionary computation algorithms, the trainables and hyperparameters are treated similarly.

Specifically, the solution vector is initialized to contain initial trainables and hyperparameters, and as the evolutionary computation proceeds, different solutions and hence, different sets of trainables and hyperparameters are obtained in each iteration. Finally, when the iterations stop, the set of trainables and hyperparameters are segregated from the obtained optimal solution.

8.3 *Exploration Vs. Exploitation*

Exploration and exploitation are important considerations for any evolutionary algorithm and are controlled by the different hyperparameters of the evolutionary algorithm. Specifically, exploration occurs when the algorithm tries to reach the area in the parameter search space where the chance of getting the optimal solution is maximal. In this way, exploration brings about large changes in the solution and its fitness with each iteration. Once the algorithm reaches the vicinity of the optimal solution, the exploration must stop, and the focus must shift to making small changes to the solution so that the algorithm can get to the optimal solution. As a result, exploitation is a slow process and aims at finding the optimal solution starting with sub-optimal solutions in its vicinity. Exploration is increased by reducing the inertia component Ω ; exploitation is controlled using cognitive and social components in context to the PSO algorithm. In the proposed experiments, the solution contains the trainables and the model hyperparameters; therefore, exploration and exploitation of these trainables and hyperparameters occur simultaneously.

9 Conclusion

In this work, a meta-heuristics-based evolutionary algorithm is used for learning parameters and setting hyperparameters in the transfer learning paradigm. Specifically, dictionary learning-based self-taught and zero-shot learners are proposed for the classification of lung diseases and novel concepts, respectively and PSO algorithm derives the model parameters and hyperparameters. Empirical results obtained on Covid-19 Lungs CT Scan dataset show that the proposed self-taught learner with parameters and hyperparameters derived from the PSO achieves a classification accuracy of 95.27% with the self-taught features. The recognition performance of the proposed zero-shot learner on the Caltech-UCSD Birds 200 (CUB) is 14.5% and 8.5% in the conventional and generalized settings, respectively. In future, different evolutionary algorithms will be explored for training transfer learning-based classifiers.

References

1. Sen PC, Hajra M, Ghosh M. Supervised classification algorithms in machine learning: a survey and review. In: Emerging technology in modelling and graphics: proceedings of IEM graph 2018. Singapore: Springer; 2020. p. 99–111.
2. Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, et al. A state-of-the-art survey on deep learning theory and architectures. *Electronics*. 2019;8(3):292.
3. Wu X, Sahoo D, Hoi SC. Recent advances in deep learning for object detection. *Neurocomputing*. 2020;396:39–64.
4. Latif S, Rana R, Khalifa S, Jurdak R, Qadir J, Schuller BW. Deep representation learning in speech processing: challenges, recent advances, and future trends. 2020; arXiv preprint arXiv:2001.00378.
5. Li Y. Research and application of deep learning in image recognition. In: 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA). IEEE; 2022. p. 994–99.
6. Zaidi SSA, Ansari MS, Aslam A, Kanwal N, Asghar M, Lee B. A survey of modern deep learning based object detection models. *Digit Signal Process*. 2022;126:103514.
7. Sharma A, Amrita, Chakraborty S, Kumar S. Named entity recognition in natural language processing: a systematic review. In: Proceedings of second doctoral symposium on computational intelligence: DoSCI 2021. Singapore: Springer; 2022. p. 817–28.
8. Wankhade M, Rao ACS, Kulkarni C. A survey on sentiment analysis methods, applications, and challenges. *Artif Intell Rev*. 2022;55(7):5731–80.
9. Dara S, Dhamercherla S, Jadav SS, Babu CM, Ahsan MJ. Machine learning in drug discovery: a review. *Artif Intell Rev*. 2022;55(3):1947–99.
10. Bhangale KB, Kothandaraman M. Survey of deep learning paradigms for speech processing. *Wirel Pers Commun*. 2022;125(2):1913–49.
11. Chen Y, Mancini M, Zhu X, Akata Z. Semi-supervised and unsupervised deep visual learning: a survey. *IEEE Trans Pattern Anal Mach Intell*. 2022;99:1–23.
12. Pan SJ, Yang Q. A survey on transfer learning. *IEEE Trans Knowl Data Eng*. 2009;22(10):1345–59.
13. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, et al. A comprehensive survey on transfer learning. *Proc IEEE*. 2020;109(1):43–76.
14. Slowik A, Kwasnicka H. Evolutionary algorithms and their applications to engineering problems. *Neural Comput Appl*. 2020;32:12363–79.
15. Liang J, Ban X, Yu K, Qu B, Qiao K, Yue C, et al. A survey on evolutionary constrained multiobjective optimization. *IEEE Trans Evol Comput*. 2022;27(2):201–21.
16. Zhan ZH, Li JY, Zhang J. Evolutionary deep learning: a survey. *Neurocomputing*. 2022;483:42–58.
17. Raji ID, Bello-Salau H, Umoh JJ, Onumanyi AJ, Adegbeye MA, Salawudeen AT. Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models. *Appl Sci*. 2022;12(3):1186.
18. Sethi Y, Jain V, Singh KP, Ojha M. Isomap based self-taught transfer learning for image classification. In: 2017 14th IEEE India Council International Conference (INDICON). IEEE; 2017. p. 1–6.
19. Zhang X, Liu Q, Wang D, Zhao L, Gu N, Maybank S. Self-taught semisupervised dictionary learning with nonnegative constraint. *IEEE Trans Industr Inform*. 2019;16(1):532–43.
20. Kemker R, Kanan C. Self-taught feature learning for hyperspectral image classification. *IEEE Trans Geosci Remote Sens*. 2017;55(5):2693–705.
21. Liu F, Ma J, Zhao R, Wang Q. Online dictionary self-taught learning for hyperspectral image classification. In: 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE; 2018. p. 1–5.

22. Fayyaz M, Hajizadeh-Saffar M, Sabokrou M, Hoseini M, Fathy M. A novel approach for finger vein verification based on self-taught learning. In: 2015 9th Iranian Conference on Machine Vision and Image Processing (MVIP). IEEE; 2015. p. 88–91.
23. Zhao Z, Zhang X, Chen C, Li W, Peng S, Wang J, et al. Semi-supervised self-taught deep learning for finger bones segmentation. In: 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI). IEEE; 2019. p. 1–4.
24. Laghos A, Zaphiris P. Social network analysis of self-taught e-learning communities. *Int J Knowl Learn.* 2007;3(4–5):465–82.
25. Markov K, Matsui T. Nonnegative matrix factorization based self-taught learning with application to music genre classification. In: 2012 IEEE International Workshop on Machine Learning for Signal Processing. IEEE; 2012. p. 1–5.
26. Kamath U, Liu J, Whitaker J, Kamath U, Liu J, Whitaker J. Transfer learning: scenarios, self-taught learning, and multitask learning. In: Kamath U, Liu J, Whitaker J, editors. Deep learning for NLP and speech recognition. Cham: Springer; 2019. p. 463–93.
27. Wang W, Zheng VW, Yu H, Miao C. A survey of zero-shot learning: settings, methods, and applications. *ACM Trans Intell Syst Technol.* 2019;10(2):1–37.
28. Xian Y, Akata Z, Sharma G, Nguyen Q, Hein M, Schiele B. Latent embeddings for zero-shot classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. p. 69–77.
29. Song J, Shen C, Yang Y, Liu Y, Song M. Transductive unbiased embedding for zero-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 1024–33.
30. Kodirov E, Xiang T, Gong S. Semantic autoencoder for zero-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. p. 3174–83.
31. Kodirov E, Xiang T, Fu Z, Gong S. Unsupervised domain adaptation for zero-shot learning. In: Proceedings of the IEEE International Conference on Computer Vision. 2015. p. 2452–60.
32. Fu Y, Xiang T, Jiang YG, Xue X, Sigal L, Gong S. Recent advances in zero-shot recognition: toward data-efficient understanding of visual content. *IEEE Signal Process Mag.* 2018;35(1):112–25.
33. Jiang H, Wang R, Shan S, Chen X. Transferable contrastive network for generalized zero-shot learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. p. 9765–74.
34. Huang H, Wang C, Yu PS, Wang CD. Generative dual adversarial network for generalized zero-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. p. 801–10.
35. Singh UP, Singh KP, Thakur M. NucNormZSL: nuclear norm-based domain adaptation in zero-shot learning. *Neural Comput Appl.* 2022;34:2353–74.
36. Afif M, Ayachi R, Said Y, Atri M. A transfer learning approach for smart home application based on evolutionary algorithms. In: Handbook of research on AI methods and applications in computer engineering. Hershey: IGI Global; 2023. p. 434–50.
37. Hu C, Zeng S, Li C. Hyperparameters adaptive sharing based on transfer learning for scalable GPs. In: 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2022. p. 1–7.
38. Zheng RZ, Zhang Y, Yang K. A transfer learning-based particle swarm optimization algorithm for travelling salesman problem. *J Comput Des Eng.* 2022;9(3):933–48.
39. Eberhart R, Kennedy J. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (Vol. 4). 1995. p. 1942–48.
40. Ghaderzadeh M, Asadi F, Jafari R, Bashash D, Abolghasemi H, Aria M. Deep convolutional neural network-based computer-aided detection system for COVID-19 using multiple lung scans: design and implementation study. *J Med Internet Res.* 2021;23(4):e27468.

41. Van Wieringen WN. Lecture notes on ridge regression. 2015;arXiv preprint arXiv:1509.09169.
42. Zha Z, Wen B, Zhang J, Zhou J, Zhu C. A comparative study for the nuclear norms minimization methods. In: 2019 IEEE International Conference on Image Processing (ICIP). IEEE; 2019. p. 2050–54.
43. Zhu X, Ghahramani Z. Learning from labeled and unlabeled data with label propagation. 2002.

Machine Learning Applications of Evolutionary and Metaheuristic Algorithms



Anupam Yadav and Shrishti Chamoli

Abstract Evolutionary algorithms and other meta-heuristic approaches enjoy widespread popularity when dealing with challenging engineering design problems that prove to be a formidable challenge for conventional optimization methods. In recent times, population-based meta-heuristics and evolutionary algorithms have been successfully used in machine learning problems. This chapter focuses on such machine learning applications using population-based algorithms. This chapter is curated into two parts, the first part discusses the various real-life applications which has single objective solution with meta-heuristics and evolutionary algorithms. The other half describes the use of these algorithms for multi-objective machine learning applications. Parameter tuning, cancel detection, feature selection, clustering, disease prediction, fault diagnosis, price forecasting, and data mining are the thrust areas that are discussed.

Keywords Machine learning · Evolutionary algorithms · Meta-heuristics · Classification · Clustering

1 Introduction

Metaheuristics are advanced algorithmic structures that include a general heuristic method that may be utilized to deal with a broad range of problems. Among the numerous instances of metaheuristics are genetic algorithms (GA), ABC (Artificial Bee Colony), Tabu search, PSO (Particle Swarm Optimization), and ACO (Ant Colony Optimization). Metaheuristic techniques differ from standard techniques which guarantee that the optimal answer will be found in a limited (though often unreasonable) amount of time. Metaheuristics are framed accordingly to discover an approximate and usable answer that is good enough in a small processing time.

A. Yadav (✉) · S. Chamoli

Department of Mathematics and Computing, Dr. B. R. Ambedkar National Institute of Technology Jalandhar, Jalandhar, Punjab, India
e-mail: anupam@nitj.ac.in

Subsequently, they are not prone to combinatorial explosion, which occurs when the computing time necessary to find the optimal solution to an NP-hard issue rises exponentially with problem size. In many cases, especially for intricate problems, metaheuristics might provide a superior trade-off [1] between the quality of the solution as well as processing time. Additionally, metaheuristics are more adaptable than traditional approaches in many ways. For beginners, Metaheuristic algorithms can be adapted to most practical optimization problems, as they are broadly defined in terms of solution quality and computation time. Second, metaheuristics do not place any limitations on objective functions in the optimization problem formulation. This flexibility, however, comes at the expense of significant problem-specific customization to attain high performance.

Evolutionary algorithms (EAs) are one of the types of metaheuristic algorithms. Metaheuristics are broadly categorized as single solution-based metaheuristics and population-based metaheuristics. EAs are classified as population-based metaheuristics.

The need for metaheuristics arises due to the limitations of traditional methods. Traditional methods are simple to use, but selecting the right parameters is sometimes very challenging. These techniques need extra decision-maker involvement, which is frequently subjective. It is typically difficult to select the appropriate parameters. And just one answer is obtained in a single simulation.

Machine learning, particularly data-driven optimization and modeling, has recently become more prevalent in this discipline due to its accuracy while working with large datasets. Data-driven strategies make extensive use of that help accelerate the evolutionary search in practically every component of an EA. Data-driven evolutionary modeling and optimization is an integrative method for solving complicated issues that integrate principles from data science, EAs, and optimization approaches. Data-driven evolutionary modeling and optimization have a wide range of applications, including enhancing the supply chain process, altering hyperparameters of machine learning, building efficient circuits, identifying drug candidates, and many more. This method uses data to direct the optimization technique, resulting in better and more efficient solutions in a variety of fields such as finance, accounting, engineering, the healthcare sector, and others.

Few instances of the applications of metaheuristics in machine learning are: a two phase hybrid model [2] integrating iBPSO (improved-Binary Particle Swarm Optimization) and CFS (Correlation-based Feature Selection) is suggested for cancer classification; using BBHA (Binary Black Hole Algorithm) and BPSO (Binary Particle Swarm Optimization), a hybrid meta-heuristic model [3] was put forward that prioritises gene selection; using IPSO (improved Particle Swarm Optimization) and LSSVM (least square support vector machine), a method [4] is proposed to forecast material price; a unique approach [5] for representing the electrocardiogram (ECG) signal in time-frequency domain has been presented; a new ABC with Q-learning [6] is presented to reduce the maximum tardiness of the 3-stage Assembly Scheduling Problem; Y Khoudifi and M Bahaj [7] used the Fast Correlation-Based Feature Selection approach to enhance the precision of classification of heart

diseases; an attempt [8] was made to reduce classification mistakes by basing CNN (convolution neural networks) classifier's starting weights on the results produced by the ABC technique; a method [9] based on hybrid ML (Machine Learning) model was presented to identify DDoS (Distributed denial of service) attacks and, [10] put forward an automatic pattern classification approach to categorise four diseases-liver condition, breast cancer, diabetes, and hepatitis into two groups. Multi-agent AEFA (Artificial electric field algorithm) [11] is also a notable approach to train multilayer perceptron.

The organization of this chapter is as follows: Section 2 provides a quick overview of the single objective optimization problem. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), PAEFA [12] and Differential Evolution (DE) are a few strategies for dealing with single objective optimization issues discussed in Sect. 3. The pseudocode of each technique is mentioned, and some of their applications in machine learning are discussed. Section 4 focuses on multi-objective optimization. Objectives of Multi-objective optimization are briefly discussed. Section 5 focuses on major methods to deal with MOOP. Following a brief description of certain traditional approaches, various multi-objective EA and their practical uses in machine learning are presented.

1.1 Advantages/Disadvantages of Evolutionary Algorithms Over Classical Algorithms

Evolutionary algorithms, which are inspired by processes of nature, use intricate algorithms to address complicated problems in a variety of fields. They can handle evaluation functions that do not produce a reasonable result within a certain amount of time and are resilient to noisy objective functions. The algorithms are easily adaptable to the problem at hand.

Classical algorithms focus on solving explicit problems and give precise or approximate solutions. In classical algorithms, the search is conducted in the neighborhood of the present solution, which frequently relies on local search techniques. The search approach used is deterministic, that is, it adheres to a set of predetermined set of rules and does not rely on randomness.

Evolutionary algorithms are used to solve implicit problems. Evolutionary algorithms employ a stochastic approach, that is, solutions are selected and generated randomly. They are generative in nature, that is, they produce novel solutions through mechanisms like reproduction, mutation, and crossover. This enables a global search technique that covers a bigger amount of the solution space. Capable of locating global optima in intricate search spaces, and dealing with noisy or dynamic surroundings, evolutionary algorithms keep a diversified population of solutions, avoiding getting caught in local optima.

Because evolutionary algorithms are stochastic, it can be challenging to reproduce results, which limits their applicability in situations that demand a deterministic approach to solutions. The results of evolutionary algorithms can vary depending on the initial population as well as the strategy used for selection, and they are not always guaranteed to identify the global optimal solution. The convergence of evolutionary algorithms can be tardy, particularly for difficult problems with wide search areas.

2 Single-Objective Optimization Problems in Machine Learning

Finding the optimum solution, which is the lowest or largest value of a single-objective function that integrates all feasible objectives into one, is single-objective optimization's primary objective. This type of optimization can offer decision-makers an insight into the problem's nature in question, but it is rarely able to deliver a set of actual solutions that trade-off many goals.

2.1 Definition

Single-objective optimization refers to an optimization problem where we have to maximize or minimize the objective function based on a single variable given a constraint or an unconstrained situation. Equation (1) shows the formula of the SOOP in its general form.

2.2 Formulation

$$\begin{aligned}
 & \text{Minimize/Maximize} && h(u) \\
 & \text{subject to} && f_a(u) \geq 0, \quad a = 1, 2, \dots, A; \\
 & && g_b(u) = 0, \quad b = 1, 2, \dots, B; \\
 & && u_m^{(L)} \leq u_m \leq u_m^{(U)}, \quad m = 1, 2, \dots, M.
 \end{aligned} \tag{1}$$

In this context, the vector $u = (u_1, u_2, \dots, u_m)^T$ represents a collection of n decision variables, with each variable u_i subject to limitations imposed by the final set of variable boundaries. These constraints confine each u_i to a specific range defined by

a lower bound $u_m^{(L)}$ and an upper bound $u_m^{(U)}$. D , the decision variable space, is made up of these boundaries. Expressions $f_a(u)$ and $g_b(u)$ are the constrained functions. There are several metaheuristic methods to deal with single-objective optimization problems, few popular metaheuristics and EAs are discussed along with their applications in machine learning:

2.3 *Particle Swarm Optimization (PSO)*

PSO is a population-based metaheuristic optimization algorithm. Kennedy and Eberhart [13] first proposed PSO in 1995. The method can be viewed as a distributive cognitive algorithm that executes a multidimensional search (in its more general variant). To increase the performance of PSO, many variants have been developed throughout time. In [14], Karbassi Yazdi et al. developed a Binary PSO variation (BPSO) for effectively resolving the issue of ship scheduling and routing in the extraction, transportation, and regasification of liquefied natural gas (LNG). To tackle combinatorial optimization issues, Xu et al. introduced a novel Chaotic Search PSO algorithm (CS-PSO) in [15]. By including the CS technique, this strategy enhances the PSO algorithm. Zhang et al. [16] created a multi-objective particle swarm optimizer based on an adversarial process, in which the current swarm participates in paired contests at each iteration, which are subsequently utilized to update the particle positions. A generic algorithm of PSO is presented in Algorithm 1.

2.3.1 Applications of PSO in Machine Learning

Several applications of PSO can be found in the literature which includes but are not limited to healthcare, the environment, business, industry, smart cities, and more generally in areas like service allocation, segmenting images, security management, and prediction. Some notable and important applications are discussed here:

Algorithm 1 PSO Algorithm

```

1. for  $j = 1$  to  $N$            { $N$  = Size of Population}
2.   Randomly set up  $Q(j)$       {Population of particles =  $Q$ }
3.   Set up  $v(j) = 0$           {Speed of each particle =  $v$ }
4.   Evaluate  $Q(j)$ 
5.    $Gbest$  = Best particle discovered in  $Q(j)$ 
6. End
7. for  $j = 1$  to  $N$ 
8.    $Pbest(j) = Q(j)$           {Set up the memory of each particle}
9. End
10. Repeat
11.   for  $j = 1$  to  $N$ 
12.      $v(j) = g \times v(j) + A_1 \times B_1 \times (Pbest(j) - Q(j)) + A_2 \times B_2 \times$ 
         $(Pbest(Gbest) - Q(j))$           {Calculate each particle's speed}
13.     {inertia weight =  $g$ ,  $A_1$  and  $A_2$  are positive constants and  $B_1$  and  $B_2$  are
        taken randomly in  $[0,1]$ }
14.      $QoQ(j) = Q(i) + v(j)$ 
15.     A particle is reintegrated to its limits if it exits the predetermined
        hypercube.
16.     Evaluate  $Q(j)$ 
17.     If the new best position is better than  $Pbest(j) = P(j)$ 
         $Gbest$  = Best particle discovered in  $Q(j)$ 
18. End
19. Till termination requirement is satisfied

```

1. **Cancer Classification:** For the classification of cancer, Indu et al. [2] suggested a two-phase hybrid model that combines improved binary Particle Swarm Optimization (iBPSO) and Correlation-based Feature Selection (CFS). Using a Naive-Bayes classifier with stratified tenfold cross-validation, this model identifies a low-dimensional collection of prognostic genes to categorize biological samples of binary and multi-class tumors. The suggested methodology was tested on 11 standard microarray datasets from various cancer types. When the experimental findings were compared to those of seven other well-known approaches, the proposed model outperformed them regarding both precision in classification and the number of chosen genes in the majority of situations.
2. **Gene Selection:** A hybrid meta-heuristic model that prioritizes gene selection, which plays an important role in developing a successful system to diagnose diseases for microarray data, was put forth by Pashae et al. [3] using the Binary Black Hole Algorithm (BBHA) and Binary Particle Swarm Optimisation (BPSO) (4-2). Two benchmark microarrays and three clinical microarrays were used to assess the performance of the suggested technique. The experimental results and statistical analysis demonstrate that when it comes to avoiding local minima, converging speed, accuracy, and number of chosen genes, the BPSO (4-2)-BBHA surpasses the BBHA, the BPSO (4-2), and numerous cutting-edge

approaches. The results further show that clinical datasets may be successfully used to locate known physiologically and statistically significant genes using the BPSO (4-2)-BBHA technique.

3. **Nuclear Power Plant Fault Diagnosis:** Wang et al. [17] presented the outcomes of their research on hybrid fault identification approaches. The suggested technique improves the accuracy of process fault diagnostics significantly. More significantly, this technology may be linked to the current computerized operator support system and expanded to identify issues in complex thermal plants and other nuclear power plants (NPP) components to ensure system safety and dependability.
4. **Material Price Forecasting:** Tang et al. [4] proposed a method for material price forecasting based on improved PSO (IPSO) and least square support vector machine (LSSVM). The price prediction model with LSSVM is used to predict future material prices after being trained and tested. Because the typical PSO's optimization method is prone to premature convergence and local optimum trapping, the average particle distance and fitness variance are incorporated. Utilizing MATLAB, a simulation evaluation is performed, and both the algorithm's physical and virtual performance are assessed. Comparing the suggested technique to other conventional algorithms like neural networks and time series prediction, it has the benefits of high prediction accuracy, quick convergence rate, and strong generalization ability. As a result, it has real application value and can predict construction material costs with reasonable accuracy as well as the optimal time to make a purchase.
5. **Appliance Scheduling:** In addressing the task of scheduling household appliances within apartment settings, an appliance scheduling model is introduced for the Home Energy Management System (HEMS). This model is centered on day-ahead electricity trading and incorporates photovoltaic (PV) generation. The study was put forth to minimize the discontentment of consumers, payments of electricity, and CO₂ emissions. Ma et al. [18] adopt a cooperative multi-swarm PSO to tackle the combinational optimization problem. Simulation shows that cooperative multi-swarm PSO performs well in terms of convergence under various conditions.
6. **ECG Signals Classification:** To analyze ECG signals, signal processing techniques are a crystal clear choice. Traditional signal processing approaches are incapable of dealing with the ECG signal's non-stationary character. Raj and Ray [5], in this respect, introduce a novel method, the discrete orthogonal stockwell transform, which effectively represents the ECG signal in time-frequency space. ECG data from the benchmark MIT-BIH arrhythmia dataset displaying 16 classes of the most commonly occurring arrhythmic events are used to validate the suggested technique. When compared to current methodologies in the literature, the experimental findings produced an enhanced overall precision, sensitivity (Sp), and positive predictivity (Pp) of 98.82%.

2.4 Ant Colony Optimization (ACO)

ACO is a metaheuristic approach that operates on a population-based principle. This metaheuristic, conceived by Marco Dorigo, draws inspiration from the foraging behavior of ants [19–21]. Central to this behavior lies the indirect communication of ants through chemical pheromone trails, enabling them to discover efficient routes between their colony and sources of food. Gambardella and Dorigo [22] created the Ant-Q family of algorithms after realizing that the AS might be regarded as a specific type of distributed learning technique. The AS and Q-learning are actually hybridized in this family of algorithms. In Algorithm 2, [23] a pseudocode of ACO is presented.

2.4.1 Applications of ACO in Machine Learning

Initially employed to address the infamous traveling salesperson problem, ACO is now used in nanoelectronics, image processing, antenna optimization and synthesis, data mining, system identification, intelligent testing systems, etc. Some good applications are collected here:

- 1. Chronic Kidney Disease Prediction:** In the present circumstances, chronic kidney disease (CKD) is regarded as a significant threat to public health. Despite having constraints in the field of optimization, Extreme Learning Machine (ELM) combined with various feature mapping approaches can be implied directly for classification purposes. S.B V.J Sara and Kalaiselvi [24] proposed an ACO based ELM which is used to estimate the likelihood of CKD. The study involves gathering data from 400 patients, including both missing and noisy information. Among these patients, 250 have chronic kidney disease (CKD), while the remaining 150 do not have CKD. To identify a unique CKD employing the ELM technique with the lowest computation interval and best possible accuracy, a cost-accuracy trade-off research is completed.
- 2. Heart Disease Classification:** In today's environment, cardiovascular disease (CVD) is on the rise. Machine learning finds practical application in the prediction of the probability of heart disease. Y Khourdifi and M Bahaj [7] employed the Fast Correlation-Based Feature Selection (FCBF) method to eliminate irrelevant features, thereby enhancing the accuracy of the categorization of cardiovascular disease. The classification was performed using various algorithms such as SVM (Support Vector Machine), Naive Bayes, K-Nearest Neighbour, Random Forest, and a Multilayer Perception—ANN optimized using PSO and ACO. The hybrid approach under consideration is applied to a dataset related to heart disease. The findings show that the recommended hybrid strategy is reliable and efficient in analyzing various types of data to identify heart disease. As a consequence, this study evaluates several machine learning algorithms and

compares the outcomes using different performance indicators, such as accuracy, precision, recall, f1-score, and so on. Using the optimized model proposed by FCBF, PSO, and ACO, a maximum classification accuracy of 99.65% is achieved.

3. **Multi-Label Feature Selection:** A brand-new Ant Colony Optimisation (ACO)-based multi-label feature selection technique was put out by Paniri et al. [25]. The suggested method significantly differs from all ACO-based feature selection techniques in which it adopts a heuristic learning strategy rather than a static heuristic function. On various data sets, numerous experiments were conducted and classification performance was compared. The findings reveal that the suggested strategy greatly outperforms competing methods.
4. **Filter-Wrapper Feature Selection Method:** To mitigate computational complexity, Ghosh et al. [26] introduce a combination of ACO in the form of a wrapper-filter approach. They integrate subset evaluation by employing a filter method rather than a wrapper method. Their proposed approach was assessed using K-nearest neighbors and multi-layer perceptron classifiers on real-world datasets sourced from the UCI Machine Learning repository and the NIPS2003 FS challenge. The experimental outcomes were contrasted with several well-known Feature Selection methods. The comparison of outcomes demonstrates that the technique outperforms the majority of the standard FS algorithms. The suggested model was further examined using face emotion detection and microarray datasets in order to gauge its robustness.
5. **Crop Administration:** PAG (Precision Agriculture) helps productivity by administering the health of crops and soil. Usually, traditional recommender systems are used by agriculturists for farming. Mythili and Rangaraj [27] put forward a DLT (Deep Learning Technique) based recommender system that uses collected historical data of climate and crops for its recommendations. The suggested scheme is a hybrid approach that uses ACO that optimizes DCNN (Deep convolution Neural Networks) and LSTM (Long Short Term Memory) network inputs known as (ACO-IDCNN-LSTM) for the prediction of crops. The recommender system generated acceptable results.

Algorithm 2 ACO Algorithm

```

1. Randomly set up  $R(a, b)$ 
2. for  $j = 1$  to  $M$            { $M = \text{total episodes}$ }
3.     for  $j = 1$  to  $n$          { $n = \text{total agents}$ }
4.         For the  $n$  agents, set up  $a = a_0$ 
5.         Repeat each of the episode's  $g$  steps
6.             for  $j = 1$  to  $n$ 
7.                 Select  $b$  in  $a$  using the action selection rule
                    
$$a' = \{\arg\max_{a'} C(a, a')\} \quad \text{if } s \leq$$

                    
$$s_0 \quad a_{rand} \quad \text{otherwise}$$

8.                 Apply  $b$  and observe  $p, a'$ 
9.                  $R(a, b) \leftarrow R(a, b) + \gamma[\beta \max_{a'} R(a', b') - R(a, b)]$ 
10.                 $a \leftarrow a'$ 
11.            End
12.        Till  $a$  is final stage
13.    End
14. Comparing  $n$  options, Choose the best one
15. For every  $R(a, b)$  in the optimal solution
16.      $R(a, b) \leftarrow R(a, b) + \gamma[p + \beta \max_{a'} R(a', b') - R(a, b)]$ 
17.     {(Learning step =  $\gamma$ )}
18.     {(Discount factor =  $\beta$ )}
19. End
20. Report best solution found

```

2.5 Artificial Bee Colony (ABC)

The Artificial Bee Colony Algorithm [28], which draws its inspiration from the swarming behavior of bee colonies, randomly visits multiple points in the solution space before settling on one. Having the ability to communicate information, memorize the environment, store and disseminate knowledge, and base choices on that knowledge are only a few of the traits of the honey bee swarm. The swarm adjusts to environmental changes by dynamically allocating tasks and advancing through social learning and teaching. A pseudocode for ABC is provided in Algorithm 3.

2.5.1 Applications of ABC in Machine Learning

- 1. Pattern Recognition:** One of the most effective deep learning approaches for handling various pattern recognition problems is convolution neural networks (CNN). However, CNNs are vulnerable to multiple local optima, much like the majority of ANNs (artificial neural networks). Thus, enhancement of the CNNs is necessary to prevent being trapped inside the local optima. This study suggests

the ABC method as a different technique for improving CNN performance. By basing the CNN classifier's starting weights on the results produced by the ABC technique, Anan [8] attempts to reduce classification mistakes. Furthermore, when working with big training datasets, the distributed ABC is provided as a solution to reduce the amount of time required to complete the process. The experiment findings show that the suggested strategy can increase the functioning of standard CNNs in the context of both accuracy in recognition and processing time.

Algorithm 3 ABC Algorithm

-
1. Scout bees should get to the primary food sources. Here, the worker bee, whose food supply has been used up by the other bees, is transformed into a scout bee.
 2. Repeat
 3. Worker bees are sent to the source of food to determine how much nectar is needed. Here, the number of worker bees = the number of sources of food in neighborhood of hive.
 4. Calculate the likelihood value of favored sources by the spectator bees.
 5. Spectator bees are dispatched to food sources to determine the nectar quantity
 6. End exploitation process of the sources that the bees have used up.
 7. Scout bees are deployed randomly within the search area to discover new food sources.
 8. Remember the finest source of food discovered till now.
 9. Till (conditions have been satisfied)
-

2. **Diabetic Retinopathy:** Retinal fundus image analysis (RFIA) of diabetic retinopathy pictures is frequently used to evaluate the risk of visual disabilities in patients with diabetes. In order to collect segmented retinal features, the ADL-CNN architecture of CNN was created. In [29], The image is segmented into segments using the ABC approach as part of the Active Deep Learning (ADL) system preparation, using a threshold value selected based on the outcomes of the histogram of images. This makes it simpler to detect retinal lesions. The performance of the ADL-CNN model is evaluated on the identical dataset by comparing it with some of the state-of-the-art methods. The suggested technique performs well in recognizing DR lesions and estimating the severity level in diverse fundus pictures.
3. **Clustering:** In [30], a novel clustering method called CEABC (Comprehensively Enhanced Artificial Bee Colony method), which can be utilized to address clustering problems, is presented. It is an unsupervised learning problem. It is advanced by many operators, including K-means, and PSO, and is driven by the Gbest mechanism. A test employing two fictitious data sets and four of the most typical UCI machine learning data sets confirmed the newly put forward CEABC algorithm's clustering performance. The experimental findings are also contrasted with those of the conventional ABC algorithm, a number of

other recently suggested ABC-improved algorithms, and traditional clustering techniques. The results unequivocally indicate that the novel ABC method introduced in this research is a more effective clustering algorithm, as it addresses clustering challenges with higher precision and stability.

4. **Diabetes and Liver Disease Diagnosis:** To determine how the removal of outdated and unnecessary dataset properties affects the efficacy of categorization using the SVM classifier, Uzer et al. [31] offer a hybrid technique. The established method is often employed in the diagnosis of diabetes and liver illnesses, both of which are frequently seen and have a negative impact on quality of life. It is a supervised learning problem. Hepatitis, liver problems, and diabetes datasets from the UCI dataset were employed for the diagnosis of these diseases, and the suggested approach achieved classification accuracy rates of 94.92%, 74.81%, and 79.29%, respectively. The categorization precision for these sets of data was evaluated using the tenfold cross-validation method. The outcomes demonstrate that the method's performance is quite effective when compared to other findings obtained and appears to be highly promising for pattern recognition applications.
5. **Assembly Scheduling Problem:** The combination of reinforcement learning with meta-heuristics can significantly increase meta-heuristic performance and tackle the distributed 3-stage assembly scheduling problem. In the proposed study, a mathematical model is presented for the distributed 3-stage assembly scheduling issue with $DP_m \rightarrow 1$ layout and maintenance at all three levels. To reduce the maximum tardiness of the three-stage Assembly Scheduling Problem (ASP), a novel artificial bee colony with Q-learning (QABC) is presented [6].
6. **Feature Subset Selection:** Feature subset selection eliminates noise and unnecessary features so that optimum feature subsets can be selected and precision is increased. An abundance of features within the dataset escalates computational complexity, leading to a decline in performance. To address this issue, the problem of feature subset selection is transformed into a four-dimensional continuous optimization task, as outlined by Yang et al. in their 2016 work [32], utilizing the angle modulation technique instead of representing it as a high-dimensional binary vector. To showcase the effectiveness of presenting the problem through angle modulation and to assess the performance of the proposed approach, six different versions of ABC algorithms employ angle modulation for feature selection. The utilization of Angle Modulated ABC techniques enhanced the classification accuracy with reduced feature subsets, as demonstrated through the results of experiments conducted on six high-dimensional datasets.

2.6 Differential Evolution (DE)

Differential Evolution (DE) [33], a contemporary heuristic intended to optimize issues spanning continuous domains, was put out by Kenneth Price and Rainer

Storn. DE is comparable to conventional EAs. However, unlike a straightforward genetic algorithm, it does not employ binary encoding, and unlike an evolution strategy, It does not self-adapt to the parameters using a probability density function. Instead, DE carries out mutation depending on how the solutions are distributed throughout the present population. In this approach, the positioning of the individuals chosen for estimating the mutation values has an impact on the search directions and potential step sizes. An algorithm of one of the variants of DE, DE/rand/1/bin, is presented in Algorithm 4 [23]. Within the context of “rand/1/bin”, “DE” signifies Differential Evolution. The term “rand” implies that the selection of individuals for calculating mutation values is done randomly. The “1” represents the number of pairs of individuals selected, and lastly, “bin” denotes the utilization of binomial crossover.

Algorithm 4 DE/rand/1/bin Algorithm

```

1. Begin
2.  $R = 0$ 
3. Randomly initialize a population  $p_{j,R} \quad \forall j, j = 1, \dots, B$ 
4. Evaluate  $g(p_{j,R}) \quad \forall j = 1, \dots, B$ 
5. for  $R = 1$  to  $Max$  do
6.   for  $j = 1$  to  $B$  do
7.     Randomly select  $a_1 \neq a_2 \neq a_3 :$ 
8.      $i_{rand} = randint(1, C)$ 
9.     for  $i = 1$  to  $C$  do
10.       if  $(rand_i[0,1] < D \text{ or } i = i_{rand})$  then
11.          $v_{j,i,R+1} = p_{a_3,i,r} + E(p_{a_1,i,r} - p_{a_2,i,r})$ 
12.       else
13.          $v_{j,i,R+1} = p_{j,i,r}$ 
14.       end
15.     end
16.     if  $(g(v_j, R + 1) \leq g(p_j, R))$  then
17.        $p_{j,R+1} = v_{j,R+1}$ 
18.     else
19.        $p_{j,R+1} = p_{j,R}$ 
20.     end
21.   end
22.    $R = R + 1$ 
23. end
24. end

```

2.6.1 Applications of DE in Machine Learning

- 1. Recognition of Invasive Species:** The expansion of invasive species is one of the primary results of variations in the climate, which poses a major and increasingly growing danger to ecology. Despite their significant biological differences, these

species do not appear to have significant physical distinctions. As a result, their identification is frequently challenging. Konstantinos and Lazaros [34] have created an advanced computer vision system designed to identify invasive or unfamiliar species by analyzing their physical characteristics. This research introduces an innovative Extreme Learning Machine (ELM) model, which has been optimized using the Adaptive Elitist Differential Evolution (AEDE) approach. The AEDE method represents an enhanced version of the Differential Evolution (DE) technique, specifically tailored for handling large datasets.

2. **Denoising Medical Images:** Given the inherent noise in the image generation process, the task of image denoising plays a pivotal role in various computer vision and image processing techniques. The realistic noise in medical pictures is often heterogeneous, complicated, and unexpected, making it challenging to construct an efficient denoising network. Rajesh and Sushil [35] built an autonomous network evolution model based on Differential Evolution (DE) to optimize network designs and hyperparameters by finding the fittest parameters. On various medical image datasets, the suggested model was tested. The results obtained at various noise levels demonstrate the ability of the proposed model termed DEvoNet to determine the ideal parameters for developing a high-performance denoising network structure.
3. **Ovarian Cancer Detection:** Ovarian cancer ranks as one of the most prevalent causes of mortality among women. Monitoring pertinent signs in the patient is the most frequently used method for diagnosing ovarian cancer. Despite the focus on biomarkers, aspects physiological in nature may be included in feature subsets. To identify the incidence of ovarian cancer, Filbert et al. [36] offer an ideal simultaneous feature weighting/parameter optimization using Adaptive Differential Evolution (ADE). The suggested strategy successfully condenses the characteristics with encouraging outcomes, leading to a less complicated classification scheme for ovarian cancer detection.
4. **Hyperparameter Tuning of Supervised Learning Algorithms for Classification Tasks:** Schmidt et al. [37] examine how well DE performs when tuning supervised learning systems' hyperparameters for classification tasks. This empirical research compares the effectiveness of DE with Sequential Model based Algorithm Configuration (SMAC), a reference Bayesian Optimisation technique, using a variety of different machine learning algorithms and datasets with varying properties. The results show that when tweaking a given machine learning method, Differential Evolution performs better than SMAC for most datasets, especially when breaking ties in a first-to-report manner.
5. **Detection of DDoS Attacks:** The effective detection of distributed denial of service (DDoS) attacks, a critical challenge in the field of cloud computing, is made possible by machine learning frameworks. Kushwah and Ranga [9] present a hybrid machine learning model-based method to identify DDoS attacks. The effectiveness of the suggested attack detection system is assessed using three cutting-edge datasets: NSL-KDD, ISCX IDS 2012, and CIDDS-001. The rest of the chapter focuses on multi-objective optimization problems and machine learning applications.

3 Multi-objective Optimization Problem

The majority of real-world issues are typically characterized by multi-objective characteristics, i.e., they call for the concurrent optimization of many objective functions. The MOOP does not possess one optimum solution, in contrast to the single-objective optimization problem. Instead, it yields a collection of ideal solutions that illustrate various trade-offs between the objectives that make up a set of Pareto optimal solutions. The Pareto optimum front is a representation of the Pareto optimal solution set in function space.

3.1 Definition

The enhancement of one aim occurs at the price of another in MOOPs because there are two or more optimization goals involved that are at odds with one another. In order for MOOP to succeed, a variety of various and frequently competing goals must all be met at once.

Multi-objective optimization refers to an optimization problem where we must simultaneously optimize at least two objective functions.

3.2 Formulation

$$\begin{aligned}
 & \text{Minimize/Maximize} && h_n(u), \quad n = 1, 2, \dots, N; \\
 & \text{subject to} && f_a(u) \geq 0, \quad a = 1, 2, \dots, A; \\
 & && g_b(u) = 0, \quad b = 1, 2, \dots, B; \\
 & && u_m^{(L)} \leq u_m \leq u_m^{(U)}, \quad m = 1, 2, \dots, M.
 \end{aligned} \tag{2}$$

A solution $u = (u_1, u_2, \dots, u_m)^T$ represents a collection of n decision variables, with each decision variable u_i subject to restrictions imposed by the last set of constraints, known as variable boundaries, to a range that lies within a lower bound $u_m^{(L)}$ and an upper bound $u_m^{(U)}$. The decision variable space D is made up of these boundaries. Expressions $f_a(u)$ and $g_b(u)$ are the constrained functions.

3.3 Objectives in Multi-objective Optimization

Two objectives should be assessed when designing MOEAs: (1) minimizing the distance between feasible solutions and the solution which is Pareto optimal

(convergence) and (2) increasing the diversity of feasible solutions. However, it is typically believed that these two objectives are at odds with one another, i.e., conflicting. Maintaining an appropriate balance between convergence and variety in many-objective optimization is the key challenge addressed by a significant number of Pareto dominance-based multi-objective evolutionary algorithms.

4 Major Methods to Deal with MOOP

4.1 Classical Methods

Finding a single trade-off solution after reducing the multi-objective optimization problem to a single objective optimization problem is the most popular method for locating Pareto optimal solutions. The weighted sum method, $\in-$ constraint method, weighted metric methods, goal programming, value function method, etc. are just a few of the approaches that can be used to change a MOOP into a SOOP. These techniques are known as classical techniques. There have been traditional multi-objective optimization techniques for at least the last forty years. Below is a discussion of a few traditional ways.

4.1.1 Weighted Sum Method

The weighted sum technique is the most basic and widely used conventional method [1]. By pre-multiplying each aim with a user-supplied weight, the approach scalarizes a series of objectives into only one objective.

$$\begin{aligned} \text{Minimize} \quad & H(u) = \sum_{n=1}^N w_n h_n(u), \\ \text{subject to} \quad & f_a(u) \geq 0, \quad a = 1, 2, \dots, A; \\ & g_c(u) = 0, \quad c = 1, 2, \dots, C; \\ & u_b^{(L)} \leq u_b \leq u_b^{(U)}, \quad b = 1, 2, \dots, B. \end{aligned} \tag{3}$$

Here, the weight of the b -th objective function is $w_n \in [0, 1]$. An objective's weight is often determined in accordance with its relative significance to the situation.

4.1.2 $\in-$ Constraint Method

In this approach, the MOOP is reformed by preserving only one aim while limiting the other objectives to values that the user specifies [38]. The amended problem is as follows:

$$\begin{aligned}
& \text{Minimize} && h_\mu(u), \\
& \text{subject to} && h_n(u) \leq \in_n \quad n = 1, 2, \dots, N \text{ and } n \neq \mu \\
& && f_a(u) \geq 0, \quad a = 1, 2, \dots, A; \\
& && g_b(u) = 0, \quad b = 1, 2, \dots, B; \\
& && u_m^{(L)} \leq u_m \leq u_m^{(U)}, \quad m = 1, 2, \dots, M.
\end{aligned} \tag{4}$$

The parameter \in_n does not always denote a small value close to zero but rather the upper bound of the value of h_n . For each given upper bound vector $\in = (\in_1, \dots, \in_\mu - 1, \in_\mu + 1, \dots, \in_N)^T$, the unique solution is Pareto-optimal for the \in -constraint problem.

4.1.3 Weighted Metric Method

By using weighted metrics like l_p and l_{inf} distance metrics, a problem with several objectives is reduced to a single objective problem in this approach. We can minimise the weighted l_p distance measure between any solution u and the ideal solution v as follows:

$$\begin{aligned}
& \text{Minimize} && l_p(u) = \left(\sum_{n=1}^N w_n |h_n(u) - v|^p \right)^{\frac{1}{p}}, \\
& \text{subject to} && f_a(u) \geq 0, \quad a = 1, 2, \dots, A; \\
& && g_b(u) = 0, \quad b = 1, 2, \dots, B; \\
& && u_m^{(L)} \leq u_m \leq u_m^{(U)}, \quad m = 1, 2, \dots, M.
\end{aligned} \tag{5}$$

5 Applications of Multi-objective Evolutionary Algorithms in Machine Learning

5.1 Nondominated Sorting Genetic Algorithm II (NSGA-II)

Since 1989, genetic algorithms (GAs) [39] built on evolutionary concepts have been widely employed in large-scale computer tools for optimization, design, simulation, process management, and so on. Such approaches have various applications in research, health, social sciences, sustainability, industry, manufacturing, transportation, technology, accounting, and many other disciplines, and their popularity appears to be growing with time. GAs can be used as an optimization, training, modeling, design, or procedure control tool for both static and dynamic applications. In [39], Paszkowicz conducted a bibliometric study of the scientific literature to show how GAs have evolved into a variety of applications. Because early

evolutionary algorithms lacked elitism and leaned on a sharing parameter to maintain a diversified Pareto set, NSGA-II [40] was created as a remedy. The NSGA-II incorporates crowded comparison, elitism, sharing, and a fast non-dominated sorting algorithm, building upon the original NSGA's framework. It starts by forming a cohort of competing individuals, evaluating and classifying them according to their non-dominance level. Subsequently, it employs evolutionary procedures to generate a fresh pool of offspring, combines these offspring with the parent individuals, and then organizes the merged population into different tiers or fronts. Furthermore, the NSGA-II implements a niching strategy by assigning a crowding distance to each member. This crowding distance is strategically used in the selection process to ensure that every individual maintains a minimum separation, thereby promoting diversity within the population and assisting the algorithm in its exploration of the fitness landscape. It addresses the drawbacks of NSGA-I [41], which lacks elitism, requires the sharing parameter to be specified, and has a high computational complexity of (MN^3) . Elitism indicates that in the current iteration, the best answers from the prior one are maintained. The algorithm's rate of convergence is accelerated dramatically as a result. Its usage of a quick non-dominated sorting algorithm also significantly reduces the complexity of its computations. A generic algorithm of NSGA-II is provided in Algorithm 5. MONNA [42] is also a nondominated sorting-based algorithm for solving multi-objective optimization problems.

Algorithm 5 NSGA-II Algorithm

-
1. NSGA-II ($N, t, f_i(x_i)$) {To solve $f_i(x)$, N members went through t generations}
 2. Set up the population P_0 .
 3. Randomly size the population N
 4. Evaluate the values of objectives
 5. Rank is assigned using dominance depth method
 6. Create offspring
 7. Crowded binary tournament selection
 8. Merge population
 9. Perform mutation
 10. for $j = 1$ to t do
 11. for each parent and each offspring in P_0 do
 12. Rank is assigned using dominance depth method
 13. Along $P'F$ produce sets of non-dominated vectors
 14. until N individuals are located, loop (within) by addition of solutions to the following generation beginning with the first front
 15. End
 16. Choose elite points outside a crowding distance on the lower front (i.e., those whose rank is lower)
 17. Produce the following generation
 18. Crowded binary tournament selection
 19. Merge population
 20. Perform mutation
 21. End
 22. End
-

5.1.1 Applications of NSGA-II in Machine Learning

1. **Aerogel Glazing System:** In their research, Zhou and Zheng [43] investigate the heat transfer, solar radiation transmission, and indoor lighting properties of an aerogel glazing system. They employ a computational model that has been rigorously validated through experimental data. The study uncovers a notable energy-related conflict between the direct sunlight's impact on indoor illuminance and the heat gain within the space. It offers valuable insights into the usage of aerogel glazing systems in subtropical climates, along with suitable solutions. The primary goal of this study is to gain a deeper understanding of the processes involving heat and solar radiation transmission through nanoporous aerogel particles and to identify the optimal multi-variable configuration for designing and operating a reliable system. It provides an overarching framework and technical guidance for a new multi-objective optimization approach.
2. **Data Categorization:** DL (Deep Learning) models are used extensively for classification tasks but they are completely deterministic, sensitive to noise, and unable to deal with vague data. In such high-dimensional situations, DL models need a significant quantity of data, which often increases exponentially with the number of features. Low performance in tasks of classification is caused by DL model issues with irrelevant features and data ambiguity. An optimized fuzzy deep learning (OFDL) model for data categorization based on NSGA-II is proposed by Yazdinezad et al. When compared to fuzzy classifiers, OFDL performs well when it comes to F-measure, recall, accuracy, precision, and True Positive Rate (TPR). Furthermore, compared to prior fuzzy DNN models, OFDL performs classification tasks more accurately [44].
3. **Automated Pattern Classification for Disease Diagnosis:** Patterns in medical diagnosis include visible symptoms and the findings of diagnostic tests, each of which has a range of risks and expenses. An automatic pattern classification approach was proposed by Zangooei et al. [10] for categorizing four diseases—liver conditions, breast cancer, diabetes, and hepatitis into two groups. The study's findings are contrasted with those from a few other investigations that used the identical UCI machine learning dataset to detect four different illnesses. The outcomes of the experiments demonstrate that the suggested approach produces an exceptional and competitive outcome in all four of these real-world datasets.
4. **Atkinson Cycle Engines:** The study of Tong et al. [45] marks the pioneering effort in creating digital twins for an Atkinson cycle gasoline engine compliant with China VI emissions regulations. They employ the GT-Power software and multi-objective evolutionary optimization (MOEO) in this innovative study. Their work not only lays a robust theoretical groundwork but also provides digital model support for the development of efficient and energy-conserving

Atkinson cycle engines, thus promoting the integration of these engines in environmentally friendly vehicles.

5. **COVID-19 Screening:** COVID-19, being a highly contagious inter-person disease, requires adequate tools for screening to provide results in a short time. In their study, Soui et al. [46] propose a prediction algorithm that differentiates COVID-19 cases with infections, relying on clinical symptoms and characteristics, to reduce the risk of transmission. This model offers a practical means for individuals to identify potential infections without the need for hospital visits. To select the most relevant features, NSGA-II is employed to strike the optimal balance between two competing objectives: minimizing the number of features while maximizing the importance of selected characteristics. The next step is classifying the data employing an AdaBoost classifier. Two separate datasets are used to assess the proposed model. It displays better categorization results than the current techniques.

5.2 *Niched-Pareto Genetic Algorithm (NPGA)*

It is a multi-objective EA for tournament selection that is based on Pareto dominance. A subpopulation, typically constituting approximately 10% of the total population, is pitted against a considerable number of randomly selected individuals, often involving two candidates simultaneously. In the event that one of these candidates is dominated by randomly selected individuals from the population and the other is not, the undominated one emerges as the winner. In cases of a tie between the competitors, the outcome of the competition is determined using fitness sharing. A pseudocode of NPGA is presented in Algorithm 6.

Algorithm 6 NPGA Algorithm

```

Algorithm 6 NPGA Algorithm

1. NPGA ( $N, t, f_i(x_i)$ ) {To solve  $f_i(x)$ ,  $N$  members went through  $t$  generations}
2. Set up the population  $P_0$ .
3. Evaluate the values of objectives
4. for  $i = 1$  to  $t$  do
5.     Perform specialized tournament selection
6.     Start
7.         if Only Competitor  $A$  is dominated then
8.             Select Competitor  $B$ 
9.         else if Only Competitor  $B$  is dominated then
10.            Select Competitor  $A$ 
11.         else if Both Competitors are tied then
12.             Carrying out fitness sharing for result
13.             Return Competitor having lesser niche count
14.         End
15.     End
16.     Perform crossover (one point)
17.     Perform mutation
18.     Calculate the value of objectives
19. End for
20. End

```

5.2.1 Applications of NPGA in Machine Learning

- 1. Feature Selection for Nuclear Transients Classification:** The difficulty of selecting among numerous monitored characteristics to employ for quickly recognizing evolving transient patterns is known as feature selection for transient classification. The adoption of “on condition” diagnostic procedures in complex systems, such as nuclear power plants where numerous attributes are measured, is a crucial concern. It has been demonstrated that noisy and irrelevant attributes needlessly augment the complexity of classification and diminish diagnostic accuracy. Using multi-objective evolutionary algorithms, Baraldi et al. [47] take on the challenge of choosing the characteristics to be applied for effective transient categorization. The approach is assessed using a diagnostic problem that involves a wealth of process parameters. It aims to classify simulated transients occurring in the feedwater system of a boiling water reactor.
- 2. Extracting Multi-objective Rules from Large Datasets:** For the purpose of extracting multi-objective rules from huge databases, an effective genetic algorithm is presented. The improved niched NPGA (INPGA) that Lu et al. [48] suggest combines BNPGA and SDNPGA to accurately pick candidates while also speeding up the selection process. They suggested a clustering-based sampling strategy since the influence of the selection operator depends on the

samples, and also took the case of a zero niche count into consideration. They contrasted the INPGA's execution time and rule generation with those of the BNPGA and SDNPGA. The experimental findings support the superiority of the technique over BNPGA and SDNPGA.

3. **Data Mining:** For mining extremely accurate and comprehensible classification rules from large data sets, Dehuri and Mall [49] offer the enhanced niched Pareto genetic algorithm (INPGA). There is frequently a conflict between the regulations' accuracy and their comprehensibility. As a result, it is an optimization problem that is extremely challenging to address well. For this, they put forth the enhanced INPGA method. The simple genetic algorithm (SGA) and the fundamental NPGA have been contrasted with the rule generation produced by INPGA. The experimental outcome supports the fact that the rule generation clearly outperforms SGA and NPGA.
4. **Fuzzy Classifier:** The creation of fuzzy if-then rules and membership functions is one of the key aspects of fuzzy classifier design. Kannan and Thanapal [50] present an NPGA approach to obtain the optimal fuzzy if-then rule set and the membership function. The membership functions and rule set for the fuzzy system are simultaneously developed using NPGA and encoded into the chromosome. Through the creation of a fuzzy classifier on Iris data that is accessible in the UCI machine learning repository, the performance of the suggested technique is proven. According to the simulation research, NPGA creates a fuzzy classifier with the fewest possible rules and the highest level of classification accuracy when compared to other techniques.

5.3 Pareto Archived Evolution Strategy (PAES)

A historical archive that contains some of the previously discovered non-dominated solutions is combined with a (1 + 1) evolution strategy (i.e., one parent that produces one offspring). Each altered person is compared to this archive, which serves as a benchmark set. The method employs a novel strategy to maintain the diversity that entails a crowding procedure that recursively splits objective space. Based on the values of each solution's objectives, each solution is assigned a certain location in the grid. The number of solutions that are present in each grid location is shown on a map of this grid. Algorithm 7 provides a pseudocode of PAES.

