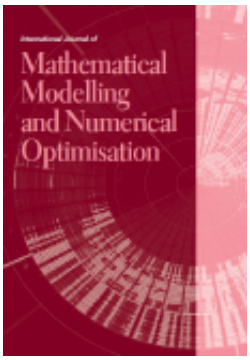


International Journal of Mathematical Modelling and Numerical Optimisation

ISSN online: 2040-3615 - ISSN print: 2040-3607
https://www.inderscience.com/ijmmno

Bio-inspired mix design optimisation of self-compacting concrete using machine learning algorithms

Sriman Pankaj Boindala, Vasan Arunachalam, Jabez J. Christopher

DOI: 10.1504/IJMMNO.2024.10058653

Article History:

Received:	07 December 2021
Last revised:	13 February 2023
Accepted:	14 April 2023
Published online:	11 December 2024

Bio-inspired mix design optimisation of self-compacting concrete using machine learning algorithms

Sriman Pankaj Boindala

Department of Civil and Environmental Engineering,
Technion – Israel Institute of Technology,
Haifa 32000, Israel
Email: srimanpankaj@gmail.com

Vasan Arunachalam*

Department of Civil Engineering,
Birla Institute of Technology and Science-Pilani,
Hyderabad Campus, Telangana, 500078, India
Email: vasan@hyderabad.bits-pilani.ac.in
*Corresponding author

Jabez J. Christopher

Department of Computer Science and Information Systems,
Birla Institute of Technology and Science-Pilani,
Hyderabad Campus, Telangana, 500078, India
Email: jabezc@hyderabad.bits-pilani.ac.in

Abstract: This study focuses on optimising concrete mix design using a hybrid bio-inspired optimisation algorithm that combines differential evolution (DE) and cuckoo search (CS). The study also evaluates the performance of two strength prediction models, artificial neural networks (ANNs) and support vector machine regression (SVR), in determining optimal mix proportions. The hybrid algorithm is tested using 11 benchmark test functions and the best approach is chosen to solve a mix design optimisation problem with the objectives of maximising compressive strength, minimising carbon emissions, and minimising cost. Results show that ANN outperforms SVR in terms of compressive strength, with a 30% increase observed. Both prediction models produce optimal mix proportions with minimal variation for cost and embodied carbon minimisation scenarios. The study demonstrates the efficacy of the hybrid optimisation algorithm in conjunction with a prediction model in determining optimal concrete mix proportions.

Keywords: bio-inspired optimisation; swarm intelligence algorithms; machine learning; compressive strength prediction model; concrete mix design optimisation; cuckoo search; differential evolution; differential cuckoo search; DCS; artificial neural networks; ANNs; support vector machine regression; SVR.

Reference to this paper should be made as follows: Boindala, S.P., Arunachalam, V. and Christopher, J.J. (2024) 'Bio-inspired mix design optimisation of self-compacting concrete using machine learning algorithms', *Int. J. Mathematical Modelling and Numerical Optimisation*, Vol. 14, Nos. 1/2, pp.1–31.

Biographical notes: Sriman Pankaj Boindala is a graduate researcher working towards his PhD in Civil and Environmental Engineering Faculty at Technion-Israel institute of Technology, Israel.

Vasan Arunachalam is currently a Professor in the Department of Civil Engineering at BITS Pilani, Hyderabad Campus. He holds a PhD in Water Resources Engineering and did his post-doctoral studies at Western University, Canada. His research interests include optimisation of reservoir operation using nature inspired algorithms; water distribution network design optimisation; leak detection in water distribution networks using machine learning and IoTs. He is a recipient of awards for various research papers and has also received sponsored research funding from various agencies. He has published more than 85 research publications and has been a reviewer for various reputed international journals.

Jabez J. Christopher received his PhD from Anna University, Chennai in 2017. He currently works as an Assistant Professor at Department of Computer Science, BITS Pilani, Hyderabad Campus. His research interests include data mining, optimisation of learning algorithms, fuzzy systems, and AI for health informatics. Contributions of his research mainly focus on assisting medical centres with strategic knowledge management tasks and clinical decision-making. It involves application of computer methods and programs for healthcare.

1 Introduction

The process of determining the optimal proportions of concrete constituents to achieve the desired strength is known as concrete mix design. There are various empirical methods and standard ratios for determining mix proportions for normal concrete according to different codes. With advancements in the construction industry, there is an increased demand for properties such as high strength, high workability, and sustainability, leading to the development of new types of concrete, such as self compacting concrete (SCC), high-performance concrete (HPC), and ultra high-performance concrete (UHPC). These concretes incorporate mineral and chemical admixtures that not only reduce cost and improve eco-friendliness, but also enhance certain mechanical properties.

One of the challenges in mix design for SCC is the presence of multiple ingredients, each of which can significantly impact the concrete's compressive strength. There are currently no Indian standards for preparing SCC, so this study proposes a machine learning-based optimisation approach to determine the optimal mix design for three different scenarios. The first scenario focuses on maximising compressive strength, the second on minimising cost while maintaining compressive strength and durability, and the third on obtaining an eco-friendly mix design while maintaining compressive strength.

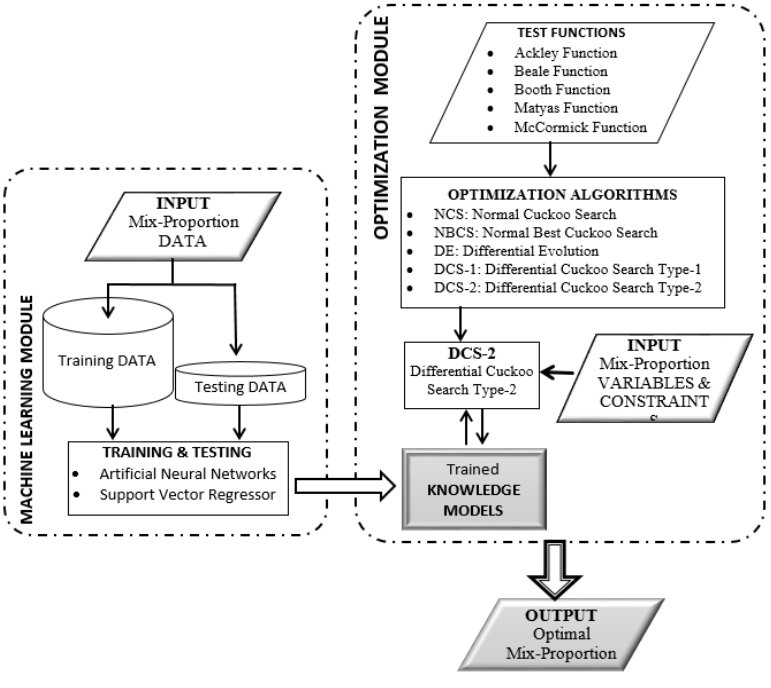
Accurate prediction of compressive strength is critical to the success of this approach. In the past, researchers have used various machine learning algorithms to predict the compressive strengths of concrete, including support vector machines (SVMs) (Gupta, 2007), multiple nonlinear regression models (Zain and Abd, 2009), tree-based machine learning models (Deepa et al., 2010), neural networks (Huang et al., 2011), and comparison of different machine learning techniques (Dutta et al., 2018; Chopra et al., 2018). Yeh (1999) optimised concrete mix proportions using a combination of neural networks and linear programming. Cheng et al. (2014) employed a genetic algorithm (GA)-based evolutionary SVM for HPC mixture optimisation. Lee and Yoon (2009) used a modified harmony search algorithm and neural networks for mix proportion design, while Reddy et al. (1993) proposed an expert system for optimum concrete design. Pham et al. (2016) predicted compressive strength using metaheuristic-optimised least squares support vector regression (SVR), and Chou et al. (2011) compared data-mining techniques to optimise compressive strength prediction accuracy. Naseri et al. (2020) developed a method to generate sustainable mix proportions for concrete strengths of 20–70 MPa. Azimi-Pour et al. (2020) created linear and nonlinear SVM models to predict the properties of high-volume fly ash concrete. Neural networks and SVMs have been successfully used in the recent past as a surrogate compressive strength prediction models (Amin et al., 2022; Ghafor, 2022). The current study aims to utilise neural networks and SVMs as a surrogate compressive strength prediction model to generate a knowledge-based compressive strength prediction model. The proposed model will be combined with a hybrid meta-heuristic optimisation algorithm to attain optimal mix proportions.

A hybrid optimisation algorithm is a method that integrates two or more optimisation algorithms to mitigate the limitations of a single optimisation algorithm. Differential evolution (DE), introduced by Storn and Price in 1997, is a population-based optimisation algorithm that is both simple and effective in addressing a wide range of optimisation problems. The algorithm operates by maintaining a population of candidate solutions and continually updating the population through the generation of new candidate solutions, which are based on the differences between existing solutions. The algorithm is noted for its strong performance in local optima searches and robustness (Mallipeddi et al., 2011). This algorithm has been applied in the past to solve complicated problems in civil engineering (Vasan and Simonovic, 2010; Hargrave and Chen, 2013; Al-Saffar and Al-Mahdi, 2015b). Moreover, recent extensions of DE are found to be efficient with several optimisation test suites (Nadimi-Shahraki and Zamani, 2022; Wang et al., 2022).

Cuckoo search (CS), introduced by Yang and Deb in 2009, is a metaheuristic optimisation algorithm inspired by the behaviour of cuckoos in nature. The optimisation process in CS simulates the behaviour of cuckoos searching for a suitable nest, with candidate solutions updated through the generation of new solutions based on the current best solution and random walks. CS has demonstrated strong ability in global search and has a small number of parameters (Yang et al., 2018). However, it may be prone to trapping into local optima and reducing population density, affecting its robustness (Yang et al., 2018). This algorithm has also been applied in the past as an optimisation tool to solve complex civil engineering problems (Fu and Wang, 2013; Al-Saffar and Al-Mahdi, 2015a; Boindala and Arunachalam, 2020). Moreover, recent extensions of CS are found to be efficient with several optimisation test suites (Peng et al., 2021; Cheng and Xiong, 2022).

