Solving Fuel-based Unit Commitment Problem Using Improved Binary Bald Eagle Search

Sharaz Ali[1], Mohammed Azmi Al-Betar[1], Mohamed Nasor[1], Mohammed A. Awadallah[2,3*]

[1]Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, P.O.Box 346, Ajman, United Arab Emirates.

[2]Department of Computer Science, Al-Aqsa University, 4051, Gaza Palestine.

[3]Artificial Intelligence Research Center (AIRC), Ajman University, P.O.Box 346, Ajman, United Arab Emirates.

*Corresponding author(s). E-mail(s): ma.awadallah@alaqsa.edu.ps;
Contributing authors: sharaz.ali@ajman.ac.ae; m.albetar@ajman.ac.ae; m.nasor@ajman.ac.ae;

Abstract

The Unit Commitment Problem (UCP) corresponds to the planning of power generation schedules. The objective of the fuel-based unit commitment problem is to determine the optimal schedule of power generators needed to meet the power demand, which also minimizes the total operating cost while adhering to different constraints such as power generation limits, unit startup, and shutdown times. In this paper, four different binary variants of the Bald Eagle Search (BES) algorithm, were introduced, which used two variants using S-shape, U-shape, and V-shape transfer functions. In addition, the best-performing variant (using an S-shape transfer function) was selected and improved further by incorporating two binary operators: swap-window and window-mutation. This variation is labeled Improved Binary Bald Eagle Search (IBBESS2). All five variants of the proposed algorithm were successfully adopted to solve the fuel-based unit commitment problem using seven test cases of 4-, 10-, 20-, 40-, 60-, 80-, and 100-unit. For comparative evaluation, 34 comparative methods from existing literature were compared, in which IBBESS2 achieved competitive scores against other optimization techniques. In other words, the proposed IBBESS2 performs better than all other competitors by achieving the best average scores in 20-, 40-, 60-, 80-, and 100-unit problems. Furthermore, IBBESS2 demonstrated quicker

convergence to an optimal solution than other algorithms, especially in large-scale unit commitment problems. The Friedman statistical test further validates the results, where the proposed IBBESS2 is ranked the best. In conclusion, the proposed IBBESS2 can be considered a powerful method for solving large-scale UCP and other related problems.

Keywords: Swarm intelligence, Bald eagle search, Unit commitment problem, Optimization

1 Introduction

Grid-connected power networks depend on power generation systems to deliver electricity securely and economically [1]. Balancing the power output from many generators to fulfill the demand is a difficult task in the field of power generation. This coordination involves monitoring the various restrictions imposed on individual power systems, such as power-output limits and ramping limits, in addition to various constraints imposed on the power network as a whole, such as network constraints. Figure 1 shows an example of a small power generation network that has to balance the load demand among the power generators to minimize operating costs. Additionally, the expenses of starting up and turning off a power generation unit must be accounted for. The start-up cost of a power generation unit that has been out of commission for a few days, might occasionally be rather substantial. Therefore, to reduce the expenses of the power-producing system, it is essential to plan the power schedule [2].

Unit Commitment Problem (UCP) by definition, is the problem of planning the power generation schedule [1]. Subsequently, the goal of UCP is to obtain an optimal power generation schedule that not only meets the load demands but also satisfies the constraints and most importantly, minimizes the total operating cost [4]. Due to the presence of operating and power generation constraints, UCP is a computationally demanding and complex problem. Therefore, it can be categorized as an NP-hard problem [5]. NP-hard encompasses those problems that might not be solvable in polynomial time [6]. As the number of power generation units and scheduling hours increases, the problem complexity increases exponentially [7]. UCP has many variations, and three of the most commonly used variations in the literature include Fuel-based, Stochastic-based, and Profit-based UCP [8]. Fuel-based UCP focuses on minimizing the cost of operation as an objective function. Stochastic-based UCP incorporates randomness into the objective function because power demand is difficult to predict exactly. Profit-based UCP focuses on maximizing the profit for the energy generation company [8].

