Date of publication DEC-31, 2024, date of current version MAY-18, 2024.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.23.4.3753

Optimizing Multi-Layer Perceptron for Predicting Learner Migration Patterns: A Methodological Exploration

FRANS RAMPHELE[1], ZENGHUI WANG[2], ADEDAYO YUSUFF[2]
[1]Department, Department of Computer Science, University of South Africa, Florida, Johannesburg, South Africa,1709
[2]Department of Electrical Engineering, University of South Africa, Florida, Johannesburg, South Africa,1709

Corresponding author: Frans Ramphele (e-mail: letsukulo.ramphele@ gmail.com).

ABSTRACT This research presents a novel approach to multilayer perceptron's (MLP) hyper-parameter optimization in solving learner migration problems in Limpopo, South Africa. While acknowledging the presence of various hyper-parameter optimization techniques, their applicability, strengths, and limitations differ. Our approach utilizes meta-heuristics, offering an efficient and adaptable method for complex search spaces and global exploration of optimal solution candidates. The social ski-driver (SSD) algorithm *-originally designed for optimizing support vector machines (SVMs)-* and cultural algorithm (CA) were utilized to determine the optimal hyper-parameter configuration for the MLP. The MLP was intended to predict the likelihood of a learner migrating, the reasons for migration, and the distance the learner will migrate to the next school. The two optimizers were run on sample data split into five folds, producing ten hyper-parameter sets (five pairs). The MLP was then built with each parameter set and subsequently run on a new dataset partitioned into five folds. The model performance results were compared using evaluation metrics such as the f1 score, variance analysis, and the Wilcoxon Signed-Rank test. There were no significant performance differences between the SSD and CA hyper-parameters, demonstrating the effectiveness of the SSD algorithm in optimizing neural networks. The CA-derived parameter set was selected due to its lowest variances across the datasets and its strong alignment with the convergence principles of the Berger-Tal multidisciplinary framework on the exploration-exploitation trade-off, providing a solid foundation for our findings.

KEYWORDS Multi-Layer Perceptron (MLP), Cultural algorithm (CA), Social Ski-driver (SSD), Multi-Objective Optimization, Learner Migration, Exploration-Exploitation framework

I. INTRODUCTION

THE nature of human migration and displacement is inherently complex, characterized by a combination of predictable and unpredictable occurrences [1]. Education planners face significant challenges in this space as they navigate a planning landscape that is marked by volatility, uncertainty, complexity, and ambiguity caused by spontaneous learner migration phenomena. These challenges demand optimized inputs into educational policies that clearly define the problem and promote efficient learning and global competitiveness [2], [3]. Developing such efficient solutions has been of interest for a considerable period in EDM (Education Data Mining) [2]. However, most of the optimization problems are characterized by the presence of highly non-linear objective and constraint functions with mixed types of variables. In addition, real-world optimization problems lack a fixed form and the objective and constraints functions as well as their derivatives are not always available. Finding acceptable solutions in this case is not always easy and remains an open problem in this area [4]. Meta-heuristic algorithms have gained popularity and acceptance among researchers in reducing the uncertainty around optimization problems. This form of algorithm is inspired by successful processes in nature which among others include observable physical, ecological, and social phenomena [4].

The Culture Algorithm (CA) and Social Ski-Driver algorithms, which will serve as the foundation of this study, drew inspiration from the evolutionary aspects of human culture and the downhill behavior of ski drivers, respectively [5], [6]. Some of the meta-heuristic in this group include the Harmony Search algorithm (HS) inspired by musicians' harmony improvisation act; the Fireworks Algorithm (FA) inspired by observing fireworks explosions; Teaching learning-based algorithm (TLBO) inspired by the interaction between a teacher and students; Imperialist Competitive Algorithm (ICA); Exchange Market Algorithm (EMA); Soccer League Competition (SLC); Brainstorm Optimization (BSO) etc. [5].

The human-inspired meta-heuristic have proven themselves and gained recognition in numerous applications such as process scheduling, fault tolerance scheduling, image processing, engineering optimization problems, etc. [4], [7]. Nonetheless, there are some challenges, mainly from a theoretical point of view, related to the human-inspired meta-heuristic. First, the precise circumstances in which these algorithms must be deployed remain one of the greatest challenges. Second, the human-inspired meta-heuristic includes parameters that essentially rely on algorithms. The lack of a general mechanism to methodically fine-tune the parameters to improve the performance of the underlying algorithm is another challenge. Lastly, the need to compare different algorithms for performance forces researchers to look ahead before selecting appropriate performance parameters [7].

Maheri et al. [4] undertook a detailed review of the Culture Algorithm (CA) and its applications in science and engineering, focusing on recent developments. The study argued that the CA has demonstrated success in addressing diverse optimization challenges across various engineering and science disciplines. The study emphasized the heightened interest in CAs among researchers. Notably, the review finds different types of CAs, including hybridized, improved, multi-objective, multi-population, chaotic, and fuzzy. Despite this maturity, the review identified several critical issues that merit attention. First, the study advocates exploring hybridization possibilities with recently introduced CA variants beyond genetic algorithms (GA), differential evolution (DE), and particle swarm optimization (PSO). It was further argued that there is a need to develop efficient influence and update functions to enhance the utilization of knowledge sources and maintain population diversity to prevent premature convergence. Furthermore, the study highlighted the need for a rigorous examination of the influence of different knowledge sources on CA performance across various problem domains.

On the contrary, the social ski-driver algorithms represent a relatively recent development with limited documented applications and achievements in the existing literature. The SSD was first created to improve the hyper-parameters of Support Vector Machines (SVM) and class imbalance issues in that area [8]. The SSD has recently been explored for novel applications, especially in feature selection. Notably, a study has emerged wherein the SSD was integrated with Late Acceptance Hill Climbing (LAHC) for feature selection [9]. The findings suggest that the SSD (LAHC) outperformed other meta-heuristic wrapper-based feature selection methods, including but not limited to Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Lion Optimizer, Gray Wolf Optimizer (GWO), Multi-Objective GA (HMOGA), Tribe Competition-based GA (TCbGA), and Grasshopper Optimization Algorithm (GOA) [9].

The study by Ahmad et al. [10] provides insights and a comprehensive picture of potential challenges and prior contributions in the EDM. The study reviewed 1497 publications from 1990 to 2022 (32 years) and found that most of the contributions are in the form of recommendation systems and students' performance evaluation systems. There has been very little effort to use meta-heuristics in solving multi-objective optimization problems in the context of learner migration. The focus of this study is to demonstrate the use of CA and SSD to optimize the *MLP (Multi-Layer Perceptron) hyper-parameters and subsequently predict the likelihood of a learner to migrate, the reasons for migration and the range of distance they will migrate to the next school*.

This paper advances the field of Education Data Mining (EDM) research through innovative applications of meta-heuristics to tackle intricate educational challenges. The application of evolutionary algorithms for optimizing multilayer perceptron (MLP) hyperparameters represents a significant contribution, notably expanding the scope of SSD from its initial focus on SVM hyperparameters optimization. Additionally, the distinctive contribution is underscored by incorporating learner migration data from Limpopo, South Africa, offering a context-specific dimension to the optimization approach. The paper's outline includes a literature review, theoretical framework, methods and materials, experiment design and execution. Furthermore, the findings are presented, and their implications and future work are discussed.

II. THEORITICAL FRAMEWORK

A. HYPER-PARAMETER SELECTION

Hyper-parameter optimization is considered a key phase in building effective machine learning models, especially for deep neural networks and decision trees with inherently many hyper-parameters [11]. In recent times, there have been numerous techniques to optimize hyper-parameters. These techniques can broadly be categorized into an estimation of generalization error, numerical optimization methods, and non-numerical optimization methods [12].

The estimation of generalization error is the most used technique for hyperparameter optimization. What is core in its architecture is the search space (candidate hyper-parameters) and estimation strategy [13]. Typical search spaces include manual search, random search, and grid search, which are sometimes referred to as decision-theoretic search spaces. This nomenclature stems from the intrinsic mechanism they employ, connecting the concept of search space with the estimation strategy[14]. Grid search (GS) explores a fixed domain, while random search (RS) randomly selects combinations within time and resource constraints. In this category of optimizers, we also have Bayesian optimization (BO) models which determine the next hyper-parameter value based on the previously tested hyper-parameter results, reducing unnecessary evaluations, and identifying optimal combinations in fewer iterations compared to GS and RS [11]. The grid and manual search are the most used search spaces due to their simplicity and independence from prior information.

Gradient descent and quasi-Newton techniques are numerical optimization methods used to find the minimum of a function. They aim to adjust model parameters to minimize a cost or objective function. Gradient descent moves toward the steepest decrease, while quasi-Newton uses an approximation of the Hessian matrix to determine step direction. These algorithms are more sophisticated and can converge faster in some cases.

However, they also suffer issues such as local optima and are dependent on a specific starting point [13], [15].

Non-numerical optimization methods like evolutionary algorithms are effective for non-differentiable and non-convex problems. However, these methods often have high time complexities due to inefficient search strategies and require significant processing power [12],[16]. Furthermore, each presents distinct traits, making them appropriate for addressing specific problems but not necessarily all. It is, therefore, crucial to carry out comprehensive comparisons among these techniques prior to their implementation [7], [11]. Algorithms in this group include culture algorithm (CA), social ski-driver (SSD), Genetic algorithms (GA) and Particle swarm optimization (PSO).

B. CULTURE ALGORITHM

The culture algorithm belongs to the evolutionary computing family. Algorithms in this family use metaphorical concepts, principles, and mechanisms inspired by natural systems and evolution to solve complex computing problems. Most work in evolutionary computing focuses on the processes of natural selection and genetics. However, there is a strong belief among researchers that cultural evolution allows societies to evolve or adapt to their environment faster than biological evolutionary processes [17]. The culture algorithm (CA) was originally developed by Reynolds as an extension of the genetic algorithm. However, it has evolved to incorporate other evolutionary algorithms as the basis of its framework [18]. Structurally, CA aligns with Renfrew's THINK model in terms of a dual inheritance framework that includes a belief space composed of individual and group mappa and a trait-based population space [19]. The two components are linked by communication protocols. The belief space is shared amongst the population, which exposes the principle of collective intelligence [20]. The representation in Fig. 1 shows a basic computational framework of the CA.

