Chapter 8

Application of artificial intelligence in estimating mining capital expenditure using radial basis function neural network optimized by metaheuristic algorithms

Erkan Topal[a], Nguyen Thi Kim Ngan[b], Xuan-Nam Bui[c,d], and Hoang Nguyen[c,d]

[a]*Department of Mining Engineering and Metallurgical Engineering, WA School of Mines, Curtin University, Bentley, Perth, WA, Australia,* [b]*Faculty of Economics and Business Administration, Hanoi University of Mining and Geology, Hanoi, Vietnam,* [c]*Department of Surface Mining, Mining Faculty, Hanoi University of Mining and Geology, Hanoi, Vietnam,* [d]*Innovations for Sustainable and Responsible Mining (ISRM) Research Group, Hanoi University of Mining and Geology, Hanoi, Vietnam*

1 Introduction

Capital expenditure (CAPEX) of a mining project includes the investments required in fixed assets to bring the project into production. In open-pit mining projects, mining capital expenditure can have a significant impact on the profitability of the investment. Underestimation of CAPEX may cause a delay of the project in construction as well as in the production phase and over estimation of it can decrease the overall value of the project and may cause a delay or abandonment of the project [1]. In traditional cost estimation for a mining project, mining capital expenditure estimation relies on the cost of similar mining operations with adjustment to specific site conditions [2].

Over the years, the capital cost of mining projects has been estimated with a significant deviation from the actual capital cost value and these deviations mostly occurred as an underestimation. Furthermore, previous studies demonstrated that high financial risks occurred for mining companies with inaccurate CAPEX estimation, and they indicated that the true ground of CAPEX is exceeded from 10% to 35% compared to the estimated values [3–6].

Applications of Artificial Intelligence in Mining and Geotechnical Engineering
https://doi.org/10.1016/B978-0-443-18764-3.00015-1

To overcome this problem, artificial intelligence (AI) techniques have been introduced and applied to estimate capital cost estimation, as well as other problems in real life [7–12]. For estimating CAPEX, Nourali and Osanloo [1,2] applied two machine learning algorithms, including a support vector machine (SVM) and classification and regression trees (CART). Their findings revealed that AI techniques are potential methods for forecasting CAPEX with a reliable range of errors. Based on the artificial neural network approach (ANN), Guo et al. [13] demonstrated that ANN was an excellent choice for forecasting CAPEX with high accuracy. Inheriting the study of Guo et al. [13], Zhang et al. [14] developed a deep learning ANN (abbreviated as DNN) and optimized it by the ant colony optimization (ACO), named ACO-DNN, for forecasting mining capital expenditure. Finally, they interpreted that the ACO-DNN can improve the accuracy of the traditional ANN model in terms of CAPEX forecast. Zheng et al. [15] also estimated CAPEX through the production factors using a cascade forward neural network and salp swarm optimization algorithm with an accuracy of 98%.

A review of previous works shows that AI techniques have strong applicability in the prediction of mining capital expenditure with promising results. In this book chapter, the radial basis function neural network model (RBFNN) and four metaheuristic algorithms (i.e., genetic algorithm—GA; particle swarm optimization—PSO; moth-flame optimization—MFO; Harris Hawks optimization—HHO) were developed and applied to estimate CAPEX of a mining project, named as GA-RBFNN, PSO-RBFNN, MFO-RBFNN, and HHO-RBFNN.

2 Methodology

2.1 Radial basis function neural network (RBFNN)

RBFNN is an enhanced version of the MLP (multilayer perceptron neural network) that was introduced by Broomhead and Lowe [16]. It consists of three layers: input, hidden, and output layers. Unlike MLP, the RBFNN uses unsupervised methods to train the network under the linear weights. Firstly, RBFNN uses the K-means clustering algorithm to select the weights randomly [17]. Then, matrix multiplication or gradient descent algorithms are used to calculate the weights between the hidden and output layers [18,19]. Remarkably, it does not use any activation functions during training the network like MLP, and unsupervised processes are used to model the nonlinear relationships of the datasets. The general architecture of RBFNN for predicting CAPEX is illustrated in Fig. 1.

In Fig. 1, the input vectors contain the input variables, including annual mine production (MineAP), million tons; stripping ratio (SR); annual production of the mill (MillAP), thousand tons; reserve mean grade % Cu EQU (RMG) and the mine life (LOM).

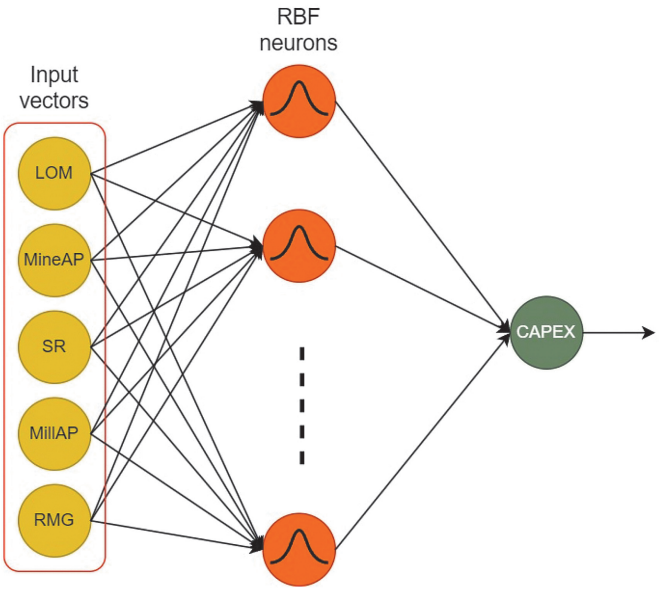


FIG. 1 General architecture of RBFNN for estimating CAPEX.