2 Literature Review

The UCP has been extensively studied and worked on by optimization research committees, and many techniques have been introduced to tackle this problem. An overview of some of these research works can be seen in Table 1. Zhao et al. [4] used

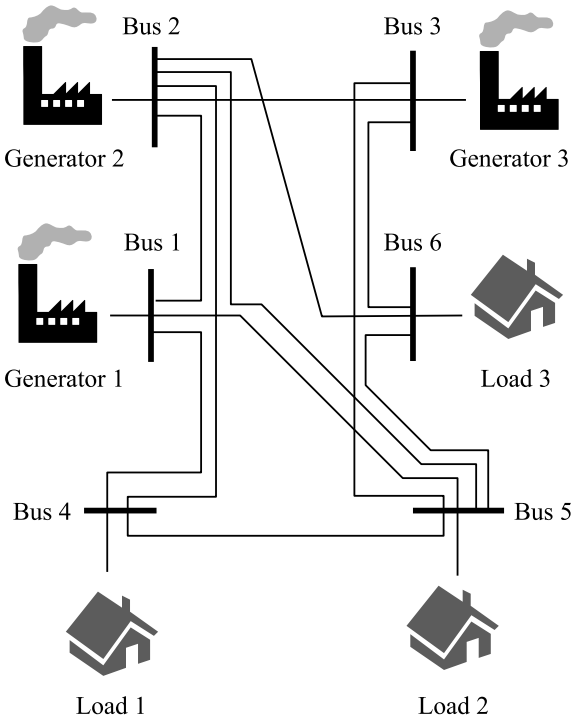


Fig. 1: Power generation load distribution [3].

an improved version of the Binary Cuckoo Search Algorithm (BCSA) to solve the fuel-based version of UCP using five constraints: unit generation limit, system power balance, spinning reserve, ramp-up/down, and minimum downtime constraints. In addition, the authors included a priority list search method in the Improved Binary Cuckoo Search Algorithm (IBCSA) to escape the local optimum. Results showed that IBCSA could dominate other methods when comparing the best, worst, and average costs. Reddy et al. [7] used an Improved Binary Grey Wolf Optimizer (IBGWO) to solve the fuel-based UCP using four constraints: system power balance, spinning reserve, generation limit, and minimum up and downtime constraints. Concerning the average fitness value, an improved solution quality in all test cases was observed. El-Shorbagy et al. [9] used a Binary-real-coded Genetic Algorithm (BRCGA) and K-means clustering technique for solving fuel-based Multi-objective Unit Commitment Problem (MOUCP) using four constraints: system power balance, spinning reserve, generation limit, and minimum up and downtime constraints. Results showed that the combined models of BRCGA and K-means increase the efficiency of solving the MOUCP. Ul Haq et al. [10] used a binary variant model of swarm-inspired polar bear optimization algorithm to solve the fuel-based UCP using five constraints: unit generation limit, system power balance, spinning reserve, ramp-up/down, and minimum downtime constraints. Results demonstrated the superior performance of the

tested algorithms when compared with existing techniques in the literature. Panwar et al. [11] used the Binary Grey Wolf Optimizer (BGWO) to solve the fuel-based UCP using five constraints: unit generation limit, system power balance, spinning reserve, initial status, and minimum up and down-time constraints. BGWO demonstrated competitive results in solving fuel-based UCP on seven different test cases, 4-, 10-, 20-, 40-, 60-, 80-, and 100-unit, compared to existing techniques in the literature. Pan et al. [12] used Binary Fish Migration Optimization (BFMO) and Advanced Binary Fish Migration Optimization (ABFMO) algorithms to solve the fuel-based UCP using four constraints: unit generation limit, system power balance, spinning reserve, and minimum downtime constraints. Results demonstrated that the tested algorithms achieved good performance compared to previous techniques.

Table 1: Examples of existing techniques used to solve the unit commitment problem.