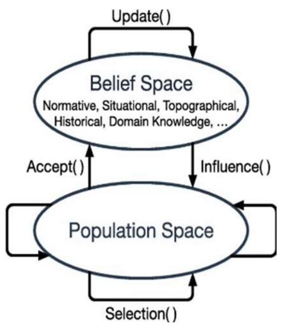
Update( )
Belief Space
Normative, Situational, Topographical, Historical, Domain Knowledge, ...
Accept( )
Influence( )
Population Space
Selection( )

Figure 1. Culture Algorithm diagram [18]

The population space consists of autonomous solution candidates, while the belief space is considered a repository for evolutionary global knowledge. The evolutionary knowledge stored in the belief space can influence agents in the population of space through the influence function, and knowledge from the population space can be transferred to belief through the acceptance function. The population space in CAs can be modelled by population-based evolutionary meta-heuristics such as Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO) [21].

The process of CA evolution starts with the random initialization of the population while simultaneously building the initial knowledge base. Initially, the two spaces devolve independently. Then selected agents from the population space are used to update the belief space. Once the knowledge sources are updated, the belief space will guide the development of the population space in reverse. These procedures continue until the termination state is reached [21], [22]. There are five categories of knowledge sources in the CA that can be used to solve a range of problems [21]:

Table 1. CA Knowledge Space

Knowledge Space	Purpose
Situational	Introduced by Chung to solve real-valued function optimization problems in the static environment. The best solutions of all the generations are stored in this component
Normative	The normative knowledge describes the range of acceptable behaviour for solution candidates.
Topographic	Topographic knowledge sources can articulate the spatial pattern of individual behaviour
Domain	Used to solve dynamic optimization problems through dynamic monitoring of the environment and predicting the evolutionary trend
Historical	Is considered the log, in which important events are recorded during the evolution of the population

The mathematical representation of the culture algorithm is depicted below [19], [20]. The belief space is represented as:

$$B = (S, N) \tag{1}$$

where $\boldsymbol{S}$ is the situational knowledge component which consists of the best solutions from all generations, and $\boldsymbol{N}$ is the normative knowledge component. The $\boldsymbol{S}$ and $\boldsymbol{N}$ in (1) can further be expressed as:

$$S = \{y_l : l = 1, \dots, n_s\} \tag{2}$$
$$N = (X_1, X_2, \dots X_{nx}) \tag{3}$$

where $\boldsymbol{n_s}$ is the total number of solution candidates in the situational knowledge component and $\boldsymbol{n_x}$ is the total number of dimensions in the normative knowledge component. Each dimension $\boldsymbol{X_j}$ stores the following information:

$$X_j = (I_j, L_j, U_j) \tag{4}$$

where $\boldsymbol{I_j}$ signifies the closed interval for each solution, $I_j = [x_{min,j}, x_{max,j}] = \{x : x_{min,j} \le x \le x_{max,j}\}$. $\boldsymbol{L_j}$ and $\boldsymbol{U_j}$ represent the scores of the lower and upper bound respectively.

The other knowledge spaces (e.g *domain, historical, topographic*) can be accessed by updating x_{min} and x_{max} for the jth variable in each iteration to reach the new bounds. For example, adjusting the normative knowledge components allows an accepted response like $\boldsymbol{x_{1,j}}$ to change the space as follows:

$$x_{min,j}^{new} = \begin{cases} x_{1,j} & x_{1,j} \leq x_{min,j}\ or\ f(x_{1,j}) \leq L_j \\ x_{min,j} & otherwise \end{cases} \quad (5)$$

$$x_{max,j}^{new} = \begin{cases} x_{1,j} & x_{1,j} \leq x_{max,j}\ or\ f(x_{1,j}) \leq U_j \\ x_{max,j} & otherwise \end{cases} \quad (6)$$

$$L_j^{new} = \begin{cases} f(x_{1,j})\ x_{1,j} \leq x_{min,j}\ or\ f(x_{1,j}) \leq L_j \\ L_j \quad otherwise \end{cases} \quad (7)$$

$$U_j^{new} = \begin{cases} f(x_{1,j})\ x_{1,j} \leq x_{max,j}\ or\ f(x_{1,j}) \leq U_j \\ U_j \quad otherwise \end{cases} \quad (8)$$

where $f(.)$ is the fitness function. Equations (5) and (6) suggest that the new response is equal to the recently accepted component OR otherwise is equal to the previous values. Similarly, in (7) and (8), the new upper and lower cost functions must be lower than the previous one to be accepted or otherwise the previous values remain. Similar to the formula mentioned earlier for normative knowledge components, the situational knowledge component can be modified as follows:

$$S^{new} = y^{new} = \begin{cases} min_{l=1,\dots,n_B}\{x_l\} & if\ f(min_{l=1,\dots,n_B}\{x_l\} < f(y) \\ y & otherwise \end{cases} \quad (9)$$

where n_B is the total number of accepted solutions. At each iteration, the situational knowledge component - as shown in (9) - keeps only one optimal solution. The acceptance function is used to select the optimal solutions from the population to guide the belief space.

The acceptance function can either be static or dynamic. The static acceptance function will select the top n% (fixed) of solutions. The utilization of dynamic selection allows for the establishment of diverse methodologies to determine the appropriate number of individuals through the implementation of a dynamic acceptance function. The belief space can be updated by the situational and normative knowledge components for the accepted individuals. Similarly, the belief space must update population space through the influence function. The influence function can be updated using three strategies outlined in (10), (11) and (12) based on the knowledge component. The first option involves using the normative knowledge component to determine the step sizes in the offspring generation process and is expressed as follows:

$$x_{i,j}^{new} = x_{i,j} + \delta_j.N(0,1) \quad (10)$$

where $\delta_j = \alpha(x_{max,j} - x_{min,j})$ is the step size, N is a random value and α is a coefficient, both between 0 and 1.

The second option involves using the situational knowledge component to determine the change in direction. In this situation, the direction of the response is calculated as:

$$x_{i,j}^{new} = \begin{cases} x_{i,j} + \delta_j.|N(0,1)| x_{i,j} < y_i \\ x_{i,j} - \delta_j.|N(0,1)| x_{i,j} < y_i \\ x_{i,j} + \delta_j.N(0,1) otherwise \end{cases} \quad (11)$$

And lastly, the normative knowledge component is used for both search direction and step size and can be expressed as follows:

$$x_{i,j}^{new} = \begin{cases} x_{i,j} + \delta_j.|N(0,1)| x_{i,j} \leq x_{min,j} \\ x_{i,j} - \delta_j.|N(0,1)| x_{i,j} \geq x_{max,j} \\ x_{i,j} + \delta_j.N(0,1) otherwise \end{cases} \quad (12)$$

where the step size is equal to equation (10).

C. SOCIAL SKI-DRIVER (SSD) ALGORITHM

The social ski-driver (SSD) optimization algorithm is a human-based evolutionary metaheuristic inspired by PSO (Particle Swarm Optimizer, GWO (Gray wolf optimizer) and SCA (Sine cosine algorithm) evolutionary algorithms [23]. The algorithm adopts an iterative approach to search for optimal solutions, emulating the behaviour of ski drivers when they go downhill [6]. The algorithm was initially developed with the sole purpose of optimizing the parameters of SVMs and dealing with class-imbalanced datasets in that space [8]. The SSD algorithm uses the following parameters in its computational framework [8], [23].

•***Agent position*** represented as *(*$x_i \in \mathcal{R}^d$*), where* d is a dimensional search space and x_i is an agent occupying a position in the search space

•***Previous best position*** represented *as* P_i.As the SSD executes, all the solutions are at some point evaluated for their fitness. Each agent's fitness value is compared with the fitness value associated with its current position. The position that demonstrates the superior fitness value is stored in the P_i.

•***Mean global solution*** represented *as* **M***:* The **M** represents either a convergence point or optimal solution and is calculated as follows:

$$M^t = \frac{x_\alpha + x_\beta + x_\gamma}{3} \quad (13)$$

where $x_\alpha, x_\beta, x_\gamma$ indicate the best three solutions and M^t indicates the mean of the best three solutions in the current iteration *t.*

The velocity of the agents is represented as v_i. *The* 'agents' positions are adjusted by adding the velocity (v_i) to the current position of the agent using the following equation:

$$x_i^{t+1} = x_i^t + v_i^t \tag{14}$$

where

$$v_1^{t+1} = \begin{cases} c\sin(r_1)\,(p_i^t - x_i^t) + \sin(r_1)\,(M^t - x_i^t)\ if\ r_2 \le 0.5 \\ c\cos(r_1)\,(p_i^t - x_i^t) + \cos(r_1)\,(M^t - x_i^t)\ if\ r_2 > 0.5 \end{cases} \tag{15}$$

where

v_i is the velocity of the agent x_i
r_1, r_2 are random numbers in the interval [0,1]
p_i^t is the best solution of the tth iteration
c is the parameter that controls the balance between exploration and exploitation

Fig. 2 depicts a movement of agents in the SSD proximal space. The movement of agents towards the convergence point (**M**) is not linear because of the sin and cos functions. The SSD resembles the PSO logic when calculating the previous position of the agent and the GWO when the agents converge to the global minima [23].

The algorithm begins by creating agents and randomly initializing parameters such as the agent positions (x_i) and velocities (v_i). The maximum number of iterations (epoch) and the population size are derived from the defined problem. During processing, the agents update their position in (14) and adjust their velocity in (15).

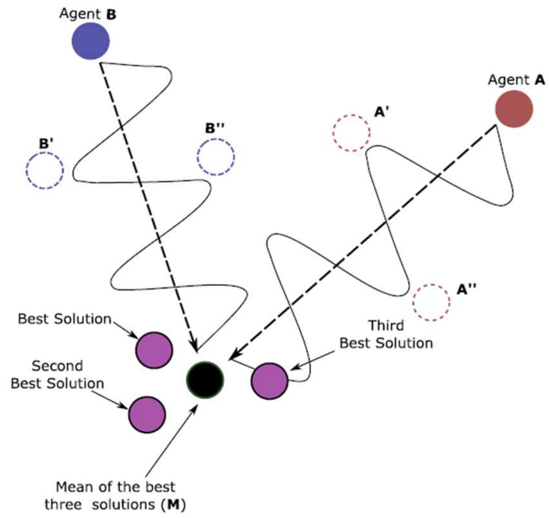

Figure 2. Demonstration of agent movement in the SSD space [23]

The agents update velocity (v_i) on two terms; the distance between the previous best position and the current position ($p_i^t - x_i^t$) ; and the distance between the mean global solution and the current position ($M^t - x_i^t$). The algorithm adapts a parameter *c* linearly over iterations. Agents within the population move in search of optimal solutions through a combination of exploration and exploitation which is controlled by the parameter *c*. The algorithm employs sine and cosine functions based on a random selection criterion (r_1 *or* r_2) to decide on the movement strategy for each agent. With the sine function active, the algorithm updates the velocity, considering the best local and global solutions, and adjusts the agent's movement accordingly. With the cosine function active, the algorithm adjusts the velocity using a different movement strategy to balance exploration and exploitation. Agents generate new positions by combining their velocity vectors with random elements. The newly generated positions are corrected to fit within the problem's solution space. The algorithm evaluates the fitness of the new positions and replaces the existing solutions in the population. The process continues for a specified number of iterations or until a termination criterion is met [6], [8], [23].

