People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Mohamed El Bachir El Ibrahimi University of Bordj Bou Arréridj

Faculty of Mathematics and Computer Science

Department of Computer Science

THESIS

In order to obtain the Doctorate degree in LMD (3rd cycle)

Branch: Computer Science

Option: Ingénierie de l'informatique décisionnelles

THEME

Intelligent algorithms for feature selection in supervised and unsupervised classification

By: Khaoula Zineb Legoui

Publicly defended on (13/01/2026)

In front of the jury composed of:

Pr.	Farid Nouioua	University of Bordj Bou Arreridj	President
Pr.	Abdelouahab Attia	University of Bordj Bou Arreridj	Supervisor
Dr.	Sofiane Maza	University of Bordj Bou Arreridj	Co-Supervisor
Pr.	Allaoua Hemmak	University of M'Sila	Examiner
Dr.	Lyazid Sabri	University of Bordj Bou Arreridj	Examiner
Dr.	Abdelbasset Barkat	University of M'Sila	Examiner

Acknowledgments

First and foremost, all praise and gratitude are due to Allah 'Alhamdulillah', whose granted me success and surrounded me with His care throughout this work and my academic journey.

I would like to express my deepest appreciation to my supervisors, Pr. Abdelouahab Attia and Maza Sofiane, for their continuous support, insightful feedback, and patience throughout every stage of this research. I am especially grateful for their understanding during the final stages, in light of the changes and preoccupations that limited my availability. Their guidance not only helped shape the direction of this thesis but also inspired my growth as a researcher.

I am grateful to Pr. Essam H. Houssein for his contribution and the valuable feedback he provided, which greatly enriched this work.

Special thanks go to my colleagues and fellow researchers, especially those who reviewed my work and provided valuable discussions and support during the development and publication of the research papers included in this thesis.

To my family, thank you for your unwavering belief in me. Your emotional support, sacrifices, and prayers have been my foundation. Without your strength, this journey would not have been possible. To my friends and everyone who believed in me and encouraged me along the way—thank you.

Lastly, I extend my thanks to everyone who contributed—directly or indirectly—to the completion of this thesis. Your presence, feedback, and motivation are sincerely appreciated.

Table of Contents

List of Tables

List of Figures

Abstract

The growing availability of high-dimensional datasets across various domains has made feature selection a critical step in machine learning pipelines, as reducing irrelevant or redundant features enhances model interpretability and generalization. Due to the combinatorial nature of feature selection, traditional methods often lack the scalability and adaptability required for real-world problems. In response, this thesis investigates the application of intelligent metaheuristic algorithms to feature selection in both supervised and unsupervised learning settings. First, a comparative analysis of the Equilibrium Optimizer (EO) and Henry Gas Solubility Optimization (HGSO) algorithms is conducted for supervised classification tasks. Both algorithms are adapted to a binary feature space and evaluated on benchmark datasets using classification accuracy and feature reduction as performance criteria, highlighting their respective strengths and motivating a hybrid approach. Consequently, this thesis proposes HGSOEO, a hybrid algorithm that integrates the complementary exploration and exploitation capabilities of HGSO and EO. The proposed HGSOEO algorithm is evaluated on the Twitter Spam Detection dataset and demonstrates superior performance in terms of classification accuracy and the number of selected features when compared to conventional metaheuristic and classical feature selection methods. Furthermore, the application of EO is extended to feature selection for clustering tasks, where labeled data are unavailable, by employing clustering validity criteria such as the Adjusted Rand Index (ARI) to guide the selection process. Experimental results across multiple datasets confirm the effectiveness and robustness of the proposed approaches. Overall, the findings of this thesis demonstrate that intelligent metaheuristic algorithms provide efficient and scalable solutions to the feature selection problem in both supervised and unsupervised learning contexts.

Keywords: Feature Selection, Intelligent Algorithms, Metaheuristic Algorithms, Equilibrium Optimizer (EO), Henry Gas Solubility Optimization (HGSO), Hybrid Algorithms, Classification, Clustering

Résumé

La disponibilité croissante d'ensembles de données à haute dimension dans divers domaines a rendu la sélection de caractéristiques (feature selection) une étape cruciale dans les pipelines d'apprentissage automatique, car la réduction des caractéristiques redondantes ou non pertinentes améliore l'interprétabilité et la capacité de généralisation des modèles. En raison de la nature combinatoire de la sélection de caractéristiques, les méthodes traditionnelles manquent souvent de scalabilité et d'adaptabilité pour résoudre des problèmes réels. Cette thèse explore l'application des algorithmes métaheuristiques intelligents à la sélection de caractéristiques dans des contextes supervisés et non supervisés. Tout d'abord, une analyse comparative de l'Equilibrium Optimizer (EO) et de l'Henry Gas Solubility Optimization (HGSO) est effectuée pour des tâches de classification supervisée. Les deux algorithmes sont adaptés à un espace de caractéristiques binaires et évalués sur des ensembles de données de référence en utilisant l'exactitude de classification et la réduction des caractéristiques comme critères de performance, mettant en évidence leurs forces respectives et justifiant le développement d'une approche hybride. Par conséquent, cette thèse propose HGSOEO, un algorithme hybride combinant les capacités complémentaires d'exploration et d'exploitation de HGSO et EO. HGSOEO est évalué sur l'ensemble de données Twitter Spam Detection et démontre des performances supérieures en termes d'exactitude de classification et de nombre de caractéristiques sélectionnées par rapport aux méthodes métaheuristiques conventionnelles et aux méthodes classiques de sélection de caractéristiques. De plus, EO est étendu à la sélection de caractéristiques pour des tâches de clustering où les données ne sont pas étiquetées, en utilisant des critères de validité de clustering tels que l'indice de Rand ajusté (ARI) pour guider le processus de sélection. Les résultats expérimentaux obtenus sur plusieurs ensembles de données confirment l'efficacité et la robustesse des approches proposées. Dans l'ensemble, cette thèse démontre que les algorithmes métaheuristiques intelligents constituent une solution efficace et évolutive à la problématique de la sélection de caractéristiques dans des contextes d'apprentissage supervisé et non supervisé.

Mots-clés: Sélection des attributs, Algorithmes métaheuristiques, Optimiseur d'Équilibre (EO), Optimisation de la Solubilité des Gaz de Henry (HGSO), Algorithmes hybrides, Classification, Regroupement.

الملخص

أدى التوفر المتزايد لمجموعات بيانات عالية الأبعاد في مجالات متعددة إلى جعل اختيار الميزات خطوة حاسمة في أنظمة التعلم الآلي، حيث إن تقليل الميزات غير الضرورية أو المتكررة يعزز من قابلية تفسير النموذج وقدرته على التعميم. وبسبب الطبيعة التوافقية لمشكلة اختيار الميزات، غالبًا ما تفتقر الطرق التقليدية إلى القدرة على التوسع والمرونة المطلوبة لمعالجة المشاكل الواقعية. تبحث هذه الرسالة في تطبيق الخوارزميات الميتاهيوريستية الذكية لاختيار الميزات في سياقات التعلم المراقب وغير المراقب. أولاً، تم إجراء تحليل مقارن لخوارزميتي Equilibrium Optimizer (EO) و Henry Gas Solubility Optimization (HGSO) لمهام التصنيف المراقب. تم تكييف كلا الخوارزميتين للعمل في فضاء ميزات ثنائي وتم تقييمهما على مجموعات بيانات مرجعية باستخدام دقة التصنيف وتقليل الميزات كمقاييس للأداء، مما يبرز نقاط القوة لكل منهما ويبرر تطوير نهج هجين. بناءً عليه، تقترح هذه الرسالة HGSOEO، وهي خوارزمية هجينة تجمع بين القدرات التكميلية للاستكشاف والاستغلال لكل من HGSO و EO. تم تقييم HGSOEO على مجموعة بيانات Twitter Spam Detection وأظهرت الخوارزمية أداءً متفوقًا من حيث دقة التصنيف وعدد الميزات المختارة مقارنةً بالطرق الميتاهيوريستية التقليدية وطرق اختيار الميزات الكلاسيكية. بالإضافة إلى ذلك، تم توسيع استخدام EO لاختيار الميزات لمهام التجميع(Clustering) حيث لا تتوفر بيانات موسومة، باستخدام معايير صحة التجميع مثل مؤشر راند المعدل (ARI)لتوجيه عملية اختيار الميزات. تؤكد النتائج التجريبية على عدة مجموعات بيانات فعالية وموثوقية الأساليب المقترحة. بشكل عام، تؤكد نتائج هذه الرسالة أن الخوارزميات الميتاهيوريستية الذكية توفر حلولاً فعالة وقابلة للتوسع لمشكلة اختيار الميزات في سياقات التعلم المراقب وغير المراقب.

الكلمات المفتاحية: اختيار الميزات، الخوارزميات الذكية، الخوارزميات فوق الاستدلالية، خوارزمية التوازن، خوارزمية ذوبانية غاز هنري، الخوارزميات الهجينة، التصنيف، التجميع .

Chapter 1

Introduction

1.1 Context

Huge increase in data has been witnessed recently because of - technological progress and advancement. In today's world, high dimensional data science is revolutionising various fields like computer vision, bioinformatics, business analytics, healthcare, finance, social media, fraud detection and many more [1][2]. In these fields, they are now observing high dimensional datasets which has a large number of features. Many of those features might be noisy, irrelevant, or redundant. To make sense of the data and to discover meaningful patterns, rules and knowledge from the data, scientists apply data mining techniques such as classification and clustering. These techniques are crucial for extracting valuable insights from large datasets and making informed decisions across various sectors.

Nonetheless, the enormity and complexity of contemporary datasets pose serious challenges. The learning algorithms may slow down, consume more memory, and overfit in the presence of unnecessary features which degrades the performance of the model. Additionally, it may make the outputs difficult to interpret and explain, which is a major issue in some areas. This is the reason behind making feature selection (FS) a necessary and basic step in a machine learning pipeline[3].

Feature selection is a process of selecting and retaining the most pertinent features from the

initial feature set, while simultaneously eliminating those with minimal or adverse effects on the performance of a machine learning model[4]. In contrast to feature extraction, which converts features into new representations, feature selection retains the original meaning of the features, hence making it highly applicable in areas where interpretability is of paramount concern[5][6].

Feature selection is not merely a preprocessing step; it is indeed crucial to enhance the efficiency, accuracy, and generalizability of machine learning models. In supervised learning tasks, such as classification, feature selection serves to enhance the performance of classifiers by accentuating informative patterns. When used in unsupervised learning, such as clustering, feature selection improves the cohesion of cluster formation by eliminating noisy and irrelevant dimensions that could otherwise bias similarity measures.[7]

Yet, the problem of finding the best subset of features in a dataset is an NP-hard combinatorial optimization problem with an exponentially increasing search space relative to the number of features[8]. This inherent complexity renders exhaustive search approaches infeasible with increasing sizes and complexities of datasets. In order to combat this difficulty, researchers have more often utilized advanced optimization methods, specifically metaheuristic algorithms, which are capable of effectively searching large search spaces by achieving a good balance between exploration and exploitation.

Metaheuristic algorithms, which include, for example, Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and more recently like the Equilibrium Optimizer (EO) and Henry Gas Solubility Optimization (HGSO), have proven effectively their ability in solving the FS problem[9][10][8][11]. These algorithms mimic natural phenomena to direct the search toward optimal/near-optimal solution sets. Moreover, their flexibility, and the ability to avoid local optima make them excellent choice for the most optimization problems, like Feature Selection.

1.2 Research Problem and Motivations

Despite the growing success of metaheuristic algorithms in solving feature selection (FS) problems, several critical challenges remain. Feature selection is a well-known NP-hard combinatorial optimization problem due to the exponential growth of the search space with increasing feature dimensions [8]. Exhaustive search methods quickly become impractical for large datasets, and many traditional FS techniques fail to maintain performance when faced with noise, redundancy, or limited supervision[12].

Moreover, while various metaheuristic algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have been applied to FS, they often show inconsistent performance across different learning contexts [13]. Recent methods like the Equilibrium Optimizer (EO) [14] and Henry Gas Solubility Optimization (HGSO) [15] offer promising characteristics, but their application to FS remains relatively underexplored—particularly in hybrid settings and unsupervised learning. Furthermore, many real-world datasets are unlabeled. In such cases, supervised FS methods are insufficient. There is a growing need for intelligent FS techniques that also perform well in unsupervised contexts, improving clustering quality without relying on labeled instances.

Based on these gaps, this research aims to address the following questions:

- How to Efficiently Use and Adapt Recents Intelligent Algorithms (EO, HGSO) for FS in a Various Classification Case?
- Can we fully implement a hybrid approach for better performance?
- How well do these techniques work on both real and benchmark data sets?
- How to use this Intelligent Algorithms for FS in the clustering problems?

The increasing need for precise, interpretable, and efficient machine learning models has made intelligent feature selection algorithms a popular research focus. The motivation for

this thesis comes from the need to design, implement, and evaluate intelligent metaheuristic algorithms for the FS problem in various learning paradigms. By producing new algorithmic solutions to this problem and conducting exhaustive benchmarking on real and synthetic datasets, this thesis aims to improve the state of the art of intelligent feature selection.

1.3 Contributions

This thesis presents a series of contributions toward tackling the problems of feature selection in both supervised and unsupervised learning contexts. The work is organized into three phases, each of which explains and extends recent advances in metaheuristic optimization.

- Comparative Study for Classification: The first contribution is the comprehensive comparative study of two recent metaheuristic algorithms: the Equilibrium Optimizer (EO) and the Henry Gas Solubility Optimization (HGSO). The two algorithms were modified to fit the binary search space of feature selection and compared using several supervised classification benchmarks included Random Forest, Naïve Bayes and KNN. The comparison was conducted across ten datasets, evaluating both classification accuracy and feature reduction. The findings indicated that HGSO significantly outperformed EO in classification accuracy and feature reduction. This comparison provided valuable insights into the behavior of each algorithm and identified opportunities for improvement. This study is presented in [16].

- Proposed Hybrid Model (HGSOEO): the second major contribution is the design and implementation of a novel hybrid metaheuristic algorithm, referred to as HGSOEO. The algorithm integrates EO's convergence-oriented mechanism with HGSO's group-based exploration dynamics. Hybridization was pursued with the purpose of attaining a better balance between global exploration and local exploitation. HGSOEO was tested on a twitter spam detection dataset and its accuracy was compared against its constituents separately (EO and HGSO) and alongside other popular metaheuristics.

The results showed that HGSOEO consistently had better classification accuracy with smaller feature subsets. This validated the notion that the combination of techniques can lead to performance that cannot be attained by single algorithms. This contribution is detailed in [17].

- Feature Selection for Clustering: The third contribution takes the application of EO to the unsupervised learning scenario, more specifically feature selection for clustering tasks. Labeled data may not be available in real-world situations, and hence unsupervised FS is an important research topic. EO was modified to work without label information, and its effect on clustering performance was measured using clustering methods like Gaussian Mixture Models (GMM). The Adjusted Rand Index (ARI) measures was employed to measure performance. The studies showed that EO-based FS enhanced clustering cohesion and separation and also minimized the number of features. This study paves the way for the use of metaheuristic-based FS techniques in unsupervised scenarios, which is a less researched field of label-free dimensionality reduction. The approach and findings are presented in [18].

All these stages show that intelligent optimization algorithms can overcome high-dimensional problems in various learning applications.

1.4 Thesis Organization

The thesis is organized as follow:

- **Chapter 1: Introduction -** This chapter provides the general background of the research, identify the research problem and motivation, state the key contributions, and give an overview of the thesis. We also enumerate the academic publications resulting from this research.