Method used	**Transfer function**	**UCP cases**	**Reference**
Cuckoo Search Algorithm	S-shape	4-unit	Zhao et al. [4]
Grey Wolf Optimizer	S-shape	10-, 20-, 40-, 60-, 80-, 100-unit	Reddy et al. [7]
Binary-real-coded Genetic Algorithm	-	10-, 60-unit	El-Shorbagy et al. [9]
Polar Bear Optimization	S-shape	4-, 5-, 10-, 20-, 40-, 60-, 80-, 100-unit	Ul Haq et al. [10]
Grey Wolf Optimizer	S-shape	4-, 10-, 20-, 40-, 60-, 80-, 100-unit	Panwar et al. [11]
Fish Migration Optimization	S-, V-shape	6-, 7-, 10-, 19-, 20-, 40-, 60-, 80-, 100-unit	Pan et al. [12]
Whale Optimization Algorithm	S-shape	4-, 10-, 20-, 38-, 40-, 54-, 80-, 100-unit	Kumar et al. [13]
Modified Binary Differential Evolution	-	10-, 20-, 40-, 60-, 80-, 100-unit	Dhaliwal et al. [14]
Improved Simulated Annealing Particle Swarm Optimization	S-shape	4-, 10-, 20-unit	Zhai et al. [15]
Hybrid binary differential evolution and binary local search optimizer	S-shape	10-, 40-, 100-unit	Dhaliwal et al. [16]
Artificial Fish Swarm Algorithm	S-shape	10-, 20-, 40-, 60-, 80-, 100-, 200-, 800-, 1000-unit	Zhu et al. [17]

Kumar et al. [13] used the Binary Whale Optimization Algorithm (BWOA) algorithm to solve the fuel-based UCP using five constraints: unit generation limit, system power balance, spinning reserve, ramp-up/down, and minimum downtime constraints. The proposed approach dominated other techniques by reporting lower production costs. Dhaliwal et al. [14] modified Binary Differential Evolution (BDE) algorithm to solve the fuel-based UCP using four constraints: unit generation limit, system power balance, spinning reserve, and minimum downtime constraints. The proposed Modified Binary Differential Evolution Algorithm (MBDEA) was tested on six different systems of 10-, 20-, 40-, 60-, 80-, and 100-units. The proposed methods produced better scores than other competing techniques when solving the fuel-based UCP. Zhai et al. [15] used a two-layer algorithm based on Particle Swarm Optimization (PSO) to solve the fuel-based UCP using four constraints: unit generation limit, system power balance, spinning reserve, and minimum downtime constraints. The highlights of this algorithm, Improved Simulated Annealing Particle Swarm Optimization (ISAPSO),

include its two-layer structure which is reported to help in simplifying the UCP, and results reinforce this claim by showing superior performance of ISAPSO over competing techniques. Dhaliwal et al. [16] introduced a hybrid variant of binary local search optimizer and binary differential evolution to solve multi-objective profit-based unit commitment problems using five constraints: unit generation limit, system power balance, spinning reserve, ramp-up/down, and minimum downtime constraints. The results of the proposed algorithms are reported to dominate other existing techniques in literature to solve the MOUCP.

Zhu et al. [17] introduced an Improved Binary Artificial Fish Swarm Algorithm (IBAFSA) combined with Fast Constraint Processing (FCP) to solve larger-scale fuel-based UCP using five constraints: unit generation limit, system power balance, spinning reserve, ramp-up/down and minimum downtime constraints. Simulations show that the proposed algorithm achieved an excellent score in a reasonable time frame.

Although many researchers have solved the UCP using different meta-heuristic algorithms [4, 7, 10, 11, 12, 13, 18], finding an exact solution to the problem is still an open research domain. Therefore, there is a window to improve upon existing solutions. This is widely known as the No Free Lunch (NFL) theorem [19], and it opens up the possibility that a new meta-heuristic algorithm could be developed to solve a particular optimization problem better than previous algorithms.

The swarm-intelligence algorithm, Bald Eagle Search (BES) was recently introduced by Alsattar et al. [20], and it has several advantages over other meta-heuristic algorithms such as, it is derivative free, sound and complete, flexible, adaptable, easy-to-use. It can strike a suitable balance between exploration and exploitation during the search process. Recently, BES has been successfully and efficiently utilized to solve several types of optimization problems [21, 22, 23, 24].