The current study proposes a novel approach that employs a hybrid optimisation technique, combining CS and DE algorithms, in a comprehensive framework for designing optimal concrete mixes. The effectiveness of the new hybrid algorithms is evaluated against their parent algorithms (DE and CS) using a set of 11 mathematical test functions. The most robust hybrid optimisation algorithm is then selected to predict the optimal mix proportions through a knowledge-based machine learning model. The data for this research work to train the ML models is a compilation of past published research works and databases, and all the features (independent variables) are of the continuous (numeric) data type. The proposed work compares two different prediction models, namely SVM, artificial neural network (ANN). These models are combined with the hybrid optimisation algorithm to obtain the optimal mix proportions for different scenarios. These scenarios include finding the mix proportion that provides the highest compressive strength, the most economical mix proportion for a desired compressive strength, and the mix proportion with the lowest embodied carbon for a specified compressive strength. The proposed hybrid optimisation approach and the framework it is a part of represents a novel approach in the field of concrete mix design. The proposed framework can be explained into two modules where the first module is a machine learning module, and the other is an optimisation module which is schematically presented in Figure 1.

Figure 1 The proposed framework for mix design optimisation



1.1 Module-1: machine learning module

In this study, two machine learning models, ANN and SVM, are trained and evaluated using existing mix proportions and corresponding compressive strength data collected

from prior research and databases. The obtained knowledge models are then utilised in the subsequent optimisation module.

1.2 Module-2: optimisation module

This module involves two phases: algorithm selection and ML optimisation.

1.2.1 Stage-1: selecting the robust bio-inspired optimisation algorithm

Two hybrid bio-inspired algorithms, differential cuckoo search (DCS) 1 and 2, are compared with CS, DE, and modified CS algorithms. The comparison is based on 11 benchmark mathematical test functions, and the best algorithm is selected based on the performance metrics. The shapes and landscapes of the test functions and a description of each function are presented in Figure 3 and Table 6 respectively. The population size, number of function evaluations, and algorithm-specific parameters for each algorithm are kept constant to ensure a robust comparison.

1.2.2 Stage-2: machine learning based optimisation to obtain optimal mix proportions

In this stage, the robust algorithm obtained from Stage 1 will be used to find the optimal mix proportion for three different scenarios, with the help of the knowledge models from the machine learning module.

Scenario 1 *Maximising compressive strength*

The robust bio-inspired optimisation algorithm will be used to find the quantities of each component of the SCC mix within its specified boundaries, subject to the constraints of the EFNARC (2002) guidelines. In every iteration, the generated mix proportion will be fed into the knowledge model to obtain its predicted compressive strength; the process will continue until the algorithm converges to the maximum compressive strength.

Scenario 2 *Obtaining economical mix proportion with desired compressive strength*

The objective in this scenario is to find the mix proportion with the lowest cost and the desired compressive strength. The robust bio-inspired optimisation algorithm will search within its search space and satisfy the EFNARC constraints and the added constraint of compressive strength. The compressive strength in each iteration will be obtained from the knowledge models, as in Scenario 1. The search will terminate when the algorithm converges to a mix proportion with the lowest cost and compressive strength greater than the desired value.

Scenario 3 *Minimising embodied carbon*

The objective in this scenario is to find the eco-friendliest mix proportion with the least embodied carbon. The working mechanism is the same as in Scenario 2, but instead of targeting a mix proportion with the lowest cost, the algorithm will target a mix proportion with the least embodied carbon.

At the end of the process, an optimal mix proportion for any of the three scenarios can be obtained.

2 Machine learning module

2.1 Data set description

The strength and accuracy of any machine learning model depend on the database's comprehensiveness or learning data. The dataset for the current study consists of 708 records which were collected from several reputed research works. The data set contains mix proportions and their 28-day compressive strength values. The materials involved are fly ash, GGBS, cement, coarse aggregate, fine aggregate, superplasticiser and water. The statics of these materials in the database is shown in Table 1. As there are 7 independent variables (mix proportions) and 1 dependent variable, a feature selection technique is used to identify the most effective variable (feature) from the seven independent variables; Entropy-based information gain technique is used to assign weights to each independent variable. The weights obtained are listed in Table 2. Based on this feature selection method, we can see that the contribution of superplasticiser and coarse aggregate is negligible compared to other mix constituents. The complete data set is split into training and testing in a ratio of 70:30, and the two models are trained and tested with this data.

Table 1 Data statistics

Material (kg/m³)	*Max*	*Min*	*Mean*	*Variance*
Cement	540	102	261.693	11,238.970
Blast furnace slag	359.4	0	92.679	7,303.041
Fly ash	200.1	0	68.206	4,444.715
Coarse aggregate	1145	801	944.486	6,701.340
Water	247	121.75	182.790	374.588
Superplasticiser	32.2	0	7.859	26.878
Fine aggregate	992.6	594	763.278	5,398.075
Concrete compressive strength (MPa)	81.751	8.535	37.239	226.223

Table 2 Information gain (weights) of variables

Variable name	*Weight*
Cement (C)	0.3396
Blast furnace slag (G)	0.2358
Fly ash (F)	0.1157
Fine aggregate (FA)	0.1202
Coarse aggregate (CA)	0.0479
Water (W)	0.1757
Super plasticiser (SP)	0.0580

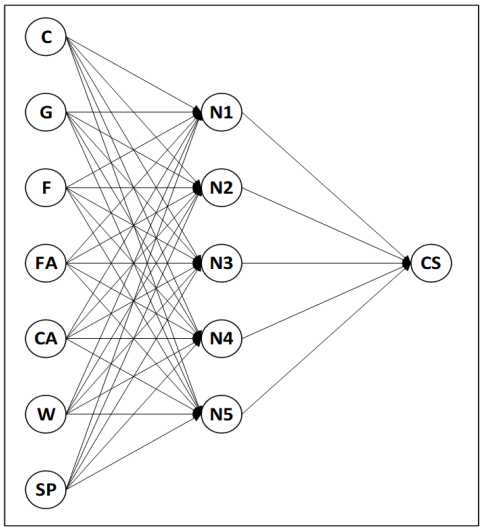
2.2 Artificial neural network

A neural network is an artificial system modelled after the human brain, which consists of an input layer, an output layer, and at least one hidden layer. The input layer receives signals from the environment, the hidden layer processes these signals through sequential nonlinear computations, and the output layer provides a response based on the processed information.

In this article, neural networks are employed as a machine learning tool to predict the compressive strength of self-compacting concrete (SCC) mixture. The relationship between the mix proportions and compressive strength can be complex and unknown; thus, neural networks are utilised to establish this relationship.

Many researchers in the past have used a similar approach to predict the compressive strength of various types of concrete (Yang, 2014). For the current problem, each component used in the mixture is taken as a separate neuron; therefore, the input layer comprises of 7 neurons and a single hidden layer is assumed and the number of hidden nodes is taken to be five after various trials. The network architecture is shown in Figure 2.

Figure 2 Neural network architecture



2.3 Support vector machine

The support vector (SV) algorithm is a statistical learning algorithm that was developed in the 1960s for the purpose of generalising well from training data to testing data. It was later adapted to develop the SVM, which has become a widely used tool for solving classification problems. The SVM algorithm is based on the concept of finding a hyperplane that maximises the margin between the data points in the input space. This approach is especially useful for linearly separable data, where the hyperplane acts as a boundary between different classes. However, in practice, the data is often not linearly separable, which is why the SVM algorithm includes the ability to use kernel functions to map the input data into a higher-dimensional space, where a linear boundary can be

found. In the mid-1990s, the SV learning approach was applied to regression problems, giving rise to the SVR model. Unlike the SVM, which is used for classification problems, the SVR model is used for making predictions based on continuous target values. The SVR algorithm minimises the deviation between the actual and predicted target values, subject to a specified tolerance level (epsilon). This deviation is measured as the mean square error between the predicted and closest points on the hyperplane. In the SVR model, the choice of kernel function is crucial for the accuracy of the predictions. The most commonly used kernel functions are linear, polynomial, and radial basis functions. In this study, we use the radial basis function for computing the kernel matrix, as it is well-suited for nonlinear data. The epsilon value is set to its default value of 0.1, which is a commonly used value in SVR applications

2.4 Model evaluation

2.4.1 Pearson correlation coefficient (R)

The Pearson correlation coefficient is a statistical measure used to calculate the strength between two sets of parameters. R ranges from –1 to 1. The efficiency of the model in predicting the output can be obtained through the analysis of R values. The Pearson correlation coefficient measures the extent of deviation of the observed value from the best-fit line. It is defined as:

$$R = \frac{n\left(\sum_{i=1}^{n} O_i - P_i\right) - \left(\sum_{i=1}^{n} O_i\right) \cdot \left(\sum_{i=1}^{n} P_i\right)}{\sqrt{\left[n\sum_{i=1}^{n} O_i^2 - \left(\sum_{i=1}^{n} O_i\right)^2\right] \cdot \left[n\sum_{i=1}^{n} P_i^2 - \left(\sum_{i=1}^{n} P_i\right)^2\right]}} \tag{1}$$

2.4.2 Coefficient of determination (R^2)

Coefficient of determination is the square of Pearson correlation coefficient. It gives the variance between the predicted and the observed values. It gives a measure of the ability of the regression line to predict the data. R^2 is calculated as:

$$R^2 = \frac{\left[\sum_{i=1}^{n} \left(O_i - \overline{O}_i\right) \cdot \left(P_i - \overline{P}_i\right)\right]^2}{\sum_{i=1}^{n} \left(O_i - \overline{O}_i\right) \cdot \sum_{i=1}^{n} \left(P_i - \overline{P}_i\right)} \tag{2}$$

2.4.3 Mean absolute error

Mean absolute error (MAE) is the error calculated between two continues variables of the same parameter; in the present study, it refers to compressive strength. MAE is equal to the average distance between the y = x line (identity line) and the point in the scatter plot of (x, y), where x and y are observed and predicted values of compressive strength. The formula for MAE is:

$$MAE = \frac{\sum_{i=1}^{n} \left|O_i - P_i\right|}{n} \tag{3}$$

2.4.4 Root mean square error

Root mean square deviation or error is the measure of the standard deviation of prediction errors (residuals). It represents the degree of concentration of data points around the best-fit line. Root mean square error (RMSE) is evaluated using the following expression:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(P_i - O_i\right)^2}{n}} \tag{5}$$

2.4.5 Mean absolute percentage error

Mean absolute percentage error (MAPE) evaluates the accuracy of a predicted model in terms of percentage. MAPE is expressed as follows:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{O_i - P_i}{O_i}\right| \tag{6}$$

in which O_i, P_i represents the observed and predicted values, while n represents the number of test data. Using the above-mentioned formulae for the evaluation metric and the output from different ML techniques viz., observed and predicted values, the efficiency of the ML technique is analysed and the results are discussed in the following section.