- **Chapter 2: Feature Selection in Machine Learning -** This chapter gives a thor-

ough background of the feature selection problem, its significance, and procedure. It presents intelligent algorithms with emphasis on metaheuristics and describes their general framework. The chapter also shows how these algorithms are developed and modified to tackle the feature selection problem, such as binary encoding, fitness function, and evaluation measures.

- **Chapter 3: Contributions -** This chapter outlines the thesis's original contributions. These consist of a comparative analysis of the Equilibrium Optimizer (EO) and Henry Gas Solubility Optimization (HGSO) algorithms for supervised feature selection, the development of a hybrid HGSOEO algorithm and its validation using real-world data, and the extension of EO to feature selection for unsupervised clustering problems.

- **Chapter 4: Results and Analysis -** This chapter reports and analyzes the experimental results on benchmark and real datasets, including comparisons of performance with state-of-the-art algorithms, stability and convergence analysis, and evaluations with various classifiers and clustering metrics. The effectiveness and scalability of the proposed approaches are elaborated in detail.

- **Chapter 5: Conclusion and Future Work -** This chapter finalizes the thesis by recapitulating the key findings as well as contributions. It also suggests some future research directions, such as extensions to the hybrid algorithm and continued investigation of unsupervised feature selection methods.

1.5 Academic Publications

- LEGOUI, Khaoula Zineb, MAZA, Sofiane, et ATTIA, Abdelouahab. Equilibrium optimizer and henry gas solubility optimization algorithms for feature selection: comparison study. In : 2022 5th International Symposium on Informatics and its Applications (ISIA). IEEE, 2022. p. 1-6.

- LEGOUI, Khaoula Zineb, MAZA, Sofiane, ATTIA, Abdelouahab, et HOUSSEIN, Essam H. Hybrid Henry gas solubility optimization and the equilibrium optimizer for feature selection: real cases with Twitter spam detection. Knowledge and Information Systems, 2024, vol. 66, no 5, p. 3055-3084.

- LEGOUI, Khaoula Zineb, ATTIA, Abdelouahab, et MAZA, Sofiane. Equilibrium Optimizer for Feature Selection in Clustering. In : 2024 International Conference of the African Federation of Operational Research Societies (AFROS). IEEE, 2024. p. 1-5.

Chapter 2

Feature Selection in Machine Learning

2.1 Introduction

Feature selection (FS) is a critical process in enhancing the performance, efficiency, and interpretability of machine learning models by selecting and keeping only the most significant features. Due to its combinatorial nature, FS is usually addressed by metaheuristic algorithms, which are known for their strong search in high-dimensional spaces. This chapter provides an overview of classical feature selection (FS) methods and introduces intelligent algorithms—specifically metaheuristics—that have gained attention for their effectiveness in solving FS problems. It also presents an Intelligent Algorithm for the Feature Selection Problem, including the problem formulation, solution encoding, fitness functions used, and evaluation criteria.

2.2 Feature Selection

Feature selection is one of the core preprocessing techniques in machine learning and data analysis that is especially crucial when dealing with high-dimensional data. Its primary objective is to enhance predictive performance, mitigate overfitting, and facilitate model interpretability by choosing a subset of the most informative and relevant features from the original feature set. Such dimensionality reduction not only lowers computational expense

but also removes noise and redundancy, enabling learning algorithms to concentrate on the most significant patterns in the data.

2.2.1 Feature Selection Techniques

Feature selection techniques are typically divided into two broad categories: filter and wrapper methods. The two categories adopt a different approach to feature evaluation and selection, and the decision among them is based on the characteristics of the dataset, the available computational budget, and the target application[19][20].

- **Filter based (independent criteria)** Filter methods assess the quality of feature subsets exclusively in terms of their inherent characteristics, without involving any learning algorithm. They use statistical or information-theoretic metrics like correlation coefficients or mutual information to rank and pick features. Since they are not tied to any particular classifier, filter approaches are computationally efficient and can scale to extremely high-dimensional data sets, and can be applied prior to selecting a specific model. However, their main limitation is that they do not account for how features interact within a specific learning algorithm, possibly leading to poor performance for some models[21][22][7].

- **wrapper based (dependent criteria)** Wrapper methods employ a predictive model to evaluate feature subsets, identifying those that provide optimal performance for a designated classifier. These methods iteratively train and assess the model with various subsets, following a search strategy that may include forward selection, backward elimination, or metaheuristic optimization. Although wrapper methods frequently generate superior outcomes compared to filter methods by taking into account feature interactions, they tend to be computationally intensive, particularly when applied to large datasets[22][7].

2.2.2 Feature Selection Procedure

The feature selection procedure generally involves four primary phases, as shown in Figure 2.1: subset generation, subset evaluation, stopping criteria, and subset validation. The process iterates between generating candidate feature subsets and evaluating their quality according to a specific criterion. Once the stopping condition is met—whether based on time, performance plateau, or iteration limit—the best-performing subset is finally validated using a learning model or evaluation measure.

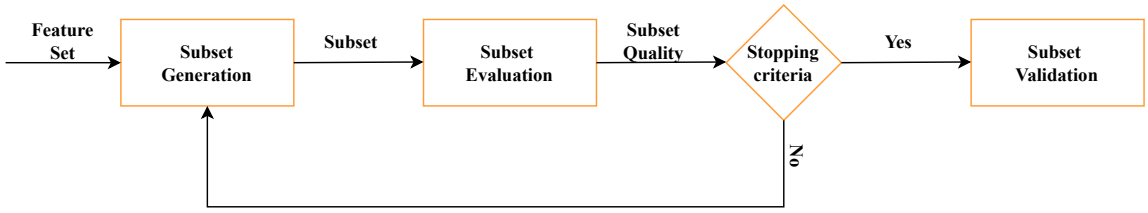

Figure 2.1: General procedure of Feature Selection[23].

2.2.2.1 Subset generation

In this step, the key element is the formation of a feature subset. This involves selecting a subset of features from the overall set of features. The number of candidate subsets is related to the number of features, where there are 2^N possible subsets for a dataset with N features. There are three types of strategies to search for subsets. Complete search, Sequential Search, and Random search.

- **Complete search** is an exhaustive search. It calculates the performance of all possible subsets of features in order to select which set lead to the best model performance. Complete search is guaranteed to have the best result because of considering all possible subsets starting from one feature to all available features. Yet, it is very computationally expensive. This is impractical to do for large datasets, since the number of subsets is growing exponentially with a number of features.

- **Sequential search** is an incremental process for feature selection. There are three variations for this type: forward selection, backward selection, and hybrid selection. The first method starts with no features and in each iteration adds a feature, testing its impact on the performance; it stops when the adding of features no longer improves the performance. In contrast, the second method starts with all features and removes a feature in each iteration until further removal doesn't improve the performance. The last method is the combination of the two types.

- **Random search** is a different process that follows a different search method for feature subsets compared to the complete or sequential search. It doesn't use systematic exploration of all subsets or a logical progression through them but uses the randomness as a primary key that means It selects combinations at random and evaluates them. The process is repeated for a certain number of iterations or until a stopping criterion is met. Although it may not always reach the optimum solution, it is considered the best in terms of computational time, especially for a high number of features, unlike complete or sequential search, which is high.

2.2.2.2 Subset evaluation

In this step, the subsets generated during the previous step have to be evaluated in order to assess their quality based on evaluation criteria. The goodness of a subset of features depends on the chosen criterion, which means the good choice of evaluation metric plays a crucial role, as it directly impacts the effectiveness of the feature selection process. The choice of evaluation method depends on the type of feature selection method 2.2.1:

- **Filter methods** use statistical scores such as correlation, information gain, or mutual information without involving a classifier.

- **Wrapper methods** evaluate subsets by training a specific learning algorithm and measuring the performance.

2.2.2.3 Stopping criteria

A stopping criterion is a condition that is defined under which the feature selection process stops. This will prevent the process from continuing unnecessarily, saving time and avoiding the overfitting issue. This step is no less important than the previous ones, as it directs the process towards the best result without going through the evaluation of all the combinations. Among the conditions that can be determined are:

- Maximum number of iterations.
- Maximum number of features.
- No further improvement in performance.
- Thresholding performance.

2.2.2.4 Subset validation

The results can be validated in two ways, depending on the type of data. For synthetic datasets or well-studied domains where we work on a dataset that we already have prior knowledge of the subset of features that we want to reach, we directly compare the selected features with the known subset. Moreover, knowing which features are irrelevant or redundant allows us to ascertain the validity of the feature selection algorithm by verifying whether these have been excluded from the final subset. For real-world datasets, it is rare to have prior knowledge about which features are important. In such cases, the validation of the subset is done by observing how the selected features impact the model performance. This involves comparing the performance of the model before and after selecting the features (using all features versus using the subset of features).

2.3 Intelligent Algorithms

After outlining the principles, types, and procedures of feature selection, this section focuses on the notion intelligent algorithms and their application in solving optimization problems, particularly in feature selection. It initially provides the background of metaheuristic algorithms, followed by their general framework to clarify their fundamental constituents and workflow. Then, it gives comprehensive explanations of two algorithms utilized in this thesis, namely the Equilibrium Optimizer (EO) and the Henry Gas Solubility Optimization (HGSO), which are employed as the main optimization methods in the proposed methods.

2.3.1 Metaheuristic algorithms

The field of computer science identifies a large group of optimization problems that includes the feature selection (FS) dilemma as NP-hard[24][25]. The classification indicates these problems become unsolvable easily through existing deterministic methods when dimensions increase. The number of features directly correlates to an exponential increase in search space dimensions, making exhaustive search methods impractical for computational use. Often, finding an exact solution is practically impossible within realistic time constraints. To overcome those issues, researchers use heuristic and metaheuristic algorithms as modern tools to achieve effective approximate solutions through intelligent search process guidance. Heuristic algorithms can efficiently solve certain complex and advanced types of optimization problems. However, due to their reliance on specific domain knowledge, they are not generally applicable. In contrast, meta-heuristic algorithms have more flexibility through a general adaptable structure. Metaheuristics are problem-independent and therefore applicable to a wide range of problems. Their general design principles, along with stochastic search methods, help it efficiently search different complex solution spaces[26].

Metaheuristic algorithms can be broadly categorized into two main classes: single-solution-based and population-based approaches [27, 28]. Single-solution-based metaheuristics, such

as Simulated Annealing and Tabu Search, operate by iteratively improving a single candidate solution [28, 29]. In contrast, population-based metaheuristics, including Genetic Algorithms, Particle Swarm Optimization, and swarm-intelligence-based methods, evolve a set of candidate solutions simultaneously [30, 31, 32]. Population-based approaches are particularly effective for complex and multi-modal optimization problems due to their inherent diversity and parallel search capability [33]. The search process of any metaheuristic algorithm relies on two key functions that guide the search effectively: exploration and exploitation[8][34].

- **Exploration:** enables the algorithm to investigate new and diverse areas of the search space, thereby avoiding premature convergence to local optima.
- **exploitation:** focuses on refining the best solutions found so far to converge toward an optimal or near-optimal solution.

The algorithm requires proper alignment between exploration and exploitation to achieve its best results. The convergence rate slows down when exploration exceed limits and excessive focus on exploitation might lead to stagnant performance at mediocre solutions. High-dimensional feature selection remains a challenging task due to feature redundancy, noise, and complex interdependencies. Traditional approaches, such as filter and wrapper methods, often suffer from poor scalability and a tendency to become trapped in local optima when dealing with large and complex search spaces. Consequently, recent research has increasingly focused on metaheuristic algorithms, as they provide flexible and effective solutions for hard optimization problems. These algorithms are typically inspired by natural phenomena, including biological evolution, social behavior, and physical processes. As illustrated in Fig.2.2, metaheuristic algorithms can be further classified based on their source of inspiration [26, 35, 36].

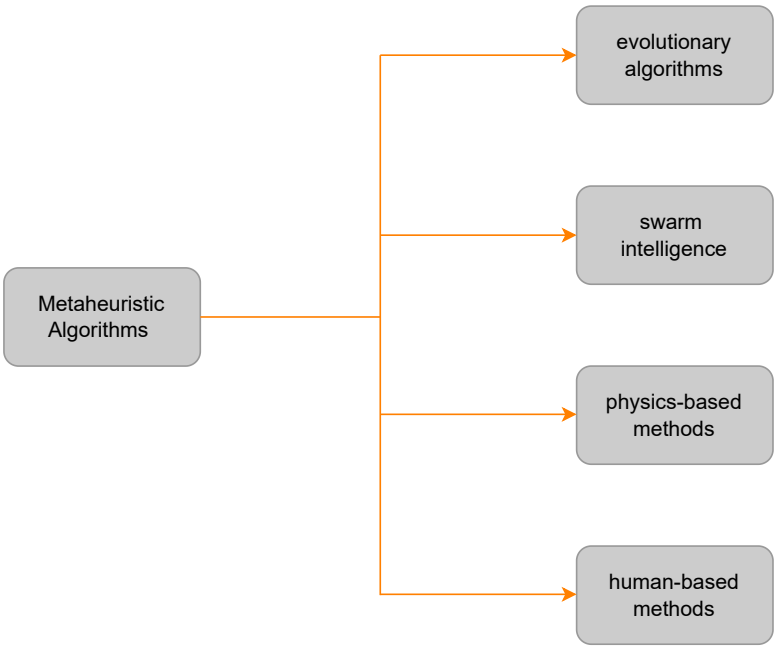

Figure 2.2: Metaheuristic algorithms categories[37].

These algorithms employ probabilistic search mechanisms to iteratively refine candidate solutions. In population-based metaheuristics, multiple candidate solutions evolve simultaneously through fitness-based selection and adaptive operators such as crossover, mutation, velocity updates, or pheromone intensification. This intrinsic diversity enables an effective balance between exploration and exploitation, significantly increasing the likelihood of identifying optimal or near-optimal feature subsets in complex and multi-modal search spaces.

2.3.2 General framework of metaheuristic algorithms

Metaheuristics are generally classified as population-based stochastic search algorithms used to solve complex optimization problems. The majority of these algorithms follow a common mechanism aimed at identifying optimal or near-optimal solutions. This mechanism typically consists of several essential components: parameter setting, population initialization, global search, local refinement, and stopping criteria[38].

- **Parameter Setting:** involves all configurations of the algorithm parameters, included population size, learning rates, exploration/exploitation balance (not all algorithms require parameter tuning). Proper tuning at this stage significantly influences convergence speed and solution quality.

- **Population Initialization:** involves the generation of an initial population of solution candidates (usually randomly generated) used to start the search procedure.

- **Global Search:** This element governs the searching of the solution space through the application of position or solution updates founded upon interactions, random disturbances, or information exchange among individuals.

- **Local Search:** There are algorithms that have mechanisms for refining solutions through the exploitation of neighborhoods around hopeful candidates. This refines convergence accuracy and prevents premature convergence

- **Stopping Conditions:** The algorithm continues the search until a stopping criterion is satisfied, i.e., until a maximum number of iterations or a desired level of fitness is reached.

This general framework enables metaheuristic algorithms to be modified and implemented for a broad range of optimization problems. In following, two particular metaheuristic algorithms—Equilibrium Optimizer (EO) and Henry Gas Solubility Optimization (HGSO)—are introduced in their original versions.

2.3.3 Equilibrium Optimizer (EO)

The Equilibrium Optimizer (EO) is a metaheuristic algorithm inspired by physics, which was introduced by Faramarzi et al. [14]. It is motivated by the mass balance dynamics of a control volume within a fluid system, in which concentrations attempt to achieve an equilibrium condition by diffusion and mixing processes.

EO models the candidate solutions as concentrations that move towards equilibrium following a model based on a concentration update rule inspired by physical laws. The algorithm uses a memory pool of the best concentrations, referred to as the equilibrium candidates, to guide the population.