The present research focused on solving the fuel-based UCP using improved binary versions of the recently introduced BES algorithm [20]. The objective of the fuel-based UCP is to minimize the combined operating expenses of the power generation units through meticulous planning of the power schedule. However, five constraints are enforced on the fuel-based UCP, which makes finding a feasible power schedule more difficult. These constraints include system power constraints, minimum up and down constraints, generator capacity constraints, spinning reserve constraints, and generator initial status constraints. To solve the fuel-based UCP with the constraints, BES was first converted to the binary version, Binary Bald Eagle Search (BBES), using four different transfer functions. These include two variants of S-shape (BBESS1 and BBESS2), one variant of U-shape (BBESU1), and one variant of V-shape (BBESV1) transfer functions. Based on the test results of the four binary variants of BES, BBESS2 was selected. Thereafter, it is improved by incorporating two operators, swap-window and window-mutation, and introducing an improved version of BBESS2 (i.e., IBBESS2).

In summary, the contributions of the current research to the body of existing knowledge are as follows:

- Developing binary variations of the recently introduced meta-heuristic algorithm, BES, using different transfer functions such as two S-, one U- and one V-shape

transfer functions to solve fuel-based UCP on seven test cases, 4-, 10-, 20-, 40-, 60-, 80-, 100-unit.
- Improving the performance of the best-performed binary variations of BES by incorporating operators such as swap-window and window-mutation operators. The improved version is called IBBESS2.
- Using Friedman's and Holm's tests to determine the statistical significance of the proposed algorithm, IBBESS2.

The results of the proposed versions show that the algorithm can match competitively against existing techniques when applied to the unit commitment problem. A comparison of existing methods in solving UCP is conducted, and the results are compared with the performance of many other meta-heuristic algorithms from Panwar et al. [11].

This paper is spread over six sections. Section 3 delves deeper into the problem formulation, detailing the fuel-based UCP, including the objective function and the constraints involved. Section 4 provides the basic foundation theory of the BES algorithm. Moreover, the binary conversion of BES and the improved version, IBBESS2, are also detailed extensively. Section 5 details the proposed technique's implementation and how it is adapted to solve the UCP. Section 6 focuses on the results and provides a comparison with previous techniques. Finally, Section 7 ends with the conclusion and summarizes the entire research achievements with possible future research suggestions.

3 Fuel-based Unit Commitment Problem Formulation

The fuel-based UCP minimizes operating costs when planning the unit commitment schedule. Primarily, three costs have to be taken into consideration: fuel cost, start-up cost, and shut-down cost. An overview of the fuel-based UCP can be viewed in Figure 2. A detailed breakdown of these costs is provided in the following sections.

3.1 Objective Function

The objective of the fuel-based UCP is to minimize the total operating cost by meticulously planning the power schedule of all generating units within the operating hours while also satisfying the constraints. The objective function of the fuel-based UCP can be mathematically formulated as:

$$\text{Min} \sum_{t=1}^{T} \sum_{i=0}^{N} F(P_{i,t}) \cdot U_{i,t} + ST \cdot U_{i,t} + SD \cdot U_{i,t} \tag{1}$$

Note that $P_{i,t}$ is the power generated by the unit i at time t, $U_{i,t}$ stands for the status of the unit i at time t (whether the unit is in ON or OFF state, 1 or 0), $F(P_{i,t})$ is the fuel cost of the unit i at time t, $ST \cdot U_{i,t}$ is the start-up cost and $SD \cdot U_{i,t}$ is the shut-down cost. Equation 1 shows the three primary costs that need to be minimized. The first is the fuel cost ($F(P_{i,t}) \cdot U_{i,t}$) followed by the start-up cost ($ST \cdot U_{i,t}$) and