2.2 Metaheuristic algorithms

Brief principle of GA algorithm

In this chapter, four metaheuristic algorithms were applied to optimize the RBFNN model for estimating CAPEX, including GA, PSO, MFO, and HHO. GA is one of the metaheuristic algorithms which is often used to solve optimization problems. The GA was first presented by Holland [20] and is considered one of the earliest population-based stochastic algorithms. In GA, each solution candidate is represented by a chromosome and each parameter is represented by a gene. The process begins with a random initial population, generated within the lower and upper limits of the parameters. A fitness function is then used to evaluate each individual in the initial population, with a numerical value assigned to indicate its fitness. The best solutions, with higher fitness values, are selected through a selection mechanism for further genetic operations such as crossover and mutation. These operations lead to the renewal of the population and the GA is expected to converge toward the best solution. For more details on GA, please refer to the following reference materials [20–23]. The pseudo-code of the GA algorithm is presented in Fig. 2.

The pseudocode of the Genetic algorithm

```
Set parameters
Choose encode method
Generate the initial population
    current generation = 1
    for i=1 to N do
        for j=1 to L do
            p[i,j]=random number (0,1)
        end for
    end for
Evaluate the fitness of population
    for i=1 to N do
        evaluate p[i]
    end for
end
while i < MaxIteration and Bestfitness < Maxfitness do
    Compute the fitness of each population
    Order the population, compute evaluation value
    Selection
        for i=1 to N do
            parents[i]=tournament (p, tournament size)
        end for
    Crossover
        for i=1 to N
            evaluate p[i]
            for j=1 to L do
                if j<= crossover point
                    offspring[i,j]=parents[i,j]
                    offspring[i+1,j]=pareents[i+1,j]
                else
                    offspring[i,j]=parents[i+1,j]
                    offspring[i+1,j]=pareents[i,j]
                end if
            end for
        end for
    Mutation
        for i=1 to N do
            if (random number < mutation rate)
                mutation(offspring[i])
            end if
        end for
    Update the population for the next generation
end while
Decode the individual with maximum Fitness
return the best solution
```

FIG. 2 Pseudo-code of the GA algorithm.

Brief principle of PSO algorithm

The PSO algorithm mimics the behavior of birds searching for food, and views the solution space of a problem as a foraging ground, which was proposed by Kennedy [24]. In PSO, the solution to the optimization problem is represented by the position of particles in the search space. The algorithm focuses on areas with high-fitness particles by assigning a larger search space probability to these

areas, while a lower search space probability is assigned to particles with low fitness. The algorithm starts by randomly initializing a group of particles to form an initial population, which is distributed in a *d*-dimensional space where *d* is the number of parameters being optimized. The size of the particle swarm affects both the search space and computational complexity, with larger particle swarm offering a more comprehensive search space but increased computational complexity, while smaller particle swarm offers a reduced computational complexity but a limited search space. Generally, a particle swarm size of between 10 and 50 is ideal for most problems, but for complex problems such as multi-dimensional global optimization or multi-objective optimization, a particle swarm size of 100–200 or more might be necessary for better results [25]. The PSO algorithm involves two important concepts: "exploration," which refers to the process of particles constantly searching for new solutions, and "development," which refers to the process of particles finding local optimal solutions near a feasible solution. For more details on PSO, please refer to the following reference materials [26–29]. The pseudo-code of the PSO algorithm is shown in Fig. 3.

Algorithm: Particle swarm optimization (PSO)

```
for each particle i
    for each dimention d
        Initialize position x_id randomly within permissible range
        Initialize velocity v_id randomly within permissible range
    end for
end for
Iteration k = 1
do
    for each particle i
        Calculate fitness value
        if the fitness value is better than p_best_id in history
            Set current fitness value as the p_best_id
        end if
    end for
    Choose the particle having the best fitness value as the g_best_id
    for each particle i
        for each dimention d
            Calculate velocity according to the equation
v_j^{i+1} = w v_j^{(i)} + (c_1 × r_1 × (local best_j − x_j^{(i)})) + (c_2 × r_2 × (global best_j − x_j^{(i)})), v_min ≤ v_j^{(i)} ≤ v_max
            Update particle position according to the equation
x_j^{i+1} = x_j^{(i)} + v_j^{(i+1)}; j = 1, 2, ..., n
        end for
    end for
k = k+1
while maximum iterations or minimum error criteria are not attained
```

FIG. 3 Pseudo-code of the PSO algorithm.

Brief principle of MFO algorithm

The MFO is also a metaheuristic algorithm that simulates the navigation behavior of moths, which fly with a constant angle relative to the moon at night and use an intelligent technique to cover long distances. It is proposed by Mirjalili [30] since 2015 for solving optimization problems. However, they become trapped in a destructive spiral pattern when they encounter artificial lights. This behavior is modeled mathematically with the positions of the moths representing the model parameters and the optimal position being considered the "flame." The MFO algorithm involves a random population, a movement function that explains the moth's exploration of the search space, and model parameters that determine the end of the optimization process. For more details on MFO, please refer to the following reference materials [30–32]. The pseudo-code of the MFO algorithm is shown in Fig. 4.