D. MULTILAYER PERCEPTRON

Neural networks are a subset of machine learning algorithms inspired by the logic that governs the human brain. They have gained considerable attention in multidisciplinary areas like economics, logistics, medicine, security, and finance, as well as in diverse, active research spaces. Neural networks present promising prospects in addressing challenges around handwriting and speech recognition, natural language processing, image recognition and compression [24], [25]. There are different types of Neural networks, e.g., Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Multilayer Perceptron (MLP). These networks differ in their architecture and learning strategies, among others. Henceforth, the discussion will be limited to Multilayer Perceptron (MLP). Fig. 3 shows a computational framework of a simple MLP.

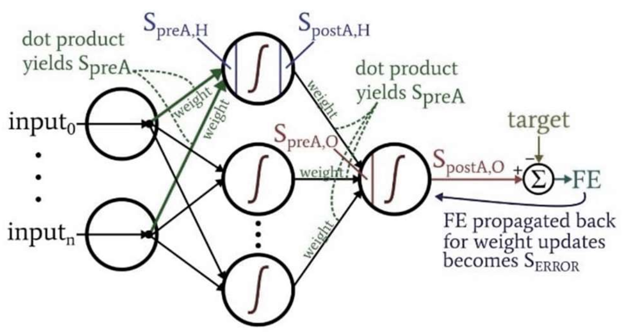
$S_{preA,H}$
$S_{postA,H}$
dot product yields S_{preA}
dot product yields S_{preA}
$input_0$
$input_n$
weight
weight
weight
weight
weight
$S_{preA,O}$
$S_{postA,O}$
target
Σ
FE
FE propagated back for weight updates becomes S_{ERROR}

Figure 3. Multilayer Perceptron [26]

MLP is composed of three or more layers of fully interconnected neurons that compute and transmit information across the network [24], [27], [28]. The calculations performed on each neuron are informed by parameters such as weights and biases [28]. Specifically, in the key formula that drives the computation of an MLP, the goal is to minimize the error function $\boldsymbol{e}(\boldsymbol{w})$ in relation to the weights of the connections [26]. The operation of the multilayer perceptron illustrated in Fig. 3 can be outlined in the following steps: [29]

Step 1: MLP Initialization. The weights and thresholds are initialized using random values. The gradients of weights and current error are also initialized ($\Delta w_{ij} = 0\,; E = 0$), where Δw_{ij} is the gradient of the weight and E is the current error.

Step 2: Training the MLP. Data observations are fed into the model. During processing, the gradient of the weights are memorized and adjusted after each model in the training set and at the end of a training epoch. After processing the entire test set, the model will appropriately adjust the weights.

Step 3: The forward propagation of the signals. Each neuron processes the input data, resulting in the following output :

$$y_j(p) = f\left(\sum_{i=1}^{n} x_i\,(p).w_{ij} - \theta_j\right) \qquad (13)$$

where $\boldsymbol{n}$ is the number of hidden layer inputs for neuron $\boldsymbol{j}$ and $\boldsymbol{f}$ is the activation function. On the output layer, (14) signifies the results of the entire network:

$$y_k(p) = f\left(\sum_{i=1}^{m} x_{jk}\,(p).w_{jk}(p) - \theta_k\right) \qquad (14)$$

where $\boldsymbol{m}$ is the number of inputs for the neuron $\boldsymbol{k}$ from the output layer. The error per epoch is calculated as follows:

$$E = E\frac{(e_k(p))^2}{2} \qquad (15)$$

Step 4: Backpropagation and weight adjustments. Artificial neural networks learn continuously by using corrective feedback loops called backpropagation. This process allows the neural network to adjust the weights and minimize the loss function to improve their predictive and generalization ability [28], [30], [31]. After the error is calculated in (15), its gradient is computed and fed back - *through backpropagation*- into the network to modify the weights. The gradient of the error is then calculated as follows:

$$\delta_k(p) = f'.e_k(p) \qquad (16)$$

where f' is a derivative of the activation function, and the *error*

$$e_k(p) = y_{d,k}(p) - y_k\,(p) \qquad (17)$$

The computational power of MLP comes from the activation functions. Specifically, activation functions in neural networks are used to introduce non-linearity into the output of a neuron. Without activation functions, a neural network would only be able to learn linear functions, which would limit its ability to learn more complex patterns in the data. Common activation functions include the sigmoid function, the ReLU (Rectified Linear Unit) function and the tanh (hyperbolic tangent) function and their formulas are represented below [28], [32].

Table 2. Activation Functions

Sigmoid $f(x) = \frac{1}{1+e^{-x}}$	**Leaky ReLU** $\max(0.1x, x)$
Tanh $\mathbf{tanh}\,(x)$	**Maxout** $\mathbf{max}\,(w_1^T x + b_1, w_2^T x + b2)$
ReLU $Max(0,x)$	**ELU** $\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

say the sigmoid activation function is used; the derivative of its equation is written as follows,

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x).\left(1 - f(x)\right) \qquad (18)$$

equation (16) can therefore be written as follows,

$$\delta_k(p) = y_k\,(p).\,(1 - y_k).e_k(p) \qquad (19)$$

at this point, the gradients of the weights between the hidden and output layers can be updated as follows,

$$\Delta w_{jk}(p) = \Delta w_{jk}(p) + y_j\,(p).\,\delta_k\,(p) \qquad (20)$$

the gradients of errors for neurons in the hidden layer can be calculated as follows,

$$\delta_j(p) = y_j(p).\left(l - y_j\,(p)\right).\sum_{k=1}^{l}\delta_k\,(p).w_{jk}(p) \qquad (21)$$

where $\boldsymbol{l}$ is the number of outputs for the network. The gradients of the weights connecting the input and output layer are then updated as follows:

$$\Delta w_{jj}(p) = \Delta w_{jj}(p) + x_j\,(p).\,\delta_j\,(p) \qquad (22)$$

Step 5: New Iteration. If there are still test vectors in the current training epoch, proceed to step 3. If not, the weights of all the connections will be updated based on the gradients of the weights in (23):

$$w_{jj} = w_{jj} + \eta.\Delta w_{jj} \qquad (23)$$

where η is the learning rate. If the epoch is completed, it is then tested to determine if it fulfills the criterion for termination (*E<Emax or a maximum number of training epochs has been reached*). If not, proceed to step 2; else, the algorithm ends.

Training neural networks require a substantial computational capacity, particularly when dealing with large data sets or networks with numerous layers. This is due to the large number of parameters that need to be optimized during training [28]. Neural networks have a finite capacity to represent and model relationships in the data. If the capacity of the network is too small, it may not be able to accurately represent the underlying patterns in the data. On the other hand, if the capacity is too large, it may overfit the training data and perform poorly on unseen data. The theory of capacity of neural networks states that a neural network with enough capacity can approximate any continuous function to arbitrary accuracy [28].

The ability of a neural network to generalize from the examples in the training set to new cases is an important property. A neural network that has memorized the training examples but does not generalize well to new cases is said to have high variance and low bias. Generalization can be improved by techniques such as regularization, early stopping or dropout. The choice of model architecture, learning rate, and other hyper-parameters can greatly affect the performance of a neural network. These hyperparameters must be carefully chosen and tuned to achieve optimal performance. Hyper-parameter tuning can be done using grid search, random search, or Bayesian optimization. The theory of hyperparameter optimization in neural networks states that there exists a set of hyperparameters that can be found by optimization techniques that will achieve the best performance on a given task [27], [33].

E. MULTI-OBJECTIVE OPTIMIZATION

Multi-Objective Optimization (MOO) is a method used to optimize a system with multiple conflicting objectives. These objectives are typically quantified as mathematical functions, and the goal of MOO is to find the best solution that balances the conflicting objectives [34]. An optimization problem can generally be written in the following generic form [35].

$$\begin{gathered} minimise_{x \epsilon \mathcal{R}^n}\ f_i(x),\ \ (i = 1,2,\dots,M) \\ subject\ to\ \ h_j(x) = 0,\ \ (j = 1,2,\dots,J) \\ g_k(x) \leq 0,\ \ (k = 1,2,\dots,k) \end{gathered}$$

where $f_i(x),\ \ h_j(x) and\ \ g_k(x)$ are functions of the vector,

$$x = (x_1\ , x_2, \dots, x_n)^T$$

The variable x_i of x are called decision variables. The function $f_i(x)$ where $i = 1,2,\dots,M$ is called the objective or cost function, and M is the number of objectives. The space spanned by the decision variables is called the search space R^n while the space formed by the objective function is called the solution space. The h_j and g_k are called constraints.

Multiple objective optimization (MOO) problems are often characterized by complexity and challenge due to the presence of competing objectives. Researchers use various solution philosophies and goals to address MOO problems, depending on their specific perspective and the nature of the problem. Common goals include identifying a representative set of Pareto optimal solutions, comprehending the trade-offs among various objectives, and finding a single solution that aligns with decision-makers preferences [34], [36].A solution is considered pareto optimal, pareto efficient, non-inferior, or nondominated when no other objective can be improved without reducing the value of other objectives [37]. Given the absence of additional information, there may be an infinite number of solutions that are all deemed good. Researchers tackle these problems from different perspectives and have different goals when solving them. These goals may include identifying a representative set of pareto optimal solutions, comprehending the trade-offs among various objectives, and finding a single solution that aligns with decision-makers preferences [36].

F. EXPLORATION-EXPLOITATION DILEMMA and OPTIMAL CONTRACTION THEOREM

The exploration-exploitation dilemma is a fundamental problem in data-driven decision-making processes. It involves the delicate balance between two conflicting strategies. Berger-Tal et al. [38] proposed a multidisciplinary framework to guide discussions on the exploration-exploitation trade-off in assessing the performance of decision-making processes, including machine learning algorithms. The framework provides four knowledge phases: knowledge establishment, knowledge accumulation, knowledge maintenance, and knowledge exploitation. In the knowledge establishment phase, an entity acquires basic information about the environment and resources, relying on internal reserves. In knowledge accumulation, the entity prioritizes obtaining new information while exploiting existing knowledge at a minimal rate. The knowledge maintenance phase focuses on utilizing resources while maintaining knowledge at an optimal level. The final phase, knowledge exploitation, occurs towards the end of an entity's lifespan, shifting focus to exploiting accumulated knowledge at 100 % with zero exploitation. The lifespan of the entity, environmental predictability, learning effectiveness, and temporal unpredictability all have an impact on the length and intensity of these phases. The framework offers insights into the interplay between exploration and exploitation strategies, emphasizing their adaptive roles across different life stages and environmental conditions.