3 Optimisation module

3.1 DCS algorithm

DCS Algorithm combines two popular optimisation algorithms, CS and DE. This hybrid algorithm has been proposed to overcome the limitations of the individual parent algorithms, and has been observed to demonstrate improved convergence and robustness. To grasp a comprehensive understanding of DCS Algorithm, it is imperative to have a clear understanding of both CS and DE algorithms. CS is a nature-inspired optimisation algorithm based on cuckoos' breeding behaviour. It uses the principle of survival of the fittest to evolve the solutions and find the optimal solution. On the other hand, DE is a heuristic optimisation algorithm that uses the differences between randomly selected solutions to guide the evolution of the solutions. By combining the strengths of both these algorithms, the DCS Algorithm seeks to provide an optimised and robust solution to various optimisation problems.

3.1.1 Differential evolution

DE is a bio-inspired algorithm developed by Storn and Price in 1997, which is based on the principles of evolution. It is a vector-based optimisation method that draws similarities to both GA and pattern search (PS). However, it provides several advantages over GA, making it a superior algorithm. One of the key features of DE is that it uses both updating equations and vector updates, which eliminates the need for encoding and

decoding as in GA, and reduces the computational costs associated with these processes. This makes DE easy to implement and allows for modifications to be made with ease.

DE has proven to be a more robust and efficient optimisation algorithm compared to GA (Price et al., 2006), as it uses a simple trial vector generation method and a more sophisticated mechanism for controlling the mutation step. This leads to a better convergence of the optimisation process and helps to avoid the problem of premature convergence that is common in other optimisation algorithms. The ease of implementation and the ability to work on various modifications are additional advantages of DE.

Consider an objective function containing variables; assume an initial population as n.

$$x_i^{cur} = \left(x_{i,1}^{cur}, x_{i,2}^{cur}, x_{i,3}^{cur}, ..., x_{i,t=d}^{cur} \right) \tag{7}$$

In equation (1), *cur* means a current solution, 'i' can take values from 1 to n corresponding to each population and d is the number of variables of objective function $f(x)$. As this is an evolutionary-based algorithm, this is considered to be a chromosome. There are three main steps in DE

1 mutation

2 crossover

3 selection.

$$\text{Mutation} : v_i^{updt} = x_i^{cur} + F * \left(x_j^{cur} - x_k^{cur} \right). \tag{8}$$

Here updt means updated chromosome, i, j, $k \in [1, n]$ and j, k are randomly chosen $F \in [0, 2]$ is a mutation parameter.

$$\text{Crossover } u_{i,t}^{updt} = \begin{cases} v_{i,t}^{updt} \ if \ r_i \le C \\ x_{i,t}^{cur} \end{cases}. \tag{9}$$

Here C is the crossover parameter and r_i is a random number generated for i^{th} population and $t \in [1, d]$ and this operation is done for every chromosome.

$$\text{Selection } x_i^{new} = \begin{cases} u_i^{updt} \ if \ f\left(u_i^{updt}\right) \le f\left(x_i^{cur}\right) \\ x_i^{cur} \end{cases} \tag{10}$$

In this process, the new population is generated and is selected between the current and updated population based on their objective function value. The whole DE algorithm is based on two parameters F and C. By varying these, the exploration and exploitation of the search mechanism can be controlled.

3.1.2 CS algorithm

CS is a swarm intelligence (SI) based, bio-inspired optimisation algorithm. SI is a collective intelligence of self-adaptive systems. We can see most of this kind of behaviour in nature for survival by many species. CS is based on one such behaviour of cuckoo birds in their breeding methods (Gandomi et al., 2013). Compared to many other

SI algorithms, the CS uses levy flights instead of conventional random walk. This makes its search much better than many other SI based optimisation algorithms.

Algorithm

Assume an objective function containing '*d*' variables (dimensions) and a number' *n*' nests. Here, instead of the population, we use nests as the algorithm is based on the breeding pattern of cuckoo birds.

$$x_i^{cur} = \left(x_{i,1}^{cur}, x_{i,2}^{cur}, x_{i,3}^{cur}, ...x_{i,t=d}^{cur} \right) \tag{11}$$

Here cur means current solution, *i* can take values from 1 to n corresponding to each population and d is the number of variables of objective function $f(x)$. In CS, the searching happens in two mechanisms:

1 global search

2 local search (Yang and Deb, 2009).

The shift is controlled by a switching parameter P_a. The global search happens using a special random walk called levy flight. Levy flight is a random number generator which uses levy distribution (Chechkin et al., 2008). One of the popular methods to generate random numbers using levy distribution is using Magenta algorithm.

$$step = \frac{u}{|v|^{\frac{1}{\beta}}} \tag{12}$$

Here, *s* is the step length and u and v are drawn from the normal distribution. $u \sim N(0, \sigma u^2)$ and $v \sim N(0, \sigma v^2)$.

$$\sigma u = \left\{ \frac{\Gamma(1+\beta) * \sin\left(\pi * \frac{\beta}{2} \right)}{\Gamma\left[\frac{1+\beta}{2} \right] * \beta * 2^{\frac{\beta-1}{2}}} \right\}^{1/\beta} \quad , \sigma v = 1 \tag{13}$$

The new solution is updated using this same magenta algorithm.

$$x_i^{new} = x_i^{cur} + \alpha * \left(x_j^{cur} - x_k^{cur} \right) \tag{14}$$

The local search is done by conventional random walk

$$x_i^{new} = x_i^{cur} + \alpha * \left(x_j^{cur} - x_k^{cur} \right) \tag{15}$$

Here, $i, j, k \in [1, n]$ and are chosen randomly.

The main advantage of CS is its searching mechanism which almost mimics the natural phenomenon. This advantage is obtained due to levy-flight based search steps. Taking this levy-flight based random search steps and combining them with the search mechanism of DE, the rate of convergence and robustness of the algorithm is improved. Two versions of hybridisation, namely, DCS-type 1, and type 2 are compared in this manuscript.

3.1.3 Differential cuckoo search

The DCS algorithm is a unique algorithm that combines the best features of both DE and CS. The main objective of this hybrid algorithm is to achieve a faster convergence rate while maintaining the robustness of the parent algorithms. The natural search mechanism of CS, which is inspired by its breeding patterns, has been incorporated into the DE algorithm to enhance the diverse search mechanism. This results in an algorithm that provides both diverse and effective search mechanisms, which results in improved optimisation performance. By combining the strengths of both algorithms, this hybrid algorithm provides an effective solution to various optimisation problems. Table 4 shows the modifications of all the hybrid algorithms considered for this study.

3.2 Mix design optimisation

3.2.1 Mix design optimisation

Finding the optimal mix proportion that results in the maximum compressive strength of concrete is a task that many researchers are interested in. The traditional method of accomplishing this involves preparing multiple concrete samples with varying mix proportions, testing their compressive strength after 28 days, and determining the mix that yields the highest strength as the best mix. However, this approach assumes that the mix proportion with the highest compressive strength is among the ones being studied, which may not always be the case. To overcome this issue and reduce the time and effort required for casting, curing, and testing, the proposed method can be employed.

We can develop compressive strength prediction models using the ML models from Section 3. These ML models can be used along with the best optimisation algorithm obtained from Section 2 to achieve the best mix proportions, which will maximise the compressive strength of concrete. The mathematical model for the same is as follows:

- *objective function:* maximise compressive strength (output from developed knowledge models)
- *constraints:* the mix pro portions should follow EFNARC guidelines.

$$380 < Cement + Flyash + Ground\ Grannular\ Blastfurnace\ Slag < 600$$

$$150 < Water < 210$$

$$750 < Coarse\ Aggregate < 1000$$

$$0.48 * Total\ Aggregate < Fine\ Aggregate < 0.55 * Total\ Aggregate$$

Note: **All the constraints are defined by the mass of material for making 1 m^3 of concrete

$$Convergence\ percentage = (Number\ of\ \ populations\ converged) / (total\ number\ of\ \ population = 50) * 100.$$

Each optimisation algorithm is run for 60 times, each time for 100 iterations using 50 population. The minimum, maximum and standard deviation for minimum function value is also reported similarly. Let us consider that we are trying to optimise an optimisation function where the global optima are zero. After 100 iterations with 50 population, we get the optimal solution (least value among all the 50 population) to be 10^{-6}. For a robust optimisation algorithm, it should try to bring most of the population to the near-optimal

value (in this case, 10^{-6}). So, taking 10^{-5} as a tolerance value, this work assumes that all the population which are close to a near-optimal value within 10^{-5} tolerance to be converged (i.e. absolute difference between the best population – other population $< 10^{-5}$).

3.2.2 Scenario 2

A contractor generally aims to obtain an economical mix proportion for a desired compressive strength. It is always a tedious task to find least-cost mix proportions. So, the objective function is to minimise the cost and an additional constraint for desired compressive strength is added to the constraints mentioned above. The cost for each material is obtained from a local material supplier and is presented in Table 3. The cost of superplasticiser is not included in the study as it varies extensively and depends on the supplier. So, the obtained cost is not the final cost of the mix but is a representation of the cost.

Objective function:

$$Cost = 8 * Cement + 0.75 * FlyAsh + 3.5 * GGBS + 0.85 * Coarse\ Aggregate + 3.5 * Fine\ Aggregate \tag{16}$$

Constraints:

$$Predicted\ Compressive\ strenght\ from\ ML\ model > desired\ compressive\ strength$$

The mix proportions should follow EFNARC and NRMC guidelines (Medgar et al., 2007).

$$380 < Cement + Flyash + Ground\ Grannular\ Blastfurnace\ Slag < 600$$

$$150 < Water < 210$$

$$750 < Coarse\ Aggregate < 1000$$

$$0.48 * Total\ Aggregate < Fine\ Aggregate < 0.55 * Total\ Aggregate$$

Note: All the constraints and objective function are defined by the mass of material for making 1 m^3 of concrete.