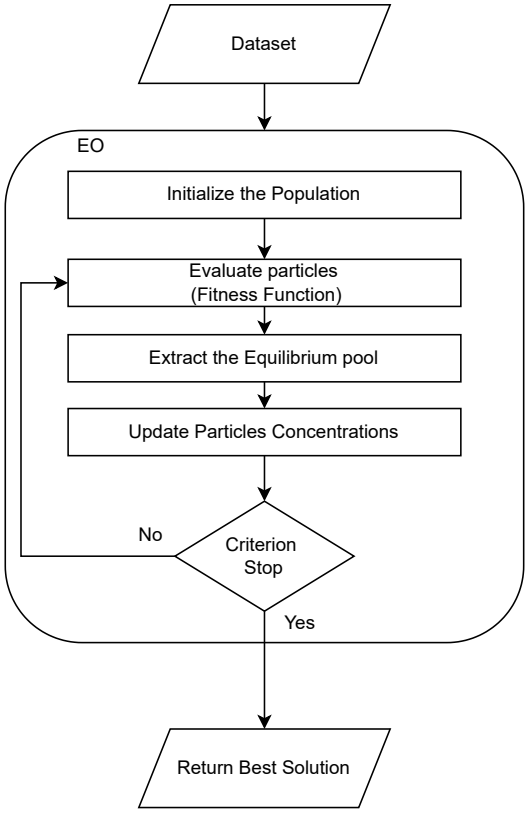

Figure 2.3: EO Framework Model.

a) Initialization

In EO, each candidate solution is represented by a concentration vector $\vec{C}$. The population is initialized randomly within the problem's bounds:

$$C_i = C_{min} + rand \times (C_{max} - C_{min}) \tag{2.1}$$

where C_i is the concentration (position) of the i^{th} particle, and $rand$ is a uniform random number in the range $[0, 1]$.

b) Fitness Evaluation

Each candidate solution is evaluated using a predefined objective (fitness) function appropriate for the problem domain. The goal is to minimize or maximize this function depending on the task.

c) Equilibrium Pool Formation

The top four solutions with the best fitness values are selected to form the equilibrium pool. Additionally, the average of these four solutions is computed and added to the pool, creating five guiding vectors in total. These represent potential equilibrium states for the population.

d) Equilibrium Update Mechanism

Candidate solutions are iteratively updated based on a mathematical model inspired by mass balance equations. The update formula is defined as:

$$C_i^{t+1} = C_{eq} + (C_i^t - C_{eq}) \cdot \vec{F} + \frac{\vec{G}}{\lambda \vec{V}} \cdot (1 - \vec{F}) \tag{2.2}$$

where:

$$\vec{F} = a_1 \cdot \text{sign}(\vec{r} - 0.5) \cdot (e^{\lambda t} - 1) \tag{2.3}$$

$$\vec{G} = \vec{G_0} \cdot \vec{F} \tag{2.4}$$

$$\vec{G_0} = G\vec{C}P \cdot (\vec{C_{eq}} - \lambda \vec{C}) \tag{2.5}$$

$$G\vec{C}P = \begin{cases} 0.5 \cdot r_1, & \text{if } r_2 \geq GP \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

Where:

- C_i^t: concentration i at iteration t,
- C_{eq}: selected equilibrium candidate from the pool,
- $\vec{F}$: control vector to balance exploration and exploitation,
- $\vec{G}$: generation rate vector,
- $G\vec{C}P$: generation control probability,
- $\vec{V}$: a unit vector,
- r_1, r_2: random numbers uniformly distributed in $[0, 1]$,
- GP: generation probability.

This update mechanism guides concentrations toward one of the equilibrium states while preserving exploration.

e) Termination

The update process is repeated until a termination condition is satisfied,such as reaching a maximum number of iterations, a target fitness value, or stagnation in solution improvement. The best solution found during the process is returned as the final output.

Algorithm 1 Pseudo code of the EO algorithm.

Population initialization using (2.1), $i = 1, 2, 3, ...N$
Set a large fitness values to the equilibrium candidates
Set the parameters of EO $a_1 = 2$; $a_2 = 1$; $GP = 0.5$
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of solutions (N)) **do**
        Calculate fitness of i^{th} particle
        if $fit(\vec{C}_i) < fit(\vec{C}_{eq1})$ **then**
            Replace $\vec{C}_{eq1}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq1})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2})$ **then**
            Replace $\vec{C}_{eq2}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq2})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq3})$ **then**
            Replace $\vec{C}_{eq3}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq3})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq3}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq4})$ **then**
            Replace $\vec{C}_{eq4}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq4})$ with $fit(\vec{C}_i)$
        end if
    end for
    $\vec{C}_{evg} = (\vec{C}_{eq1} + \vec{C}_{eq2} + \vec{C}_{eq3} + \vec{C}_{eq4})/4$
    Construct the equilibrium pool $\vec{C}_{eq1}, \vec{C}_{eq2}, \vec{C}_{eq3}, \vec{C}_{eq4}, \vec{C}_{evg}$
    Accomplish memory saving ($if Iter > 1$).
    Assign $t = (1 - \frac{Iter}{Max\_Iter})^{(a_2 \frac{Iter}{Max\_Iter})}$.
    for ($i = 1$ to number of particles (N)) **do**
        Randomly choose one candidate from the equilibrium pool.
        Generate random vectors of $\vec{\lambda}, \vec{r}$
        Construct $\vec{F}, G\vec{C}P, \vec{G}_0$, and $\vec{G}$ using the following equations respectively (2.3), (2.6), (2.5), (2.4).
        Update concentration $\vec{C}$ using (2.2).
    end for
    Iter = Iter + 1
end while
return

2.3.4 Henry Gas Solubility Optimization (HGSO)

Henry Gas Solubility Optimization (HGSO) is a population-based metaheuristic algorithm derived from Henry's law in chemistry, which governs the solubility of a gas in a liquid under pressure. Proposed by Hashim et al. [15], HGSO takes the behavior of gas molecules as they dissolve in a liquid medium and employs this analogy to develop a search mechanism

for global optimization problems.

The algorithm splits the population into a number of groups, with each group being a different gas type. Candidate solutions traverse the search space by modifying their solubility and partial pressure, converging toward optimum solutions through interaction among group members and the best so-far solution.

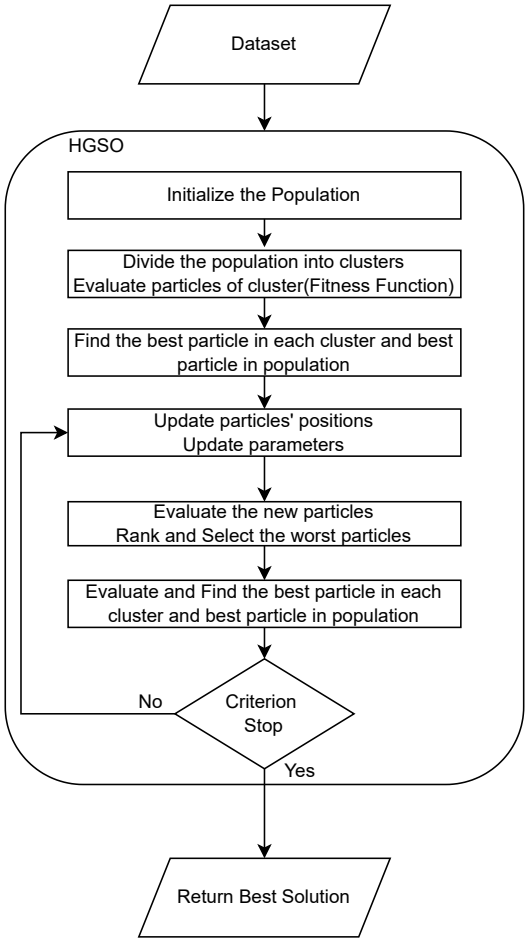

Figure 2.4: HGSO Framework Model.

a) Initialization and Clustering

The Henry Gas Solubility Optimization (HGSO) algorithm begins by randomly generating a population of candidate solutions, where each solution is treated as a gas

molecule encoded in binary form to represent a feature subset. The initial positions of these particles, along with key parameters—namely Henry's constant H_j, partial pressure $P_{i,j}$, and solubility constant C_j—are initialized as follows:

$$X_i = X_{\min} + \text{rand} \times (X_{\max} - X_{\min}) \tag{2.7}$$

$$H_j(t) = l_1 \times \text{rand}, \quad P_{i,j} = l_2 \times \text{rand}, \quad C_j = l_3 \times \text{rand} \tag{2.8}$$

where $l_1 = 5 \times 10^{-2}$, $l_2 = 100$, and $l_3 = 10^{-2}$. The population is then grouped into equal-sized clusters, with each cluster representing a distinct gas type, all sharing the same Henry's constant.

b) Fitness Evaluation

Each cluster is evaluated independently to determine the best-performing gas particle within it. Additionally, the best particle across all clusters is identified. The fitness of each solution is computed using a predefined fitness function suited to the nature of the task.

c) Standard HGSO Update Mechanism

HGSO employs a physics-inspired update mechanism based on Henry's law and solubility principles. This involves updating the Henry's constant H_j, computing the solubility $S_{i,j}$, and adjusting the particle positions. The updates are governed by the following formulas:

$$\begin{aligned} H_j^{t+1} &= H_j \times \exp\left[-C_j \left(\frac{1}{T^t} - \frac{1}{T^\theta}\right)\right], \\ T^t &= \exp\left(\frac{t}{\text{Max\_Iter}}\right) \end{aligned} \tag{2.9}$$

$$S_{i,j}^t = K \times H_j^{t+1} \times P_{i,j}^t \tag{2.10}$$

$$
\begin{aligned}
X_{i,j}^{t+1} &= X_{i,j}^{t} + F \cdot r \cdot \gamma \cdot (X_{i,j}^{t} - X_{j,best}^{t}) \\
&\quad + F \cdot r \cdot \alpha \cdot (S_{i,j}(t) \cdot X_{best}^{t} - X_{i,j}^{t}) \\
\gamma &= \beta \cdot \exp\left(-\frac{F_{best}^{t} + \epsilon}{F_{i,j}^{t} + \epsilon}\right), \quad \epsilon = 0.05
\end{aligned}
\tag{2.11}
$$

Where:

- $X_{i,j}^{t}$: position of particle i in cluster j at iteration t,
- $X_{j,best}^{t}$: best particle in cluster j,
- X_{best}^{t}: global best particle in the population,
- F: a control flag that dictates search direction,
- $S_{i,j}^{t}$: solubility of particle i in cluster j,
- H_j: Henry's coefficient for gas type j,
- $P_{i,j}$: partial pressure acting on particle i in cluster j.

These equations ensure a physically inspired behavior that balances intensification (exploitation) and diversification (exploration) in the search process.

d) Local Optima Escape

To enhance diversity and escape local minima, a portion of the worst-performing solutions is reinitialized using:

$$
N_w = N \cdot (\text{rand} \cdot (c_2 - c_1) + c_1), \quad c_1 = 0.1, \quad c_2 = 0.2 \tag{2.12}
$$

$$
G_{i,j} = G_{\min(i,j)} + r \cdot (G_{\max(i,j)} - G_{\min(i,j)}) \tag{2.13}
$$

This helps maintain exploration capability, preventing premature convergence by introducing randomness in poorly performing agents.

e) Termination

The algorithm iteratively repeats the above steps until a termination criterion is met, such as reaching a maximum number of iterations, achieving a threshold fitness value, or observing no further improvement. The best-performing particle found during the entire process is then returned as the final solution.

Algorithm 2 Pseudo code of the HGSO algorithm.

population initialization using (2.7), $i = 1, 2, 3, ...N$, number of gas types i.
set the parameters of HGSO l_1,l_2,l_3 and initialize the properties H_j,$P_{i,j}$,C_j using (2.8).
Divide the population of gas particles into a number of gas types (cluster) with the same Henry's constant value H_j.
Evaluate each cluster j.
Get the best gas particle $X_{i,best}$ in each cluster, and the best particle in population X_{best} .
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of particles (N)) **do**
        Update the positions of each gas particle using (2.11).
    end for
    Update Henry's coefficient of each gas type using (2.9).
    Update solubility of each gas particle using (2.10).
    Rank and select the number of worst particles using (2.12).
    Update the position of the worst agents using (2.13).
    Update the best gas particle $X_{i,best}$, and the best search agent gas particle in the population X_{best}.
    Iter = Iter + 1
end while
return X_{best}

2.4 Intelligent Algorithm for Feature Selection Problem

This section is devoted to the adaptation of intelligent algorithms to the feature selection problem specifically. The main objective is to present how metaheuristic algorithms, originally designed for continuous optimization, can be reformulated and adapted to address binary feature selection tasks. We discuss problem formulation and the fitness function

guiding the search process, followed by solution codification, binary adaptation techniques, and the performance metrics used for evaluation.

2.4.1 Problem Formulation and Fitness Function

Feature selection is one of the key pre-processing steps in machine learning and data mining in general. It becomes more important when the dataset is high-dimensional. The aim is to find a minimal set of relevant features that maintains the performance, or optimally improves the performance, of learning algorithms. The problem can be framed as a combinatorial optimization problem associated with finding the optimal feature subset in the set of all features provided.

The objective is to select a subset of features so that $S \subseteq F$ where $F = \{f_1, f_2, f_3, ..., f_d\}$ and d is the number of features, such that this subset guarantees the following optimization goals :

- Maximize the classification (or clustering) performance, and
- Minimize the number of selected features.

In this work, the FS task is formulated in both supervised classification and unsupervised clustering contexts, with different fitness functions guiding the optimization in each case.

2.4.1.1 Supervised Classification – Fitness Function

Formally, the problem can be formed as a bi-objective optimization task but in this research we agregate them to uni-objective:

$$Fitness(S) = \alpha \cdot Error(S) + \beta \cdot \frac{|S|}{|F|} \tag{2.14}$$

Here:

$$\text{Error}(S) = 1 - Accuracy(S)$$

- $Accuracy(S)$ denotes the classification accuracy obtained using the feature subset S. The detailed formulation of accuracy is presented in Section 2.4.4,
- $Error(S)$ denotes the classification error using the subset S,
- $\frac{|S|}{|F|}$ represents the proportion of selected features, where $|S|$ is the number of selected features and $|F|$ is the total number of original features,
- α and β are the weight parameters that control the trade-off between classification performance and feature reduction.

In this study, the weighting parameters are empirically set to $\alpha = 0.99$ and $\beta = 0.01$. This choice reflects a strong emphasis on maximizing classification performance while still encouraging feature reduction. Such a configuration is particularly suitable for classification tasks where predictive accuracy is the primary objective.

2.4.1.2 Unsupervised Clustering – Fitness Function

In unsupervised learning, class labels are not utilized in the training process and feature selection itself is evaluated based on the quality of the clusters. In this case the Adjusted Rand Index (ARI) is used to compare the clustering result to the true class labels, which are accessible only for evaluation purposes. In this case the fitness function is defined as:

$$Fitness(S) = 1 - ARI(S) \tag{2.15}$$

Here, $ARI(S)$ denotes the Adjusted Rand Index after clustering with Gaussian Mixture Model (GMM)[39] using the selected features S. A lower fitness value indicates better clustering performance (i.e., higher ARI score). This mono-objective formulation directs the optimization algorithm to find feature subsets that maximize the quality of clusters. The detailed formulation of the ARI is presented in Section 2.4.4.

2.4.2 Solution Codification

In all metaheuristic algorithms implemented in this study each candidate solution (also named as individual, particle, or agent) is represented by a binary vector of length d, where d is the number of total features in the data set.

The state of each bit in the binary vector signifies whether a corresponding feature is included or excluded:

$$S = [f_1, f_2, f_3, ..., f_d], f_i \in \{0, 1\}$$

- $f_i = 1$: The i^{th} feature is selected.
- $f_i = 0$: The i^{th} feature is excluded.