Algorithm 1: Moth Flame Optimization

Input: int NumMoths, int MaxIter
Initialize Moth Population $M \leftarrow Init()$;
OM $\leftarrow Fitness(M)$;
$t \leftarrow 0$;
while $t < MaxIter$ **do**
  if $t == 1$ **then**
    $F \leftarrow sorted(M)$;
    $OF \leftarrow sorted(OM)$;
  else
    $F \leftarrow sorted(F_{t-1}, M_t)$;
    $OF \leftarrow sorted(OF_{t-1}, OM_t)$;
  end
  for $i \leftarrow 0$ **to** n **do**
    Calculate D for the corresponding Moth and Flame;
    Update Moth(i);
  end
  $t \leftarrow t + 1$;
end
$F \leftarrow sorted(F_{t-1}, M_t)$;
return $F[0]$ *//Best Solution*

FIG. 4 Pseudo-code of the MFO algorithm.

Brief principle of the HHO algorithm

The HHO is a newly developed algorithm by Heidari et al. [33] since 2019 for solving optimization problems, based on the concept of swarm intelligence. It was first introduced by Heidari and colleagues. The algorithm is named after the hunting behavior of Harris Hawks, where different strategies are used to tire the rabbit before capturing it. The mathematical model of this process consists of three phases: exploration, transfer, and exploitation. In the exploration phase, the position of the hawks is determined before the rabbit is found. The transition

from exploration to the exploitation phase models the loss of energy by the rabbit during escape, while the exploitation phase outlines the escape of the rabbit and the hawk's pursuit strategies. For more details on HHO, please refer to the following reference materials [33–35]. The pseudo-code of the HHO algorithm is presented in Fig. 5.

Algorithm 1 The standard HHO algorithm

Input: population size h and Max_iter

Initialize a population of h random hawks $x_i(i = 1, \ldots, h)$
Calculate the fitness of each hawk
x_{rabbit}= best position of minimum fitness
$t = 1$
while ($t \leq Max\_iter$)
  Update E using Eq. (3)
  if ($|E| \geq 1$)
    Update $x\ (t + 1)$ using Eq. (1)
  if ($|E| < 1$)
    if ($p \geq 0.5$ &&$|E| \geq 0.5$)
      Update $x(t + 1)$ using Eq. (4)
    if ($p \geq 0.5$ &&$|E| < 0.5$)
      Update $x(t + 1)$ using Eq. (7)
    if ($p < 0.5$ &&$|E| \geq 0.5$)
      Update $x(t + 1)$ using soft besiege progressive rapid dives
    if ($p < 0.5$ &&$|E| \geq 0.5$)
      Update $x(t + 1)$ using hard besiege progressive rapid dives
  if(fitness($x(t + 1)$) < fitness (x_{rabbit}))
    Update x_{rabbit}= $x(t + 1)$
  $t++$

Output: rabbit position x_{rabbit}

FIG. 5 Pseudo-code of the HHO algorithm.

2.3 Proposing the metaheuristics-based RBFNN models for estimating CAPEX

In this chapter, the RBFNN model will be used to estimate CAPEX under the optimization of four metaheuristic algorithms (MHA), including GA, PSO, MFO, and HHO.

To do this, an initial RBFNN structure and parameters of the algorithms will be designed first. Then, the MHA will be applied to generate sets of weights randomly and the optimization processes then will be activated to select the best one. For metaheuristic algorithms, each population will act as a potential solution, and a fitness function, such as RMSE, MSE, MAPE, or MAE, can be used to evaluate the fitness of solution. In addition, most of the optimization mechanisms of metaheuristic algorithms are based on the sharing and updating positions to gain better performance during optimization progress. Therefore, a maximum number of iterations is necessary to ensure the convergence of the

algorithms. The optimized weights are then added to the designed RBFNN structure and calculate the error of the model in estimating CAPEX. Finally, the best prediction model will be selected with the lowest error. The framework of the MHA-RBFNN models for estimating CAPEX is shown in Fig. 6.

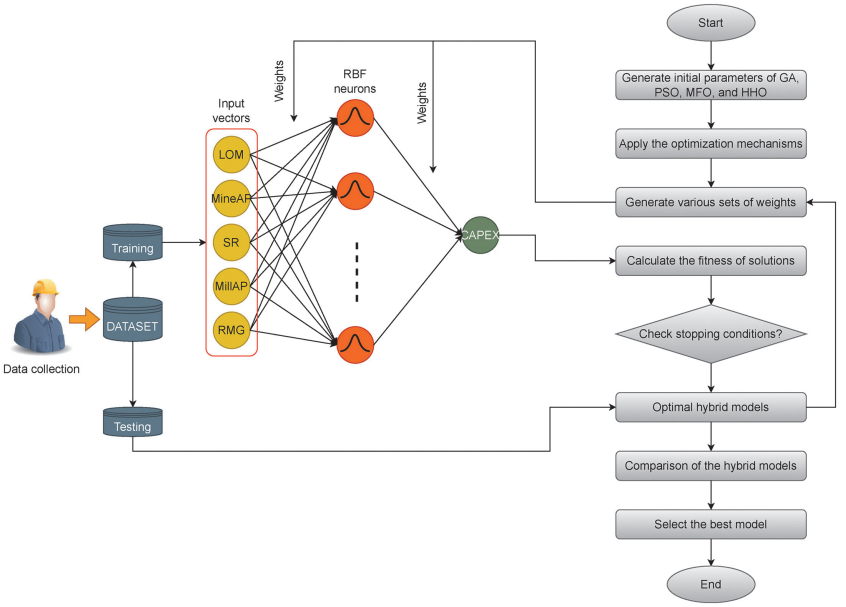


FIG. 6 Framework of the MHA-RBFNN models for estimating CAPEX.