Algorithm 7 PAES Algorithm

```

1. PAES ( $f_i(u)$ )
2. repeat
3.   Add to the archive  $A_0$  after initializing the single parent population  $P'$ .
4.   Perform mutation on  $P'$  to create offspring  $P''$  and fitness evaluation
5.   if  $P' > P''$  then
6.     reject  $P''$ 
7.   else if  $P'' > P'$  then
8.     replace  $P'$  with  $P''$ , and add  $P'$  to  $A_0$ 
9.   else if  $\exists P''' \in A_0 (P''' > P')$  then
10.    reject  $P''$ 
11.   else
12.     To identify which is the fresh current answer and if we need to add  $P''$  to
         $A_0$ , use test  $(P', P'', A_0)$ .
13.   End
14. Till the stopping conditions are satisfied
15. End

```

5.3.1 Applications of PAES in Machine Learning

1. **Concurrently Learning Rule and Data Bases of Linguistic Fuzzy-Rule-Based Systems:** Alcal et al. [51] suggest using a multiobjective evolutionary method to create a collection of linguistic fuzzy-rule-based systems with various accuracy and interpretability trade-offs. Approximation error and rule base (RB) complexity are used to quantify accuracy and interpretability, respectively. The suggested method is based on simultaneously learning RBs and membership function parameters of the related linguistic labels. The method was put to the test on nine real-life data sets of varying sizes and number of variables. The results validate the usefulness of the technique, particularly for (potentially high-dimensional) datasets with high complexity metric values for the related language labels.
2. **Selective Ensemble Learning Method for Belief-Rule-Base Classification System:** To achieve better outcomes than a single belief-rule-based system, traditional BRB ensemble learning approaches incorporate all of the trained sub-BRB systems. The shrinking gap between subsystems causes a significant number of duplicate sub-BRB systems to be produced as the number of BRB systems engaged in ensemble learning rises. This significantly slows down prediction speed and raises storage needs for BRB systems. To solve these problems, Liu et al. [52] propose BRBCS-PAES which is a multi-objective PAES based selective ensemble learning method for BRB Classification Systems (BRBCS).
3. **Generating a Set of Mamdani Fuzzy Systems::** In recent years, the numerous successful applications of fuzzy rule-based systems (FRBSs) in diverse domains

have generated significant interest in techniques for deriving FRBSs from data. However, many approaches proposed in the literature primarily emphasize performance optimization while overlooking the comprehensibility of FRBSs. Despite the intrinsic nature of fuzzy logic, the challenge of striking the right balance between efficiency and comprehensibility has only recently garnered increased attention, spurring the development of approaches that account for both aspects. In their work, Cococcioni et al. [53] introduce a Pareto-based multi-objective evolutionary method, based on a variation of the well-known (2 + 2) PAES, for generating a set of Mamdani fuzzy systems from numerical data.

4. Finding Good Parameters in an ANN and Training a Classifier on a Large Dataset:

Shenfield and Rostami [54] propose to address the challenges of identifying optimal structural and parametric configurations in an Artificial Neural Network (ANN) and training a classifier with a highly imbalanced dataset using a novel multi-objective optimization technique. They employ the advanced CMA-PAES-HAGA approach [55] for simultaneous optimization of the structure, weights, and biases of a population of ANNs. This optimization focuses not only on the overall classification accuracy but also on the accuracy of classifying individual target classes. The effectiveness of this approach is demonstrated using a real-world medical diagnostic problem, specifically the categorization of fetal cardiotocograms. In this problem, more than 75% of the data falls into a single majority class, with the remaining data distributed across two minority classes. Following optimization, the ANN outperforms a standard feed-forward ANN in terms of recognizing minority classes, although it exhibits somewhat lower overall classification accuracy.

6 Conclusion

In recent years, metaheuristics have evolved into indispensable instruments for addressing complex optimization challenges encountered across various domains, including industrial, engineering, biomedicine, image processing, and theoretical fields. These metaheuristics encompass a diverse array of techniques, and ongoing developments continue to introduce new ones. To cope with various global and engineering optimization challenges, as well as real-world applications, metaheuristics have been integrated with multiple machine learning approaches. Through this chapter, we looked into many possibilities for incorporating meta-heuristics into ML. A brief examination of applications of the meta-heuristic algorithms employed in machine learning is provided. This study shows that there is an excellent scope of meta-heuristics in machine learning problems, especially with very large data sets.

References

1. Miettinen K. Nonlinear multiobjective optimization. New York: Springer Science and Business Media; 1999.
2. Jain I, Jain VK, Jain R. Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. *Appl Soft Comput.* 2018;62:203–15.
3. Pashaei E, Pashaei E, Aydin N. Gene selection using hybrid binary black hole algorithm and modified binary particle swarm optimization. *Genomics.* 2019;111(4):669–86.
4. Tang BQ, Han J, Guo GF, Chen Y, Zhang S. Building material prices forecasting based on least square support vector machine and improved particle swarm optimization. *Archit Eng Design Manage.* 2019;15(3):196–212.
5. Raj S, Ray KC. ECG signal analysis using DCT-based DOST and PSO optimized SVM. *IEEE Trans Instrum Meas.* 2017;66(3):470–8.
6. Wang J, Lei D, Cai J. An adaptive artificial bee colony with reinforcement learning for distributed three stage assembly scheduling with maintenance. *Appl Soft Comput.* 2022;117:108371.
7. Khourdifi Y, Bahaj M. Heart disease prediction and classification using machine learning algorithms optimized by particle swarm optimization and ant colony optimization. *Int J Intell Eng Syst.* 2019;12(1):242–52.
8. Banharnsakun A. Towards improving the convolutional neural networks for deep learning using the distributed artificial bee colony method. *Int J Mach Learn Cybern.* 2019;10(6):1301–11.
9. Kushwah GS, Ranga V. Detecting DDoS attacks in cloud computing using extreme learning machine and adaptive differential evolution. *Wirel Pers Commun.* 2022;124(3):2613–36.
10. Zangooei MH, Habibi J, Alizadehsani R. Disease diagnosis with a hybrid method SVR using NSGA-II. *Neurocomputing.* 2014;136:14–29.
11. Chauhan D, Yadav A, Neri F. A multi-agent optimization algorithm and its application to training multilayer perceptron models. *Evol Syst.* 2023; <https://doi.org/10.1007/s12530-023-09518-9>.
12. Chauhan D, Yadav A. A competitive and collaborative-based multilevel hierarchical artificial electric field algorithm for global optimization. *Inf Sci.* 2023;648:119535.
13. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 4). IEEE; 27 Nov 1995. p. 1942–48.
14. Karbassi Yazdi A, Kaviani MA, Emrouznejad A, Sahebi H. A binary particle swarm optimization algorithm for ship routing and scheduling of liquefied natural gas transportation. *Transport Lett.* 2020;12(4):223–32.
15. Xu X, Rong H, Trovati M, Liptrott M, Bessis N. CS-PSO: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems. *Soft Comput.* 2018;22:783–95.
16. Zhang X, Zheng X, Cheng R, Qiu J, Jin Y. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Inf Sci.* 2018;427:63–76.
17. Wang H, Peng MJ, Hines JW, Zheng GY, Liu YK, Upadhyaya BR. A hybrid fault diagnosis methodology with support vector machine and improved particle swarm optimization for nuclear power plants. *ISA Trans.* 2019;95:358–71.
18. Ma K, Hu S, Yang J, Xu X, Guan X. Appliances scheduling via cooperative multi-swarm PSO under day-ahead prices and photovoltaic generation. *Appl Soft Comput.* 2018;62:504–13.
19. Corne D, Dorigo M, Glover F, Dasgupta D, Moscato P, Poli R, Price KV, editors. New ideas in optimization. London: McGraw-Hill; 1999.
20. Dorigo M. The ant system optimization by a colony of cooperating agents. *IEEE Trans Syst Man Cybernet Part B.* 1996;26(1):1–3.
21. Dorigo M, Stutzle T. Ant colony optimization. Cambridge, MA: MIT Press.
22. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. *IEEE Comput Intell Mag.* 2006;1(4):28–39.

23. Coello CA. Evolutionary algorithms for solving multi-objective problems. New York: Springer; 2007.
24. Sara SBVJ, Kalaiselvi K. Ant colony optimization (ACO) based feature selection and extreme learning machine (ELM) for chronic kidney disease detection. *Int J Adv Stud Sci Res.* 2019;4(1):1–8.
25. Paniri M, Dowlatshahi MB, Nezamabadi-pour H. Ant-TD: ant colony optimization plus temporal difference reinforcement learning for multi-label feature selection. *Swarm Evol Comput.* 2021;64:100892.
26. Ghosh M, Guha R, Sarkar R, Abraham A. A wrapper-filter feature selection technique based on ant colony optimization. *Neural Comput & Applic.* 2020;32:7839–57.
27. Mythili K, Rangaraj R. Crop recommendation for better crop yield for precision agriculture using ant colony optimization with deep learning method. *Ann Romanian Soc Cell Biol.* 2021;13:4783–94.
28. Karaboga D. Artificial bee colony algorithm. *Scholarpedia.* 2010;5(3):6915.
29. Ozbay E. An active deep learning method for diabetic retinopathy detection in segmented fundus images using artificial bee colony algorithm. *Artif Intell Rev.* 2022;56:1–28.
30. Pu Q, Xu C, Wang H, Zhao L. A novel artificial bee colony clustering algorithm with comprehensive improvement. *Vis Comput.* 2022;38(4):1395–410.
31. Uzer MS, Yilmaz N, Inan O. Feature selection method based on artificial bee colony algorithm and support vector machines for medical datasets classification. *Sci World J.* 2013;28:2013.
32. Yavuz G, Aydin D. Angle modulated artificial bee colony algorithms for feature selection. *Appl Comput Intell Soft Comput.* 2016;2016:7.
33. Storn R, Price K. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optim.* 1997;11(4):341.
34. Demertzis K, Iliadis L. Adaptive elitist differential evolution extreme learning machines on big data: intelligent recognition of invasive species. In: Advances in Big Data: Proceedings of the 2nd INNS Conference on Big Data, October 23–25, 2016, Thessaloniki, Greece 2. Springer International Publishing; 2017. p. 333–45.
35. Rajesh C, Kumar S. An evolutionary block based network for medical image denoising using differential evolution. *Appl Soft Comput.* 2022;121:108776.
36. Juwono FH, Wong WK, Pek HT, Sivakumar S, Acula DD. Ovarian cancer detection using optimized machine learning models with adaptive differential evolution. *Biomed Signal Proc Control.* 2022;77:103785.
37. Schmidt M, Safarani S, Gastinger J, Jacobs T, Nicolas S, Schulke A. On the performance of differential evolution for hyperparameter tuning. In: 2019 International Joint Conference on neural networks (IJCNN). IEEE; 14 July 2019. p. 1–8.
38. Griffel D. Multi-objective optimization using evolutionary algorithms, by Kalyanmoy Deb, Pp. 487.£ 60. 2001. ISBN 0 471 87339 X (Wiley). *Math Gaz.* 2003;87(509):409–10.
39. Paszkowicz W. Increasing importance of genetic algorithms in science and technology: linear trends over the period from year 1989 to 2022. *Mater Manuf Process.* 2023;6:1–20.
40. Deb K, Pratap A, Agarwal S, Meyarivan TA. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput.* 2002;6(2):182–97.
41. Srinivas N, Deb K. Multibjective optimization using nondominated sorting in genetic algorithms. *Evol Comput.* 1994;2(3):221–48.
42. Khurana D, Yadav A, Sadollah A. A non-dominated sorting based multi-objective neural network algorithm. *MethodsX.* 2023;10:102152.
43. Zhou Y, Zheng S. Machine learning-based multi-objective optimization of an aerogel glazing system using NSGA-II—study of modelling and application in the subtropical climate Hong Kong. *J Clean Prod.* 2020;253:119964.
44. Yazdinejad A, Dehghantanha A, Parizi RM, Epiphanou G. An optimized fuzzy deep learning model for data classification based on NSGA-II. *Neurocomputing.* 2023;522:116–28.

45. Tong J, Li Y, Liu J, Cheng R, Guan J, Wang S, Liu S, Hu S, Guo T. Experiment analysis and computational optimization of the Atkinson cycle gasoline engine through NSGA II algorithm using machine learning. *Energy Convers Manag.* 2021;238:113871.
46. Soui M, Mansouri N, Alhamad R, Kessentini M, Ghedira K. NSGA-II as feature selection technique and AdaBoost classifier for COVID-19 prediction using patient's symptoms. *Nonlinear Dyn.* 2021;106(2):1453–75.
47. Baraldi P, Pedroni N, Zio E. Application of a niched Pareto genetic algorithm for selecting features for nuclear transients classification. *Int J Intell Syst.* 2009;24(2):118–51.
48. Lu J, Yang F, Li M, Wang L. Multi-objective rule discovery using the improved niched Pareto genetic algorithm. In: 2011 Third International Conference on Measuring Technology and Mechatronics Automation (Vol. 2). IEEE; 6 Jan 2011. p. 657–61.
49. Dehuri S, Mall R. Predictive and comprehensible rule discovery using a multi-objective genetic algorithm. *Knowl-Based Syst.* 2006;19(6):413–21.
50. Kannan AK, Thanapal P. A hybrid evolutionary approach for optimal fuzzy classifier design. In: 2010 International Conference on Communication Control and Computing Technologies. IEEE; 7 Oct 2010. p. 835–40.
51. Alcalá R, Ducange P, Herrera F, Lazzerini B, Marcelloni F. A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems. *IEEE Trans Fuzzy Syst.* 2009;17(5):1106–22.
52. Liu W, Wu W, Wang Y, Fu Y, Lin Y. Selective ensemble learning method for belief-rule-base classification system based on PAES. *Big Data Mining Anal.* 2019;2(4):306–18.
53. Cococcioni M, Ducange P, Lazzerini B, Marcelloni F. A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. *Soft Comput.* 2007;11:1013–31.
54. Shenfield A, Rostami S. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In: 2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). IEEE. 23 Aug 2017. p. 1–8.
55. Rostami S, Neri F. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integr Comput Aided Eng.* 2016;23(4):313–29.

Machine Learning Assisted Metaheuristic Based Optimization of Mixed Suspension Mixed Product Removal Process



Ravi Kiran Inapakurthi, Sakshi S. Naik, and Kishalay Mitra

Abstract Mixed-Suspension Mixed-Product Removal (MSMPR) process plays key role in pharmaceutical industry as it is vital for separation and purification stages. A first principles-based detailed model is required to capture physical phenomena occurring in the process. However, such models are time consuming, and sometimes numerically unstable preventing their online implementation. Concurrent optimization can be achieved using data-based modelling. Finite high-fidelity data obtained from the MSMPR model was used to develop surrogates using Support Vector Regression (SVR). The model development is achieved in an optimization framework which results in multiple Pareto optimal SVR models. Generic model development approach is facilitated by assigning each input a different kernel parameter and exploring multiple kernel functions. To ensure model development with limited data points, sample size estimation algorithm is proposed specifically curated for the SVR technique. An evolutionary algorithm is used to find the optimal SVR models, which are fast and accurate, thereby enabling faster optimization of such processes. The study indicates many-fold decrease in optimization time of the MSMPR process thereby facilitating realization of real-time optimization. Comparison with baseline techniques like ridge regression and multi-linear regression techniques establishes the competitiveness of the proposed algorithm.

R. K. Inapakurthi · S. S. Naik

Global Optimization and Knowledge Unearthing Laboratory, Department of Chemical Engineering, Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Telangana, India
e-mail: snaik@andrew.cmu.edu

K. Mitra (✉)

Global Optimization and Knowledge Unearthing Laboratory, Department of Chemical Engineering, Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Telangana, India

Department of Artificial Intelligence, Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Telangana, India
e-mail: kishalay@che.iith.ac.in

Keywords Machine learning · Support vector machine · Evolutionary algorithms · Process optimization · Multi-objective optimization · Pareto · Crystallization · Overfitting

1 Introduction

Crystallization plays a major role in the pharmaceutical industry. Properties such as size, purity and stability of the crystalline solids are essential to ensure the safety and reliability of the drug [1]. MSMPR crystallizers have started gaining popularity as chemical industries move towards continuous manufacturing while trying to maintain product quality similar to that of batch processes. Population balance models for MSMPR crystallization with an integrated wet-mill for crystal breakage have also been analyzed [2]. MSMPR cascades with phenomena such as growth, nucleation and agglomeration have been modeled and analyzed using quadrature method of moments to obtain numerical solutions. Combining method of characteristics and moments for validation, crystal size distribution (CSD) has been reconstructed to study processes where crystal growth is the primary phenomenon [3]. To address this problem of tracking the CSD in the presence of multiple phenomena like growth, nucleation, breakage and agglomeration, researchers have used High Resolution Finite Volume Methods (HR-FVM) to discretize the PBEs [2]. These methods discretize the PDEs to obtain algebraic equations or set of ODEs which can be solved using numerical solvers. Although these methods track the CSD accurately, they are expensive [4]. The optimization and control of the MSMPR crystallization process is further challenging due to the highly non-linear nature of the model equations.

Recent trends in the use of machine learning to model industrial crystallization processes have seen an upward trend. Reinforcement learning has been used to learn an optimal control policy for a crystallization model using convolutional neural networks to measure the CSD [5]. Machine learning models using recurrent neural networks and auto-encoders for predictive control of batch crystallization processes have also been shown to achieve desired properties of the product CSD [6]. Artificial neural networks were used for optimization of an industrial crystallization units [7]. In this chapter we describe the use of Support Vector Regression (SVR) for concurrent optimization of a cascaded MSMPR.

For this, surrogate models can be explored. Surrogate techniques were used in crystallization process [8] to build less complex and less time-consuming models. As an added advantage, numerical integration issues are naturally avoided with the use of surrogate models. Based on the statistical learning theory SVR are gaining momentum in recent times. However, despite the widely acknowledged success of SVRs, the following drawbacks daunt their image:

1. The $O(N^3)$ time for training SVRs is criticized to be huge [9].
2. The use of fixed kernel for function approximation plagues the literature and other potentially superior kernel functions are rarely explored for modelling [10].

3. Another important factor which is widely prevalent in the literature is the use of a sole kernel parameter σ for the inputs [11].
4. The selection of the bounds on hyper-parameters is arbitrary [12] creating doubts about the quality of the developed models.
5. Model development is usually targeted using grid search approach of optimization [13].

To develop high accuracy models for the MSMPR process, an evolutionary algorithm-based optimization formulation is presented which optimizes the decision variables of SVR. The proposed algorithm helps alleviate the challenges thereby paving way for Surrogate Assisted Optimization (SAO). The following points are noted as the novelties of this study:

1. A sample size estimation algorithm is developed to limit the use of redundant points. This directly translates to less model development time.
2. To facilitate models with reasonable accuracy and maintain a balance between accuracy and the sample size, a multi-objective optimization algorithm is presented. This helps in obtaining multiple equally applicable and competitive SVR models for the MSMPR crystallization process.
3. Three new kernel functions are explored for modelling in addition to the standard Gaussian kernel function.
4. To have a provision through which each input is assigned different relative importance, each input is assigned a kernel function and is used in the optimization formulation.
5. The developed models are then used for SVR based SAO for the MSMPR process. To establish the competitiveness of the developed SVR models, comparison with baseline regression techniques like ridge regression and multiple linear regression is done.

This chapter is organized in the following way. In Sect. 2, description of MSMPR process and SVR are explained. In Sect. 3, the developed algorithm is detailed. Section 4 covers the results and analysis. This covers the SAO and the comparison with baseline regression techniques like ridge regression and multiple linear regression. This is followed by the conclusions in Sect. 5.

2 Materials and Methods

2.1 *MSMPR Crystallization Process*

We consider a three-stage cascaded MSMPR process for crystallization as shown in Fig. 1. A hot slurry of saturated solution with seed crystals is fed continuously into the first MSMPR crystallizer. In the first crystallizer, the temperature of the slurry is reduced using cooling jackets leading to the growth of crystals. The slurry from the i^{th} crystallizer is sent to the $(i + 1)^{th}$ crystallizer, where the crystals grow further in

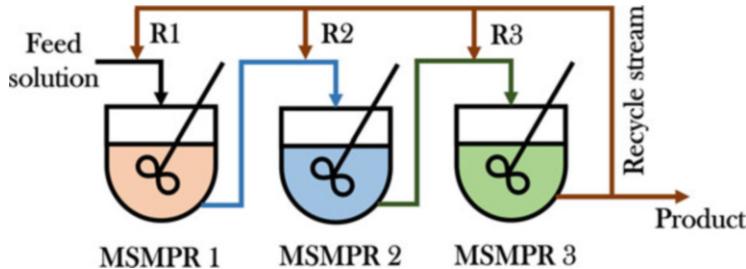


Fig. 1 Process flow diagram of cascaded MSMPR crystallizer used in this study

size, nucleate, and break simultaneously, which leads to birth of new crystals as well as death of crystals due to breakage. For compounds whose grate rates are very low, an increase in residence time is necessary to develop larger crystal size and thus a high yield [14]. Experimental studies have shown that adding a recycle stream to a continuous MSMPR crystallization process can increase the yield and API purity. To facilitate that, a recycle stream is added which recycles the crystals back to each crystallizer. The product stream is collected from the last crystallizer. Recycle stream is usually provided to ensure product quality. The system is modeled dynamically till it reaches a steady state.

2.1.1 Mathematical Model

The detailed process is modeled as Partial Integro-Differential Equations (PIDEs) consisting of size independent growth, secondary nucleation, and breakage as the dominating phenomena. The PBE and mass balance in the MSMPR are written as a partial differential equation as shown in Eqs. (1) and (2). For i^{th} MSMPR in the cascade, $n_i(L, t)$ is the mono-variate CSD function which provides the crystal quantity with size between L and $L + dL$ per unit suspension volume at time t . G represents the growth rate of the crystals which is independent of size, B is the secondary nucleation rate leading to birth of nucleons of size L_n . B_{bre} and D_{bre} are birth rate and death rates of a crystal of size L due to breakage. R is the recycle ratio and τ is the residence time in the MSMPR. c_{rec} is the recycle stream concentration, ρ_c is the crystal density and k_v is the volume shape factor.

$$\frac{\partial n_i(L, t)}{\partial t} + G_i \frac{\partial n_i(L, t)}{\partial L} = B_i \delta(L - L_n) + (B_{bre,i}(L) - D_{bre,i}(L)) + \frac{(1 - R_i)n_{i-1}(L, t) + R_i(n_{rec}(L, t)) - n_i(L, t)}{\tau_i} \quad (1)$$

$$\frac{\partial c_i}{\partial t} = -\rho_c k_v \left[B_i L_n^3 + 3G_i \int_0^{L_{max}} L^2 n_i(L, t) dL \right] + \frac{(1-R_i)c_{i-1} + (R_i)c_{rec} - c_i}{\tau_i} \quad (2)$$

The growth rate is given by Eq. (3) while the secondary nucleation rate is given by Eq. (4). The kinetic parameters k_g , g , k_b , b , k_d and d are taken from literature available for Ortho-amino benzoic acid (oABA) [2]

$$G_i = \begin{cases} k_g \left| 1 - \frac{c_i}{c_{i,sat}} \right|^g, & \text{if } c_i \geq c_{i,sat} \\ -k_d \left| 1 - \frac{c_i}{c_{i,sat}} \right|^d, & \text{if } c_i < c_{i,sat} \end{cases} \quad (3)$$

$$B_i = \begin{cases} k_b \left| 1 - \frac{c_i}{c_{i,sat}} \right|^b, & \text{if } c_i \geq c_{i,sat} \\ 0, & \text{if } c_i < c_{i,sat} \end{cases} \quad (4)$$

The temperature dependency of the solute solubility is taken to be of the following form.

$$c_{i,sat} = a_0 + a_1 T_i + a_2 T_i^2 \quad (5)$$

The birth of new crystals of size L can occur due to breakage of crystals of size $\lambda > L$. This phenomenon is represented by Eq. (6) where $b(L|\lambda)$ is the daughter crystals distribution function which gives the probability that a λ size crystal breaks to form a crystal of size L . $S_0(\lambda)$ is the breakage selection function given by a power law such that it considers that larger crystals have a higher probability of breaking. The exponent k in $S_0(\lambda)$ depends on the material and the system specifications.

$$B_{bre,i}(L) = \int_{\lambda}^{L_{max}} b(L|\lambda) b_0 S_0(\lambda) n_i(\lambda, t) d\lambda \quad (6)$$

where $b(L|\lambda) = \frac{2}{\lambda}$, and $S_0(\lambda) = \lambda^k$. The death of crystals of size L due to breakage is given by Eq. (7).

$$D_{bre,i}(L) = b_0 S_0(L) n_i(L, t) \quad (7)$$

In this study we use an exponentially decreasing cooling profile in the MSMPRs shown in Eq. (8).

$$T_i = T_{i-1} - \theta_i \left(1 - e^{-\frac{t}{100}} \right) \quad (8)$$

2.1.2 Optimization Setup

MSMPR typically has conflicting performance objectives like maximizing the average crystal size, minimizing the CSD, achieving a target aspect ratio, etc. [15]. Optimizing these objectives individually can lead to significantly different optimal operation conditions as there are trade-offs among them. Therefore, it is necessary to optimize them simultaneously in a multi-objective optimization setup.

In this chapter, maximizing the Mean Crystal Size (MCS) while minimizing the coefficient of variation of the distribution and minimizing the residence time in each MSMPR is targeted. These are conflicting objectives as decreasing the residence time will decrease the MCS. While a larger residence time will lead to a more spread out CSD. The recycle ratio also impacts the average size of the product crystals. Having a high recycle ratio increases the average crystal size but at the same time increases the spread of the CSD. Hence, it is necessary to carefully choose the decision variables for optimal operation. The degrees of freedom in this problem are the recycle ratio, the cooling temperature of each MSMPR crystallizer and the residence time. The optimization formulation is given by Eqs. (9), (10), (11), (12), (13), (14), (15), and (16).

$$\underset{R_i, \tau_i, \theta_i, T_0}{\text{Minimize}} -\bar{L} = -\frac{\sum_{j=1}^N L_j n_j}{\sum_{j=1}^N n_j} \quad (9)$$

$$\underset{R_i, \tau_i, \theta_i, T_0}{\text{Minimize}} \bar{\sigma} = \sqrt{\frac{\sum_{j=1}^N n_j (L_j - \bar{L})^2}{\sum_{j=1}^N n_j}} \quad (10)$$

$$\underset{R_i, \tau_i, \theta_i, T_0}{\text{Minimize}} \tau_{total} = \sum_{i=1}^M \tau_i \quad (11)$$

Subject to:

$$R_{min} \leq R_i \leq R_{max} \quad (12)$$

$$T_{min} \leq T_0 \leq T_{max} \quad (13)$$

$$\theta_{min} \leq \theta_i \leq \theta_{max} \quad (14)$$

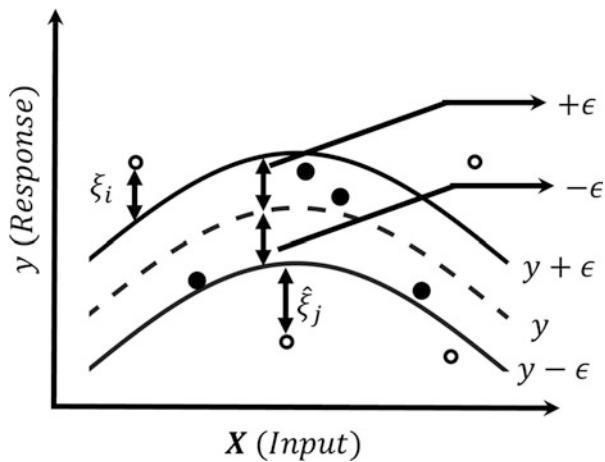
$$T_0 \geq T_1 \geq \dots \geq T_M \quad (15)$$

$$\tau_{min} \leq \tau_i \leq \tau_{max} \quad (16)$$

Table 1 Upper and lower limits on the optimization variables of MSMPR

Bounds	Description	Value
T_{\min}	Minimum initial temperature in MSMPR 1 ($^{\circ}\text{C}$)	31
T_{\max}	Maximum initial temperature in MSMPR 1 ($^{\circ}\text{C}$)	36
θ_{\min}	Minimum decrease in temperature while cooling ($^{\circ}\text{C}$)	1
θ_{\max}	Maximum decrease in temperature while cooling ($^{\circ}\text{C}$)	4
R_{\min}	Minimum allowed value of Recycle ratio	0
R_{\max}	Maximum allowed value of Recycle ratio	0.3
τ_{\min}	Minimum allowed value of Residence time (s)	200
τ_{\max}	Maximum allowed value of Residence time (s)	360

Fig. 2 The MSMPR process output is depicted using y and the SVR prediction of the MSMPR process is expected to fall within $y \pm \epsilon$. Predictions within this limit are shown in filled black circles while predictions beyond these limits are shown in unfilled circles



\bar{L} is the average crystal size of the product crystals. $\bar{\sigma}$ is the coefficient of variation of the CSD. τ_{total} is the Total Mean Residence Time (TMRT) of the slurry in the crystallization system. The bounds for the optimization variables are shown in Table 1.

2.2 Support Vector Regression

Assume that there are N input-output data pairs collected from the MSMPR crystallization process, which can be represented to be of the form $\{X_i, y_i\}_{i=1}^N$. The inputs of the MSMPR process are denoted as $X (= [x_1, \dots, x_n]) \in \mathbb{R}^n$ and the corresponding output is represented by $y \in \mathbb{R}$. Without loss of generality, the multi-input multi-output process can be modelled in a multi-input single-output process for each of the outputs. The underlying mechanism of SVR is shown in Fig. 2. Predictions for new inputs of MSMPR process can be obtained using the following equation [16]:

$$f(\mathbf{X}) = \sum_{i=1}^N \mathbf{w}^T \Phi(\mathbf{X}_i) + w_0 \quad (17)$$

where w_0 is the bias, $\mathbf{w} \in \mathbb{R}^n$ is the weight vector. The widely used primal formulation of SVR in the literature is presented below in Eqs. (18), (19), (20), and (21).

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \hat{\xi}_i) \quad (18)$$

Subject to,

$$y_i - f(\mathbf{X}_i) \leq \epsilon + \xi_i \quad (19)$$

$$f(\mathbf{X}_i) - y_i \leq \epsilon + \hat{\xi}_i \quad (20)$$

$$\xi_i, \hat{\xi}_i \geq 0; \forall i = 1, \dots, N \quad (21)$$

C converts the multiple requirements of model simplicity and extent of penalty into one expression. In the literature, the primal formulation is converted into its equivalent dual formulation using Lagrange multipliers. Lagrange multipliers $\{\alpha_i, \hat{\alpha}_i, \mu_i, \hat{\mu}_i\} \geq 0 \forall i = 1, \dots, N$ are introduced and multiplied for all constraints (Eqs. 22, 23, and 24) and the equivalent dual formulation is derived as shown below.

$$\begin{aligned} \max_{\alpha, \hat{\alpha}} & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) k(\mathbf{X}_i, \mathbf{X}_j) - \epsilon \sum_{i=1}^N (\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^N \\ & \times (\alpha_i - \hat{\alpha}_i)y_i \end{aligned} \quad (22)$$

subject to,

$$0 \leq \alpha_i, \hat{\alpha}_i \leq C \quad (23)$$

$$\sum_{i=1}^N (\alpha_i - \hat{\alpha}_i) = 0 \quad (24)$$

where, $k(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)$. $\Phi(\mathbf{X}_j)$ is the kernel function.

The nonlinear mapping relating the inputs and the outputs of the MSMPR process is captured using the kernel function $k(\mathbf{X}_i, \mathbf{X}_j)$. Multiple kernel functions are explored viz., Gaussian, quadric, multi-quadric, and inverse multi-quadric [17] and their functional form is presented below:

$$Gaussian : \exp \left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2} \right) \quad (25)$$

$$Multi \text{ quadric} : \sqrt{1 + \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}} \quad (26)$$

$$Inverse \text{ quadric} : \frac{1}{1 + \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}} \quad (27)$$

$$Inverse \text{ multi } quadric : \frac{1}{\sqrt{1 + \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}}} \quad (28)$$

3 Proposed Algorithm

The multiple conflicting requirements of accurate modeling and availability and use of less sample size during modelling exercise are exploited to frame a multi-objective optimization formulation. These objectives are optimized using an evolutionary algorithm to get multiple equally competitive SVR models. The formulation is presented first. This is followed by the two pseudo codes, one each for the sample size estimation and optimal hyper-parameter determination.

3.1 Optimization of SVR Hyper-parameters (MOOP II)

$$\min_{C, \epsilon, Kernel \text{ type}, \sigma_{1:n}} RMSE = \sqrt{\frac{\sum_{i=1}^{N_{val}} (y_i - f_i)^2}{N_{val}}} \quad (29)$$

$$\min_{C, \epsilon, Kernel \text{ type}, \sigma_{1:n}} N_{samp} \quad (30)$$

subject to,

$$Kernel \text{ type} = \begin{cases} 1, Gaussian \\ 2, Multi \text{ quadric} \\ 3, Inverse \text{ quadric} \\ 4, Inverse \text{ Multi } quadric \end{cases} \quad (31)$$

Table 2 Algorithmic flow for the developed algorithmAlgorithm-1: Proposed algorithm

Initialize: # binary decision and real variables as 1 and 2 + n , respectively. Set NSGA-II parameters. Set $gen = 0$, $pop = N_{pop}$.

Normalization: Normalize the training and validation data between 0 to 1.

$\forall i = 1$ to gen_{max}

if ($i = 1$)

1. Initialize N_{pop} parent population with random sampling. Each population is a combination of the hyper-parameters.
2. For all populations, do:
 - if j^{th} population is unexplored
 - a. Run sample size algorithm with j^{th} population.
 - b. Add j^{th} population to database.
 - c. Return objectives of MOOP II.

end

else:

1. Perform the operations of evolutionary algorithm.
2. Select the best N_{pop} candidate solutions and as parent population for next generation.
3. For all populations, do:
 - if j^{th} population is unexplored
 - a. Run sample size algorithm with j^{th} population.
 - b. Add j^{th} population to database.
 - c. Return objectives of MOOP II.

end loop.

end

end

$$0 < \sigma_{1:n} \leq \frac{|\max(x_{1:n}) - \min(x_{1:n})|}{2} \quad (32)$$

$$0 \leq \epsilon \leq 3 * noise * \sqrt{\frac{\ln N}{N}} \quad (33)$$

$$0 < C \leq \max(|\bar{y} + 3\sigma_y|, |\bar{y} - 3\sigma_y|) \quad (34)$$

where N_{val} is the cardinality of the validation set, \bar{y} is the average value of the actual MSMPR response, σ_y is the standard deviation of the actual MSMPR response and $\sigma_1 : n$ are the parameters of the kernel function assigned for each of the input. Eqs. (32), (33), and (34), which provide the limits on the hyper-parameters of SVR are obtained from [18]. To determine the optimal values of the hyper-parameters of SVR, binary-coded NSGA-II is used. The approach followed to find the optimal hyper-parameters is presented in Table 2. The sample size estimation algorithm is presented in Table 3.

Table 3 Algorithmic flow for the sample size estimation algorithm

Algorithm-2: Sample size estimation	
Input:	SVR model which is $[C, \epsilon, \sigma_1, \dots \sigma_n, \text{kernel type}]$
Initialize:	Set T as normalized training data, V as normalized validation data, $i = 0$, $slope ratio = 100$.
1.	Select N_{samp}^i number of training data and call it $T_{\text{subset}} = \{X_i, y_i\}_{i=1}^{N_{\text{samp}}^i}$. Assign $T_{\text{samp remaining}} = T \setminus T_{\text{subset}}$.
2.	Train SVR using the provided combination of hyper-parameter and T_{subset} .
3.	Call SVs from the training exercise as SV_{set} . Assign RMSE of SVR model on V as $RMSE^i$. do while ($slope ratio > \text{termination value}$) a) $i = i + 1$. b) Assign the k-NN of the SV points with the points in $T_{\text{samp remaining}}$ as kNN_{SV} . c) Select random points from $T_{\text{samp remaining}} \setminus kNN_{\text{SV}}$. Combine with SV_{set} , and kNN_{SV} to get T_{subset} . Update $T_{\text{samp remaining}}$ and N_{samp}^i . d) Train SVR on T_{subset} and get the SV_{set} . e) Assign RMSE of SVR model on V as $RMSE^i$. f) $slope ratio = \frac{\text{abs}(RMSE^i - RMSE^{i-1})}{\text{abs}(N_{\text{samp}}^i - N_{\text{samp}}^{i-1})}$. end
4.	Output: Return objectives of MOOP II.

4 Results and Discussions

First, the results of MOOP II are presented in Sect. 4.1. In Sect. 4.2, the optimization results are demonstrated with one surrogate model. This is compared with that of the MSMPR model. The quality of the PFs is evaluated using various metrics. The proposed algorithm is compared with the baseline regression technique and fivefold cross validation technique in Sects. 4.3 and 4.4, respectively.

4.1 Hyper-parameter Optimization

For the training of SVR, 3000 points were generated using SOBOL. To validate the trained SVR model of the MSMPR process, 300 data points were generated using LHS sampling plan.

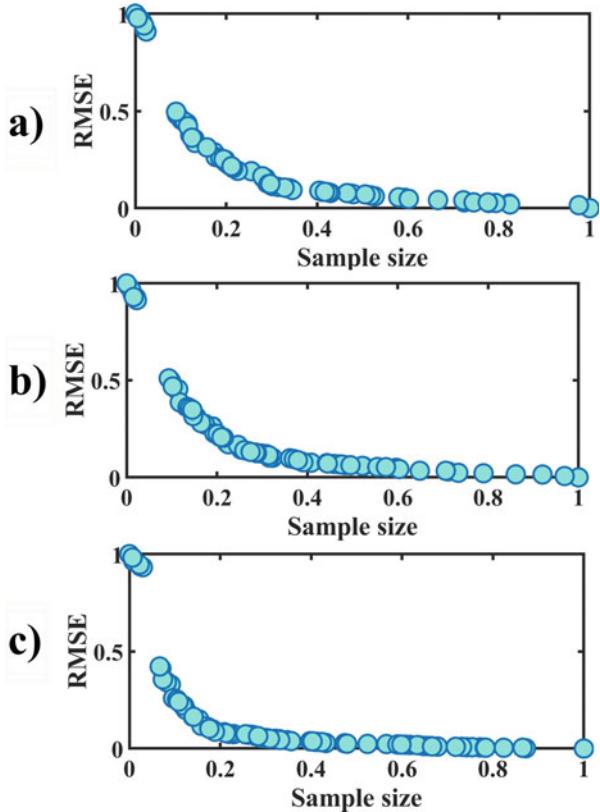
The number of generations for NSGA-II was set to 100 with an equal number of populations in each generation. Other parameter values are as follows: number of kNN for selecting within the vicinity of a SV is 5, termination value for a particular hyper-parameter combination is 0.00001, fraction in sample size algorithm 0.01, and initial sample size 30. The optimization exercise resulted in 77, 71 and 81 models each for MCS, spread of CSD and TMRT, respectively. A few of these models are shown in Table 4. From the table, inverse multi-quadric was selected as the optimal choice for mapping the MSMPR process. We would like to draw the attention of the readers towards literature where Gaussian kernel was the unambiguous choice as the kernel function. From the list, a model based on Hannan-Quinn Criterion (HQC) was

Table 4 Pareto SVR models for all the outputs of MSMPR. Model with highest HQC for each output is in bold

<i>MCS</i>	C	ϵ	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}	Kernel type	RMSSE	N_{samp}	Time (s)	HQC	R^2
0.861	0.001	0.405	0.480	0.477	0.480	0.464	0.424	0.403	0.310	0.372	0.423	4	0.0143	3000	37.05	-24.095	0.993	
1.130	0.077	0.379	0.440	0.488	0.495	0.493	0.477	0.473	0.457	0.372	0.477	4	0.042	746	0.234	-1956.986	0.954	
0.861	0.033	0.408	0.480	0.477	0.486	0.487	0.425	0.404	0.311	0.373	0.475	4	0.024	1581	5.102	-4849.666	0.982	
1.150	0.148	0.330	0.482	0.488	0.489	0.493	0.488	0.471	0.456	0.487	0.487	4	0.081	386	0.047	-816.674	0.908	
0.843	0.112	0.414	0.043	0.487	0.491	0.493	0.477	0.465	0.024	0.499	0.393	4	0.146	143	0.031	-178.668	0.292	
1.085	0.051	0.403	0.482	0.488	0.489	0.492	0.488	0.471	0.363	0.490	0.487	4	0.031	981	1.063	-2791.791	0.974	
0.883	0.027	0.410	0.480	0.462	0.486	0.498	0.491	0.421	0.310	0.448	0.440	4	0.021	1856	7.555	-5911.232	0.986	
1.103	0.037	0.410	0.481	0.478	0.489	0.486	0.495	0.471	0.364	0.490	0.480	4	0.026	1326	3.023	-3968.009	0.980	
1.130	0.077	0.379	0.440	0.488	0.495	0.493	0.477	0.473	0.457	0.372	0.497	4	0.043	736	0.234	-1912.185	0.951	
0.847	0.147	0.414	0.048	0.477	0.486	0.453	0.488	0.472	0.024	0.491	0.358	4	0.149	128	0.016	-70.32	0.264	
0.850	0.086	0.222	0.500	0.485	0.494	0.467	0.474	0.465	0.452	0.370	0.482	4	0.048	686	0.203	-1521.43	0.941	
<i>Spread of CSD</i>																		
1.10	0.006	0.354	0.485	0.486	0.385	0.496	0.498	0.416	0.282	0.446	0.477	4	0.0129	3000	37.658	-24.831	0.994	
1.101	0.061	0.385	0.485	0.462	0.470	0.495	0.469	0.441	0.317	0.432	0.477	4	0.0356	766	0.234	-4881	0.964	
1.102	0.099	0.385	0.474	0.491	0.470	0.500	0.499	0.443	0.330	0.422	0.493	4	0.053	551	0.094	-1352.25	0.949	
1.101	0.032	0.378	0.486	0.473	0.470	0.496	0.480	0.441	0.290	0.434	0.458	4	0.022	1406	3.648	-4384.41	0.983	
1.135	0.032	0.385	0.485	0.474	0.480	0.500	0.478	0.437	0.289	0.433	0.494	4	0.021	1551	4.945	-4941.54	0.986	
1.127	0.115	0.384	0.485	0.486	0.480	0.500	0.479	0.437	0.343	0.449	0.478	4	0.062	466	0.078	-1087.81	0.931	
1.127	0.115	0.386	0.485	0.487	0.480	0.500	0.479	0.437	0.343	0.449	0.478	4	0.059	516	0.094	-1227.48	0.937	
1.098	0.107	0.385	0.473	0.486	0.464	0.499	0.471	0.461	0.320	0.426	0.447	4	0.058	531	0.094	-1267.11	0.945	
1.101	0.038	0.385	0.486	0.486	0.472	0.498	0.479	0.410	0.255	0.497	0.493	4	0.024	1216	2.555	-3701.03	0.980	

0.752	0.010	0.331	0.039	0.363	0.262	0.471	0.463	0.461	0.022	0.364	0.397	4	0.130	193	0.047	-212.432	0.380
1.137	0.110	0.385	0.487	0.485	0.481	0.500	0.479	0.439	0.343	0.450	0.478	4	0.059	506	0.094	-1199.96	0.938
1.113	0.053	0.385	0.468	0.457	0.387	0.497	0.497	0.409	0.319	0.361	0.478	4	0.029	956	1.264	-2801.78	0.974
<i>TMRT</i>																	
1.001	0.003	0.458	0.442	0.472	0.478	0.331	0.291	0.327	0.492	0.489	0.461	4	0.0090	3000	26.238	-27.229	0.997
0.464	0.479	0.344	0.405	0.325	0.464	0.492	0.458	4	0.0150	1261	0.464	0.479	0.344	0.405	2.793	-10,161	0.993
1.018	0.029	0.468	0.441	0.473	0.481	0.377	0.382	0.330	0.492	0.489	0.473	4	0.017	1006	1.434	-3410.83	0.991
1.002	0.009	0.468	0.443	0.473	0.478	0.332	0.291	0.329	0.493	0.496	0.464	4	0.010	2376	13.492	-9179.18	0.997
0.051	0.114	0.477	0.467	0.458	0.110	0.347	0.085	0.431	0.016	0.486	0.453	4	0.165	133	0.016	-161.82	0.252
0.264	0.071	0.462	0.479	0.462	0.482	0.332	0.362	0.424	0.493	0.491	0.462	4	0.047	426	0.063	-1094.83	0.958
0.289	0.036	0.478	0.443	0.473	0.485	0.379	0.383	0.330	0.492	0.486	0.491	4	0.025	621	0.121	-1908.58	0.985
0.970	0.012	0.455	0.465	0.464	0.485	0.302	0.319	0.265	0.464	0.492	0.455	4	0.011	2036	9.891	-7688.31	0.996
0.931	0.030	0.479	0.468	0.472	0.478	0.331	0.359	0.333	0.465	0.489	0.490	4	0.019	931	0.793	-3068.67	0.991
0.969	0.019	0.455	0.455	0.464	0.479	0.303	0.310	0.264	0.464	0.492	0.457	4	0.013	1496	4.336	-5416.43	0.995
0.289	0.036	0.458	0.443	0.473	0.479	0.379	0.383	0.330	0.466	0.479	0.491	4	0.023	656	0.141	-2069.761	0.985
0.970	0.015	0.458	0.441	0.473	0.479	0.332	0.315	0.267	0.464	0.489	0.462	4	0.012	1841	7.223	-6759.808	0.996

Fig. 3 Normalized Pareto fronts for (a) MCS, (b) spread of CSD and (c) TMRT



selected for SVR based optimization of the MSMPR process. PF of MOOP II simulations is shown in Fig. 3. In Fig. 4a–c, parity plots for all the outputs of MSMPR process for the selected model are presented. From the figure, the accuracy of the SVR predictions can be judged.

4.2 SVR Based Optimization of MSMPR

The models tested in the previous section can be explored for performing Surrogate Assisted Optimization (SAO) of the MSMPR process. Since multi-input single-output type of modelling was performed in the previous section, a model mapping the input-output relation for all the outputs would be considered here for SAO. These models would be used instead of the mathematical model in MOOP I. For optimization, two algorithms—IBEA [19] and NSGA-II [20] are selected as they are widely used in the literature. For the SAO, the number of generations for running IBEA and NSGA-II was set to 30 and 200 populations were used within each

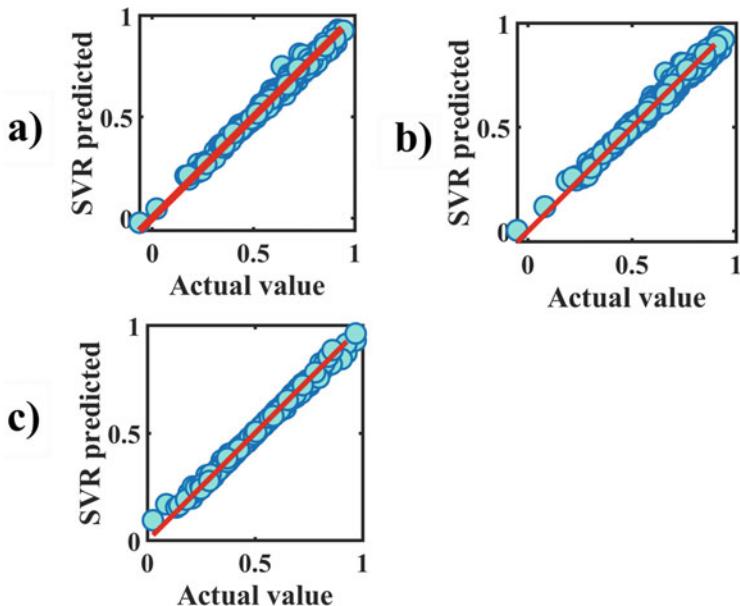


Fig. 4 Normalized Parity plots for the best HQC model (a) MCS, (b) spread of CSD and (c) TMRT

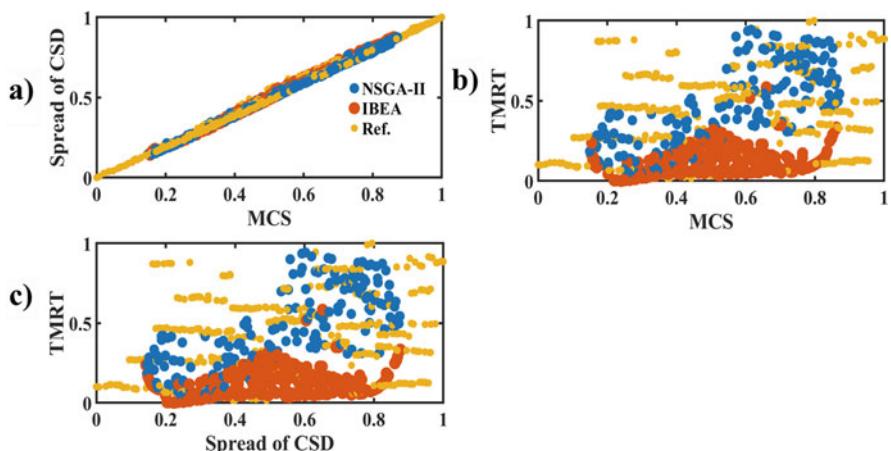


Fig. 5 For the best HQC model, (a) MCS and spread of CSD, (b) MCS and TMRT and (c) spread of CSD and TMRT

generation. Owing to the stochastic nature of the EAs, the optimization exercise is run multiple times with both IBEA and NSGA-II and the best values are selected for reporting. It is to be noted that this is a standard practice followed in the research community. The PFs generated by the SAO step is compared with the reference PF

Table 5 Evaluation of front generated by the best model. For this, SVR predictions and corrected with MSMPR model are used. The best value is highlighted

Metric\Algorithm	Preferred value	Minimum HQC model			
		SVR (approximated)		Corrected	
		IBEA	NSGA-II	IBEA	NSGA-II
IGD	Minimum	0.0404	0.0389	0.0397	0.0431
HV	Maximum	0.7396	0.7279	0.8625	0.8281
Spacing	Minimum	0.0201	0.0216	0.0253	0.0243
GD	Minimum	0.0032	0.0032	0.0032	0.0036
DeltaP	Minimum	0.1639	0.0878	0.1604	0.0828
DM	Maximum	0.6592	0.7130	0.6021	0.6722
PD	Maximum	95,524	158,558	88,268	154,445

(generated using mathematical model) in Fig. 5 in which various 2-dimensional (2D) projections are presented for better visualization of the results. It can be said that the SAO exercise has resulted in the following benefits:

1. Generation of additional Pareto optimal points in the region in which there were no points provided by the mathematical model and
2. In the places where Pareto optimal points were found out by the mathematical model, the SAO could get close to the points.

Based on Fig. 5, it can also be said that the convergence of IBEA to the actual PF is superior to that of the convergence of NSGA-II. However, the diversity of the solutions of NSGA-II is superior to the diversity of the solutions of IBEA. To further ascertain this claim, seven performance evaluation metrics measuring the convergence and diversity aspects of the surrogate PF with the actual PF are used. The metrics are Inverted Generational Distance (IGD) [21], Generational Distance (GD), DeltaP, Pure Diversity (PD), Metric for Diversity (DM), Spacing and Hyper Volume (HV). The comparison is presented in Table 5. The desired values (either minimum or maximum) of the respective performance evaluation metrics are given in the second column. Clearly, in terms of most of the metrics considered, NSGA-II outperforms IBEA. These metrics measure the convergence and diversity aspect of the SAO PF with the actual PF.

The approximation error induced using surrogate model in the optimization of the MSMPR crystallization process can be corrected by passing the decision variables of the SAO exercise through the mathematical model of the MSMPR process. This helps in gaining better understanding as the induced error is eliminated altogether. The improved and the SVR predicted outputs are presented in a parity plot form in Fig. 6. Across all outputs and with both the algorithms, the improved values are similar to the SVR predicted values. This indicates that the approximation error induced using SVR models directly for performing optimization of the MSMPR process is very less. The performance evaluation metrics used in the previous case can be used for the improved case as well as shown in Table 5. Again, NSGA-II outperforms IBEA in most of the performance evaluation metrics compared. The

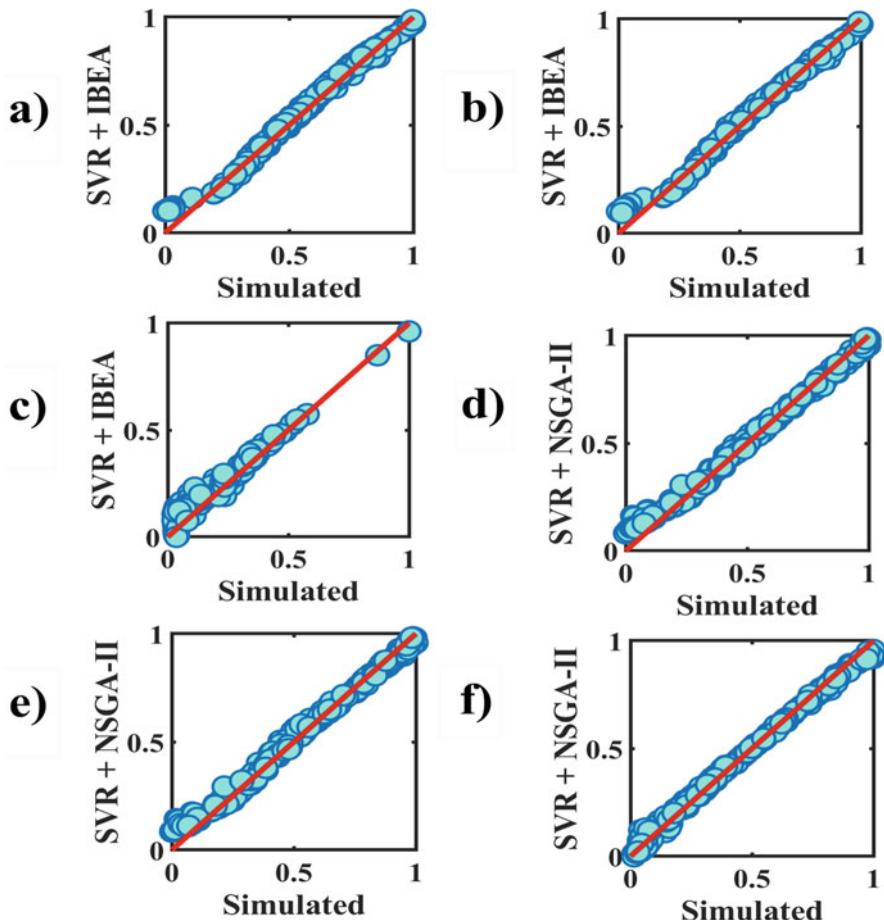


Fig. 6 Parity plot of IBEA and simulated for (a) MCS, (b) spread of CSD, (c) TMRT; NSGA-II and simulated for (d) MCS, (e) spread of CSD, and (f) TMRT

time for SAO is approximately 40 min while the time for optimization with the mathematical model is approximately 60 days. This results in many folds improvement in time required for optimization of the MSMPR process. With this significant reduction in time for optimization, real-time optimization might even become a reality.