Table 3 Materials' cost and embodied carbon

Material (kg/m³)	*Rupees/kg*	*Kg-e CO_2/kg*
Cement	8	0.913
Flash	0.75	0.004
GGBS	3.5	0.067
Coarse aggregate	0.85	0.005
Fine aggregate	3.5	0.005
Water	-	0.001
Superplasticiser	-	0.01

Table 4 Comparison of the algorithms' mechanism used

NCS	*DE*	*NBCS*	*DCS-1*	*DCS-2*
1 *Initialise 'n' nests in the search space*	**1 Initialise 'n' population in the search space**	**1 Initialise 'n' nests in the search space**	**1 Initialise 'n' nests in the search space**	**1 Initialise 'n' nests in the search space**
for $i = 1{:}n$	for $i = 1{:}n$	for $i = 1{:}n$	for $i = 1{:}n$	for $i = 1{:}n$
$\boldsymbol{nest(i,:) = lb} + \{(\boldsymbol{ub - lb}) * \boldsymbol{rand(1, nv)}\}$	$pop(i,:) = lb + \{(ub - lb) * rand(1, nv)\}$	$nest(i,:) = lb + \{(ub - lb) * rand(1, nv)\}$	$nest(i,:) = lb + \{(ub - lb) * rand(1, nv)\}$	$nest(\mathrm{i},:) = lb + \{(ub - lb) * rand(1, nv)\}$
lb = lower bound	*lb = lower bound*	*lb = lower bound*	*lb = lower bound*	*lb = lower bound*
ub = upper bound	*ub = upper bound*	*ub = upper bound*	*ub = upper bound*	*ub = upper bound*
nv = number of variables	*nv = number of variables*	*nv = number of variables*	*nv = number of variables*	*nv = number of variables*
rand = random number in [0,1]	*rand = random number in [0,1]*	*rand = random number in [0,1]*	*rand = random number in [0,1]*	*rand = random number in [0,1]*
2 Generation of new nests	**2 Generation of new population**	**2 Generation of new nests**	**2 Generation of new nests**	**2 Generation of new nests**
r1 = randperm(n,cuckoo) {Selection}	For i = 1:n	For i = 1:n	For i = 1:n	For i = 1:n
for i = 1:cuckoo	*Mutation*	$nest1 = nest(i,:) + levy$	$nest1 = nest(i,:) + levy + DE$	$nest1 = nest(i,:) + levy + DE$
$\boldsymbol{nest1 = nest(r1(i){:}) + levy}$	$\boldsymbol{m(i,:)}$			
Mantegna's Algorithm for generation of random step sizes	$= \boldsymbol{pop}(\mathrm{i},:) + \boldsymbol{F} * (\boldsymbol{pop(m,:)}$	**Mantegna's Algorithm for generation of random step sizes**	**Mantegna's Algorithm for generation of random step sizes**	**Mantegna's Algorithm for generation of random step sizes**
$U = N(0,\sigma^2)$	$-\boldsymbol{pop(j,:)}$	$U = N(0,\sigma^2)$	$U = N(0,\sigma^2)$	$U = N(0,\sigma^2)$
$V = N(0, 1)$	Cross over	$V = N(0, 1)$	$V = N(0, 1)$	$V = N(0, 1)$
$\sigma^2 = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma\left(\frac{1+\beta}{2}\right)\times 2^{\frac{\beta-1}{2}}}\right\}$	For k = 1:nd If r1 < Cr $npop(i,k) = m(i,k)$ else	$\sigma^2 = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma\left(\frac{1+\beta}{2}\right)\times 2^{\frac{\beta-1}{2}}}\right\}$	$\sigma^2 = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma\left(\frac{1+\beta}{2}\right)\times 2^{\frac{\beta-1}{2}}}\right\}$	$\sigma^2 = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma\left(\frac{1+\beta}{2}\right)\times 2^{\frac{\beta-1}{2}}}\right\}$
$step = \frac{u}{\lvert v\rvert^{\frac{1}{\beta}}}$	$npop(i,k) = pop(i,k)$ end end	$step = \frac{u}{\lvert v\rvert^{\frac{1}{\beta}}}$	$step = \frac{u}{\lvert v\rvert^{\frac{1}{\beta}}}$	$step = \frac{u}{\lvert v\rvert^{\frac{1}{\beta}}}$

Notes: **NCS – Normal cuckoo search, NBCS – Normal best cuckoo search, DE – Differential evolution, DCS-1 – Differential cuckoo search type 1, DCS-2 – Differential Cuckoo Search Type-2.

Table 4 Comparison of the algorithms' mechanism used

NCS	*DE*	*NBCS*	*DCS-1*	*DCS-2*
2 **Generation of new nests** $s = \alpha \times step \times \begin{Bmatrix} nest(j,:) \\ -nest(m,:) \end{Bmatrix}$ $j,m \in [1,2,...,n]$ $levy = randn \times s$ If nest1 has a better fitness than the original nest, update the nest. 3 **Elimination** $\boldsymbol{k = rand(size(nest)) \leq p}$ $\boldsymbol{step = rand \times k \times \{nest(j,:) - nest(m,:)\}}$ for i = 1:n $\boldsymbol{nest1 = nest(i,:) + step}$ If nest1 has a bett, fitness than the original nest, update the nest. 4 **Repeat step 2 and 3 until termination criteria are satisfied.**	2 **Generation of new population** Selection If npop has better fitness than pop update pop with npop. 3 **Repeat step 2 until termination condition**	2 **Generation of new nests** $s = \alpha \times step \times \begin{Bmatrix} nest(i,:) \\ -bestnest \end{Bmatrix}$ $levy = randn \times s$ If nest1 has a better fitness than the original nest, update the nest. 3 **Elimination** $k = rand(size(nest)) \leq p$ $step = rand \times k \times \{nest(j,:) - nest(m,:)\}$ for i = 1:n $nest1 = nest(i,:) + step$ If nest1 has a better fitness than the original nest, update the nest. 4 **Repeat step 2 and 3 until termination criteria is satisfied.**	2 **Generation of new nests** $s = \alpha \times step \times \begin{Bmatrix} nest(j,:) \\ -nest(m,:) \end{Bmatrix}$ $levy = randn \times s$ $DE = 0.9 \times \{nest(k,:) - nest(l,:)\}$ $j,m,k,l \in [1,2,3...n]$ If nest1 has a better fitness than the original nest, update the nest. 3 **Elimination** $k = rand(size(nest)) \leq p$ $step = rand \times k \times \{nest(j,:) - nest(m,:)\}$ for i = 1:n $nest1 = nest(i,:) + step$ If nest1 has a better fitness than the original nest, update the nest. 4 **Repeat step 2 and 3 until termination criteria is satisfied.**	2 **Generation of new nests** $s = \alpha \times step \times \begin{Bmatrix} nest(i,:) \\ -bestnest \end{Bmatrix}$ $levy = randn \times s$ $DE = 0.9 \times \{nest(k,:) - nest(l,:)\}$ $j,m,k,l \in [1,2,3...n]$ If nest1 has a better fitness than the original nest, update the nest. 3 **Elimination** $k = rand(size(nest)) \leq p$ $step = rand \times k \times \{nest(j,:) - nest(m,:)\}$ for i = 1:n $nest1 = nest(i,:) + step$ If nest1 has a better fitness than the original nest, update the nest. 4 **Repeat step 2 and 3 until termination criteria is satisfied.**

Notes: **NCS – Normal cuckoo search, NBCS – Normal best cuckoo search, DE – Differential evolution, DCS-1 – Differential cuckoo search type 1, DCS-2 – Differential Cuckoo Search Type-2.

3.2.3 Scenario 3

With an increase on awareness of environmental degradation and the role of human beings in that, research has moved towards eco-friendly concrete. Every manufactured material is associated with a carbon index, i.e. the amount of carbon dioxide released in its manufacturing process. To obtain a good eco-friendly mix, the Carbon-Index of the concrete needs to be reduced. The carbon index associated with each material as shown in Table 3 is taken from United Kingdom Fact Sheet (MPA, 2008) and few research papers (Siddique et al., 2011; Turner and Collins, 2013; Purnell and Black, 2012) Similar to cost optimisation, a desired compressive strength will be a constraint along with EFNARC guidelines (EFNARC, 1999, 2002; Group, 2005).

Objective function:

$$\begin{aligned} CO_2\ emmisions = {} & 0.913 * Cement + 0.004 * FlyAsh + 0.067 * GGBS \\ & + 0.005 * Coarse\ Aggregate + 0.005 * Fine\ Aggregate \\ & 0.01 * Superplaticiser + 0.001 * Water \end{aligned} \tag{17}$$

Constraints:

$$Predicted\ Compressive\ strength\ from\ ANN > desired\ compressive\ strength$$

The mix proportions should follow guidelines of EFNARC and Mineral Products Association (MPA, 2008; Group, 2005).

$$380 < Cement + Flyash + Ground\ Grannular\ Blastfurnace\ Slag < 600$$

$$150 < Water < 210$$

$$750 < Coarse\ Aggregate < 1000$$

$$0.48 * Total\ Aggregate < Fine\ Aggregate < 0.55 * Total\ Aggregate$$

All the constraints and objective function are defined by the mass of material for making 1 m^3 of concrete.

The single objective optimisation is done for all the three scenarios by using the robust optimisation technique obtained from the comparison done in stage 1. The constraints are included in the optimisation algorithm using exterior penalty approach. The results obtained are summarised in Section 4.

4 Results and discussion

4.1 Evaluation of robust optimisation algorithm

CS, DE and the three proposed modifications are compared for their efficiency using 11 optimisation test functions. For all the algorithms, the 50 search agents are used to maintain uniformity, and the algorithm specific parameters used are shown in Table 5. The comparative analysis is done by two criteria:

1 function value

2 percentage of convergence.

After running the algorithm for 100 runs for each objective function the best, the worst and the standard deviation of the function values are shown in the Table 7. The percentage conversion is also performed for every run and the results are depicted for the same. This is calculated by finding a number of nests having fitness value less than the desired number, for example, if the global optimum is 0 then all the nests having fitness values less than 10^{-5} are considered as converged nests.

$$convergence = \frac{total\ nests\ having\ \ fitness\ less\ than\ a\ desired\ value}{total\ number\ of\ \ nests\ taken} * 100 \tag{18}$$

An optimisation algorithm is considered best if it can produce the least fitness value, can produce it more frequently or in other words, is independent of the initial nest values. As we can see from Table 7, when the CS is enhanced with DE (DCS), the functionality of the algorithm is better compared to individual/parent algorithms. The best optimisation algorithm 'DCS – 1' is selected to solve the mix design optimisation problem.

The box plots (as seen in Figure 4) show the deviations between the various algorithms. Mean, median, standard deviation, outlier limits and various other central tendencies of the distribution can be observed. Moreover, to estimate the statistical significance of the differences more effectively, we have used paired t-test. In our application, all test-functions (f_1 to f_{11}) are minimisation functions. The *t.test*() function in R package is used for estimating the statistical significance of the results. The alternative option-string (alt) is set to '*less*'; R checks whether the difference in mean of the values contained in the vector 1 is less of the mean of the values contained in the vector 2. The baseline groups (vector 2), variable groups (vector 1), their corresponding computed t-statistic, and their probability values (p-values) are presented in Table 8. If the p-value is well above 0.05, it leads us to conclude that we can reject the null hypothesis H_0 in favour of the alternative hypothesis H_1. If p-value is greater than the confidence interval (0.05) then the 2nd sample (baseline group algorithm) is better than the 1st sample (variable group algorithm). From a geometric perspective on the landscapes of the test functions, it can be noticed that the proposed hybrid approaches perform well in smooth landscapes more than rough landscapes with many local minima. From the observations made for the baseline algorithms DCS-1 and DCS-2, the former is better for 29 comparisons and the latter is better for 28 comparisons. Hence, the formulated mathematical model under different scenarios would be optimised using DCS-1.