2.4 Performance metrics for evaluation

For assessment of the models' efficiency, three metrics, including RMSE, R^2, and mean absolute percentage error (MAPE) were used as described in Eqs. (1)–(3).

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \widehat{y}_i)^2}{n}} \tag{1}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_{\text{CAPEX}})^2} \tag{2}$$

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \widehat{y}_i|}{y_i} \tag{3}$$

where n is the total number of datasets; y_i stands for the ith actual CAPEX; $\widehat{y}_i$ denotes the ith predicted CAPEX; $\overline{y}_{\text{CAPEX}}$ represents for the mean of actual CAPEX.

3 Data preparation

A dataset from 80 different surface mining operation have been collected and used including annual mine production (MineAP), million tons; stripping ratio (SR); annual production of the mill (MillAP), thousand tons; reserve mean grade % Cu EQU (RMG) and the mine life (LOM) was used. Of those, 70% of the whole dataset was randomly separated to train the mentioned models, and the remaining 30% of the dataset was used for testing the performance once the models were well developed with five-folds cross-validation and three repeats. The five-folds cross-validation technique is simulated in Fig. 7. Note that, the testing dataset was considered as unseen dataset and it has not been used to train the models. In addition, the five-folds cross-validation technique was also applied for the testing dataset to evaluate the predictive models more objectively. The training and testing datasets are summarized in Tables 1 and 2.

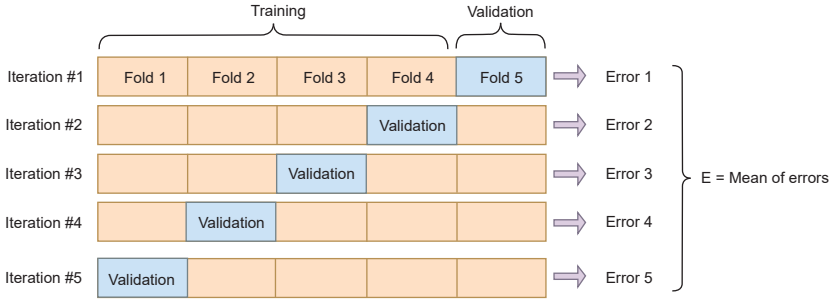


FIG. 7 Simulation of how the dataset is separated by the five-folds cross-validation technique and how to use it.

TABLE 1 Summarize the training dataset used for estimating CAPEX.

MineAP	SR	MillAP	RMG	LOM	CAPEX
Min.: 4.00	Min.: 0.300	Min.: 40.0	Min.: 0.200	Min.: 0.00	Min.: 452
First Qu.: 21.50	First Qu.: 1.843	First Qu.: 372.8	First Qu.: 0.480	First Qu.: 19.75	First Qu.: 1370
Median: 37.00	Median: 3.268	Median: 588.0	Median: 0.940	Median: 30.00	Median: 2668
Mean: 34.79	Mean: 2.829	Mean: 585.9	Mean: 1.345	Mean: 29.02	Mean: 2632
Third Qu.: 48.00	Third Qu.: 3.579	Third Qu.: 795.8	Third Qu.: 2.075	Third Qu.: 40.25	Third Qu.: 3577
Max.: 64.00	Max.: 5.050	Max.: 1215.0	Max.: 3.375	Max.: 54.00	Max.: 6373

TABLE 2 Summarize of the testing dataset used for estimating CAPEX.

MineAP	SR	MillAP	RMG	LOM	CAPEX
Min.: 6.00	Min.: 0.210	Min.: 51.0	Min.: 0.1950	Min.: 9.00	Min.: 406
First Qu.: 16.50	First Qu.: 1.215	First Qu.: 327.2	First Qu.: 0.2988	First Qu.: 17.75	First Qu.: 1043
Median: 24.00	Median: 2.155	Median: 422.5	Median: 0.7925	Median: 22.50	Median: 1818
Mean: 27.21	Mean: 2.087	Mean: 451.9	Mean: 0.9323	Mean: 23.75	Mean: 1910
Third Qu.: 36.00	Third Qu.: 2.999	Third Qu.: 584.2	Third Qu.: 1.0950	Third Qu.: 28.50	Third Qu.: 2381
Max.: 59.00	Max.: 3.716	Max.: 983.0	Max.: 3.1050	Max.: 40.00	Max.: 5369

4 Results and discussions

Before developing the models, the training dataset was normalized using the Min-Max scaling method aimed at transforming the features' scale to the common range between 0 and 1. This helps to prevent features with large scales from dominating the training process and improving the performance and stability of the models for estimating CAPEX. Furthermore, the algorithms' parameters are important and they have a significant effect on the performance of the hybrid models. Therefore, five-folds cross-validation technique was applied during the training of the models to find the best parameters of the models, and the results are shown in Table 3.

TABLE 3 Performance of the MHA-RBFNN models for estimating CAPEX with five-folds cross-validation.