Table 6 Comparison of the proposed algorithm with Ridge regression and multiple linear regression. The R^2 and RMSE values are indicated in %. Best values are highlighted

Method	Mean crystal size		Spread of CSD		Total mean residence time		Average of all outputs	
	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
Ridge regression	95.48	48.09	93.92	17.94	99.99	0.286	96.46	22.105
Multiple linear regression	95.48	48.09	93.92	17.94	99.99	0.286	96.46	22.105
Proposed algorithm	99.32	1.43	99.39	1.29	99.76	0.90	99.49	1.206

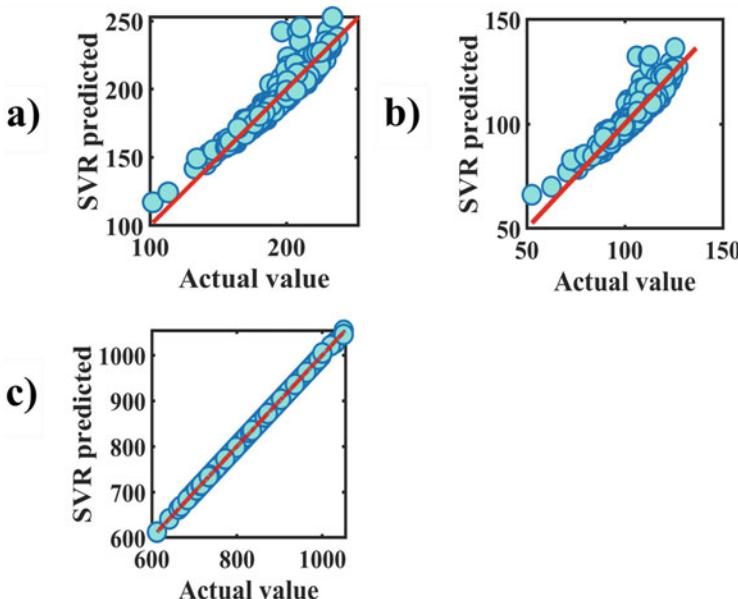


Fig. 7 Parity plot for Ridge regression for (a) MCS, (b) spread of CSD and (c) TMRT

4.3 Comparison of Proposed Algorithm with Baseline Regression Techniques

To compare the results with baseline regression techniques, ridge regression and multiple linear regression technique are used. The R^2 and RMSE of these models on the validation data is reported in Table 6. The parity plots for the ridge regression and multiple linear regression are given in Figs. 7 and 8, respectively. From this table, baseline regression techniques though outperform the proposed algorithm for easy-to-model outputs like the TMRT, face difficulty in modelling outputs like spread of CSD and MCS (as evident from the inferior values of R^2 and RMSE when compared with the proposed algorithm). The proposed algorithm could, however, ensure that these outputs also have superior values of R^2 and RMSE. Additionally, the baseline regression techniques could only have a minuscule advantage over the proposed

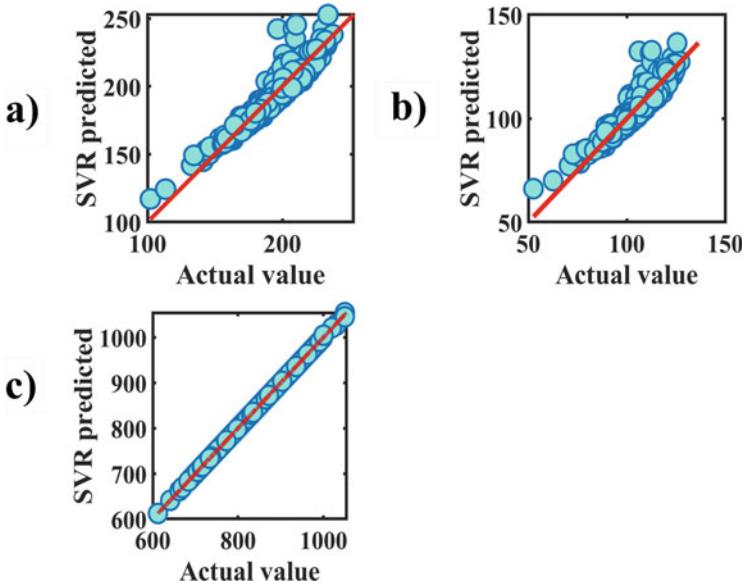


Fig. 8 Parity plot for multiple linear regression for (a) MCS, (b) spread of CSD and (c) TMRT

algorithm in terms of R^2 and RMSE for the easy-to-model output. Also, the average R^2 and RMSE of the proposed algorithm over all the outputs is better than the corresponding values of the baseline regression techniques.

4.4 Comparison of Proposed Algorithm with Fivefold Cross Validation

The training data set is divided into batch of 600 data points to perform fivefold cross validation. Each batch of 600 data points is used for validation exactly once while the remaining batch of data points are used for training the SVR model. Fivefold cross validation is performed only on the finally obtained Pareto optimal SVR models for all the outputs. The training and testing RMSE for each of the five batches along with the average RMSE of the training and testing is provided in Tables 7, 8, and 9 alongside the RMSE of the test data with the proposed algorithm. Additionally, the computational time for these methods is also provided.

Table 7 Comparison results of fivefold cross validation with proposed algorithm for MCS

Results of fivefold cross validation					Validation RMSE for batch no.					Avg. training RMSE			Avg. validation RMSE			Time (s)		Results of proposed algorithm	
Training RMSE for batch no.			1	2	3	4	5	1	2	3	4	5	1	2	3	Validation RMSE	Validation Time (s)	Validation Time (s)	
0.0878	0.0876	0.0852	0.0853	0.1129	0.1043	0.1028	0.1037	0.1047	0.1284	0.0918	0.1088	0.225.02	0.0479	0.186					
0.0748	0.0730	0.0740	0.0737	0.0704	0.0781	0.0761	0.0749	0.0722	0.0738	0.0732	0.0750	0.884.28	0.0782	0.062					
0.0142	0.0121	0.0137	0.0141	0.0169	0.0195	0.0182	0.0191	0.0183	0.0217	0.0142	0.0194	204.88	0.145	0.031					
0.0350	0.0351	0.0344	0.0336	0.0339	0.0388	0.0396	0.0384	0.0350	0.0380	0.0344	0.0379	323.25	0.0143	37.05					
0.0093	0.0095	0.0101	0.0203	0.0093	0.0162	0.0166	0.0171	0.0237	0.0169	0.0117	0.0181	670.42	0.04	0.699					
0.0630	0.0618	0.0625	0.0616	0.0651	0.0661	0.0652	0.0639	0.0610	0.0678	0.0628	0.0648	659.66	0.0245	4.496					
0.0265	0.0268	0.0265	0.0297	0.0268	0.0305	0.0317	0.0316	0.0328	0.0314	0.0273	0.0316	249.31	0.0353	1.031					
0.0610	0.0634	0.0621	0.0628	0.0614	0.0640	0.0667	0.0635	0.0621	0.0637	0.0621	0.0640	518.05	0.0624	0.093					
0.0186	0.0167	0.0178	0.0148	0.0142	0.0232	0.0222	0.0227	0.0227	0.0203	0.0209	0.0164	0.0219	706.17	0.0226	5.710				
0.0312	0.0278	0.0288	0.0271	0.0282	0.0343	0.0322	0.0327	0.0294	0.0330	0.0286	0.0323	311.20	0.065	0.078					
0.0812	0.0777	0.0773	0.0797	0.0802	0.0834	0.0806	0.0777	0.0785	0.0826	0.0792	0.0806	567.94	0.0532	0.125					
0.0475	0.0505	0.0469	0.0478	0.0451	0.0513	0.0541	0.0496	0.0483	0.0482	0.0476	0.0503	812.66	0.0215	7.503					
0.0093	0.0167	0.0109	0.0203	0.0203	0.0165	0.0222	0.0172	0.0235	0.0247	0.0155	0.0208	305.03	0.0291	1.685					
0.0753	0.0706	0.0720	0.0733	0.0749	0.0786	0.0736	0.0726	0.0720	0.0780	0.0732	0.0749	737.88	0.0485	0.156					
0.0092	0.0205	0.0160	0.0094	0.0198	0.0166	0.0250	0.0207	0.0156	0.0240	0.0150	0.0204	391.53	0.0342	0.890					
0.0347	0.0334	0.0391	0.0349	0.0338	0.0386	0.0381	0.0424	0.0366	0.0370	0.0352	0.0385	776.78	0.0262	3.105					
0.0726	0.0691	0.0690	0.0718	0.0706	0.0759	0.0722	0.0700	0.0704	0.0739	0.0706	0.0725	564.92	0.0175	21.90					
0.0472	0.0478	0.0465	0.0422	0.0471	0.0506	0.0517	0.0491	0.0424	0.0507	0.0462	0.0489	337.56	0.0741	0.062					
0.0186	0.0206	0.0115	0.0165	0.0206	0.0229	0.0235	0.0173	0.0206	0.0251	0.0176	0.0219	140.69	0.0403	0.265					
0.0262	0.0277	0.0274	0.0268	0.0276	0.0307	0.0326	0.0325	0.0296	0.0322	0.0272	0.0315	350.31	0.0489	0.187					
0.0565	0.0599	0.0526	0.0559	0.0574	0.0600	0.0633	0.0546	0.0558	0.0609	0.0564	0.0589	533.83	0.0354	0.843					
0.0253	0.0216	0.0224	0.0203	0.0253	0.0289	0.0266	0.0278	0.0245	0.0298	0.0230	0.0275	200.22	0.0169	15.39					

0.0127	0.0136	0.0203	0.0210	0.0120	0.0187	0.0246	0.0193	0.0159	0.0214	0.331.06	0.0365	0.779		
0.0229	0.0166	0.0166	0.0169	0.0181	0.0262	0.0225	0.0226	0.0218	0.0238	0.0182	0.0234	373.58	0.0264	3.007
0.0201	0.0191	0.0197	0.0213	0.0208	0.0246	0.0245	0.0253	0.0246	0.0255	0.0202	0.0249	273.80	0.0283	2.046
0.0434	0.0402	0.0430	0.0465	0.0410	0.0472	0.0443	0.0460	0.0473	0.0446	0.0428	0.0459	1187.86	0.02	10.093
0.0687	0.0686	0.0690	0.0700	0.0813	0.0944	0.0963	0.0977	0.0985	0.1079	0.0715	0.0990	196.59	0.0302	1.343
0.0139	0.0180	0.0137	0.0181	0.0139	0.0202	0.0230	0.0201	0.0224	0.0205	0.0155	0.0212	579.45	0.0254	3.171
0.0214	0.0065	0.0066	0.0158	0.0133	0.0251	0.0148	0.0152	0.0201	0.0193	0.0127	0.0189	702.80	0.1369	0.031
0.0398	0.0413	0.0417	0.0437	0.0413	0.0440	0.0451	0.0447	0.0448	0.0449	0.0416	0.0447	605.97	0.0219	7.257
0.0658	0.0668	0.0656	0.0980	0.0779	0.0875	0.0920	0.0936	0.1178	0.1033	0.0748	0.0988	174.77	0.1434	0.031
0.0192	0.0198	0.0065	0.0101	0.0190	0.0232	0.0231	0.0147	0.0153	0.0234	0.0149	0.0199	328.94	0.018	13.476
0.0240	0.0226	0.0250	0.0242	0.0271	0.0272	0.0278	0.0296	0.0275	0.0315	0.0246	0.0287	271.81	0.1411	0.031
0.0249	0.0291	0.0279	0.0265	0.0283	0.0290	0.0335	0.0317	0.0288	0.0317	0.0273	0.0309	651.14	0.0249	3.210
0.0142	0.0142	0.0138	0.0168	0.0150	0.0202	0.0207	0.0202	0.0210	0.0212	0.0148	0.0207	544.89	0.0286	1.671
0.0206	0.0207	0.0092	0.0196	0.0094	0.0252	0.0252	0.0172	0.0239	0.0173	0.0159	0.0218	367.91	0.0254	3.187
0.0404	0.0426	0.0415	0.0416	0.0401	0.0440	0.0465	0.0445	0.0429	0.0439	0.0412	0.0444	517.84	0.0184	13.164

Table 8 Comparison results of fivefold cross validation with proposed algorithm for spread of CSD

Results of fivefold cross validation					Validation RMSE for batch no.					Results of proposed algorithm				
Training RMSE for batch no.														
1	2	3	4	5	1	2	3	4	5	Avg. training RMSE	Avg. validation RMSE	Time (s)	Validation RMSE	Time (s)
0.0372	0.0364	0.0349	0.0352	0.0365	0.0404	0.0401	0.0379	0.0367	0.0404	0.0360	0.0391	267.13	0.0129	37.658
0.0941	0.0768	0.0753	0.0909	0.0923	0.1100	0.0972	0.0953	0.1102	0.1133	0.0859	0.1052	118.34	0.1333	0.046
0.0255	0.0251	0.0238	0.0291	0.0242	0.0292	0.0295	0.0282	0.0316	0.0295	0.0255	0.0296	970.19	0.0301	0.75
0.0538	0.0530	0.0554	0.0561	0.0514	0.0563	0.0566	0.0563	0.0560	0.0543	0.0539	0.0559	709.13	0.0526	0.109
0.0306	0.0322	0.0307	0.0284	0.0299	0.0339	0.0366	0.0339	0.0309	0.0341	0.0303	0.0339	339.66	0.0347	0.265
0.0396	0.0365	0.0392	0.0463	0.0412	0.0429	0.0404	0.0428	0.0472	0.0456	0.0406	0.0438	594.83	0.1391	0.015
0.0175	0.0147	0.0139	0.0172	0.0134	0.0215	0.0203	0.0199	0.0213	0.0200	0.0154	0.0206	308.58	0.0204	5.724
0.0261	0.0263	0.0271	0.0264	0.0259	0.0285	0.0309	0.0311	0.0289	0.0307	0.0264	0.0300	502.23	0.0173	9.546
0.0494	0.0492	0.0444	0.0471	0.0485	0.0521	0.0531	0.0463	0.0475	0.0520	0.0477	0.0502	702.14	0.0210	4.773
0.0268	0.0265	0.0289	0.0265	0.0264	0.0302	0.0311	0.0327	0.0296	0.0312	0.0270	0.0310	1227.09	0.0215	4.367
0.0256	0.0269	0.0258	0.0267	0.0265	0.0289	0.0314	0.0297	0.0290	0.0308	0.0263	0.0299	515.08	0.0420	0.155
0.0392	0.0366	0.0332	0.0375	0.0360	0.0424	0.0407	0.0369	0.0387	0.0402	0.0365	0.0398	577.05	0.0309	0.937
0.0339	0.0395	0.0368	0.0333	0.0342	0.0370	0.0435	0.0398	0.0347	0.0377	0.0355	0.0385	430.95	0.071	0.078
0.0792	0.0825	0.0796	0.0780	0.0510	0.0931	0.0971	0.0955	0.0964	0.0748	0.0741	0.0914	119.28	0.0258	1.562
0.0229	0.0218	0.0223	0.0217	0.0228	0.0266	0.0263	0.0263	0.0252	0.0281	0.0223	0.0265	319.13	0.0255	2.093
0.0092	0.0092	0.0105	0.0093	0.0192	0.0156	0.0160	0.0161	0.0150	0.0246	0.0115	0.0175	780.02	0.0198	6.317
0.0252	0.0203	0.0225	0.0242	0.0232	0.0279	0.0246	0.0265	0.0266	0.0278	0.0230	0.0267	106.42	0.0218	4.101
0.0123	0.0166	0.0143	0.0166	0.0140	0.0180	0.0212	0.0188	0.0203	0.0202	0.0148	0.0197	368.03	0.1314	0.046
0.0224	0.0258	0.0232	0.0233	0.0248	0.0262	0.0295	0.0272	0.0263	0.0290	0.0239	0.0276	307.02	0.0168	11.593
0.0460	0.0480	0.0452	0.0459	0.0512	0.0486	0.0517	0.0468	0.0465	0.0543	0.0473	0.0496	740.27	0.1363	0.015
0.0766	0.0739	0.0763	0.0750	0.0765	0.0784	0.0773	0.0761	0.0744	0.0793	0.0757	0.0771	815.50	0.0192	7.722
0.0152	0.0190	0.0121	0.0187	0.0185	0.0194	0.0227	0.0170	0.0217	0.0234	0.0167	0.0208	128.75	0.0283	1.339

	0.0578	0.0583	0.0607	0.0619	0.0617	0.0598	0.0619	0.0616	0.0646	0.0601	0.0619	0.0645	0.0488	0.109
0.0762	0.0732	0.0774	0.0736	0.0786	0.0781	0.0770	0.0770	0.0731	0.0818	0.0758	0.0774	286.70	0.0148	22.820
0.0251	0.0257	0.0258	0.0278	0.0273	0.0294	0.0301	0.0294	0.0309	0.0316	0.0263	0.0303	632.48	0.0223	3.617
0.0259	0.0231	0.0248	0.0243	0.0268	0.0296	0.0271	0.0292	0.0272	0.0318	0.0250	0.0290	663.41	0.0276	1.527
0.0174	0.0214	0.0211	0.0183	0.0172	0.0212	0.0250	0.0248	0.0214	0.0232	0.0191	0.0231	158.17	0.0214	4.195
0.0242	0.0250	0.0272	0.0256	0.0268	0.0280	0.0292	0.0310	0.0280	0.0318	0.0258	0.0296	285.88	0.078	0.046
0.0169	0.0171	0.0186	0.0182	0.0164	0.0212	0.0222	0.0231	0.0220	0.0222	0.0174	0.0221	273.72	0.0293	1.468
0.0137	0.0151	0.0134	0.0134	0.0148	0.0189	0.0207	0.0192	0.0182	0.0208	0.0141	0.0196	551.22	0.0156	13.867
0.0731	0.0704	0.0734	0.0710	0.0737	0.0751	0.0736	0.0735	0.0704	0.0764	0.0723	0.0738	712.05	0.0297	1.515
0.0611	0.0617	0.0639	0.0643	0.0643	0.0633	0.0653	0.0645	0.0639	0.0671	0.0631	0.0648	695.88	0.0197	6.851
0.0152	0.0185	0.0184	0.0143	0.0181	0.0189	0.0224	0.0222	0.0185	0.0231	0.0169	0.0210	117.72	0.0726	0.062
0.0522	0.0537	0.0525	0.0558	0.0541	0.0547	0.0574	0.0536	0.0557	0.0572	0.0536	0.0557	609.19	0.0137	35.070
0.0105	0.0103	0.0188	0.0102	0.0180	0.0162	0.0168	0.0228	0.0154	0.0228	0.0135	0.0188	320.52	0.0572	0.093
0.0752	0.0613	0.0598	0.0760	0.0599	0.0905	0.0739	0.0741	0.0922	0.0799	0.0665	0.0821	129.33	0.0172	11.125
0.0818	0.0664	0.0841	0.0549	0.0826	0.0951	0.0825	0.0992	0.0778	0.1006	0.0740	0.0910	129.83	0.1317	0.015

Table 9 Comparison results of fivefold cross validation with proposed algorithm for TMRT

Results of fivefold cross validation						Validation RMSE for batch no.						Results of proposed algorithm		
Training RMSE for batch no.									Avg. training RMSE			Avg. validation RMSE		
1	2	3	4	5	1	2	3	4	5	Time (s)	Validation RMSE	Time (s)	Validation RMSE	
0.0114	0.0115	0.0124	0.0126	0.0137	0.0139	0.0154	0.0141	0.0153	0.0184	0.0123	0.0154	707.66	0.017	1.513
0.0438	0.0446	0.0440	0.0435	0.0439	0.0446	0.0464	0.0440	0.0446	0.0460	0.0440	0.0451	409.73	0.0612	0.046
0.0041	0.0045	0.0042	0.0050	0.0045	0.0072	0.0087	0.0072	0.0087	0.0110	0.0044	0.0086	894.59	0.0101	12.167
0.0138	0.0159	0.0152	0.0140	0.0162	0.0157	0.0196	0.0167	0.0166	0.0198	0.0150	0.0177	683.42	0.0203	0.484
0.0967	0.0829	0.0887	0.0894	0.0890	0.1172	0.1033	0.1025	0.1041	0.1028	0.0894	0.1060	626.45	0.162	0.031
0.0277	0.0277	0.0275	0.0287	0.0277	0.0291	0.0300	0.0282	0.0297	0.0307	0.0279	0.0295	637.11	0.0439	0.062
0.0070	0.0073	0.0068	0.0066	0.0067	0.0092	0.0114	0.0093	0.0099	0.0126	0.0069	0.0105	514.69	0.0207	0.230
0.0988	0.1043	0.1048	0.0986	0.1003	0.1187	0.1191	0.1140	0.1133	0.1119	0.1014	0.1154	543.09	0.1577	0.046
0.0154	0.0164	0.0164	0.0149	0.0166	0.0175	0.0200	0.0176	0.0174	0.0209	0.0159	0.0187	997.88	0.16759	0
0.0068	0.0069	0.0071	0.0077	0.0071	0.0097	0.0113	0.0096	0.0111	0.0128	0.0071	0.0109	927.02	0.0203	0.484
0.0163	0.0151	0.0155	0.0157	0.0146	0.0183	0.0183	0.0168	0.0182	0.0189	0.0155	0.0181	556.11	0.0246	0.125
0.0141	0.0153	0.0142	0.0144	0.0139	0.0164	0.0191	0.0156	0.0172	0.0184	0.0144	0.0173	424.44	0.0135	3.554
0.0067	0.0064	0.0067	0.0084	0.0085	0.0093	0.0108	0.0093	0.0116	0.0142	0.0074	0.0110	1516.97	0.0122	7.582
0.0103	0.0104	0.0098	0.0095	0.0115	0.0130	0.0145	0.0119	0.0126	0.0166	0.0103	0.0137	1022.81	0.0142	3.167
0.0124	0.0141	0.0127	0.0118	0.0135	0.0149	0.0179	0.0142	0.0144	0.0182	0.0129	0.0159	739.88	0.0327	0.093
0.0140	0.0139	0.0122	0.0119	0.0138	0.0162	0.0176	0.0138	0.0145	0.0183	0.0131	0.0161	1036.31	0.0191	0.718
0.0115	0.0116	0.0111	0.0117	0.0128	0.0140	0.0154	0.0128	0.0145	0.0175	0.0117	0.0148	422.08	0.0158	2.058
0.0097	0.0104	0.0094	0.0098	0.0105	0.0124	0.0141	0.0120	0.0129	0.0155	0.0100	0.0134	301.83	0.011	9.546
0.0055	0.0056	0.0055	0.0063	0.0086	0.0097	0.0081	0.0089	0.0125	0.0056	0.0096	865.94	0.0133	4.058	
0.0147	0.0138	0.0149	0.0156	0.0138	0.0166	0.0173	0.0161	0.0179	0.0184	0.0146	0.0173	704.06	0.0222	0.156
0.0119	0.0124	0.0119	0.0125	0.0127	0.0141	0.0157	0.0136	0.0150	0.0173	0.0123	0.0151	354.17	0.0407	0.062
0.0418	0.0423	0.0427	0.0425	0.0428	0.0426	0.0442	0.0427	0.0437	0.0448	0.0424	0.0436	336.09	0.0096	14.804

0.0049	0.0043	0.0041	0.0043	0.0043	0.0082	0.0084	0.0071	0.0079	0.0108	0.0044	0.0085	872.06	0.0654	0.031
0.0127	0.0102	0.0100	0.0122	0.0103	0.0150	0.0141	0.0121	0.0148	0.0153	0.0110	0.0143	901.66	0.0759	0.031
0.0144	0.0130	0.0135	0.0138	0.0121	0.0166	0.0166	0.0150	0.0163	0.0171	0.0124	0.0163	616.89	0.0101	12.511
0.0885	0.0954	0.0950	0.0947	0.0684	0.1164	0.1128	0.1060	0.1126	0.0897	0.0884	0.1075	387.91	0.0206	0.421
0.0096	0.0093	0.0100	0.0114	0.0116	0.0123	0.0134	0.0122	0.0142	0.0167	0.0104	0.0137	1239.02	0.0144	3.199
0.0043	0.0043	0.0045	0.0043	0.0044	0.0073	0.0086	0.0073	0.0079	0.0108	0.0044	0.0084	634.48	0.0176	1.167
0.0154	0.0161	0.0159	0.0156	0.0160	0.0172	0.0195	0.0171	0.0179	0.0202	0.0158	0.0184	977.75	0.0142	3.320
0.0161	0.0158	0.0161	0.0158	0.0152	0.0181	0.0191	0.0171	0.0181	0.0195	0.0158	0.0184	406.02	0.0268	0.125
0.0136	0.0137	0.0129	0.0153	0.0155	0.0159	0.0174	0.0143	0.0175	0.0198	0.0142	0.0170	704.00	0.0090	26.238
0.0062	0.0067	0.0062	0.0068	0.0062	0.0091	0.0108	0.0086	0.0099	0.0123	0.0064	0.0101	543.81	0.0275	0.109
0.0248	0.0277	0.0279	0.0282	0.0280	0.0264	0.0305	0.0284	0.0299	0.0309	0.0273	0.0292	565.50	0.0149	3.101
0.0114	0.0134	0.0127	0.0111	0.0137	0.0139	0.0171	0.0142	0.0137	0.0183	0.0125	0.0155	913.00	0.0117	8.394
0.0973	0.0932	0.1022	0.0948	0.1104	0.1133	0.1070	0.1082	0.1058	0.1157	0.0996	0.1100	775.34	0.0097	13.371
0.0160	0.0156	0.0159	0.0157	0.0149	0.0179	0.0190	0.0170	0.0179	0.0192	0.0156	0.0182	751.02	0.0118	8.796
0.0084	0.0080	0.0099	0.0076	0.0100	0.0110	0.0115	0.0118	0.0108	0.0153	0.0088	0.0121	980.70	0.0349	0.0781

5 Conclusion

Research work on MSMPR crystallization process has advanced significantly. However, the capability to have fast computational models remains evasive. To facilitate concurrent optimization of the MSMPR, we propose a data-based modeling approach followed by optimization using the developed model. The highlights are listed below:

1. To help manage redundant data points a sample size estimation algorithm is presented. This algorithm helps in avoiding data points with similar characteristics.
2. The trade-off balancing approach for optimizing the hyper-parameters of SVR has resulted in the simultaneous development of plethora of SVR models.
3. The optimal way of finding the kernel function has resulted in better function approximation. This is additionally spear-headed by the judicious use of kernel parameter for the inputs.
4. Simulation studies indicate that the concurrent optimization of complex processes like MSMPR is indeed feasible.
5. Comparison with baseline regression techniques establishes the superiority of the models developed using the proposed algorithm both in terms of $RMSE$, R^2 and training time.

References

1. Lyu D, Lyu X, Huang L, Fang B. Effects of three kinds of anti-amyloid- β drugs on clinical, biomarker, neuroimaging outcomes and safety indexes: a systematic review and meta-analysis of phase II/III clinical trials in Alzheimer's disease. *Ageing Res Rev.* 2023;88:101959.
2. Szilágyi B. Modeling and analysis of MSMPR cascades involving nucleation, growth and agglomeration mechanisms with slurry recycling. *Chem Eng Res Des.* 2021;174:42–56.
3. Aamir E, Nagy ZK, Rielly CD, Kleinert T, Judat B. Combined quadrature method of moments and method of characteristics approach for efficient solution of population balance models for dynamic modeling and crystal size distribution control of crystallization processes. *Ind Eng Chem Res.* 2009;48:8575–84.
4. Szilágyi B, Nagy ZK. Graphical processing unit (GPU) acceleration for numerical solution of population balance models using high resolution finite volume algorithm. *Comput Chem Eng.* 2016;91:167–81.
5. Manee V, Baratti R, Romagnoli JA. Learning to navigate a crystallization model with deep reinforcement learning. *Chem Eng Res Des.* 2022;178:111–23.
6. Zheng Y, Wang X, Wu Z. Machine learning modeling and predictive control of the batch crystallization process. *Ind Eng Chem Res.* 2022;61:5578–92.
7. Miriyala SS, Pujari KNS, Naik S, Mitra K. Evolutionary neural architecture search for surrogate models to enable optimization of industrial continuous crystallization process. *Powder Technol.* 2022;405:117527.
8. Griffin DJ, Grover MA, Kawajiri Y, Rousseau RW. Data-driven modeling and dynamic programming applied to batch cooling crystallization. *Ind Eng Chem Res.* 2016;55(5): 1361–72. <https://doi.org/10.1021/acs.iecr.5b03635>.

9. Scholkopf B, Smola AJ. Learning with kernels support vector machines, regularization, optimization, and beyond. Cambridge, MA: MIT Press; 2002.
10. Huang J, Jin T, Liang M, Chen H. Prediction of heat exchanger performance in cryogenic oscillating flow conditions by support vector machine. *Appl Thermal Eng.* 2021;182:116053. <https://doi.org/10.1016/J.APPLTHERMALENG.2020.116053>.
11. Peng X, Xu D. Projection support vector regression algorithms for data regression. *Knowl-Based Syst.* 2016;112:54–66.
12. Chou JS, Ngo NT. Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns. *Appl Energy.* 2016;177:751–70.
13. Yang YL, Che JX, Li YY, Zhao YJ, Zhu SL. An incremental electric load forecasting model based on support vector regression. *Energy.* 2016;113:796–808.
14. Li J, Trout BL, Myerson AS. Multistage continuous mixed-suspension, mixed-product removal (MSMPR) crystallization with solids recycle. *Org Process Res Dev.* 2016;20:510–6.
15. Acevedo D, Tandy Y, Nagy ZK. Multiobjective optimization of an unseeded batch cooling crystallizer for shape and size manipulation. *Ind Eng Chem Res.* 2015;54:2156–66.
16. Bishop C. Pattern recognition and machine learning. Singapore: Springer; 2006.
17. Fasshauer GE. Meshfree approximation methods with MATLAB. London: World Scientific; 2007. <https://doi.org/10.1142/6437>.
18. Zhao W, Tao T, Zio E. System reliability prediction by support vector regression with analytic selection and genetic algorithm parameters selection. *Appl Soft Comput.* 2015;30:792–802. <https://doi.org/10.1016/j.asoc.2015.02.026>.
19. Ishibuchi H, Tsukamoto N, Sakane Y, Nojima Y. Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In: Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10. 2010. <https://doi.org/10.1145/1830483.1830578>.
20. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., et al. Parallel problem solving from nature PPSN VI. PPSN 2000. Lecture notes in computer science. Springer, Berlin, doi:https://doi.org/10.1007/3-540-45356-3_83.
21. Inapakurthi RK, Pantula PD, Miriyala SS, Mitra K. Data driven robust optimization of grinding process under uncertainty. *Mater Manuf Process.* 2020;35:1870–6. <https://doi.org/10.1080/1042691420201802042>.

Machine Learning Based Intelligent RPL Attack Detection System for IoT Networks



A. Kannan, M. Selvi, S. V. N. Santhosh Kumar, K. Thangaramya,
and S. Shalini

Abstract Routing Protocol for Low-Power and Lossy Networks (RPL) has been used in the Internet of Things (IoT) with Wireless Sensor Networks (WSNs), but it does not give much focus to protect against routing attacks. Therefore, it is possible for an attacker to utilize the RPL routing system as an initial platform for devastating and crippling attacks on an IoT network. In IoT based networks, RPL provides a minimal level of protection against a wide variety of attacks, which are unique to RPL and are launched in WSNs. Moreover, the traditional Internet and routing security solutions also have memory, processing, and resource limitations that make them ineffective for IoT devices. Several mitigation schemes, such as those based on rule-based learning algorithms, Intrusion Detection Systems (IDSs), and trust computation and management techniques were proposed in the past to improve the safety of IoT networks and routing, but they do not provide the required security. To overcome these security issues, we propose an intelligent IDS in this article to detect and isolate the RPL attacks. For this purpose, we propose a new classification algorithm based on neural networks and genetic algorithms. Moreover, the proposed Neuro Genetic Classification Algorithm (NGCA) detects the nodes which launch RPL and other attacks more accurately and hence it increases the network security in IoT. From the experiments done with NSL-KDD data-set, it is proved that the proposed NGCA is detecting the attacks with higher accuracy than the other classifiers namely Decision Trees, Logistic Regression and Support Vector Machines. It also helps to reduce the RPL attacks in the routing process with enhanced detection rate with reduced false positives.

Keywords IDS · IoT · RPL · ANN · GA and ML

A. Kannan · M. Selvi (✉) · K. Thangaramya · S. Shalini
School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
e-mail: kannan.a@vit.ac.in; selvi.m@vit.ac.in; thangaramya.k@vit.ac.in

S. V. N. Santhosh Kumar
School of Computer Science Engineering and Information Systems, Vellore, Tamil Nadu, India
e-mail: santhoshkumar.svn@vit.ac.in

1 Introduction

The Internet of Things (IoT) consists of a group of objects having their own identifiers linked together over the internet. This can include anything from the tags to the sensors in Wireless Sensor Networks (WSNs) and machine networks to cell phones, networks, sensors and connected cars. However, there are no uniform protocols or specifications for securing the communication from RPL attacks in the IoT [1] based WSN. Major enablers for the rollout of IoT are networks like Low Power and Lossy Networks (LLNs), consisting of a number of devices with minimum resources. Real-world examples of heavy reliance on such networks include [2] smart houses, healthcare and smart cities.

Throughout the IoT, routing is a crucial procedure for linking together the various network nodes that make up the various apps. The IoT is relying fully on the routing system named Routing for Low-Power and Lossy Networks (RPL). Thus, RPL is essential for many IoT based applications. Made from scratch, this protocol is both adaptable and robust, allowing it to function even over connections with poor throughput and a high error rate [3]. RPL's many impressive features include its manipulation of the trickle algorithm, its adaptability to changes in RPL control message structure and frequency, as well as its use of metrics for routing. Although RPL is more suitable for many IoT uses, it is prone to different kinds of attacks [4]. The power of RPL protocol is diminished in IoT networks, because RPL uses distance for its working with LLN, it is represented as a graph [5]. Now, the neighbouring nodes choose this coordinating node as their parent by transmitting Destination Advertisement Object (DAO), and this node is ranked 2. Until all nodes have joined the RPL network, this procedure will be repeated. However, the RPL attack is more vulnerable due to its structure and communication. So, it is necessary to safeguard message exchanges between IoT devices from two major Denial of Service (DoS) threat namely the selected data forwarding attacks and flooding attacks. Both can be started through the RPL protocol while data packets are being sent to their final destination. Distributed DoS (DDoS) attacks must also be taken into account by an intelligent Intrusion Detection System (IDS) [6, 7]. As IDSs can analyse network traffic trends and sensor node behaviour to single out bad actors, they are better able to find DoS attacks. Designing an IDS to IoT based WSN is important because it helps to prevent the malicious users from breaching the network and stealing data, changing communications, or destroying messages. Further, IDS-based IoT network security can lessen the threats to the aforementioned three pillars of information and network security namely privacy, integrity and availability.

Many different IDSs have been developed in the past to track down the intruders responsible for DoS assaults on the Internet. Also, IDSs can defend against threats from both inside and outside the network in an IoT setting. Because they are able to counteract the efforts of assailants who target the IoT, IDSs serve as a protective barrier for IoT data transmissions. Anomaly-based and specification-based IDS approaches are the most useful IDSs in an RPL-based network for identifying

internal threats [8]. Signature-based systems for intrusion detection keep monitoring for activity that is similar to that of recognised threats. While it will be able to spot both common and uncommon forms of assault, it may be hopeless at predicting precisely when they attack. As an added measure, a worldwide anomaly detection system should build a picture of normal network activity and flag any outliers. The existing IDSs have a higher false alarm rate, which makes it harder to detect future attacks. Therefore, the current IDSs are not reliable enough based on their ability to anticipate future events. Machine learning (ML) algorithms have shown promise in improving security and could be used to build an improved IDS. Researchers, especially those interested in using AI to create IDS [9], have made ML methods a priority in recent years. IDS is used in these setups to increase the safety of WSN and consequently boost the network's overall efficiency. Anomaly-based IDS systems aim to keep an eye out for anomalies, both long-term and short-term, to assist with efficient decision-making in the secure routing process [10]. However, it is much simpler to track down instances of misuse because only authorised users can commit such breaches. Spotting attacks that come from the outside is the true test. Therefore, a new IDS has been created in this work which is a robust anomaly detection-based IDS, constructed using ML techniques to lower the number of false alarms and to protect the data transmissions better within the RPL network.

In RPL network, there are different types of attacks that are prone to disturb the network in its communication. In order to handle this, we propose a new IDS for detecting and mitigating the DoS based Intelligent RPL attacks with secure routing in IoT [11]. This work provides a new IDS in which feature selection is done using the back propagation neural networks algorithm [12] and it performs classification using the proposed Neuro Genetic Classification Algorithm (NGCA) with both full features and also with selected features to detect the malicious nodes. Therefore, a Neural Networks (NN) based feature reduction algorithm [13] is developed in this work for performing effective Feature Selection (FS) to support the IDS by integrating it with NN based FS algorithm for picking the optimal and useful features. The features are used for classifying the attacks present in the dataset into four types of attacks such as DoS, User-to-Root (U2R), Remote-to-Local (R2L) and probe attacks. The NGCA has been evaluated and compared to the classifiers namely the Logistic Regression (LR) based classifier, the Decision Tree (DT) and Support Vector Machine (SVM) classification algorithm. Finally, an extended Low-energy adaptive clustering hierarchy (LEACH) protocol [14, 15] is used in this work to perform the routing in this work and it uses the IDS results for performing the secured routing. The evaluation of this work has been focusing on the False Positive Rate (FPR) analysis and time analysis. Based on the experiments, it is shown that this secure routing algorithm provides increase in detection accuracy and to provide reduced FPR in optimal time. The major contribution proposed IDS include accurate detection of intrusions and reduction in FPR.

The rest of this article is given by this sequence. Section 2 is giving a detailed review on RPL attacks and compares the existing work with proposed model. Section 3 makes the discussions on the proposed system. Section 4 gives the

performance analysis on results. Section 5 is on the conclusions and further extensions.

2 Related Works

The works on computer networks using IDSs, security attacks, key management, access control and secure routing algorithms are available now [16, 17]. In particular, the attacks present in IoT were analysed in the past work on mitigation of attacks namely as sink hole, worm hole, sybil and decreased attack with selective forwarding and flooding based DDoS [18] and DoS [19] attacks. Airehrour et al. [20] proposed an extended RPL for protecting the IoT network from rank as well as sybil attacks. Their system relies on mutual trust to identify threats, quarantine infected hosts, and enhances the network's efficiency and ensures reliability. Their RPL attack detection system isolates the malicious nodes from the network and makes optimal routing choices based on the evaluation of trusted nodes. Their work needs to be expanded to handle other types of colluding attacks. Pu [21] proposed a routing protocol for RPL-based IoT to protect against a Sybil attack. Their model can be extended to handle other attacks as well.

Le et al. [22] suggested that RPL is protected from topology attacks like rank attacks, local repair attacks, sinkhole attacks, neighbour attacks, and information solicitation attacks with a specification-based intrusion detection system. Their way of finding problems is to use specifications to build a semi-automatic IDS model, which is then used to protect an RPL-based network architecture. In [23], a secure routing scheme based on game methods and statistical modelling was proposed for IoT security. The secure RPL improves the accuracy in the detection of intruders while drastically cutting down on resource usage. Using the knowledge of IoT context information, this RPL-specific scheme ensures the detection of routing misbehaviour and also the packet dropping based attacks. The limitations of their work is that they improve the detection accuracy further.

Kiran et al. [24] suggested that routing attacks could be found with an IDS for RPL-based IoT. Their solution to this problem uses a set of rules based on both signatures and oddities to look for signs of bad behaviour in the network data. Their work has limitations in its ability to perform in a real-world testbed and to extend its attack detection algorithm to generalise feature behaviours across various devices on large-scale networks. They used Network Simulator to evaluate how well their IDS is performing in comparison to the current standard measure. An IDS using a distributed monitoring architecture in RPL-based routing environments was proposed by Mayzaud et al. [25]. Extensive tests were conducted by them to determine their model's solution effectiveness, and its scalability is measured in light of a monitoring node placement technique. Their work could be improved by conducting complementary experiments in real-world infrastructures using a wider variety of devices that support the RPL. Rani et al. [26] depicted about existing works on RPL attacks originated as sink hole attacks. Muzammal et al. [27] proposed a trust aware

secured routing algorithm to safeguard the IoT from RPL attacks. The objective is to offer protection against standard RPL assaults. When compared to traditional trust models, their work was better in terms of throughput, detection accuracy, mobility, and scalability. Due to their limited resources, IoT gadgets could benefit from a trust-based strategy that is also relatively inexpensive. In order to overcome the limitations of their work, they need to conduct extensive testing in a real-world setting and implement a wide variety of tactics.

Even though many algorithms are already available, the FPR provided in the current systems are high and hence they are not enough for providing security in WSN. Hence, an IDS and NGCA is proposed in this work for the reduction of the FPR and also to improve the detection rate. This is supported by an existing neural network-based FS algorithm for further improvement in performance.

3 Proposed Methodology

In this proposed work, an intelligent IDS is proposed for detecting the RPL attacks using machine learning based classification algorithm called NGCA. In addition, an existing feature selection algorithm based on neural networks is applied for FS. This NGCA classifies DoS, U2R, R2L and probe attacks more accurately. Moreover, the RPL attacks are detected in the routing process by using the IDS developed there by applying the NGCA developed for securing the communication in IoT based WSN. The feature selection phase of this work uses back propagation neural networks for selecting the most contributing features. After this, the proposed classification algorithm called NGCA is used to perform the classification by applying the neural networks with genetic algorithm based weight optimization for enhancing the detection accuracy. The proposed work uses RPL for routing with feedback from the IDS for maintaining security in the routing process.

3.1 Description of Dataset

In this work, the benchmark NSL-KDD dataset was utilised to train and test the proposed IDS model. This dataset contains a total of 41 network-related characteristics, which have been further classified into normal, attack and countermeasure categories. For this dataset, the class label can be considered the 42nd attribute out of a possible 42. The class label contains both positive and negative signs. More than a million entries make up the dataset, but this work has considered only about 100,000 of them by picking out the rows with no null values and different properties. The tenfold cross-validation method was applied to validate the proposed system, wherein 90% and 10% of data were respectively used in the phases of training and testing of the model. The classification times has been decreased by compressing the training and testing datasets and also by removing the records that are ineffective in

the classification process. Since, the NSL-KDD is widely used as a validation dataset in the current literature, many researchers are turning to it to demonstrate how well their IDS performs in contrast to the standard methods. In intrusion detection, the methods that use machine learning are fully explored and analysed in [28]. Based on research in the literature, we found that this dataset is more appropriate for the evaluation of the proposed IDS. Moreover, this research work has used the ANN algorithm for FS and the proposed NGCA for classification of the dataset.

3.2 Intrusion Detection System

An IDS looks for suspicious activity on a network or system and notifies them to the administrators or other users so that any potential threats can be dealt with before they do a lot of damage. IDSs are prone to increase FPR and false negatives because there are so many different kinds of attacks and new attacks are hard to predict (not raising an alarm when there is an attack). Two broad types of intrusion detection systems (IDSs) exist: those that rely on signatures and those that rely on anomalies. Signature-based detections compare the current activity on a network or device to attack patterns, or signatures, that have already been set up. This method has a significant number of false negatives, despite its low error rate, cannot recognise new attacks, requires specialised knowledge of each attack, and requires a lot of storage space. The cost of storage increases as more attacks are detected. As an example, anomaly-based detections learn how a network or device is supposed to behave and then alert administrators to any unexpected behaviour. Although this method is effective at spotting novel attacks, it is prone to high rates of false positives and false negatives because it may trigger false alerts and/or fail to spot attacks when they only display slight deviations from the baseline. According to Sharma and Verma [29], IDS for 6LoWPAN networks should protect against attacks from the traditional Internet, take into account the fact that intruders in a 6LoWPAN can cause damage to both the 6LoWPAN and the Internet, and balance these competing expenses.

In this work, we designed a new secured communication model using RPL and also based on an IDS that uses feature selection with neural networks and classification based on the proposed classification algorithm named NGCA. This IDS looks for intruders in the WSN using the existing neural-network based FS algorithm and the proposed NGCA classification algorithm is introduced in this work as new classifier. The main uses of this IDS are increase in detection accuracy, Packet Delivery Rate (PDR), network throughput with reduction of delay as well as FPR.

3.2.1 Eliminating Malicious Nodes from RPL

Once the hostile nodes have been identified in routing process, they must be removed from the network immediately. When encountering a suspicious

component, it's best to simply disregard it, and this calls for verification logical and physical addresses in the IoT are readily spoofable and expose devices to cyberattacks. Both whitelists and blacklists can be used here to avoid connecting to malicious networks. In contrast to a blacklist, which includes only malicious nodes, a whitelist only contains legal ones. When compared to blacklisting, whitelisting is simpler to maintain but has limited scalability. In existing systems, the authors proposed methods using a whitelist because it is simple to manage in the presence of many attackers, and there will only be a small number of devices under a single 6BR (Broader Router) or in a single RPL DODAG [30]. However, an RPL network may contain hundreds of nodes. Blacklists simplify administration in such massive networks. To prevent flooding attacks the proposed IDS, we use the proposed NGCA for improving the performance of detection accuracy more efficiently in IoT.

3.2.2 Feature Reduction

FS is for reducing and selecting relevant features from the whole set of features because learning is hindered by these superfluous features, and learned models benefit little from them. To simplify the learning process and improve the generalisation ability of the resulting models, the FS method is frequently employed. The FS part is most important in the forecasting of future and it also improves the performance model [12, 13]. As a rule, the information undergoes some form of pre-processing before the artificial neural network model is implemented.

3.3 Proposed Neuro Genetic Algorithm for Classification

Artificial Neural Networks (ANNs) are modelled based on neurons present in human brains. An ANN predicts the output from inputs, based on the training procedure. All ANNs require three distinct layers: input, hidden and output layers [31]. Weights will be placed on each unit's connection to all others. The source's effect is defined by the weight.

An artificial neuro genetic algorithm called NGCA is proposed for optimal classification. For this purpose, the ANN takes n inputs and provides k outputs. It uses the product of inputs and weights added with bias function (bf) value to get an input to the activation function that gives the output z using the formula given in Eq. (1).

$$z = f \left(\sum_{r=1}^n w_{tr} * x_r + bf \right) \quad (1)$$

Here, wt represents weights and x represents inputs. In order to instruct a neural network, we need both the input and the associated outputs. During the training period, the error can be kept to a minimum by gradually increasing the weights. The neural network is then put to further use in testing the information after it has been trained. If the bias function value becomes zero, then the output is represented by Eq. (2).

$$z = f \left(\sum_{r=1}^n w t_r * x_r \right) \quad (2)$$

3.3.1 Reproduction and Mutation

In GA, how well the process works depends a lot on how well the individuals reproduce and change from one generation to the next. After assessing the present generation, a list of prospective parents can be made. All members of the current group are represented here, along with estimates of their relative fitness. Crossover likelihood and individual health have to be taken into account when deciding on a set of two parents. In this work, two communicating nodes are selected as parents. They are made to perform reproduction using cross over and mutation operations. For this purpose, the data pertaining to the nodes are encoded using binary encoding. Next, single point crossover operation is applied to get the new generations. In addition, one mutation operation is performed either on the last bit of the binary representation obtained after cross over or it follows the random selection for mutation. This procedure continues until a fully formed generation has been produced. Here, the terminating condition is either the maximum number of iterations is over or two consecutive iterations produced the same value. The child's bit is set to 1 if both parents have that trait; otherwise it is set to 0. In the random selection for mutation, a random bit position is generated, and the offspring bit at that location undergoes a mutation with a predetermined chance. Until all the bits in the baby are one, the procedure will keep repeating itself (k features). Since GA hinges on both variety and fitness, a healthy equilibrium between the two is essential.

3.3.2 Fitness Function

In GA, the fitness function finds the best parents for the next generation. This helps to increase the overall fitness of the population. The proposed neuro genetic [29] classification algorithm uses back propagation neural networks for performing the basic classification. In this model, the weight adjustments are carried out using neuro genetic classification algorithms [30]. The neuro genetic algorithm performs the selection of most optimal values for results by using the operations namely selection of parents, crossover and mutation. Moreover, we use a fitness function that was

proposed in the IDS developed by Subramani et al. [32]. The fitness function and its corresponding accuracy is defined in Eqs. (3), (4), and (5) as follows:

$$\begin{aligned} Fitness_{value} = & W_1 * Accuracy + W_2 * \left(\frac{1}{CountofOnes} \right) + W_3 \\ & * \left(\frac{1}{CountofZeros} \right) \end{aligned} \quad (3)$$

$$Accuracy = \frac{TPR + TNR}{TPR + FPR + FNR + TNR} \quad (4)$$

or

$$Accuracy = \frac{NumberofSelectedAttributes}{TotalNumberofAttributes} \quad (5)$$

Here, the weights are $W_1 = 0.4$, $W_2 = 0.4$, $W_3 = 0.2$. Moreover, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR) have been used to find the accuracy. Moreover, we use a bias function in the proposed classifier for enhancing the classification accuracy which is defined here in Eq. (6).

$$bf(s) = \frac{(s - 1)}{(s + 1)}, 0 \leq s \leq 1 \quad (6)$$

This work uses min-max normalization, activation function, bias function and GA for classification. The steps of the proposed NGCA are as follows:

Algorithm 1 Neuro Genetic Classification Algorithm (NGCA)

Input: Dataset, full features and genetic rules.

Output :1. Attacks data on DoS attacks, U2R, R2L and probe attacks.

2. Classification of nodes as genuine nodes and attackers.

Step-1: Read the node information and the data regarding their behaviour from the dataset.

Step-2: Read the full set of attributes.

Step-3: Form input dataset x_1, x_2, \dots, x_n from the dataset for selecting the attributes A_1, A_2, \dots, A_m .

Step-4: Read the initial weights wt_1, wt_2, \dots, wt_n and set the activation function for the neuro genetic classifier.

Step-5: Compute the value of

$$f(x) = wt_1 * x_1 + wt_2 * x_2 + \dots + wt_n * x_n + bf(s)$$

$$Sf(x) = \frac{1}{1 + e^{-x}}$$

Step-6: Apply sigmoidal function for x using the formula

Step-7: Apply $f(x)$ having weighted sum of data plus bias function on the value of $y = Sf(x)$ where the bias function $bf(s)$ is computed using the formula,

$$bf(s) = \frac{(s-1)}{(s+1)}, 0 \leq s \leq 1$$

Here, $bf(s)$ is the bias function for the s^{th} layer.

Step-8: Compute the error function values using the root mean square values.

Step-9: Perform back propagation by applying genetic rules for performing weight adjustment as follows:

Step-9a: Select two inputs x_i and x_j as parents through tournament selection.

Step-9b: Apply single point crossover and mutation on the input values.

Step-9c: Compute fitness value using the formula:

$$\begin{aligned} \text{Fitness}_{\text{value}} &= W_1 * \text{Accuracy} + W_2 * \left(\frac{1}{\text{Count of Ones}} \right) + \\ &\quad W_3 * \left(\frac{1}{\text{Count of Zeros}} \right) \\ \text{Accuracy} &= \frac{\text{TPR} + \text{TNR}}{\text{TPR} + \text{FPR} + \text{FNR} + \text{TNR}} \end{aligned}$$

Step-9d: Use the fitness values and find the new weights and perform the computations again until convergence to perform feature selection.

Step-10: Apply the attributes selected from feature selection component and use the dataset along with selected features to perform the classification of data to obtain the attack details with respect to DoS, probe, U2R and R2L attacks.

Step-11: Identify the attacker nodes that generated the attacks and isolate them to form the genuine set of nodes.

Step-12 a: Read the value of k for forming clusters.

Step-12 b: Form clusters of genuine nodes and select CHs for each cluster by applying k-means clustering algorithm.

Step-13: Apply LEACH protocol and find the route.

Step-14: Route the data through genuine nodes to the base station.

Step-15: Return node classification details and attack details.

Step-16: End

The new classifier called NGCA explained here is used for detecting and identifying malicious nodes in IoT based WSN to develop a new IDS. In addition, the LEACH protocol has been used here for performing the cluster-based routing which is working by finding the best and secured route in WSN using the detection of attacks. Furthermore, the proposed classification algorithm namely NGCA is able to perform the intrusion detection more accurately. The main advantages of the classification algorithm include increased security, reduced FPR and delay and increases the overall network performance. This work achieves energy efficiency [33] by the deployment of the proposed IDS in the CHs and also at the base station. Moreover, the proposed NGCA provides higher classification accuracy than the ensemble-based classification algorithms [34].

4 Results and Discussions

In this work, an IDS has been implemented and tested using the NS-3 simulator and then it is tested in Python with real-world network trace data using the python programming environment. This work has been tested with NSL-KDD 2018 dataset and finally using the network trace data obtained from NS-3 simulator. For carrying out the simulations, attacks were generated and tested in NS-3 simulations as well as python programming by considering the four types of attacks namely DoS, Probing, U2R and R2L attack_types. These were simulated for 15 min duration and five experiments were conducted with five different sub sets of data taken from NSL-KDD benchmark dataset. Here, first the NN algorithms is executed by providing the 41 attributes (features) from the given dataset. The number of attributes selected by the algorithm is more optimal and it has selected only the attributes which are contributing the classification process. This system was evaluated using the metrics namely detection accuracy (Table 1).

Figure 1 depicts accuracy of classification with full features for detecting DoS attacks and comparison using different ML techniques namely Decision Tree (DT) algorithm, Logistic Regression (LR) classifier, Support Vector Machine (SVM) classifier with the proposed NGCA.

From Fig. 1, we can understand that NGCA provides higher classification and intrusion detection accuracy than related classifiers based on genetic optimization. Figure 2 shows the detection accuracy comparison for selected features with NGCA

Table 1 Simulation parameters

Parameters	Value
Deployment area	200 m × 200 m
Sensors deployed	100–500 nodes
Starting energy in sensors	2 J
Network packet size	4096 bits
Routing protocol used	LEACH
Mobility modelling	Random waypoint Model

Fig. 1 Detection accuracy analysis on full features for DoS

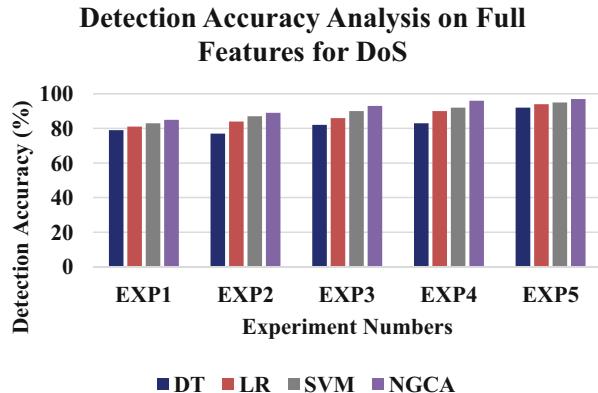


Fig. 2 Detection accuracy analysis with selected FS for DoS attacks

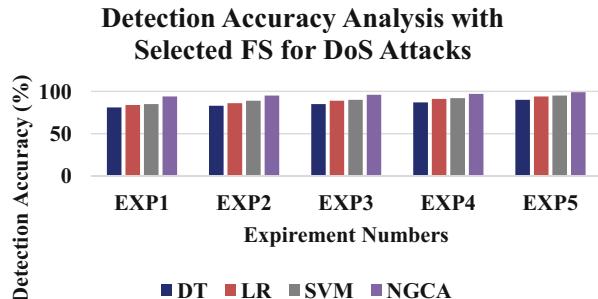
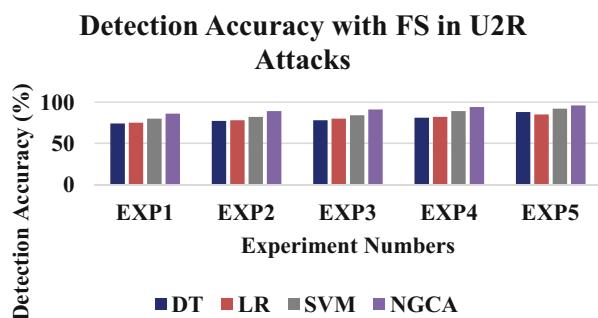


Fig. 3 Detection accuracy FS in U2R attacks



classifier. From Fig. 2, we see that feature selection improves the classification accuracy in NGCA than the other classifiers. This improvement in NGCA is higher than the other classifiers based on genetic optimization. Figure 3 gives the accuracy analysis for U2R attack detection.

From Fig. 3, we prove that NGCA method provides higher detection accuracy in U2R attacks detection than DT, LR and SVM classifiers. Figure 4 depicts the R2L attack detection using NGCA and other ML based classifiers.