Table 5 Summary of key parameter values of all optimisation algorithms

Parameter	*DE*	*CS*	*DCS*
Number of search agents (n)	50 agents	50 agents	50 agents
Crossover coefficient (CR)	0.7	-	0.7
Mutation coefficient (F) / Differential weight	0.4	-	0.4
Levy step length *(α – Alpha)*	-	0.01	0.01
Levy distribution parameter *(β – Beta)*	-	1.5	1.5
Mantegna gamma function *(Γ – Gamma)*	-	(n-1)!	(n-1)!

Figure 3 Shapes of benchmark test functions, (a) f_1 Ackley function (b) f_2 Beale function (c) f_3 booth function (d) f_4 Matyas function (e) f_5 McCormick function (f) f_6 egg holder function (g) f_7 De Jong function (h) f_8 drop-wave function (i) f_9 Rosenbrock function (j) f_{10} Shubert function (k) f_{11} sphere function (see online version for colours)

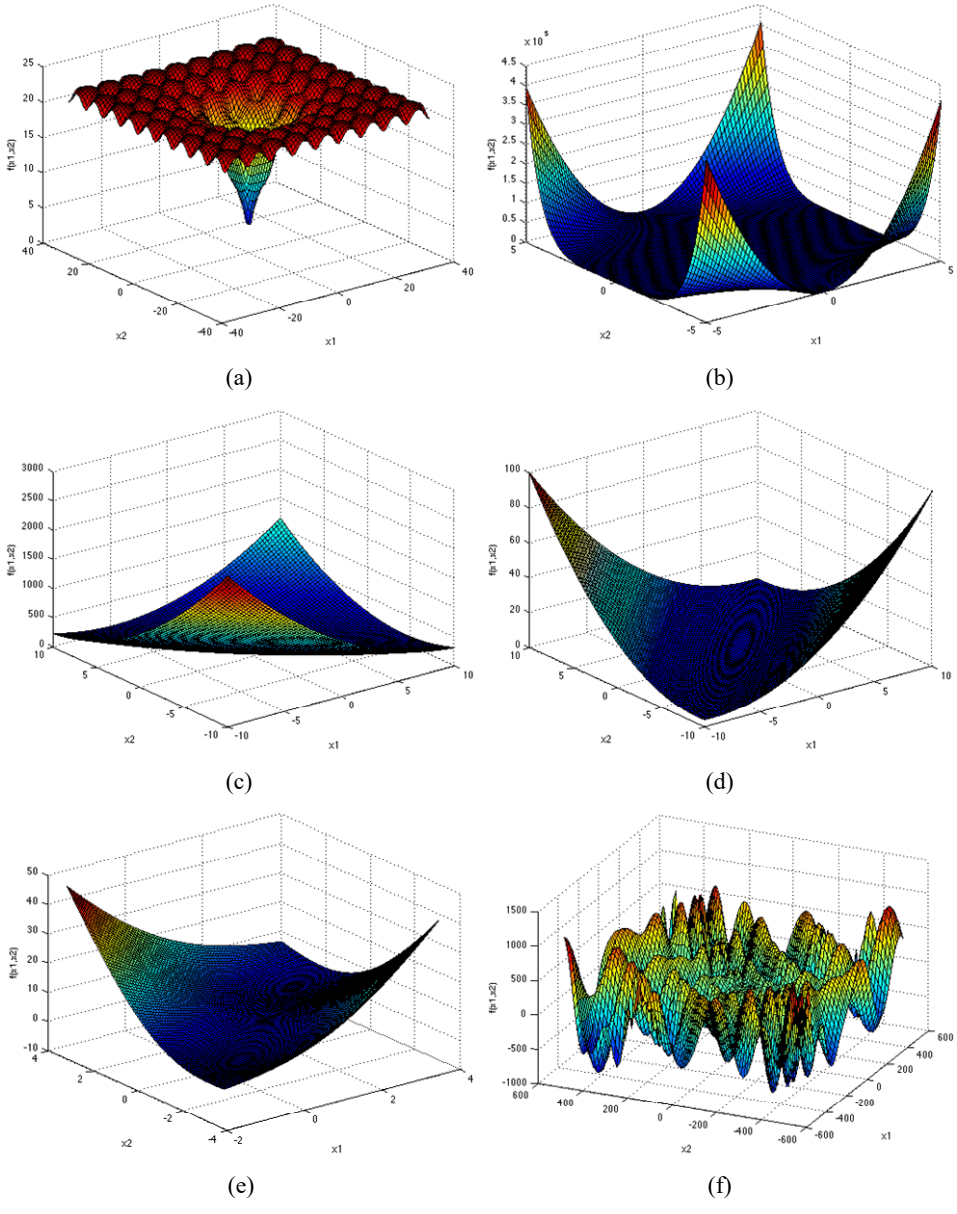
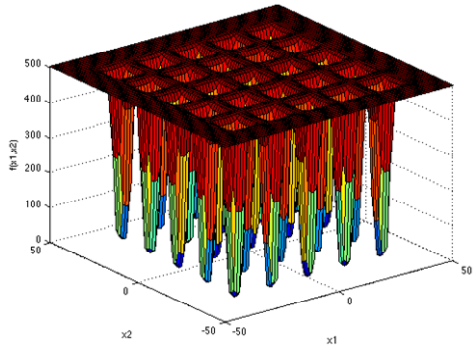
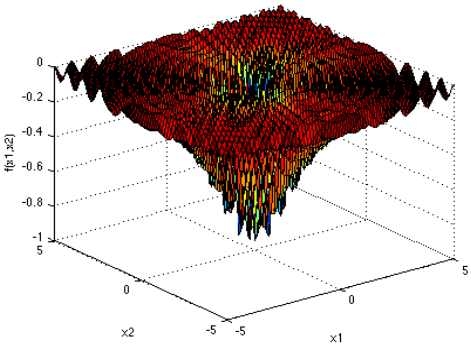


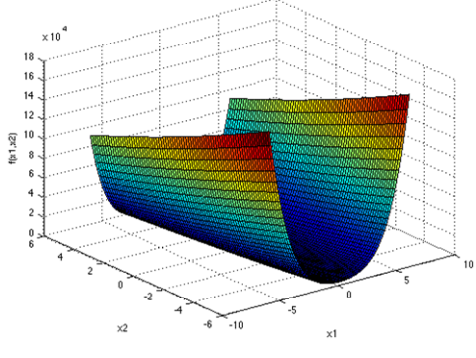
Figure 3 Shapes of benchmark test functions, (a) f_1 Ackley function (b) f_2 Beale function (c) f_3 booth function (d) f_4 Matyas function (e) f_5 McCormick function (f) f_6 egg holder function (g) f_7 De Jong function (h) f_8 drop-wave function (i) f_9 Rosenbrock function (j) f_{10} Shubert function (k) f_{11} sphere function (continued) (see online version for colours)



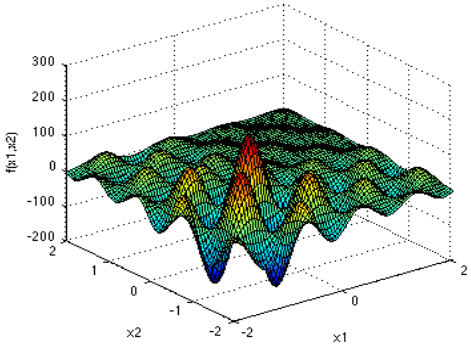
(g)



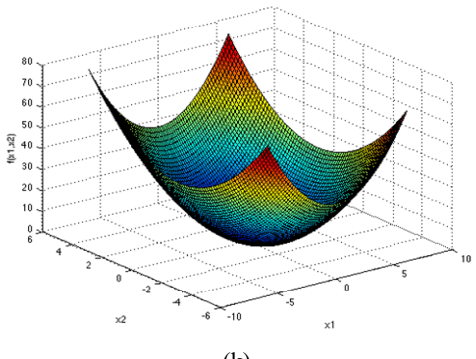
(h)



(i)



(j)



(k)

Table 6 Description of the benchmark test functions

		Description	*Input domain*	*Optimal solution*
f_1	Ackley function	$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$	$x_i \in [-32.76, 32.76]$ for all i = 1, …, d. Dimensions (d) = 16. Parameters: a = 20 b = 0.2 c = 2π	$f(x^*) = 0$ at $x^* = (0, 0)$
f_2	Beale function	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$x_i \in [-4.5, 4.5]$ for all i = 1, …, d. Dimensions (d) = 2	$f(x^*) = 0$ at $x^* = (3, 0.5)$
f_3	Booth function	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$x_i \in [-10, 10]$, for all i = 1, …, d. Dimensions (d) = 2	$f(x^*) = 0$ at $x^* = (1, 3)$
f_4	Matyas function	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$x_i \in [-10, 10]$, for all i = 1, …, d. Dimensions (d) = 2.	$f(x^*) = 0$ at $x^* = (0, 0)$
f_5	McCormick function	$f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	$x_1 \in [-1.5, 4]$, $x_2 \in [-3, 4]$ for all i = 1, …, d. Dimensions (d) = 2	$f(x^*) = -1.9133$ at $x^* = (-0.541, -1.547)$
f_6	Egg holder function	$f(x^*) = -959.6407$, at $x^* = (512, 404.2319)$	$x_i \in [-512, 512]$, for all i = 1, 2.	$f(x^*) = -959.6407$ $x^* = (512, 2319)$

Table 6 Description of the benchmark test functions

	Description	*Input domain*	*Optimal solution*
f_7	De Jong function $f(x) = \left(0.002 + \sum_{j=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6}\right)^{-1}$, where $a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$	$x_i \in [-65.536, 65.536]$, for all i = 1, 2.	$f(x^*) = 0$ $x^* = (0,\dots, 0)$
f_8	Drop-wave function $f(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5\left(x_1^2 + x_2^2\right) + 2}$	$x_i \in [-5.12, 5.12]$, for all i = 1, 2.	$f(x^*) = -1$ $x^* = (0, 0)$
f_9	Rosenbrock function $f(x) = \sum_{i=1}^{d-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$x_i \in [-5, 10]$, for all i = 1, …, d	$f(x^*) = 0$ $x^* = (1,\dots, 1)$
f_{10}	Shubert function $f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	$x_i \in [-10, 10]$, for all i = 1, 2	$f(x^*) = -186.7309$
f_{11}	Sphere function $f(x) = \sum_{i=1}^{d} x_i^2$	$x_i \in [-5.12, 5.12]$, for all i = 1, …, d.	$f(x^*) = 0$ $x^* = (0,\dots, 0)$