The binary encoding reformulates the feature selection problem as a combinatorial search problem in the 2^d space of possible subsets. It is particularly well-fitted for the binary form of metaheuristic algorithms, which allows for good exploration and exploitation of the search space.

The population P is then composed of N such binary vectors where each vector S_i represents a candidate solution:

$$P = [S_1, S_2, S_3, ..., S_N]$$

Table 2.1 illustrates the representation of the population in binary form.

Table 2.1: Population representation.

Features / Subsets	f_1	f_2	f_3	$\cdots$	f_d
S_1	0	1	$\ldots$	0	0
S_2	1	0	$\ldots$	1	1
S_3	1	0	$\ldots$	0	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
S_N	1	1	$\ldots$	1	0

2.4.3 Binary Adaptation for Feature Selection

To apply the proposed methods to binary feature selection, a transfer function is used to convert the continuous solution values into binary ones. The transfer function is applied to each dimension as follows:

$$S(C_{i,j}^{\text{bin}}) = \left| \frac{C_{i,j}}{\sqrt{1 + C_{i,j}^2}} \right|$$
$$C_{i,j}^{\text{bin}} = \begin{cases} 1, & \text{if } S(C_{i,j}) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

This binary codification enables the algorithms to generate candidate feature subsets, where each bit indicates whether a specific feature is selected (1) or excluded (0).

2.4.4 Evaluation Metrics

In order to comprehensively compare the performance of the proposed feature selection algorithms, some standard evaluation metrics are employed. They are employed to assess both the predictive accuracy of the obtained feature subsets and the effectiveness in reducing dimensionality. The evaluations are conducted under supervised and unsupervised learning scenarios.

2.4.4.1 Classification Performance Metrics

In supervised learning settings, the following classification metrics are used to evaluate how well the selected feature subsets preserve class separability:

- **Accuracy:** Measures the proportion of correctly classified instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

- **Precision:** Represents the proportion of true positive predictions among all positive predictions made.

$$Precision = \frac{TP}{TP + FP} \tag{2.18}$$

- **Recall (Sensitivity):** Indicates the proportion of actual positives that were correctly identified.

$$Recall = \frac{TP}{TP + FN} \tag{2.19}$$

- **Specificity:** Measures the proportion of actual negatives that were correctly identified.

$$Specificity = \frac{TN}{TN + FP} \tag{2.20}$$

- **AUC (Area Under the ROC Curve):** Reflects the classifier's ability to distinguish between classes, with values ranging from 0.5 (random) to 1 (perfect). A higher AUC indicates better classification performance across thresholds.

These metrics provide a comprehensive understanding of the model's behavior.

2.4.4.2 Clustering Performance Metric

For unsupervised learning tasks, the Adjusted Rand Index is used as a validation metric. Let:

- **a** be the number of pairs of points in the same cluster in both partitions.
- **b** be the number of pairs in different clusters in both partitions.
- **c** and **d** account for mismatched pairings.

The Rand Index (RI) is defined as:

$$Rand\ Index\ (RI) = \frac{a + b}{a + b + c + d} \tag{2.21}$$

The Adjusted Rand Index is computed as:

$$ARI = \frac{RI - Expected\ RI}{(Max\ RI - Expected\ RI)} \tag{2.22}$$

2.5 Conclusion

This chapter reviewed the foundations of feature selection and introduced intelligent metaheuristic algorithms as effective solutions for high-dimensional problems. It also presented the general structure of such algorithms, their adaptation to feature selection tasks, and the key components required for implementation. In the following chapter, the focus shifts to the development and implementation of the metaheuristic-based proposed methods for feature selection.

Chapter 3

Contributions

3.1 Introduction

This chapter presents the main contributions of this thesis, which aim to improve feature selection using metaheuristic optimization techniques. The work focuses on three directions: a comparative study between Equilibrium Optimizer (EO) and Henry Gas Solubility Optimization (HGSO), the design of a hybrid EO-HGSO algorithm for classification tasks, and the application of EO for feature selection in clustering problems.

3.2 Equilibrium Optimizer and Henry Gas Solubility Optimization Algorithms for Feature Selection: Comparison Study

3.2.1 Equilibrium Optimizer for Feature Selection

In order to adapte the Equilibrium Optimizer (EO) algorithm for the feature selection problem, a set of modifications was implemented to transform the original continuous optimizer to a discrete binary variant for subset feature selection. The transformation included designing a binary solution representation, with each individual in the population representing

a binary vector that expresses the inclusion or exclusion of features. A transfer function was used to translate continuous EO updates into binary decisions, which ensured that the search process was still effective in the discrete feature space. The fitness function was also tailored to jointly assess the classification accuracy and the feature reduction rate so that EO can manage the trade-off between performance and dimensionality. These adaptations allowed the EO algorithm to efficiently search through the combinatorial search space of feature selection problems.

a) Initialization

The algorithm begins by initializing using 2.1 a population of concentration vectors, represents a potential subset of features encoded as a binary string of length d (number of features).

This modification enables the conversion of vector values into binary format

$$C_{i,j} = \begin{cases} 1, & \text{if } S(C_{i,j}) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

b) Fitness Evaluation

The evaluation of each particle is performed using the fitness function mentioned in Chapter.3 (Eq.2.14), based on two objectives: the classification error and the number of features considered. Since this is a mono-objective function, the resultant particles are guided to converge towards feature subsets that achieve high classification performance while using less number of features.

c) Equilibrium Pool Formation

The equilibrium-pool is made up of the four best particles based on fitness and the average of those four particles (five reference vectors). These vectors function as guides and maintain the position updates for the remaining population.

d) Standard EO Update Mechanism

In this stage, the algorithm updates all its parameters, as well as the position of each particle. The modification occurs during the position update, which is performed using Equation 2.2. Each particle C_i^{t+1} is then converted into a binary vector using the transfer function defined in Equation 2.16.

e) Termination

The process repeats for a predefined number of iterations until the algorithm then returns the best-performing feature subset as the optimal solution.

Algorithm 3 Pseudo code of the EO for feature selection algorithm.

Population initialization using (2.1), $i = 1, 2, 3, ...N$
Set a large fitness values to the equilibrium candidates
Set the parameters of EO $a_1 = 2$; $a_2 = 1$; $GP = 0.5$
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of particles (N)) **do**
        Calculate fitness of i^{th} particle using 2.14
        if $fit(\vec{C}_i) < fit(\vec{C}_{eq1})$ **then**
            Replace $\vec{C}_{eq1}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq1})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2})$ **then**
            Replace $\vec{C}_{eq2}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq2})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq3})$ **then**
            Replace $\vec{C}_{eq3}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq3})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq3}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq4})$ **then**
            Replace $\vec{C}_{eq4}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq4})$ with $fit(\vec{C}_i)$
        end if
    end for
    $\vec{C}_{evg} = (\vec{C}_{eq1} + \vec{C}_{eq2} + \vec{C}_{eq3} + \vec{C}_{eq4})/4$
    Construct the equilibrium pool $\vec{C}_{eq1}, \vec{C}_{eq2}, \vec{C}_{eq3}, \vec{C}_{eq4}, \vec{C}_{evg}$
    Accomplish memory saving ($if Iter > 1$).
    Assign $t = (1 - \frac{Iter}{Max\_Iter})^{(a_2 \frac{Iter}{Max\_Iter})}$.
    for ($i = 1$ to number of particles (N)) **do**
        Randomly choose one candidate from the equilibrium pool.
        Generate random vectors of $\vec{\lambda}, \vec{r}$
        Construct $\vec{F}, G\vec{C}P, \vec{G}_0$, and $\vec{G}$ using the following equations respectively (2.3), (2.6), (2.5), (2.4).
        Update concentration $\vec{C}$ using (2.2).
        Convert concentration $\vec{C}$ to binary vector using (2.16).
    end for
    Iter = Iter + 1
end while
return

3.2.2 Henry Gas Solubility Optimization for Feature Selection

As in the case of the Equilibrium Optimizer, the Henry Gas Solubility Optimization (HGSO) algorithm underwent several modifications to be transformed for the feature selection problem. Its original continuous representation was changed to binary encoding, with each candidate solution representing selected features by binary values. An appropriate transfer

function was incorporated to transform HGSO's continuous updates into binary decisions so that the algorithm can work within the discrete search space of feature subsets. The fitness function was also modified to simultaneously measure classification accuracy and subset size so that the algorithm can keep a balance between performance and dimensionality. All these transformations made HGSO suitable for binary feature selection in classification problems.

a) Initialization and Clustering

The algorithm starts off by generating a random population of particles to represent a potential feature subset, in binary. The initial position of each particle and parameters (Henry's constant H_j, partial pressure $P_{i,j}$, and constant value C_j) are initialized using 2.7 and 2.8.

$$X_{i,j} = \begin{cases} 1, & \text{if } S(X_{i,j}) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

The population is then divided into equal-sized clusters.

b) Fitness Evaluation

The assessment of every cluster is done independently by identifying the best particle in the cluster and identifying the best particle globally. The fitness function used is mentioned in Chapter.3 (Eq.2.14).

c) Standard HGSO Update Mechanism

HGSO updates Henry's constant H_j, solubility $S_{i,j}$, and the position of each particle based on the update function 2.11. The updated position of each particle X_i^{t+1} is then converted into a binary vector using the transfer function defined in Equation 2.16.

d) Local Optima Escape

To prevent early convergence, part of the worst-performing particles are reinitialized using 2.13.

e) Termination

The algorithm keeps iterating until the maximum number of iterations is reached. As a result, the best solution found will be returned as the final selected feature subset.

Algorithm 4 Pseudo code of the HGSO for feature selection algorithm.

population initialization using (2.7), $i = 1, 2, 3, ...N$.
set the parameters of HGSO l_1,l_2,l_3 and initialize the properties H_j,$P_{i,j}$,C_j using (2.8).
Divide the population of gas particles into a number of gas types (cluster) with the same Henry's constant value H_j.
Evaluate each cluster j using 2.15.
Get the best gas particle $X_{i,best}$ in each cluster, and the best particle in population X_{best} .
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of particles (N)) **do**
        Update the positions of each gas particle using (2.11).
        Convert the positions of each gas particle to binary using (2.16).
    end for
    Update Henry's coefficient of each gas type using (2.9).
    Update solubility of each gas particle using (2.10).
    Rank and select the number of worst particles using (2.12).
    Update the position of the worst agents using (2.13).
    Update the best gas particle $X_{i,best}$, and the best search agent gas particle in the population X_{best}.
    Iter = Iter + 1
end while
return X_{best}

3.3 Hybrid Henry Gas Solubility Optimization and the Equilibrium Optimizer for Feature Selection: Real Cases with Twitter Spam Detection

3.3.1 Algorithm process

HGSOEO is a hybrid feature selection method that integrates Henry's Gas Solubility Optimization (HGSO) and the Equilibrium Optimizer (EO). The hybridization is motivated by

the strengths and promising results demonstrated by HGSO and EO in comparison to other well-known algorithms in the literature. The primary goal of HGSOEO is to identify the optimal subset of features, minimizing the number of selected features while ensuring high performance, all within a short computational time. This makes it a highly efficient and effective wrapper-based feature selection approach.

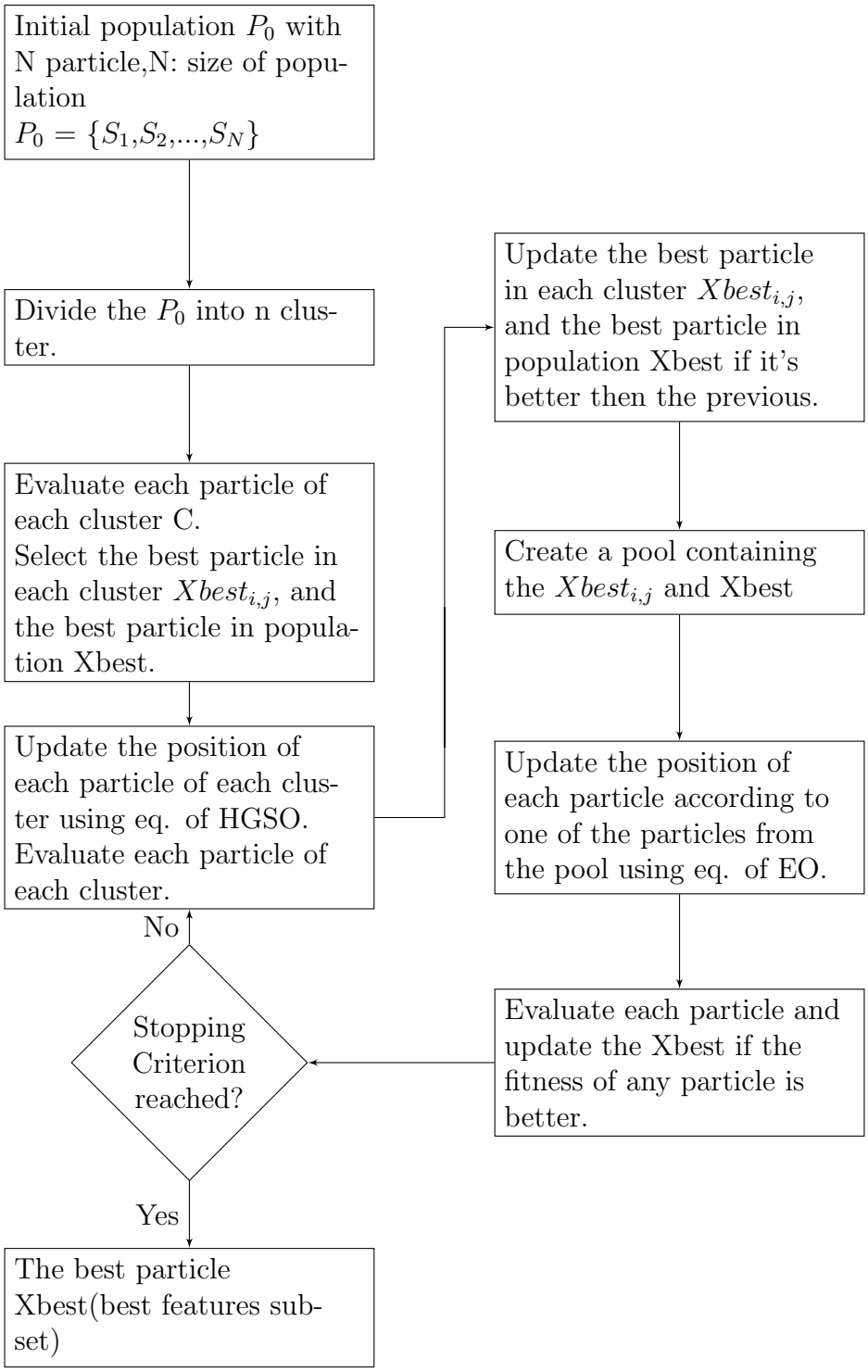

Figure 3.1: The proposed method architect [17].

Algorithm 5 Pseudo code of the HGSOEO algorithm.

population initialization, $i = 1, 2, 3, ...N$
set the parameters of HGSO l_1,l_2,l_3 and initialize the properties H_j,$P_{i,j}$,C_j using (2.8).
Divide the population into clusters with the same Henry's constant value H_j.
Evaluate each cluster j.
Get the best particle $X_{i,best}$ in each cluster, and the best particle in population X_{best} .
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of particles (N)) **do**
        Update the positions of each particle using (2.11).
    end for
    Update the best particle $X_{i,best}$ in each cluster, and the best particle in population X_{best}.
    Construct the equilibrium pool $X_{i,best}, X_{best}$
    for ($i = 1$ to number of particles (N)) **do**
        Randomly choose one candidate from the equilibrium pool.
        Generate random vectors of $\vec{\lambda}, \vec{r}$
        Construct $\vec{F}, G\vec{C}P, \vec{G_0}$, and $\vec{G}$ using the following equations respectively (2.3), (2.6), (2.5), (2.4).
        Update the positions of each particle using (2.2).
    end for
    Update the best particle $X_{i,best}$, and the best particle in the population X_{best}.
    Iter = Iter + 1
end while
return X_{best}

The process of the HGSOEO begins with the initialization of the first population (P), composed of N particles where each particle represents a potential candidate solution, corresponding to a subset of features. This population is further split into clusters with the same number of particles. Afterwards, the process selects based on the low fitness value the best particle from each cluster, along with the best global particle from the entire population. The HGSOEO then performed an iterative process that ends when it reaches the maximum number of iterations. During each iteration, the particles update their positions using the HGSO updating mechanism. The updated population is evaluated, and the best particles from both the groups and the overall population are replaced if the new particles have better fitness values. These updated best particles, along with the global best particle, are added to a pool, where each particle has an equal probability of being selected. This pool is used

in the EO updating phase, balancing exploration and exploitation by refining the positions of particles.