Fig. 4 Detection accuracy analysis FS in R2L attacks

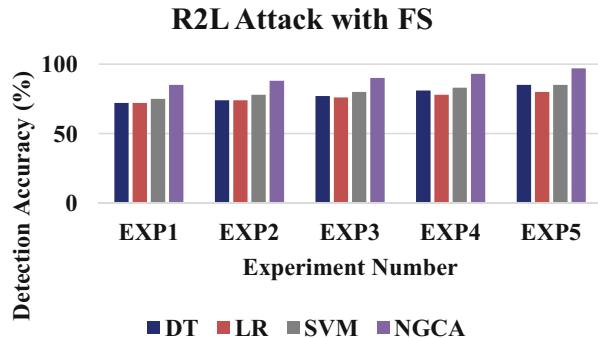


Fig. 5 Detection accuracy with FS for probe attacks

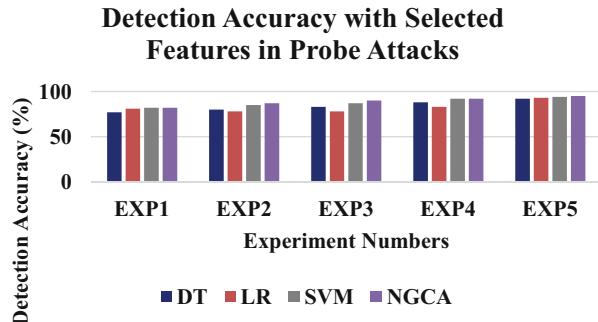
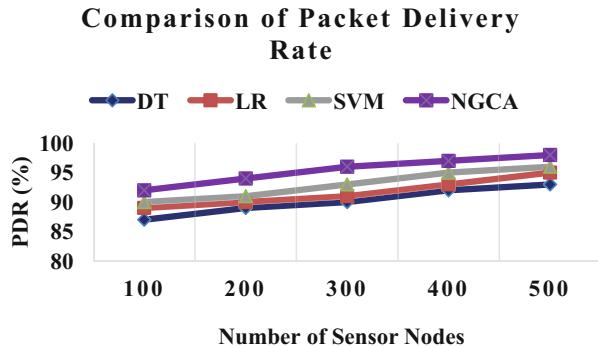
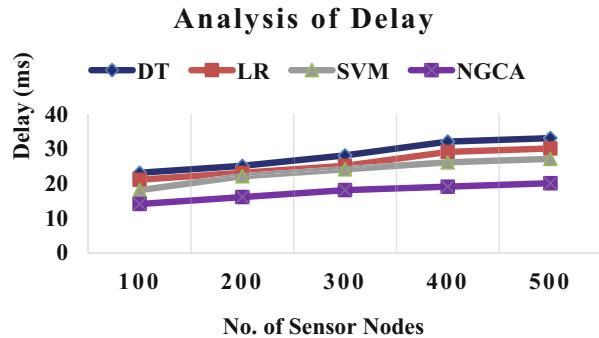
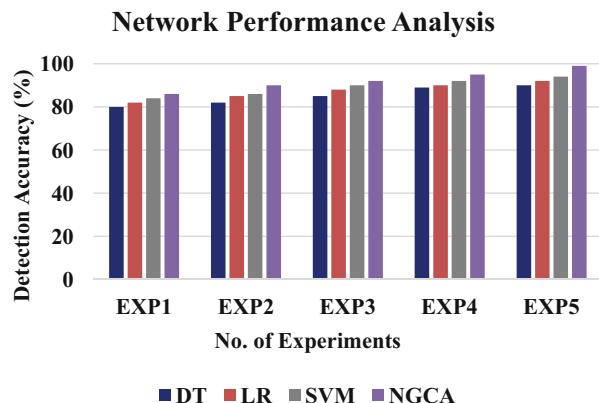


Fig. 6 Comparison of packet delivery rate



From Fig. 4, it can be noted as FS improves the R2L attack detection accuracy in all the classifiers, but NGCA outperform the other classifiers by the effective elimination of attacker nodes using GA. Figure 5 depicts the accuracy comparison of probe attacks between NGCA and 3 existing classifiers with selected features.

From Fig. 5, we see that the feature selection and the use of GA increases the detection accuracy in NGCA than the other classification algorithms. The packet delivery rate comparison is depicted in Fig. 6 for comparing routing with IDS based on NGCA and other machine learning algorithms.

Fig. 7 Analysis of delay**Fig. 8** Network performance analysis

From Fig. 6, we see that the PDR of NGCA is more than the IDS developed using the existing methods namely DT, LR and SVM.

The comparison of delay is depicted in Fig. 7 by considering secure routing with IDS based on DT, LR, SVM and the proposed NGCA with LEACH for routing. From Fig. 7, we observed that the NGCA based IDS is decreasing the delay than DT, LR and SVM classification by using weight optimisation.

Figure 8 shows the overall network performance comparison. From Fig. 8, we see that the network performance obtained using the IDS-based routing protocol developed with the combination of NGCA and LEACH is high when it is compared with the combination of LEACH with other existing methods namely DT, LR and SVM for classifying the attacks in IDS.

5 Conclusion

In this work, we proposed a new NGCA classifier for optimal routing in IoT based WSN by building an IDS. The proposed model performs attacks detection and applies that knowledge for secure routing. In NGCA, genetic algorithms reduce the search space and optimizes the overall convergence criteria and thus produces the cost reduction in terms of reduction in delay. It also increases PDR and network throughput. The major advantages of this IDS are reduction in false positive rate and increase in overall network performance. Future works on this can be the introduction of agent-based communication and the detection of distributed denial of service attacks for enhancing the performance further.

References

1. Son D, Huh S, Lee T, Kwak J. Internet of things system technologies based on quality of experience. *Peer Peer Netw Appl.* 2020;13:475–88.
2. Din IU, Guizani M, Hassan S, Kim BS, Khan MK, Atiquzzaman M, Ahmed SH. The Internet of Things: a review of enabled technologies and future challenges. *IEEE Access.* 2018;7:7606–40.
3. Kim HS, Ko J, Culler DE, Paek J. Challenging the IPv6 routing protocol for low-power and Lossy networks (RPL): a survey. *IEEE Commun Surv Tutor.* 2017;19(4):2502–25.
4. Raouf A, Matrawy A, Lung CH. Routing attacks and mitigation methods for RPL-based Internet of Things. *IEEE Commun Surv Tutor.* 2018;21(2):1582–606.
5. Medjek F, Tandjaoui D, Djedjig N, Romdhani I. Fault-tolerant AI-driven intrusion detection system for the internet of things. *Int J Crit Infrastruct Prot.* 2021;34:100436.
6. Labiod Y, Korba AA, Ghoualmi-Zine N. Detecting DDoS attacks in IoT environment. *Int J Inf Secur Priv.* 2021;15(2):145–80.
7. Subramani S, Selvi M. Comprehensive review on distributed denial of service attacks in wireless sensor networks. *Int J Inf Comput Secur.* 2023;20(3–4):414–38.
8. Khan K, Mehmood A, Khan S, Khan MA, Iqbal Z, Mashwani WK. A survey on intrusion detection and prevention in wireless ad-hoc networks. *J Syst Archit.* 2020;105:101701.
9. Baraneetharan E. Role of machine learning algorithms intrusion detection in WSNs: a survey. *J Inf Technol.* 2020;2(03):161–73.
10. Jinisha JJ. Survey on various attacks and intrusion detection mechanisms in wireless sensor networks. *Turkish J Comput Math Educ.* 2021;12(11):3694–704.
11. Raghavendra T, Anand M, Selvi M, Thangaramya K, Kumar SS, Kannan A. An intelligent RPL attack detection using machine learning-based intrusion detection system for Internet of Things. *Procedia Comput Sci.* 2022;215:61–70.
12. Verikas A, Bacauskiene M. Feature selection with neural networks. *Pattern Recogn Lett.* 2002;23(11):1323–35.
13. Kabir MM, Islam MM, Murase K. A new wrapper feature selection approach using neural network. *Neurocomputing.* 2010;73(16–18):3273–83.
14. Heinzelman WR, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocols for wireless microsensor networks. In: Proceeding of 33rd Hawaiian International Conference on Systems Sciences HICSS 2000, Hawaii, USA. 2000. p. 1–10.
15. Heinzelman WR, Chandrakasan A, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans Wirel Commun.* 2002;1(4):660–70.

16. Thangaramya K, Kulothungan K, Logambigai R, Selvi M, Ganapathy S, Kannan A. Energy aware cluster and neuro-fuzzy based routing algorithm for wireless sensor networks in IoT. *Comput Netw*. 2019;151:211–23.
17. Subramani S, Selvi M. Multi-objective PSO based feature selection for intrusion detection in IoT based wireless sensor networks. *Optik*. 2023;273:170419.
18. Al-Hadhrami Y, Hussain FK. DDoS attacks in IoT networks: a comprehensive systematic literature review. *World Wide Web*. 2021;24(3):971–1001.
19. Liang L, Zheng K, Sheng Q, Huang X. A denial of service attack method for an IoT system. In: 2016 8th International Conference on Information Technology in Medicine and Education (ITME). IEEE; December 2016. pp. 360–64.
20. Airehrour D, Gutierrez JA, Ray SK. SecTrust-RPL: a secure trust-aware RPL routing protocol for Internet of Things. *Futur Gener Comput Syst*. 2019;93:860–76.
21. Pu C. Sybil attack in RPL-based internet of things: analysis and defenses. *IEEE Internet Things J*. 2020;7(6):4937–49.
22. Le A, Loo J, Luo Y, Lasebae A. Specification-based IDS for securing RPL from topology attacks. In: 2011 IFIP Wireless Days (WD). IEEE; October 2011. p. 1–3.
23. Kiran V, Rani S, Singh P. Towards a light weight routing security in IoT using non-cooperative game models and Dempster–Shaffer theory. *Wirel Pers Commun*. 2020;110(4):1729–49.
24. Agiollo A, Conti M, Kaliyar P, Lin TN, Pajola L. DETONAR: detection of routing attacks in RPL-based IoT. *IEEE Trans Netw Serv Manag*. 2021;18(2):1178–90.
25. Mayzaud A, Badonnel R, Chrisment I. Detecting version number attacks in RPL-based networks using a distributed monitoring architecture. In: 2016 12th International Conference on Network and Service Management (CNSM). IEEE; October 2016. p. 127–35.
26. Rani J, Dhingra A, Sindhu V. A detailed review of the IoT with detection of sinkhole attacks in RPL based network. In: 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT). IEEE; March 2022. p. 1–6.
27. Muzammal SM, Murugesan RK, Jhanjhi NZ, Jung LT. SMTrust: proposing trust-based secure routing protocol for RPL attacks for IoT applications. In: 2020 International Conference on Computational Intelligence (ICCI). IEEE; October 2020. p. 305–10.
28. Mishra P, Varadarajan V, Tupakula U, Pilli ES. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Commun Surv Tutor*. 2018;21(1): 686–728.
29. Sharma S, Verma VK. AIEMLA: artificial intelligence enabled machine learning approach for routing attacks on internet of things. *J Supercomput*. 2021;77(12):13757–87.
30. Wallgren L, Raza S, Voigt T. Routing attacks and countermeasures in the RPL-based internet of things. *Int J Distrib Sens Netw*. 2013;9(8):794326.
31. Osman M, He J, Mokbal FMM, Zhu N. Artificial neural network model for decreased rank attack detection in RPL based on IoT networks. *Int J Netw Secur*. 2021;23(3):496–503.
32. Subramani S, Selvi M. Intrusion detection system using RBPSO and fuzzy neuro-genetic classification algorithms in wireless sensor networks. *Int J Inf Comput Secur*. 2023;20(3–4): 439–61.
33. Subramani S, Selvi M, Kumar SS, Kannan A. Energy efficient clustering routing protocol and ACO algorithm in WSN. In: Advances in Computing and Data Sciences: 5th International Conference, ICACDS 2021, Nashik, India, April 23–24, 2021, Revised Selected Papers, Part I 5. Springer International Publishing; 2021. p. 68–80.
34. Zhou Y, Cheng G, Jiang S, Dai M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Comput Netw*. 2020;174:107247.

Shallow and Deep Evolutionary Neural Networks Applications in Solid Mechanics



Anna Malá, Zdeněk Padovc, Tomáš Mareš, and Nirupam Chakraborti

Abstract Data-driven models are now successfully used to solve complex multi-objective optimisation problems in materials mechanics. This chapter deals with the creation of surrogate models and its subsequent application for optimisation by evolution-based algorithms. Models representing curved beams or frames fabricated from composite tubes with circular cross sections are discussed here. Some learning strategies based on Evolutionary Neural Network (EvoNN), Bi-objective Genetic Programming (BioGP), and Evolutionary Deep Neural Net (EvoDN2) algorithms were applied. The surrogate models obtained from these algorithms were subsequently subjected to multi-optimisation for computing the Pareto fronts as the resulting output. Simple geometries could be very efficiently trained and optimised by each of the above-mentioned approaches, but more complex configurations (e.g., curved beams and frames made up of more tubes or tubes with additional layers) were effectively solved only by the EvoDN2 algorithm, as demonstrated in examples.

Keywords Composite material · Tube · Curved beam · Frame · Data-driven modelling · Evolutionary computation · Reference vector algorithm · Multi-objective optimisation · Deep neural nets · BioGP · EvoNN · EvoDN2

1 Introduction

In the process of designing machines, production lines, means of transport, sports equipment, etc., some specific requirements usually influence it. For example, such requirements could be associated with the reduction of emissions or the use of lighter materials, among other things. Furthermore, such requirements may also relate to the

A. Malá (✉) · Z. Padovc · T. Mareš · N. Chakraborti

Department of Mechanics, Biomechanics and Mechatronics, Faculty of Mechanical Engineering, Czech Technical University in Prague, Prague, Czech Republic
e-mail: anna.mala@fs.cvut.cz; zdenek.padovc@fs.cvut.cz; tomas.mares@fs.cvut.cz; nirupam.chakraborti@fs.cvut.cz

limitation of heat transfer to components that serve as a thermal barrier and at the same time as a clamping device, whose natural frequency should be in a given range.

In many applications, laminated composite materials are often used because of their lightweight, stiffness, thermal conduction, or convenient manufacturing process. The parts built from composite materials are typical for airplanes, space stations, sports equipment, and cars, for example. Experience gained from previous development is often used in the design of these parts. At the same time, new methods are evolving with increasing capabilities of computers. The 3D models prepared for the parts to be manufactured can now be changed using advanced optimisation software that is capable of preparing a design with sufficient properties for a given application. Optimisation methods often work well for isotropic materials. Composite materials, such as laminates, whose properties are not isotropic but orthotropic, may also be subjected to the process of selecting the most suitable composition; the orthotropy of these materials renders the process more complex to be performed, but still, a vast number of possible solutions can be generated.

This chapter deals with the determination of the optimum design variables for fibre-wound composite beams to build a given structure while the required properties are satisfied. Evolutionary algorithms are used to determine the most suitable design variables. These design variables encompass various parameters, such as winding characteristics (e.g., thickness and angle of tube plies), material selection, and geometry specification (e.g., tube radius and cross-section). This chapter presents the usage of neural network approaches in metamodeling, which is favourable in this type of assignment due to its capability to solve complex tasks. In the scenarios we explore here, the geometrical and material parameters are predefined, and our focus is on determining the optimal winding parameters. We delve into both shallow and deep evolutionary neural network approaches, demonstrating their efficacy in solving complex problems related to composite laminate tube stacking sequences. It is evident that as tasks grow in complexity, a more sophisticated toolset becomes essential to derive optimal solutions.

2 Optimal Design Using Material Mechanics

The use of suitable materials for the given application is one of the major requirements to be met. Since antiquity, when the first tools and machines were made, people thought about the components of the materials used and their properties. When high strength was needed, sufficiently hard materials were used. Conversely, where a certain degree of flexibility was needed, softer and more compliant materials were used. Similarly, other mechanical properties such as thermal conductivity, strength, or elasticity were gradually taken into account with the advancement of manufacturing technology.

One of the most important laws of material mechanics is Hooke's law (proposed in 1678), which connects the extension of the material with applied force [1]. Other important scientists of the seventeenth to nineteenth century, such as Jacob

Bernoulli, Leonard Euler, Thomas Young, Gottfried Wilhelm Leibniz, Charles-Augustin Coulomb, and Augustin-Louis Cauchy, extended Hooke's law. Subsequently, the theories of beams, plates, and shells were developed, which were based on stress, strain, and elasticity laws previously detected by the scientists [2]. Material properties such as thermal conductivity, roughness, porosity, etc. are also decisive. Thus, a comprehensive theory of material mechanics is very important for structure design to satisfy the needs of consumers. We will now explore how these are pertinent for the laminate structures.

2.1 *Laminates Usage*

Using a different material for the same application while maintaining the same product properties requirements can substantially change the shape of the part. This often happens when using laminates and other composite materials [3]. Their properties, as well as those of the entire structure, are affected by composition and these materials are often used due to the possibility of achieving good design parameters with a lower weight [4–6]. A convenient composition is obtained by experience or, more effectively, by using optimisation methods that try to find suitable parameters for any given purpose.

The material properties of the laminates strongly depend on the stacking sequence (thickness and angle of fibre orientation of each layer). They should be used with respect to their stacking sequence and orientation [7–9].

The use of the 0-degree orientation is required in applications where bending and tensions are the main applied loads, and the 45-degree orientation in applications lead to a situation where the torsional load dominates. Usually, parts used are loaded with combinations of loads mentioned above; therefore, the stacking sequence should be in the best setup for the given load. When the thickness of the laminate increases, the stiffness, strength, or thermal resistance usually increases as well [7–9].

2.2 *Decision Variable in Laminates Optimisation*

The decision variable most commonly used in laminate optimisation is the stacking sequence, as stated in [10]. The effect of other parameters such as boundary conditions, temperature, structure geometry, etc. is noticed together with the design variables. The most common objective functions for a multi-objective optimisation study are the stiffness and the structure fundamental frequency for maximisation along with weight minimisation.

The optimisation of composite tubes is done in [11], where a buckling load is applied and the best element thickness and fibre angle is found with the use of the finite element model, which is now ubiquitous in manufacturing technology.

Optimisation of the topology of tubes loaded with buckling and bending is carried out in [12], where the objective function, total compliance, is minimised with the fulfilment of the buckling factor requirements.

2.3 *Approaches to Fibre Composites Optimisation*

The first approaches to optimise fibre composites were built on gradient-based methods because fibre angles were considered as a continuous design variable. It led to finding more local minima [13]. Therefore, metaheuristic algorithms such as the genetic algorithm were developed to design laminates due to their effectiveness and stability [14, 15]. The computational cost of these algorithms increases with the complexity of the structures, and therefore the codes need to be constantly improved [10]. In another study, the genetic algorithm (GA) sought the most suitable ply angles and thicknesses with a two-level approximation considering global structural responses such as critical buckling factors and frequencies [16] and was further extended using strength constraints and adaptive approaches in GA [17].

An example of algorithm development is given in [13], where changes in harmonic memory modify the Harmony Search Algorithm inspired by improvisation in music playing with performance improvements.

2.4 *Optimisation of Structures from Laminated Materials*

The method of designing composite frame structures was carried out in [18]. The frame structures consisted of laminated tubes that were designed subject to manufacturing constraints. The inner radius and stacking sequence of the tube layers were determined for the maximum fundamental frequency optimisation performed using the sequential linear programming (SLP) method.

Similarly, concurrent multiscale design optimisation of composite frames can also be done considering the weight and the stiffness of the construction under the given load. Topology optimisation and material selection can be performed using a discrete material optimisation approach at the same time [19].

Data-Driven Evolutionary Algorithms were used for the design of the pressure vessel dome shape. These were manufactured from glass/epoxy and carbon/epoxy filament wounds. Classical mechanical approaches in combination with evolutionary optimisation algorithms led to optimal design considering the required parameters [20]. Similar to the use of the evolutionary deep neural net algorithm, the design of additively manufactured truss topology components with respect to structural rigidity and heat transfer in [21].

3 Data-Driven Modelling Approaches

Many problems in mechanical engineering, materials engineering, thermodynamics, mechanics, etc., can be described by analytical or numerical equations, which are very complicated for some problems. Also, proper description of a particular problem by these equations is often not possible due to the complexity, boundary conditions, and so on. The computational cost using the analytical model increases enormously with the complexity of the problem, adding to the further difficulties [22].

Such problems can be very easily handled by creating a surrogate model from the system data set—this is the data-driven modelling approach, which is increasingly becoming popular [22]. Physical aspects are not directly used in these models but are still implicitly present. A data-driven model can be used, for example, as a meta-model, consisting of a smaller number of parameters with sufficient accuracy than, for example, a very rigorous physical model. Computation costs can be significantly saved in these cases. The use of a data-driven model is advantageous in situations where the relationships between the parameters of the problem are very complex or unknown. The method then ensures a sufficient description and enables subsequent work on the problem in hand [22].

A prepared data-driven model is subsequently used, for example, to find the best combination of the problem parameters—some of them are decision variables and the others can be a number of objective functions, which require to be minimised or maximised simultaneously to ensure the optimal design. These optimal combinations are the so-called Pareto optimal front, a set of optimal states from which it is most appropriate to choose the most advantageous one for the concerned user. To obtain the Pareto optimal front, both evolutionary and classical optimisation methods are used [23, 24], which will be elaborated in the following sections.

3.1 Data-Driven Evolutionary Algorithm

Multi-objective problems are often present in the industry, and their solving demands sophisticated methods such as Genetic Algorithms based on natural principles. This approach can solve many problems with sufficient accuracy and acceptable computational cost [25]. Complicated solutions to other multi-objective problems led to the development of data-driven strategies using neural networks, genetic programming, etc. The Artificial Neural Network (ANN) was connected to the Multi-objective Evolutionary Algorithm as Evolutionary Neural Network paradigm (EvoNN) [23] and the use of a binary tree architecture leads to the Bi-Objective Genetic Programming (BioGP) algorithm [22].

In short, the evolutionary algorithm works by creating a first random generation and, in subsequent generations, the offspring are generated by crossover, mutation, or another evolutionary principle. The fitness function and the feasibility of each

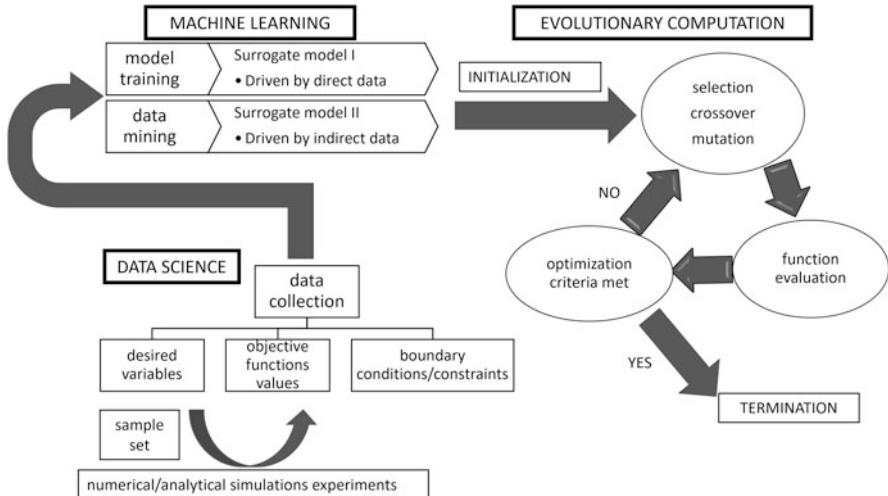


Fig. 1 Main components of data-driven evolutionary optimisation

member are evaluated to select members for the next generation. Members are selected from offspring only or from both offspring and parents [26]. The typical workflow of data-driven evolutionary optimisation following [26] is schematically shown in Fig. 1. First Evolutionary Computation represents population-based metaheuristic method such as genetic algorithm (GA), particle swarm optimisation (PSO), ant colony optimisation (ACO), shuffled frog leaping algorithm (SFLA) etc. [27]. The Second section Machine Learning represents techniques, and third section is Data Science.

3.2 Algorithms for Evolutionary Metamodelling

The increasing data volume along with the associated hardware requirements often bring some difficult challenges. Despite hardware performance still increasing, as well as software performance, traditional methods are often not usable. In this case, the use of ANN is advantageous. For effective processing of large data, large neural networks should be used. The problem of data processing can be the low quality or dissimilar source of the data, their structure, classification, veracity, etc. Big data analytics is the process of finding rules or patterns of behaviour [28]. In this chapter, we will focus on the EvoNN algorithm and its advanced versions. The EvoNN algorithm has been used successfully in several studies [29, 30]. In EvoNN only one hidden layer is used due to the overfitting avoidance. The multi-objective evolutionary algorithm is used only for the lower part for training, and the upper part uses a

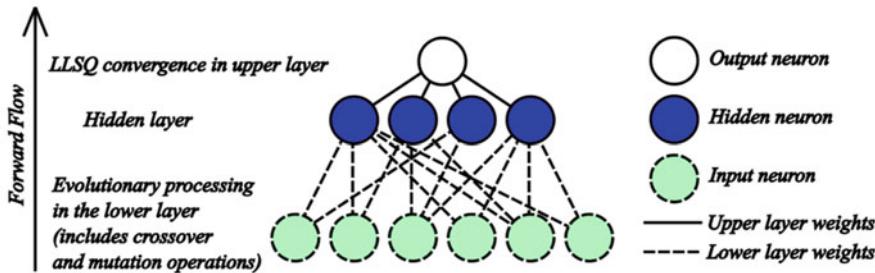


Fig. 2 A typical Individual in EvoNN. It needs to be emphasised that in the lower layer, any input node may or may not pass on information to every neuron in the hidden layer

linear least squares (LLSQ) algorithm, see Fig. 2, which provides a schematic representation of this architecture [22]. For effective data analysis enhancement, the subsequent evolutionary deep neural net algorithms EvoDN and EvoDN2 algorithms were developed [23, 31, 32].

The population of EvoNN can be written as a 3D matrix, which is built by the connection of 2D matrices (which have the dimension: number of input nodes x number of output nodes plus one extra row for biases). The EvoDN [33] network configuration is simply the stacking of more hidden layers and their connections. Due to the diverse numbers of input and output nodes in each hidden layer, the 2D matrix dimension of each population is not similar. The matrices are stacked together for each layer in the 3D matrices. These create the whole population by joining themselves in the cell structure. Evolutionary training and LLSQ optimisation are done in a similar way as EvoNN.

The idea here is to create an intelligent model, which will neither overfit nor underfit the training data [22]. Adding more parameters, which determines the complexity of the model, increases the accuracy with a possibility of overfitting. On the other hand, using just a limited number of parameters may lead to underfitting of the data. Therefore, what is needed is an optimum tradeoff between these two tendencies. To achieve this, the algorithms used here attempt to create an optimum tradeoff between the complexity and accuracy of the models by treating it as a bi-objective optimization problem. A predator-prey genetic algorithm [34, 35] and also a reference vector evolutionary algorithm [36] are used for this purpose which are briefly described in the text further.

In EvoNN complexity is defined by weights, the number of non-zero weights in the network's lower part. It works well for the EvoNN, where one hidden layer is employed, but it is not usable for more hidden layers. Some connections are more important than the others; therefore, in EvoDN the simple counting of the weights is not sufficient, and the magnitudes of them are subsequently processed. Each pathway is evaluated, and the nonactive connection inside cancels the whole path, while the connection with higher weight participates more intensively in the complexity measurement parameter.

Accuracy is expressed as the training error value. This is solved identically to EvoNN by the relative root mean square error (RMSE) by applying the logistic regression cost function [37]. Due to the convexity function, it has only one extremum, which contributes to its dependability.

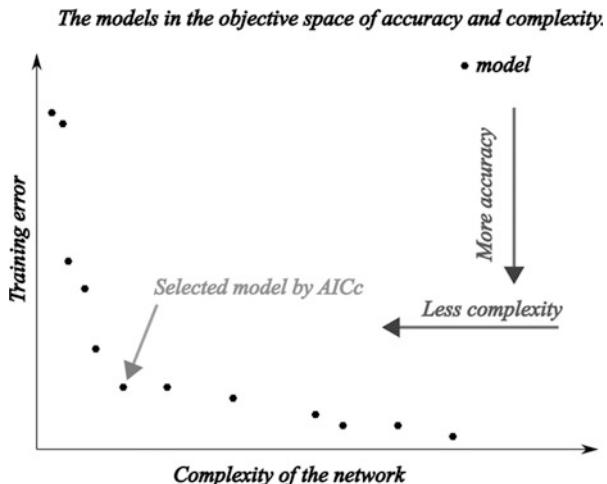
Mutation and crossover strategies were redefined in the structure of deep neural networks due to the decreased effectiveness through the count of the layers and the practicability of such complex structures, which are solved through the algorithm. The mathematical definition was modified for the self-adaption factor. In the EvoNN algorithm, the factor is defined through the weights of the randomly selected population members through a mutation process, but in the EvoDN algorithm, the factor is defined using the standard deviation in the population. The calculation is performed only once, unlike the previous definition. The mutation depends on the variation in the population, and the standard deviation decreases with convergence. Due to the usage of standard deviation, the mutation is self-adaptive.

EvoNN uses the corrected Akaike information criteria (AICc) [38] to find the usable optimum solution from the Pareto set provided by the predator-prey genetic algorithm, as shown in Fig. 3. In case of multi-objective optimization, which is used for building the models in this situation, the solution is not unique, rather it is a set of alternate optimum solutions, out of which a suitable solution is identified using the AICc criterion. The sample size is not the issue here. For algorithms such as EvoDN, the choice of the most accurate model is usually beneficial.

The EvoDN algorithm initially had limited success. Subsequently, it was upgraded to its enhanced version EvoDN2 [39] which was used with great success in several studies [31].

EvoNN employs a population of shallow neural networks, where the number of hidden layers is restricted to 1, to avoid data overfitting. Such an arrangement works fine until and unless the amount of data being processed is not very large. In case of a large volume of data, particularly the big data, the processing of which is now quite

Fig. 3 Identification of an optimum model using the AICc criterion



common in many real-life situations, such an arrangement is not very useful, as has also been found in this study. To avoid this problem, some deep learning strategies [40] must be employed so that significant features in a large volume of noisy data can be efficiently determined. In order to achieve this, the EvoDN2 algorithm was used in the present study.

In EvoDN2, which is an upgraded version of EvoDN, the following features are similar to EvoNN:

- scheme of the mutation and crossover
- use of predator-prey genetic algorithm or reference vector evolutionary algorithm
- use of LLSQ algorithm

The algorithm upgrade represents the following changes:

- Multiple smaller subnets are combined into an evolutionary deep neural net structure.
- Each variable is used at least in one of the subnets.
- Both PPGA and reference vector evolutionary algorithms (RVEA) are possible to use in the optimisation runs here for two objective functions. More than two objective functions are possible to optimise with the RVEA algorithm alone.

The flow chart and architecture of EvoDN2 are shown in Fig. 4. As depicted there, the decision variables are distributed over a number of subnets of varying depth, which evolve in an evolutionary fashion by interacting with the other members of the population having some similar architecture. The final convergence is obtained through a deterministic LLSQ algorithm. Further details are available elsewhere [22].

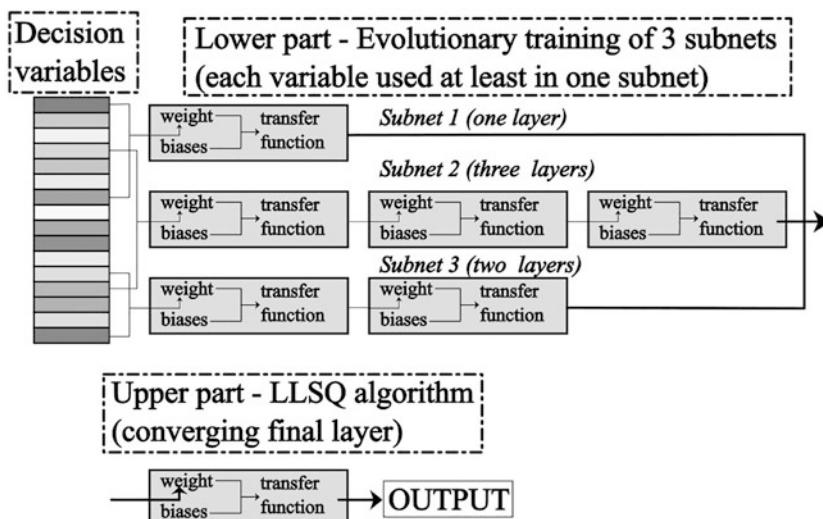


Fig. 4 EvoDN2 architecture

3.2.1 Predator-Prey Genetic Algorithm

The predator-prey genetic algorithm (PPGA), described in [34, 41], has been used successfully in numerous tasks, in [29, 42] for example. As has been mentioned before, the PPGA is established for bi-objective problems. Traditional evolutionary techniques are used for the random generation of prey (candidate solutions) in the space of decision variables. Predators are inserted from the outside to reduce the prey population based on a fitness function. The fitness function depends on the objectives F_1 and F_2 . Each predator has attributed the weight value between zero and one ω_j , which distributes the objective functions to the prey fitness value (fitness = $\omega_j \cdot F_1 + (1 - \omega_j) \cdot F_2$). This defines the vulnerability of the prey for each predator separately due to the unique weight value of each predator.

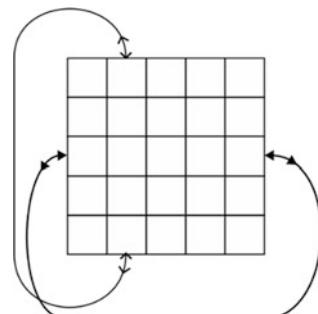
The algorithm uses a 2D lattice and a Moore neighbourhood. Predators and preys have their own Moore neighbourhood. The lattice does not copy the Cartesian system, but the first and last rows and columns, as well as diagonally opposite corners, are connected, which creates the computational space shown schematically in Fig. 5. Predators move one step at a time and hunt one by one. A maximum number of steps is calculated for each generation. If the predator completes all allowed steps, the most vulnerable individual of the prey population in the Moore neighbourhood is eliminated and its position is taken over by the predator. If no prey is located there, the predator moves on and continues hunting until the allowed number of steps is reached. Hunting does not take place if the predator moves to a position occupied by another inactive predator.

The number of moves allowed is dynamically adjusted based upon the defined target number of the prey and its current number at the beginning of a generation. The number of preys in the population is often much larger than the number of target preys. This prevents excessive elimination or uncontrolled population growth.

The prey population can move after the end of predator movement and killing. Preys move one by one, one step at a time, to the positions where nobody is present. The new positions allow for additional connections for crossover and mutation. These are performed randomly between individuals in their neighbourhoods.

Some of the available crossover procedures can be used. After the crossover and mutation, the children are placed in the unoccupied position in the lattice. It is

Fig. 5 Typical neighboured grids at the opposite edges are shown with arrows



defined a priori how many times such a placement can be attempted, and if the placement fails within the prescribed number of attempts, then the offspring is deleted. Thus, the gene pool is well mixed and distributed due to migration. Children can participate in crossbreeding in the next generation. Predators do not cross or evolve, and their numbers remain constant throughout the process.

With the new offspring of prey, the next generation begins, and genetic processing continues. The prey population is sorted using the Fonseca scheme [43] and a certain part is preserved up to the given rank and the rest is removed. It was introduced in this algorithm by [41] and led to an increase in the efficiency of the algorithm for complicated problems. Once the algorithm stabilises, sorting is done, and the non-dominating set at position 1 is accepted as the final solution.

3.2.2 Reference Vector Evolutionary Algorithm (RVEA)

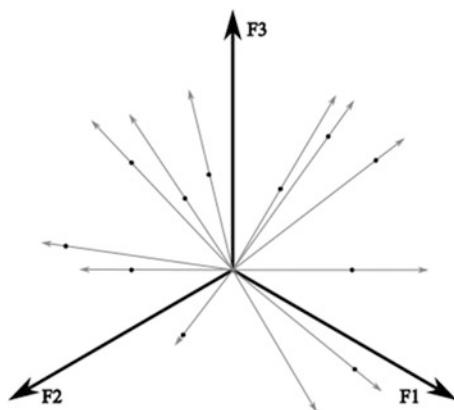
The reference vector evolutionary algorithm is described in [44].

This algorithm uses unit vectors with the position at the origin and the terminal point in the first quadrant. These are generated by the method described in [45]. N points are distributed in a unit hyperplane using the canonical simplex lattice design method [46]. Their notation is for M objective functions:

$$\mathbf{u}_i = (u_i^1, u_i^2, \dots, u_i^M); u_i^j \in \left[\frac{0}{H}, \frac{1}{H}, \dots, \frac{H}{H} \right]; \sum_{j=1}^M u_i^j = 1 \quad (1)$$

Where H is a positive integer for the simplex-lattice design. The reference vectors \mathbf{v}_i are obtained by mapping the reference points to a hyperspace (see Fig. 6). Mathematically formula (division \mathbf{u}_i by its norm/magnitude/length) is:

Fig. 6 Reference vectors generated by mapping uniformly distributed reference points to a hypersphere (three-objective space)



$$\mathbf{v}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|} \quad (2)$$

The total number of such vectors is based on the theory of the canonical simplex-lattice strategy. Each member of the population, in other words, an objective vector is individually assigned to the reference vector closest to it with respect to the angular distance between them. In this way, subpopulations are created, and the elitism strategy is adopted in a combination of offspring and the parent population. The offspring are obtained by the mutation and crossover processes. The constrained version of the reference vector algorithm (cRVEA) is used for the optimisation run described in this chapter because the usage of constraints is advantageous in cases where only positive values of some objective functions are required, for example.

3.3 Data-Driven Model in Material Engineering

In materials engineering, real encoded data is assumed and successfully used for data-driven model creation [22]. A learning strategy based on an existing evolutionary neural network (EvoNN) and a bi-objective genetic programming algorithm (BioGP) was introduced in various material engineering problems with good results in alloy composition, blast furnace or interatomic potential parameterisation [23]. Microscale material properties, such as prediction grain boundary chemical composition of W-25 at.% Re alloy or investigation of uncertainty in W-Ti alloy phase diagram was researched by data-driven material modelling in [47]. The development of alloys, as well as the development of fibre composites using data-driven models, is ongoing [48]. The load vs. displacement curve of cotton fibre-reinforced polymers is predicted by data-driven artificial neural network (ANN) models [49]. The properties of fibre-reinforced composite materials depend on the curing cycles of the manufacturer. This dependence is well described by the data-driven model, which provides an easier option for designing laminates with a decreasing number of experiments [50].

4 Optimisation of Mechanical Properties of Composite Beam Structures

Fibre-wound composite tubes and frames are widely used in applications where the advantages of laminates (favourable strength-to-weight and stiffness-to-weight ratio, good resistance to the surrounding environment, durability, design flexibility, etc.) overbalance their disadvantages (cost, brittleness, possibility of repairing and inspecting composite parts, dimensional tolerance, etc.). Some of the advantages

of wounded tubes and frames are the possibility of designing each ply to the most advantageous lay-up due to the orthotropic properties [51].

Laminates lead to more complex structures that are present in more lay-ups/plies and the determination of the mechanical properties of the whole structure is difficult [7–9]. The process of finding the most suitable composition of each part of the structure is then an interesting task, where the performance of traditional optimisation methods is often insufficient [22]. Our object of interest will be curved beams and frames composed of composite tubes, the material parameters of which depend on the stacking sequence of each tube. We will use BioGP, EvoNN, and EvoDN2 training algorithms to construct models from the data sets.

4.1 Problem Description

Here, the object of optimisation is a three-dimensional frame structure or a curved beam composed of beam elements. They are made from laminated composite tubes of circular cross sections manufactured by filament winding. External forces act at the nodes of the frame structure. To find out the internal forces we express local and global stiffness matrices. We focus on the trade-off/balance of three objective functions:

- ***objective function F1***: the compliance of the structure expressed through the total potential energy.
- ***objective function F2***: the strength coefficient, which is a function of stresses. (Failure index)
- ***objective function F3***: the volume of the structure.

Each objective function is minimised.

The determination of the objective function was done using Python code. A beam model was created using the finite element method tools [52, 53], where a 2-node Timoshenko beam element was defined in a 3D space. The stiffness matrix of those elements \mathbf{K}_e :

$$K_e = \begin{bmatrix} k_{11}^x & 0 & 0 & 0 & 0 & 0 & -k_{11}^x & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{11}^z & 0 & 0 & 0 & k_{12}^z & 0 & -k_{11}^z & 0 & 0 & 0 & k_{12}^z \\ 0 & 0 & k_{11}^y & 0 & -k_{12}^y & 0 & 0 & 0 & -k_{11}^y & 0 & -k_{12}^y & 0 \\ 0 & 0 & 0 & k_{11}^p & 0 & 0 & 0 & 0 & 0 & -k_{11}^p & 0 & 0 \\ 0 & 0 & -k_{12}^y & 0 & k_{22}^y & 0 & 0 & 0 & k_{12}^y & 0 & k_{21}^y & 0 \\ 0 & k_{12}^z & 0 & 0 & 0 & k_{22}^z & 0 & -k_{12}^z & 0 & 0 & 0 & k_{21}^z \\ -k_{11}^x & 0 & 0 & 0 & 0 & 0 & k_{11}^x & 0 & 0 & 0 & 0 & 0 \\ 0 & -k_{11}^z & 0 & 0 & 0 & -k_{12}^z & 0 & k_{11}^z & 0 & 0 & 0 & -k_{12}^z \\ 0 & 0 & -k_{11}^y & 0 & k_{12}^y & 0 & 0 & 0 & k_{11}^y & 0 & k_{12}^y & 0 \\ 0 & 0 & 0 & -k_{11}^p & 0 & 0 & 0 & 0 & 0 & k_{11}^p & 0 & 0 \\ 0 & 0 & -k_{12}^y & 0 & k_{21}^y & 0 & 0 & 0 & k_{12}^y & 0 & k_{22}^y & 0 \\ 0 & k_{12}^z & 0 & 0 & 0 & k_{21}^z & 0 & -k_{12}^z & 0 & 0 & 0 & k_{22}^z \end{bmatrix} \quad (3)$$

$$k_{11}^x = \frac{(EA)_{eq}}{L}; k_{11}^p = \frac{(GJ_p)_{eq}}{L}$$

$$\begin{bmatrix} k_{11}^z & k_{12}^z \\ k_{21}^z & k_{22}^z \end{bmatrix} = \frac{(EJ_z)_{eq}}{(1 + \Phi_z)L^3} \begin{bmatrix} 12 & 6L \\ (2 - \Phi_z)L^2 & (4 + \Phi_z)L^2 \end{bmatrix}; \Phi_z = \frac{12(EJ_z)_{eq}}{L^2(\kappa GA)_{eq}}$$

$$\begin{bmatrix} k_{11}^y & k_{12}^y \\ k_{21}^y & k_{22}^y \end{bmatrix} = \frac{(EJ_y)_{eq}}{(1 + \Phi_y)L^3} \begin{bmatrix} 12 & 6L \\ (2 - \Phi_y)L^2 & (4 + \Phi_y)L^2 \end{bmatrix}; \Phi_y = \frac{12(EJ_y)_{eq}}{L^2(\kappa GA)_{eq}}$$

depends on the length of the beam \mathbf{L} , the equivalent tensile stiffness of the element $(\mathbf{EA})_{eq}$, the equivalent bending stiffness in the z -direction of the element $(\mathbf{EJ}_z)_{eq}$, the equivalent bending stiffness in the y -direction of the element $(\mathbf{EJ}_y)_{eq}$, the equivalent torsional stiffness of the element $(\mathbf{GJ}_p)_{eq}$, and the equivalent shear stiffness of the element $(\kappa \mathbf{GA})_{eq}$. Due to the circular cross section, the bending stiffnesses $(\mathbf{EJ}_z)_{eq}$ and $(\mathbf{EJ}_y)_{eq}$ are equal.

Each equivalent stiffness is calculated by the theory of laminates [7–9]. Specifically, using a minimum complementary energy theory:

$$(EA)_{eq} = \sum_{i=1}^n \frac{\pi(r_{outi}^2 - r_{ini}^2)}{C_{ti}(1, 1)} \quad (4)$$

Table 1 Material properties of carbon epoxy t700/E

Material property	Symbol	Value	Unit
Longitudinal Young's modulus	E_1	128,000	MPa
Transverse Young's modulus	E_2	5060	MPa
Shear modulus	G_{12}	3400	MPa
Poisson ratio	ν_{12}	0.345	—
Longitudinal tensile strength	X_T	2500	MPa
Longitudinal compressive strength	X_C	900	MPa
Transverse tensile strength	Y_T	50	MPa
Transverse compressive strength	Y_C	230	MPa
Shear strength	S	87	MPa

$$(EJ_z)_{eq} = (EJ_y)_{eq} = \sum_{i=1}^n \frac{\pi r_i^3 t_i}{C_{ti}(1,1)} \quad (5)$$

$$(GJ_p)_{eq} = \sum_{i=1}^n \frac{4A_{ci}^2 t_i}{c_i C_{ti}(3,3)}; A_{ci} = \pi r_i^2, c_i = 2\pi r_i \quad (6)$$

$$(\kappa GA)_{eq} = \sum_{i=1}^n \frac{\pi (r_{outi}^2 - r_{ini}^2)}{C_{ti}(3,3)} \quad (7)$$

where \mathbf{r}_{out} , \mathbf{r} , and \mathbf{r}_{in} is the ply outer, mean, and inner radius; \mathbf{t} is the ply thickness; $C_t(1,1)$, resp. $(3,3)$ is $(1,1)$, resp. $(3,3)$ member of the compliance matrix of the lamina in the laminate coordinate system; c_i is the ply circumference of the midline and A_{ci} is the area within c_i .

Equivalent stiffnesses depend on the properties of the material used, the thickness and orientation of the ply, the stacking sequence and the cross section of the tube. The ply thickness and orientation of each layer were chosen as decision variables. Each tube is made up of the same material T700/epoxy laminate, the material parameters of which are listed in Table 1. Tubes are all circular with a given inner radius of 11 mm.

The nodes define the geometry. Nodes displacements are given by acting external forces \mathbf{F}_{ext} and the global stiffness matrix \mathbf{K}_{glob} . The displacement vector \mathbf{u}_{glob} in combination with the stiffness matrix gives the objective function:

$$F1 = f(F_{ext}; \mathbf{u}_{glob}) \quad (8)$$

which expresses compliance of the structure and should be minimised to achieve the stiffest design.

The external forces are recalculated to the internal forces \mathbf{F}_{int} with the use of global and local \mathbf{K}_{lok} stiffness matrices. The estimated internal forces (tension/compression force \mathbf{N} , torque \mathbf{T} and bending moment \mathbf{M}) in combination with

transformation matrix $\mathbf{T}_{\sigma_{ix}}$ are used to get ply stresses σ_{Li} , σ_{Ti} , and σ_{LTi} in material coordinates:

$$\begin{bmatrix} \sigma_{Li} \\ \sigma_{Ti} \\ \sigma_{LTi} \end{bmatrix} = T_{\sigma_{ix}} \begin{bmatrix} \sigma_{xi} \\ 0 \\ \tau_{xy_i} \end{bmatrix} \quad (9)$$

$$\sigma_{xi} = f(N, M)$$

$$\tau_{xy_i} = f(T)$$

which are used to gain strength coefficients of ply i according to the maximum stress criterion $smax_i$, the Tsai-Hill criterion TH_i , the Hoffmann criterion H_i , and the Tsai-Wu criterion TW_i [7–9]:

$$\begin{aligned} smax_{Li} &= -\frac{\sigma_{Li}}{X_C} \text{ for } \sigma_{Li} < 0; smax_{Li} = \frac{\sigma_{Li}}{X_T} \text{ for } \sigma_{Li} > 0 \\ smax_{Ti} &= -\frac{\sigma_{Ti}}{Y_C} \text{ for } \sigma_{Ti} < 0; smax_{Ti} = \frac{\sigma_{Ti}}{Y_T} \text{ for } \sigma_{Ti} > 0 \\ smax_{LTi} &= -\frac{\sigma_{LTi}}{S} \text{ for } \sigma_{LTi} < 0; smax_{LTi} = \frac{\sigma_{LTi}}{S} \text{ for } \sigma_{LTi} > 0 \\ smax_i &= \max(smax_{Li}, smax_{Ti}, smax_{LTi}) \end{aligned} \quad (10)$$

$$\begin{aligned} TH_i &= \left(\frac{\sigma_{Li}}{X_C}\right)^2 + \left(\frac{\sigma_{Ti}}{Y_C}\right)^2 + \left(\frac{\sigma_{LTi}}{S}\right)^2 - \frac{\sigma_{Li}\sigma_{Ti}}{X_C} \text{ for } \sigma_{Li} \leq 0 \text{ and } \sigma_{Ti} < 0 \\ TH_i &= \left(\frac{\sigma_{Li}}{X_C}\right)^2 + \left(\frac{\sigma_{Ti}}{Y_T}\right)^2 + \left(\frac{\sigma_{LTi}}{S}\right)^2 - \frac{\sigma_{Li}\sigma_{Ti}}{X_C} \text{ for } \sigma_{Li} \leq 0 \text{ and } \sigma_{Ti} \geq 0 \\ TH_i &= \left(\frac{\sigma_{Li}}{X_T}\right)^2 + \left(\frac{\sigma_{Ti}}{Y_C}\right)^2 + \left(\frac{\sigma_{LTi}}{S}\right)^2 - \frac{\sigma_{Li}\sigma_{Ti}}{X_T} \text{ for } \sigma_{Li} > 0 \text{ and } \sigma_{Ti} < 0 \\ TH_i &= \left(\frac{\sigma_{Li}}{X_T}\right)^2 + \left(\frac{\sigma_{Ti}}{Y_T}\right)^2 + \left(\frac{\sigma_{LTi}}{S}\right)^2 - \frac{\sigma_{Li}\sigma_{Ti}}{X_T} \text{ for } \sigma_{Li} > 0 \text{ and } \sigma_{Ti} \geq 0 \end{aligned} \quad (11)$$

$$H_i = \frac{\sigma_{Li}^2}{X_T X_C} + \frac{\sigma_{Ti}^2}{Y_T Y_C} + \frac{\sigma_{Li}\sigma_{Ti}}{X_T X_C} + \sigma_{Li} \frac{(X_C - X_T)}{X_C X_T} + \sigma_{Ti} \frac{(Y_C - Y_T)}{Y_C Y_T} + \left(\frac{\sigma_{LTi}}{S}\right)^2 \quad (12)$$

$$\begin{aligned} TW_i &= \frac{\sigma_{Li}^2}{X_T X_C} + \frac{\sigma_{Ti}^2}{Y_T Y_C} + \frac{\sigma_{Li}\sigma_{Ti}}{\sqrt{X_T X_C Y_T Y_C}} + \sigma_{Li} \frac{(X_C - X_T)}{X_C X_T} + \sigma_{Ti} \frac{(Y_C - Y_T)}{Y_C Y_T} \\ &\quad + \left(\frac{\sigma_{LTi}}{S}\right)^2 \end{aligned} \quad (13)$$

The highest one for each theory is chosen as the objective function F2.

$$\begin{aligned}
 F2_{smax} &= \max(smax_i) \\
 F2_{TH} &= \max(TH_i) \\
 F2_H &= \max(H_i) \\
 F2_{TW} &= \max(TW_i)
 \end{aligned} \tag{14}$$

These coefficients are the objective function F2. Its minimisation leads to a higher strength of the structure.

The objective function F3 is defined as the volume of the structure.

$$F3 = f(\text{geometry; tubes thicknesses}) \tag{15}$$

Its value depends on the thicknesses of the tubes and its minimisation leads to achieving a lighter structure.

4.2 Utilisation of Different Optimisation Approaches in the Simple Composite Beam Structures

Firstly, the simple geometry of the cantilever with a full moment connection—geometry 1—and simple curved beam—geometry 2, both in Fig. 7, were trained by algorithms BioGP, EvoNN and EvoDN2 to determine their effectiveness for a given task. BioGP and EvoNN trained two of the three objective functions together because these algorithms are not usable for more than two objective functions. The effectiveness of both algorithms for geometry 1 was excellent, and the optimisation provided us with Pareto front sets as a result. The effectiveness of both algorithms in geometry 2 was also excellent for the 1-ply tubes. The optimised meta-modelling was, however, successful for F1 and F3 only. This is the reason why the second version of the evolutionary deep neural net algorithm EvoDN2 was used to solve more complex geometries.

The EvoDN2 training procedure was conducted on both geometries. All three objective functions were trained and subsequently the decision variables were

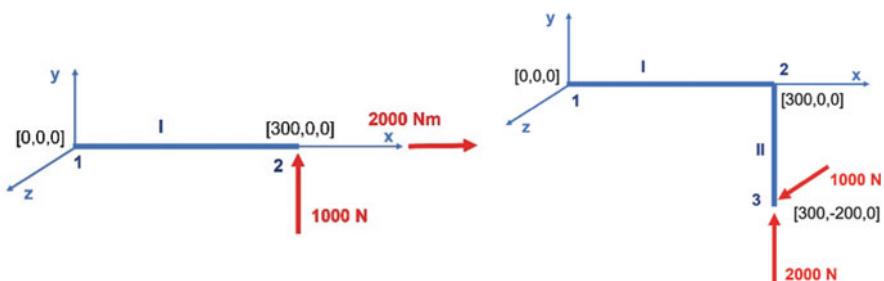
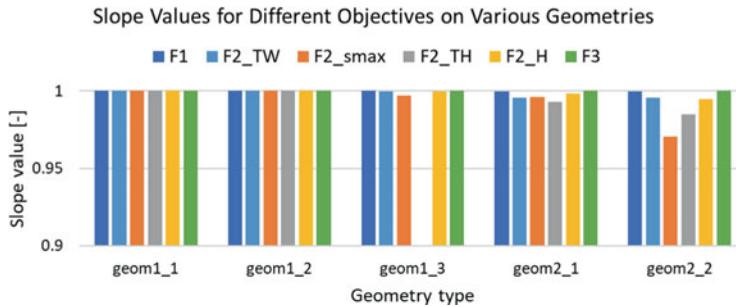


Fig. 7 Geometry 1 (left) and geometry 2 (right)

Table 2 Geometry 1 and 2 parameters

	Ply thickness [mm]	Tube thickness [mm]	Ply count [-]	Winding angle [DEG]	Combinations
geom1_1	0, 1.5 or 3	$1.5 \leq T_t \leq 4.5$	1	15-45-75	9
geom1_2	0, 1.5 or 3	$1.5 \leq T_t \leq 4.5$	1	0-15-30...-90	21
geom1_3	0, 1.5 or 3	$1.5 \leq T_t \leq 4.5$	2	0-15-30...-90	441
geom2_1	0, 1.5 or 3	$1.5 \leq T_t \leq 4.5$	1	15-45-75	81
geom2_2	0, 1.5 or 3	$1.5 \leq T_t \leq 4.5$	1	0-15-30...-90	441

**Fig. 8** Geometry 1 and 2—slope values for different objectives on various geometries

optimised. The decision variables (potential ply thicknesses and winding angles), the number of combinations, and the ply count are stated in Table 2. The slopes of the fitted objectives with respect to their actual values are presented in Fig. 8. The first and third objectives were trained with excellent effectiveness, which was given by a value close to 1 on the y-axis representing the slope value. On the other hand, the second objective training procedure needed some extra attention due to some slope values far from the maximum value equal to 1. To achieve a good slope value there, more subnets and additional nodes were needed in these subnets.

In Fig. 9, a 3D plot is used to illustrate the complete Pareto front of Geometry2_2, employing the Tsai-Wu definition of the second objective. This allows for a comprehensive visualisation of the entire Pareto set. Subsequently, in Fig. 10, a parallel plot [22] is used to provide a more detailed perspective. Figure 10 comprises two charts, the upper chart corresponds to the parallel plot of the entire Pareto front set, echoing the 3D representation from Fig. 9. Here, the y-axis encompasses decision variables, including ply angles and thicknesses, and the objectives. Each member of the Pareto front is represented by a connecting line, elegantly illustrating the relationships between these variables and objective values, thus defining the entire Pareto front set. The lower chart focusses exclusively on the selected subset within the Pareto front, in which individuals are selected in dependence on their objective values, which should be within the lowest 10% of the computed objective values.