Table 7 Comparison between all the algorithms using 11 different test functions

		NCS		*DE*		*NBCS*		*DCS-1*		*DCS-2*	
		Function	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*
f1	min	1.46636E-06	04.000	2.85799635	100	6.58483E-07	36.000	7.55351E-11	100	*6.10436E-11*	*100*
	std	0.000425426	23.842	0.73464445	0	3.14384E-06	09.730	1.36451E-09	0	*1.81492E-09*	*0*
	max	0.003412398	100	7.26860074	100	1.96842E-05	100	1.1165E-08	100	*1.40554E-08*	*100*
f2	min	6.29442E-11	04.000	*6.0318E-11*	*100*	8.61245E-09	04.000	4.48104E-15	12.000	9.51963E-15	14.000
	std	1.31537E-05	08.523	*0.647598!*	*0*	0.000134179	02.955	6.21601E-07	14.371	5.52069E-08	08.920
	max	7.09565E-05	44.000	*3.48245313*	*100*	0.000770731	20.000	6.2275E-06	100	5.15419E-07	100
f3	min	3.44993E-11	08.000	5.8207E-12	100	7.02148E-10	04.000	*8.16648E-18*	*96.000*	5.38752E-17	96.000
	std	7.67798E-07	11.306	3.41810635	0	1.90967E-06	10.879	*1.27297E-12*	*01.020*	1.96689E-12	0.6823
	max	4.97372E-06	68.000	24.9687148	100	1.10263E-05	64.000	*1.11429E-11*	*100*	1.34883E-11	100
f4	min	1.28693E-11	48.000	1.4902E-18	100	1.42394E-11	40.000	*1.0667E-17*	*96.000*	9.83559E-19	96.000
	std	2.23242E-08	09.886	0.0707281	0	7.77554E-08	10.073	*3.76918E-14*	*0.6823*	8.78123E-14	0.6823
	max	1.46699E-07	92.000	0.36648093	100	5.072E-07	92.000	*2.81028E-13*	*100*	7.24297E-13	100
f5	min	–1.913222955	28.000	*–1.913223*	*100*	–1.913222954	20.000	–1.913222955	92.000	*–1.913222955*	*96.000*
	std	2.03998E-07	10.812	0.207510	*0*	3.52048E-07	12.236	9.06913E-12	01.459	*3.14652E-13*	*01.345*
	max	–1.913220986	80.000	–1.022469	*100*	–1.913220276	72.000	–1.913222955	100	*–1.913222955*	*100*
f6	Min	–959.641	02.000	*–959.64066*	*100*	–959.641000	24.000	–959.641	70.000	–959.641	70.000
	Std	1.319238	01.7192	*17.180812*	*0*	1.85E-13	08.768	0.00000	04.645	0.00000	04.416
	max	–946.467	06.000	*–858.60121*	*100*	–959.641000	78.000	–959.641	94.000	–959.641	96.000

Table 7 Comparison between all the algorithms using 11 different test functions (continued)

		NCS		*DE*		*NBCS*		*DCS-1*		*DCS-2*	
		Function	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*	*Function*	*Convergence*
f7	Min	0.998004	02.000	0.998004	100	0.99800400	84.000	0.998004	92.000	*0.998004*	*92.000*
	Std	4.45E-10	05.993	0.546043	0	1.12E-16	2.4790	1.12E-16	01.831	*1.12E-16*	*01.827*
	max	0.998004	30.000	5.928845	100	0.99800400	100	0.998004	100	*0.998004*	*100*
f8	Min	–1.00000	00.000	*–1.00000*	*100*	–1.00000	42.000	–1.00000	76.000	–1.00000	72.000
	Std	0.007631	02.334	*0.00000*	*0*	3.42E-13	07.556	0.00000	04.566	0.00000	04.571
	max	–0.93625	16.000	*–1.00000*	*100*	–1.00000	76.000	–1.00000	98.000	–1.00000	98.000
f9	Min	3.49E-29	78.000	2.766649	100	5.85E-07	0	8.59E-16	96.000	*7.17E-16*	*96.000*
	Std	1.67E-17	03.562	0.933633	0	0.006154	02.324	2.59E-10	01.011	*3.37E-11*	*01.088*
	max	1.28E-16	96.000	8.292449	100	0.039016	12.000	2.13E-09	100	*3.02E-10*	*100*
f10	Min	–186.731	0.0000	*–186.730909*	*100*	–186.731	76.000	–186.731	44.000	–186.731	46.000
	Std	0.000443	0.2814	*9.6298E-15*	*0*	4.91E-11	4.33034	4.65E-09	07.965	9.18E-09	07.581
	max	–186.729	2.0000	*–186.730909*	*100*	–186.731	98.000	–186.731	82.000	–186.731	82.000
f11	Min	4.89E-17	100	5.3964E-133	100	2.17E-33	100	8.65E-137	100	*1.25E-210*	*100*
	Std	1.95E-16	0	3.3056E-129	0	2.29E-31	0	5.22E-135	0	*6.78E-156*	*0*
	max	1.18E-15	100	2.5035E-128	100	2.17E-30	100	3.97E-134	100	*6.65E-155*	*100*

Figure 4 Box plot of the performance of the algorithms, (a) f_1 Ackley function (b) f_2 Beale function (c) f_3 Booth function (d) f_4 Matyas function (e) f_5 McCormick function (f) f_6 Egg holder function (g) f_7 De Jong function (h) f_8 Drop-wave function (i) f_9 Rosenbrock function (j) f_{10} Shubert function (k) f_{11} Sphere function

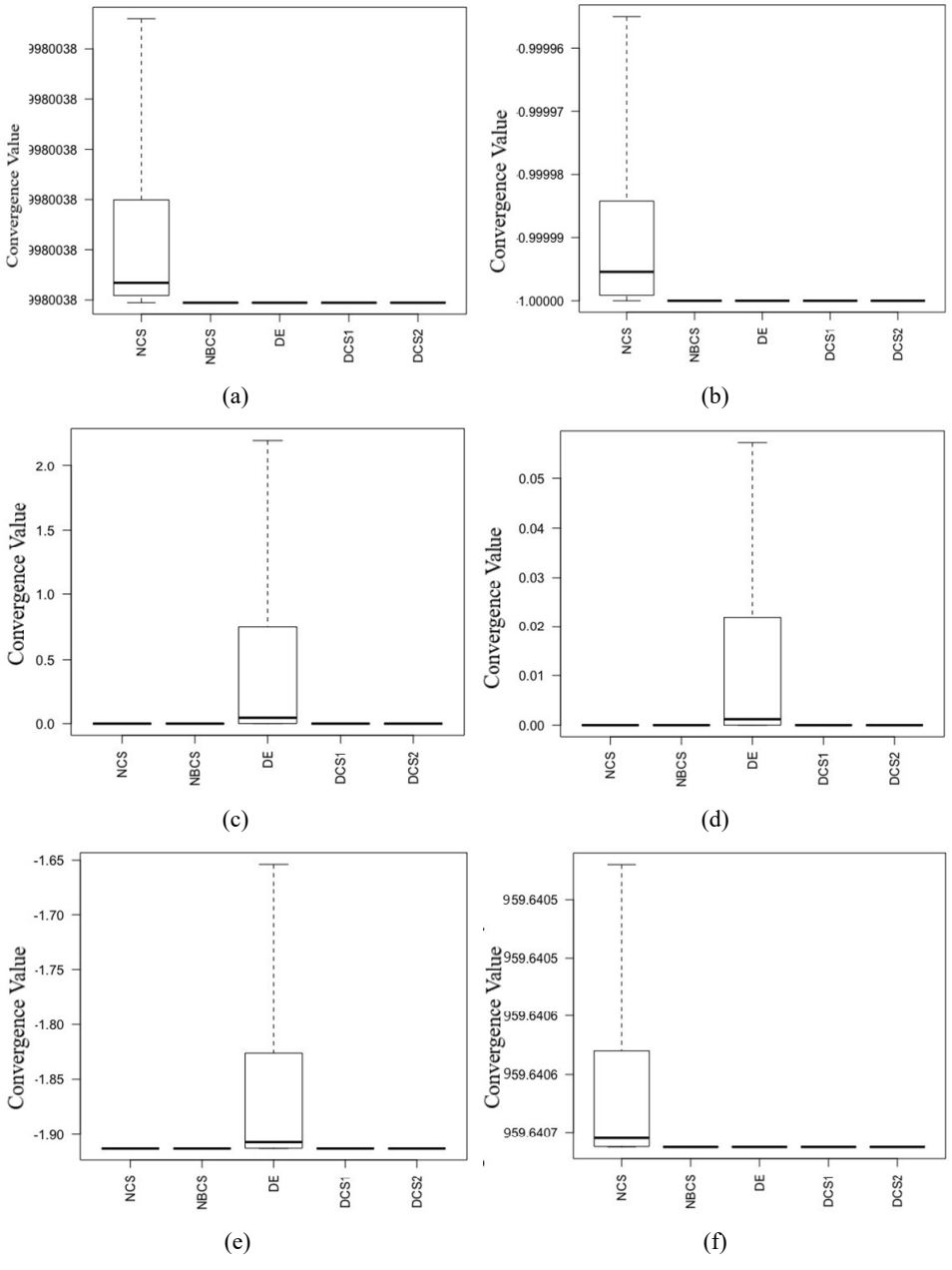
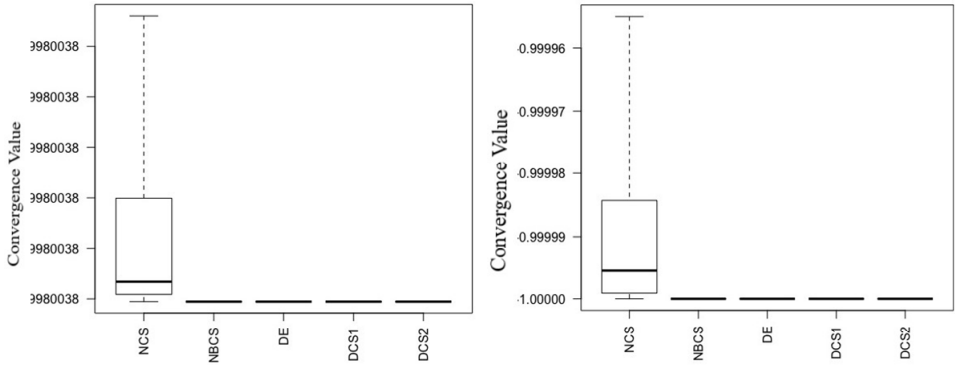
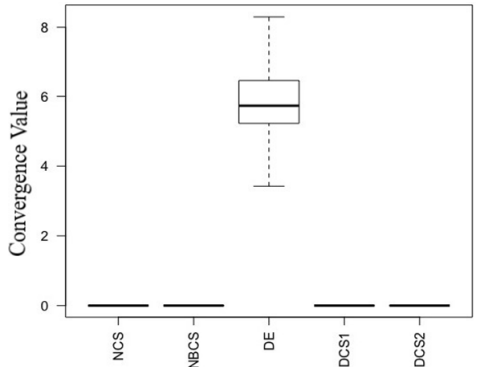