After the EO update, the population is evaluated again to identify the best particle in each group and the overall best particle in the population. If any new particles outperform the existing best particles, replacements are made. This process ensures continuous improvement in the solution quality. Once the stopping condition is satisfied, the best global particle, representing the optimal subset of features, is selected as the final output. By combining HGSO and EO, this hybrid algorithm efficiently identifies a minimal set of features while maintaining high performance. The steps are detailed below:

Step 1: Population Initialization

This step concerns the creation of an initial population with N particles. Each particle is a subset of features. At this stage, the search space for the feature selection process is established, where each particle constitutes a candidate solution to the problem.

Step 2: Evaluate the Initial Population

Once the population is created, it is divided into equal clusters. Each cluster is evaluated to identify its best particle, while the best global particle is determined from the entire population. The fitness of each particle is assessed based on the error rate and the feature subset reduction rate, using a random forest classifier.

Step 3: HGSO-Based Update

In this step, the positions of all particles are updated based on the best global particle. The position updating metric of HGSO guides each particle to move towards better solutions by taking into account the influence of the global best particle, ensuring the exploration in the search space.

Step 4: Evaluate the New Population

After the new population is formed, the fitness for each particle in each cluster is recomputed, and it checks whether the newly evaluated particles are better than the

previous ones so as to update the best particle in each cluster and the best global particle. This step ensures that the population continues to improve iteratively.

Step 5: The Equilibrium Pool Formation

An equilibrium pool made of the best particle from each cluster, plus the best global particle, is built. Its size is set to the number of clusters increased by one. It constitutes the memory of high-quality solutions used during the HGSO phase for further updating positions of particles.

Step 6: EO-Based Update

The clusters are merged together into one population, and then the position of all particles is updated using the EO's update metric. In this step, each particle is updated by randomly selecting one particle from the equilibrium pool, with each having an equal probability of being chosen. It exploits the good trade-off between exploration and exploitation since the particles adjust the positions based on the promising solutions.

Step 7: Evaluate the New Population

The population is redivided into clusters in which each particle within the clusters is reassessed. The best particle in each cluster as well as the best global particle are updated if better solutions are identified. If the stopping criterion is not met, the process loops back to Step 3. If the stopping condition is satisfied, the algorithm will terminate and return the best global particle as the optimal subset of features.

3.3.2 Integration Strategy

As previously discussed, the proposed hybrid method HGSOEO combines the strengths of both the Henry Gas Solubility Optimization (HGSO) and the Equilibrium Optimizer (EO) algorithms. The theoretical framework of this fusion is illustrated in Figure 3.2, highlighting the specific stage at which the EO component is integrated within the HGSO framework.

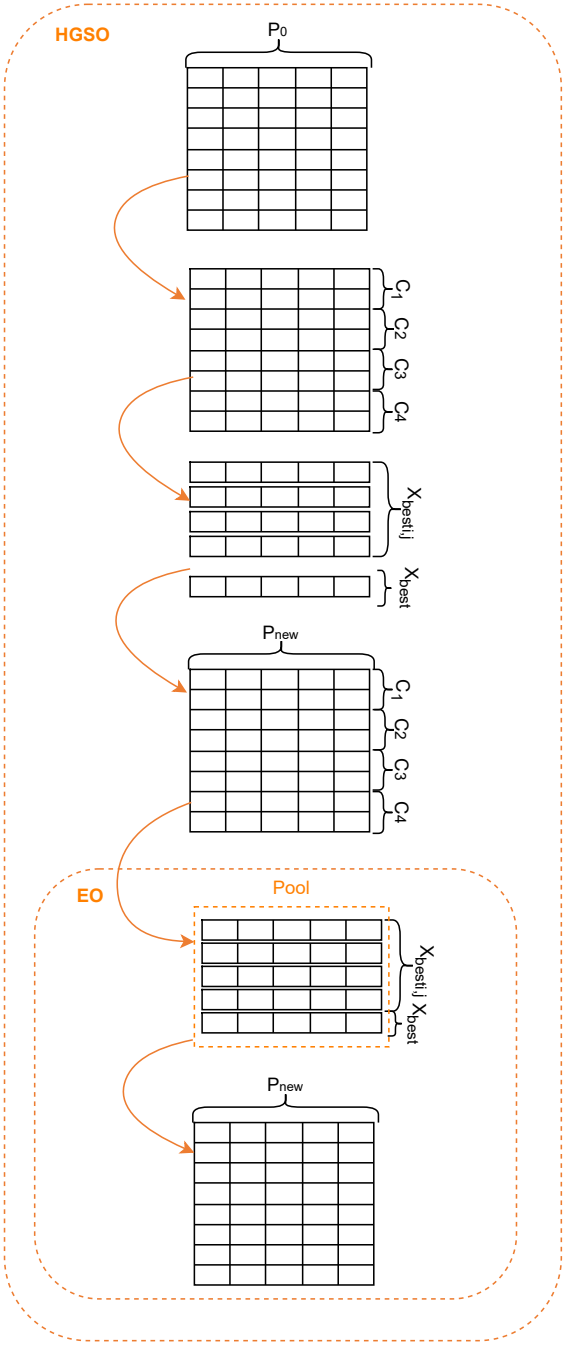

Figure 3.2: Fusion Schema[17].

The hybridization works by executing the normal HGSO algorithm to create a new population. A collection of good solutions is then chosen from this population and employed to form an equilibrium pool. Rather than using EO separately, its fundamental updating mechanism is selectively incorporated into the work flow of HGSO. More specifically, EO

is utilized to improve HGSO's worst individuals. These poor agents are re-evaluated and updated by EO's position update equations, led by one of the equilibrium candidates. As a result, the combined method benefits from improved balance between exploration and exploitation, leading to better search efficiency and more accurate feature subset identification.

3.4 Equilibrium Optimizer for Feature Selection in Clustering

As previously described, adapting the Equilibrium Optimizer (EO) to the feature selection problem required several changes to transform it into a binary variant. In this case, however, the focus is on applying the algorithm to a clustering task. The adaptation steps and mechanisms remain the same; the only modification lies in the fitness function.

a) Initialization

Initialize a population of particles, where each particle is a binary vector representing a candidate feature subset.

b) Fitness Evaluation

Unlike the classification version, the clustering variant evaluates each particle using the Equation. 2.15.

Each particle (feature subset) is evaluated using the Gaussian Mixture Model (GMM) for clustering. The clustering results (labels) are then compared with the true data structure using the Adjusted Rand Index (ARI) to calculate the fitness of each particle.

c) Equilibrium Pool Formation

Select the top four particles with the best fitness values and compute their average to form the equilibrium pool.

d) Standard EO Update Mechanism

Update each particle's position based on the EO update equations. Then apply a transfer function to convert the continuous values into binary format.

e) Termination

Repeat the evaluation and update steps until the maximum number of iterations is reached. Return the best particle as the optimal feature subset.

Algorithm 6 Pseudo code of the EO for feature selection algorithm in Clustering.

Population initialization using (2.1), $i = 1, 2, 3, ...N$
Set a large fitness values to the equilibrium candidates
Set the parameters of EO $a_1 = 2$; $a_2 = 1$; $GP = 0.5$
while $Iter < Max\_Iter$ **do**
    for ($i = 1$ to number of particles (N)) **do**
        Calculate fitness of i^{th} particle using 2.15
        if $fit(\vec{C}_i) < fit(\vec{C}_{eq1})$ **then**
            Replace $\vec{C}_{eq1}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq1})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2})$ **then**
            Replace $\vec{C}_{eq2}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq2})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq3})$ **then**
            Replace $\vec{C}_{eq3}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq3})$ with $fit(\vec{C}_i)$
        else if $fit(\vec{C}_i) > fit(\vec{C}_{eq1}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq2}) \& fit(\vec{C}_i) > fit(\vec{C}_{eq3}) \& fit(\vec{C}_i) < fit(\vec{C}_{eq4})$ **then**
            Replace $\vec{C}_{eq4}$ with $\vec{C}_i$ and $fit(\vec{C}_{eq4})$ with $fit(\vec{C}_i)$
        end if
    end for
    $\vec{C}_{evg} = (\vec{C}_{eq1} + \vec{C}_{eq2} + \vec{C}_{eq3} + \vec{C}_{eq4})/4$
    Construct the equilibrium pool $\vec{C}_{eq1}, \vec{C}_{eq2}, \vec{C}_{eq3}, \vec{C}_{eq4}, \vec{C}_{evg}$
    Accomplish memory saving ($if Iter > 1$).
    Assign $t = (1 - \frac{Iter}{Max\_Iter})^{(a_2 \frac{Iter}{Max\_Iter})}$.
    for ($i = 1$ to number of particles (N)) **do**
        Randomly choose one candidate from the equilibrium pool.
        Generate random vectors of $\vec{\lambda}, \vec{r}$
        Construct $\vec{F}, G\vec{C}P, \vec{G}_0$, and $\vec{G}$ using the following equations respectively (2.3), (2.6), (2.5), (2.4).
        Update concentration $\vec{C}$ using (2.2).
        Convert concentration $\vec{C}$ to binary vector using (2.16).
    end for
    Iter = Iter + 1
end while
return

The architecture shown in Figure 3.3 illustrates the workflow of applying the Equilibrium Optimizer (EO) to the feature selection problem in clustering context.

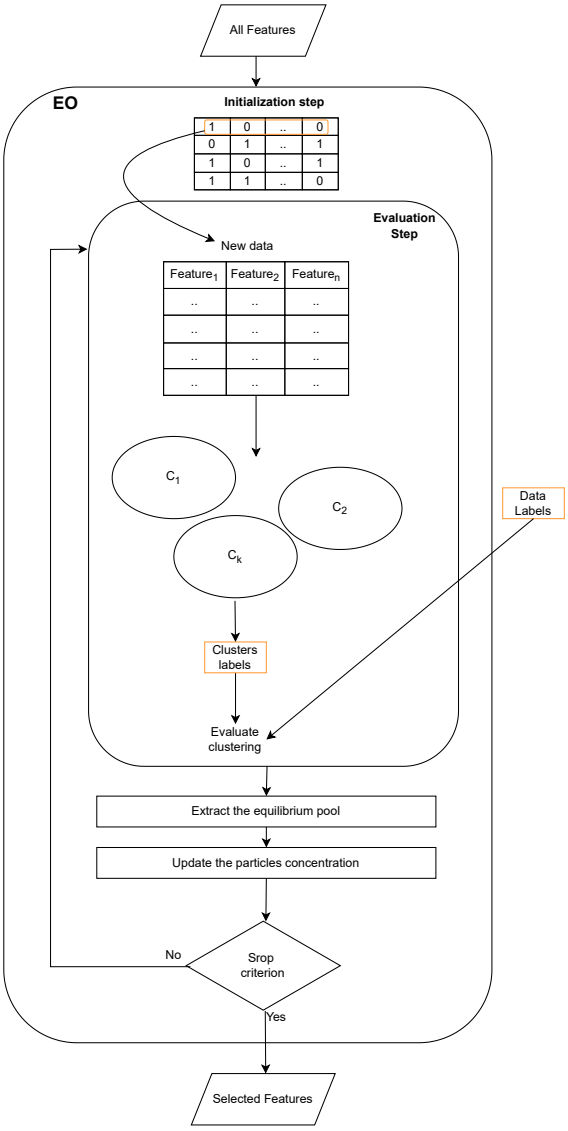

Figure 3.3: Proposed Method Model[18].

3.5 Conclusion

This chapter detailed the core contributions of the thesis. EO and HGSO were analyzed and compared in supervised feature selection tasks, leading to the development of a hybrid approach (HGSOEO) that demonstrated improved performance. Additionally, EO was successfully adapted for unsupervised feature selection in clustering. These contributions form the basis for the experimental analysis in the next chapter.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter presents and analyzes the experimental results of the proposed feature selection approaches. It includes a comparison between EO and HGSO, the evaluation of the hybrid HGSOEO method, and the use of EO for unsupervised feature selection. Results are reported on benchmark datasets and a real Twitter Spam dataset using various evaluation metrics.

4.2 Datasets

To evaluate the effectiveness of the proposed metaheuristic-based feature selection algorithms, several benchmark and real-world datasets were employed. These datasets represent different domains and levels of complexity, enabling a robust assessment across both supervised and unsupervised learning tasks. This section provides an overview of the datasets used.

4.2.1 Benchmark datasets

To evaluate the effectiveness and robustness of the proposed feature selection approaches—namely, the Equilibrium Optimizer for Feature Selection (EOFS) and Henry Gas Solubility Optimiza-

tion for Feature Selection (HGSOFS)—a diverse set of benchmark datasets was employed. These datasets were selected to cover both supervised and unsupervised learning scenarios, particularly classification and clustering tasks. The datasets vary in terms of dimensionality, number of instances, and number of classes, thereby providing a comprehensive evaluation environment for feature selection performance. The details of each are mentioned in Tab.4.1.

Table 4.1: Summary of Benchmark Datasets

No.	Dataset	Features	Instances	Classes	Used In
1	Iris	4	150	3	Classification / Clustering
2	Banknote Authentication	5	1372	2	Classification
3	Glass	10	214	7	Classification
4	Wine	12	178	3	Classification / Clustering
5	Segment	19	2310	7	Classification / Clustering
6	Breast Cancer	30	569	2	Clustering
7	Ionosphere	34	351	2	Clustering
8	Lung Cancer	56	32	3	Classification
9	Spambase	57	4601	2	Classification / Clustering
10	Libras Movement	91	360	15	Classification
11	Hill-Valley	101	606	2	Classification
12	Musk	168	476	2	Classification

4.2.2 Twitter Spam Detection Dataset

In order to assess the performance of the proposed HGSOEO feature selection approach in detecting spam on Twitter, a real-world dataset was obtained from the Social Honeypot dataset[40]. This dataset was compiled between December 30, 2009 and August 2, 2010, which contains a wealth of information pertaining to Twitter users and the associated tweets. It was originally designed to capture the behavior of legitimate users and spam accounts using honeypot strategies.

The original dataset comprises 22,223 spammer profiles (content polluters) associated with 2,353,473 tweets, and 19,276 legitimate profiles (non-spammers) with 3,259,693 tweets. However, before applying the feature selection algorithm, a significant preprocessing step was

performed to enhance the dataset's reliability and relevance:

- **Non-spam profiles** were verified for current availability. Profiles still active at the time of inspection were retained, resulting in 14,821 legitimate profiles.
- **Spam profiles** Spam profiles were filtered to retain only accounts that were either banned or no longer available—an indicator of spam activity. This yielded 5,366 spam profiles.

The final cleaned and balanced dataset thus consisted of 11,398 profiles, out of which 5,366 were spam and 6,050 were non-spam accounts—with a balanced distribution of 47% spam and 53% non-spam. The balanced makeup gives a fair evaluation during the classification and feature selection process.