Upon inspection, the existence of distinct members within the Pareto front set becomes evident. To further refine our selection, additional conditions or restrictions can be applied. This multistep selection process begins with an initial parallel plot

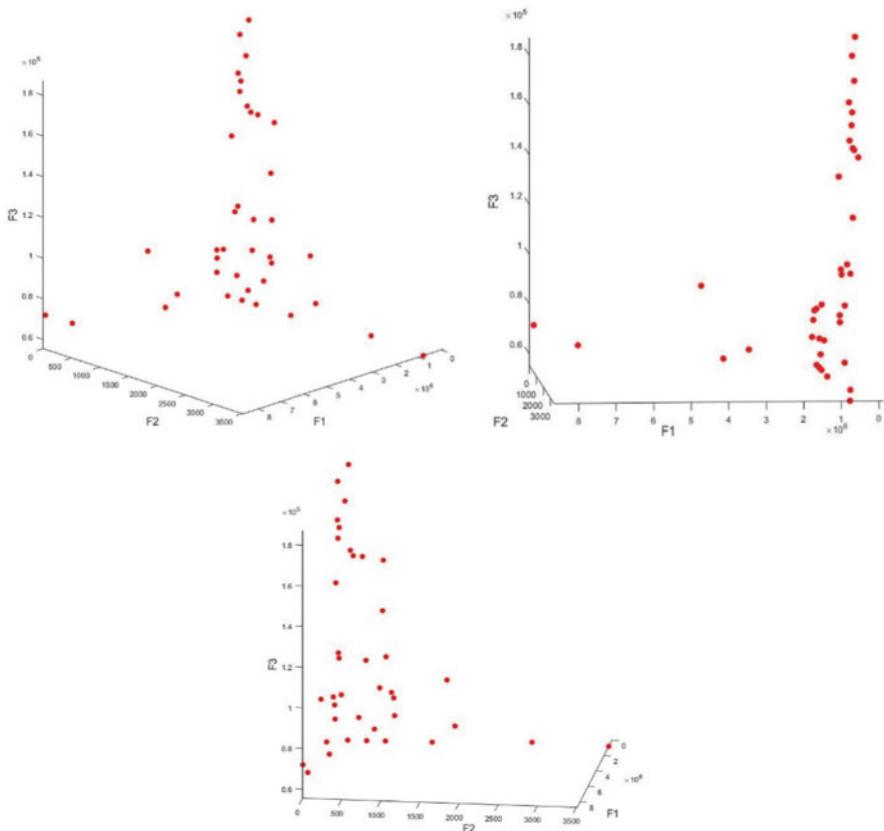


Fig. 9 Geometry2_2 (F2_TW)—Pareto Front with Tsai-Wu Definition plotted on a 3D chart

based on the first set of criteria, followed by more specific selections according to additional established criteria. Figure 10 plays a pivotal role in facilitating these analytical steps, aiding in the exploration and understanding of the Pareto front's composition.

A two-layer tube is used for geom1_3, and its total thickness is determined. This simple geometry was easily processed by the given algorithm, but it will not be used for more complex constructions due to the increased complexity and computational cost. Therefore, the method of including multiply tubes in structures will be a task of further research.

Both geometries were subjected to the location of the decision variables according to the established objective functions. It was done with excellent results, with an optimal set of solutions being found.

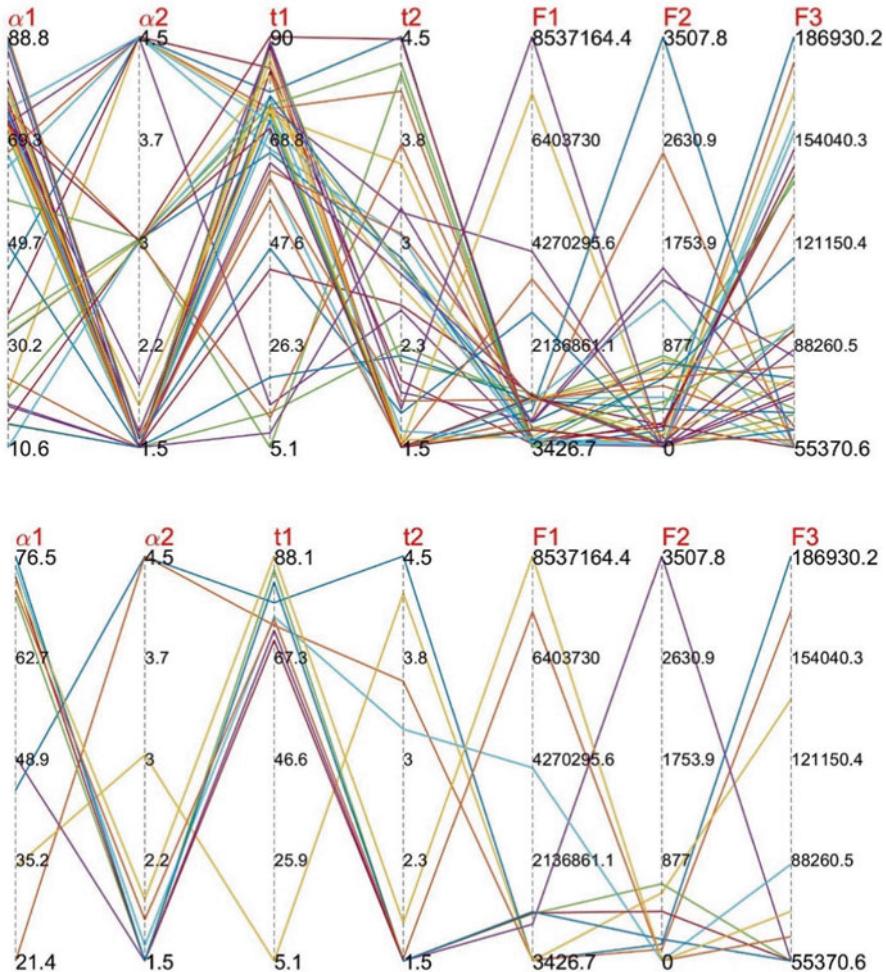


Fig. 10 Geometry2_2 (F2_TW)—Pareto Front Visualization in Parallel Plots: The Entire Pareto Front Set (Upper Chart) and Chosen Members with Extremal Values (Lower Chart)

4.3 Data-Driven Algorithm Applied to Curved Beams and Frames

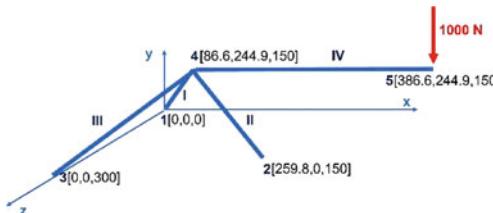
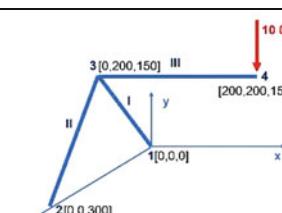
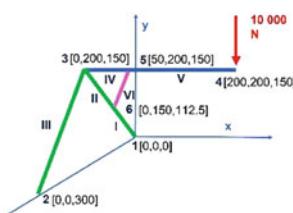
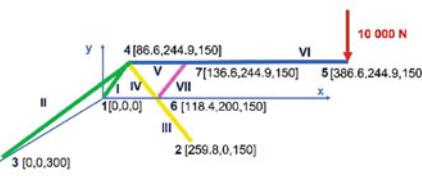
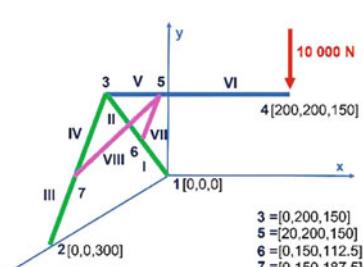
The data-driven algorithm EvoDN2 has been tested on simple geometries and they have worked well. Therefore, they have been applied to more complex tasks, for which the geometries are shown in Table 3. The Arabic numerals are used for the nodes, and the Roman numerals are used for the beams. Each tube in geometry A, C, to E has its own ply thickness and orientation angle. The tubes in geometry B and geometry F to H have been coloured—one colour is connected to the same thickness

Table 3 Frames and curved beams geometries schematics and parameters

	Schematic of geometry	Parameters
Geometry A		<p>ply angle: 0-15-30-...-90 DEG tube thickness: 1.5-3-4.5 mm combinations: 9261 BC: 1 [1;1;1;1;1]</p>
Geometry A – LC2		<p>ply angle: 0-15-30-...-90 DEG tube thickness: 1.5-3-4.5 mm combinations: 9261 BC: 1 [1;1;1;1;1]</p>
Geometry A – LC3		<p>ply angle: 0-15-30-...-90 DEG tube thickness: 1.5-3-4.5 mm combinations: 9261 BC: 1 [1;1;1;1;1]</p>
Geometry B		<p>ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 3 375 BC: 1 [1;1;1;0;0;0] 2 [0;1;1;0;0;0] 3 [1;1;0;0;0;0] 4 [0;1;1;0;0;0]</p>
Geometry C		<p>ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 3 375 BC: nodes 1,2 and 3 [1;1;0;0;0;0]</p>

(continued)

Table 3 (continued)

Geometry D		ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 50 625 BC: nodes 1,2 and 3 [1;1;1;0;0;0]
Geometry E		ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 3 375 BC: nodes 1 and 2 [1;1;1;1;1;1]
Geometry F		ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 3 375 BC: nodes 1 and 2 [1;1;1;1;1;1;1]
Geometry G		ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 50 625 BC: nodes 1,2 and 3 [1;1;1;0;0;0]
Geometry H		ply angle: 5-25-45-65-85 DEG tube thickness: 1 – 3 – 5 mm combinations: 3 375 BC: nodes 1 and 2 [1;1;1;1;1;1;1]

and orientation angle. Tubes are made up of one ply only. Its feasible orientation and thickness are listed in the Parameters column as well as boundary conditions (BC). Boundary conditions are listed using the row vectors for the nodes where the geometry is mounted. The first three positions are for translation in the x, y, and z directions, and the following three positions indicate the rotation of the node in the x, y, and z directions. The zero value signifies the possible movement in the given direction, while the one value means that there is no movement in the given direction (constraint). In addition, the number of combinations of ply angle orientation and ply thickness of the whole structure is mentioned.

4.3.1 Training Run

The adjustment process was complicated for the objective function F2 when Geometries 1 and 2 were trained. To streamline this procedure and mitigate potential challenges, we conducted a dedicated training session for the objective function F2 using Geometry A. The algorithm parameters, including the number of generations (GEN), subnets (SN), nodes in subnets (SN-nodes), the lattice size x similar to y (Lattice), population size (Popsize), and number of predators (Predators), the fitting quality (slope) and computational time (Time) are detailed in 0.

In pursuit of the most efficient parameter configuration, we systematically analysed the values in Table 4, carefully evaluating their impact on computational time and fitting quality, which is highlighted in colour or bold. The fitting parameter should be close to the number 1. After rigorous assessment, variant 14, characterised by specific parameter settings (GEN, SN, SN-nodes, Lattice, Popsize, and Predators), emerged as the optimal choice. This selection was made with keen consideration of both computational efficiency and the quality of the fitting parameters. Eventually, variant 14 was chosen as the preferred configuration for the following structures, ensuring a balance between computational expediency and fitting precision.

Each geometry was trained by three sets of the parameters SN and SN-nodes. The best slopes of all the objective functions are mentioned in Table 5. Expectedly, a better value of the fitted slope would cost more computer time and require more subnets and a greater number of nodes in those subnets. The balance of the fittings rate and computation cost was found for each geometry. The first and third objective functions fit well, while the second objective function was trained in more forms—4 definitions were used to reach the best slope values.

In Table 5 the best second objective definition for each case is chosen and written in bold. It is evident that the best slope is not always detected for one definition and vice versa; each definition finds its use at least in one geometry. On the other hand, slope values higher than 0.75 are in the coloured cells (its intensity increases with the value increasing). It is evident that the smax definition of the second objective function works well for all geometries. This is the reason for selecting F2—smax for the optimisation run.

Table 4 Training of the EvoDN2 Algorithm of F2 on Geometry A

Variant	Gen	SN	SN-nodes	Lattice	PopSize	Predators	Slope	Time
5	100	3	20	20	500	100	0.58	20 min
6	100	6	8	20	500	100	0.60	6 h
7	100	6	8	30	400	100	0.67	6.75 h
8	100	6	8	30	250	100	0.71	4.5 h
9	100	6	20	30	250	100	0.76	10.75 h
10	10	6	20	150	100	70	0.71	9.75 h
11	10	6	20	50	250	100	0.71	3.75 h
12	100	6	20	30	400	100	0.76	19.75 h
13	100	6	20	50	500	100	0.76	29 h
14	10	6	30	50	250	100	0.76	3.5 h
15	15	6	30	50	250	100	0.77	8.25 h
16	10	6	40	50	250	100	0.79	9 h
17	10	6	30	50	200	70	0.79	7.25 h
18	10	6	40	50	200	70	0.79	21.25 h
19	10	6	60	50	250	100	0.81	31 h
20	10	6	30	70	250	100	0.77	24.5 h
21	10	6	40	70	200	70	0.78	29.5 h
22	10	6	60	70	200	70	0.81	35 h
23	10	9	30	50	250	100	0.78	8.5 h
24	10	9	Var*5	50	250	100	0.73	5.5 h

Table 5 Fitting parameters of all objective functions

All	Fitting parameters of all objective functions					
	F1	F2				
		TW	smax	TH	Hoffman	
A	0.97	0.76	0.92	0.71	0.77	1
A-LC2	0.93856	0.74284	0.88	0.73529	0.74868	1
A-LC3	0.99682	0.91	0.95952	0.72	0.91	1
B	1	0.98791	0.99949	0.99931	1	1
C	0.99991	0.85641	0.8837	0.28864	0.84992	1
D	0.99964	0.99526	0.96354	0.8537	0.99523	1
E	0.99772	0.89179	0.92127	0.78727	0.89176	1
F	0.99983	0.95484	0.94807	0.98936	0.95205	1
G	0.99859	0.96764	0.93176	0.81627	0.96744	1
H	0.99952	0.94771	0.95083	0.99648	0.93607	1

4.3.2 Optimisation Run

Once the data-driven model of the problem was formed, it was then prepared for the optimisation run. The cRVEA algorithm was used for the procedure. Due to the very effective procedure of the evolutionary algorithm used, despite using a large number

Fig. 11 cRVEA optimisation configuration

```
%%cRVEA Optimization Configs=====
cRVEAopt.obj = [1 1]; %set 1 for min and -1 for max
cRVEAopt.Generations = 300;
cRVEAopt.p1p2 = num2cell([20 20]); %[p1 p2] define the number
of reference vectors. p1 is the number of divisions along an axis
cRVEAopt.N = 1000; %defines the population size
cRVEAopt.alpha = 0.05; % the parameter in APD, the bigger,
the faster cRVEA converges
cRVEAopt.fr = 0.1; % frequency to call reference vector
cRVEAopt.ieqCon{1} = 'obj1'; %inequality constraints(f(Var,Obj)>0)
cRVEAopt.ieqCon{2} = 'obj2'; %inequality constraints(f(Var,Obj)>0)
cRVEAopt.ieqCon{3} = 'obj3'; %inequality constraints(f(Var,Obj)>0)
```

of generations (300) and population size (1000), as well as a large number of reference vectors, the computation could proceed sufficiently fast. Each objective is constrained by zero, i.e., only positive values are expected. The function value of the second objective should be greater than 0.5 and fewer than 1. The parameters are mentioned in Fig. 11.

4.3.3 Results

The Pareto set is obtained by an optimisation run. The number of members in the Pareto set is different for each geometry and depends on the trained model. The results are shown in parallel plots, which enable the comparison between objective functions and decision variables. As is mentioned in Fig. 12, members of the Pareto front are very different considering the objective functions as well as the decision variables. The upper graph represents the Pareto front set obtained when the requirement about a positive objective value is applied. The lower graph draws additional conditions about the second objective, which should be less than 1 and greater than 0.5.

The decisive parameters to determine the final optimised outputs will be the manufacturer's constraints and the value of the objective function F2, which should be within 0.5 and 1. The Pareto set, which takes into account the boundary condition of the second objective, should satisfy the manufacturer's restriction—the thickness should be multiple of 0.5 mm (± 0.1 mm) and the winding angle should be multiple of 5 DEG (± 1 DEG). The parameters of the members of the Pareto set mentioned for the geometry G are provided in Table 6. The constraints of the decision variables are fulfilled in the cells that are highlighted.

For most of the geometries tested, at least one member of the Pareto front was found, which satisfied all the requirements. When the requirements are not met, the closest possible result is chosen: first, the thickness requirement should be met, due to the higher ability to produce the tube with the winding angle outside the stated bounds.

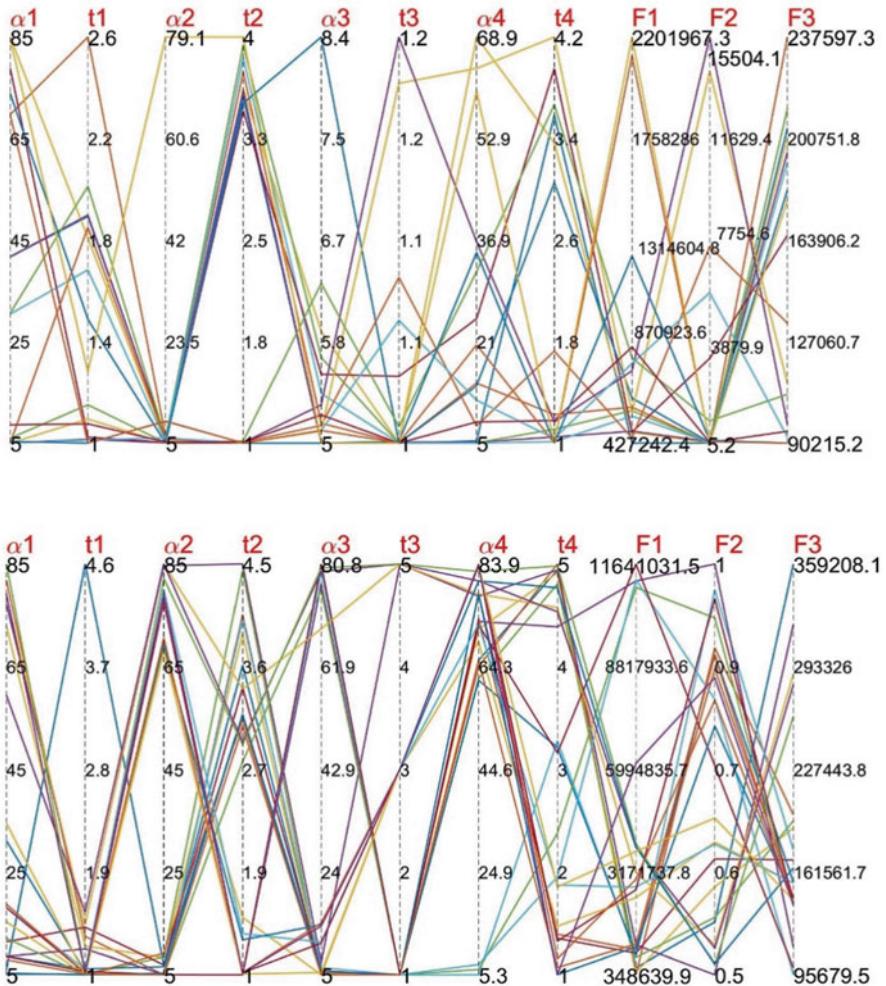


Fig. 12 Geometry G—Pareto Front Visualization in Parallel Plots (tube winding angles α_1 – α_4 and layer thicknesses t_1 – t_4 as decision variables and F_1 – F_3 as objective functions): The Entire Pareto Front Set (Upper Chart) and Chosen Members with Extremal Values (Lower Chart)

It is evident that geometry G has one member (member 18), which fulfils all the requirements, and there are some other members, for which parameters are very close to the specified values (members 10 or 16 for example).

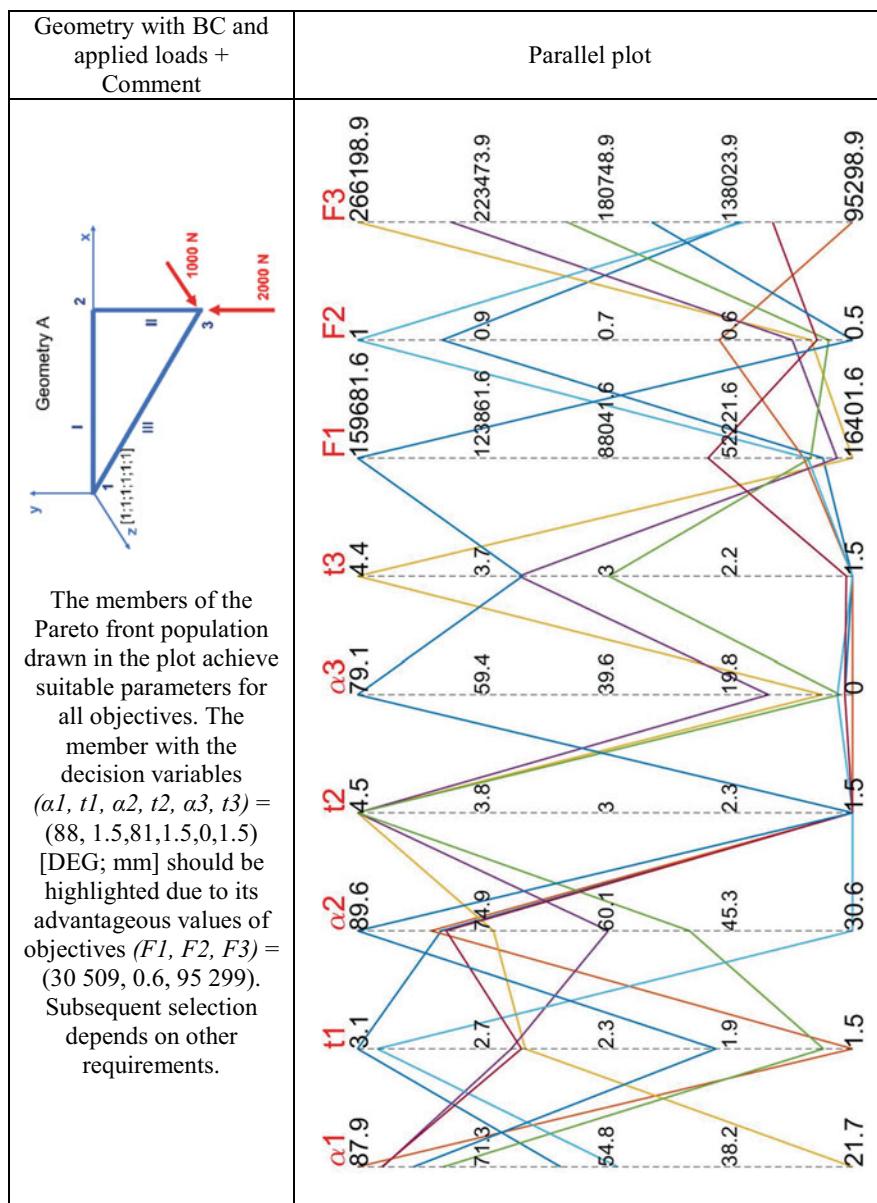
The results of other geometries investigated are shown in Table 7. Only the members of the Pareto front that fulfil all the requirements or are very close to the specified values are included there.

Table 6 The Pareto front members of geometry G considering the manufacturer's constraints

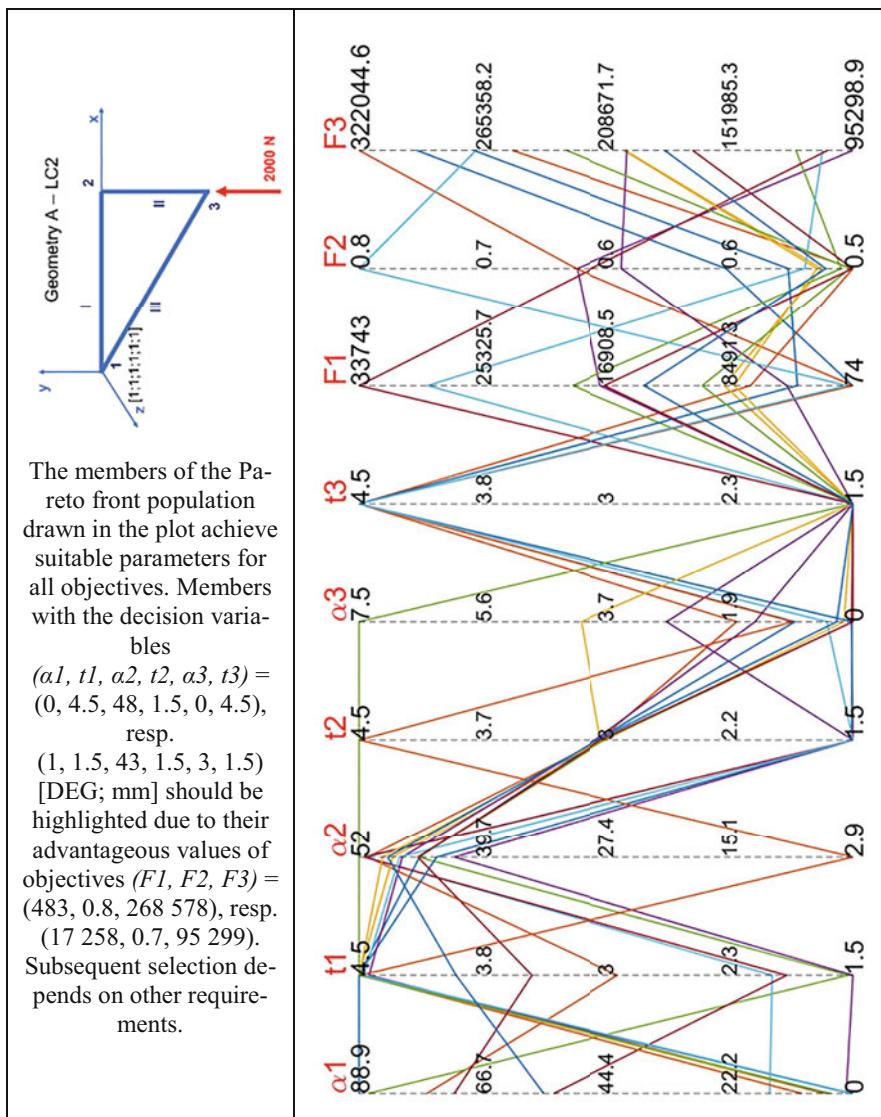
Member	α_1	t_1	α_2	t_2	α_3	t_3	α_4	t_4	F1	F2	F3
1	6.0	4.6	6.9	3.6	5.2	1.0	65.5	4.9	539,201	0.6	359,208
2	81.9	1.3	5.0	4.0	5.0	1.0	64.3	5.0	348,640	0.9	198,201
3	73.0	1.2	8.4	3.9	5.3	1.0	71.4	4.9	410,085	0.6	189,638
4	77.2	1.4	84.7	4.5	7.3	5.0	77.8	4.9	1,166,225	0.5	320,520
5	5.3	1.0	8.5	4.4	5.0	1.0	62.0	4.8	824,207	0.6	194,480
6	5.0	1.0	5.0	4.0	6.3	1.0	5.3	3.3	842,100	1.0	175,837
7	11.4	1.4	6.9	3.4	5.7	1.0	78.3	1.4	941,285	0.6	169,391
8	8.6	1.1	6.6	3.2	5.6	1.0	61.7	3.2	902,973	0.9	155,202
9	5.2	1.0	9.2	3.2	5.0	1.0	73.6	1.4	1,038,498	0.9	145,423
10	15.3	1.0	67.9	1.5	5.0	3.0	69.0	1.5	2,470,705	0.7	154,693
11	8.6	1.2	5.0	1.0	77.2	1.0	73.0	4.4	11,173,439	1.0	113,430
12	12.1	1.0	5.0	2.8	76.0	1.0	6.3	2.4	11,001,053	0.9	140,055
13	8.5	1.0	5.0	1.0	79.5	1.0	7.2	1.9	11,216,664	0.8	95,680
14	8.5	1.0	5.0	1.0	80.8	1.0	72.3	3.1	11,641,031	0.8	99,878
15	5.3	1.0	5.3	3.2	5.0	1.0	78.4	1.0	1,111,475	0.8	144,431
16	7.9	1.0	5.0	3.1	5.0	1.0	65.6	1.1	1,189,946	0.9	142,446
17	78.8	1.4	85.0	3.4	68.9	5.0	78.2	4.6	3,164,040	0.5	288,840
18	59.4	1.5	85.0	3.0	79.4	5.0	80.7	4.5	3,852,211	0.5	281,806
19	85.0	1.0	82.2	2.9	80.1	5.0	82.5	5.0	3,958,520	0.5	261,339
20	18.1	1.0	80.1	13	11.0	3.0	72.5	1.9	2,757,858	0.7	154,556
21	18.0	1.0	77.7	1.0	10.8	3.0	83.9	1.3	2,686,893	1.0	142,235
22	31.1	1.0	69.3	1.3	14.2	3.0	80.7	4.7	3,867,554	0.5	164,300
23	18.7	1.0	70.6	1.0	14.5	3.0	83.7	1.2	2,976,314	0.8	141,378
24	34.5	1.0	68.4	1.0	5.4	3.0	83.7	1.8	3,742,356	0.7	143,511
25	79.2	1.0	79.3	1.0	14.0	3.0	83.6	1.0	6,161,756	0.9	140,420

5 Conclusions

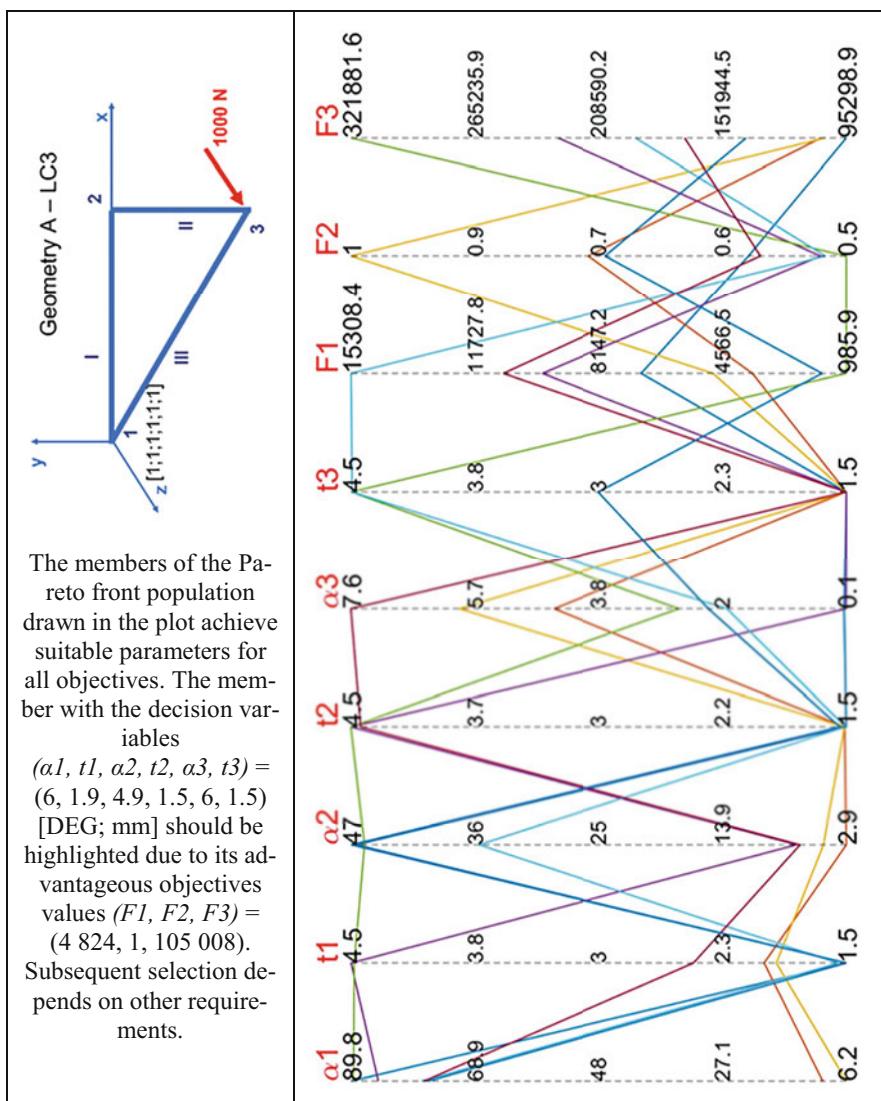
Structures should be designed considering the material used. The use of orthotropic materials such as wound composite tubes allows for the purposeful selection of material parameters with regard to the required properties of the laminate structure. Metaheuristic algorithms are usually used for the laminate optimisation process due to their noncontinuous design variable represented stacking sequence: ply thickness and wound angle. More complex problems are solved with benefit by using data-driven model approaches, which allow one to contain physical essence of the given problem without directly invoking it. Data-driven evolutionary algorithms were used to optimise the laminate stacking sequence of curved beams and frames. Simple geometries were successfully solved using Bi-Objective Genetic Programming (BioGP), Evolutionary Neural Network (EvoNN), and Evolutionary Deep Neural Net (EvoDN2) algorithms. More complex geometries require the use of the more complex EvoDN2 algorithm to be solved efficiently and successfully. EvoDN2 due to its deployment of separate subnets with more hidden layers handled the stated

Table 7 The Pareto front members of all geometries

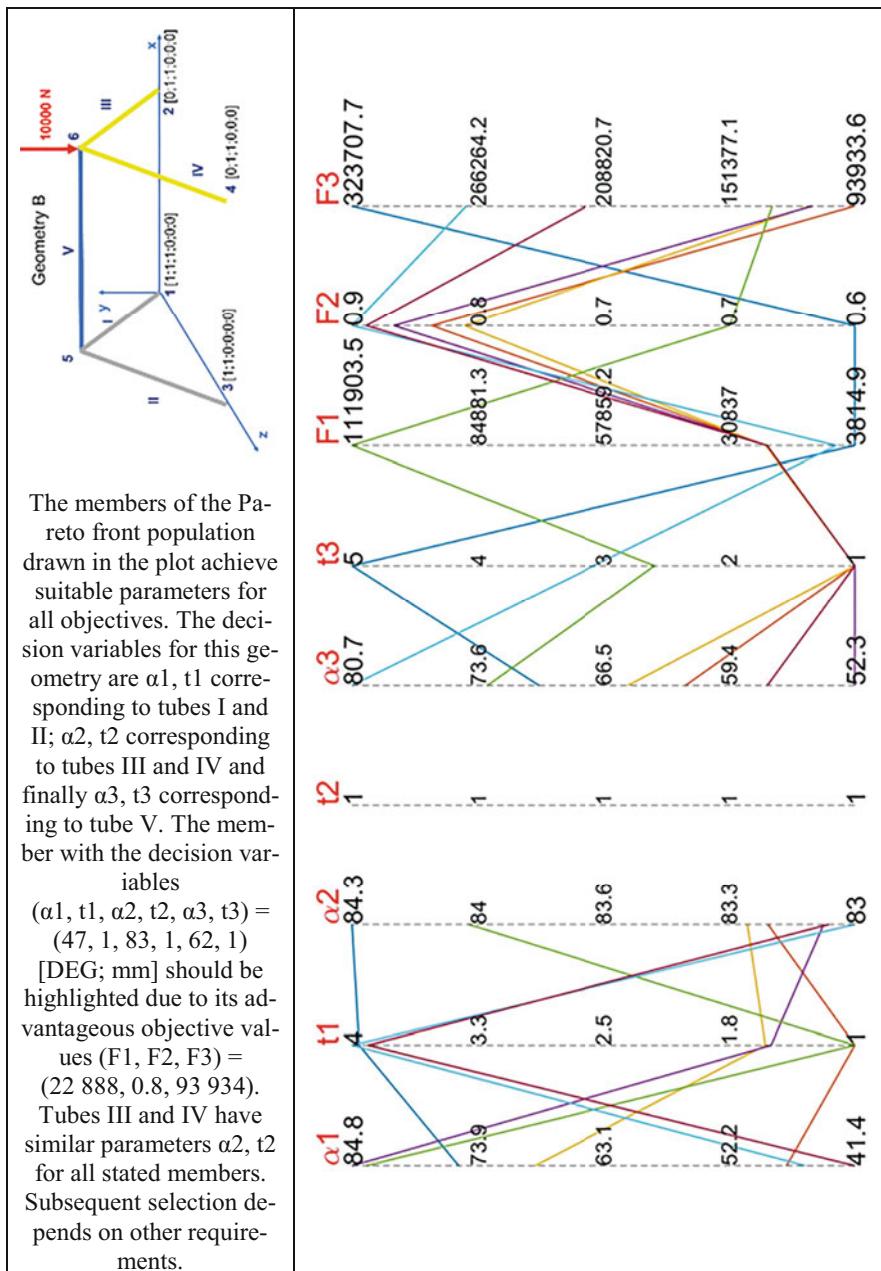
(continued)

Table 7 (continued)

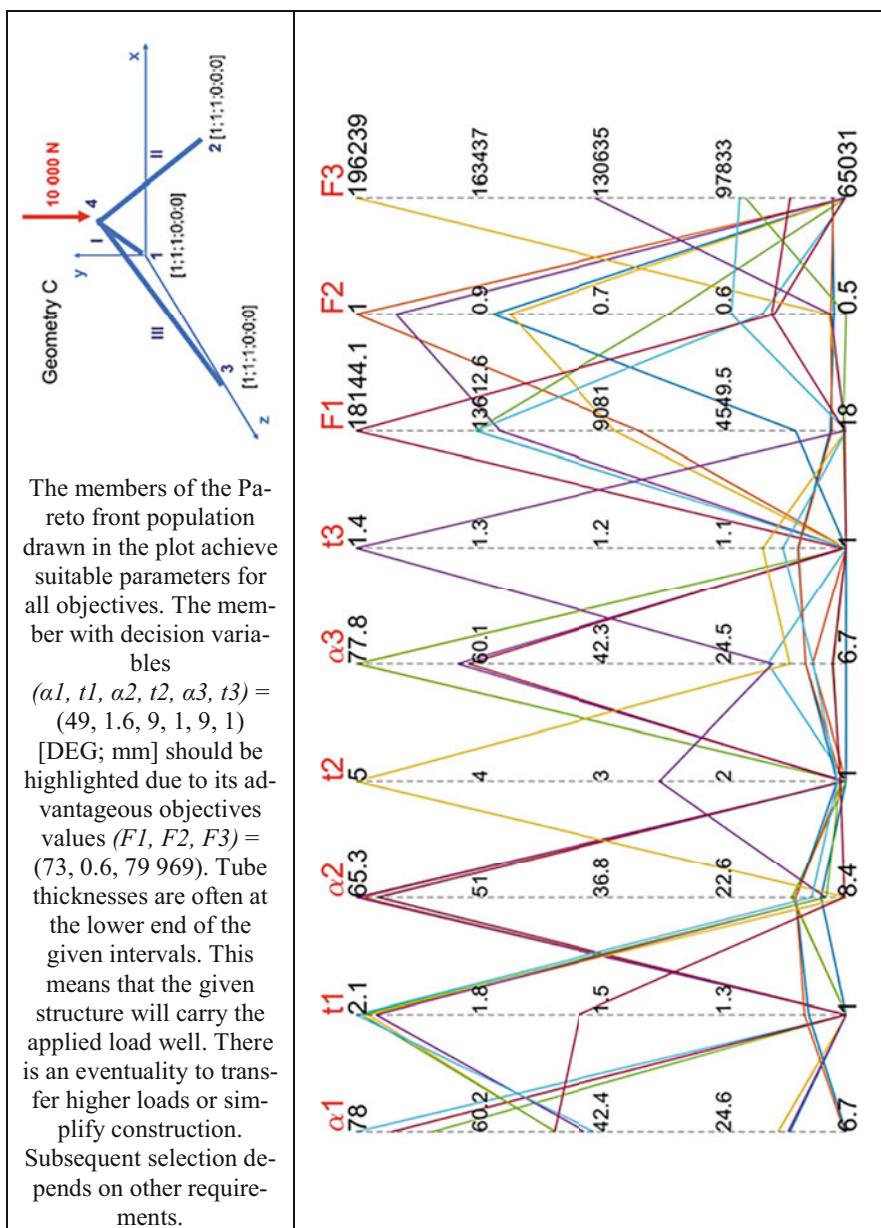
(continued)

Table 7 (continued)

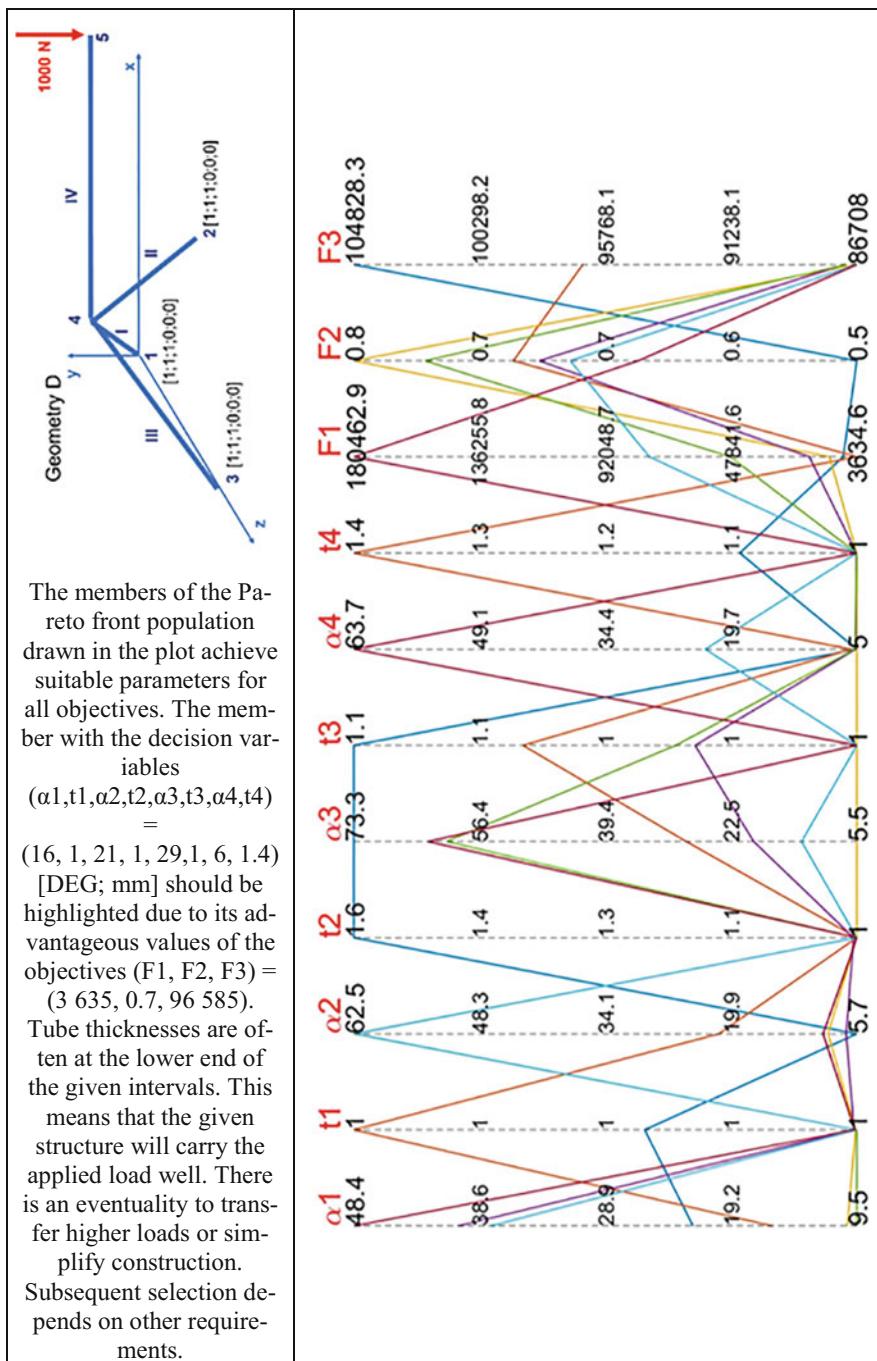
(continued)

Table 7 (continued)

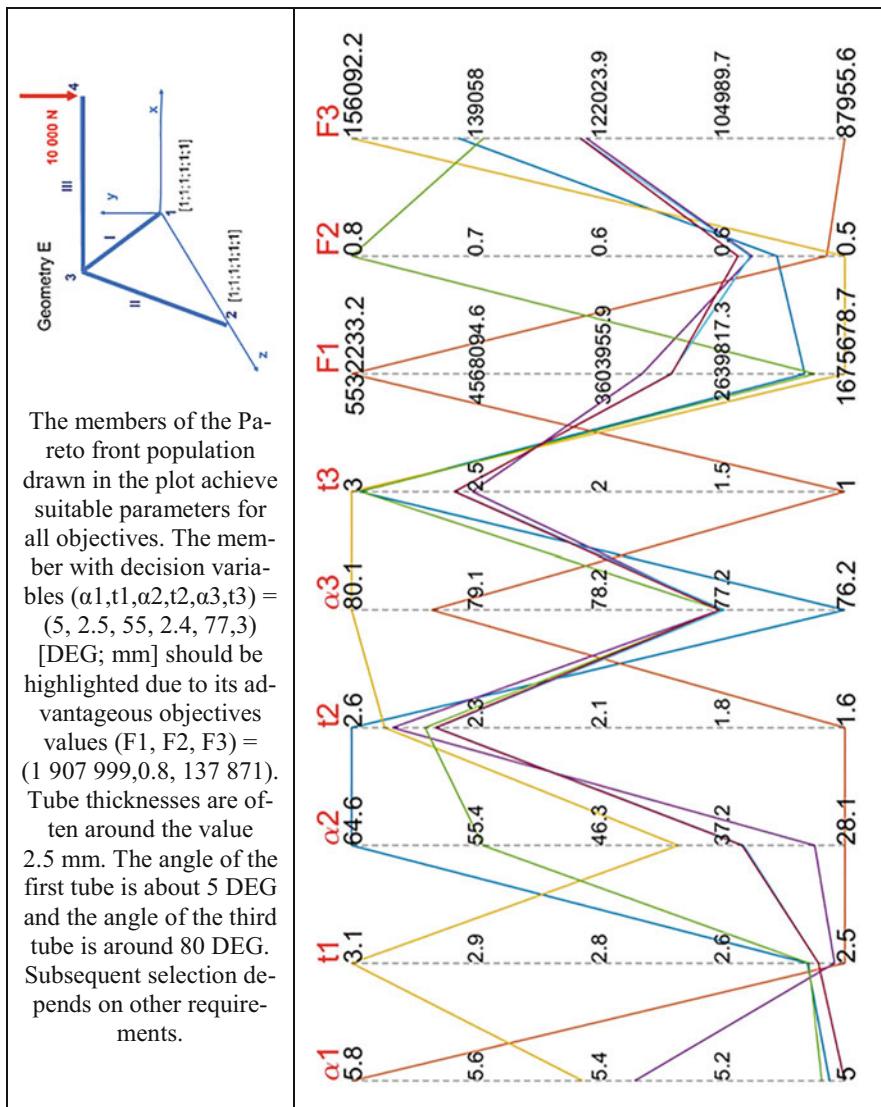
(continued)

Table 7 (continued)

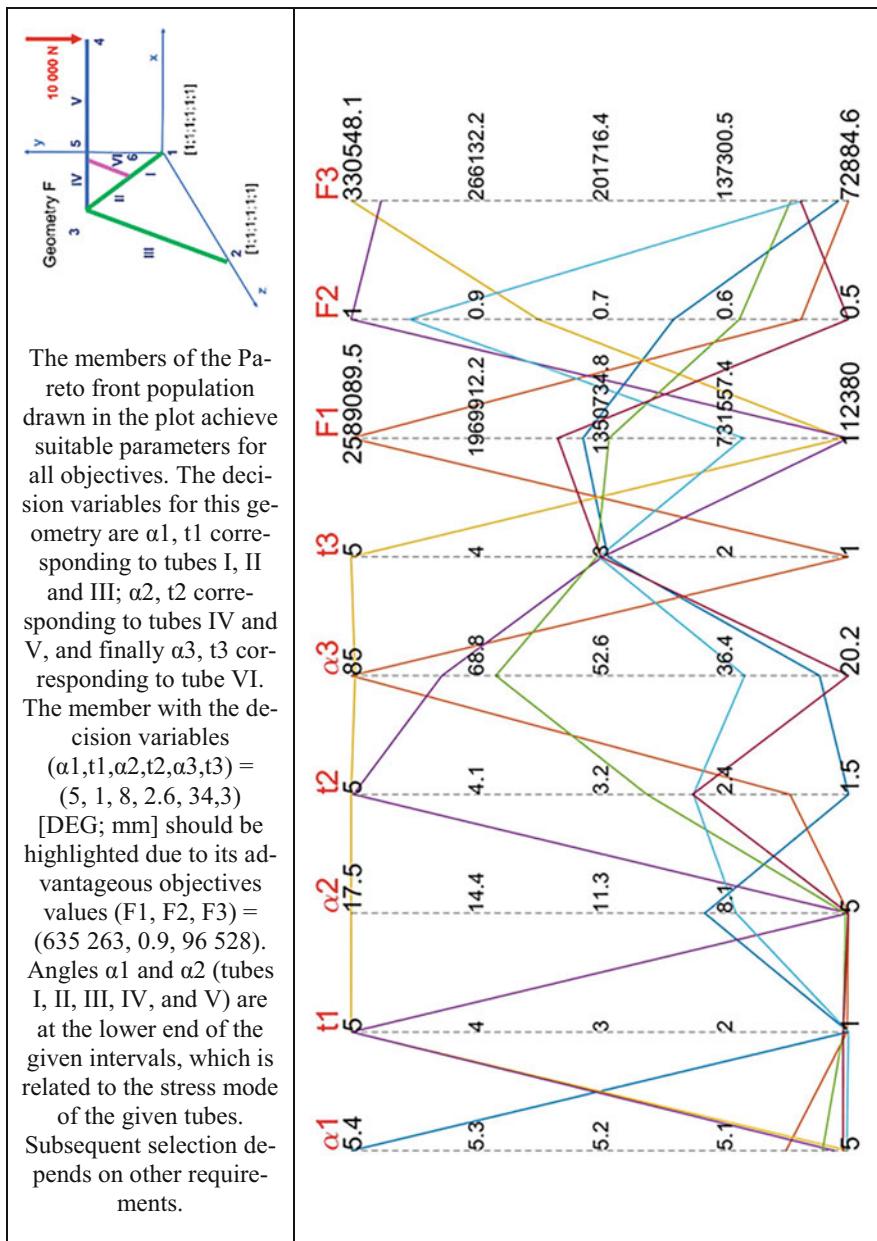
(continued)

Table 7 (continued)

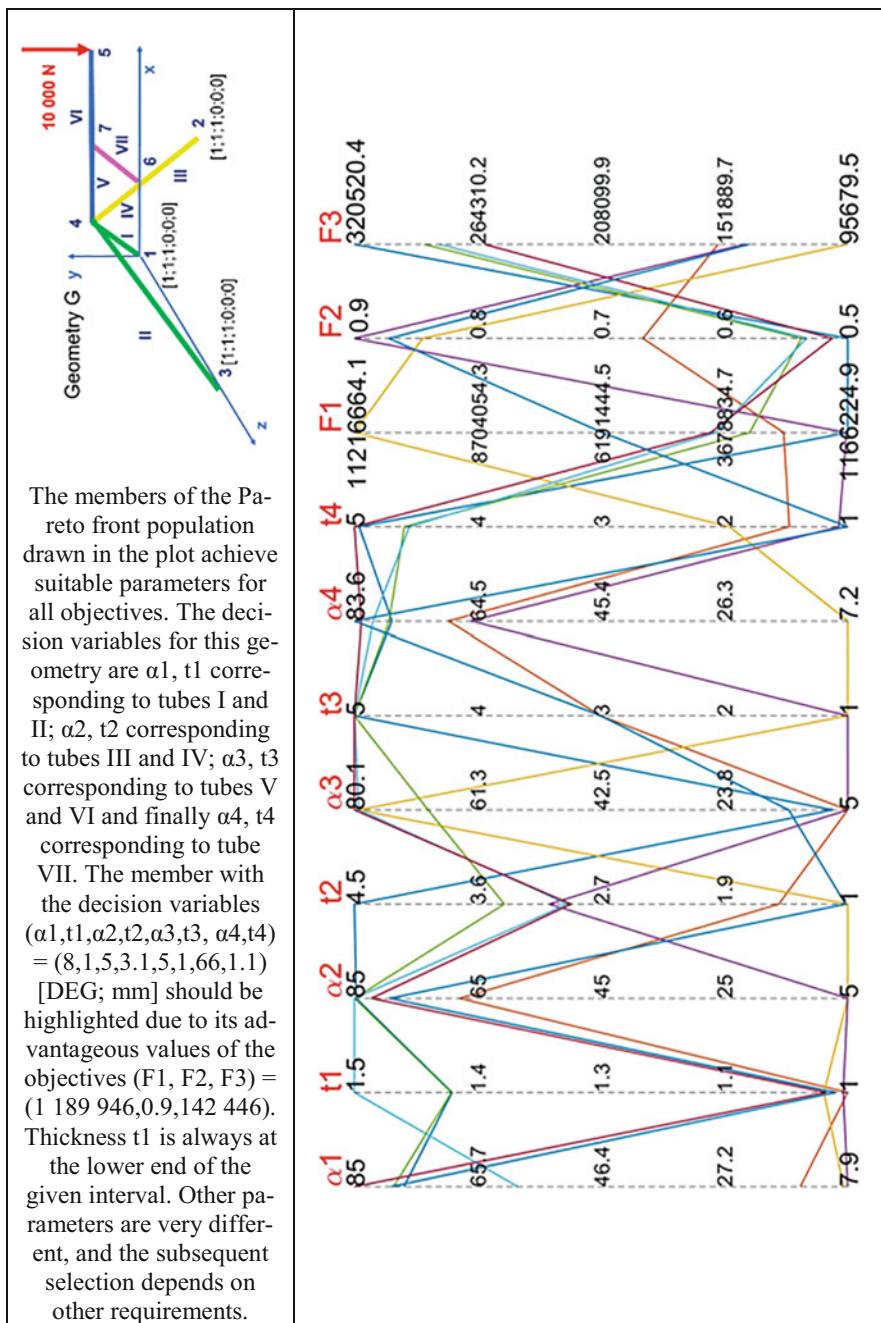
(continued)

Table 7 (continued)

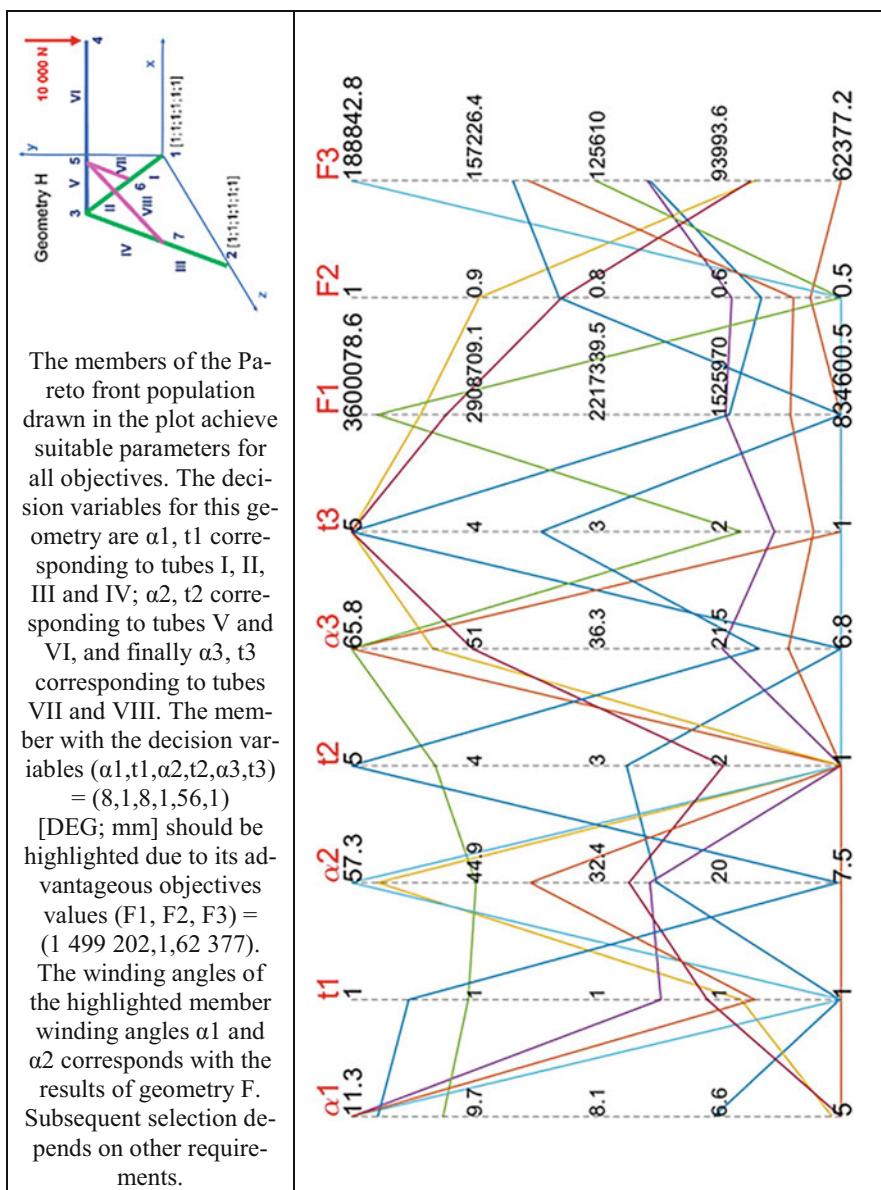
(continued)

Table 7 (continued)

(continued)

Table 7 (continued)

(continued)

Table 7 (continued)

problems well. Pareto front sets were found. Selection of the specific Pareto front member needed a determination of additional requirements. Incorporation of these methods into the laminate design process can lead to finding the required material parameters without spending a very large amount of time on the computation of complex composite material mechanics equations.