	Training phase			Testing phase		
Model	*RMSE*	R^2	*MAPE*	*RMSE*	R^2	*MAPE*
GA-RBFN	527.529	0.863	0.252	549.451	0.825	0.270
PSO-RBFN	454.669	0.898	0.223	544.993	0.823	0.267
MFO-RBFN	433.318	0.907	0.216	532.229	0.837	0.268
HHO-RBFN	497.538	0.878	0.246	544.639	0.825	0.273

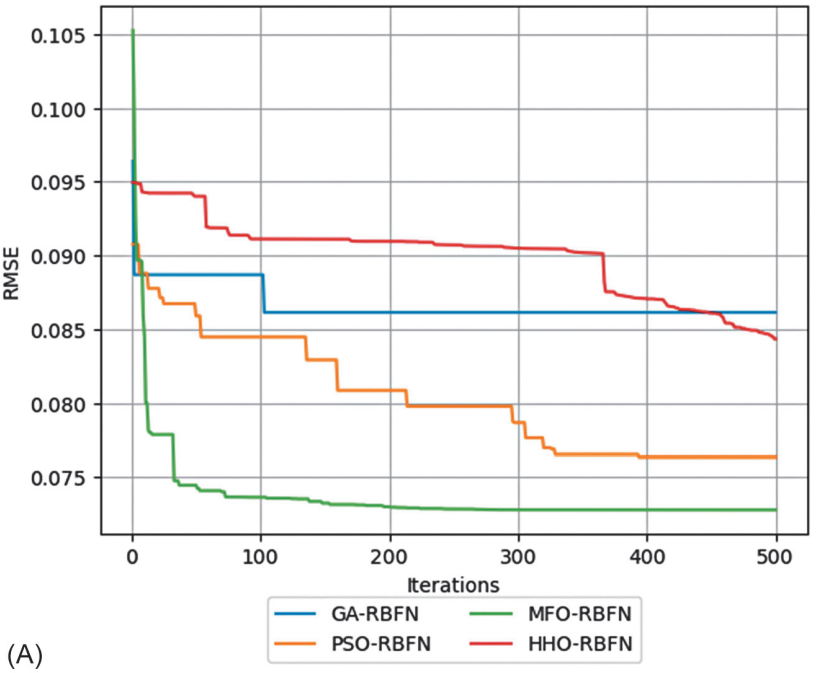
Finally, the best parameters of the GA-RBFNN, PSO-RBFNN, MFO-RBFNN, and HHO-RBFNN models were selected as follows:

- GA-RBFNN model: crossover probability (p_c) = 0.85; mutation probability (p_m) = 0.05; number of populations = 300; iterations = 500.
- PSO-RBFNN model: local coefficient (c_1) = 1.2; global coefficient (c_2) = 1.2; weight min factor (w_{min}) = 0.4; weight max factor (w_{max}) = 0.9; number of populations = 300; iterations = 500.
- MFO-RBFNN and HHO-RBFNN models are non-parameters models and only number of populations and iterations were considered when training these models: number of populations = 300; iterations = 500.

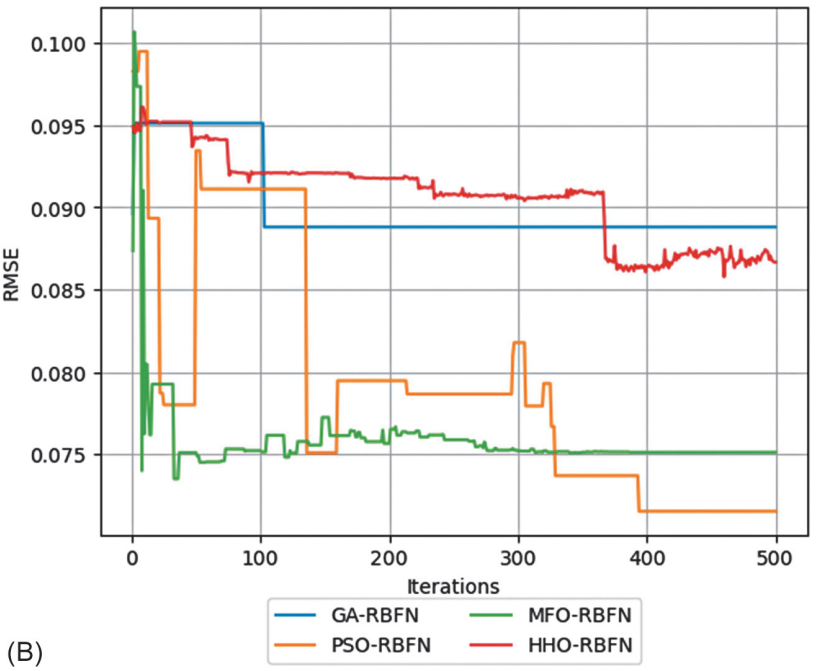
After the best parameters of the training algorithms were selected, the models were re-trained, and their performance in the training and testing phases is shown in Fig. 8.

From the perspective of optimization in Fig. 8, it can be seen that the MFO-RBFNN model was better optimized on the training dataset. However, it is not the best one on the testing dataset as it performed on the training dataset. The PSO-RBFNN model predicted CAPEX better than the MFO-RBFNN model. The remaining models provided lower performances than those of the MFO-RBFNN and PSO-RBFNN models on both the training and testing dataset. A similar problem was found in these models and it is hard to evaluate which model is better than the remaining one. Thus, the performance metrics (introduced in Eqs. 1–3) were calculated based on the actual and estimated CAPEX values, as listed in Table 4. Then, the training and testing performances were evaluated through the bench mark ranking method, as calculated in Table 5 to indicate which model is the best for estimating CAPEX.