Features:

In order to detect spam profiles on Twitter effectively, the dataset was expanded on the basis of feature extraction. This was necessary to present important inputs to the feature selection algorithm and classification models. Twitter accounts and their tweets provide a wide array of information, all of which can categorically be grouped into three types of features: user-based, content-based, and behavior-based features.

In all, 40 features were extracted. These features comprised both raw attributes in the original Social Honeypot dataset, and new features constructed from analytical transformation of the original features. Feature extraction was able to capture a range of differences in user behavior and tweet content, and will allow for better differentiation between spam and legitimate accounts.

4.3 Experimental Setup

This section describes the experimental setup and environment to evaluate the performance of the proposed metaheuristic algorithms for feature selection problems. The experiments

Table 4.2: Extracted features[17]

Feature No.	Feature	Description	Category
1	Number of followings	Number of accounts followed by the user	User-based
2	Number of followers	Number of followers of the user	User-based
3	Number of tweets	Number of statuses posted by the user	User-based
4	Length of screen name	Length of the user's screen name	User-based
5	Length of profile description	Length of user's bio or profile description	User-based
6	Account age (days)	Account creation date minus collection date	User-based
7	Fifo score	Followings / Followers	User-based
8	Reputation score	Followers / (Followers + Followings)	User-based
9	Age by followings	Account age / Followings	User-based
10	Age by followers	Account age / Followers	User-based
11	Tweet frequency	Tweets / Account age in days	User-based
12	Max tweet length	Length of the longest tweet	Content-based
13	Tweet length ratio	Total tweet length / (Tweets × 140)[1]	Content-based
14	Similar tweets count	Similarity between tweets via Levenshtein distance	Content-based
15	Similar tweets ratio	Similar tweets / Total tweets	Content-based
16	Total repeated words	Number of repeated words across tweets	Content-based
17	Repeated word ratio	Repeated words / Total words	Content-based
18	Total spam words	Number of spam trigger words[2]	Content-based
19	Spam word ratio	Spam words / Total words	Content-based
20	Total uppercase words	Number of uppercase words used	Content-based
21	Uppercase word ratio	Uppercase words / Total words	Content-based
22	Total stop words	Stop words used in tweets	Content-based
23	Stop word ratio	Stop words / Total words	Content-based
24	Total exclamation marks	Number of exclamation marks used	Content-based
25	Total question marks	Number of question marks used	Content-based
26	Total digit count	Digits used in tweets	Content-based
27	Total emoji count	Emojis used in tweets	Content-based
28	Total mentions count	Mentions used in tweets	Content-based
29	Total URLs count	URLs used in tweets	Content-based
30	Total hashtags count	Hashtags used in tweets	Content-based
31	Mentions ratio	Mentions / Total tweets	Content-based
32	URLs ratio	URLs / Total tweets	Content-based
33	Hashtags ratio	Hashtags / Total tweets	Content-based
34	Unique mentions count	Number of unique mentions	Content-based
35	Unique URLs count	Number of unique URLs	Content-based
36	Unique mention ratio	Unique mentions / Total mentions	Content-based
37	Unique URL ratio	Unique URLs / Total URLs	Content-based
38	Time between tweets (min)	Time between consecutive tweets in minutes	Behavior-based
39	Avg. time between tweets	Total time between tweets / Tweet count	Behavior-based
40	Max tweets per day	Maximum tweets posted in a day	Behavior-based

[a] Before 2017, tweets were limited to 140 characters.

[b] `https://snov.io/blog/550-spam-trigger-words-to-avoid/`

were performed in both supervised and unsupervised environments on benchmark and real-world datasets, as discussed in Section 4.2.

The Experimental setup includes hardware and software specifications, as well as parameter initialization details.

4.3.1 Hardware and Software Environment

All experiments were executed on a standard computing platform with the following specifications:

- **Processor:** AMD Ryzen 7 3700U running at 2.30 GHz,
- **RAM:** 8 GB,
- **Operating System:** Windows 11,
- **Programming Language:** Python, using the Jupyter Notebook environment,
- **Libraries:** NumPy, Pandas, and Scikit-learn for data handling and machine learning; Matplotlib and Seaborn for visualization.

4.3.2 Algorithm Parameters

Each algorithm was configured using parameters to ensure fair comparison and reliable convergence. Table 4.3 summarizes the key parameter settings used for EO, HGSO, and HGSOEO, including population size, maximum iterations, and other specific settings.

Table 4.3: Algorithm parameter settings.

Algorithm	**Population Size**	**Max Iterations**	**Other Parameters**
EO	10	100	$a_1 = 2$, $a_2 = 1$, Generation probability (GP) = 0.5
HGSO	10	100	Number of clusters = 4, $C_1 = 0.1$, $C_2 = 0.2$, $L_1 = 5 \times 10^{-3}$, $L_2 = 100$, $L_3 = 1 \times 10^{-2}$
HGSOEO	20	100	Same parameters as EO and HGSO

4.4 Equilibrium Optimizer and Henry Gas Solubility Optimization Algorithms for Feature Selection: Comparison Study

This section presents the experimental results and comparative study of Equilibrium Optimizer Feature Selection (EOFS) and Henry Gas Solubility Optimization Feature Selection (HGSOFS) algorithms. Performance is computed on ten traditional benchmark datasets and three classifiers, namely, Naive Bayes (NB), k-Nearest Neighbors (KNN), and Random Forest (RF). All experiments are executed with the same parameter settings reported earlier in Subsection 4.3.2. Performance is compared based on classification accuracy and number of selected features. Moreover, HGSOFS is also compared with two recent metaheuristics: Particle Swarm Optimization Feature Selection (PSO-FS) and Firefly Algorithm Feature Selection (FAFS).

4.4.1 Classification Results with EOFS and HGSOFS

The performance of EOFS and HGSOFS is evaluated using NB, KNN, and RF classifiers. Tables 4.4, 4.5 and 4.6 illustrate the results.

Table 4.4: Comparison of EOFS and HGSOFS using the NB classifier.

No.	Dataset	EOFS with NB		HGSOFS with NB	
		No. of selected features	Accuracy	No. of selected features	Accuracy
1	Iris	1	0.96	1	0.96
2	Banknote Authentication	2	0.87	**1**	**0.87**
3	Glass	2	0.65	**1**	**0.65**
4	Wine	2	1.0	**1**	**1.0**
5	Segment	3	0.84	**1**	**0.90**
6	Lung Cancer	2	1.0	**1**	**1.0**
7	Spambase	6	0.89	**2**	**0.88**
8	Libras Movement	10	0.70	**5**	**0.68**
9	Hill-Valley	1	0.52	1	0.52
10	Musk	7	0.90	**1**	**0.87**

Table 4.5: Comparison of EOFS and HGSOFS using the KNN classifier.

No.	Dataset	EOFS with Knn		HGSOFS with Knn	
		No. of selected features	Accuracy	No. of selected features	Accuracy
1	Iris	1	0.96	**1**	**1.0**
2	Banknote Authentication	4	1.0	**1**	**1.0**
3	Glass	3	0.81	**1**	**0.81**
4	Wine	2	0.97	**1**	**1.0**
5	Segment	5	0.96	**1**	**0.97**
6	Lung Cancer	3	1.0	**1**	**1.0**
7	Spambase	13	0.92	**10**	**0.91**
8	Libras Movement	8	0.75	**2**	**0.75**
9	Hill-Valley	8	0.64	**2**	**0.65**
10	Musk	19	0.93	**5**	**0.91**

Table 4.6: Comparison of EOFS and HGSOFS using the RF classifier.

No.	Dataset	EOFS with RF		HGSOFS with RF	
		No. of selected features	Accuracy	No. of selected features	Accuracy
1	Iris	1	0.96	**1**	**1.0**
2	Banknote Authentication	3	1.0	**1**	**1.0**
3	Glass	3	0.88	**1**	**0.86**
4	Wine	2	1.0	**1**	**1.0**
5	Segment	4	0.98	**1**	**0.98**
6	Lung Cancer	3	1.0	**1**	**1.0**
7	Spambase	13	0.92	**6**	**0.95**
8	Libras Movement	8	0.81	**2**	**0.83**
9	Hill-Valley	6	0.65	**1**	**0.67**
10	Musk	17	0.96	**4**	**0.96**

In nearly all the datasets and classifiers tested, HGSOFS significantly outperforms. For example, on the Wine dataset, HGSOFS achieves a perfect accuracy of 100% with a single feature when combined with all three classifiers—NB, KNN, and RF—thus surpassing EOFS, which uses two features to achieve similar accuracy. This trend is reproducibly observed in a number of other datasets, including Segment, Lung Cancer, and Musk, where HGSOFS decreases the number of selected features while either preserving or slightly enhancing the classification accuracy.

An interesting outcome is observed on the Spambase dataset. Both methods are effective, yet HGSOFS provides 95.1% accuracy with RF using just six features, outperforming EOFS, which requires thirteen features to produce 92% accuracy. This indicates HGSOFS's ability to capture more informative details using fewer features, significantly reducing computational complexity without sacrificing performance.

Another highlight is the performance on high-dimensional datasets such as Hill-Valley and Libras Movement. In spite of the complexity and high feature space, HGSOFS continues to work well and consistently chooses fewer features and gives competitive accuracy compared to EOFS.

4.4.2 Comparison with PSO-FS and FAFS

To further evaluate the effectiveness of HGSOFS, the algorithm is also benchmarked against two popular metaheuristic-based feature selection algorithms—Particle Swarm Optimization for Feature Selection (PSO-FS) and Firefly Algorithm for Feature Selection (FAFS), as shown in Tables 4.7 and 4.8.

Table 4.7: Comparison between HGSOFS and PSO-FS

No.	Dataset	PSO-FS		HGSOFS with RF	
		No. of selected features	Accuracy	No. of selected features	Accuracy
1	Iris	1	96	**1**	**100**
2	Banknote Authentication	3	99.85	**1**	**100**
3	Glass	**1**	**98.13**	1	86.04
4	Wine	4	95.55	**1**	**100**
5	Segment	4	94.97	**1**	**98.48**
6	Lung Cancer	10	87.5	**1**	**100**
7	Spambase	14	91.26	**6**	**95.11**
8	Libras Movement	29	88.88	**2**	**83.88**
9	Hill-Valley	35	61.38	1	67.21
10	Musk	76	94.11	**4**	**96.87**

Table 4.8: Comparison between HGSOFS and FAFS

No.	Dataset	FAFS		HGSOFS with RF	
		No. of selected features	Accuracy	No. of selected features	Accuracy
1	Iris	1	75	**1**	**100**
2	Banknote Authentication	1	75	**1**	**100**
3	Glass	**1**	**89.88**	1	86.04
4	Wine	1	92.31	**1**	**100**
5	Segment	2	91.2	**1**	**98.48**
6	Lung Cancer	3	94.64	**1**	**100**
7	Spambase	**1**	**98.25**	6	95.11
8	Libras Movement	**2**	**93.26**	2	83.88
9	Hill-Valley	**2**	**98**	1	67.21
10	Musk	6	96.41	**4**	**96.87**

In all the datasets, HGSOFS outperforms PSO-FS in terms of both selected features and classification accuracy. For instance, on the Musk dataset, PSO-FS uses 76 features to attain 94.11% accuracy, whereas HGSOFS uses only 4 features and reaches 96.87%. The large feature difference on all datasets implies that HGSOFS possesses better search control, adaptive balancing, and local optima escaping ability.

Although FAFS performs well, HGSOFS performs better in 6 out of 10 datasets, particularly where a delicate trade-off between exploration and exploitation of resources is required. FAFS seems to perform too well on certain datasets (such as Hill-Valley and Spambase), whereas HGSOFS performs more consistently.

Overall, the results validate that HGSOFS achieves a superior balance between exploration and exploitation throughout the search process, thereby finding feature subsets of better quality. Its persistent superiority across a variety of datasets and classifiers confirms its efficacy and generality, making it a reliable feature selection method for classification tasks.

4.5 Hybrid Henry Gas Solubility Optimization and the Equilibrium Optimizer for Feature Selection: Real Cases with Twitter Spam Detection

This section outlines and investigates the results obtained from the application of the suggested HGSOEO hybrid algorithm for feature selection within Twitter spam detection. The evaluation is structured to highlight the performance of the hybrid model in terms of both dimensionality reduction and classification accuracy, followed by a comprehensive comparison with baseline techniques and state-of-the-art methods. All experiments uses the parameter settings reported in Table 4.3, Subsection 4.3.2.

4.5.1 Stability and Convergence Analysis

In order to evaluate the stability of the proposed approach HGSOEO, 20 independent runs are performed, where each run is executed over all instances of the dataset. The classification accuracy obtained in each run is illustrated in Fig.4.1

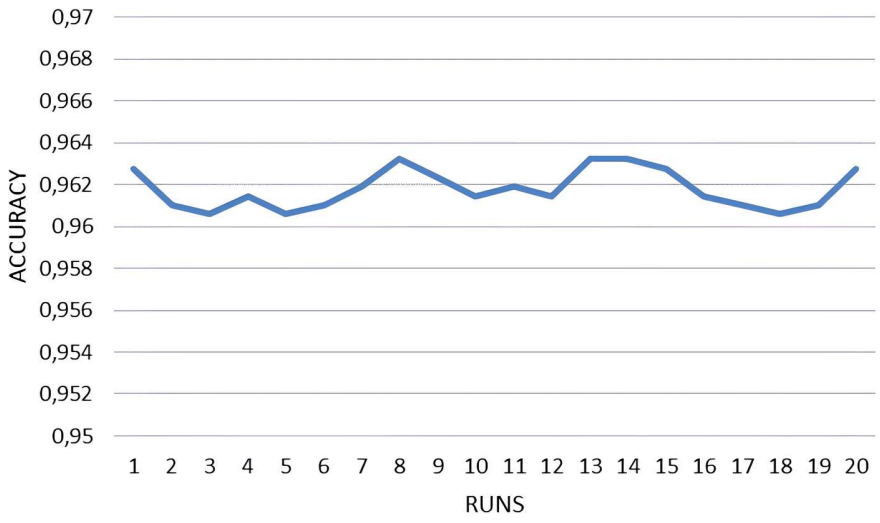

Figure 4.1: HGSOEO Accuracy during runs

The outcomes show a great level of performance consistency, with every run achieving accuracies within a tight margin around 96%. The stability indicates the robustness of the proposed approach. To proceed, the best-performing run was determined by both classification accuracy and the number of selected features.

Table 4.9 shows a comparison of the classification performance when using all features and the reduced set found by HGSOEO.

Table 4.9: Comparison of classification performance using all features vs. HGSOEO-selected subset.

Features	#features	Accuracy	Precision	Recall	Specificity	F-measure	Error	Fitness Score
All Features	40	95.97%	0,960	0,960	0.952	0.960	0.0428	0.0523
Subset 1,9,11,17,18,20, 28,32,37,40	10	96.32%	0.963	0.963	0.954	0.963	0.0367	0.0389

As evident from Table 4.9, the proposed method's selected feature subset not only achieved 75% dimensionality reduction, but also performed better on all the evaluation measures. In particular, the model's accuracy, precision, recall, specificity, and F-measure were higher, while the error rate and fitness score were lower when using the reduced feature set as compared to the entire feature set.

Fitness Convergence

The convergence behavior of the HGSOEO algorithm is illustrated in Fig. 4.2, which shows the evolution of the fitness value across iterations.