References

1. Hooke R. *Lectures de Potentia Restitutiva, or of spring explaining the power of springing bodies*. London: John Martyn; 1678.
2. Rice JR. Mechanics of solids. Encyclopedia Britannica. 2017. <https://www.britannica.com/science/mechanics-of-solids>. Accessed 18 Mar 2023.
3. Lu C, Pilla S. Design of automotive composites. Warrendale: SAE International; 2014.
4. Mahanthesh MR, Prashanth RK, Kirankumar NA. Numerical analysis for a bicycle frame made of mild steel and composite. *Int J Sci Eng Dev Res*. 2017;3(4):40–6.
5. Elanchezhian C, et al. Design and comparison of the strength and efficiency of drive shaft made of steel and composite materials. *Mater Today Proceed*. 2018;5:1000–7.
6. Padovec Z, Křena J, Sedláček R, et al. Experimental and numerical analyses of optimized composite profiles for aircraft construction. *Mech Compos Mater*. 2022;58:283–92. <https://doi.org/10.1007/s11029-022-10029-y>.
7. Jones RM. Mechanics of composite materials. 2nd ed. London: Taylor & Francis; 1999.
8. Vasiliev VV, Morozov EV. Advanced mechanics of composite materials. 2nd ed. Amsterdam: Elsevier; 2007.
9. Gibson RF. Principles of composite material mechanics. 3rd ed. Abingdon: Taylor & Francis; 2011.
10. Nikbakt S, Kamarian S, Shakeri M. A review on optimization of composite structures Part I: laminated composites. *Compos Struct*. 2018;195:158–85.
11. Almeida JHS, et al. Buckling optimization of composite cylinders for axial compression: a design methodology considering a variable-axial fiber layout. *Compos Struct*. 2019;222: 110928.
12. Nelson SM, Thota J, O'Toole B. Optimization of carbon fiber/epoxy tubes loaded in bending and compression. In: International SAMPE Symposium and Exhibition, Seattle WA, 17–20 May 2010.
13. Schaedler de Almeida F. Optimization of laminated composite structures using harmony search algorithm. *Compos Struct*. 2019;221:110852.
14. Gürdal Z, Haftka R, Hajela P. Design and optimization of laminated composite materials. New York: Wiley; 1999.
15. Todoroki A, Haftka RT. Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy. *Compos Part B*. 1998;29:277–85.
16. Chen S, Lin Z, An H, et al. Stacking sequence optimization with genetic algorithm using a two-level approximation. *Struct Multidiscip Optim*. 2013;48:795–805.
17. An H, Chen S, Huang H. Laminate stacking sequence optimization with strength constraints using two-level approximations and adaptive genetic algorithm. *Struct Multidiscip Optim*. 2015;51:903–18.
18. Duan Z, Yan J, Lee I, et al. Discrete material selection and structural topology optimization of composite frames for maximum fundamental frequency with manufacturing constraints. *Struct Multidiscip Optim*. 2019;60:1741–58.
19. Yan J, Duan Z, Lund E, et al. Concurrent multi-scale design optimization of composite frames with manufacturing constraints. *Struct Multidiscip Optim*. 2017;56:519–33.

20. Vondráček D, Padovec Z, Mareš T, Chakraborti N. Optimization of dome shape for filament wound pressure vessels using data-driven evolutionary algorithms. *Mater Manuf Process.* 2023; <https://doi.org/10.1080/10426914.2023.2187823>.
21. David P, Mareš T, Chakraborti N. Evolutionary multi-objective optimization of truss topology for additively manufactured components. *Mater Manuf Process.* 2023;38(3):1–10.
22. Chakraborti N. Data-driven evolutionary modeling in materials technology. 1st ed. Boca Raton: CRC Press; 2022.
23. Roy S, Chakraborti N. Development of an evolutionary deep neural net for materials research. In: TMS 2020 149th annual meeting & exhibition supplemental proceedings. Cham: Springer; 2020. p. 817–28.
24. Miettinen K. Nonlinear multiobjective optimization. Boston: Kluwer Academic Publishers; 1999.
25. Long Q, Wu C, Wang X, Jiang L, Li J. A multi-objective genetic algorithm based on a discrete selection procedure. *Math Probl Eng.* 2015;2015(349781)
26. Jin Y, Wang H, Chugh T, Guo D, Miettinen K. Data-driven evolutionary optimization: an overview and case studies. *IEEE Trans Evol Comput.* 2019;23:442–58.
27. He T, Wang H, Yoon SW. Comparison of four population-based meta-heuristic algorithms on pick-and-place optimization. *Proc Manuf.* 2018;17:944–51.
28. Naeem M, et al. Trends and future perspective challenges in big data. In: Pan JS, Balas VE, Chen CM, editors. Advances in intelligent data analysis and applications. smart innovation, systems and technologies, vol. 253. Singapore: Springer; 2022. p. 309–25.
29. Chakraborti N. Evolutionary data-driven modeling. In: Krishna R, editor. Informatics for materials science and engineering, vol. 2013. Oxford: Butterworth-Heinemann; 2013. p. 71–95.
30. Chakraborti N. Promise of multiobjective genetic algorithms in coating performance formulation. *Surf Eng.* 2014;30:79–82.
31. Roy S, Chakraborti N. Novel strategies for data-driven evolutionary optimization. In: Tuovinen T, Periaux J, Neittaanmäki P, editors. Computational sciences and artificial intelligence in industry. Intelligent systems, control and automation: science and engineering, vol. 76. Cham: Springer; 2022. p. 11–25.
32. Roy S, Saini B, Chakrabarti D, Chakraborti N. Mechanical properties of micro-alloyed steels studied using an evolutionary deep neural network. *Mater Manuf Process.* 2019;35:611–24.
33. Saini B, Chakraborti N. Unpublished research. Kharagpur: Indian Institute of Technology; 2018.
34. Li X. A real-coded predator-prey genetic algorithm for multiobjective optimization. In: International conference on evolutionary multi-criterion optimization (EMO-2003), vol. 55. Berlin: Springer; 2003. p. 207–21.
35. Pettersson F, Chakraborti N, Singh SB. Neural networks analysis of steel plate processing augmented by multi-objective genetic algorithms. *Steel Res Int.* 2007;78:890–8.
36. Cheng HC, et al. A high-throughput multiobjective genetic-algorithm workflow for in situ training of reactive molecular dynamics force fields. In: Proceedings of the 24th High Performance Computing Symposium, Society for Computer Simulation International, Pasadena, 3–6 Apr 2016.
37. Jordan MI. Why the logistic function? A tutorial discussion on probabilities and neural networks. 1995. https://www.ics.uci.edu/~smyth/courses/cs274/readings/jordan_logistic.pdf. Accessed 26 Feb 2022.
38. Agarwal A, et al. Analysing blast furnace data using evolutionary neural network and multiobjective genetic algorithms. *Ironmak Steelmak.* 2010;37:353–9.
39. Roy S, Dutta A, Chakraborti N. A novel method of determining interatomic potential for Al and Al-Li alloys and studying strength of Al-Al₃Li interphase using evolutionary algorithms. *Comput Mater Sci.* 2021;190:110258.
40. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA: MIT Press; 2017.
41. Pettersson F, Chakraborti N, Saxén H. A genetic algorithms based multi-objective neural net applied to noisy blast furnace data. *Appl Soft Comput.* 2007;7:387–97.

42. Chakraborti N. Strategies for evolutionary data-driven modeling in chemical and metallurgical systems. In: Valadi J, Siarry P, editors. Applications of metaheuristics in process engineering. Cham: Springer; 2014. p. 89–122.
43. Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Trans Syst Man Cybernet Part A Syst Hum.* 1998;28:26–37.
44. Cheng R, Jin Y, Olhofer M, Sendhoff B. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans Evol Comput.* 2016;20:773–91.
45. Cheng R, Jin Y, Narukawa K, Sendhoff B. A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling. *IEEE Trans Evol Comput.* 2015;19:838–56.
46. Cornell JA. Experiments with mixtures: designs, models, and the analysis of mixture data. Hoboken: Wiley; 2011.
47. Dössinger C, Spitaler T, Reichmann A, et al. Applications of data driven methods in computational materials design. *Berg Huettenmaenn Monatsh.* 2022;167:29–35.
48. Yuan M. Prediction of matrix-cracking-induced stiffness degradation of cross-ply laminates based on data-driven method. *Compos Sci Technol.* 2022;230:109716.
49. Kazi M-K, Eljack F, Mahdi E. Data-driven modeling to predict the load vs. displacement curves of targeted composite materials for industry 4.0 and smart manufacturing. *Compos Struct.* 2021;258:113207.
50. Shah V, Zadourian S, Yang C, Zhang Z, Gu GX. Data-driven approach for the prediction of mechanical properties of carbon fiber reinforced composites. *Mater Adv.* 2022;3:7319–27.
51. Soutis C. Carbon fiber reinforced plastics in aircraft construction. *Mater Sci Eng A.* 2005;412: 171–6.
52. Zhu B. The finite element method: fundamentals and applications in civil, hydraulic, mechanical and aeronautical engineering. Newark: Wiley; 2018.
53. Rao SS, Rao SS. The finite element method in engineering. Burlington: Elsevier Sci Technol; 2004.

Polymer and Nanocomposite Informatics: Recent Applications of Artificial Intelligence and Data Repositories



Neelesh Ashok, K. P. Soman, Madhav Samanta, M. S. Sruthi,
Prabaharan Poornachandran, Suja Devi V. G, and N. Sukumar

Abstract Artificial intelligence (AI) and machine learning (ML) have a variety of practical technological applications and are now impacting how we live and work in myriad ways. Polymer informatics, which employs AI and ML to aid in the development, design, and discovery of polymers, is a fast-expanding field. Models trained on polymer data available in databases make it possible to rapidly predict a range of polymer properties, and to screen prospective polymer candidates for desirable characteristics. AI and ML are also used to predict the ease of synthesis of a target polymer and plan its (retro)synthetic steps. Data-driven techniques are employed to develop machine-understandable polymer representations, and to handle the enormous chemical and physical variability of polymers at multiple scales. Cutting-edge generative AI methods are now being employed for inverse design of polymers with specific properties that make them attractive for various applications. These advances in polymer informatics hold out promise for improved efficiency, accelerated development, and greater ease of manufacture of a new generation of polymers useful to society. This chapter provides an overview of recent developments in AI-aided, data-driven polymer chemistry and polymer nanocomposites, and presents a few recent case studies highlighting the scope and diversity of recent applications of AI techniques in the polymer design process, as well as the state of the art and challenges in polymer informatics.

N. Ashok · K. P. Soman · M. S. Sruthi · N. Sukumar (✉)

Amrita School of Artificial Intelligence, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India

e-mail: a_neelesh@cb.amrita.edu; kp_soman@amrita.edu; ms_sruthi@cb.amrita.edu;
n_sukumar@cb.amrita.edu

M. Samanta

Department of Chemistry, School of Natural Science, Shiv Nadar Institution of Eminence, Dadri, Gautam Buddha Nagar, Uttar Pradesh, India

P. Poornachandran · S. Devi V. G

Centre for Internet Studies and Artificial Intelligence, Amrita Vishwa Vidyapeetham, Amritapuri, Clappana, Kerala, India

e-mail: praba@am.amrita.edu; sujap@am.amrita.edu

Keywords Polymer informatics · Polymer nanocomposites · Polymer data repositories · Artificial intelligence · Deep learning · Polymer chemical space · Generative models

1 Introduction

Polymer informatics is a new interdisciplinary field at the interface of polymer chemistry, computer science, informatics, and machine learning. Polymers are macromolecules, formed by chemical combination of simpler building blocks called monomers. Polymers constitute a substantial fraction of synthetic and natural materials, including many of the macromolecules found in living organisms, such as proteins and DNA. Polymeric materials are essential in daily life and are used in a variety of materials and applications, including plastics, rubber, fibres, coatings, adhesives, food packaging, water purification, clothing, housing, medical items, electronics, and transportation. Several characteristics of polymers—such as low cost, high strength-to-mass ratio, high chemical resistance, ease of synthesis and processing—make them ideal for a variety of applications. However, as polymeric materials accumulate and partially decompose into the environment, they also have hidden long-term societal costs, that are often discounted, delayed, socialized or unidentified [1, 2]. This adds to the imperative for design of new, green polymers.

Polymerization is the process of combining monomers to produce a polymer chain. The covalent bonds formed by this reaction create a long, linked chain of repeating units. Condensation and addition polymerization are the two main kinds of polymerization. A developing polymer chain is added to repeatedly during addition polymerization. The presence of an initiator, such as a radical or an ion, is often necessary for this process. The initiator breaks a bond inside the monomer to produce a reactive species, which starts the reaction. After reacting with another monomer, the reactive species creates a new bond that lengthens the polymer chain. This process keeps repeating until all the monomers are used up or the reaction is stopped. By contrast, in condensation polymerization, a small molecule, such water or an alcohol, is eliminated to create a polymer chain. Small molecules are eliminated in this manner continuously until a polymer chain is produced.

Cross-linking is the act of forming chemical links between polymer chains to create a three-dimensional network. There are several ways to cross-link materials, including using heat, radiation, or specially formulated chemical substances called cross-linking agents. The mechanical, thermal, and chemical characteristics of the polymer can be improved by cross-linking, increasing its rigidity, durability, and deformation resistance. A polymer's degree of cross-linking can range from barely cross-linked to heavily cross-linked networks. While strongly cross-linked polymers are hard and brittle, lightly cross-linked polymers maintain some flexibility and elasticity. The kind and concentration of the cross-linking agents, the chemical makeup of the polymer, and the circumstances under which the cross-linking reaction occurs are some of the variables that affect the degree of cross-linking, and help to explain why polymers have such a wide range of characteristics and are

used in so many different sectors. Cross-linking is essential in many applications. For instance, it enhances rubber's suppleness, durability, and resistance to wear and ageing. Cross-linked polymers are also often utilised in coatings, adhesives, and composites, where their improved mechanical qualities impart strength and endurance.

By 2050, there will likely be ten billion people on the planet, which will lead to an increase in demand for professional healthcare, individualized consumer goods, and high-efficiency clean energy. Traditional technologies face major challenges on account of ever-increasing demand for new materials. Polymer informatics has become a viable strategy to accelerate discoveries in the field of polymer science, by means of high-throughput computing, machine learning (ML), and artificial intelligence (AI). Predicting polymer characteristics can be facilitated using robust and trustworthy ML models based on fundamental physical and chemical properties [3, 4]. Predictive models for a target property can then be employed for high-throughput screening of potential polymer candidates. Experimental design algorithms can reduce the number of new tests necessary to achieve a design goal, because polymer synthesis can be an expensive and labour-intensive operation. For a polymeric material to be a strong contender in any given application, it must satisfy several requirements. With training datasets to enable application of contemporary ML techniques, these algorithms mine the information in the data for hidden patterns, to recommend the best candidates to achieve the design objectives. ML algorithms can now predict a variety of polymer characteristics with a fair degree of accuracy, enabling data-driven analyses, forecasts, and recommendations for polymer design. Polymer informatics can thus be a powerful technique for the discovery of new polymers, and many research groups now employ data-driven approaches to design new functional polymers or to improve their productivity [2, 5].

The typical polymer machine learning pipeline, illustrated in Fig. 1, involves several steps, including data collection, data curation and preprocessing, feature extraction, model training, validation, model evaluation and deployment.

1. *Data collection*: The first step is the collection of data (polymer chemical structures and properties) on the polymers being studied. This data can come from a variety of sources, including experimental measurements, first principles computations or simulations, and literature.
2. *Data curation and preprocessing* are necessary to ensure that the data is accurate, reliable, and representative of the polymers being studied. Preprocessing may involve data cleaning, removal of outliers, and normalization.
3. *Feature extraction*: This involves calculation of various molecular descriptors that represent the polymers being studied in numerical or categorical form. Table 1 lists some families of descriptors and fingerprints commonly used in polymer informatics.
4. *Model training*: After feature extraction, a machine learning model is trained on the data to uncover complex patterns and relationships within the data. This involves selection of an appropriate ML algorithm and optimization of the model parameters to minimize the prediction error of the model.

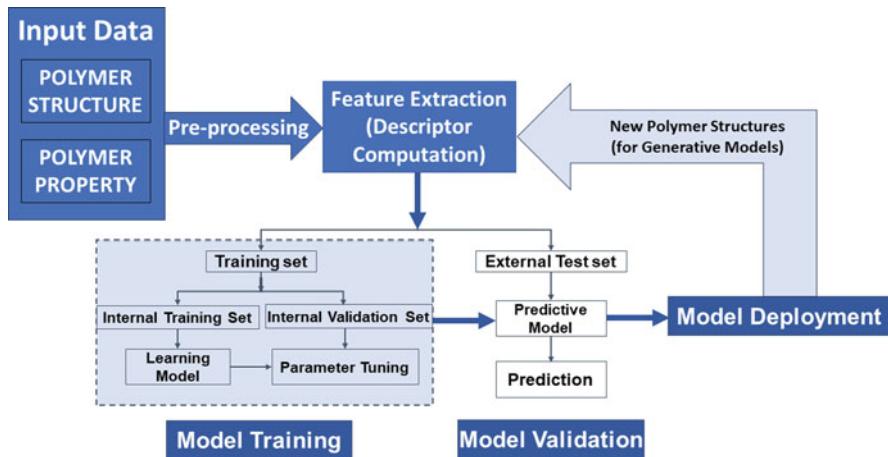


Fig. 1 The typical polymer ML pipeline

Table 1 Some families of descriptors and fingerprints used for feature extraction in polymer informatics

Descriptor/fingerprint type	Description	Applications
Physicochemical descriptors	Represent chemical properties (e.g., MW, logP)	Predicting physicochemical properties, such as toxicity, solubility
Structural descriptors	Capture molecular structure (e.g., SMILES)	Comparing polymer structures, identifying functional groups
Topological descriptors	Describe molecular connectivity (e.g., TPSA)	Quantifying molecular complexity, predicting bioavailability
Electronic descriptors	Relate to electron distribution (e.g., HOMO energy)	Predicting reactivity, electronic properties
Geometrical descriptors	Define 3D structural features (e.g., RMSD)	Analyzing molecular conformations, docking
Structural fingerprints	Binary vectors encoding structural patterns	Similarity searching, clustering, machine learning
Circular fingerprints	Capture substructures	Substructure searching, scaffold hopping
MACCS keys	Set of 166 keys representing substructures	Compound classification, diversity analysis
Extended connectivity fingerprints (ECFP)	Encode atom environments	Cheminformatics, virtual screening
Morgan fingerprint	Circular fingerprints with varying radii	Chemical similarity, compound clustering
RDKit fingerprint	Open-source toolkit molecular fingerprints	Cheminformatics, machine learning

5. *Model validation and evaluation:* The trained model must be rigorously validated to ensure that it is accurate and capable of making reliable predictions on new data that were not used in training. For this purpose, the data set is often split into a training set, an internal validation set and an external test set. The internal validation set is used to tune the parameters of the model so as to generate the most predictive model. The external test set is never seen by the model during training, and performance on this external test set is the only reliable indicator of the predictivity of the model.
6. *Model deployment:* After model evaluation and validation, it can be deployed in real-world applications to make predictions on new data sets. In generative models, this also involves generating novel structures and repeating the model training and evaluation process iteratively, to accomplish the goal of inverse design of new polymers with properties in a specific desired range.

This integrated approach drives the discovery, prediction, and optimization of polymer materials, accelerating research and catalyzing innovation across industries reliant on polymeric materials.

Researchers today are increasingly utilising polymer informatics to build novel materials and analyse their structure-property relationships. Such informatics methods are useful for prediction of polymer properties that are difficult to measure experimentally or to compute from first principles. Predictions are typically interpolative, involving first the computation of descriptors or “fingerprints”, followed by discovering correlations between the descriptors/fingerprints and the property of interest. Here a fingerprint can refer to any numerical or categorical descriptor derived from the monomer, repeat unit, polymer, or experimental design variables. Extrapolative predictions can extend the model into new material spaces, provided prediction uncertainties are considered [6]. The multiscale designing of soft materials necessitates the use of a variety of computational methods, from continuum modelling to quantum chemistry and molecular dynamics. Modern optimization algorithms and ML techniques have recently become more prevalent, accelerating the prediction of material properties and encouraging the development of hybrid ML/molecular modelling techniques that can offer physical insights not possible with purely physics-based models. The objective in soft materials design is to use ML approaches for the design of specific chemical or morphological features [7].

AI and ML can significantly accelerate the discovery, design, and characterization of new polymers. As predictive accuracy and robustness increase due to increasingly sophisticated computations and newer ML algorithms, the time and cost associated with material development will decrease. Several new kinds of materials have been discovered through materials informatics. However, there are significant obstacles to be overcome before sophisticated AI/ML strategies for developing novel materials may be used to full advantage in polymer research [1, 8–10]. Polymers are incredibly diverse in terms of their chemical properties and applications. Because the chemical space of polymers is nearly unlimited, it is difficult to locate the polymer with the optimum combination of properties for a specific application. Uncertainties arising from the polydispersity of polymers

presents further challenges. Development of suitable representations of the hierarchical structures of polymers at various scales is a prerequisite for informatics techniques to be able to effectively handle the full complexity of polymeric materials and generate robust and predictive models for polymer characteristics [11–13]. Surrogate models built using machine learning methods on existing property data could help to alleviate these challenges [14].

Machine learning and deep learning algorithms have revolutionized the field of polymer informatics and the design of new polymeric materials. These algorithms empower us to extract meaningful insights, predict and optimize material properties. Some powerful algorithms widely used in polymer informatics include:

1. Linear Regression is often used in polymer informatics to model linear relationships between input features and material properties. It is the simplest kind of model for understanding correlations between variables. According to the principle of Occam's razor, it is desirable that models be as simple and interpretable as possible.
2. Principal Component Analysis (PCA) is a dimensionality reduction technique commonly employed to simplify and visualize complex datasets. It works by transforming the data and projecting them onto a lower-dimensional principal component space, while preserving the variance. It is useful for data visualization and feature extraction.
3. K-Means and Hierarchical Clustering are unsupervised learning techniques used for grouping similar polymers based on their properties. They help in identifying polymer classes or clusters.
4. Decision Trees and their ensemble counterpart, Random Forests, can handle both regression and classification tasks. They are powerful at feature selection and material classification, and are interpretable, which is important for understanding model decisions.
5. Support Vector Machines (SVM) are useful for both regression and classification tasks in polymer informatics. They work well with high-dimensional data, are good at avoiding overfitting and can find complex decision boundaries.
6. Artificial Neural Networks (ANN) represent a broad family of ML techniques, broadly based on the organization and functioning of the brain. They consist of nodes or neurons, organized in layers, with connections (synapses) between them. Learning is accomplished by adjusting the synaptic weights to minimize an error function. ANNs are widely used in polymer informatics due to their ability to model intricate relationships in data.
7. Convolutional neural networks (CNNs) are deep learning ANN algorithms patterned after the architecture of the visual cortex of the brain. The convolutional layers act as filters, performing feature identification in a hierarchical manner. CNNs are most often used in image analysis, feature extraction from complex polymer structures, and to classify polymers based on their structures.
8. Generative Adversarial Networks (GANs) consist of two ANNs: a generator network that creates new structures, and a discriminator network that

discriminates between authentic and generated structures. The two networks compete against each other, and in the process, get better at their respective tasks. GANs have shown promise in the design of new polymers with specific properties.

9. Recurrent Neural Networks (RNNs) are a deep learning ANN architecture where the output from a given node can also be one of the inputs to the same node. RNNs are used to handle sequential data and predict the properties of polymers based on their structures.
10. Language models have recently become very popular. Such models employ Transformers that process sequential input data, such as the stream of characters representing natural language text or polymer strings, applying a self-attention mechanism that provides context for any position by determining at each step which other parts of the string are most relevant.
11. Transfer Learning allows models pre-trained on a large dataset to transfer the learned knowledge to a more complex polymer dataset, with fewer data, resulting in a faster learning process with more accurate predictions.

There are, however, several challenges that need to be addressed to fully realize the potential of deep learning in polymer informatics:

1. *Lack of data*: Paucity of data on polymer properties, as well as the diversity and structural complexity of polymers make it difficult to train robust and accurate deep learning models.
2. *Data quality*: The data used to train deep learning models must be accurate, reliable, and representative of the polymers being studied. Ensuring high quality data requires careful data curation and organization.
3. *Interpretability*: Deep learning models are often difficult to interpret.
4. *Computational resources*: Deep learning models require significant computational resources to train and optimize.

In the next section, we summarize key application areas of AI/ML in polymer informatics, and then in following sections, we review some existing polymer databases, and present a few recent case studies that highlight applications of AI techniques in the polymer design process. Our objective is not to conduct a comprehensive survey of work in the field, but to present a sampling that gives the reader a flavour of the scope and diversity of recent applications, as well as the state of the art and specific issues that make polymer informatics challenging.

2 Applications Areas

1. *Structure-Property Relationships and Polymer Property Prediction*: AI/ML techniques can enable accurate and efficient prediction of polymer properties, by discovering complex relationships between polymer structure, composition, processing conditions, and properties. They can analyse and extract valuable

information from large datasets, identifying correlations, trends, and subtle patterns that are challenging for humans to discern. This deeper understanding can guide the design of polymers with improved performance, stability, and other targeted properties, such as mechanical strength, thermal stability, or electrical conductivity, thereby empowering researchers to rapidly screen and identify promising polymer candidates with desired characteristics.

2. *High-Throughput Screening*: AI/ML enables high-throughput screening of polymer properties by rapidly analysing large volumes of experimental or simulation data. This approach allows researchers to efficiently explore the parameter space and identify polymer structure-property relationships, accelerating the discovery of promising materials. High-throughput screening, coupled with AI/ML algorithms, facilitates the identification of novel polymer compositions, processing conditions, or additives that optimize desired properties.
3. *Material Design and Optimization*: AI/ML can aid in the rational design and optimization of polymers with specific functionalities. By leveraging large datasets of experimental or computational data, and utilizing ML models, researchers can explore large chemical spaces, predict the properties of hypothetical polymer structures, and guide the synthesis or modification of polymers to achieve desired performance. This reduces the need for extensive trial-and-error experimentation and accelerates the development of novel materials. Generative algorithms can also be used to explore entirely new regions of polymer chemical space.
4. *Data Integration and Knowledge Discovery*: AI/ML techniques can integrate data from diverse sources, including experimental measurements, computational simulations, literature databases, and even textual information. By fusing multiple datasets, researchers can gain a more comprehensive understanding of polymer properties, identify knowledge gaps, and make informed decisions. This integrated approach enhances data quality, facilitates knowledge sharing, and promotes collaboration within the polymer informatics community.
5. *Database Development*: AI/ML plays a vital role in developing and maintaining materials informatics databases specific to polymers. ML algorithms can automate the extraction, organization, and analysis of data from various sources, creating a valuable resource for researchers. These databases provide convenient access to relevant polymer information, facilitate data-driven research, and foster the development of new ML models and algorithms. In the next section, we will review some polymer databases and the ways in which they have been used in polymer informatics.

3 Databases and Tools for Polymer Informatics

As large amounts of data generated by contemporary experimental techniques and first-principles computations become widely accessible [15, 16], ML technologies have created new possibilities for rational design of new polymeric systems. Many

databases for polymer informatics are now available [17], but there are still fewer open databases (Table 2) for polymers than for other classes of materials. To alleviate the problem of lack of large FAIR (Findable, Accessible, Interoperable, Reusable) databases, Tchoua et al. [18] created pipelines to harvest the enormous amounts of important experimental polymer data buried in polymer databases. The biggest Flory-Huggins chi parameter database uses crowdsourcing as a classifier to recognise potential publications that might reduce the workload of reviewing studies. In order to obtain glass transition temperatures with the least amount of human input, a natural language processing (NLP) algorithm is used in conjunction with specifically created software modules. Over 250 glass transition temperatures can be extracted using this method. All the results are available on the Polymers Property Predictor.

It is difficult for a machine to recognise or for humans to curate the names of the polymers, since they are referred to in the literature by several names, including common names, sample names, labels, etc. This non-uniformity in the representation of polymer names results in irregular polymer indexing, which is a significant barrier to the widespread use of datasets linked to polymers. This also limits the extensive application of materials informatics in polymer science. ChemProps [19] is a multi-algorithm-based mapping technique to address the polymer indexing problem. The Representational State Transfer (RESTful) API enables simple system integration and lightweight data exchange. Provided with input by the user, multiple NLP algorithms automatically generate scores for prospective chemical names, and vote on the final match. The weight assigned to each algorithm is calibrated to minimize the score difference between the potential and actual chemical names. The training set contained 160 data points, while the test set contained 54 test data points; overfitting was avoided by performing tenfold validation. The resulting set of weights achieved a test set accuracy of 100%. Reducing duplicate entries and searching by SMILES strings utilising ChemProps as a name-to-SMILES converter can help achieve greater accuracy. ChemProps' easy design makes it a convenient tool for automatically populating databases with polymer properties.

Physical properties of polymers listed in the ATHAS data bank [20] include glass transition temperature (T_g), differential heat capacity at T_g for amorphous polymers, equilibrium melting point and heat of fusion at 100% crystallinity, etc.

Accurate representation of polymers is essential for using ML techniques to quantify polymer structure-property relationships. Application of ML to polymers is limited in part due to a lack of benchmark datasets. For characterization of polymer structures, Lin et al. [21] proposed the PolyDAT scheme, which uses a multi-species format that is concurrent with experimental polymer data unlike other schemes. Examining relationships between polymer representations and their characteristics like density, melting point, and glass transition temperature allows one to assess the fidelity of various polymer representations.

The benchmark PI1M (one million polymers for polymer informatics) database [22] was created by using ~12,000 polymers from PoLyInfo [23] to train a generative model. Monomers were represented as SMILES strings (with * to indicate polymerization points), and a recurrent neural network (RNN; Fig. 2) was used to

Table 2 Online databases for polymer informatics

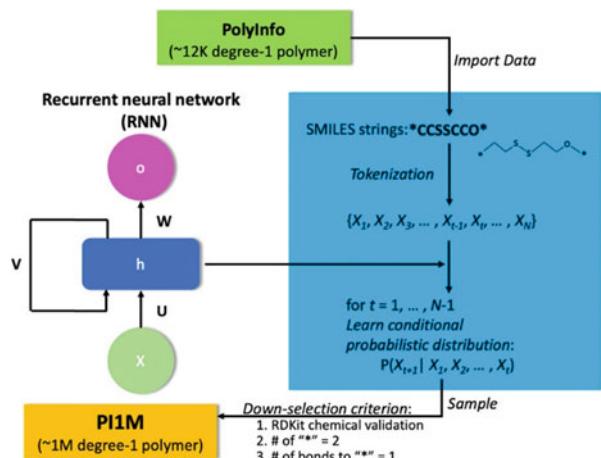
Database name and URL	Number of polymers	Class	Available properties	Tools
PolymerDB www.polymerdb.org	10,000+	All polymer classes	Structure, composition, molecular weight, glass transition temperature, melting point, etc.	Search, visualization, property prediction
PoLyInfo www.PoLyInfo.org	16,600+ homopolymers	Synthetic polymers	Mechanical properties, thermal stability, chemical resistance, etc.	Search, data analysis, property comparison
BioPolymersDB www.biopolymersdb.org	2000+	Biopolymers (proteins, DNA)	Sequences, structures, secondary structure predictions, functional annotations	Sequence search, structure visualization, comparative analysis
PolySynthDB www.polysynthdb.org	1500+	Synthetic polymers	Monomer synthesis methods, polymerization conditions, polymer structure	Synthetic route search, reaction optimization
NanoPolyDB www.nanopolydb.org	500+	Nanocomposite polymers	Nanoparticle types, dispersion methods, mechanical properties, electrical conductivity	Nanoparticle search, property modeling, performance evaluation
PI1M www.github.PI1M.com	~1,000,000	AI-generated all polymer classes	Regression tasks after generating polymer embeddings	Benchmark database for polymer informatics
Protein Data Bank (PDB) www.rcsb.org	84,535 unique protein sequences	Proteins, protein-DNA and protein-ligand complexes	FASTA sequences, 3D structures (coordinates), from X-ray crystallography, NMR spectroscopy, and 3D electron microscopy	Sequence, structure and chemical similarity search, visualization tools, browse by annotations, structure alignment
Polymer Genome www.polymergenome.org	13,388	All polymer classes—Homopolymers and Copolymers	bandgap, dielectric constant, refractive index, atomization energy, glass	Polymer property prediction and design

(continued)

Table 2 (continued)

Database name and URL	Number of polymers	Class	Available properties	Tools
			transition temperature Tg, solubility parameter, and density	
Polymer Property Predictor pppdb.uchicago.edu	1500+	All types of polymers	Flory–Huggins χ parameters and glass transition temperatures of polymers	Polymer property prediction, Glass Transition Temperature, Structure Factor, and Flory–Huggins parameters
NanoMine materialsmine.org/nm/#/	500+	Nanocomposites	Chemical Properties	Physio-chemical properties
ACD/Labs NMR Database www.acdlabs.com/products/dbs/nmr_db	1000+	Biopolymers	Polymer NMR spectra	Chemical structure properties
NIST Synthetic Polymer MALDI Recipes Database maldi.nist.gov	1250	Synthetic polymers	MALDI mass spectra	Chemical structure properties, ionic energetic properties, vibrational and electronic energies data and molecular weight
Materials Project materialsproject.org	~1,000,000 (>120,600 inorganic compounds, 35,000 molecules as well as 530,000 nanoporous materials)	All polymers	Computed properties	Database for polymer informatics
CROW Polymer Properties Database polymerdatabase.com		Thermoplastics and Thermosets as well as rubbers	Physical and Chemical Data	Thermophysical properties as a database
MATWEB Material Property Data www.matweb.com	170,000	Thermoplastic and thermoset polymers	Material properties and composition	Physical and chemical properties, polymer film data, and composition

Fig. 2 RNN. (Reprinted with permission from Ref. [22] © 2020, American Chemical Society)



learn the polymer embeddings (patterns and features present in the data, from a representation of the structures in a continuous, lower-dimensional space). An RNN is an artificial neural network (ANN) where the output of one node can also be fed back to the same node, in addition to being fed forward to the next layer, enabling the network to learn sequential data, and predict future output given present input and memory of the past. The model was employed to construct regression models for properties such as density, glass transition temperature, melting temperature, and dielectric constant, and then trained to generate novel polymer structures matching the characteristics of the training set [24]. Comparison of the polymer embeddings show that the two databases encompass a similar polymer chemical space, while PI1M fills in sparse regions of PoLyInfo.

A web-based ML system called Polymer Genome [25, 26] provides rapid predictions for several families of polymer properties, including physical, thermodynamic, electronic, dielectric, optical, thermal, and mechanical properties, and solubility in common solvents. A data set of 854 organic polymers whose characteristics like glass transition temperature, bandgap, dielectric constant, refractive index, atomization energy, Hildebrand solubility parameter and density, were known either experimentally from the literature or computationally from DFT, were employed with a fingerprinting system that represents atomistic to morphological structural properties of polymers. ML models were developed mapping the fingerprints to the target attributes. Since various properties depend on features at different length scales, the ML models were constructed using a collection of features optimised for each distinct property. The prediction models employed in Polymer Genome are Gaussian process regression (GPR) and ANN. Polymer property prediction with Polymer Genome has been discussed by Tran et al. [27].

RadonPy [28] is an open-source Python toolbox for automated all-atom classical molecular dynamics (MD) simulations. The molecular modelling, MD calculations, and property calculations for a particular polymer repeat unit can all be fully automated. Calculated values for 15 different characteristics, such as thermal

conductivity, density, specific heat capacity, thermal expansion coefficients, and refractive index, are provided for more than 1000 different amorphous polymers. Systematically comparing experimental results from PoLyInfo to the computed characteristics allows for validation. Polymer informatics can be facilitated by generation of large volumes of computational property data using RadonPy. Using RadonPy, Hayashi et al. [28] showed that eight amorphous polymers, including six with previously unknown thermal conductivities, had extraordinarily high thermal conductivities (>0.4 W/mK). Hydrogen bonding units or stiff backbones were discovered to be highly concentrated in these polymers. The mechanisms underlying high thermal conductivity of amorphous polymers were identified to be heat transfer via hydrogen bonds and dipole-dipole interactions between the polymer chains.

4 Case Studies

AI, ML and deep learning approaches provide enormous benefits and possibilities for predicting crucial mechanical characteristics of polymers. Structure representations, features/descriptors, and ML algorithms make up the three main parts of a ML model. In this section, we will discuss several case studies from the recent literature that provide illustrative examples of applications of modern AI/ML techniques in polymer informatics. Table 3 summarizes the properties studied, ML methods used, and performance metrics of the various models discussed in this section.

Prediction of glass transition temperatures and dielectric properties: An early application of informatics methods to polymer design involved the prediction of glass transition temperatures (T_g) and dielectric properties of polymers [43]. The authors generated predictive Partial Least Square (PLS) regression and Kernel Partial Least Squares (KPLS) regression models relating polymer repeat unit structure to T_g , using the Bicerano dataset [44, 45] consisting of 320 polymers. Using QSPR modeling to supplement and leverage density functional theory (DFT) computations, they optimized the polymer skeleton and functional group substitutions through several rounds of validated QSPR modeling and first-principles DFT computations, to design polymers with specific dielectric properties. Employing DFT computations and informatics, they identified promising chemical motifs leading to both large dielectric constant and large band gap [29]. This was employed to predict dielectric permittivities and bandgaps of nearly 29,365 diverse polymers, and to design novel polymers of Group IV halides. A new set of features, infinite chain descriptors (ICDs), was also introduced [30] and used in QSPR models to predict the dielectric constant, band gap, dielectric loss tangent, and glass transition temperatures. All models showed good predictive performance: the SVM model showing $R^2 = 0.97$ for the training set and 0.95 for the test set.

Virtual screening of polymer chemical space with evolutionary algorithms: Venkatraman and Alsberg [31] constructed partial least squares regression and random forest classification models for refractive indices, densities, glass transition and thermal decomposition temperatures and solubilities of polymers, using

Table 3 ML methods employed, properties modelled, and performance metrics of the models discussed in this work

References	Methods	Properties	Results
[11]	Data-driven finite element analysis and multi-task CNNs	Polymer nanocomposites	Accuracy improves by as much as 45.2% for glass modulus, by 34.2%, for rubbery modulus and by 19.7% for $\tan \delta$ peak
[12]	Sparse flexible clustered multi-task learning (FCMTL)	Synthetic data and data generated during blow molding	$R^2 = 0.716$ (LASSO-FCMTL) $R^2 = 0.736$ (JS-FCMTL)
[14]	Multi-task learning	Mechanical, thermodynamic, electronic, optical, and dielectric properties, thermal stability, and solubility	Multi-task learning Neural Networks outperform other models with $R^2 = 0.79$ (thermal properties)
[25]	Gaussian process regression (GPR)	Glass transition temperature, bandgap, dielectric constant, refractive index, atomization energy, and density	Training $R^2 = 0.92$, Test $R^2 = 0.90$, Test RMSE = 24.2 K (Tg)
[29]	Additive group contributions validated by DFT computations	Dielectric permittivity and bandgap of 29,365 polyethylene-related polymers	Motifs based on fluorides of Si, Ge, and Sn were found to enhance dielectric constant
[30]	Support vector machines (SVM) model with Infinite Chain Descriptors (ICD)	Dielectric constant, band gap, dielectric loss tangent, and glass transition temperature	Training $R^2 = 0.97$, Training RMSE = 18.05, Test $R^2 = 0.95$, Test RMSE = 23.32 (dielectric permittivity). Cross-validation $R^2 = 0.88$
[31]	Partial Least Squares Regression (PLSR) and random forest classification	Refractive index, density, solubility, glass transition and thermal decomposition temperatures	Test $R^2 = 0.80$ (PLSR for refractive index)
[32]	Kernel Partial Least Squares Regression (KPLS)	Ring-opening polymerization (ROP) Peak Temperature and Enthalpy of 194 polybenzoxazines	Training $R^2 = 0.85$ and RMSE = 20.89 kJ/mol, Test $R^2 = 0.54$ and RMSE = 27.08 kJ/mol (for Enthalpy; Training $R^2 = 0.85$ and RMSE = 13.87 °C, Test $R^2 = 0.66$ and RMSE = 14.94 °C (for Peak Temperature))

(continued)

Table 3 (continued)

References	Methods	Properties	Results
[33]	Deep neural network (DNN) classifier	Solubility	Training accuracy = 99.6% 98.1% for the validation set Test accuracy = 93.8%. AUC = 0.98
[34]	Gradient boosting model	Ionic conductivity for Lithium-ion conducting polymers	Training R ² = 0.90 Test R ² = 0.81
[35]	Transfer learning, with random forest model trained on 28 data points for λ	Thermal conductivity	MAE = 0.0204 W/mK on pre-trained models. MAE = 0.0327 W/mK for RF model
[36]	ANN model with four hidden layers	Young's modulus of polymer-carbon nanotube composites	Training R ² = 0.986, Test R ² = 0.978 Sum of Squared Errors (SSE) = 0.0018 for training and SSE = 0.0021 for testing. Akaike Information Criterion (AIC) is – 181.14. Average Discrepancy Ratio (ADR) = 1.25
[37]	Feed-forward neural networks, CNN, GCNN, RNN, random forests, SVM, LASSO regression and GPR models	Glass transition temperature	R ² = 0.90, MAE = 28.13, and RMSE = 34.28 (MD simulation with K-means clustering)
[38]	ANN, Lasso linear regression and random forest regression models	Skeletal heat capacity (heat capacity at constant volume based on skeletal vibrations) for polymers from the ATHAS data bank	R ² = 0.9997 on ATHAS data bank. R ² for Θ1 and Θ3 (characteristic parameters calculated from Tarasov equation) are 0.9925 and 0.7933, respectively
[39]	Multitask graph neural networks (polyGNN) with an encoder, a message passing block, and an estimator	36 properties such as dielectric constants, Hildebrand solubility, physical, electronic, optical, thermal, thermodynamic, and mechanical properties	RMSE = 31.7 ± 1.5 K for Tg
[40]	Software platform for accelerating polymer discovery through informatics and AI development	Polymers	R ² values range from 0.68 to 0.78

(continued)

Table 3 (continued)

References	Methods	Properties	Results
[41]	TransPolymer (Transformer-based language model)	Electrical conductivity, band gap, electron affinity, ionization energy, crystallization tendency, dielectric constant, refractive index, and OPV power conversion efficiency	RMSE = 0.67, $R^2 = 0.69$ (electrical conductivity for PE-I dataset)
[42]	Multi-task deep learning with polyBERT (Deep Bidirectional Transformer language model)	29 polymer properties of 100 million hypothetical polymers	High R^2 values for all properties $R^2 = 0.92$ (for Tg) $R^2 = 0.81$ (for Tm)

geometrical and electronic (molecular orbital-based) descriptors for the monomers, to identify novel polymers with several desirable properties. They used an evolutionary algorithm to explore polymer chemical space, and to rapidly screen virtual polymers based on properties such as refractive index, glass transition and thermal decomposition temperatures, and solubility in common solvents. The most promising monomers were validated using DFT calculations.

Prediction of polymerization temperature and enthalpy: Samanta studied a dataset of 194 polybenzoxazines [46], which are a class of thermosetting polymers formed by ring-opening polymerization (ROP) of substituted Benzoxazines. Due to their good mechanical strength, high thermal stability, flame retardance, good chemical and water resistance, near-zero shrinkage during polymerization, low dielectric properties and lower surface free energy, they have found wide applications in both industry and domestic use as adhesives, flame resistant polymers, cathode material in batteries, coatings, aerospace materials, CO₂ adsorbents, as well as in 3D printing and electronics. Samanta [32] modelled thermal properties such as the Peak Temperature and Enthalpy of ROP using RECON [47] and MOE [48] descriptors. After feature selection 39 descriptors were used for modelling enthalpy while 55 were used for modelling the peak temperature. Predictive but overfit models were obtained using the Kernel Partial Least Squares (KPLS) method, with training r^2 of 0.85 and RMSE of 20.89 kJ/mol, and test r^2 of 0.54 and RMSE of 27.08 kJ/mol set for the enthalpy; training r^2 of 0.85 and RMSE of 13.87 °C, and test r^2 of 0.66 and RMSE of 14.94 °C for the Peak Temperature. The higher performance for the training set compared to the test set (Fig. 3) is a characteristic of models exhibiting overfitting.

Design of polymers with high Tg: Kim et al. [25] used the data set of 854 organic polymers in Polymer Genome, with atomic level and morphological QSPR descriptors, in a GPR model to design polymers with high Tg. Dimension reduction by recursive feature elimination further improved performance of the ML models for Tg to $R^2 = 0.92$ for the training set, and $R^2 = 0.90$, RMSE = 24.2 K (test set).

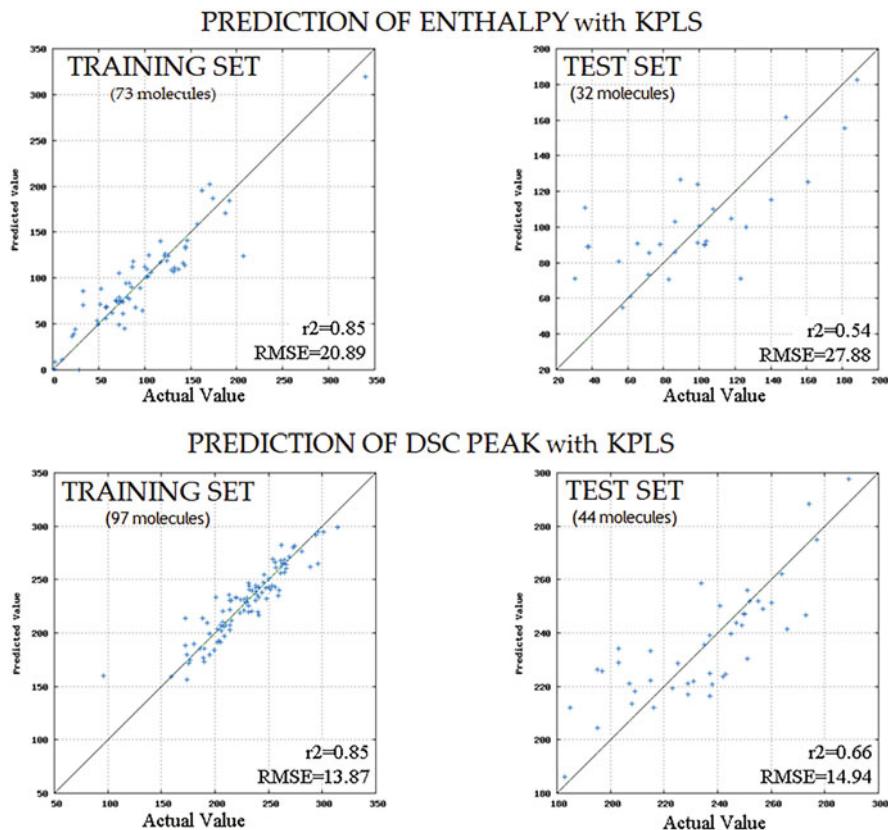


Fig. 3 QSPR models for polybenzoxazines

Predictive models were also obtained for bandgap, dielectric constant, refractive index, atomization energy, and density.

Solubility prediction: Polymer solubility is crucial for numerous applications such as drug delivery, membrane research, microlithography, and plastic recycling. Selecting the proper solvent for a novel polymer is challenging; solubility parameters and other heuristic techniques are of limited applicability. For instance, the popular Hildebrand parameter performs reasonably well in predicting solubility for non-hydrogen bonding nonpolar systems, but completely fails for hydrogen bonded polar systems. Chandrasekaran et al. [33] used a deep neural network (DNN) classifier, trained on a data set of 4595 polymers, represented as SMILES strings and encoded as numerical fingerprints, and 24 widely used solvents, for choosing the appropriate solvents (Fig. 4). A DNN is an artificial neural network with many “hidden” layers of neurons between the input layer and the output layer, where each neuron computes some function of the weighted sum of its inputs; the training process consists in adjusting the weights so as to minimize the error in the final

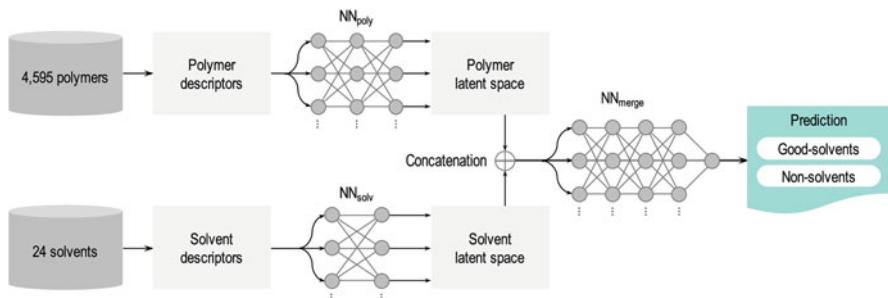


Fig. 4 Deep neural network architecture for prediction of polymer solvents. (Reprinted with permission from Ref. [33] © 2020, American Chemical Society)

output(s). They converted the high-dimensional fingerprint representation into a compact, low-dimensional chemically appropriate latent space representation of solvents and polymers. When these low-dimensional representations are displayed on a principal components plot, nonpolar, polar-aprotic, and polar-protic polymers clump into distinct clusters. Fivefold cross-validation was performed on the training set, with a separate 10% hold-out set. The model, incorporated into Polymer Genome, had a classification accuracy of over 93% on the holdout set and AUC = 0.98, outperforming other methods for evaluating room-temperature polymer–solvent compatibility.

Prediction of ionic conductivity: Hatakeyama-Sato et al. [34] modelled a data set of 240 different lithium-ion conducting solid polymer electrolytes and composites from the published literature using gradient boosting techniques and Mordred descriptors. Accurate predictions were obtained for the ionic conductivity: $R^2 = 0.90$ (training set) and 0.81 (test set). Electronegativity and polarity of the monomer were found to be the most important descriptors determining conductivity. Poly glycidyl ether derivatives were predicted to have high electrical conductivity, and confirmed experimentally after synthesis. Over 15,000 de novo-designed candidate polymers were also screened computationally to select those with high conductivity.

Transfer learning to design polymers with high thermal conductivity: Using a transfer learning approach and a very small training dataset of 28 polymers, Wu et al. [35] were able to design, synthesize and polymerize novel polymers with excellent thermal conductivity: between 0.18 and 0.41 W/mK, which are equivalent to those of modern thermoplastics. For this they exploited proxy properties related to thermal conductivity, such as glass transition temperature, melting temperature and density, and pre-trained probabilistic language models on 14,423 unique homopolymers from the PoLyInfo and QM9 data sets (Fig. 5). The polymers were represented as SMILES strings, and a Bayesian approach was used to invert the forward prediction model. High-probability regions in the inverse model were randomly sampled using a sequential Monte Carlo algorithm to identify promising monomers with the desired

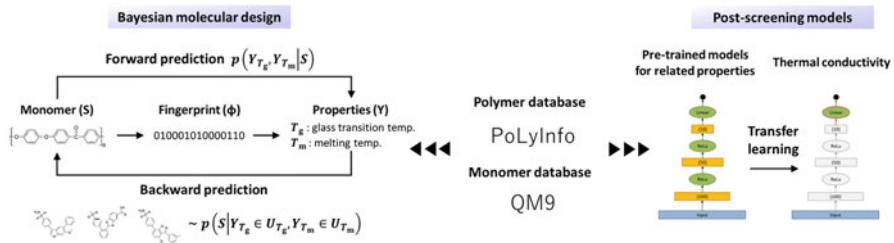


Fig. 5 De novo design of new polymers with high thermal conductivity using transfer learning. (Adapted from Ref. [35] under Creative Commons CC BY license)

property. Then the pre-trained models were fine-tuned with neural network models on the limited thermal conductivity data.

Novel Feature Selection to handle polydispersity: The uncertainty arising from the polydispersity of polymeric materials makes QSPR predictions challenging. The key issue is to effectively depict polymeric materials computationally while capturing the polydispersity data from the molecular weight distribution curve. Feature selection is a crucial step in QSPR modelling of physicochemical properties in cheminformatics. Working with an artificial data set, Cravero et al. [49] designed a novel feature selection (FS) technique, Feature Selection for Random Variables with Discrete Distribution (FS4RVDD), to handle polydisperse data. FS4RVDD was designed to handle random variables with discrete distributions as well as higher dimensional datasets. The authors assessed the scalability and robustness of the new feature selection algorithm, compared to those of traditional approaches, and found that it outperformed traditional FS techniques in classification performance under various scenarios, coping with the uncertainty polydispersity presents in man-made polymers.

ANN models: Ho et al. [36] developed an ANN model, trained on a dataset of 282 different kinds of polymer-carbon-nanotube (CNT) composites, for predicting the Young's modulus of the composites. The optimal ANN architecture comprised of four hidden layers. The model performs well, with a correlation coefficient of 0.986 for the training set and 0.978 for the test set, outperforming other ML methods such as linear regression and second order regression. The CNT weight fraction was found to be the most significant feature affecting the Young's modulus of polymer nanocomposites.

Tao et al. [37] conducted a comprehensive benchmark analysis by examining 79 different ML models, including feed-forward neural networks (FFNN), convolutional neural networks (CNN), graph convolutional neural networks (GCNN), recurrent neural networks (RNN), random forests (RF), support vector machines (SVM), LASSO regression and Gaussian process regression (GPR), trained on a data set of 566 polymers, simulated with MD, and their T_g values. The monomer, repeat unit, and oligomer were used to compute RDKit descriptors, Morgan fingerprints (MF), Morgan fingerprint including substructures' frequency of occurrence (MFF), molecular embeddings (ME) and molecular graphs (MG).

Models were evaluated with reference to their capacity for generalisation on unlabelled data, and sensitivity to polymer structure and molecular weight. The ranking of the feature representations was found to be MFf (best) > SMILES > RDKit descriptors > ME > MG > MF > 2D Images (worst), and on the basis of average performance, the Tg prediction methods were ranked as RF (best) > CNN > RNN > GPR > FFNN > GCNN > LASSO > SVM > 2D CNN (worst).