Based on the obtained results in Tables 4 and 5, it is easy to see that the MFO-RBFNN model provided the highest benchmark ranking on the datasets with the total ranking of 20. Meanwhile, the PSO-RBFNN model provided slightly lower ranking (i.e., total ranking = 18). Following are the HHO-RBFNN and GA-RBFNN models with a total ranking of 14 and 7, respectively. Taking a closer look at Table 4, we can see that the RMSE values of the MFO-RBFNN, PSO-RBFNN, and HHO-RBFNN models are not too dissimilar; however, the MAPE value of the MFO-RBFNN model is significantly lower than those of the PSO-RBFNN, and HHO-RBFNN models. Furthermore, all metrics of the MFO-RBFNN model on the training dataset are the best compared to the other models. Therefore, we have sufficient evidence to conclude that the MFO-RBFNN model is the best model with the highest level of reliability for predicting CAPEX in this study. Fig. 9 shows the relative error (RE), absolute error and linear relationship between the actual and predicted CAPEX values.



(A)



(B)

FIG. 8 Optimization progress of the MHA-RBFNN models for estimating CAPEX on the training (A) and testing datasets (B).

TABLE 4 Performance of the MHA-RBFNN models for estimating CAPEX after re-trained with the best parameters.

Model	Training phase			Testing phase		
	RMSE	R^2	*MAPE*	*RMSE*	R^2	*MAPE*
PSO-RBFN	452.047	0.900	0.229	423.419	0.862	0.304
HHO-RBFN	508.091	0.874	0.228	443.165	0.849	0.303
MFO-RBFN	430.965	0.909	0.225	444.667	0.848	0.293
GA-RBFN	507.106	0.874	0.237	479.653	0.823	0.305

TABLE 5 Benchmark ranking of the developed models based on the metrics calculated.

Model	Ranking on the metrics						Total ranking
	Training phase			Testing phase			
	RMSE	R^2	MAPE	RMSE	R^2	MAPE	
PSO-RBFN	3	3	2	4	4	2	18
HHO-RBFN	1	1	3	3	3	3	14
MFO-RBFN	4	4	4	2	2	4	20
GA-RBFN	2	1	1	1	1	1	7

5 Conclusions

Mining project requires significant capital investment and accurate estimation of CAPEX significantly impacts the viability of the project. Under estimation of CAPEX may lead to a delay in the development as well as production phases of the project as the real amount may not be freely available on the company's balance sheet. Overestimation of CAPEX can reduce the overall NPV of the project which can make the project uneconomical. In most cases, there is a significant variation between actual and estimated CAPEX values in mining projects and significant underestimation can be observed. One of the important aspects to estimate reliable CAPEX values for a mining project is the data availability as well as the selection of correct factors. In this study, the dataset contains 80 open-pit mining cases with the consideration of mine annual production (MineAP), stripping ratio (SR); mill annual production (MillAP), reserve mean