Figure 4 Box plot of the performance of the algorithms, (a) f_1 Ackley function (b) f_2 Beale function (c) f_3 Booth function (d) f_4 Matyas function (e) f_5 McCormick function (f) f_6 Egg holder function (g) f_7 De Jong function (h) f_8 Drop-wave function (i) f_9 Rosenbrock function (j) f_{10} Shubert function (k) f_{11} Sphere function

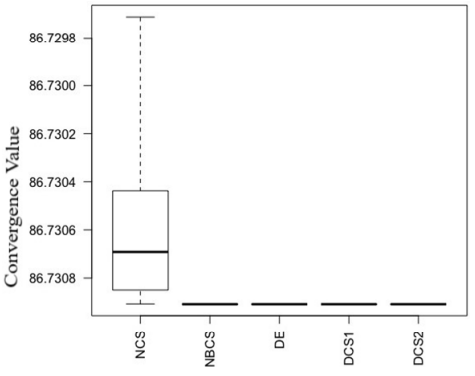


(g)

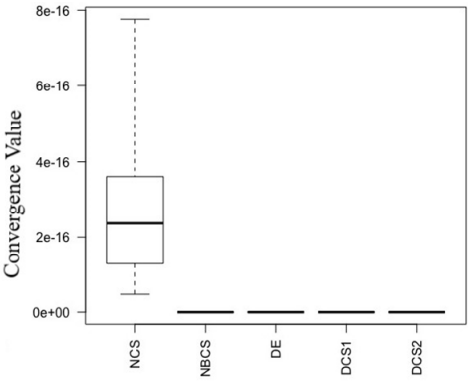
(h)



(i)



(j)



(k)

Table 8 Statistical tests for algorithms

Baseline group			*DCS1*		*DCS2*	
		Variable groups	*t-stat*	*p-value*	*t-Stat*	*p-value*
f1	Ackley function	NCS	3.5362	*0.9997*	3.7078	*0.9998*
		NBCS	13.936	*1.0000*	13.877	*1.0000*
		DE	61.793	*1.0000*	62.086	*1.0000*
		DCS (1/2)*	0.9903	*0.8378*	–0.9903	0.1622
f2	Beale function	NCS	–9.0147	7.7E-15	–8.4047	1.6E-13
		NBCS	–6.1696	7.5E-09	–4.8241	2.5E-06
		DE	–7.3987	2.2E-11	–5.3493	2.8E-07
		DCS(1/2)	–2.5228	0.0066	2.5228	0.0491
f3	Booth function	NCS	4.8190	*1.0000*	4.8190	*1.0000*
		NBCS	6.8559	*1.0000*	6.8559	*1.0000*
		DE	3.9889	*0.9999*	3.9889	*0.9999*
		DCS(1/2)	0.6376	*0.7374*	–0.6376	0.2626
f4	Matyas function	NCS	5.8003	*1.0000*	5.8003	*1.0000*
		NBCS	6.1458	*1.0000*	6.1458	*1.0000*
		DE	4.4845	*1.0000*	4.4845	*1.0000*
		DCS(1/2)	0.9750	*0.8340*	–0.9750	0.1660
f5	McCormick function	NCS	1.8937	*0.9694*	1.8937	*0.9694*
		NBCS	4.6675	*1.0000*	4.6675	*1.0000*
		DE	4.8938	*1.0000*	4.8938	*1.0000*
		DCS(1/2)	–1.0137	0.1566	1.0137	*0.8434*
f6	Egg holder function	NCS	1.0955	*0.8620*	1.0955	*0.8620*
		NBCS	1.0000	*0.8401*	1.0000	*0.8401*
		DE	2.2445	0.0065	2.2445	0.0065
		DCS(1/2)	NaN	NA	NaN	NaN
f7	De Jong function	NCS	4.1883	*1.0000*	4.1883	*1.0000*
		NBCS	0.2569	*0.6011*	0.83076	*0.7959*
		DE	2.9341	*0.9979*	2.9341	*0.9979*
		DCS(1/2)	–0.5637	0.2871	0.56375	*0.7129*
f8	Drop-wave function	NCS	1.6592	*0.9499*	1.6592	*0.9499*
		NBCS	2.6130	*0.9948*	2.6130	*0.9948*
		DE	1.4214	0.0208	1.4214	0.0208
		DCS(1/2)	NaN	NA	NaN	NaN
f9	Rosenbrock function	NCS	–1.5053	*0.0677*	–1.9718	0.0257
		NBCS	3.0451	*0.9985*	3.0451	*0.9985*
		DE	61.911	*1.0000*	61.911	*1.0000*
		DCS(1/2)	–1.2343	0.1100	1.2343	*0.8900*

Note: *DCS (1/2): If Baseline group is DCS1, then variable group is DCS 2.

Table 8 Statistical tests for algorithms (continued)

Baseline group			*DCS1*		*DCS2*	
		Variable groups	*t-stat*	*p-value*	*t-Stat*	*p-value*
f10	Shubert function	NCS	8.5560	*1.0000*	8.5560	*1.0000*
		NBCS	–7.3299	3.1E-11	–5.3142	3.3E-07
		DE	–7.3987	2.2E-11	–5.3493	2.8E-07
		DCS(1/2)	1.3932	*0.9167*	–1.3932	0.0833
f11	Sphere function	NCS	14.036	*1.0000*	14.036	*1.0000*
		NBCS	3.5522	*0.9997*	3.5535	*0.9997*
		DE	–5.8285	3.5E-08	3.9459	3.5E-08
		DCS(1/2)	–5.8285	3.5E-08	5.8285	*1.0000*

Note: *DCS (1/2): If Baseline group is DCS1, then variable group is DCS 2.

4.2 Performance metrics of machine learning models

The five statistical parameters (also known as performance metrics) to evaluate the efficiency of ML techniques used in the present study are R, R^2, RMSE, MAE and MAPE; they are summarised in Table 9. Only a limited variation in these metrics is observed on performing multiple iterations.

Table 9 Performance evaluation of ML models

ML model	*R*	*R^2*	*RMSE*	*MAE*	*MAPE*
ANN	0.919	0.845	5.697	4.191	12.866
SVR	0.923	0.852	5.608	3.904	11.137

It is observed from Table 9 that SVR performs better than ANN in terms of correlation (R&R^2) and error (RMSE). However, the variations in the performance metrics for SVR and ANN are not very significant. Both these ML techniques may be used to test the available data to obtain the knowledge model. The developed knowledge models from both the ML approaches are used for obtaining the optimal mix proportion using DCS-1 under three different scenarios. Table 10, 11 and 12 shows the results of optimal mix proportion obtained using DCS-1 for Scenario 1, 2 and 3 respectively.

Table 10 Scenario 1: optimal mix proportion for obtaining maximum compressive strength

Material (kg/m^3)	*SVR*	*ANN*
C	490.00	490.30
GGBS	110.00	0.14
FLYA	0.00	109.49
CA	900.00	870.29
FA	830.77	1022.67
W	158.54	155.25
Compressive strength (MPa)	69.14	98.75

It is observed from Table 10 that ANN achieved a 30% increased compressive strength in comparison with SVR based framework. The compressive strength obtained are 98.75 MPa using ANN and 69.14 MPa using SVR. The weight of GGBS was very low (0.14 kg/m^3) using ANN but SVR's mix design had 110 kg/m^3. Similarly, the weight of fly ash using ANN was 109.49 kg/m^3 but it was nil for the mix proportion using SVR. From Table 11, it is observed that the variation of the cost of the economical mix for a fixed compressive strength (30 MPa and 50 MPa) is relatively low in the range of 2% to 4% only while solving using SVR and ANN approaches. The water content in the ANN approach in both the cases used is around 36% more in comparison with SVR mix proportion. While optimising for cost, the mix proportions obtained by ANN and SVR eliminated the use of GGBS and fly ash in both the cases of compressive strength i.e., 30 MPa and 50 MPa.

Table 11 Scenario 2: obtaining economical mix for desirable compressive strength

Material (kg/m^3)	*CS (Expected) = 30*		*CS (Expected) = 50*	
	SVR	*ANN*	*SVR*	*ANN*
Cost (Rs./m^3)	6,100.93	6,395.33	6,769.21	6,626.52
C	380.00	416.79	380.24	441.39
GGBS	0.00	0.00	0.00	0.00
FLYA	0.00	0.00	0.00	0.00
CA	750.06	750.03	799.82	751.58
FA	692.38	692.42	870.70	701.89
W	152.21	207.42	150.01	209.38

Table 12 Scenario 3: obtaining eco-friendly mix for desirable compressive strength

Material (kg/m^3)	*CS (Expected) = 30*		*CS (Expected) =50*	
	SVR	*ANN*	*SVR*	*ANN*
Embodied carbon (Kg-e CO_2/m^3)	354.31	378.95	428.97	405.77
C	380.01	406.13	460.08	433.58
GGBS	0.00	0.00	15.12	25.11
FLYA	0.00	0.00	1.43	0.00
CA	750.03	752.15	769.89	790.51
FA	692.56	847.98	722.36	825.46
W	150.03	154.77	150.01	150.02

The results of mix proportions obtained in Scenario 3 as shown in Table 12 for embodied carbon also didn't show too much variation while solving using ANN and SVR. SVR achieved a lower embodied carbon for compressive strength of 30 MPa and ANN achieved the least embodied carbon for compressive strength of 50 MPa. In all these mix proportions it was observed that the weights of GGBS and fly ash are zero for compressive strength of 30 MPa and relatively low for compressive strength of 50 MPa. It can be inferred from all the three scenarios that for obtaining a mix proportion to achieve maximum compressive strength, ANN-based bio-inspired optimisation model works better. For minimisation of cost as well as minimisation of embodied carbon scenarios, both the models were able to produce optimal mix proportions with a very

small variation. The optimised SCC mix proportions of ingredients in all the three cases are encouraging and can be suitably used for reducing the number of trial mixtures to achieve the desired properties of SCC in the field.