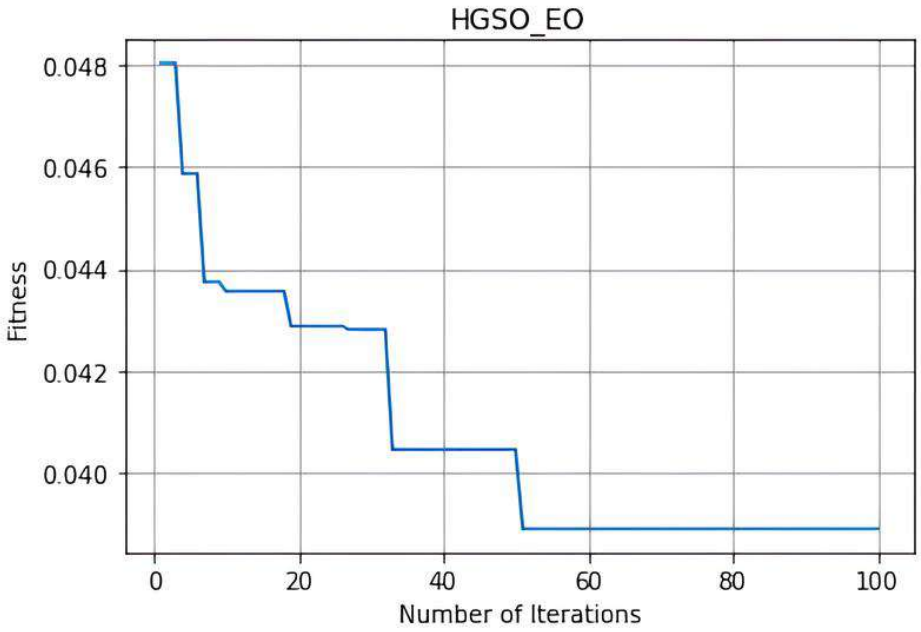

Figure 4.2: HGSOEO Fitness Curve.

Initially, the fitness value decreases consistently, revealing good exploration of the search space. The curve plateaus at around the 50th iteration, which is a sign of convergence. The lack of high oscillations and the stabilization early on are signs that the algorithm effectively evades local optima and converges to a near-optimal solution.

Accuracy Convergence

In parallel, the accuracy curve shown in Fig. 4.3 illustrates the model's performance improvement over iterations.

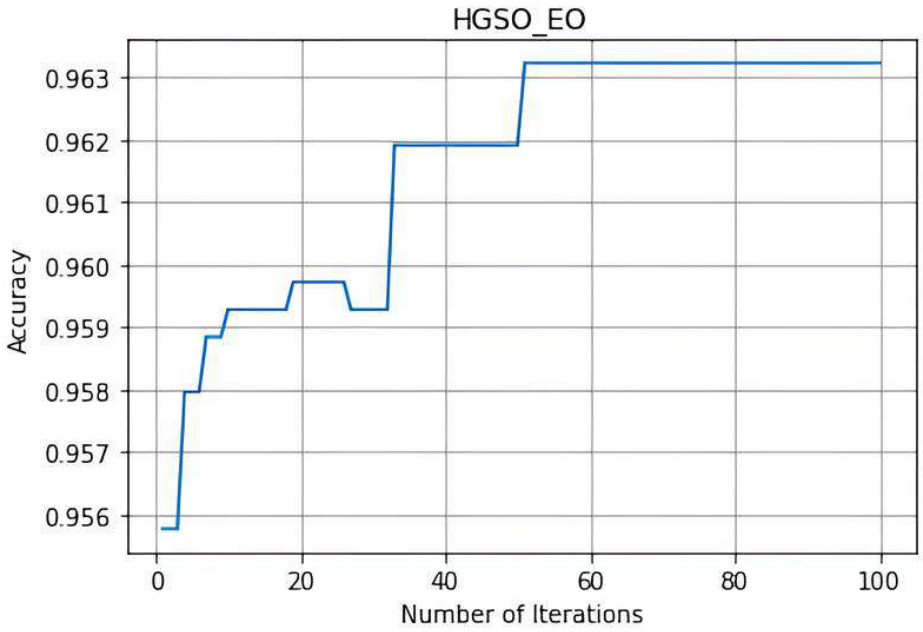

Figure 4.3: HGSOEO Accuracy Curve.

The model begins with a high baseline accuracy of around 95%, which continues to increase and stabilize around 96%. The increasing trend validates the algorithm's capacity to improve feature selection and classifier performance over time.

ROC Analysis

To assess the model's discriminative capability, the Receiver Operating Characteristic (ROC) curve is presented in Fig. 4.4.

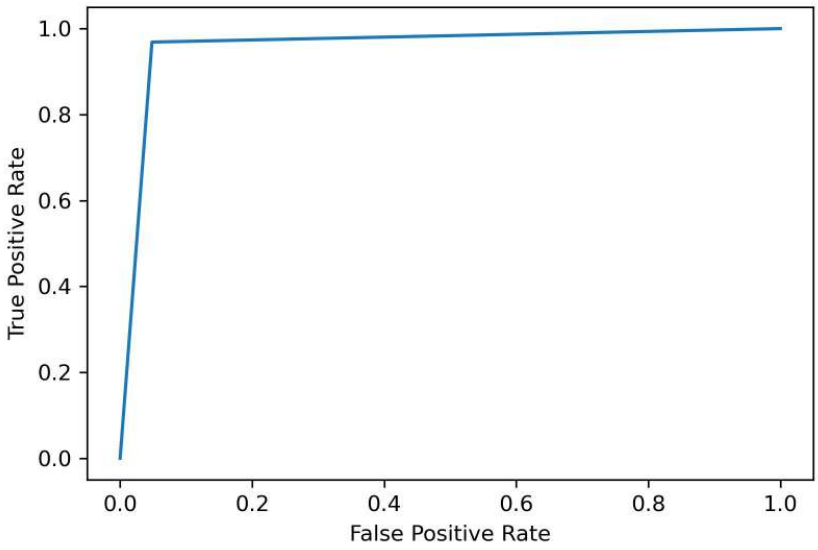

Figure 4.4: HGSOEO ROC Curve.

The ROC curve closely approximates the top-left corner of the plot, indicating good classification performance. The high value of AUC indicates the model's ability at classifying spam and non-spam profiles with minimal trade-offs.

Confusion Matrix Analysis

The final performance assessment is visualized through the confusion matrix in Fig. 4.5.

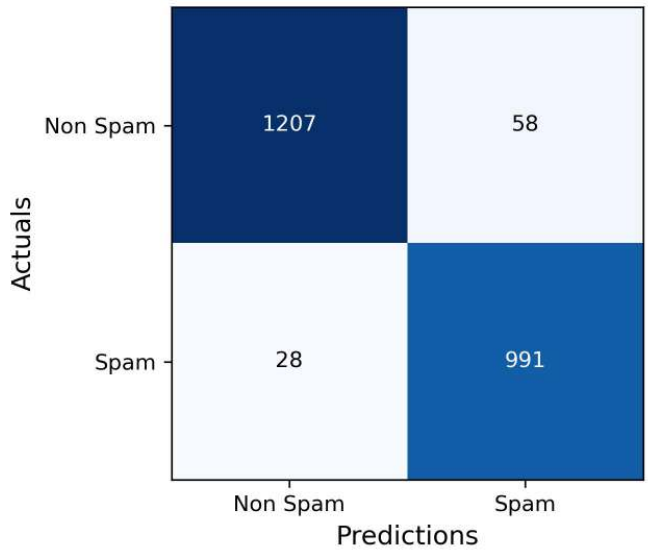

Figure 4.5: Confusion matrix showing the classification performance of HGSOEO.

The confusion matrix also confirms the classification performance of the proposed method. The high true positives and true negatives indicate the model's efficiency in classifying both spam and legitimate users, whereas the low false positive and false negative rates highlight its trustworthiness.

4.5.2 Evaluation with Classical Classifiers

In order to further confirm the generalizability of the feature subset identified by the proposed HGSOEO algorithm, a comparative analysis was performed with a suite of popular classification models. These comprise conventional classifiers namely Naive Bayes (NB), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), K-Nearest Neighbors (K-NN), and Decision Tree (J48), and deep learning models—Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). Every model was analyzed with two settings: the full feature set (40 features) and the reduced feature subset (10 features) achieved by the proposed feature selection approach.

The performance measures taken into consideration here for comparison are accuracy, pre-

cision, recall, and Area Under the ROC Curve (AUC). The performances of each of the classifiers under both settings are compared in Table 4.10.

Table 4.10: All Features versus the proposed method's Subset on different classifiers

Classifier	With All Features				With the Subset			
	Accuracy %	Precision	Recall	AUC	Accuracy %	Precision	Recall	AUC
NB	**76.620**	**0.800**	**0.766**	**0,934**	73.161	0.776	0.732	0,915
SVM	**91.812**	**0.920**	**0.918**	**0,920**	82.092	0.843	0.821	0,832
MLP	**92.994**	**0,931**	**0,930**	**0,970**	91.944	0,919	0,919	0,962
K-NN	88.485	0.885	0.885	0,884	**89.886**	**0.90**	**0.899**	**0,945**
DT(J48)	92.907	0.930	0.929	0,915	**94.483**	**0.946**	**0.945**	**0,952**
CNN	87.609	0.797	0.969	0.946	**90.06**	**0.861**	**0.927**	**0.966**
LSTM	92.425	0.887	0.951	0.978	**93.914**	**0.904**	**0.966**	**0.982**

Performance Analysis

The analysis demonstrates that the smaller feature subset tends to result in better or similar performance, depending on the classifier. In particular, the K-NN, Decision Tree (J48), CNN, and LSTM models showed significant improvements when trained on the smaller feature set:

- **K-NN**: Accuracy rose from 88.49% to 89.89%, with precision, recall, and AUC improving.
- **Decision Tree (J48)**: Accuracy increased from 92.91% to 94.48% with corresponding improvements on all other measures.
- **CNN**: About 2.5% improvement in accuracy, and AUC rose from 94.6% to 96.6%.
- **LSTM**: Attained the highest classification scores overall, with an AUC of up to 98.2% when utilizing the chosen subset.

These findings firmly indicate that the feature selection approach proposed here can not only decrease the dimensionality of the dataset but also increase the predictive ability of a number of classification models. In contrast, for the NB, SVM, and MLP classifiers, performance decreased with the reduced subset. Although the decrease was most evident in

SVM (reducing from 91.81% to 82.09% accuracy), the MLP still had a high 91.94% accuracy, and overall performance was still competitive.

Visual Comparison

To provide a visual summary of classifier performance under both settings, Fig. 4.6 presents a bar chart comparing classification accuracy across all evaluated models.

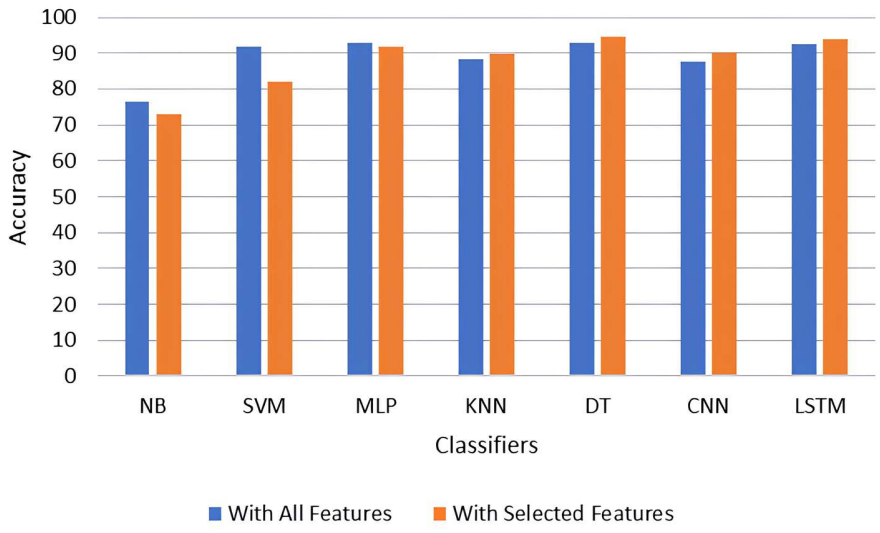

Figure 4.6: Accuracy bar chart of all features versus the subset.

The chart clearly illustrates that in several cases—particularly for decision-tree-based and deep learning classifiers—the reduced subset of features outperforms the full set. Even in cases where a slight drop is observed, the difference is marginal and may be considered acceptable given the significant reduction in feature space.

Overall, the proposed feature subset has high utility and transferability for various machine learning models. It results in competitive or better performance with less computational cost for high-dimensional input. This supports the merit of the HGSOEO feature selection method as an efficient and robust method for practical spam detection applications.

4.5.3 Comparative Analysis with Existing Methods

In order to fully evaluate the performance and effectiveness of the newly proposed HGSOEO-based feature selection method, four phases of comparison experiments were carried out. In each phase, comparing the proposed method with a certain type of feature selection methods was emphasized. All the comparisons were implemented on the dataset presented in Section 4.2 with the same experimental settings to ensure a fair and credible comparison.

Internal Comparison: HGSOEO-FS vs. HGSO-FS and EO-FS

The first comparison is intended to assess the value of combining HGSO and EO by comparing HGSOEO-FS to each of its constituent parts: HGSO-FS and EO-FS. The results are highlighted in Table 4.11.

Table 4.11: Comparison between methods

Method	# Selected Features	Accuracy %	Precision	Recall	AUC
HGSO-FS	17	95.88	0.959	0.959	0,989
EO-FS	10	95.53	0.956	0.955	0,987
Proposed system	**10**	**96.32**	**0.963**	**0.962**	**0.987**

The internal comparison against HGSO-FS and EO-FS validated the utility of hybridization. HGSOEO outperformed its underlying algorithms consistently for classification performance, even though it chose fewer features than HGSO-FS and tied with EO-FS in the number of features. This indicates that hybridization of exploration (HGSO) and exploitation (EO) strategies improves the search ability altogether, resulting in a more optimal and compact feature subset.

Comparison with Filter-Based Feature Selection Methods

In the second phase, HGSOEO-FS is compared against five traditional filter approaches in Weka: Correlation, Gain Ratio, Information Gain, Relief-F, and Symmetrical Uncertainty.

Each approach was set to choose the top 10 features to be equal to the subset size of our method. The outcomes of the comparison are shown in Table 4.12.

Table 4.12: Comparison between HGSOEO-FS and Filter-based methods

Method	Selected Features	Accuracy %	Precision	Recall	AUC
Correlation	8,28,19,6,29,4,31,24,18,25	90.98	0.911	0.910	0,966
Gain Ratio	36,9,28,31,32,34,1,10,27,8	94.48	0.946	0.945	0,985
Info Gain	9,36,28,31,1,8,32,10,7,34	94.61	0.947	0.946	0,984
Relief-F	8,23,6,36,4,22,5,27,37,28	91.02	0.911	0.910	0,967
Symmetrical Uncertainty	36,9,28,31,32,1,34,10,8,7	94.61	0.947	0.946	0,984
Proposed system	**1,9,11,17,18,20,28,32,37,40**	**96.32**	**0.963**	**0.962**	**0.987**

In comparison with the five filter approaches, HGSOEO-FS demonstrated higher classification performance across all metrics. Although there was some overlap among selected features (e.g., features such as followings, spam word counts, and URL-related features), the proposed method's retention of less noticeable but influential features (e.g., ratio of repeated words, upper-case words, tweet frequency) was likely crucial to the performance improvement. These findings highlight the limitation of univariate ranking schemes and the benefit of a population-based global search for modeling inter-feature relationships.

Comparison with Wrapper-Based Feature Selection Methods

In the third phase of evaluation, HGSOEO-FS is compared with three wrapper-based methods available in Weka: Cfs Subset Evaluation, Classifier Subset Evaluation, and Wrapper Subset Evaluation. Table 4.13 illustrates the comparison's results.