Ishikiriyama [38] constructed ANN regression models for the heat capacities of polymers listed in the ATHAS database [21], by developing separate QSPRs between ECFP4 extended-connectivity structural fingerprints of polymer repeat units and each physical characteristic in the ATHAS data pool. ANN, Lasso linear regression and random forest regression models were used, using the data in the ATHAS database for training. Contour plots of root mean square error for each physical attribute were used to establish the best hidden-layer architectures for the ANNs. The predicted and observed values in the ANN models showed a strong correlation for the training set, indicating that the physical attributes could be predicted using only the ECFP4 fingerprints of the polymer repeat units. The models were tested by predicting the physical properties of poly(p-dioxanone), which was not listed in the ATHAS data bank. The predicted low temperature properties showed large errors, but the predictions for other properties coincided with the observed values from previous studies to within 25%.

Design of polymers with desirable stress-strain characteristics: Aoyagi [50] used ANN and coarse-grained molecular dynamics (CGMD) simulations to design block copolymer architectures with different combinations of chain lengths, volume fractions, and asymmetricities, leading to polymers with desirable elastic properties. The stress-strain curves of linear block copolymers with different chain lengths, block volume fractions, and asymmetricities were obtained from CGMD simulations. A fully connected 3-layer ANN was employed to learn regression relations between the stress-strain curves and the polymer structures. The trained ANN was then used in Bayesian optimization for inverse design to produce polymer structures with desirable stress-strain characteristics. The stress-strain curve from the improved polymer structure's CGMD simulations matched the curve predicted by the ANN. Thus using ANN and CGMD simulations to build polymer structures with desired attributes is a potentially useful strategy.

Multi-task learning: Multi-task learning strategies can be employed to jointly predict and optimize multiple polymer properties that are inter-correlated, and thus leverage the power of models where training data for a given property might be limited. By training a single model to handle multiple tasks, such as predicting mechanical, thermodynamic, electronic, optical, and dielectric properties, thermal stability, and solubility of polymers, the model can leverage the common underlying features and relationships between these properties to enhance predictive accuracy. Kuenneth et al. [14] represented polymer structures by SMILES strings (with * to indicate polymerization points) and employed polymer fingerprints to capture key features of polymers at different length scales. Neural network models were then trained on 36 distinct properties of more than 13,000 polymers in Polymer Genome [26, 27]. This multi-task technique was found to be more precise, effective, scalable,

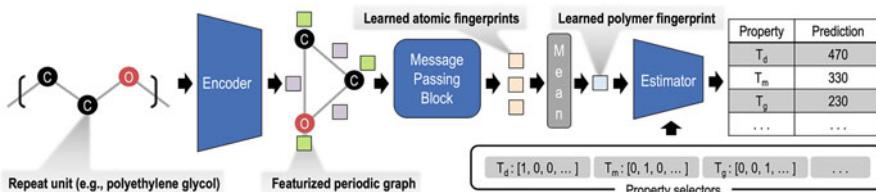


Fig. 6 PolyGNN architecture. (Reproduced from Ref. [39] under Creative Commons CC BY license)

and transferrable than traditional single-task learning models, showing that such strategies can be effective in the rational design of application-specific polymers.

Graph neural network models: Gurnani et al. [39] developed a multitask graph neural network (GNN) for polymers, and showed that it is less expensive and more practical to directly machine-learn from a polymer repeat unit than it is to extract complex human-designed features. A GNN consists of a set of nodes connected by links, where each link between a pair of nodes is associated with a weight that determines the strength of their connection. These nodes and links carry attributes and are organized into different layers. The GNN attempts to find an appropriate set of weights that minimizes the difference between the calculated outputs and the values for the properties being modelled, thereby identifying trends and patterns in the data. The architecture of PolyGNN consisted of three modules (Fig. 6): an encoder to process the repeat unit, a message passing block for fingerprinting, and an estimator that co-learns multiple properties. Using graph neural networks and multitask learning on 13,388 polymers and 36 properties in Polymer Genome [26, 27] (such as dielectric constants, Hildebrand solubility, physical, electronic, optical, thermal, thermodynamic, and mechanical properties) speeded up feature extraction by one to two orders of magnitude without sacrificing model accuracy for many, but not all, polymer property prediction tasks. This opens up the prospects of screening large polymer libraries at scale. However, the features learned by the GNNs are not readily interpretable.

Park et al. [40] designed novel ROP catalysts using a simple graph representation for polymers as well as different types of experiments and data. The graph representation was also employed to develop a generative model for block co-polymers. For this they used a regression transformer, pretrained on a ROP reaction dataset.

Transformer models: Transformer models that include attention processes [51] have recently been successfully applied across a range of NLP tasks. These are neural networks designed to process sequential input such as the stream of characters representing natural language text or SMILES strings; a self-attention mechanism applied at each step provides context by examining the input string and deciding which other parts of the string are important. TransPolymer [41] is a transformer-based language model for predicting thermal and mechanical properties of polymers. Polymers were represented by SMILES strings of their repeat units, and by structural descriptors, such as degree of polymerization, polydispersity, and chain

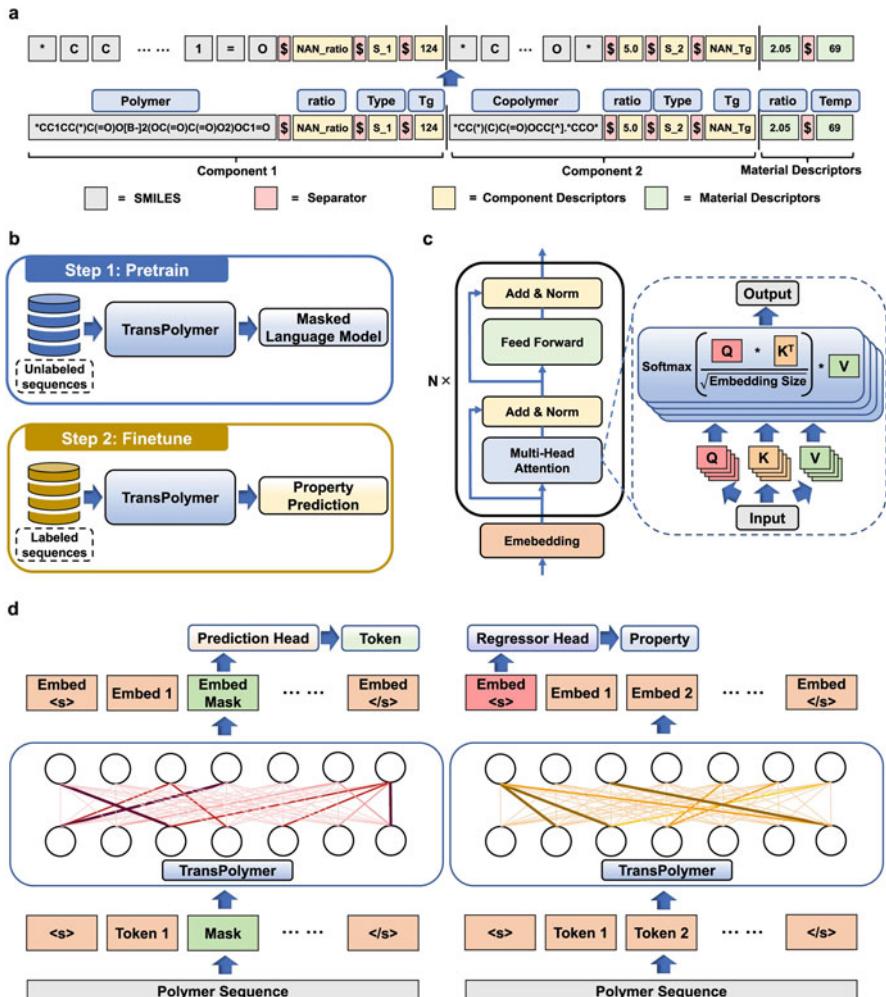


Fig. 7 Transpolymer architecture: Illustration of the pretraining (left) and finetuning (right) phases. (Adapted from Ref. [41] under Creative Commons CC BY license)

conformation, and then tokenized. The model was pretrained on the large, unlabeled PI1M dataset and then fine-tuned on downstream datasets specific to certain polymer properties, such as electrical conductivity, band gap, electron affinity, ionization energy, crystallization tendency, dielectric constant, refractive index, and OPV power conversion efficiency (Fig. 7). This approach enabled TransPolymer to learn representations directly from polymer sequences and to generate accurate predictions. On most tasks, TransPolymer significantly outperformed other models, such as random forest with ECFP fingerprints, an LSTM language model, and an unpretrained TransPolymer trained with supervised learning. These results further

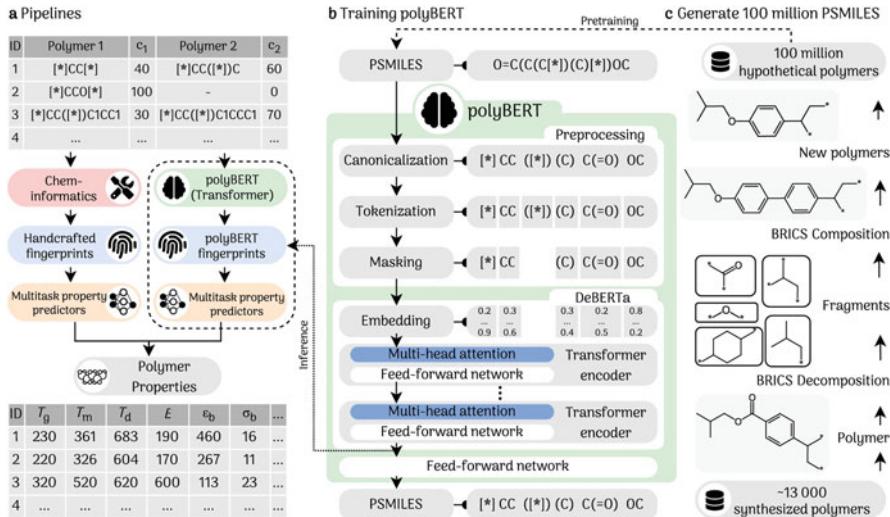


Fig. 8 Polymer informatics with polyBERT. (Reproduced from Ref. [42] under Creative Commons CC BY license)

emphasise the substantial benefits of pretraining on enormous quantities of unlabelled data in representation learning and the efficacy of the attention mechanism for understanding polymer sequences. TransPolymer is thus a workable computational tool for rational design of novel polymers. It is hoped that TransPolymer can serve as a universal pretrained model for robust prediction pf polymer properties, applicable to various properties using fine-tuning.

polyBERT [42] is another transformer-based chemical language model, adapted from Devlin's Deep Bidirectional Transformers (BERT) [52], that has been applied to polymer informatics, treating polymer structures as a chemical language. 100 million polymer SMILES strings, generated by combining chemical fragments extracted from 13,766 synthesized polymers, were used to train a language model. During the training process, polyBERT learns to translate the polymer SMILES strings to numerical representations. These were then mapped using multi-task deep learning onto 29 polymer properties, harnessesing hidden correlations in the data (Fig. 8). polyBERT shows high R² values for all properties, and outperforms other polymer property prediction models based on hand-crafted (Polymer Genome) descriptors—it is faster by two orders of magnitude, while still preserving accuracy.

5 Conclusion

Accurate, robust and efficient polymer property prediction is essential for polymer design and development. Time-consuming and expensive experimental tests or simulations have traditionally been required for assessing the properties of polymers.

Polymer informatics highlights the transformative impact of data-driven approaches in this emerging area. By integrating techniques such as machine learning, statistical modelling, data mining, and generative deep learning, researchers have been able to extract meaningful insights from large polymer datasets. These approaches have facilitated the discovery of hidden patterns, the establishment of structure–property relationships, and the accurate prediction of polymer properties. The application of graph-based generative deep learning models and transformer-based language models in polymer informatics has not only accelerated research and development, but has also enabled the exploration of new polymer chemical space and the discovery of novel polymers with desired properties. Collaboration and knowledge sharing within the scientific community have been enhanced through the availability of publicly accessible databases, collaborative platforms, and open-source informatics tools. Despite many challenges, AI and ML hold great promise for driving innovation in polymer informatics, accelerating the discovery of novel polymers useful to society, and advancing the field of polymer science.

References

1. Audus DJ, de Pablo JJ. Polymer informatics: opportunities and challenges. *ACS Macro Lett.* 2017;6(10):1078–82.
2. Chen L, Pilania G, Batra R, Huan TD, Kim C, Kuenneth C, Ramprasad R. Polymer informatics: current status and critical next steps. *Mater Sci Eng R Rep.* 2021;144:100595.
3. Sha W, Li Y, Tang S, Tian J, Zhao Y, Guo Y, Zhang W, Zhang X, Lu S, Cao Y-C, Cheng S. Machine learning in polymer informatics. *InfoMat.* 2021;3(4):353–61.
4. Adams N. Polymer informatics. In: Meier MAR, Webster DC, editors. *Polymer libraries.* Berlin: Springer; 2010. p. 107–49.
5. Wu S, Yamada H, Hayashi Y, Zamengo M, Yoshida R. Potentials and challenges of polymer informatics: exploiting machine learning for polymer design. 2020; arXiv:2010.07683.
6. Ramprasad R, Batra R, Pilania G, Mannodi-Kanakkithodi A, Kim C. Machine learning in materials informatics: recent applications and prospects. *NPJ Comput Mater.* 2017;3(1):1–13.
7. Jackson NE, Theybb MA, de Pablo JJ. Recent advances in machine learning towards multiscale soft materials design. *Curr Opin Chem Eng.* 2019;23:106–14.
8. Adams N, Murray-Rust P. Engineering polymer informatics: towards the computer-aided design of polymers. *Macromol Rapid Commun.* 2008;29(8):615–32.
9. Zeng Y, Cao H, Ouyang Q, Qian Q. Multi-task learning and data augmentation for negative thermal expansion materials property prediction. *Mater Today Comm.* 2021;27:102314.
10. Hatakeyama-Sato K. Recent advances and challenges in experiment-oriented polymer informatics. *Polym J.* 2023;55:117–31.
11. Wang Y, Zhang M, Lin A, Iyer A, Prasad AS, Li X, Zhang Y, Schadler LS, Chen W, Brinson LC. Mining structure–property relationships in polymer nanocomposites using data driven finite element analysis and multi-task convolutional neural networks. *Mol Syst Des Eng.* 2020;5(5):962–75.
12. Yamaguchi T, Yamashita Y. Quality prediction for multi-grade batch process using sparse flexible clustered multi-task learning. *Comput Chem Eng.* 2021;150:107320.
13. Cassola S, Duhovic M, Schmidt T, May D. Machine learning for polymer composites process simulation—a review. *Compos B Eng.* 2022;246:110208.
14. Kuenneth C, Rajan AC, Tran H, Chen L, Kim C, Ramprasad R. Polymer informatics with multi-task learning. *Patterns.* 2021;2(4):100238.

15. Jose R, Ramakrishna S. Materials 4.0: materials big data enabled materials discovery. *App Mater Today.* 2018;10:127–32.
16. Zhou T, Song Z, Sundmacher K. Big data creates new opportunities for materials research: a review on methods and applications of machine learning for materials design. *Engineering.* 2019;5(6):1017–26.
17. Cencer MM, Moore JS, Assary RS. Machine learning for polymeric materials: an introduction. *Polym Int.* 2022;71(5):537–42.
18. Tchoua R, Hong Z, Audus D, Patel S, Ward L, Chard K, De Pablo J, Foster I. Developing databases for polymer informatics. *Bull Amer Phys Soc.* 2020;65(1): G34.00007
19. Hu B, Lin A, Brinson LC. ChemProps: a RESTful API enabled database for composite polymer name standardization. *J Cheminformatics.* 2021;13(1):22.
20. Wunderlich B. The ATHAS database on heat capacities of polymers. *Pure Appl Chem.* 1995;67:1019–26.
21. Lin T-S, Rebello NJ, Beech HK, Wang Z, El-Zaatari B, Lundberg DJ, Johnson JA, Kalow JA, Craig SL, Olsen BD. PolyDAT: a generic data schema for polymer characterization. *J Chem Inf Model.* 2021;61(3):1150–63.
22. Ma R, Luo T. PIIM: a benchmark database for polymer informatics. *J Chem Inf Model.* 2020;60(10):4684–90.
23. Otsuka S, Kuwajima I, Hosoya J, Xu Y, Yamazaki M. PoLyInfo: polymer database for polymeric materials design. In: 2011 IEEE International Conference on Emerging Intelligent Data and Web Technologies. 2011. p. 22–29.
24. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparragirre J, Hirzel TD, Adams RP, Aspuru-Guzik A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Sci.* 2018;4(2):268–76.
25. Kim C, Chandrasekaran A, Huan TD, Das D, Ramprasad R. Polymer genome: a data-powered polymer informatics platform for property predictions. *J Phys Chem C.* 2018;122(31): 17575–85.
26. Chandrasekaran A, Kim C, Ramprasad R. Polymer genome: a polymer informatics platform to accelerate polymer discovery. In: Schütt K, Chmiela S, von Lilienfeld O, Tkatchenko A, Tsuda K, Müller KR, editors. *Machine learning meets quantum physics.* Cham: Springer; 2020. p. 397–412.
27. Tran HD, Kim C, Chen L, Chandrasekaran A, Batra R, Venkatram S, Kamal D, Lightstone JP, Gurnani R, Shetty P, Ramprasad M, Laws J, Shelton M, Ramprasad R. Machine-learning predictions of polymer properties with polymer genome. *J App Phys.* 2020;128(17):171104.
28. Hayashi Y, Shiomi J, Morikawa J, Yoshida R. RadonPy: automated physical property calculation using all-atom classical molecular dynamics simulations for polymer informatics. *NPJ Comput Mater.* 2022;8:222.
29. Pilania G, Wang C, Wu K, Sukumar N, Breneman CM, Sotzing GA, Ramprasad R. New group IV chemical motifs for improved dielectric permittivity of polyethylene. *J Chem Inf Model.* 2013;53(4):879–86.
30. Wu K, Sukumar N, Lanzillo NA, Wang C, Ramprasad R, Ma R, Baldwin AF, Sotzing G, Breneman CM. Prediction of polymer properties using infinite chain descriptors (ICD) and machine learning: towards optimized dielectric polymeric materials. *J Polymer Sci B Polymer Phys.* 2016;54:2082–91.
31. Venkatraman V, Alsberg BK. Designing high-refractive index polymers using materials informatics. *Polymers.* 2018;10(1):103.
32. Samanta M. QSPR modelling of trends in thermal properties of polybenzoxazines. Masters thesis, Shiv Nadar University. 2022.
33. Chandrasekaran A, Kim C, Venkatram S, Ramprasad R. A deep learning solvent-selection paradigm powered by a massive solvent/nonsolvent database for polymers. *Macromolecules.* 2020;53(12):4764–9.

34. Hatakeyama-Sato K, Tezuka T, Nishikitani Y, Nishide H, Oyaizu K. Synthesis of lithium-ion conducting polymers designed by machine learning-based prediction and screening. *Chem Lett.* 2019;48(2):130–2.
35. Wu S, Kondo Y, Kakimoto M-A, Yang B, Yamada H, Kuwajima I, Lambard G, Hongo K, Xu Y, Shiomi J, Schick C, Morikawa J, Yoshida R. Machine-learning-assisted discovery of polymers with high thermal conductivity using a molecular design algorithm. *NPJ Comput Mater.* 2019;5(1):1–11.
36. Ho NX, Le TT, Le MV. Development of artificial intelligence based model for the prediction of Young's modulus of polymer/carbon-nanotubes composites. *Mech Adv Mater Struct.* 2021;29: 5965–78.
37. Tao L, Varshney V, Li Y. Benchmarking machine learning models for polymer informatics: an example of glass transition temperature. *J Chem Inf Model.* 2021;61(11):5395–413.
38. Ishikiriyama K. Polymer informatics based on the quantitative structure-property relationship using a machine-learning framework for the physical properties of polymers in the ATHAS data bank. *Thermochim Acta.* 2022;708:179135.
39. Gurnani R, Kuenneth C, Toland A, Ramprasad R. Polymer informatics at-scale with multitask graph neural networks. *Chem Mater.* 2023;35(4):1560–7.
40. Park N, Manica M, Born J, Hedrick J, Erdmann T, Zubarev D, Mill N, Arrechea P. An extensible software platform for accelerating polymer discovery through informatics and artificial intelligence development. *ChemRxiv;*2022:63e668409da0bc6b33b1870c.
41. Xu C, Wang Y, Farimani AB. TransPolymer: a transformer-based language model for polymer property predictions. *NPJ Comput Mater.* 2023;9:64.
42. Kuenneth C, Ramprasad R. polyBERT: a chemical language model to enable fully machine-driven ultrafast polymer informatics. *Nat Commun.* 2023;14:4099.
43. Sukumar N, Krein M, Luo Q, Breneman CM. MQSPR modeling in materials informatics: a way to shorten design cycles? *J Mater Sci.* 2012;47(21):7703–15.
44. Bicerano J. Computational modeling of polymers, Plastics engineering series, vol. 25. New York: Marcel Dekker; 1992.
45. Bicerano J. Prediction of polymer properties. New York: Marcel Dekker; 1996.
46. Lochab B, Monisha M, Amarnath N, Sharma P, Mukherjee S, Ishida H. Review on the accelerated and low-temperature polymerization of benzoxazine resins: addition polymerizable sustainable polymers. *Polymers.* 2021;13(8):1260.
47. Whitehead CE, Breneman CM, Sukumar N, Ryan MD. Transferable atom equivalent multi-centered multipole expansion method. *J Comp Chem.* 2003;24:514–29.
48. Chemical Computing Group. Molecular operating environment. 2021. <https://www.chemcomp.com/>.
49. Cravero F, Schustik SA, Martínez MJ, Vázquez GE, Díaz MF, Ponzoni I. Feature selection for polymer informatics: evaluating scalability and robustness of the FS4RVDD algorithm using synthetic polydisperse data sets. *J Chem Inf Model.* 2020;60(2):592–603.
50. Aoyagi T. Optimization of the elastic properties of block copolymers using coarse-grained simulation and an artificial neural network. *Comput Mater Sci.* 2022;207:111286.
51. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. 2017;*arXiv:1706.03762v5 [cs.CL].*
52. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. 2019;*arXiv:1810.04805.*

Synergistic Combination of Machine Learning and Evolutionary and Heuristic Algorithms for Handling Imbalance in Biological and Biomedical Datasets



Sonal Modak, Mayur Pandya, Patrick Siarry, and Jayaraman Valadi

Abstract Due to the advent of Next Generation Sequencing and multiple innovative experimental techniques there is an exponential increase in biological data. It is necessary to capture meaningful information and valuable knowledge from this data which can immensely benefit all living things on the planet. Recent advances in machine learning and deep learning have indeed been able to handle this with the deployment of novel algorithms. Data imbalance frequently occurs in biological data mining tasks. This is because in several omics related problems data belonging to the positive class is much less than the data belonging to the negative class. Such imbalance can affect performance and produce faulty results and conclusions. Nature inspired Evolutionary and Metaheuristic algorithms are robust and can handle data imbalance quite efficiently. In this work we have described the use of these algorithms for tackling data imbalance in biological data. We have provided lucid explanations of these algorithms along with potentially important case studies.

Keywords Data imbalance · Evolutionary optimization · Bioinformatics · Metaheuristic · Undersampling · Oversampling

S. Modak
Thermo Fisher Scientific, South San Francisco, CA, USA
e-mail: sonal.modak@thermofisher.com

M. Pandya
Department of Computer Science and Engineering, Manipal Academy of Higher Education,
Manipal, Karnataka, India
e-mail: mayur.pandya@learner.manipal.edu.in

P. Siarry
University Paris-Est Créteil, Créteil, France
e-mail: siarry@u-pec.fr

J. Valadi (✉)
Department of Computing and Data Sciences, Flame University, Pune, Maharashtra, India
e-mail: jayaraman.vk@flame.edu.in

1 Introduction

Technological advances including the advent of Next Generation Sequencing (NGS) have changed the ways data is generated and handled in different domains of biological sciences. This revolution has led to an exponential rise in the rate of data generation. Biological data can range from genomics data, proteomics metabolomics and other omics data to EEG of the brain signals [1–5]. Integrating multi-omics data along with imaging and clinical data is very crucial for the success of advancements in biomedicine [6]. Novel computational methodologies need to be developed for efficient analysis and interpretation. Hence, there is a need for advanced and rigorous models to interpret the large volume of data. Recently published algorithms, models, and database cum prediction servers in bioinformatics have been immensely valuable and useful to achieve these goals.

Conventional bioinformatics was mainly restricted to alignment-based methods [7]. Machine Learning (ML) in bioinformatics provided multiple alternative alignment free methods that performed very well in predictions. The new ML base methods are very useful in different omics related problems [8]. As an example, Support Vector Machine (SVM) can provide precise prediction for viral sequence identification [9, 10].

Algorithms based on deep learning-based methods have been proven to be proficient when it comes to handling big data. These methods have attained a good amount of success in bioinformatics. With the advances of big data in different subfields of biology, it is very much evident that deep learning will become increasingly important in the field. This will lead to various incorporations of different ML algorithms in vast majorities of analysis pipelines. Deep learning has recently become crucial in bioinformatics, biomedicine, and drug discovery [11]. Deep learning has been useful in structure prediction and reconstruction, biomolecular property and function prediction, sequence analysis, biomedical image processing and diagnosis [12–17].

It is very common for data belonging to bioinformatic domains to have different degrees of imbalance. Some of the frequently occurring example scenarios related to imbalanced problem in omics studies are prediction of protein-DNA binding residues from primary sequences [18], identification of miRNAs [19], identification of phase transfer proteins [20], PPI sites prediction [21, 22], etc. Biomedical datasets pertaining to the investigations of rare diseases or events are frequently severely imbalanced and most ML algorithms are not competent in such cases [23–25]. In integrative omics modelling tasks imbalance is listed among five challenges posed for building high performance prediction models [6]. In this work we have described imbalance handling methods employing evolutionary and metaheuristic algorithms.

Evolutionary and heuristics methods are simple, fast, flexible and easily adapted to a variety of problem formulations. They can be used for solving several real-life optimization problems with reasonable computational resources [26]. Metaheuristics methods include a plethora of families of algorithms. Some of the well-known algorithms are nature inspired which include Genetic Algorithms, swarm

intelligence-based methods, gravitational search algorithm and the Black Hole Algorithm. For example, Genetic Algorithms are inspired by biological evolution [27]. These methods usually start with a set of trial solutions and employ specific heuristics to iteratively improve and finally obtain near optimal solutions.

In this work we have described stochastic and metaheuristic algorithms which have been used for handling data imbalance. We have also illustrated some important case studies apart from that we have described our own case study with the recently developed Black Hole heuristic algorithm. The rest of the chapter is organized as: Section 2 describes some of the important machine learning algorithms. Section 3 deals with data imbalance and conventional imbalance handling methods. Section 4 describes the evolutionary meta heuristic algorithms and Sect. 5 illustrates some important case studies along with the algorithms. We provide a brief description of some important imbalance related examples in Table 1.

2 Machine Learning Algorithms

The ML paradigm is consistently evolving and newer algorithms and models [44, 45] are being proposed. This paradigm can be broadly identified into three different learning schemes: Supervised learning, Unsupervised learning, and Reinforcement learning. In this work we will mainly focus on supervised learning.

2.1 Supervised Learning Algorithms

Supervised learning involves building models with prior knowledge. These algorithms can further be divided into two parts based on data mining tasks: classification and regression. In classification tasks our focus is to group data into different classes. For regression tasks the output can be a real value and the model finds the best fit which minimises a performance metric like root mean square error. In binary classification, for example, we are given two sets of protein sequences experimentally annotated as antimicrobial (Class-A) and not anti-microbial (Class-B) respectively; we can build a classification algorithm by extracting sequence features from domain knowledge and use them as input data. As we have information about the classes, we can use the class labels A and B as the output to build classification models which map output as a function of inputs. Class labels (antimicrobial or not antimicrobial) of new unseen query sequences can be predicted using supervised learning models. Figure 1 showcases an example of a classification scenario with two attributes (1 and 2). This is a simple linear classifier which can linearly separate the data into two classes (Class-A and Class-B) using these attributes. Real life datasets may not be linearly separable for these types of data we may need to use classifiers capable of separating non-linear data. Standard algorithms used for this

Table 1 Examples of imbalance in bioinformatics

References	Dataset used	Method	Performance
[28]	Ovarian cancer microarray dataset (9600 features)	PSO is used in the pipeline of improved bacterial foraging optimization algorithm. This is used for classification of imbalanced data. Further Borderline-SMOTE and Tomek link were used to pre-process imbalanced data	Average classification accuracy obtained on datasets: ovarian cancer microarray data was 93.47%. The value of the AUC was 0.98
[29]	30 datasets from KEEL repository and UCI dataset repository	Genetic programming is used here. The Fitness function here uses entropy and information gain. This is done to improve the impurity and get a balanced result without changing the original dataset	In this comparative study, there was no consistently accurate method for the 30 datasets, however, the overall performance of Entropy and Information Gain based (EIG-GA) was higher than the other methods
[30]	Imbalanced microarray datasets, including Colon, Lung cancer, Central Neural System and Glioma	ACO based undersampling method	Proposed method was effective in extracting most informative samples from the majority class
[31]	Datasets from keel data mining repository (imbalance ratio between 1.5 and 9)	PSO based adaptive dataset partitioning, which optimizes the RF hyperparameters and also optimizes the misclassification of samples in the classification problem	Good performance in terms of F1 measure was obtained
[32]	40 datasets different imbalance ratios, and an additional real imbalanced dataset with traffic information	Hybrid metaheuristic consisting of Genetic Algorithm Cross Entropy (GACE), combining a metaheuristic algorithm with a loss function	The configuration of KNN and NN were used to generate the base classifiers, with different sizes for the genetic populations. This combination yielded the best results
[33]	Nine binary-class datasets from UCI machine learning repository	Class-specific cost regulation extreme learning machine (CCR-ELM) is an approach which is combined with Variable-length Brain Storm Optimization algorithm (VLen-BSO) to improve the generalization and classification accuracy	VLen-BSO finds better parameters of CCR-ELM, resulting in the better classification accuracy compared to evolutionary optimization algorithms, such as PSO, VPSO, and GA

(continued)

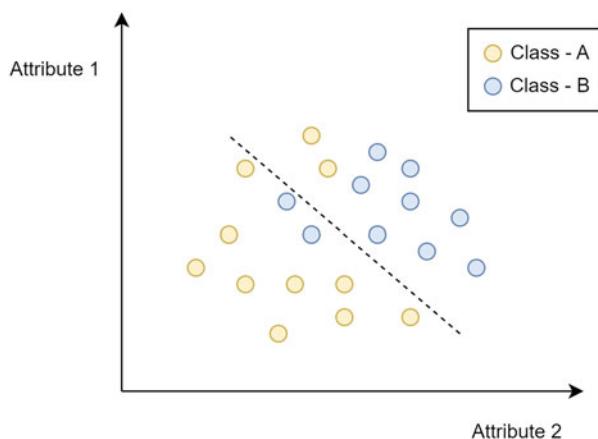
Table 1 (continued)

References	Dataset used	Method	Performance
[34]	UCI-ML repository: biological dataset on Alzheimer's disease and real-world dataset for the face recognition problem	GA-based method selects best classifiers for heterogeneous Ensemble of Classifiers (EoC). The best ensemble found is applied to the test data	The ensembles constructed are better than base classifiers in general. Further these were found to be better than those constructed by many established ensembles constructed methods
[35]	Childhood leukemia, Colon, Breast, Blood, Survival, Diabetes, Ionosphere	SVM classifier and Artificial bee colony optimization	ABC outperforms other methods in: AUC and F measure
[36]	Pima, Wisconsin, Hepatitis, SPECT	Fuzzy rule-based classifier with ACO oversampling	Excellent performance based on: Precision, Recall and F measure
[37]	Heart, Parkinson, Blood transfusion, CMC, Yeast, and Ionosphere	Decision tree classifier with GA oversampling	GA outperforms other methods in all the following metrices: Precision, Recall, F1 Score, AUC, Accuracy, Geometric Mean
[38]	Protein family database	Basic simple GA (SGA) classifier undersampling	Accuracy and AUC obtained are very good
[39]	Multiple biomedical datasets	Evolutionary cluster based oversampling ensemble	Very good AUC obtained on all the datasets
[40]	Colon, Leukaemia, lung, Tomlins, Armstrong, Golub_1990, DLBCL, su-2001, Yeoh-2002-v1	Genetic programming for classification, feature clustering, feature reduction and sampling	Accuracy obtained was very good
[41]	1016 proteins sequences belonging to Embryophy taxonomy of the Uniprot database	PSO method is used to obtain optimal balanced dataset	Computational time reduction in prediction of protein location without compromising on performance metrics
[42]	Five imbalanced medical surgery datasets based on survival	Optimization for controlling the inflation of minority class instances (swarm-SMOTE) by using stochastic swarm optimizing algorithms (PSO)	Proposed method outperforms the traditional class balancing method. Superior performance is obtained when swarm-SMOTE is coupled with classifier decision tree to solve the imbalanced dataset problem in biological medicine research

(continued)

Table 1 (continued)

References	Dataset used	Method	Performance
[43]	Four medical datasets and a genome wide association study (GWAS) dataset from genotyping of SNPs of Age-related Macular Degeneration	Approach combines PSO multiple classifiers. Majority class instances are ranked according to their merit in class imbalance compensation. Further these are combined with the minority class to form a balanced dataset	The proposed hybrid system can recover the power of classifiers towards imbalance data classification and also can indicate the relative importance of samples from majority class in contrast to samples from minority class

Fig. 1 Linear classifier used for classification task

task include KNN (k Nearest Neighbors), Naïve Bayes, Random Forest (RF), Support Vector Machine (SVM) and Artificial Neural Networks (ANNs).

We can use deep neural networks for data with large number of attributes and examples and image-based data [46–48]. We explain some of the widely used classifiers in the section below.

2.1.1 KNN (K Nearest Neighbors)

KNN is a non-parametric algorithm. It distinguishes data instances based on their proximity (closeness) and associations to other available data instances. A primary hypothesis made here is that data points belonging to a particular class are clustered together in proximity. KNN uses the distance function to find K nearest neighbors to a query instance. Figure 2 depicts an instance of KNN algorithm where two features (attributes 1 and 2) are used to classify instances into three classes (Class -A, B and C). For any new unseen example, the class label is assigned based on the frequency of classes present in the proximity of that data instance. For the task of regression

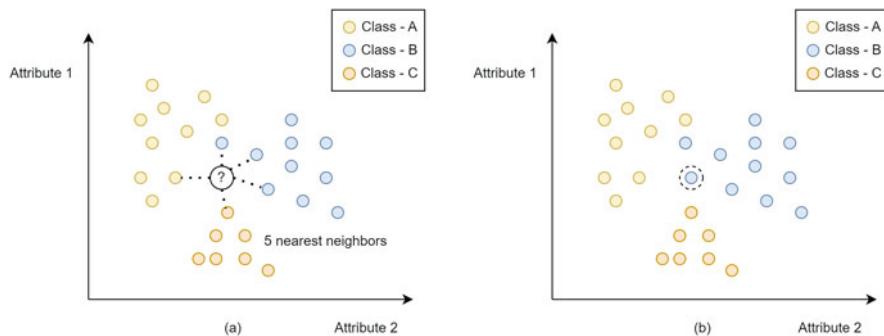


Fig. 2 (a) KNN algorithm ($K = 5$) given an unseen instance for class assignment. (b) Using the classes of 5 nearest neighbours', a class is assigned for the unseen instance

instead of frequency, the average is calculated. KNN is a simple but effective algorithm. This algorithm is sensitive to the preference of K values, as selecting a small K value can tend to make the algorithm more sensitive towards noise such as outliers. In contrast, a large value of K may result in over-smoothing of the decision surface.

2.1.2 Naïve Bayes

The Naïve Bayes classifier is based on the principle of conditional independence introduced in Bayes theorem. Naive Bayes is called “naive” because it simplifies the modelling by assuming independence among features, often oversimplified in real-world scenarios. Conditional independence assumption simplifies computation of the posterior probability of any given class for a given query example. The class label with the highest probability is then assigned to the test instance. However, despite the simplified hypothesis, Naive Bayes can perform well in practice.

2.1.3 SVM (Support Vector Machines)

SVM is a very rigorous and popular algorithm employed in many classification tasks in different areas [49–51]. For linearly separable instances SVM uses a maximum margin classifier. For data instances which are not linearly separable, SVM makes a transformation to map the instances to a higher dimensional feature space. SVM further employs a linear classifier in the mapped higher dimensional space. To deal with intractability SVM uses kernel functions which can facilitate computations in the lower dimensional space itself. Due to this, it has been deployed in various research domains like image classification, text categorization, and bioinformatics

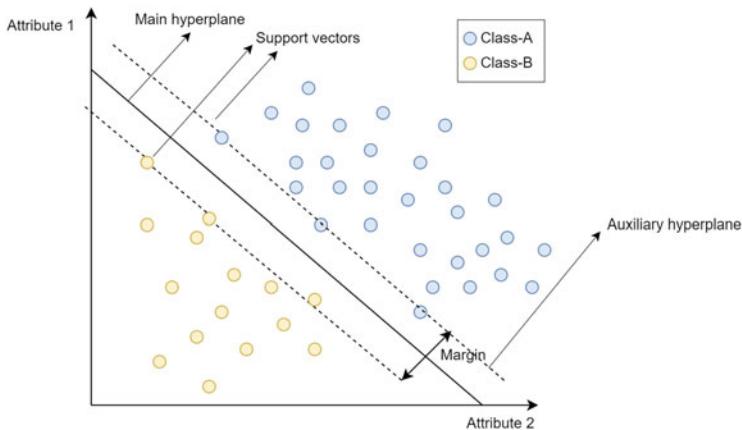


Fig. 3 Depiction of SVM as a maximum margin classifier. Instances lying on the margin are Support Vectors. Both classes are separated by a main hyperplane

[52–54]. Figure 3 below represents an SVM classifier consisting of the main hyperplane and the auxiliary hyperplane in the case of binary classification problem.

2.1.4 Random Forest (RF)

RF is a popular ensemble learning method algorithm for classification and regression tasks [55–59]. It unites the prediction powers of pooled decision trees to produce results more accurate than the individual decision trees. RF employs randomness in (1) sampling of each tree in the ensemble and (2) in the node splitting process. Every tree consists of distinct examples selected by bootstrap sampling with replacement. Also, RF uses only a random subset (predetermined constant number of features) of features while attribute splitting at every node of each tree across the ensemble. These two processes provide robustness, accuracy, and generalising capabilities. After all the decision trees are devised, predictions regarding new query instances are achieved by combining the predictions of each tree. For classification tasks, the most repeated class predicted by the decision trees is selected as the final prediction. For regression tasks, the average of the expected values is taken. RF has two different feature selection mechanisms built in the algorithm. Figure 4 depicts a RF model consisting of four decision trees.

Ensemble learning involves amalgamating several individual models into a unified and more robust model. The concept behind ensemble learning is to leverage the strengths of different models and reduce the limitations and biases of individual models.

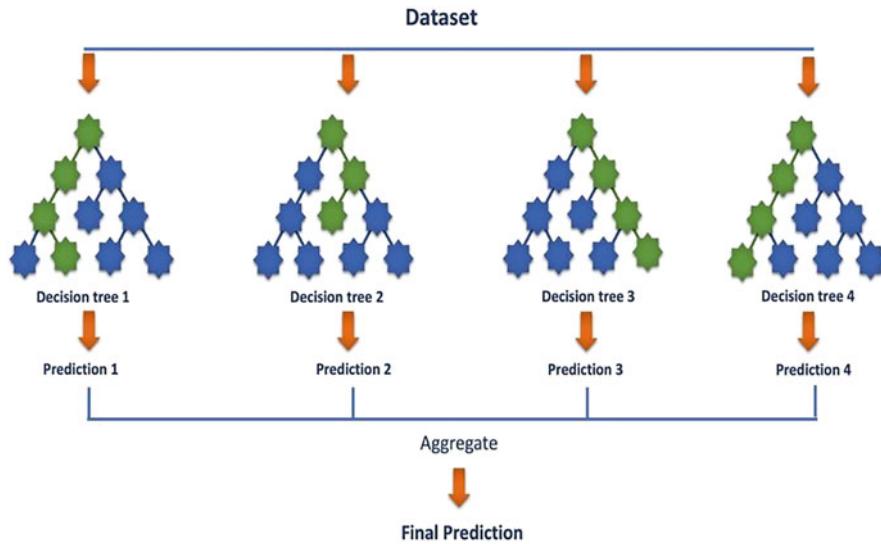
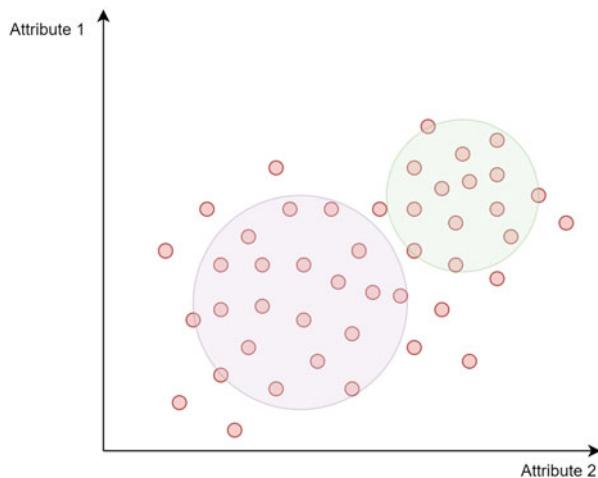


Fig. 4 Random forest overview

Fig. 5 Clustering scenario using an unsupervised learning algorithm using 2 features (Attribute 1 and 2)



2.2 Unsupervised Learning Algorithms

Unsupervised learning utilizes machine learning algorithms to analyse and group cluster unlabelled data. Without human supervision, these algorithms can identify and detect underlying patterns or data groupings. Unsupervised learning can be further classified into clustering and dimensionality reduction methods. Figure 5 below represents a clustering scenario in which the algorithm has found potentially two clusters in the dataset consisting of Attributes 1 and 2.

2.2.1 K-Means Clustering

The K-means clustering algorithm is a widely used unsupervised algorithm for clustering analysis. K-means aims to divide a dataset into K different clusters based on the similarity of datapoints. Each cluster has a centroid, which is the mean of all the data points affiliated to the cluster. This is depicted clearly in Fig. 6 The K-means algorithm steps are as highlighted below:

1. **Configuration:** Choose the value of K a priori which corresponds to the number of clusters. Randomly initialize the centroids of clusters.
2. **Assignment Step:** Designate each data instance to the cluster whose centroid lies closest to it. Closeness is determined by a distance metric.
3. **Update Step:** Recompute centroids of each cluster by utilizing mean of all the data points assigned to that cluster.
4. **Iteration:** Repeat 2 and 3 until a termination condition is achieved. Termination can include convergence which is said to occur when: a). The centroids do not change significantly. b). Completion of predefined number of iterations or c). Predefined performance metric does not change significantly.
5. **Output:** The final output of the K-means algorithm is the set of K clusters, each represented by its centroid. Additionally, each data instance is assigned to one of the K clusters.

2.3 Evaluation Measures for Classification Tasks

Evaluation measures are typically employed to assess classification models. For the task of classification, there are four types of prediction outcomes that could occur:

True positives (TP): Examples which originally belong to positive class and are correctly classified as positive.

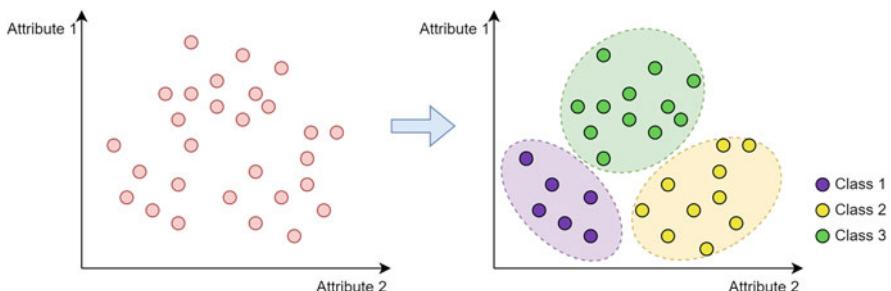


Fig. 6 K-means—(Right) Data instances plotted with attribute 1 on one axis and attribute 2 on another axis. (Left) 3 Clusters found by the K-means algorithm corresponding to 3 possible classes

Table 2 Confusion matrix for binary classification

	Actual positives	Actual negatives
Predicted positives	TP	FP
Predicted negatives	FN	TN

Table 3 Basic evaluation measures for analysing confusion matrix

Measure	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Sensitivity	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{TN+FP}$
F1-score	$2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$

True negatives (TN): Examples which originally belong to a negative class and are correctly classified as negative.

False positives (FP): Examples which originally belong to negative class but incorrectly classified as positive class.

False negatives (FN): Examples which originally belong to positive class but are incorrectly classified as negative class.

These four outcomes are often plotted on a confusion matrix. An elementary scheme of confusion matrix in binary classification is seen in Table 2.

List of some of the most common metrics used is detailed in Table 3. Some of the evaluation measures are further described in detail in the section below.

2.3.1 Area Under the Curve (AUC): Receiving Operating Characteristic (ROC)

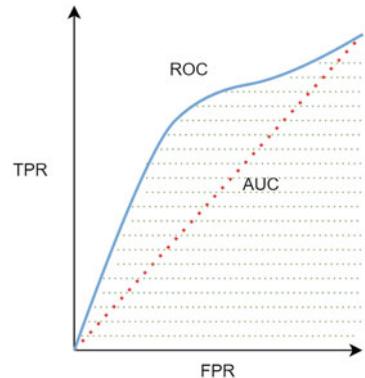
Sometimes visualization tools are required to visualize the performance of the classifier as a graphical presentation. AUC-ROC is one of the widely used performance visualization metric. ROC is a curve which displays the performance of a classification model at different set threshold levels. The curve is plotted for two parameters: True Positive Rate (TPR) and False Positive Rate (FPR). TPR is same as Precision and FPR is given as:

$$FPR = \frac{FP}{FP + TN}$$

Logistic regression model is used to find value of any point in a ROC curve. It is not an efficient approach. As a workaround AUC approach is used. It calculates the two-dimensional area, under the entire ROC curve, as shown below Fig. 7 below.

AUC gives a visual of performance across all the thresholds and provides an aggregation. AUC value ranges from 0 to 1. A model having a 100% wrong prediction will have value 0, whereas 100% correct predictions lead to 1.

Fig. 7 Plot depicting two-dimensional area under the entire ROC



In the above sections, we had dived deep into the concepts related to imbalanced data sets and established base understanding of tools which helps in dealing with such data in classification problems. In another words, we reviewed data-oriented solutions and approaches to address problem with the imbalance dataset. In further sections we will review the other side where the same problem has been addressed at algorithmic level.

2.3.2 Matthews Correlation Coefficient (MCC)

Accuracy may not always be best possible metric to be used to evaluate the classifier performance. Especially in imbalance datasets accuracy measure will be biased towards majority class. One approach to solve the problem is to use multiobjective optimization to maximize positive and negative accuracy simultaneously. Another approach which is simpler but effective is to compute the statistical measure known as Matthew Correlation Coefficient [60–62]. This is defined as:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

In this section we reviewed the machine learning algorithms available for data handling and how their performance can be quantified using valid quantifiers. Next, we take a look into some of the concepts related to imbalanced data.

3 Imbalance of Data

A balanced data set shall contain a comparatively equal number of representative samples from every class under observation. The class in the data sets with fewer samples to train the classifiers is called the minority class. In contrast, the other

classes with a relatively ample number of samples are the majority classes. A disproportionate number of samples results in poor performance of the classifiers in minority classes. Using imbalanced datasets with uneven class distribution results in a degenerated classification model since it leads to biased results. Classifiers like Support Vector Machines and Random Forest which provide excellent performance with balanced datasets can fail very badly with data imbalance. Minority examples can often overlap with other regions and prior probabilities of both majority and minority classes can be identical. Rare minority examples occurring along with outliers pose further problems as the former can be recognized as the latter and vice versa. Bioinformatics domain has problems with high imbalance ratio, small number of samples and large number of data sets. In such cases prediction performance can be very low. Thus, handling and dealing with imbalanced data sets has become an important problem in data mining tasks [63, 64].

Consider an example of cancer diagnosis, where a non-cancer or healthy class is labelled as negative, while a positive class is a particular type of cancer. A false negative classification in such a case would be missing the correct diagnosis and concluding that patient with cancer will be classified as normal person. This misclassification can have dire consequences since it delays appropriate treatment due to misdiagnosis, which could be life-threatening to the patient. Besides medical diagnosis, imbalanced datasets affect several other areas, including toxicology, classifications based on satellite imagery analysis, and text-based categorization detection of online frauds [65].

The problem of class imbalance is equally applicable to multi-class classification problems just as it is relevant in the case of binary classification. In the case of multi-class classification, there exists a possibility that more than one class falls under the minority category or that there exist multiple majority classes in the dataset. The central concern with imbalance is that these classifiers need to perform satisfactorily on them. Nearly all the classification models rely on the underlying class distribution, which can be inferred from the training dataset. Inferring underlying class distribution is done by gauging the likelihood of instance observation when the model is used for making predictions. The problem that will occur is that algorithms like k Nearest Neighbors, decision trees, and neural networks will infer that the importance of the minority class is not the same as that of the majority class.

The most prevalent solution to deal with imbalanced classification is introducing changes in the training dataset [66–71]. Approaches such as *Sampling methods* are designed to make class distribution changes in the training split of the imbalanced dataset. These methods will sample existing data instances. These are easy-to-understand methods and equally easy to implement. Most machine learning models are compatible with this kind of sampling method transformations made to the training data. Approaches such as [72, 73] focuses on tuning evaluation measures to assess the models. This model assessment measures can deal with uneven class distribution in the data. In the later sections, we will dive deep into the fundamental evaluation measures and discuss the approach to combine the measures adopted for imbalanced data learning.

3.1 Data Sampling Methods

There is a plethora of data sampling techniques that we can choose for adjusting the training dataset's class distribution. However, the optimal data sampling method does not exist, and the technique selection mainly relies on the domain knowledge and problem statement. Undersampling methods reduce the data instances in the majority class to balance the dataset. Oversampling methods increase the number of examples in the minority class. Sampling methods act differently based on the selection of the machine learning algorithm and the nature of the dataset. We must design our simulations for testing and evaluating different configurations over a combination of methods to fully understand which approach works best for the given problem statement. Some existing techniques are used more and have a higher success rate when deployed. We describe some of these techniques from the binary (two-class) classification problem perspective, which is the most common approach. However, the mentioned methods can also be utilized or adapted for imbalance classification in case of multi-class classification.

3.1.1 Oversampling Techniques

Oversampling techniques duplicate the data instances which belong to the minority class [74]. Simultaneously, they synthesize new data instances by utilizing the examples from the minority class. Some of the most implemented oversampling techniques are Random Oversampling, Synthetic Minority Oversampling Technique (SMOTE) [75], Borderline-SMOTE [76], Adaptive Synthetic Sampling (ADASYN) [77], and Borderline Oversampling with SVM. The most preliminary oversampling approach involves random duplication of minority class instances in the training data, known as random oversampling. SMOTE is the most common and, subsequently, the most successful oversampling method. Synthetic Minority Oversampling Technique acts by collecting instances that are close to one another in the feature space. It then connects the instances by drawing a line in that space and sampling new points along the line. Various extensions of SMOTE aim to be more selective towards the types of synthesized examples in the majority class. A particular approach of borderline SMOTE works by selecting instances of minority classes that are misclassified. It further utilizes these instances in generating synthetic samples that are hard to classify. This extension uses SVM to fit the training data, and its decision boundary is used to find the support vectors. These support vectors act as the basis for the generation of synthetic examples. This, again, intuitively indicates that the decision boundary is the region where more minority class instances will lie. ADASYN is a SMOTE extension that generates synthetic instances inversely proportional to the density of instances in the minority class instances. This feature ensures that the synthetic instances generated lie around the region of feature space where the density of minority classes is low.

3.1.2 Undersampling Techniques

Undersampling techniques are the ones that delete or collect a subset of instances belonging to the majority class [23]. Some of the commonly used and widely implemented undersampling techniques include Random Undersampling, Near Miss Undersampling, One-Sided Selection (OSS), Neighborhood Cleaning Rule (NCR), and Condensed Nearest Neighbour Rule (CNN). The most frequently used undersampling technique includes randomly removing instances from the training dataset corresponding to the majority class. This methodology is known as random undersampling. Determining a representative subset of the majority class's instances is one strategy. The k-nearest neighbor method requires much memory; hence, the Condensed Nearest Neighbors rule was created to reduce that. The core idea is to count every instance in the dataset and only collect them if the classifier incorrectly categorized them. Upon adding all instances from the minority class, this may be used to decrease the number of examples that belong to the majority class. Near Miss is a member of the family of techniques that choose examples from the majority class using KNN. For instance, NearMiss-1 will choose instances from the majority class whose average distance from the three nearest lying examples from the minority class is the shortest. Similar to NearMiss-2, it will choose majority class instances closest to the three examples farthest from the minority class. A different strategy includes choosing which instances from the majority class to remove. Such methods entail the identification of examples that are difficult to categorize, making the decision border more ambiguous. Tomek Links, the most well-known deletion undersampling technique, was first created as an addition to the Condensed Nearest Neighbors rule. When two instances from the training set are nearest neighbors but belong to different classes, they are said to be connected by a Tomek connection. Tomek linkages are often made up of misclassified instances close to class borders. *Edited Nearest Neighbors (ENN)* rules are a method for selecting instances for removal. This approach rule involves using k Nearest Neighbors with k value as three. KNN is used to locate misclassified instances in the dataset. ENN procedure can be repeated multiple times on the same training dataset. This repeated iteration helps with refining the majority class instances further. This approach is also known as unlimited editing. Undersampling methods usually follow the select-to-keep or select-to-delete approach. However, some approaches combine these for undersampling. One-sided selection is a technique that includes Tomek Links and the Condensed Nearest Neighbor rule. Tomek Links help in the removal of noisy instances near the class boundary. CNN helps remove redundant instances from the interior density of majority classes. Cost-sensitive learning methods are other group of methods where we provide higher misclassification class to the minority class samples. We can use SVM and ANN for cost sensitive weighting. Ensemble methods are also used to reduce class imbalance.

3.1.3 Combination Techniques

Using standalone oversampling and undersampling on training datasets can be practical, and previous experiments have proved that applying both techniques in tandem can often lead to better overall performance by fitting the model on resulting synthetic data. Common combinations include SMOTE and Random Undersampling, SMOTE, Edited Nearest Neighbors rule, and SMOTE and Tomek Links. The most used oversampling technique is SMOTE. Mostly, it is paired with a range of undersampling methods. The most used combination is SMOTE and random undersampling. It is a common practice to pair SMOTE with an undersampling method, which emphasizes deleting majority class instances. This procedure is applied after SMOTE. This aids in applying the editing step to minority and majority class instances. The underlying intention is to remove noisy points that lie along the class boundary and belong to both classes. This approach has better performance by fitting the transformed dataset.

In stratified sampling the data is first divided into subgroups based on some similarity. Subsequently samples are drawn from each subgroup proportional to the fraction of data present in a group.. This technique effectively removes bias and provides fair representation of all groups. Stratified sampling can be employed in imbalance during splitting data into train and test and during k- fold cross validation. Undersampling and oversampling can create biased estimators as they reduce fairness. It is possible to increase fairness by different methods. One way to mitigate the problem is to balance sensitive attributes along with balancing classes.

In the above sections, we had dived deep into the concepts related to imbalanced data sets and established base understanding of tools which helps in dealing with such data in classification problems. In another words, we reviewed data-oriented solutions and approaches to address problem with the imbalance dataset. In further sections we will review the other side where the same problem has been addressed at algorithmic level.