On the other hand, the Optimum Contraction Theorem by Jie Chen et al [39] posits that the complexity of an optimization problem significantly influences the trade-off between exploration and exploitation, consequently affecting the optimizer's behavior. The theorem states that if the problem is difficult for an optimizer, the exploration-exploitation trade-off will lean towards exploration. It further states that the optimizer's strategy can either be exploration-dominated or exploitation-dominated. The exploration-dominated optimizers outperform exploitation-dominated optimizers when the optimization problem becomes complex. The theorem encourages that one needs to have a prior knowledge of the optimizers and particularities of the problem. It argues that such knowledge can assist in transforming a complex problem into a relatively simple one, and assist one in selecting the correct optimizer.

The two frameworks will be used to guide discussions on the convergence behaviour of the two optimizers selected for the study.

III. METHODS AND MATERIALS

A. KNOWLEDGE DISCOVERY IN DATABASES (KDD)

The Knowledge Discovery in Databases (KDD) methodology was employed to guide the development of the MLP and address the objectives of the study. Theoretically, the KDD process is iterative and requires one to follow a machine-learning exercise in a specific order. First, a clear understanding of the problem and its domain is essential. The data preprocessing phase (*data cleaning and integration*) follows, during which data sources are identified and integrated if necessary, and issues such as missing values, data inconsistencies, duplicates, and outliers are handled. Once the data is preprocessed, it is served into the feature selection and dimensionality reduction phase, where domain knowledge and machine learning techniques are used to select valuable features for the data mining phase. The data transformation phase follows, where the data of the selected features is transformed into form and shape ready for the actual data mining phase. The *data mining* phase involves applying machine learning algorithms to expose hidden patterns (*discovered knowledge)* in the data, which is then interpreted, evaluated, and validated to ensure the relevance and reliability [40].

B. DATA ASSEMBLY

The EMIS database has over 400 tables and thousands of attributes describing a learner, an educator, and a school inventory [41]. The research employed Ravenstein's theory of migration as a conceptual framework [42], [43]. The tenets of the theoretical framework were applied to direct the data assembly process from the Education Management Information System (EMIS) database. Features which broadly conformed to the principles in the theoretical framework were selected (Table 4). Table 3 outlines Raveistein's laws of migration.

Table 3. Ravenstein laws of migration [42]

The Ravenstein laws of migration
o Most migrations are over short distances.
o Migration happens in stages and as distance increases, the volume of migration decreases.
o In general, long-range migrants move into urban areas.
o Each migration produces a reverse movement, although not necessarily in the same volume.
o Rural inhabitants are more migratory than urban inhabitants,
o Women are more migratory than men within their own country, but males are more migratory over long distances.
o Migrants are mostly adults. Families seldom migrate from their country of birth.
o Large towns grow more through migration than through natural growth.
o Migration increases with economic development.
o Migration is mostly due to economic causes

Table 4. Features selected for study (Additional details of the features can be found in Appendix A1)

	Description
f01	Learner age at the time of first admission
f02	Learner age when leaving the school
f03	Identifies if a learner is in a school hostel or not
f04	Identifies if a learner has deceased parents
f05	Learner Gender
f06	Grade of a learner
f07	Number of years a learner is admitted to the school
f08	The last grade of the schools
f09	Number of years a learner was in a grade
f10	The home language of a learner
f11	School language of learning and teaching
f12	Identifies if a learner has a disability or not
f13	Number of years in a phase
f14	Language a learner prefers to be taught with
f15	Identifies if a learner was progressed/condoned to a grade or not
f16	Identifies if a learner is benefitting from School Food Programmes
~~f17~~	~~Race of a learner~~
f18	Type of the transport a learner is using to school
f19	District of the school a learner is enrolling
~~f20~~	~~Type of the school a learner is enrolling~~
f21	The sector of the school a learner is enrolling in
f22	Poverty ranking of the school a learner is enrolling
f23	Average school performance
f24	Frequency of displacement
c25	response class to assess the possibility of learner displacement (*1= "displacement". 2= "no displacement"*)
c26	response class indicating the reason for learner displacement (*1=graduated, 2= drop out,3=transfer, 4= no movement*)
c27	response class indicating distance range a learner will displace (0km = *0, 0><6 =1, 6+=2)*

Ravenstein's theory of migration has been widely recognized as a seminal work in the field of migration studies and provides a comprehensive framework for understanding patterns of human migration [42]. The Ravenstein migration theory is determined by push-pull factors [42]. The theory refers to push factors as unfavourable conditions from the point of origin, while pull factors are favourable conditions at the destination. The theory further posits that the push and pull forces are from economic, social, and cultural space [42], [44]. The research undertaken by Raveistein established a strong foundation for subsequent studies, including the work of Lee Everett. Lee supported Raveistein and argued that the decision to migrate is influenced by various factors, such as place of origin, intended destination, intervening obstacles, and personal characteristics [43]. Lee further asserts that the migration process is selective for differentials such as gender, age, education, social class, etc., and these factors affect how one responds to push-pull factors, and the ability to overcome intervening obstacles. The systematic review by [45], [46] examined several studies exploring different factors that contribute to learner migration in South African schools and ways to mitigate their negative effects. This review found that the migration of learners from one school to another is influenced by several factors, including legal frameworks in the education system, school management and leadership practices of schools, school efficiency, infrastructure, and socio-economic factors, among others. The findings of this research align with the Ravenstein's migration theory. These findings offer empirical evidence that supports the ongoing significance of Ravenstein's migration theory in understanding learner migration dynamics

C. DATA PREPARATION

Data preparation involves several activities, such as data cleaning, integration, selection, and transformation, as discussed in the KDD process. This task is challenging, as it involves transforming raw data into a format suitable for data mining without altering its underlying structure or meaning [40], [46] .In practice, this often requires addressing issues in the raw data, such as missing or inconsistent values, outliers, non-standard attributes, duplicate records, and conducting data normalization or discretization and related mappings [40], [46], [47] . Among the 24 predictor variables in Table 4, f01, f02, f07, f23 and f24 were derived from pre-existing features within the dataset [41]. These features were created to capture additional information relevant to learner demographics and academic progress. Feature f23 was derived by combining two features related to the school promotion rate in order to decrease the complexity of the dimensions. Furthermore, the target class c27 was discretized into three intervals to enhance the model's performance and convergence. Duplicate records and records that could not be uniquely identified were removed to ensure data integrity. The labels of all the categorical data were encoded to improve compatibility with the target model, improve performance and reduce the memory footprint of the dataset. Overall, these steps were essential to ensure that the data was suitable for machine-learning exercises [47].

D. FEATURE SELECTION

The sampled data was subjected to a feature selection process to optimize training time, feature performance, and model stability. Boruta, RPART, J48, and Adaboost.M1 were utilized to generate optimal feature sets [41]. The standard deviation of each feature set was calculated and compared. The literature asserts that a low standard deviation or variance is a crucial factor in selecting feature sets, as it suggests reduced variability, indicating greater consistency and stability. It further states that stable features (*with a low standard deviation*) produce more straightforward and interpretable models, which can mitigate the risk of overfitting and enhance the model's performance [48]. The feature set produced through Boruta had a relatively lower standard deviation and aligned with learner migration traits supported by the literature [45], [49], [50], and the Ravenstein theory of migration [42], [43], [51] underpinning this study. This feature set was deemed optimal, with *f17 (learner race)* and *f20 (school type)* being rejected. Table 4 shows the final feature set for the study.

E. SAMPLING METHOD

A sample comprising 10% of the total observations ($\boldsymbol{n = 12,849}$) with 25 features (*excluding f17 and f20*), was obtained using the simple random sampling technique. This basic sampling methodology ensures that each member of the population has an equal chance of being included in the sample. In machine learning, class imbalanced data can lead to classifiers that demonstrate bias towards the dominant class, leading to increased generalization error [52]. Given the complexity of the multi-objective problem the study intends to address, our approach involved focusing solely on the primary response class (c25) to address the issue of class imbalance. The data was partitioned into training and test set using a 70% training and 30% test split, respectively. The Random Forest classifier was used to predict the response class (c25) in the test data, and the following performance information was generated [41], [53].

Table 5. Sample performance metrics

Confusion matrix	Other Performance Metrics
## Reference	##Sensitivity: 1.000
## Prediction 1 2	##Specificity: 1.000
## 1 1896 0	##Pos Pred Value: 1.000
## 2 0 1920	##Neg Pred Value: 1.000
##Accuracy: 1	##Prevalence: 0.497
##95% CI: (0.999, 1)	##Detection Rate: 0.497
##No Information Rate: 0.5031	##Detection Prevalence: 0.497
##P-Value [Acc > NIR] : < 2.2e-16	##Balanced Accuracy: 1.000
## Kappa: 1	

The model correctly recognized all observations in the primary response class *(c25)*. The accuracy, kappa and sensitivity of the model are optimal, which means that the model works well and can recognize all instances of the classes equally [54].