5 Conclusions

In this research, a combination of machine learning and bio-inspired optimisation algorithms were utilised to determine the optimal mix proportions for concrete with desired compressive strength. The study involved modelling two machine learning algorithms and comparing their efficacy through experimental verification, which demonstrated that both models could accurately predict concrete compressive strengths. Additionally, two bio-inspired hybrid optimisation algorithms were tested and compared to their parent algorithms, concluding that these algorithms provided faster convergence and better robustness.

To determine the mix proportions, the trained models were applied to three different scenarios:

1 maximising compressive strength

2 minimising cost

3 minimising embodied carbon.

The analysis showed that the ANN-based bio-inspired optimisation model performed better in obtaining the mix proportion with maximum compressive strength. However, both models successfully determined the optimal mix proportions in the minimisation of cost and embodied carbon scenarios.

The optimised mix proportions obtained in all three cases were validated and encouraged. This research provides a useful outcome for reducing the number of trial mixtures in the field for obtaining the desired properties of self-compacting concrete. The findings of this work may serve as a starting point for further research, which may involve exploring more supervised machine learning approaches to achieve even better mix designs through the integration of one or more techniques

References

Al-Saffar, A.H. and Al-Mahdi, A.M. (2015a) 'Optimal seismic design of reinforced concrete bridges using cuckoo search algorithm', *Engineering Structures*, Vol. 95, pp.55–63.

Al-Saffar, A.H. and Al-Mahdi, A.M. (2015b) 'Differential evolution optimisation of prestressed concrete bridges', *Engineering Structures*, Vol. 95, pp.245–259.

Amin, M.N., Al-Hashem, M.N., Ahmad, A., Khan, K., Ahmad, W., Qadir, M.G., ... and Al-Ahmad, Q.M. (2022) 'Application of soft-computing methods to evaluate the compressive strength of self-compacting concrete', *Materials*, Vol. 15, No. 21, p.7800.

Azimi-Pour, M., Eskandari-Naddaf, H. and Pakzad, A. (2020) 'Linear and nonlinear SVM prediction for fresh properties and compressive strength of high volume fly ash self-compacting concrete', *Construction and Building Materials*, Vol. 230, p.117021, https://doi.org/10.1016/j. conbuildmat.2019.117021.

Boindala, S.P. and Arunachalam, V. (2020) 'Concrete mix design optimisation using a multi-objective cuckoo search algorithm', in *Soft Computing: Theories and Applications: Proceedings of SoCTA 2018*, pp.119–126, Springer, Singapore.

Chechkin, A.V., Metzler, R., Klafter, J. and Gonchar, V.Y. (2008) 'Introduction to the theory of Lévy flights', *Anomalous Transport*, pp.129–162.

Cheng, J. and Xiong, Y. (2022) 'Parameter control based cuckoo search algorithm for numerical optimization', *Neural Processing Letters*, Vol. 54, No. 4, pp.3173–3200.

Cheng, M-Y., Prayogo, D. and Wu, Y-W. (2014) 'Novel genetic algorithm-based evolutionary support vector machine for optimising high-performance concrete mixture', *Journal of Computing in Civil Engineering*, Vol. 28, No. 4, p.06014003.

Chopra, P., Sharma, R.K., Kumar, M. and Chopra, T. (2018) 'Comparison of machine learning techniques for the prediction of compressive strength of concrete', *Advances in Civil Engineering*.

Chou, J-S., Chiu, C-K., Farfoura, M. and Al-Taharwa, I. (2011) 'Optimising the prediction accuracy of concrete compressive strength based on a comparison of data-mining techniques', *Journal of Computing in Civil Engineering*, Vol. 25, No. 3, pp.242–253.

Deepa, C., Sathiya, K. and Sudha, P. (2010) 'Prediction of the compressive strength of high performance concrete mix using tree based modeling', *International Journal of Computer Applications*, Vol. 6, No. 5, pp.18–24.

Dutta, S., Samui, P. and Kim, D. (2018) 'Comparison of machine learning techniques to predict compressive strength of concrete', *Computers and Concrete*, Vol. 21, No. 4, pp.463–470.

EFNARC (2002) *Guidelines for Self-Compacting Concrete*, Vol. 32, p.34, Association House, London, UK.

EFNARC Technical Committee (1999) *European Specification for Sprayed Concrete*, GUIDELINES for Specifiers and Contractors.

Fu, H. and Wang, X. (2013) 'Optimal design of high-performance concrete mixes using the cuckoo search algorithm', *Journal of Cleaner Production*, Vol. 59, No. 1, pp.204–213.

Gandomi, A.H., Yang, X-S. and Alavi, A.H. (2013) 'Cuckoo search algorithm: a metaheuristic approach to solve structural optimisation problems', *Engineering with Computers*, Vol. 29, No. 1, pp.17–35.

Ghafor, K. (2022) 'Multifunctional models, including an artificial neural network, to predict the compressive strength of self-compacting concrete', *Applied Sciences*, Vol. 12, No. 16, p.8161.

Group, Self-Compacting Concrete European Project (2005) *The European Guidelines for Self-Compacting Concrete: Specification, Production and Use*, International Bureau for Precast Concrete (BIBM).

Gupta, S.M. (2007) 'Support vector machines based modelling of concrete strength', *Int. J. Intel. Technol.*, Vol. 3, No. 1, pp.12–18.

Hargrave, J.S. and Chen, Y. (2013) 'Differential evolution optimisation for seismic design of reinforced concrete structures', *Engineering Structures*, Vol. 51, pp.112–121.

Huang, Q., Paolo, G. and Hurlebaus, S. (2011) 'Predicting concrete compressive strength using ultrasonic pulse velocity and rebound number', *ACI Materials Journal*, Vol. 108, No. 4, p.403.

Lee, J-H. and Yoon, Y-S. (2009) 'Modified harmony search algorithm and neural networks for concrete mix proportion design', *Journal of Computing in Civil Engineering*, Vol. 23, No. 1, pp.57–61.

Mallipeddi, R., Suganthan, P.N., Pan, Q.K. and Tasgetiren, M.F. (2011) 'Differential evolution algorithm with ensemble of parameters and mutation strategies', *Applied Soft Computing*, Vol. 11, No. 2, pp.1679–1696.

Medgar L., Nisbet, M.A. and VanGeem, M.G. (2007) *Life Cycle Inventory of Portland Cement Concrete*, SN3011, Portland Cement Association, Skokie, IL, p.121, Concrete CO_2 Fact Sheet – National Ready Mixed Concrete Association – Marceau.

MPA (2008) *Fact Sheet 18: Embodied CO2e of UK Cement, Additions and Cementitious Material*, The Concrete Centre, Mineral Products Association [online] http://www.concretecentre.com (accessed 16 January 2020).

Nadimi-Shahraki, M.H. and Zamani, H. (2022) 'DMDE: diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimisation', *Expert Systems with Applications*, Vol. 198, p.116895.

Naseri, H., Jahanbakhsh, H., Hosseini, P. and Moghadas Nejad, F. (2020) 'Designing sustainable concrete mixture by developing a new machine learning technique', *J. Clean. Prod.*, Vol. 258, p.120578, https://doi.org/10.1016/j.jclepro.2020.120578.

Peng, H., Zeng, Z., Deng, C. and Wu, Z. (2021) 'Multi-strategy serial cuckoo search algorithm for global optimisation', *Knowledge-Based Systems*, Vol. 214, p.106729.

Pham, A-D., Hoang, N-D. and Nguyen, Q-T. (2016) 'Predicting compressive strength of high-performance concrete using metaheuristic-optimized least squares support vector regression', *Journal of Computing in Civil Engineering*, Vol. 30, No. 3, p.06015002.

Price, K., Storn, R.M. and Lampinen, J.A. (2006) *Differential Evolution: A Practical Approach to Global Optimization,* Springer Science & Business Media.

Purnell, P. and Black, L. (2012) 'Embodied carbon dioxide in concrete: variation with common mix design parameters', *Cement and Concrete Research*, Vol. 42, No. 6, pp.874–877.

Reddy, R.R., Gupta, A. and Singh, R.P. (1993) 'Expert system for optimum design of concrete structures', *Journal of Computing in Civil Engineering*, Vol. 7, No. 2, pp.146–161.

Siddique, R., Aggarwal, P. and Aggarwal, Y. (2011) 'Prediction of compressive strength of self-compacting concrete containing bottom ash using artificial neural networks', *Advances in Engineering Software*, Vol. 42, No. 10, pp.780–786.

Storn, R. and Price, K. (1997) 'Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization*, Vol. 11, No. 4, pp.341–359.

Turner, L.K. and Collins, F.G. (2013) 'Carbon dioxide equivalent (CO2-e) emissions: a comparison between geopolymer and OPC cement concrete', *Construction and Building Materials*, Vol. 43, pp.125–130.

Vasan, A. and Simonovic, S.P. (2010) 'Optimisation of water distribution network design using differential evolution', *Journal of Water Resources Planning and Management*, Vol. 136, No. 2, pp.279–287.

Wang, M., Ma, Y. and Wang, P. (2022) 'Parameter and strategy adaptive differential evolution algorithm based on accompanying evolution', *Information Sciences*, Vol. 607, pp.1136–1157.

Yang, B., Miao, J., Fan, Z., Long, J. and Liu, X. (2018) 'Modified cuckoo search algorithm for the optimal placement of actuators problem', *Applied Soft Computing*, Vol. 67, pp.48–60.

Yang, X-S. (2014) *Nature-Inspired Optimisation Algorithms*, Elsevier, https://doi.org/10.1016/B978-0-12-416743-8.00021-X.

Yang, X-S. and Deb, S. (2009) 'Cuckoo search via Lévy Flights', in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp.210–214, IEEE.

Yeh, I-C. (1999) 'Design of high-performance concrete mixture using neural networks and nonlinear programming', *Journal of Computing in Civil Engineering*, Vol. 13, No. 1, pp.36–42.

Zain, M.F.M. and Abd, S.M. (2009) 'Multiple regression model for compressive strength prediction of high performance concrete', *Journal of Applied Sciences*, Vol. 9, No. 1, pp.155–160.