4 Evolutionary and Heuristic Algorithms

Evolutionary heuristic optimization strategies have found numerous applications for several optimization problems in different domains. These algorithms are successful mainly because of their simplicity and adaptability. Also, they are easy to parallelize. These meta-heuristic and evolutionary methods form an integral subset of AI methods [78, 79]. These evolutionary approaches have been employed in different biological domains and have been proven to give good results [80–83]. These heuristic optimization tools differ in approach from conventional mathematical programming methods. The conventional optimization tools mainly consist of gradient-based methods, which need derivatives and apply primarily to continuous objective functions. Evolutionary and heuristic computational methods do not

require derivative evaluations. These techniques have a rigorous basis and provide reasonably good candidate solutions without forming complex model equations. For handling imbalance datasets, we have focused on five approaches, which are Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) and Black Hole Algorithm (BHA). The GA is inspired by the natural evolution process of ‘survival of the fittest.’ It uses different operators for selection, crossover, and mutation mechanisms to search for solutions. Operators are used to iteratively update and arrive at the best possible solution(s). Ant Colony Optimization is based on the cooperative search behavior of real-life ants. The base inspiration lies in the idea that ants can cooperatively carry out several tasks, including optimizing their approach routes toward food sources. Route optimization is possible due to their ability to deposit a pheromone chemical. Ants can get attracted to the pheromone-rich trails left by other ants and enhance the shortest trail in an auto-catalytic feedback manner. In the case of PSO, the swarm’s behavior as a collective is portrayed. Here, the artificial swarm particles consisting of potential solutions mimic how real-life birds have been observed to cooperatively plan their movement by adjusting their speed with the swarm. ABC is an optimization algorithm based on the intelligent foraging behavior of a swarm of honeybees. The model is made up of three key components: employed and unemployed foraging bees, as well as food sources. The Black Hole algorithm is inspired by the real-life behavior of Black Holes and stars. In the following sub-sections, we elaborate upon these metaheuristic algorithms for handling imbalance. As the metaheuristic algorithms have been mainly used for undersampling the majority class examples, we first provide the general algorithm steps to carry out underdamping and then proceed to describe different metaheuristic algorithms.

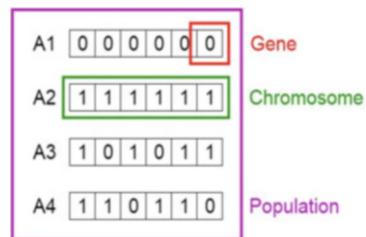
4.1 *Genetic Algorithm (GA)*

Genetic Algorithm is a metaheuristic optimization algorithm mimicking natural evolution [84]. In natural evolution entities with the highest fitness are selected for reproduction. GA are inspired by this principle for solving several real-life problems [85, 86]. In bioinformatics, GA has been used profusely in different sub domains of interest [87, 88]. GA is very attractive and popular because it is derivative free methodology. GA is very simple to use but at the same time it is backed by rigorous theory.

A singular objective unconstrained optimization problem can be defined as

$$\text{Maximize } f(x_1, x_2, \dots, x_n)$$

GA generates a certain number of trial solutions and use the genetic operators to change the solutions to increase their fitness and during this process the number of solutions remain constant. The applications of genetic operators to the newly created

Fig. 8 Components of GA

population of solutions are repeated until convergence. To tackle different types of problems multiple GA representations can be used.

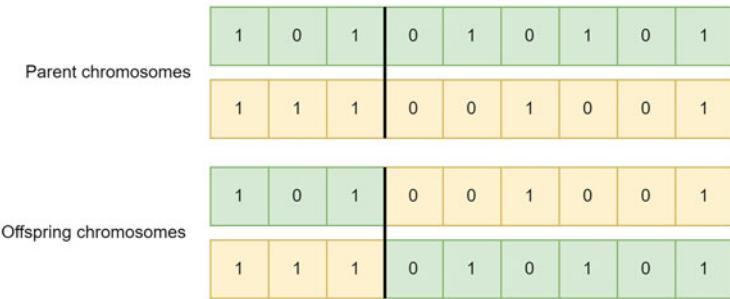
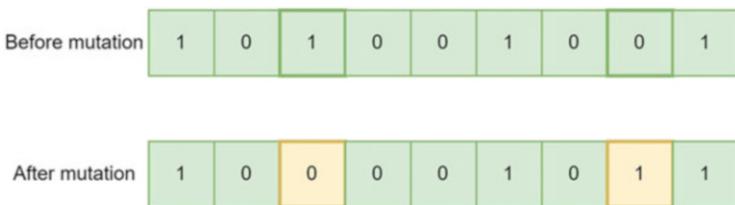
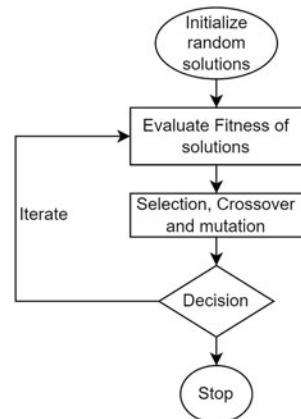
In the most common binary representation of the population each of the variables are converted to binary bits of certain length. The binary form of the solutions is called chromosomes. The bits representing a given variable are called a gene. There are multiple encoding schemes available in the literature which include Binary representation, permutation representation etc. This is depicted clearly in Fig. 8 below. The popular selection techniques are Roulette wheel selection and Tournament selection. Other selection techniques are also available in literature.

Fitness functions are used to resolve the capacity of each individual solution to compete with other solutions present in the population. After every iteration, individual solutions are evaluated using their fitness function values. Individual solutions are ranked according to fitness score. The fitness score also governs the probability of selection for that solution. Hence, the higher the fitness score, higher are the chances of selection for that solution to pass towards reproduction.

The next phase of solution i.e., selection involves collecting fitter solutions for the purpose of reproduction of offsprings. There are several selection strategies which include tournament and Roulette wheel selection. In tournament selection for example, we select population members at random (usually 2) and conduct a tournament between them. The fittest solution among these is selected. Several such tournaments are conducted until the selected solutions equal the number of solutions before the selection process. After selection the reproduction operators known as crossover and mutation operate on selected solutions to generate new offsprings.

Crossover operator plays a major role during offspring creation. Here, a crossover point must be selected at random. For example, in one point crossover, the operator swaps the genetic information of both parent solutions from the current generation and creates two new offsprings. This is depicted clearly in Fig. 9 below.

The newly created offsprings are added to the existing population. The mutation operator has the ability to insert random genes in the newly created solution (offspring). This maintains the diversity in the population. This is usually achieved by random flipping of some bits in the chromosome. Mutation is very useful for solving the problem of premature convergence. This further enhances diversification. Figure 10 depicts the mutation process. After one generation the processes of selection crossover and mutation are repeated until convergence. The entire algorithm of GA as a flowchart is represented in Fig. 11.

**Fig. 9** Single point crossover operator in GA**Fig. 10** Mutation operator in GA**Fig. 11** Basic process of GA

4.2 Ant Colony Optimization

Natural scientists and biologists have been exploring the real-life behavior of ants for solving optimization problems. This area of research is motivated by the exceptional capabilities of social insects utilizing collective intelligence to solve intricate problems. Marco Dorigo in 1992 [89, 90] demonstrated the Ant System as a metaheuristic for optimization problems and since then it extended into various

forms to address optimization problems in different domains. ACO has been effectively employed to address various practical optimization problems, including protein folding [91], parameter optimization [92], Travelling salesman problem (TSP) [93], path planning [94] etc.

The ACO algorithm mimics the real-life behavior of ants and employs this information for optimizing several real-life problems in different areas. Certain types of ants can deposit a chemical known as pheromone and they can also get attracted to pheromone rich path. This behaviour provides autocatalytic feedback which results in establishing the shortest path between their nest and food source. ACO algorithms are inspired by these pheromones mediated search and have been successfully employed for solving different types of optimization problems. Dorigo et al. [89, 90] first applied ACO algorithm for solving traveling salesman problem. In this problem, a salesman has to start with a particular city and visit all cities in the tour and must come back to the same city. The problem here is to find the optimal shortest tour. The algorithm used by them is as follows:

1. Deploy certain number of ants to conduct the tour.
2. Randomly deposit pheromone on all the link of the tour.
3. Locate the first ant in a random city.
4. Allow the ant to move from city i to city j probabilistically by using either exploration or exploitation. A parameter q (range 0 to 1) controls this selection.
5. If exploitation is selected, the ant moves to the city j (yet to be visited) which has the maximum pheromone link from city i .
6. If exploration is selected, ant moves to one of the yet to be visited city probabilistically, with the probability proportional to the pheromone value connecting city i to city j .
7. In this manner the ant visits all the cities in the tour and comes back to the original city.
8. All other ants conduct their own tour in the same manner.
9. After one iteration the pheromones of all the links are reduced by a certain amount.
10. The pheromones of the links visited by the best tour are increased by a value proportional to the quality of the tour conducted.
11. Steps 4 to 10 are repeated until convergence.

4.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a metaheuristic employed for solving real life optimization problems. This algorithm derives its inspiration from the cooperative swarm behavior of birds where they move together in order to achieve a common goal. Kennedy and Ebenhart originally modelled the optimization algorithm as the movement of particles in a swarm [95]. Each particle represents a trial solution whose fitness is evaluated based on a domain specific performance metric. The particles will traverse through the solution space of the problem and update their

initial positions based on the experience collected by them along with their neighbors. The velocity component and position component of individual particle in search space domain are crucial as they determine solution quality. The velocity determines the direction and the speed with which that particle is moving.

A fitness function is utilized to calculate the quality of each individual particle present in the population. The fitness function considers the position of the particle in the search space to determine the fitness of that particle. The next step is to update the velocity component of the particle pools. The velocity of each particle is adjusted based on two elements; the best value of fitness that particle was able to achieve until that time step and the best position found by the entire swarm. This in turn updates the position of the particle and moves it closer to the optimal solution in the search space. The PSO algorithm continues until a termination condition is not met. Termination conditions can include reaching maximum number of iterations or achieving satisfactory solution. The final best position found by swarm represents the optimized solution to a particular problem. The generalized PSO is summarized below.

Let's assume a few parameters first.

- f: Problem function to be optimized.
- Vel_i^t : Velocity of i^{th} particle.
- N: Population pool of particles
- W: Inertia weight
- C1: cognitive constant
- C2: social constant
- pos_i^t : Position of i^{th} particle.
- P_{best}^t : Personal best solution of a particle.
- g_{best}^t : Global best solution from entire population pool.

The actual algorithm has the following outline.

1. Create population pool of particles uniformly distributed over a search space.
2. Evaluate individual particle's position using f.
3. Find the best particle present in the population based on its position in the search space.
4. Update individual particle velocities:

$$Vel_i^{t+1} = W * Vel_i^t + C_1 * U_1^t (P_{best}^t - P_i^t) + C_2 * U_2^t (g_{best}^t - P_i^t)$$

Where t is the current iteration and i is the selected particle. U_1 and U_2 are random numbers in the range of 0 to 1.

5. Particle movement towards new positions

$$Pos_i^{t+1} = Pos_i^t + Vel_i^{t+1}$$

6. Repeat steps 2 to 6 until the set convergence condition is not fulfilled

Additionally, regarding the constants C_1 and C_2 .

- If C_1 and C_2 are both zero, all particles continue moving at their current velocity until they reach the search space's boundary and get stuck there.
- If $C_1 > 0$ but $C_2 = 0$, all particles are independent i.e., no swarm behaviour is observed.
- If C_1 and C_2 are equal and non-zero, all particles in the swarm are attracted towards the average of P_{best} and g_{best} .

4.4 Artificial Bee Colony (ABC)

Artificial Bee Colony optimization is a nature inspired metaheuristic algorithm which mimics the behaviour of honeybee colonies. In a real bee colony, bees communicate with each other through dancing and deposition of pheromones to find the best food sources. Similarly, in the ABC algorithm, artificial bees explore the search space to find the optimal solution for a given problem [96, 97].

The ABC algorithm works by maintaining a population of artificial bees, which are essentially candidate solutions to the optimization problem. Each artificial bee represents a potential solution, and the quality of the solution is determined by an objective function.

The algorithm consists of three main phases:

1. **Employed bees' phase:** Each employed bee explores a neighbourhood around its current solution by perturbing the value of one parameter at a time. The newly generated solutions are evaluated, and if a better solution is found, the bee updates its position iteratively. This phase aims to exploit the current promising solutions present in the solution space.
2. **Onlooker bees' phase:** Onlooker bees select solutions based on their quality and the probabilities associated with them. The fitness function is utilized to determine the quality of candidate solutions. Solutions with higher fitness values have higher chances of being selected. The onlooker bees then perform a local search around the selected solutions. This phase helps in the exploration of the search space.
3. **Scout bees' phase:** Scout bees are responsible for introducing random solutions into the population. If an employed bee exhausts its search space without finding a better solution, it becomes a scout bee and generates a new random solution to continue the search.

The algorithm iteratively repeats these phases until a termination condition is met, such as reaching a maximum number of iterations or finding an acceptable solution. The general algorithm can be given as follows:

Initialize:

Generate a population (P) of artificial bees with random solutions
 Evaluate the fitness of each candidate solution using a fitness function.
 Set the maximum number of iterations (max_i)
 Set the number of cycles without improvement for scout bees (max_i_no_imrpov)
 Set the current iteration count to 0 (current_i)

while current_i < max_i:

Employed Bee's Phase:

for each employed bee:

Select a random neighbour bee (neighbour)

Generate a new candidate solution (cd_s) by modifying current solution (c_s):

$cd_s = c_s + \text{rand}(-1, 1) * (c_s - \text{neighbour})$

Evaluate the fitness of the cd_s ($f(cd_s)$)

if $f(cd_s) > f(c_s)$:

 current_solution = new candidate_solution

Calculate the fitness probabilities for onlooker bees:

Calculate the fitness value of each food source (solutions) ($f(f_s)$)

Calculate the total fitness of all food sources (tot_f)

Calculate the probability of selecting each food source:

$$P(f_s) = f(f_s) / \text{tot_f}$$

Onlooker Bee's Phase:

for each onlooker bee:

Select a food source based on $P(f_s)$ using selection (fs_sel):

Generate a cd_s by modifying the c_s:

$cd_s = fs_sel + \text{rand}(-1, 1) * (fs_sel - c_s)$

Evaluate the fitness of the cd_s

if $f(cd_s) > f(fs_sel)$:

$fs_sel = cd_s$

Scout Bees Phase:

for each food source:

if f_s not improved for max_i_no_improv cycles:

 new_solution = generate_random_solution()

Evaluate the fitness of the new solution ($f(ns)$)

if $f(ns)$ is better than $f(fs_sel)$:

$fs_sel = ns$

 current_i = current_i + 1

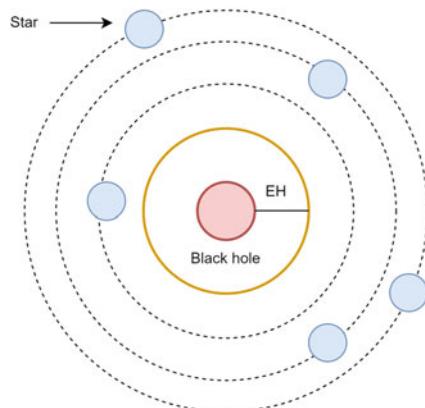
Return the best solution found

4.5 Black Hole Algorithm

Black Hole Algorithm mimics the real life behavior of black holes and stars. Just like most heuristic algorithms, Black Hole algorithm is also a population-based optimization technique [98, 99]. The initialization step for Black Hole Algorithm consists of creating a population of trial solutions of size N. This population set consists of vectors which are called stars. These stars are the trial solutions for the objective function which we need to solve. The objective function will quantify the quality of the generated trial solutions i.e., the stars. This is achieved by evaluating the fitness of each star. The fitness term can include performance metrics such as accuracy, precision, sensitivity along some constant values and a penalty term to regulate the generated stars. Based on the fitness value we select the best solution (star) as the Black Hole. The next step to carry out ‘movement’ of rest of the stars towards the selected Black Hole. The movement step consists of updating the solution vector using the Black Hole. Once the movement is carried out the next step is to calculate the event horizon. The Event horizon step consists of checking whether the updated stars have come very close to the Black Hole. This is done by checking the similarity-based closeness (distance) between the Black Hole and the candidate star. Figure 12 below shows the process of finding the distances between the stars and Black Hole respectively.

Based on the set threshold we remove the candidate star from the population pool and replace it with a randomly generated solution. Once we have carried out this check, we repeat the above-mentioned steps for a predetermined number of iterations or until convergence is achieved.

Fig. 12 Black Hole algorithm with the best solution as a black hole and other solutions as stars



5 Case Studies Involving Evolutionary Algorithms Utilization for Imbalance Handling

In the following section we will have a look at some interesting case studies where traditional algorithms were employed for pre-processing imbalanced datasets, and ultimately aiding to performance of the underlying prediction problem. Accompanying these are brief descriptions of some evolutionary metaheuristic algorithms for imbalance handling. Rare events and abnormalities detection and modeling aspects differ from imbalance handling procedures, and we have not dealt with these aspects in the present work.

5.1 General Algorithm Steps to Carry Out Undersampling

The first step is to create a population of N trial solutions as initialization step. This is done by creating vectors whose length corresponds to the number of majority class examples present in the imbalanced dataset. Each element of these vectors (trial solutions) are randomly generated 0s and 1s. Here 0 indicates that the data instance is not selected and 1 means that the instance is selected. For example, consider the following Fig. 13 which depicts population consisting of 5 vectors of length 8 each. Here 8 represents the total number of majority class examples in the dataset.

After initialization step, based on the selected instances we extract those instances from the imbalanced dataset to create a new dataset. For example, based on vector 1 present in the population of trial solutions figure (*) instances 1, 5 and 8 will be selected to generate the new dataset. These subsets consisting of selected instances are combined with the minority class examples and passed to a classifier model. Each vector is evaluated using the fitness function which consists of selected

Fig. 13 Population of 5 solution vectors each representing 8 majority class instances

1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	1
1	1	0	0	1	0	1	1
1	0	1	0	1	0	0	1
1	0	1	1	0	0	0	1

performance metrics such as accuracy, precision, recall or F1 score. Once we have evaluated all the vectors present in the population set, the next step is to employ a metaheuristic algorithm. For example, in case of GA, carry out selection using crossover and mutation and in the case of Black Hole Algorithm move the stars towards the selected Black Hole star. The next step is to evaluate this newly updated population set consisting of new vectors. Repeat these steps till we achieve convergence.

5.2 Identification of Cytokines Via an Improved GA [38]

Recent development and research interest in the field of Bioinformatics has led to significant focus towards cytokines identification. Cytokines are essentially a form of proteins which are also responsible for cell signalling as they are small signalling molecules. It is a very strong belief that cytokines can aid in prevention, diagnosis as well as curing of disease such as tumours, hematopoietic disorder, and inflammations. The authors of this research utilized N-gram, a language model for extracting features for protein [38]. The dataset consisted of 126-cytokine sequences and a 10,260-protein sequences were utilized as negative class examples. The imbalanced data had a ratio of 1: 81.43. Twenty amino acid frequencies and four hundred dipeptide frequencies were included to form 420 attributes for the input.

The classifier used here was Simple Genetic Algorithm (SGA) for training the subset. This SGA classifier was further modified to get better performance. The basic SGA classifier consisted of two steps in which the first one was about constructing a sampling model using GA. GA steps of selection such as crossover and mutation were used to get the balanced set of examples, using the procedure explained in the previous section. This was done to achieve balanced and better training dataset from the preliminary available imbalanced dataset. Once a balanced subset was achieved, the self-built classifier used the balanced data to train itself. The fitness function assessed the quality of groups formulated by the GA process. There were two sets of fitness functions used here by the authors, one was a regular fitness function and the other was an improved fitness function corresponding to the upgraded SGA model.

The regular fitness function formulated here was a simple accuracy metric.

$$f_{regular} = \frac{TP + TN}{TP + FP + TN + FN}$$

Here, TP, TN, FP and FN follow the standard nomenclature of True Positive, True Negative, False Positive and False Negative respectively. The improved fitness function used to construct the improved SGA is as follows,

$$f_{improved} = \omega * \frac{TP}{TP + FN} + (1 - \omega) * \frac{TN}{TN + FP}$$

Here, ω acted as the minority class contribution regulator, further the classification accuracy of both the binary classes were represented equally here. A simple train test split was used to maximize the fitness function value. The reported performance metrics were accuracy, G-Mean and AUC. These values were tabulated for different values of ω . Best results obtained were for $\omega = 0.5$. The accuracy reported was 0.60 ± 0.05 while the G-mean was 0.64 ± 0.03 and the AUC obtained was 0.65 ± 0.03 .

5.3 DNA Microarray Imbalance Handling Via Ant Colony Optimization [30]

In recent years, DNA microarray has emerged as one of the most important technologies of post-genomic era. Utilizing DNA microarray, biology researchers and medical professionals detect activities of thousands of genes present in the cell. Currently DNA microarray is widely used for tasks such as prediction of gene functions and gene regulatory mechanism investigations. But one of the concerning issues regarding DNA microarray is imbalanced class distributions. The authors here modified the ACO algorithm for the task of undersampling the microarray data. Further each majority class instance is ranked according to its importance via a frequency list [30]. The ACO based undersampling method utilized for imbalanced classification of DNA microarray data works is as follows:

Initially all the ants are in the nest; these ants must traverse the graph consisting of two paths for every majority class sample. These paths are denoted as 0 and 1. In the initialization phase we randomly depute a certain number of ants. We then assign random pheromone values for all the links connecting the nodes in the path. The normalized probability is then calculated:

$$P_{ij} = \frac{\rho_{ij}}{\sum_j^k \rho_{ij}}$$

Here P_{ij} is normalized probability, ρ_{ij} is the individual node pheromone quantity in the path selected by the ant. Further, i is the current majority class instance and j is the majority class sample in the path which is yet to be visited (to be assigned either 0 or 1). Based on these initializations, the first ant will probabilistically select either node 0 or 1 of the first majority class sample. If 0 is selected, then the majority class instance is dropped else it is selected. In this manner, the ant will traverse the entire graph passing through each majority class sample and reaches the food source. All the selected samples of the majority class instances are collected and added to the

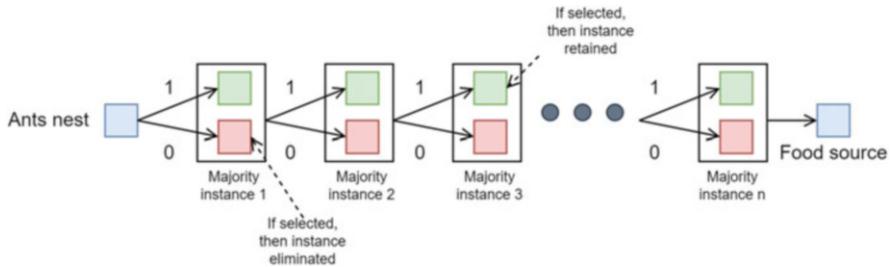


Fig. 14 Traversal process of ants from the nest towards the source

minority class instances. Figure 14 below shows the process of majority class selection using the process described above.

This transformed dataset is then fed to the Support Vector Classifier for model. Cross Validation training metrics are calculated and recorded. In this manner all the ants conduct their own tours along the path. Once this iteration is over, the pheromones of the nodes are updated thorough the entire graph. The update rule is as follows,

$$\rho_{ij}(n+1) = \epsilon * \rho_{ij}(n) + \Delta\rho_{ij}$$

Here, ϵ is the evaporation coefficient controlling the pheromone decrement and n is the current iteration number. $\Delta\rho_{ij}$ is the increased pheromone quantity of the top pathways selected. In this study, the authors add pheromone to the best 10% ants and store the pathways separately in a set M . $\Delta\rho_{ij}$ is mathematically defined as:

$$\Delta\rho_{ij} = \left\{ \frac{1}{(0.1 * no_ants)} * fitness, \text{for pathway } ij \in M \right. \\ \left. 0, \text{for pathway } ij \notin M \right\}$$

no_ants is the ant colony size. Further, the fitness constitutes of three weighted performance metrics namely F-measure, G-mean and AUC. Mathematically it is,

$$fitness = \mu * F - measure + \sigma * G - mean + \delta * AUC$$

Here, $\mu + \sigma + \delta = 1$.

These equations indicate that certain paths with better subset of majority class samples that provide better performance metrics will end up having enhanced pheromone trails. This trait ensures that these pathways will be chosen with higher probability. This iterative process is carried out for set number of iterations.

Overall, for the simulations, 100 different random partitions for test and train are utilized for average calculation. The authors have further compared different imbalance handling methods using four different metrics and four different datasets. The metrics used for this were, Accuracy, F-measure, G-mean and AUC. The four

datasets were colon, CNS, Lung and Glioma. Among all dataset simulations, it was observed that ACO based sampling provided better results compared to multiple undersampling techniques such as ORI, ROS, RUS and SMOTE. Specifically, the results of this study indicate that ACO performs best in all the four measures for lung dataset, in three measures for Glioma and Colon dataset and in two measures for CNS dataset.

5.4 Protein Data Imbalance Handling Using Binary Particle Swarm Optimization [100]

Glycosylation is a very important post translational modification of proteins. These modifications occur in eukaryotic cells. These are important for several molecular functions. Glycosylation occurs in about 50% of the proteins. A carbohydrate moiety gets linked to the hydroxyl group of serine and threonine residues in proteins. Computational prediction of O-glycosylation sites in mammalian proteins is challenging and has received considerable attention. The authors have proposed an approach utilizing a combination of PSO and RF for the prediction of O-glycosylation sites [100].

The protein sequences employed in this study had 2118 positive and 11,266 negative instances, which has an imbalance ratio of 0.188. Thus, This problem is quite imbalanced and heavily skewed in favour of negative sequences. The sequences in the data set had either Serine or Threonine at the center and the rest of the W-1 amino acids (W is the window size) were represented by the sparse coding scheme. As an example, Alanine was coded as 10000000000000000000000 and so on.

Binary PSO adapts a traditional PSO algorithm by explicitly solving problems with binary variables. Traditional PSO operates on continuous variables. Binary PSO develops the same concept for dealing with discrete variables. Binary strings represent these discrete variables. As Binary strings represent each particle here, each bit corresponds to a majority class example for this case study. The bit value is either 0 or 1, indicating the absence or presence of a particular example in the proposed solution. The primary step in binary PSO is to update the binary strings and the velocities. The constructed update rules here need to consider discrete nature and ensure that the generated solutions remain valid as binary strings. The velocity component update is similar to continuous PSO; however, the position update is slightly tweaked as it needs to handle binary strings. A widespread approach adopts the sigmoid function. Sigmoid can map the particle's velocity to a probability value. The bits at each position are later updated based on a comparison with a randomly generated number between 0 and 1. The termination conditions here are similar to that of a traditional PSO algorithm. The Algorithm for imbalance handling using PSO is as follows,

1. Initialize m number of particles. Each of these particles will have a dimension of n examples.
2. Next step is to select a carry out sample selection by generating random binary string vectors for each particle. The generated string will have either 0 or 1. Consider a scenario where in m = 5 and n = 8; We get the following population of particles.

$$\begin{aligned} i_1 &= 11010010 \\ i_2 &= 10001101 \\ i_3 &= 01110010 \\ i_4 &= 10101110 \\ i_5 &= 10010100 \end{aligned}$$

where i_1, i_2, i_3, i_4 and i_5 are the 5 particles each having a dimension of 8.

3. Undersampling can be now carried out for majority class instances by, considering only those majority class instances for which the binary bit is equal to one and discard the ones having value of zero. For the above considered scenario, considering vector i_1 , select examples one, two, four and seven only. For i_2 , consider one, five, six and eight only respectively. This data preparation step is done at step $t = 0$ timestamp.
4. Pass the considered examples along with the minority class examples to a machine learning algorithm and get the evaluation metrics. For example, in this study the classifier used is RF and the metric used is AUC.
5. Now we need to update the particles for the next iteration. The update rule is as follows.

$$Vel_i^{t+1} = W * Vel_i^t + c_1 * r_1 * (P_{best}^t - P_i^t) + c_2 * r_2 * (g_{best}^t - P_i^t)$$

Where,

- Vel_i^{t+1} is the velocity component used to calculate the next indicator function value at time stamp $t + 1$.
- W is the inertia parameter.
- r_1 and r_2 are random numbers between the value of 0 and 1.
- c_1 and c_2 are acceleration parameters.
- P_{best}^t is the best performing metric value for that example i till time stamp t . For $t = 0$ it will be the current evaluation metric value.
- g_{best} is the global best value of the evaluation metric for the m examples.
- P_i^t is the indicator function value for that example.

Based on the updated velocity, we need to find the following signum function, calculated as,

$$Sig(Vel_i^{t+1}) = \frac{1}{1 + e^{-Vel_i^{t+1}}}, \text{ where } Sig(Vel_i^{t+1}) \in [0, 1]$$

The position of the particle for the next iteration is calculated as follows,

Table 4 PSO setting parameters

Number of particles	20
Maximum iterations	100
Cognitive acceleration constant (c_1)	1.43
Social acceleration constant (c_2)	1.43
Inertia weight (w)	0.69

$$Pos_i^{t+1} = \begin{cases} 1, r_i < \text{Sig}(Vel_i^{t+1}) \\ 0, \text{otherwise} \end{cases}$$

Where r_i is a randomly generated number between 0 and 1. Now that we have Pos_i^{t+1} values for time stamp $t + 1$ we repeat the process.

The parameter values for the binary PSO were set as mentioned below in Table 4.

The proposed approach of binary PSO was combined with the Random Forest classifier for evaluating the quality of generated instances. Further this approach was compared with other widespread ML methods such as SVM, Adaboost, PSO and SVM. Considering all the approaches, the combination of PSO and RF outperformed all the other approaches in terms of better recorded AUC. The RF classifier had three variants of 100, 200 and 500 trees. The AUC recorded for all these variants was 0.95.

5.5 Application of ABC for Some Imbalanced Bioinformatics Datasets [35]

The authors benchmarked ABC on nine datasets possessing different levels of imbalance. Bioinformatics dataset which included Childhood Leukaemia [101] and five datasets from UCI repository [102] including Colon, Breast, Survival, Diabetes and blood were used for this task [35]. These datasets have varying degree of imbalance ratios between 1.34 and 3.16. They employed ABC algorithm for under sampling the majority class instances. The algorithm is similar to the one detailed in previous section, excepting (1) The randomly created input vectors are binary, 1 representing the majority class instances included and 0 representing majority class being not included. (2) In the employee bee stage finding the neighbourhood source involves a random process and real values. These real values are converted to bits through a sigmoid function. (3) At every iteration the majority class subsets selected were evaluated by SVM classifier. (4) K-fold cross validation methodology was used to maximize algorithm hyperparameters for large datasets and leave one out cross validation for small datasets. The results were compared against PSO and Random Under sampling methods. The authors also provide baseline imbalanced dataset results. The results indicate that for all the datasets the ABC algorithm was superior to the above-mentioned methods.

5.6 Data Imbalance Handling for Proinflammatory Peptides and Diabetes Identification Using Binary Black Hole Algorithm

Several peptides are now being routinely employed as therapeutic agents. Some of these peptides can induce undesirable inflammatory responses. It is very important to identify proinflammatory functions of peptides and regulate them. Thus, building accurate machine learning based models would be useful for this purpose. Bhosale et al. [103] employed distributed representation of protein sequences for this purpose. In this chapter we used the same dataset as that of [104] consisting of 729 proinflammatory peptides and 171 peptides with no proinflammatory activity. We have extracted dipeptide frequencies as attributes. We also took Pima Indian Diabetes dataset [105] from UCI for the same purpose. The diabetes dataset has 8 attributes and 768 data instances of which 500 are non-diabetics and 268 are diabetics. The dependent variable is 0 or 1, where 0 indicates a non-diabetic and 1 implies diabetic. Binary Black Hole Algorithm deals with binary classification problems where only two classes are present. It can be modified for carrying out undersampling of majority instances.

The first initialization step is to create a population of trial solutions. This population consists of N number of individual vectors called stars. The length of these vectors is equal to the number of majority class instances present in the dataset. Each star consists of randomly generated binary bits i.e., 0s and 1s. The selection or removal of majority instances from the dataset is indicated by 1 and 0 respectively. For example, consider a population pool consisting of 5 stars of length 8 each. This is depicted in the figure below.

Each star present in this population is a random subset of majority class instances from the imbalanced dataset. For example, based on star number 1 in above Fig. 15 instances number 1, 5 and 8 are selected while 2, 3, 4, 6 and 7 are discarded while formulating subset of first star. The entire algorithm steps involved are as follows:

1. Initialize population of stars (randomly generated binary vectors of bits 0 and 1). The length of vectors is equal to the number of majority class examples present in the imbalanced dataset.
2. For each star, extract the appropriate subset of majority class examples using the ones in the vector. Combine this subset with minority instances. Use these datasets to train a RF classifier and compute the respective MCC for each star.
3. Label the star with highest performance metric (MCC) as the Black Hole (BH).
4. Move the other stars by some distance towards this BH.
5. Evaluate the fitness again (MCC) using procedure explained in step 3.
6. Check for the highest fitness star and replace the BH accordingly.
7. Calculate Event Horizon (EH) step using:

Star 1	1	0	0	0	1	0	0	1
Star 2	1	0	0	1	1	0	1	1
Star 3	1	1	0	0	1	0	1	1
Star 4	1	0	1	0	1	0	0	1
Star 5	1	0	1	1	0	0	0	1

Fig. 15 Star population**Table 5** Binary Black Hole undersampling results

Performance metric	Binary Black Hole	Baseline
<i>Diabetes dataset</i>		
Accuracy	0.82	0.78
MCC	0.70	0.56
<i>Proinflam dataset</i>		
Accuracy	0.78	0.70
MCC	0.68	0.53

$$EH = \frac{f_{BH}}{\sum_{i=1}^N f_i}$$

Where, f_{BH} is the fitness of the BH and f_i is the fitness of the stars.

8. For each star calculate $\sqrt{(BH - S)^2}$. If this is greater than EH, discard the star and replace it with a newly generated random star. Evaluate fitness (MCC) as in 3.
9. Repeat steps 4 to 8 until convergence or set number of iterations.
10. The BH of final iteration is the best performing majority examples subset.

Based on this algorithm, majority class undersampling was carried out for solving class imbalance problem in the Diabetes [105] dataset and Proinflammatory peptides dataset [104]. Accuracy and MCC were the metrics which were recorded. Results are highlighted in Table 5. We can observe that the Binary Black hole provides better results compared to the imbalance baseline results for both datasets.

5.7 Data Imbalance Handling for Phase Separating Proteins Identification Using Combination of Generative Adversarial Networks and Binary Black Hole Algorithms

We employed a combination of Generative Adversarial Networks and Binary Black Hole Algorithm for handling imbalance in phase separating proteins. Phase separation proteins are very important in several biological functions which include bacterial division and tumorigenesis. Recently Bhosale et al. [103] employed the distributed representation of protein sequences both in the original form and in the reduced alphabet form for identification of liquid -liquid phase separation proteins. We have used the same dataset employed by them. The dataset consists of 400 attributes with total of 536 sequences in which 120 sequences belonged to Phase separation proteins and 416 belonged to the negative sequences.

Generative Adversarial Networks (GANs), introduced by Ian Goodfellow and his colleagues in 2014 [106], have revolutionized the field of artificial intelligence by enabling synthetic data generation across various domains [107, 108]. GAN works with the aim of employing adversarial learning to generate high quality synthetic data. A standard configuration of GAN consists of two conventional neural networks, viz., the generator and the discriminator. Starting with a random noise GAN iteratively synthesizes better and better-quality data similar to the input data distribution. While the generator does this by trying to fool the discriminator, the latter tries to improve the model by discriminating the original training data. The effect is a zero-sum game, and the equations are elaborated in [106].

We have employed a combination of a simple GAN along with the recently introduced Black Hole (BH) metaheuristic algorithm. Initially we used the minority instances present in the dataset to train the GAN network and generate synthetic minority class examples, we employed the GAN algorithm explained above for this purpose. The frequency of minority instances to be generated is purposefully kept 20–30% more than is required to balance the dataset. BH is subsequently used to find the best performing subset of these newly created synthetic minority instances (which balances the dataset). We used the BH algorithm as explained in the previous section. We also compared our approach with the standard oversampling approach of SMOTE.

The results for Phase separating proteins identification [20] are shown in Table 6. From these results we can see that GAN-BH balancing performance is superior to that of SMOTE balancing. Both the results are improved after balancing. Based on

Table 6 Results of oversampling using GAN-BH approach

Metric evaluation	Without balancing	After SMOTE balancing	After GAN-BH balancing
Accuracy	0.85	0.88	0.93
Sensitivity	0.95	0.97	0.99
Specificity	0.28	0.35	0.67
MCC	0.47	0.53	0.70

the results below we can notice that the GAN-BH combination provided superior results compared to simple SMOTE based upsampling. The GAN-BH algorithm provided better accuracy, sensitivity, specificity as well as improved MCC over the baseline and SMOTE results.

5.8 Some Imbalance Related Examples in Bioinformatics

Here, we briefly describe a set of imbalance related examples in bioinformatics. This includes the datasets utilized, the methods adopted by the researchers and the performance obtained. Illustrates some case studies which provides information about imbalance handling in bioinformatics.

6 Conclusion

In this chapter we have explained the use of evolutionary and metaheuristic algorithms in handling data imbalance in the bioinformatics domain. These algorithms are nature inspired, robust. These simple but effective algorithms have been employed for handling several potentially important problems occurring in several subareas of bioinformatics. These imbalance related problems are essentially machine learning based models. Apart from briefly explaining important machine learning algorithms along with different performance measures we have explained the working principles of general imbalance handling nature inspired algorithms which include Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony Algorithm, and the Black Hole Algorithm. These methods have been profusely used synergistically with Machine Learning algorithms. We have further explained how these methods convert the data imbalance into optimization problems. We have provided important case studies to illustrate how the combined algorithms balance the dataset and improve performance.

References

1. Koonin EV. Computational genomics. *Curr Biol*. 2001;11(5):R155–8.
2. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18, 2015. Springer International Publishing; 2015. p. 234–41.
3. Duncan JS, Insana MF, Ayache N. Biomedical imaging and analysis in the age of big data and deep learning [scanning the issue]. *Proc IEEE*. 2019;108(1):3–10.
4. Gunter TD, Terry NP. The emergence of national electronic health record architectures in the United States and Australia: models, costs, and questions. *J Med Internet Res*. 2005;7(1):e383.

5. Su C, Tong J, Zhu Y, Cui P, Wang F. Network embedding in biomedical data science. *Brief Bioinform.* 2020;21(1):182–97.
6. Mirza B, Wang W, Wang J, Choi H, Chung NC, Ping P. Machine learning and integrative analysis of biomedical big data. *Genes.* 2019;10:87.
7. Auslander N, Gussow AB, Koonin EV. Incorporating machine learning into established bioinformatics frameworks. *Int J Mol Sci.* 2021;22:2903.
8. Solis-Reyes S, Avino M, Poon A, Kari L. An open-source k-mer based machine learning tool for fast and accurate subtyping of HIV-1 genomes. *PLoS One.* 2018;13:e0206409.
9. Thomas S, Karnik S, Barai RS, Jayaraman VK, Idicula-Thomas S. CAMP: a useful resource for research on antimicrobial peptides. *Nucleic Acids Res.* 2010;38(Suppl_1):D774–80.
10. Modak S, Mehta S, Sehgal D, Valadi J. Application of support vector machines in viral biology. In: Shapshak P, Balaji S, Kangueane P, Chiappelli F, Somboonwit C, Menezes LJ, Sinnott JT, editors. *Global virology III: virology in the 21st century.* Cham: Springer; 2019. p. 361–403.
11. Li Y, Huang C, Ding L, Li Z, Pan Y, Gao X. Deep learning in bioinformatics: introduction, application, and perspective in big data era. *bioRxiv.* 2019.
12. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat Methods.* 2015;12(10):931.
13. Wang S, Peng J, Ma JZ, Xu JB. Protein secondary structure prediction using deep convolutional neural fields. *Sci Rep.* 2016;6:18962.
14. Li Y, Wang S, Umarov R, Xie B, Fan M, Li L, Gao X. Deepre: sequence-based enzyme EC number prediction by deep learning. *Bioinformatics.* 2018;34(5):760–9.
15. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature.* 2017;542(7639):115–8.
16. Ma J, Yu MK, Fong S, Ono K, Sage E, Demchak B, Sharan R, Ideker T. Using deep learning to model the hierarchical structure and function of a cell. *Nat Methods.* 2018;15(4):290.
17. Zitnik M, Agrawal M, Leskovec J. Modeling polypharmacy side effects with graph convolutional networks. 2018;arXiv preprint arXiv:1802.00543.
18. Hu J, Li Y, Zhang M, Yang X, Shen H-B, Yu D-J. Predicting protein-DNA binding residues by weightedly combining sequence-based features and boosting multiple SVMs. *IEEE/ACM Trans Comput Biol Bioinform.* 2017;14:1389–98.
19. Ding J, Zhou S, Guan J. MiRenSVM: towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features. *BMC Bioinform.* 2010;11:S11.
20. Lahorkar A, Bhosale H, Sane A, Ramakrishnan V, Jayaraman VK. Identification of phase separating proteins with distributed reduced alphabet representations of sequences. *IEEE/ACM Trans Comput Biol Bioinform.* 2022;20(1):410–20.
21. Wei Z-S, Yang J-Y, Shen H-B, Yu D-J. A cascade random forests algorithm for predicting protein-protein interaction sites. *IEEE Trans Nanobioscience.* 2015;14:746–60.
22. Wei Z-S, Han K, Yang J-Y, Shen H-B, Yu D-J. Protein–protein interaction sites prediction by ensembling SVM and sample-weighted random forests. *Neurocomputing.* 2016;193:201–12.
23. Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G. Learning from class-imbalanced data: review of methods and applications. *Expert Syst Appl.* 2017;73:220–39.
24. He H, Garcia EA. Learning from imbalanced data. *IEEE Trans Knowl Data Eng.* 2008;1263–84.
25. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority oversampling technique. *J Artif Intell Res.* 2002;16:321–57.
26. Calvet L, Benito S, Juan AA, Prados F. On the role of metaheuristic optimization in bioinformatics. *Int Trans Oper Res.* 2023;30(6):2909–44.
27. Goldberg DE. *The design of innovation: lessons from and for competent genetic algorithms*, vol. 1. Boston: Kluwer Academic Publishers; 2002.
28. Ye FL, Lee CY, Lee ZJ, Huang JQ, Tu JF. Incorporating particle swarm optimization into improved bacterial foraging optimization algorithm applied to classify imbalanced data. *Symmetry.* 2020;12(2):229.

29. Tahir MAUH, Asghar S, Manzoor A, Noor MA. A classification model for class imbalance dataset using genetic programming. *IEEE Access*. 2019;7:71013–37.
30. Yu H, Ni J, Zhao J. ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data. *Neurocomputing*. 2013;101:309–18.
31. Zheng D, Qin C, Liu P. Adaptive particle Swarm optimization algorithm ensemble model applied to classification of unbalanced data. *Sci Program*. 2021;2021:1–13.
32. Lopez-Garcia P, Masegosa AD, Osaba E, Onieva E, Perallos A. Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. *Appl Intell*. 2019;49(8):2807–22.
33. Cheng J, Chen J, Guo YN, Cheng S, Yang L, Zhang P. Adaptive CCR-ELM with variable-length brain storm optimization algorithm for class-imbalance learning. *Nat Comput*. 2021;20: 11–22.
34. Haque MN, Noman N, Berretta R, Moscato P. Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification. *PLoS One*. 2016;11(1): e0146116.
35. Braytee A, Hussain FK, Anaiisi A, Kennedy PJ. ABC-sampling for balancing imbalanced datasets based on artificial bee colony algorithm. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). IEEE; December 2015. p. 594–99.
36. Ganji MF, Abadeh MS, Hedayati M, Bakhtiari N. Fuzzy classification of imbalanced data sets for medical diagnosis. In: 2010 17th Iranian Conference of Biomedical Engineering (ICBME). IEEE; November 2010. p. 1–5.
37. Karia V, Zhang W, Naeim A, Ramezani R. Gensample: a genetic algorithm for oversampling in imbalanced datasets. 2019;arXiv preprint arXiv:1910.10806.
38. Zeng X, Yuan S, Huang X, Zou Q. Identification of cytokine via an improved genetic algorithm. *Front Comp Sci*. 2015;9:643–51.
39. Lim P, Goh CK, Tan KC. Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning. *IEEE Trans Cybernet*. 2016;47(9):2850–61.
40. Pei W, Xue B, Shang L, Zhang M. Genetic programming for high-dimensional imbalanced classification with a new fitness function and program reuse mechanism. *Soft Comput*. 2020;24:18021–38.
41. García-López S, Jaramillo-Garzón JA, Higuita-Vásquez JC, Castellanos-Domínguez CG. Wrapper and filter metrics for PSO-based class balance applied to protein subcellular localization. *Bioinformatics*. 2012;2012:214–9.
42. Li J, Fong S, Mohammed S, Fiaidhi J. Improving the classification performance of biological imbalanced datasets by swarm optimization algorithms. *J Supercomput*. 2016;72(10): 3708–28.
43. Yang P, Xu L, Zhou BB, Zhang Z, Zomaya AY. A particle swarm-based hybrid system for imbalanced medical data sampling. *BMC Genomics*. 2009;10:1–14.
44. Sarker IH. Machine learning: algorithms, real-world applications and research directions. *SN Comput Sci*. 2021;2:160.
45. Barua A, Ahmed MU, Begum S. A systematic literature review on multimodal machine learning: applications, challenges, gaps and future directions. *IEEE Access*. 2023;11:14804–31.
46. Angermueller C, Pärnamaa T, Parts L, Stegle O. Deep learning for computational biology. *Mol Syst Biol*. 2016;12:878.
47. Ker J, Wang L, Rao JP, Lim TC. Deep learning applications in medical image analysis. *IEEE Access*. 2018;6:9375–89.
48. Razzak MI, Naz S, Zaib A. Deep learning for medical image processing: overview, challenges and future. 2017; arxiv, abs/1704.06825.
49. Redshaw J, Ting DS, Brown A, Hirst J, Gärtner T. Krein support vector machine classification of antimicrobial peptides. *Digit Discov*. 2023;2:502–11.
50. Roy A, Chakraborty S. Support vector machine in structural reliability analysis: a review. *Reliab Eng Syst Saf*. 2023;233:109126.

51. Elshewey AM, Shams MY, El-Rashidy N, Elhady AM, Shohieb SM, Tarek Z. Bayesian optimization with support vector machine model for Parkinson disease classification. *Sensors (Basel)*. 2023;23:2085.
52. Bawa A, Samanta S, Himanshu SK, Singh J, Kim J, Zhang T, Chang A, Jung J, DeLaune PB, Bordovsky JP, Barnes EM, Ale S. A support vector machine and image processing based approach for counting open cotton bolls and estimating lint yield from UAV imagery. *Smart Agric Technol*. 2022;3:100140.
53. Kesav N, M.G, J. A deep learning approach with Bayesian optimized Kernel support vector machine for Covid-19 diagnosis. *Comput Methods Biomed Eng Imaging Vis*. 2022;11:623–37.
54. Seyedmohammadi J, Zeinadini A, Navidi MN, McDowell RW. A new robust hybrid model based on support vector machine and firefly meta-heuristic algorithm to predict pistachio yields and select effective soil variables. *Ecol Inform*. 2023;74:102002.
55. Breiman L. Random forests. *Mach Learn*. 2001;45:5–32.
56. Shaheed K, Szczuko P, Abbas Q, Hussain A, Albatthan M. Computer-aided diagnosis of COVID-19 from chest X-ray images using hybrid-features and random forest classifier. *Healthcare*. 2023;11:837.
57. Alice K, Deepa N, Devi T, BeenaRani BB, Bharatha Devi N, Nagaraju V. Effect of multi filters in glaucoma detection using random forest classifier. *Meas Sens*. 2023;25:100566.
58. Stojadinovic MM, Milićević B, Jankovic S. Improved prediction of significant prostate cancer following repeated prostate biopsy by the random forest classifier. *J Med Biol Eng*. 2022;43: 83–92.
59. Sun Z, Wang G, Li P, Wang H, Zhang M, Liang X. An improved random forest based on the classification accuracy and correlation measurement of decision trees. *Expert Syst Appl*. 2023;237(18):121549.
60. Chicco D, Jurman G. The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification. *BioData Mining*. 2023;16:4.
61. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*. 2020;21:6.
62. Zhu QA. On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset. *Pattern Recogn Lett*. 2020;136:71–80.
63. Cohen G, Hilario M, Sax H, Hugonnet S, Geissbuhler A. Learning from imbalanced data in surveillance of nosocomial infection. *Artif Intell Med*. 2006;37(1):7–18.
64. Yang Q, Wu X. 10 challenging problems in data mining research. *Int J Inf Technol Decis Mak*. 2006;5(04):597–604.
65. Vajda S, Fink GA. Strategies for training robust neural network based digit recognizers on unbalanced data sets. In: 2010 12th International Conference on Frontiers in Handwriting Recognition 2010 Nov 16. IEEE; 2010. p. 148–53.
66. Kirui C, Hong L, Kirui E. Handling class imbalance in mobile telecoms customer churn prediction. *Int J Comput Appl*. 2013;72(23):7–13.
67. Ding Z. Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics. Doctoral dissertation, Georgia State University. 2011.
68. Hido S, Kashima H, Takahashi Y. Roughly balanced bagging for imbalanced data. *Stat Anal Data Min*. 2009;2(5–6):412–26.
69. Kerdprasop K, Kerdprasop N. A data mining approach to automate fault detection model development in the semiconductor manufacturing process. *Int J Mech*. 2011;5(4):336–44.
70. Nálevka P, Svátek V. Improving efficiency of telemedical prevention programs through data-mining on diagnostic data. In: 4th International Conference on Bioinformatics and Biomedical Technology IPCBEE 2012 (Vol. 29). 2012.
71. Zhang Y, Wang D. A cost-sensitive ensemble method for class-imbalanced datasets. *Abstr Appl Anal*. 2013;2013:196256.
72. Weiss GM. Mining with rarity: a unifying framework. *ACM Sigkdd Explor Newslett*. 2004;6 (1):7–19.

73. Bekkar M, Djemaa HK, Alitouche TA. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl.* 2013;3(10):27–38.
74. Sharma S, Gosain A, Jain S. A review of the oversampling techniques in class imbalance problem. In: International conference on innovative computing and communications: proceedings of ICICC 2021, vol. 1. Singapore: Springer; 2022. p. 459–72.
75. Chawla N, Bowyer K, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. 2002;arxiv, abs/1106.1813.
76. Han H, Wang WY, Mao BH. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing. Berlin: Springer; 2005. p. 878–87.
77. He H, Bai Y, Garcia EA, Li S. ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE; June 2008. p. 1322–28.
78. Chadaga K, Prabhu S, Vivekananda Bhat K, Umakanth S, Sampathila N. Medical diagnosis of COVID-19 using blood tests and machine learning. *J Phys.* 2022;2161(1):012017.
79. Polce EM, Kunze KN. A guide for the application of statistics in biomedical studies concerning machine learning and artificial intelligence. *Arthroscopy.* 2023;39(2):151–8.
80. Neely BA, Dorfer V, Martens L, Bludau I, Bouwmeester R, Degroeve S, Deutsch EW, Gessulat S, Käll L, Palczynski P, Payne SH, Rehfeldt TG, Schmidt T, Schwämmle V, Uszkoreit J, Vizcaíno JA, Wilhelm M, Palmblad M. Toward an integrated machine learning model of a proteomics experiment. *J Proteome Res.* 2023;22:681–96.
81. Nazari E, Pourali G, Khazaei M, Avan A, Asadnia A, Dashtiahangar M, Mohit R, Maftooh M, Nassiri MT, Hassanian SM, Ghayour-Mobarhan M, Ferns GA, Shahidsales S. Identification of potential biomarkers in stomach adenocarcinoma using machine learning approaches. *Curr Bioinforma.* 2023;18(4):320–33.
82. Chakraborty S, Mali K. An overview of biomedical image analysis from the deep learning perspective. In: Research Anthology on Improving Medical Imaging Techniques for Analysis and Intervention. Hershey: IGI Global; 2020. p. 43–59.
83. Cox J. Prediction of peptide mass spectral libraries with machine learning. *Nat Biotechnol.* 2022;41:33–43.
84. Lambora A, Gupta K, Chopra K. Genetic algorithm—a literature review. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). 2019. p. 380–84.
85. Su FC, Wu WL. Design and testing of a genetic algorithm neural network in the assessment of gait patterns. *Med Eng Phys.* 2000;22(1):67–74.
86. Alba E. Cellular genetic algorithms. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. 2014.
87. Selvanathan N, Tee WJ. A genetic algorithm solution to solve the shortest path problem in OSPF and MPLS. *Malays J Comput Sci.* 1970;16:58–67.
88. Verma A. A survey on image contrast enhancement using genetic algorithm. *Int J Sci Res Publ.* 2012;2(7):1–5.
89. Colorni A, Dorigo M, Maniezzo V. An investigation of some properties of an “Ant Algorithm”. In: Ppsn 1992 Sep 28 (Vol. 92, No. 1992).
90. Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life (Vol. 142). 1991. p. 134–42.
91. Shmygelska A, Hoos HH. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinform.* 2005;6:1–22.
92. Zhang X, Chen X, He Z. An ACO-based algorithm for parameter optimization of support vector machines. *Expert Syst Appl.* 2010;37(9):6618–28.
93. Uğur A, Aydin D. An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms. *Adv Eng Softw.* 2009;40(5):341–9.
94. Duan H, Yu Y, Zhang X, Shao S. Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. *Simul Model Pract Theory.* 2010;18(8):1104–15.

95. Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. MHS'95. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. 1995. p. 39–43.
96. Batista GE, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newslett. 2004;6(1):20–9.
97. Yang P, Xu L, et al. A particle swarm based hybrid system for imbalanced medical data sampling. BMC Genomics. 2009;10(Suppl 3):S34.
98. Abualigah LM, Elaziz ME, Sumari P, Khasawneh AM, Alshinwan M, Mirjalili S, Shehab M, Abuaddous H, Gandomi AH. Black hole algorithm: a comprehensive survey. Appl Intell. 2022;52:11892–915.
99. Kumar S, Datta D, Singh SK. Black hole algorithm and its applications. In: Azar A, Vaidyanathan S, editors. Computational intelligence applications in modeling and control. Cham: Springer; 2015. p. 147–70.
100. Hassan HA, Abdelhalim MB, Badr A. Prediction of O-glycosylation sites in proteins using PSO-based data balancing and random forest. Bioinform Biol Insights. 2015;9:103–9.
101. <http://pob.abcc.ncifcrf.gov/cgibin/JK>.
102. <http://archive.ics.uci.edu/ml>.
103. Bhosale H, Sane A, Ramakrishnan V, Jayaraman VK. Distributed reduced alphabet representation for predicting proinflammatory peptides. In: International conference on data management, analytics & innovation. Singapore: Springer Nature Singapore; 2023. p. 161–73.
104. Gupta S, Madhu MK, Sharma AK, Sharma VK. ProInflam: a webserver for the prediction of proinflammatory antigenicity of peptides and proteins. J Transl Med. 2016;14(1):178. <https://doi.org/10.1186/s12967-016-0928-3>.
105. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the Symposium on Computer Applications and Medical Care. IEEE Computer Society Press; 1988. p. 261–65.
106. Goodfellow I, et al. Generative adversarial nets. Adv Neural Inf Proces Syst. 2014;27:1–9.
107. Yoon J, et al. Gain from the generator: novel synthetic data for deep anomaly detection. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.
108. Nam S, Kim Y, Kim SJ. Text-adaptive generative adversarial networks: manipulating images with natural language. 2018;arxiv, abs/1810.11919.