E. EXPERIMENT DESIGN

The study used a Multilayer Perceptron (MLP) to predict three distinct objectives (*c25, c26, and c27*). The initial phase of the experiment involved optimizing hyper-parameters for the MLP. The Scikit Python package was used to implement the MLP. A range of hyper-parameters were identified for optimization. These hyper-parameters include the activation function (*identity, logistic, tanh, ReLU*); the number of hidden layers; maximum iteration bounds ranging from 200 to 500; alpha, which represents the strength of L2 regularization; varying batch sizes (*2, 4, 8, 16, 32*); optimization functions (*LBFGS, SGD, Adam*); and learning rates (*constant, invscaling, adaptive*). The Culture Algorithm (CA) and Social Ski-Driver (SSD) from the Mealpy Python package were used to optimize the hyperparameters of the MLP. The tuner utility in the Mealpy package was implemented to allow the two optimizers to choose their parameters, e.g., the *acceptance rate in the case of CA*, while keeping the epoch and population size the same in both optimizers. The epoch and population size were set to 30 and 25, respectively. The cleansed dataset from the EMIS system was partitioned into DatasetA and DatasetB, containing 128,495 and 314,635 learner records, respectively. DatasetA contained learner records spanning admissions from 2011 to 2016 and was used to generate a 10% sample. The sample was split into 5 folds (*dataset_a1, dataset_a2, dataset_a3, dataset_a4, and dataset_a5*), with approximately 2,570 observations in each fold. Both optimizers were run on the five folds using a 70/30 train/test split, producing ten sets of what are considered, pareto optimal hyper-parameters: *ca_dataset_a1*, *ca_dataset_a2*, *ca_dataset_a3, ca_dataset_a4, ca_dataset_a5, ssd_dataset_a1, ssd_dataset_a2, ssd_dataset_a3, ssd_dataset_a4, and ssd_dataset_a5.* Dataset B contained learner records with admissions from 2017 to 2021. It was split into five folds (*dataset_b1, dataset_b2, dataset_b3, dataset_b4, and dataset_b5*), each containing approximately 62,927 observations. These five folds were used to assess the performance of ten sets of hyper-parameters using a 20/80 train/test split. Considering that all the parameter sets are pareto optimal solutions, it was difficult to randomly select the optimal parameter set for the study. Given this, the ten hyper-parameter sets were assessed using the algorithm's log data to evaluate various performance metrics, including fitness, diversity, the exploration vs. exploitation trade-off, and epoch time. The F1 score was then used to calculate the variability, or standard deviation, of the F1 weights across datasets for each hyper-parameter set (Table 7). The hyper-parameter set with low variability was given a higher ranking. The Wilcoxon Signed-Rank test was further conducted on performance values (F1 score weight of three objectives) derived from the evaluation of ten sets of hyper-parameters. The test was intended to assess whether there are statistically significant differences in performance among the pairs of optimizers (CA and SSD) in each fold of DatasetB.

IV. RESULTS AND DISCUSSION

A. MODEL'S PERFORMANCE

The results offer insight into the functionality and behaviour of the CA and SSD algorithms across multiple aspects. Existing literature and theoretical frameworks were used to understand each model's subtle strategies in optimizing the multi-layer perceptron's hyper-parameters and to address the learner migration problem discussed earlier. One critical concept in the performance of meta-heuristics is convergence analysis. The theoretical development of population-based meta-heuristics is still maturing and the mathematical analysis of the rate of convergence is very difficult to formulate, if not impossible. In many cases, the performance analysis is either done through a comparison of the algorithms and applications on the known problems or through the analysis of the exploration-exploitation tradeoff [55]. Exploration-exploitation trade-off is fundamental in meta-heuristics as it guides the algorithm into the global space where optimum convergence should occur. Aspects such as population diversity, the spread of solutions, epoch time and the algorithm's ability to escape local optima collectively capture the exploration-exploitation trade-off and are integral in understanding how well a meta-heuristic performed [39].

Table 6 compares the CA and SSD across the five-folds of the sample, each with approximately 2570 observations. The two optimizers performed well with an average difference of 0.007 calculated as the average of the absolute difference from the mean performance of the optimizers. The two algorithms selected the activation function randomly based on the emerging patterns in the data, but have all agreed on LBFGS(Limited-memory Broyden-Fletcher-Goldfarb-Shanno) as the choice of the solver/optimizer (Table 6). The alpha values, representing the strength of L2 regularization, vary between CA and SSD, with both adapting the parameter to dataset-specific needs. The learning rate parameter also varies, with *invscaling* and *adaptive* being frequently favoured. The preliminary results show that CA configured with relu activation function, LBFGS optimizer and Invscalling learning rate achieved a higher performance in dataset_a5 (Table 6). Table 7 comparatively shows the mean f1 score performance of the parameter sets and the related ranking based on the standard deviations of the performance values and mean f1 scores across the datasets. Once more, the hyper-parameter set "*ca_dataset_a3*" from CA with the LBFGS optimizer, invscaling learning rate, and *ReLU* activation function achieved a higher mean performance across the five folds of DatasetB and had a low standard deviation (Table 7). Interestingly, the hidden layers of the parameter set "*ca_dataset_a5*" are double that of "*ca_dataset_a3*" while the activation function, solver and learning rate are the same.

Table 6. Hyper-Parameters for global best and worst solutions

dataset	dataset_a1	dataset_a1	dataset_a2	dataset_a2	dataset_a3	dataset_a3	dataset_a4	dataset_a4	dataset_a5	dataset_a5
model	CA	SSD	CA	SSD	CA	SSD	CA	SSD	CA	SSD
Parameter setid	ca_dataset_a1	ssd_dataset_a1	ca_dataset_a2	ssd_dataset_a2	ca_dataset_a3	ssd_dataset_a3	ca_dataset_a4	ssd_dataset_a4	ca_dataset_a5	ssd_dataset_a5
model fitness	2.6397	2.6448	2.6314	2.6346	2.6466	2.6479	2.6508	2.6430	2.6703	2.6492
best hidden layer size	68	11	41	41	44	10	86	20	88	31
best activation function	identity	logistic	relu	logistic	relu	logistic	relu	tanh	relu	tanh
best solver	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs	lbfgs
best alpha	0.0563	0.1000	0.0891	0.1000	0.0395	0.1000	0.0465	0.1000	0.0211	0.1000
best learning rate	invscaling	adaptive	constant	adaptive	invscaling	adaptive	constant	invscaling	invscaling	invscaling
best max iteration	575	250	775	325	475	500	400	625	575	200
best batch size	8	2	16	32	4	2	2	2	4	32

Table 7. Parameter set ranking from F1 score weight of three response variables

Dataset	ca_dataset_a1	ssd_dataset_a1	ca_dataset_a2	ssd_dataset_a2	ca_dataset_a3	ssd_dataset_a3	ca_dataset_a4	ssd_dataset_a4	ca_dataset_a5	ssd_dataset_a5
dataset_b1	2.6119	2.5493	2.5968	2.5476	2.5941	2.5662	2.5733	2.5803	2.5443	2.5793
dataset_b2	2.5957	2.5674	2.6083	2.5628	2.5989	2.5324	2.5683	2.5929	2.5758	2.5958
dataset_b3	2.6125	2.5499	2.5657	2.5664	2.6012	2.5879	2.6131	2.5704	2.6061	2.5683
dataset_b4	2.5894	2.5651	2.6012	2.5649	2.6094	2.5631	2.5837	2.5424	2.5807	2.5844
dataset_b5	2.5906	2.5952	2.6027	2.5538	2.6063	2.5597	2.6313	2.6025	2.5793	2.5763
mean	2.6000	2.5654	2.5949	2.5591	2.6020	2.5618	2.5939	2.5777	2.5772	2.5808
variance	0.0001	0.0003	0.0003	0.0001	0.0000	0.0004	0.0007	0.0005	0.0005	0.0001
standard deviation	0.0114	0.0186	0.0169	0.0081	0.0060	0.0198	0.0272	0.0232	0.0220	0.0102
ranking by mean f1 weight	2	8	3	10	1	9	4	6	7	5
ranking by std of the f1 weight	4	6	5	2	1	7	10	9	8	3

Table 8. Wilcoxon Signed-Rank Test of paired parameter set

Paired parameter set	Test statistic	P-value	Outcome
1. (ca_dataset_a1, ssd_dataset_a1)	1	0.125	There is no evidence of a significant difference between paired groups.
2. (ca_dataset_a2, ssd_dataset_a2)	1	0.125	There is no evidence of a significant difference between paired groups.
3. (ca_dataset_a3, ssd_dataset_a3)	0	0.0625	There is no evidence of a significant difference between paired groups.
4. (ca_dataset_a4, ssd_dataset_a4)	3	0.3125	There is no evidence of a significant difference between paired groups.
5. (ca_dataset_a5, ssd_dataset_a5)	6	0.8125	There is no evidence of a significant difference between paired groups.

B. EXPLORATION-EXPLOITATION TRADEOFF

Fig. 4 shows an exploration-exploitation graph, which is crucial for one to understand at which stage the model searches for new solutions or uses the existing ones. This succinctly explains the model's internal dynamics and processing. In the initial epochs, both CA and SSD showed heightened exploration to establish new knowledge of the solution space. This exploration is crucial for understanding the landscape and identifying potential areas of interest within the given dataset. As training progresses, there is a discernible shift in behaviour. The culture algorithm transitioned smoothly from exploration to exploitation, where it started to leverage accumulated knowledge more efficiently and allocate less time and resources to exploration. On the contrary, SSD maintained a relatively higher level of exploration than CA.

The exploration-exploitation dynamics, illustrated in Fig. 4, can be discussed in the context of the multi-disciplinary framework on the exploration-exploitation dilemma and the optimal contraction theorem [38], [39]. The concept of the "*exploration-exploitation tradeoff*" describes the ideal equilibrium between the search for new solutions and the exploitation of existing ones. Achieving an optimal balance is a complex task due to the potential drawbacks associated with both excessive exploration and excessive exploitation. Excessive exploration can lead to inefficiency and slower convergence, while excessive exploitation may result in premature convergence to suboptimal solutions [38],[56].