Table 4.13: Comparison between HGSOEO-FS and Wrapper-based methods

Method	# Selected Features	Selected Features	Accuracy %	Precision	Recall	AUC
Cfs Subset Evaluation	11	1,4,8,9,10,19,27,31,32,36,40	95.27	0.954	0.953	0,987
Classifier Subset Evaluation	23	2,4,5,6,7,9,11,13,14,16,17,18, 21,22,24,25,26,29,30,33,35,36,38	95.09	0.952	0.951	0,987
Wrapper Subset Evaluation	09	1,2,9,11,18,23,26,32,36	95.84	0.959	0.958	0,986
Proposed system	10	**1,9,11,17,18,20,28,32,37,40**	**96.32**	**0.963**	**0.962**	**0.987**

Compared to wrapper-based approaches from Weka, HGSOEO-FS sustained higher performance with fewer or an equal number of features. Most notable is its edge over Classifier Subset Evaluation, which chose over twice the number of features with less accuracy. This underlines the proposed method's capability of selecting a minimal but effective feature subset without sacrificing predictive performance.

Comparison with Metaheuristic Algorithms for Feature Selection

The last phase is to compare HGSOEO-FS with three classical evolutionary algorithms popular in the feature selection problem: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Whale Optimization Algorithm (WOA). The experiments were set with the same classifier parameters and assessment protocols for all of them. The results are presented in Table 4.14.

Table 4.14: Comparison between the proposed method and others

Method	# Selected Features	Accuracy %	Precision	Recall	AUC
PSO-FS	21	96.15	0,962	0,961	0,991
GA-FS	15	95.93	0,96	0,959	0,990
WOA-FS	11	95.14	0.952	0.951	0,986
Proposed system	**10**	**96.32**	**0.963**	**0.962**	**0.987**

In comparison to PSO-FS, GA-FS, and WOA-FS, the proposed method had the best accuracy, precision, and recall despite having select the least number of features. Although PSO-FS had a slight edge over HGSOEO-FS in AUC, it had to use more than twice the number of features. This supports the assertion that HGSOEO-FS provides a better balance between performance and feature compactness and is thus extremely well-suited for practical applications like social media spam detection where model simplicity and speed are paramount.

4.6 Equilibrium Optimizer for Feature Selection in Clustering

This section presents and discusses the results obtained from the use of the Equilibrium Optimizer (EO) for feature selection in clustering tasks. The objective is to enhance clustering quality by identifying a small subset of relevant features. The proposed method is evaluated across six benchmark datasets using the Adjusted Rand Index (ARI) as the metric to assess clustering performance and the parameters settings in Table 4.3, Subsection 4.3.2.

4.6.1 Internal comparison

This work evaluates the performance of the proposed EO-based feature selection method through an internal comparison that considers both the number of selected features and the resulting clustering quality.

4.6.1.1 Results of the proposed method on benchmarks

As shown in Table 4.15, the method achieves substantial reduction of features on all datasets, while also maintaining strong performance in clustering.

Table 4.15: The approach's outcomes measured by the number of features and fitness score.

No.	Dataset	No. of selected features	Reducing percentage	Fitness score
1	Iris	2	50%	0.290
2	Wine	6	50%	0.034
3	Segment	6	68%	0.326
4	Breast Cancer	6	80%	0.096
5	Ionosphere	13	61%	0.277
6	Spambase	19	66%	0.427

These findings show that the method effectively reduces the dimensionality whilst maintaining low fitness scores, which is consistent with high ARI values. The highest reduction

is observed in the Breast Cancer dataset, where 80% of the features are eliminated. This overall indicates that the method rids the non-contributing features and retains the most relevant ones for clustering.

4.6.1.2 Clustering Performance Before and After Feature Selection

To further validate the method, we compare the ARI score before and after feature selection to assess the impact of the selected features on clustering quality. The results are shown in Table 4.16.

Table 4.16: Comparison between the ARI score with all features and the selected features.

No.	Dataset	ARI with all features	ARI with the selected features
1	Iris	0.594	0.709
2	Wine	0.901	0.960
3	Segment	0.438	0.673
4	Breast Cancer	0.649	0.903
5	Ionosphere	0.394	0.722
6	Spambase	0.313	0.572

In all cases, the ARI score improves after applying feature selection, confirming that the elimination of irrelevant or redundant features results in higher quality clustering. The most improvements are observed in high-dimensional datasets such as Ionosphere and Breast Cancer, highlighting the capability of the method in dealing with complex data.

4.6.2 External comparison

An external comparison is conducted between the EO-based method and the widely used Particle Swarm Optimization (PSO) algorithm under identical experimental conditions. The outcomes are summarized in Table 4.17.

Table 4.17: Comparison between EO-FS and PSO-FS

No.	Dataset	PSO-FS		The proposed method	
		No. of selected features	ARI %	No. of selected features	ARI %
1	Iris	**2**	**70.93**	**2**	**70.93**
2	Wine	11	93.45	**6**	**96.50**
3	Segment	13	56.52	**6**	**67.33**
4	Breast Cancer	16	82.47	**6**	**90.34**
5	Ionosphere	33	34.60	**13**	**72.21**
6	Spambase	34	52.82	**19**	**57.24**

On most datasets, EO-FS outperforms PSO-FS both in terms of selected features and the ARI score. For instance, on the Ionosphere dataset, EO-FS achieves more than double the ARI score that PSO-FS achieves with fewer than half the number of features. The consistency of these improvements across datasets further confirms the efficiency and strength of the proposed method.

Overall, the experimental outcomes obtained via various evaluation perspectives all consistently validate the efficacy of the proposed EO-based feature selection method for clustering. The method not only attains notable dimensionality reduction, but it also enhances the quality of the clustering in all the datasets. Its superiority over PSO-FS in the majority of instances further enhances its practical applicability. The findings substantiate the premise that the integration of feature selection with unsupervised clustering produces more structured data and significant clusters.

4.7 Conclusion

This chapter presented and analyzed the performance of the proposed metaheuristic-based feature selection approaches on different datasets and learning paradigms. The comparative analysis between EO and HGSO illustrated the respective strengths of both algorithms for supervised classification problems. The hybrid HGSOEO method was found to be effective

in achieving a trade-off between accuracy and reduction in dimensionality, especially on real data like the Twitter Spam Detection dataset. In addition, the extension of EO for clustering-based feature selection yielded promising results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The research in this thesis tackles one of the most significant challenges in contemporary machine learning and data science: the effective selection of pertinent features from large-scale high-dimensional data via intelligent optimization methods. With growing data volume and dimensionality, the demand for precise and efficient feature selection is expanding not just for enhancing predictive accuracy but also for decreasing computational expenses and elevating model interpretability.

The study explored three natural phenomenon-inspired metaheuristic algorithms for feature selection: the Equilibrium Optimizer (EO), the Henry Gas Solubility Optimization (HGSO), and a novel hybrid model (HGSOEO) of the two. Each algorithm was developed in its binary version and incorporated in a general feature selection framework. The algorithms' performance was tested on various benchmark datasets from traditional UCI classification datasets and a real-world application Twitter Spam Detection dataset.

In the supervised feature selection scenario, EO and HGSO were compared with various classifiers on benchmark datasets. The latter results showed that HGSO was superior over EO in terms of classification accuracy and feature selection in general, by virtue of the group-based searching and considering the thermodynamics.

To overcome the limitations of both EO and HGSO, the hybrid HGSOEO algorithm was proposed and evaluated. The experimental results demonstrated that HGSOEO consistently outperformed either of its components individually. It improved classification accuracy and achieved outstanding feature reduction, especially on high-dimensional datasets. These improvements were attributed to the algorithm's ability to effectively balance exploration and exploitation during the search process.

In the unsupervised setting, EO was adapted for feature selection prior to clustering. The algorithm was evaluated using measures such as the Adjusted Rand Index (ARI) and Silhouette Score. EO improved the structure and quality of clustering results and demonstrated promise in unsupervised settings. The selected feature subsets retained the most informative dimensions so that clustering models were better able to discern the underlying structure of the data.

Overall, the results in this thesis validate that metaheuristic algorithms—particularly when hybridized—can provide enormous benefits in feature selection. They can efficiently handle large, complicated feature spaces and determine compact, high-quality subsets that enhance learning performance and model interpretability.

5.2 Future Work

Although the findings presented in this thesis are promising, some areas for additional research exist:

- **Multi-Objective Optimization:** Although this study combined classification error and number of features into one fitness function, the adoption of a multi-objective optimization strategy (e.g., Pareto-based) can allow non-compensatory and concurrent optimization of competing objectives.

- **Adaptive Parameter Control:** The implementation of the algorithms relies on manually adjusted parameters depending on the experiment to determine the appro-

priate parameters. The use of dynamic or self-adaptive parameter control mechanisms would enable the algorithms to modify their behavior throughout the search, thereby enhancing convergence rates and robustness over a wider variety of problem instances.

- **Expanded Use in Unsupervised Learning:** Future work must place a stronger focus on feature selection in clustering and unsupervised applications, particularly the usual nature of the real-world datasets, which often lack labeled instances. In such scenarios, identifying relevant features without supervision is essential for discovering meaningful structures and patterns within the data.

This thesis has proven the effectiveness of the intelligent algorithms not only in handling the complexity of feature selection but also in outperforming the traditional techniques. Through empirical validation and hybridization, this work adds to research that connects nature-inspired computing and applied machine learning. With further refinement and integration, such methods hold strong potential for widespread application in data-intensive and decision-critical domains.

Bibliography

[1] Neha Sharma, Reecha Sharma, and Neeru Jindal. "Machine learning and deep learning applications-a vision". In: *Global Transitions Proceedings* 2.1 (2021), pp. 24–28.

[2] G Thippa Reddy et al. "Analysis of dimensionality reduction techniques on big data". In: *Ieee Access* 8 (2020), pp. 54776–54788.

[3] Amirreza Rouhi and Hossein Nezamabadi-Pour. "Feature selection in high-dimensional data". In: *Optimization, learning, and control for interdependent complex networks* (2020), pp. 85–128.

[4] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. "Feature selection for high-dimensional data". In: *Progress in Artificial Intelligence* 5 (2016), pp. 65–75.

[5] Xuan Huang, Lei Wu, and Yinsong Ye. "A review on dimensionality reduction techniques". In: *International Journal of Pattern Recognition and Artificial Intelligence* 33.10 (2019), p. 1950017.

[6] Dipti Theng and Kishor K Bhoyar. "Feature selection techniques for machine learning: a survey of more than two decades of research". In: *Knowledge and Information Systems* 66.3 (2024), pp. 1575–1637.

[7] Jie Cai et al. "Feature selection in machine learning: A new perspective". In: *Neurocomputing* 300 (2018), pp. 70–79.

[8] Tansel Dokeroglu, Ayça Deniz, and Hakan Ezgi Kiziloz. "A comprehensive survey on recent metaheuristics for feature selection". In: *Neurocomputing* 494 (2022), pp. 269–296.

[9] Yuanyuan Gao, Yongquan Zhou, and Qifang Luo. "An efficient binary equilibrium optimizer algorithm for feature selection". In: *IEEE Access* 8 (2020), pp. 140936–140963.

[10] Prachi Agrawal et al. "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)". In: *IEEE Access* 9 (2021), pp. 26766–26791.

[11] Soumaia Kahloul and Djaafar Zouache. "An enhanced henry gas solubility optimization algorithm using transfer functions for feature selection problem". In: *Multimedia Tools and Applications* (2025), pp. 1–45.

[12] Michal Moran and Goren Gordon. "Deep curious feature selection: A recurrent, intrinsic-reward reinforcement learning approach to feature selection". In: *IEEE Transactions on Artificial Intelligence* 5.3 (2023), pp. 1174–1184.

[13] Zohre Sadeghian et al. "A review of feature selection methods based on meta-heuristic algorithms". In: *Journal of Experimental & Theoretical Artificial Intelligence* 37.1 (2025), pp. 1–51.

[14] Afshin Faramarzi et al. "Equilibrium optimizer: A novel optimization algorithm". In: *Knowledge-Based Systems* 191 (2020), p. 105190.

[15] Fatma A Hashim et al. "Henry gas solubility optimization: A novel physics-based algorithm". In: *Future Generation Computer Systems* 101 (2019), pp. 646–667.

[16] Khaoula Zineb Legoui, Sofiane Maza, and Abdelouahab Attia. "Equilibrium optimizer and henry gas solubility optimization algorithms for feature selection: comparison study". In: *2022 5th International Symposium on Informatics and its Applications (ISIA)*. IEEE. 2022, pp. 1–6.

[17] Khaoula Zineb Legoui et al. "Hybrid Henry gas solubility optimization and the equilibrium optimizer for feature selection: real cases with Twitter spam detection". In: *Knowledge and Information Systems* (2024), pp. 1–30.

[18] Khaoula Zineb Legoui, Abdelouahab Attia, and Sofiane Maza. "Equilibrium Optimizer for Feature Selection in Clustering". In: *2024 International Conference of the African Federation of Operational Research Societies (AFROS)*. IEEE. 2024, pp. 1–5.

[19] Yun Li, Tao Li, and Huan Liu. "Recent advances in feature selection and its applications". In: *Knowledge and Information Systems* 53 (2017), pp. 551–577.

[20] Gopi Manikandan and S Abirami. "Feature selection is important: state-of-the-art methods and application domains of feature selection on high-dimensional data". In: *Applications in Ubiquitous Computing* (2021), pp. 177–196.

[21] Vipin Kumar and Sonajharia Minz. "Feature selection". In: *SmartCR* 4.3 (2014), pp. 211–229.

[22] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers & electrical engineering* 40.1 (2014), pp. 16–28.

[23] Manoranjan Dash and Huan Liu. "Feature selection for classification". In: *Intelligent data analysis* 1.1-4 (1997), pp. 131–156.

[24] Avrim L Blum and Pat Langley. "Selection of relevant features and examples in machine learning". In: *Artificial intelligence* 97.1-2 (1997), pp. 245–271.

[25] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29. wh freeman New York, 2002.

[26] Mohamed Abdel-Basset, Laila Abdel-Fatah, and Arun Kumar Sangaiah. "Metaheuristic algorithms: A comprehensive review". In: *Computational intelligence for multimedia big data on the cloud with engineering applications* (2018), pp. 185–231.

[27] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In: *Computers & Operations Research* 13.5 (1986), pp. 533–549.

[28] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. "Optimization by simulated annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[29] Fred Glover. *Tabu Search*. ORSA Journal on Computing, 1989.

[30] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[31] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. ieee. 1995, pp. 1942–1948.

[32] Marco Dorigo, Mauro Birattari, and Thomas Stützle. "Ant colony optimization". In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39.

[33] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing, 1997.

[34] Maha Nssibi, Ghaith Manita, and Ouajdi Korbaa. "Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey". In: *Computer Science Review* 49 (2023), p. 100559.

[35] Kanchan Rajwar, Kusum Deep, and Swagatam Das. "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges". In: *Artificial Intelligence Review* 56.11 (2023), pp. 13187–13257.

[36] Zhongqiang Ma et al. "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms". In: *Swarm and Evolutionary Computation* 77 (2023), p. 101248.

[37] Mohamed Abd Elaziz et al. "Advanced metaheuristic optimization techniques in applications of deep neural networks: a review". In: *Neural Computing and Applications* (2021), pp. 1–21.

[38] Mohammad Reza Hassanzadeh and Farshid Keynia. “An overview of the concepts, classifications, and methods of population initialization in metaheuristic algorithms”. In: *Journal of Advances in Computer Engineering and Technology* 7.1 (2021), pp. 35–54.

[39] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. “A robust EM clustering algorithm for Gaussian mixture models”. In: *Pattern Recognition* 45.11 (2012), pp. 3950–3961.

[40] Kyumin Lee, Brian Eoff, and James Caverlee. “Seven months with the devils: A long-term study of content polluters on twitter”. In: *Proceedings of the international AAAI conference on web and social media*. Vol. 5. 1. 2011, pp. 185–192.