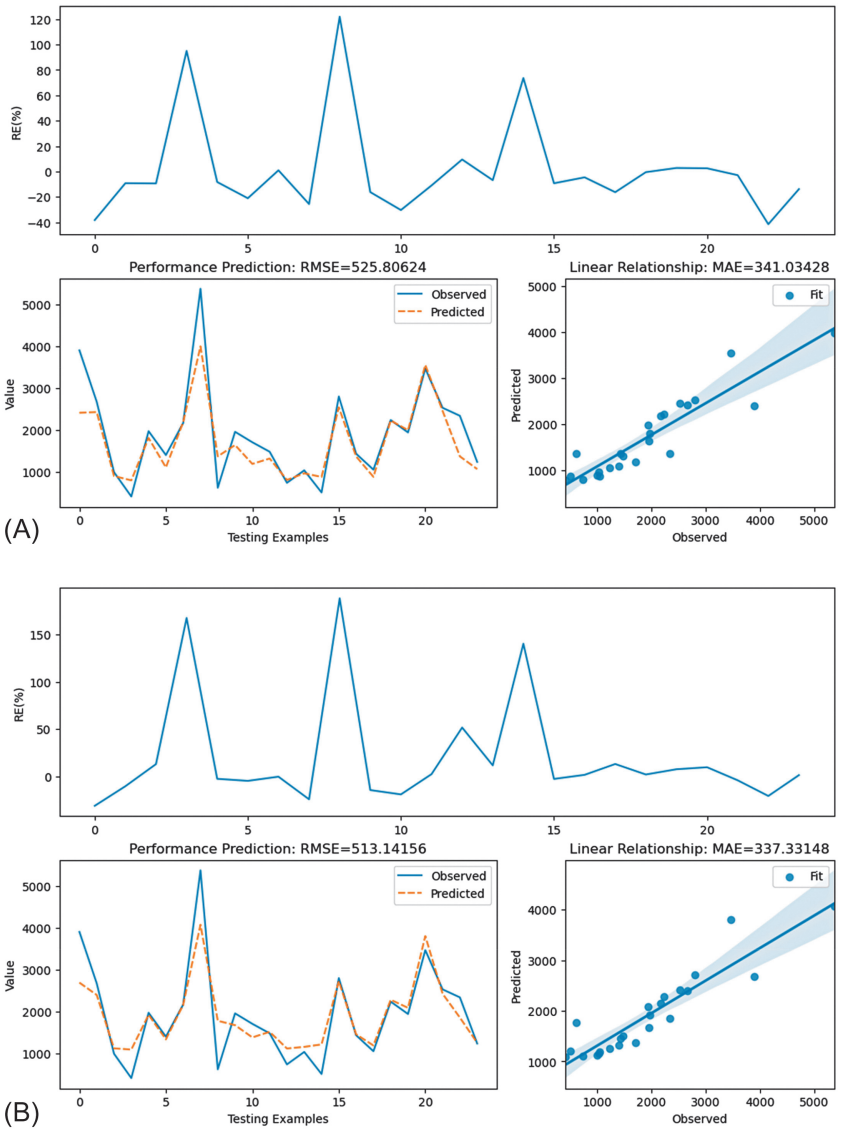


FIG. 9 Performance of the MHA-RBFNN models for estimating CAPEX on the testing datasets: (A) GA-RBFNN model; (B) HHO-RBFNN model;

(Continued)

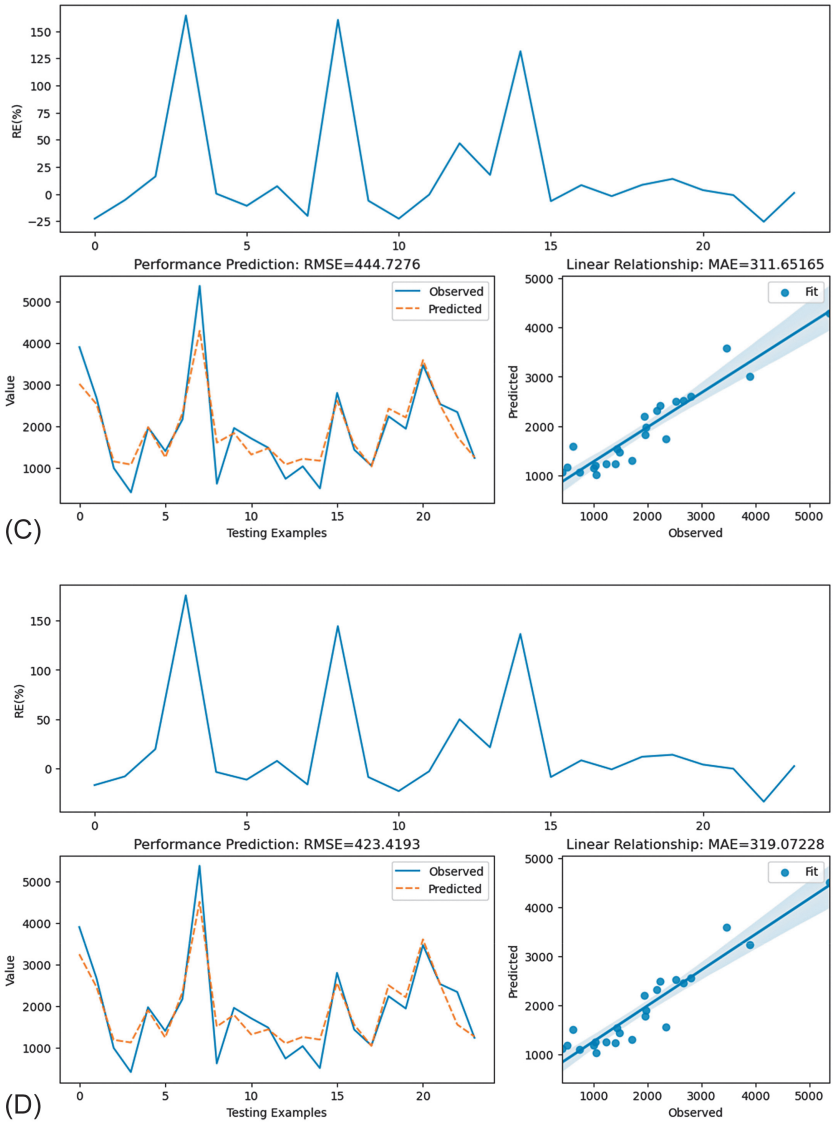


FIG. 9, CONT'D (C) MFO-RBFNN model; and (D) PSO-RBFNN model.

grade % Cu EQU (AG), and the mine life (LOM) was collected and used in the cost prediction models based on AI methods. The radial basis function neural network model (RBFNN) and four metaheuristic algorithms were developed and implemented to estimate CAPEX of a mining project. Application results revealed that RBFNN can be a potential model for estimating the CAPEX of a

mining project. Within this category, the PSO-RBFNN model demonstrated the most accurate estimation, and it should be considered to use in estimating mining CAPEX.

In future studies, more commodity specific data can be collected and different machine learning methods can be utilized and compared.

Acknowledgments

The authors extend heartfelt gratitude to Curtin University (Australia) and Hanoi University of Mining and Geology (Vietnam) for their invaluable cooperation and support throughout this study.