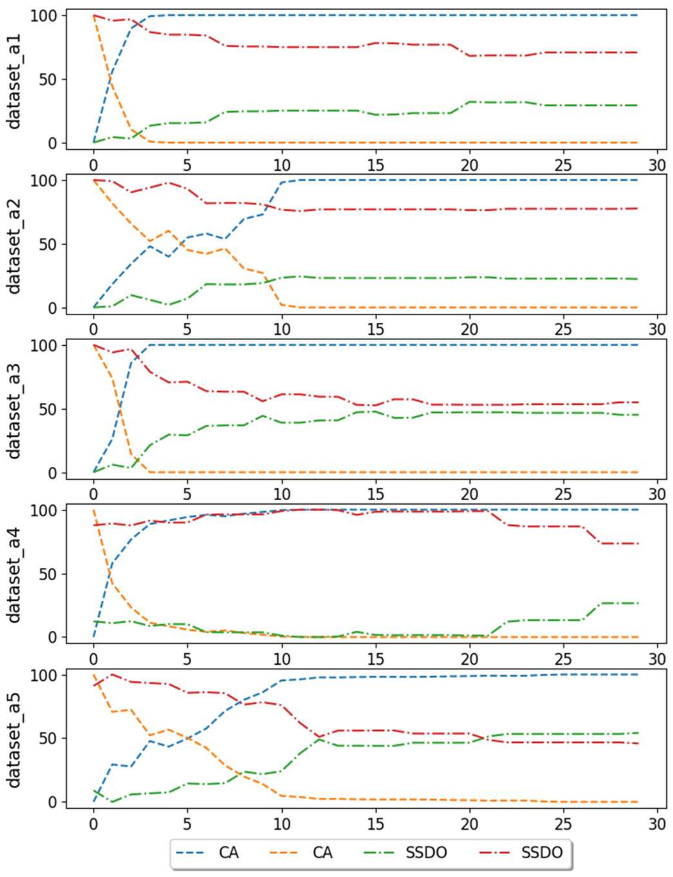

Figure 4. Exploration -Exploitation graph

Berger-Tal et al. [38] suggested a novel multidisciplinary framework to guide discussions around the exploration-exploitation trade-off in assessing the performance of the algorithms. A particular emphasis was placed on the expected performance of the two optimizers in the knowledge establishment phase, where exploration is expected to be at 100 and exploitation at 0. Both CA and SSD exhibited heightened exploration in the initial epochs, contributing to knowledge establishment as required. Based on the Berger-Tal et al framework, the SSD did not reach full exploitation in the 30 epochs provided compared to the CA. The SSD model seems trapped in the knowledge maintenance phase, which focuses on utilizing existing knowledge while maintaining constant exploration. On the other hand, CA seems to have gone through all the phases and increasingly relying on exploitation to generate new solutions. It reached the stage where exploitation is at 100 and exploration at 0, which is the ultimate goal in the exploration-exploitation dilemma. This suggests that the CA was able to effectively use the information from the best solutions found to guide the search space toward better solutions. The stagnation of SSD in the knowledge maintenance phase can be explained in the context of the optimal contraction theorem. The SSD's sustained exploration over the 30 epochs resonates with the complexity of the problem on the optimizer, especially in datasets with temporal unpredictability. The behaviour of SSD suggests that it employed a strategy of continuous learning to adapt to changing conditions in the dataset. Approaching the final epochs, CA appears to shift predominantly towards exploitation and focuses solely on exploiting the knowledge it has acquired, while the SSD continues to explore the search space.

C. POPULATION DIVERSITY

Fig. 5 shows the diversity graph which illustrates the degree of variation or spread among the solution candidates found by the two algorithms. The CA algorithm tends to converge to zero diversity, especially from the 13th epoch, indicating that, it either found a stable solution quickly or stuck in local optima. Unlike the CA, the SSD maintained reasonable diversity, decreasing at a very low rate throughout the 30 epochs, indicating its resilience to finding better solutions.

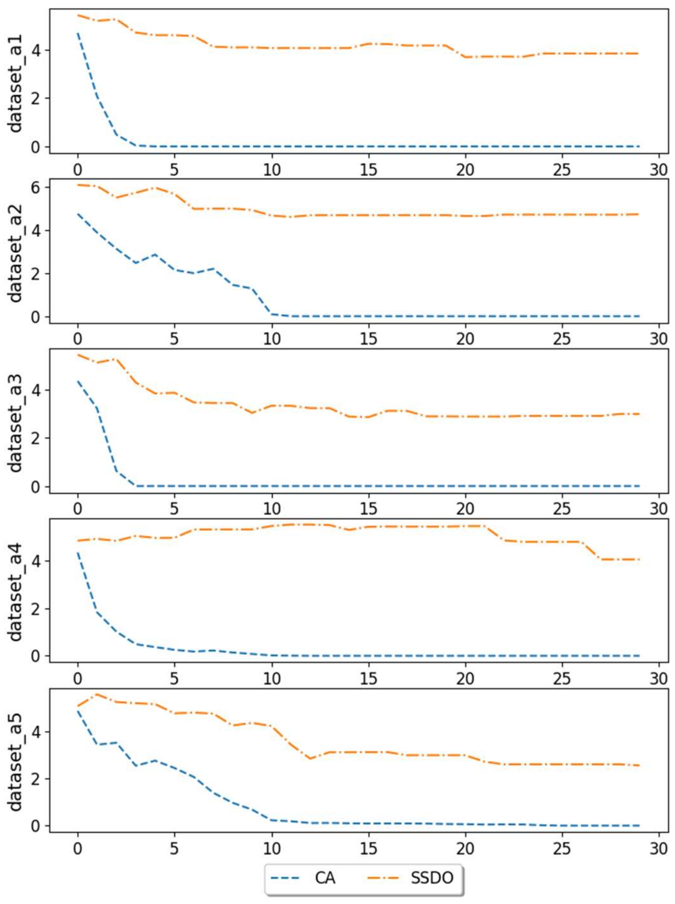

Figure 5. Diversity

The diversity metric of CA diminished early in all the folds compared to SSD which consistently tried to maintain the diversity over the 30 epochs. Although diversity is acknowledged as important in evolutionary algorithms for managing exploration-exploitation tradeoffs, the concept of productive diversity target exists, which is sufficient to produce optimal fitness [57]. The latter suggests that higher and prolonged diversity does not always convert to optimal solutions, but could also suggest that the problem is hard for the optimizer [39]. The diversity of CA diminished early in all folds as it did not see any need to extend its investment in exploration. This behaviour was necessary to balance exploration and exploitation at the culmination of its lifespan in the knowledge exploitation phase [38].

D. EPOCH TIME

Fig. 6 shows the epoch time graph. The SSD model demonstrated higher efficiency in epoch processing time. The model took approximately 609 seconds (average= 122/fold) to complete 5 folds of data with 30 epochs each, compared to CA's which took 1421 (average=284/fold) seconds. The CA showed stable processing time in all epochs, while the SSD varied significantly.

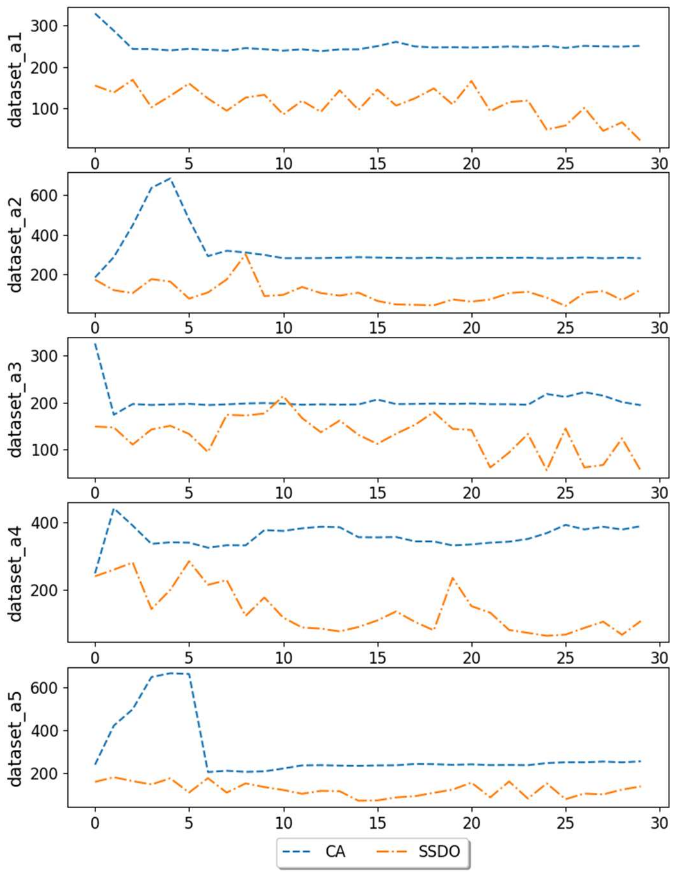

Figure 6. Epoch Time

The fluctuations in epoch processing time suggest variations in learning complexity. The SSD's efficient time utilization, high variability in epoch processing time and consistent diversity as discussed earlier, suggest flexibility in search strategy while the CA behaviour suggests a consistent approach to problem solving. The SSD demonstrated efficiency in epoch time compared to CA (Fig. 6). This trait is more common in metaheuristics based on swarm intelligence (*inclusive of SSD*), as it allows them to adapt to dynamic environments and avoid premature convergence [58].This stability and efficiency in epoch time position SSD as a potential candidate for real-time scenarios where quick adaptation to changing solution spaces is paramount. On the other hand, CA's consistent convergence fitness aligns with stability-focused algorithms.[58]

E.GLOBAL FITNESS

Fig. 7 shows the global fitness graph. The analysis of the global fitness for CA and SSD across 30 epochs in 5 folds reveals distinct patterns. The performance of the CA and SSD varied across the five folds. SSD demonstrated slight advantages over CA in folds *dataset_a1, dataset_a2, and dataset_a3*, while CA outperformed SSD in folds *dataset_a4* and *dataset_a5*.

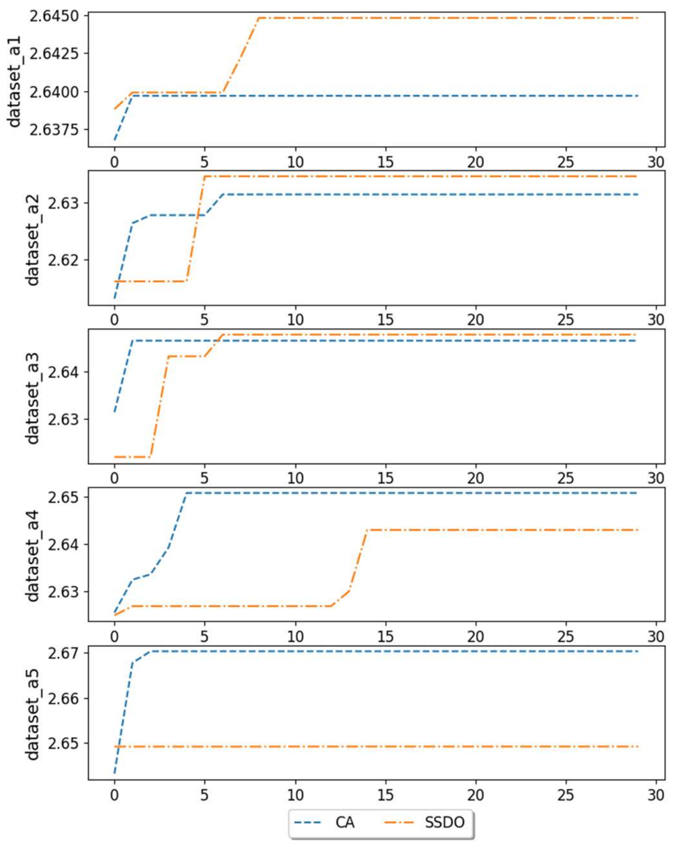

Figure 7. Global Fitness

It is worth noting that the two models achieved comparable global fitness levels. The suitability of each algorithm depends on the optimization problem's characteristics. The CA algorithm potentially fits tasks with consistent solution landscapes, while SSD's adaptability may be advantageous in more varied or changing environments.

The Wilcoxon Signed-Rank test was also conducted to assess whether there is any evidence of statistically significant differences between the pairs of optimizers (CA and SSD) and their associated hyperparameter sets. Table 8 shows the results of the Wilcoxon Signed-Rank test. The results suggest no significant difference between the parameter sets generated by CA and SSD across all folds (Table 8). The lack of statistically significant performance difference between CA and SSD makes it more arduous to attribute performance solely to specific hyperparameter configurations. It is therefore important to use task-specific characteristics to inform algorithm selection[39]. Factors such as computational efficiency, robustness, ease of implementation, or specific application requirements may determine the decision [59].

Table 7 shows the analysis of *f1 fitness weights, the f1 weight mean, f1 weight standard deviations and related rankings. Parameter set "ca_dataset_a3" has the highest mean of f1 weight score and low variance across the five folds*. These positive attributes suggest a commendable degree of generalization, indicating that the models' (MLP) performance will remain robust and effective across diverse datasets.

VI. CONCLUSION

In conclusion, the findings of this paper enrich our understanding of the strengths and limitations of CA and SSD. The theoretical lens provided by Berger-Tal et al.'s framework enhances our grasp of how exploration and exploitation strategies contribute to the convergence behaviour of these optimization algorithms. The results of the study showed statistically insignificant performance differences to the parameters produced by CA and SSD models. This highlights the effectiveness of the SSD algorithm in optimizing neural networks. The behaviour of SSD concerning its efficiency in epoch time, maintaining diversity and adaptability position it as a potential candidate for real-time scenarios where quick adaptation to changing solution spaces is paramount.

The hyper-parameter set *"ca_dataset_a3"* produced by culture algorithm (CA) yielded the highest mean of f1 weight score and low variance across the five folds and was therefore selected as optimal. These positive attributes suggest a commendable degree of generalization, indicating that the models' (MLP) performance will remain robust and effective across diverse datasets. Researchers and practitioners need to consider these findings when considering the application of machine learning and predictive models to solve learner migration problems.

VII. LIMITATION OF THE STUDY AND FUTURE WORK

The SSD model did not complete the exploration-exploitation tradeoff dynamics suggested by Berger-Tal et al in the 30 epochs provided. Although it is acknowledged that the SSD model performed comparatively well, with a fitness score similar to the CA, which was fully aligned with the model convergence characteristics and principles outlined in the Berger-Tal framework on the exploration-exploitation tradeoff.

While the results of SSD are promising, they may not fully represent the model's capabilities when subjected to a more intensive analysis. Future research directions could include analyzing the SSD's behavior against Berger-Tal et al.'s exploration-exploitation framework in the learner migration context and investigating the transferability of learned knowledge across diverse domains. Additionally, future studies might explore the effects of varying the number of epochs, adjusting parameter settings, and experimenting with different model initialization strategies to gain deeper insights into the SSD's convergence dynamics.

VII. SOFTWARE

The experiments in the study were conducted using R Studio (R editor) and Wing (python editor). R Studio was used for feature selection, while Wing IDE with mealpy and scikit libraries were used for MLP hyperparameter optimization.

VIII. DATA AVAILABILITY STATEMENT

This article is accompanied by the data used in the study, the reports generated during model processing, data analysis reports, and source codes to ensure replicability

APPENDIX A1: META-DATA

ID	Type	Category	List Values
f01	numeric	derived	continues variable
f02	numeric	derived	continues variable
f03	nominal	learner biographic	0=not specified; 1=yes; 2=no
f04	nominal	learner biographic	0=no specified; 1=single parent; 2=both parents; 3=
f05	nominal	learner biographic	1=male; 2=female
f06	nominal	learner biographic	10=grade 10; 11=grade 11; 12=grade 12
f07	numeric	biographic	
f08	nominal	school	10=grade 10; 11=grade 11; 12=grade 12
f09	numeric	biographic	continues variable
f10	nominal	learner biographic	0=not specified; 1=afrikaans; 2=english; 3=isindebele; 4=siswati; 5=isixhosa; 6=isizulu; 7=sesotho; 8=sepedi; 9=setswana; 10=tshivenda; 11=xitsonga; 12=sign; 13=other
f11	nominal	learner biographic	0=not specified; 1=afrikaans; 2=english; 3=isindebele; 4=siswati; 5=isixhosa; 6=isizulu; 7=sesotho; 8=sepedi; 9=setswana; 10=tshivenda; 11=xitsonga; 12=sign; 13=other
f12	nominal	biographic	0=not specified; 1=no; 2=yes; 3=unknown
f13	numeric	biographic	
f14	nominal	biographic	0=not specified; 1=afrikaans; 2=english; 3=isindebele; 4=siswati; 5=isixhosa; 6=isizulu; 7=sesotho; 8=sepedi; 9=setswana; 10=tshivenda; 11=xitsonga; 12=sign; 13=other
f15	nominal	biographic	1=true; 2=false
f16	nominal	biographic	0=not applicable; 1=yes; 2=no
f17	nominal	biographic	1=african/black; 2=asian/indian; 3=coloured; 5=other; 4=white
f18	nominal	learner biographic	0=not specified; 1=by foot 2km less; 10=hostel; 11=train; 12=by foot 5km to 10km; 13=private bus transport; 2=by foot 2km to 5km; 3=by foot 10km +; 4=bicycle; 5=motor cycle; 6=motor car; 7=taxi; 8=employer bus; 9=government bus (transport scheme)
f19	nominal	school demographic	1=capricorn north; 2=capricorn south; 3=mogalakwena; 4=mopani east; 5=mopani west; 6=sekhukhune east; 7=sekhukhune south; 8=vhembe east; 9=vhembe west; 10=waterberg
f20	nominal	school	1=ordinary school; 2=special needs education centre
f21	nominal	school	1=public; 2=independent
f22	nominal	school	0=not specified; 1=poverty index 1; 2=poverty index 2; 3=poverty index 3; 4=poverty index 4; 5=poverty index 5
f23	numeric	school	1=0 - 29; 2=30 - 39; 3=40 - 49; 4=50 - 59; 5=60 - 69; 6=70 - 79; 7=80 - 100
f24	numeric	biographic	continues variable
c25	nominal	biographic	1=displacement; 2=no displacement
c26	nominal	learner biographic	1=graduated; 2=dropped out; 3=transfer to another school; 4=no reason; 5=no movement
c27	nominal	learner biographic	0=0km; 1=between 0km and 1km; 2=between 1km and 2km; 3=between 2km and 3km; 4=between 3km and 4km; 5=between 4km and 5km; 6=above 5km

References

[1] A. Mountz and S. Mohan, "Human migration in a new era of mobility: intersectional and transnational approaches," *Global Social Challenges Journal*, vol. 1, no. 1, pp. 59–75, 2022, https://doi.org/10.1332/RFXW5601.

[2] A. Algarni, "Data mining in Education," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 7, no. 6, pp. 58–77, 2016, https://doi.org/10.4018/978-1-5225-1877-8.ch005.

[3] D. Hrehová and K. Teplická, "The informational communication technology is a tool of global education," in Globalization and its Socio-Economic Consequences, Slovakia: Sciences, EDP, 2020, pp. 1–6. https://doi.org/10.1051/shsconf/20207406008.

[4] A. Maheri, S. Jalili, Y. Hosseinzadeh, R. Khani, and M. Miryahyavi, "A comprehensive survey on cultural algorithms," Swarm Evol Comput, vol. 62, pp. 4–63, 2021, https://doi.org/10.1016/j.swevo.2021.100846.

[5] Y. Abdi and Y. Seyfari, "Search manager: A framework for hybridizing different search strategies," International Journal of Advanced Computer Science and Applications, vol. 9, no. 5, pp. 525–540, 2018, https://doi.org/10.14569/IJACSA.2018.090568.

[6] P. Pramanik, S. Mukhopadhyay, S. Mirjalili, and R. Sarkar, "Deep feature selection using local search embedded social ski-driver optimization algorithm for breast cancer detection in mammograms," Neural Comput Appl, vol. 35, no. 7, pp. 5479–5499, Mar. 2023, https://doi.org/10.1007/s00521-022-07895-x.

[7] Z. Shao, H. Sun, X. Wang, and Z. Sun, "An optimized mining algorithm for analyzing students' learning degree based on dynamic data," IEEE Access, vol. 8, pp. 1–16, 2020, https://doi.org/10.1109/ACCESS.2020.3001749.

[8] A. Tharwat and T. Gabel, "Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm," Neural Comput Appl, vol. 32, no. 11, pp. 6925–6938, Jun. 2020, https://doi.org/10.1007/s00521-019-04159-z.

[9] B. Chatterjee, T. Bhattacharyya, K. K. Ghosh, P. K. Singh, Z. W. Geem, and R. Sarkar, "Late Acceptance Hill Climbing Based Social Ski Driver Algorithm for Feature Selection," *IEEE Access*, vol. 8, pp. 75393–75408, 2020, https://doi.org/10.1109/ACCESS.2020.2988157.

[10] S. Ahmad, M. A. El-Affendi, M. S. Anwar, and R. Iqbal, "Potential Future Directions in Optimization of Students' Performance Prediction System," *Comput Intell Neurosci*, vol. 2022, 2022, https://doi.org/10.1155/2022/6864955.

[11] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, https://doi.org/10.1016/j.neucom.2020.07.061.

[12] Y. Shi, H. Qi, X. Qi, and X. Mu, "An efficient hyper-parameter optimization method for supervised learning," *Appl Soft Comput*, vol. 126, p. 109266, Sep. 2022, https://doi.org/10.1016/j.asoc.2022.109266.

[13] G. Moore, C. Bergeron, and K. P. Bennett, "Model selection for primal SVM," *Mach Learn*, vol. 85, no. 1–2, pp. 175–208, Oct. 2011, https://doi.org/10.1007/s10994-011-5246-7.

[14] Lei Xu, "Ying-Yang Learning [from:The Handbook f Brain Theory and Neural Networks]," 2002.

[15] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based Hyperparameter Optimization through Reversible Learning," *Journal of Machine Learning Research*, vol. 37, pp. 2113–2122, 2015.

[16] Y. Bao, Z. Hu, and T. Xiong, "A PSO and pattern search based memetic algorithm for SVMs parameters optimization," *Neurocomputing*, vol. 117, pp. 98–106, Oct. 2013, https://doi.org/10.1016/j.neucom.2013.01.027.

[17] G. A. Galanti, "An introduction to cultural differences," *Western Journal of Medicine*, vol. 172, no. 5, pp. 335–336, 2000, https://doi.org/10.1136/ewjm.172.5.335.