References

[1] H. Nourali, M. Osanloo, Mining capital expenditure estimation using support vector regression (SVR), Resour. Policy 62 (2019) 527–540.
[2] H. Nourali, M. Osanloo, A regression-tree-based model for mining capital expenditure estimation, Int. J. Min. Reclam. Environ. 34 (2) (2020) 88–100.
[3] G. Castle, Feasibility Studies and Other Pre-Project Estimates: How Reliable Are They, Proceedings of the Finance for the Minerals Industry, 1985.
[4] R. Bennet, Technical due diligence requirements for mining project finance, in: Randol at Vancouver 1996 85th Annual Global Mining Opportunities and 2nd Annual Copper Hydromet Rountable, Conference Proceedings, 1996.
[5] S. Thomas, Project development costs—estimates versus reality, in: Mineral Economics and Management Society, Tenth Annual Conference, Houghton, Michigan, 2001.
[6] C. Gypton, How have we done? Eng. Min. J. 203 (1) (2002) 40.
[7] D.H. Vu, H.T. Nguyen, Estimation of shale volume from well logging data using artificial neural network, Tạp chí Khoa học kỹ thuật Mỏ—Địa chất 62 (3) (2021) 46–52.
[8] H. Nguyen, V. Geology, Application of the k-nearest neighbors algorithm for predicting blast-induced ground vibration in open-pit coal mines: a case study, Tạp chí Khoa học kỹ thuật Mỏ—Địa chất 61 (6) (2020) 22–29.
[9] D.N. Le, C.V. Dang, Application of fuzzy-logic to design fuzzy compensation controller for speed control system to reduce vibration of СВШ-250T drilling machine in mining industry, Tạp chí Khoa học kỹ thuật Mỏ—Địa chất 61 (6) (2020) 90–96.
[10] X. Sun, Y. Lei, Research on financial early warning of mining listed companies based on BP neural network model, Resour. Policy 73 (2021) 102223.
[11] G. Franco-Sepúlveda, J.C. Del Rio-Cuervo, M.A. Pachón-Hernández, State of the art about metaheuristics and artificial neural networks applied to open pit mining, Resour. Policy 60 (2019) 125–133.
[12] X. Wang, et al., Production process optimization of metal mines considering economic benefit and resource efficiency using an NSGA-II model, Processes 6 (11) (2018) 228.
[13] H. Guo, et al., Forecasting mining capital expenditure for open-pit mining projects based on artificial neural network approach, Resour. Policy (2019) 101474.
[14] H. Zhang, et al., Developing a novel artificial intelligence model to estimate the capital cost of mining projects using deep neural network-based ant colony optimization algorithm, Resour. Policy 66 (2020) 101604.

[15] X. Zheng, H. Nguyen, X.-N. Bui, Exploring the relation between production factors, ore grades, and life of mine for forecasting mining capital expenditure through a novel cascade forward neural network-based salp swarm optimization model, Resour. Policy 74 (2021) 102300.
[16] D.S. Broomhead, D. Lowe, Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks, Royal Signals and Radar Establishment Malvern, United Kingdom, 1988.
[17] L. Liang, et al., Radial basis function neural network for prediction of medium-frequency sound absorption coefficient of composite structure open-cell aluminum foam, Appl. Acoust. 170 (2020) 107505.
[18] A.P. Markopoulos, S. Georgiopoulos, D.E. Manolakos, On the use of back propagation and radial basis function neural networks in surface roughness prediction, J. Ind. Eng. Int. 12 (3) (2016) 389–400.
[19] M. Mohammadi, et al., A hardware architecture for radial basis function neural network classifier, IEEE Trans. Parallel Distrib. Syst. 29 (3) (2017) 481–495.
[20] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[21] J.H. Holland, Genetic algorithms and adaptation, in: Adaptive Control of Ill-Defined Systems, 1984, pp. 317–333.
[22] S. Sivanandam, et al., Genetic Algorithms, Springer, 2008.
[23] O. Kramer, O. Kramer, Genetic Algorithms, Springer, 2017.
[24] J. Kennedy, Particle swarm optimization, in: C. Sammut, G.I. Webb (Eds.), Encyclopedia of Machine Learning, Springer US, Boston, MA, 2010, pp. 760–766.
[25] B. Su, et al., Sewage treatment system for improving energy efficiency based on particle swarm optimization algorithm, Energy Rep. 8 (2022) 8701–8708.
[26] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.
[27] M. Shanthi, D.K. Meenakshi, P.K. Ramesh, Particle swarm optimization, in: Advances in Swarm Intelligence for Optimizing Problems in Computer Science, Chapman and Hall/CRC, 2018, pp. 115–144.
[28] A. Slowik, Particle swarm optimization, Int. Underw. Syst. Des. (2018).
[29] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft. Comput. 22 (2) (2018) 387–408.
[30] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl. Based Syst. 89 (2015) 228–249.
[31] S.H.H. Mehne, S. Mirjalili, Moth-flame optimization algorithm: theory, literature review, and application in optimal nonlinear feedback control design, in: Nature-Inspired Optimizers, Springer, 2020, pp. 143–166.
[32] M. Shehab, et al., Moth–flame optimization algorithm: variants and applications, Neural Comput. Applic. 32 (14) (2020) 9859–9884.
[33] A.A. Heidari, et al., Harris Hawks optimization: algorithm and applications, Futur. Gener. Comput. Syst. 97 (2019) 849–872.
[34] C. Qu, et al., Harris Hawks optimization with information exchange, App. Math. Model. 84 (2020) 52–75.
[35] M.A. Al-Betar, et al., Survival exploration strategies for Harris Hawks optimizer, Expert Syst. Appl. 168 (2021) 114243.