[18] S. Jalili and Y. Hosseinzadeh, "A cultural algorithm for optimal design of truss structures," *Latin American Journal of Solids and Structures*, vol. 12, no. 9, pp. 1721–1747, 2015, https://doi.org/10.1590/1679-78251547.

[19] R. G. Reynolds, "An introduction to cultural algorithms," in *An Introduction to Cultural Algorithms*, ResearchGate, 2014, p. 10. Accessed: Sep. 04, 2023. [Online]. Available at: https://www.researchgate.net/publication/201976967

[20] Y. Khorrami, D. Fathi, and R. Rumpf, "Fast optimal design of optical components using the cultural algorithm," *Opt Express*, vol. 28, no. 11, p. 15954, 2020, https://doi.org/10.1364/OE.391354.

[21] Y. Xuesong, L. Wei, C. Wei, L. Wenjing, C. Zhang, and L. Hanmin, "Cultural algorithm for engineering design problems," *Int J Comp Sci*, vol. 9, no. 6, pp. 53–61, 2012.

[22] S. Upadhyayula, "Dominance in multi-population cultural algorithm," Thesis, University of Windsor, 2015. https://doi.org/10.1109/ICMLA.2015.102.

[23] A. Tharwat, A. Darwish, and A. E. Hassanien, "Rough sets and social ski-driver optimization for drug toxicity analysis," *Comput Methods Programs Biomed*, vol. 197, p. 105702, Dec. 2020, https://doi.org/10.1016/j.cmpb.2020.105702.

[24] H. Su-Hyun, K. Ko Woon, K. SangYun, and Y. Young Chul, "Artificial Neural Network: Understanding the Basic Concepts without Mathematics," *Dement Neurocogn Disord*, vol. 17, no. 3, p. 83, 2018, https://doi.org/10.12779/dnd.2018.17.3.83.

[25] K. Shiruru, "An introduction to artificial neural network," *International Journal of Advance Research And Innovative Ideas In Education*, vol. 1, no. 5, pp. 27–30, 2016, [Online]. Available at: https://www.researchgate.net/publication/319903816

[26] R. Keim, "How to Create a Multilayer Perceptron Neural Network in Python," Technical Article. Accessed: Aug. 04, 2024. [Online]. Available at: https://www.allaboutcircuits.com/technical-articles/

[27] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," *Foreign Affairs*, vol. 91, no. 5, pp. 1689–1699, 2016.

[28] L. Vanneschi and M. Castelli, *Multilayer perceptrons*, vol. 1–3. 2018. https://doi.org/10.1016/B978-0-12-809633-8.20339-7.

[29] V. E. Balas, N. E. Mastorakis, M.-C. Popescu, and V. E. Balas, "Multilayer perceptron and neural networks HISTORICAL PHOTOGRAPHS View project BioCell-NanoART = Novel Bio-inspired Cellular Nano-Architectures-For Digital Integrated Circuits View project Multilayer Perceptron and Neural Networks," 2009. [Online]. Available at: https://www.researchgate.net/publication/228340819

[30] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, https://doi.org/10.1038/nature14539.

[31] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, https://doi.org/10.1038/323533a0.

[32] H. Wang, R. Czerminski, and A. C. Jamieson, "Neural Networks and Deep Learning," *The Machine Age of Customer Insight*, pp. 91–101, 2021, https://doi.org/10.1108/978-1-83909-694-520211010.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical Learning*, vol. 26, no. 4. 1967.

[34] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," *ETH Zurich Research Collection*, pp. 95–100, 2001, doi: 10.1.1.28.7571.

[35] Wikipedia Contributors., "Multi Objective Optimization," *Wikipedia*. Wikepidia, 2023.

[36] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. 2007. https://doi.org/10.1007/978-0-387-36797-2.

[37] J. Reddy and N. Kumar, "Multi-Objective optimization using evolutionary algorithms," *Water Resources Management*, vol. 20, no. 6, pp. 861–878, 2006. https://doi.org/10.1007/s11269-005-9011-1.

[38] O. Berger-Tal, J. Nathan, E. Meron, and D. Saltz, "The exploration-exploitation dilemma: A multidisciplinary framework," *PLoS One*, vol. 9, no. 4, p. 95693, 2014, https://doi.org/10.1371/journal.pone.0095693.t001.

[39] Jie Chen, Bin Xin, Zhihong Peng, Lihua Dou, and Juan Zhang, "Optimal Contraction Theorem for Exploration–Exploitation Tradeoff in Search and Optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 3, pp. 680–691, May 2009, https://doi.org/10.1109/TSMCA.2009.2012436.

[40] S. Akanmu and S. Jaja, "Knowledge Discovery in Database: A knowledge management strategic approach," Oct. 2012.

[41] F. Ramphele, Z. Wang, and A. Yusuff, "Determination of the Best Feature Subset for Learner Migration in Limpopo," *International Journal of Computing*, vol. 23, no. 2, pp. 165–176, Jul. 2024, https://doi.org/10.47839/ijc.23.2.3534.

[42] D. B. Grigg, "E . G . Ravenstein and the ' laws of migration ,'" *J Hist Geogr*, vol. 3, no. 1, pp. 41–54, 1977. https://doi.org/10.1016/0305-7488(77)90143-8.

[43] E. S. Lee, "A Theory of Migration," *Demography*, vol. 3, no. 1, pp. 47–57, 1996, Accessed: Sep. 04, 2023. https://doi.org/10.2307/2060063

[44] E. G. Raveinstein, "The Lawas of Migration," *Journal of Statistical Society of London*, vol. 48, no. 2, pp. 167–235, 1885. https://doi.org/10.2307/2979181.

[45] R. J. Botha and T. G. Neluvhola, "An investigation into factors that contribute to learner migration in South African schools," *The Journal of Social Sciences Research*, vol. 6, no. 63, pp. 224–235, 2020, https://doi.org/10.32861/jssr.63.224.235.

[46] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining," *Applied Artificial Intelligence*, vol. 17, no. 5–6, pp. 375–381, May 2003, https://doi.org/10.1080/713827180.

[47] A. Kochański, "Data preparation," *Computer Methods In Materials Science*, vol. 10, no. 1, 2010, Accessed: Sep. 04, 2023. [Online]. Available at: https://www.researchgate.net/publication/299350639

[48] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[49] I. C. Simelani, "Learner migration and its Impact on rural schools : A case study of two rural schools in Kwazulu- Natal (Publication No.12637)," Masters Thesis, University of Kwazulu-Natal, Durban, 2016. Accessed: Aug. 25, 2023. [Online]. Available at: https://researchspace.ukzn.ac.za/handle/10413/12637

[50] H. Van der Merwe, "Migration patterns in rural schools in South Africa: Moving away from poor quality education," *Education as Change*, vol. 15, no. 1, pp. 107–120, 2011, https://doi.org/10.1080/16823206.2011.576652.

[51] S. Gregor, "A Theory of Theories in Information Systems," *Information Systems Foundations*, pp. 1–18, 2002. https://doi.org/10.3127/ajis.v10i1.439.

[52] H. Patel, D. S. Rajput, G. T. Reddy, C. Iwendi, K. A. Bashir, and O. Jo, "A review on classification of imbalanced data for wireless sensor networks," *Int J Distrib Sens Netw*, vol. 16, no. 4, pp. 1–15, 2020, https://doi.org/10.1177/1550147720916404.

[53] P. Thereza P. P., G. Lumacad, and R. Catrambone, "Predicting Student Performance Using Feature Selection Algorithms for Deep Learning Models," in *2021 XVI Latin American Conference on Learning Technologies (LACLO)*, IEEE, Oct. 2021, pp. 1–7. https://doi.org/10.1109/LACLO54177.2021.00009.

[54] F. Afghah, A. Razi, R. Soroushmehr, H. Ghanbari, and K. Najarian, "Game theoretic approach for systematic feature selection: Application in false alarm detection in intensive care units," *Entropy*, vol. 20, pp. 1–16, 2018, https://doi.org/10.3390/e20030190.

[55] X.-S. Yang, "Metaheuristic Optimization: Algorithm Analysis and Open Problems," Dec. 2012. https://doi.org/10.1007/978-3-642-20662-7_2.

[56] V. Osuna-Enciso, E. Cuevas, and B. Morales Castañeda, "A diversity metric for population-based metaheuristic algorithms," *Inf Sci (N Y)*, vol. 586, pp. 192–208, Mar. 2022, https://doi.org/10.1016/j.ins.2021.11.073.

[57] T. Gabor, T. Phan, and C. Linnhoff-Popien, "Productive fitness in diversity-aware evolutionary algorithms," *Nat Comput*, vol. 20, no. 3, pp. 363–376, Sep. 2021, https://doi.org/10.1007/s11047-021-09853-3.

[58] A. E. Ezugwu *et al.*, "Metaheuristics: a comprehensive overview and classification along with bibliometric analysis," *Artif Intell Rev*, vol. 54, no. 6, pp. 4237–4316, Aug. 2021, https://doi.org/10.1007/s10462-020-09952-0.

[59] Z. Raziei, R. Tavakkoli-Moghaddam, and S. Tabrizian, "Performance Analysis of Meta-heuristic Algorithms for a Quadratic Assignment Problem," Jul. 2020.

***FRANS RAMPHELE** (MPHIL, University of Cape Town) is currently a Director in the Limpopo Department responsible for the Education Management Information System (EMIS). He has considerable knowledge and experience in the areas of ICT/IS management, software development, business intelligence, artificial intelligence and machine learning, data mining and database management Among other things, he is a member of the South African Higher Education Committee (HEDCOM) for e-Education, which oversees the development of ICT guidelines for e-Education.*

***ZENGHUI WANG** (PhD, Nankai University, China) is a National Research Foundation (NRF) C2-rated researcher in Electrical Engineering. He is currently a Professor at the University of South Africa (Unisa) in the Department of Electrical and Mining Engineering. Prof Wang specialises in Electrical Engineering and Computer Science. He is the leader of Intelligent System research group/laboratory. He has been involved in more than ten major research projects in South Africa and China. Up until February 2022, he has approximately 180 papers published or accepted, including 90 ISI master indexed journal papers.*

***ADEDAYO YUSUFF** (D.Tech.) is a lecturer and a researcher in Electrical Engineering. He is currently a Professor at the University of South Africa (Unisa) in the Department of Electrical and Mining Engineering. Adedayo Yusuff has extensive experience in the industry in the areas of signal processing and control, network optimisation, and machine translation. Prof Yusuff specialises in adaptive electric power transmission networks; application of computational intelligence schemes for operation, control, and protection of advanced power grid; and integration of intermittent and renewable energy sources to power grid. He has been involved in research projects in South Africa and Nigeria.*