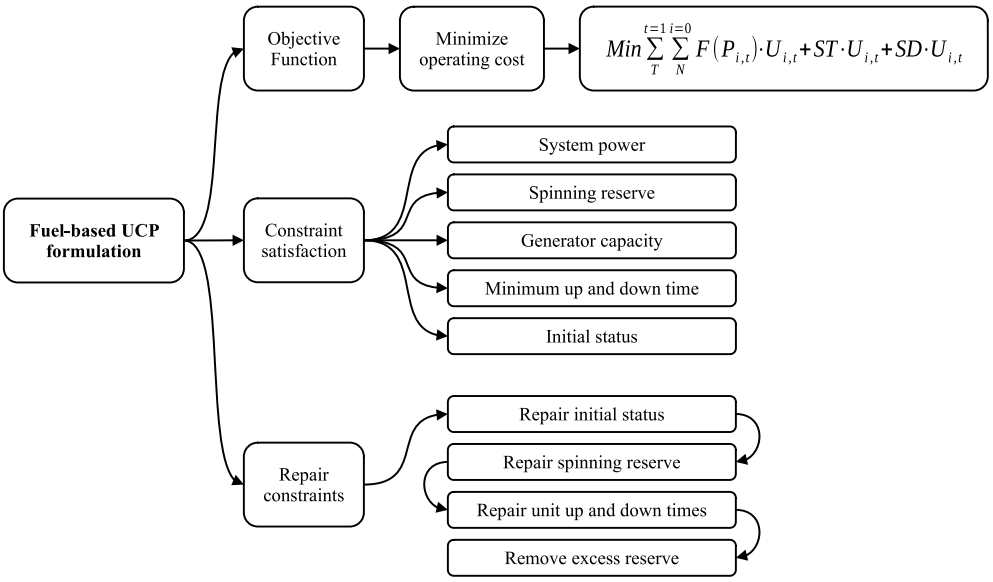


Fig. 2: Overview of the fuel-based UCP formulation.

shut-down costs ($SD \cdot U_{i,t}$) respectively [25]. These costs will be thoroughly formulated as follows:

3.1.1 Fuel Cost

Since fuel is the necessary driving force for thermal generation units, it adds to the operating cost of these generation units. The fuel cost can be calculated using Equation 2:

$$F(P_{i,t}) = a_i + b_i \cdot P_{i,t} + c_i \cdot P_{i,t}^2 \tag{2}$$

Note that a_i, b_i, and c_i represent the fuel cost coefficients of the power generation unit i [25].

3.1.2 Start-up Cost

Thermal generation units have start-up costs associated with them because changes in the boiler temperature take time and are not instantaneous. The start-up cost depends on the number of hours the thermal generation unit was in a decommissioned state (OFF state). If the unit was in a decommissioned state for more than or equal to the cold-start hours, then the cold-start cost is considered, else the hot-start cost is considered in the calculations for the start-up cost. Therefore, the start-up cost of the unit i can be expressed as follows:

$$ST_i = \begin{cases} HS_i, & T^t_{i,\text{off}} \leq T_{i,\text{down}} + T_{i,\text{cold}} \\ CS_i, & T^t_{i,\text{off}} \geq T_{i,\text{down}} + T_{i,\text{cold}} \end{cases} \tag{3}$$

Note that HS_i is the hot start-up cost, and CS_i is the cold start-up cost of unit i [25].

3.1.3 Shut-down Cost

As discussed previously, thermal generation units contain boilers whose temperatures take time to change according to the heat provided by the fuel and in reality, this process is not instantaneous. The cost associated with shutting down a unit is evaluated as a constant and can usually be neglected in the total operating cost calculations. The mathematical formulation for the shut-down cost is:

$$SD \cdot U_{i,t} = KP_{i,t} \tag{4}$$

Note that K is the incremental shut-down cost [25].

3.2 UCP Constraints

UCP contains five constraints such as power generation limits for each unit, meeting load demand, maintaining a set percentage of load demand as a spinning reserve in cases of power fluctuations, adhering to the minimum up and down times for each generating unit, and checking the initial status of generators before operating schedule. The following constraints are used when solving the UCP:

3.2.1 System Power/Load Balance Constraint

The first and most important of the constraints is the system load balance constraint. The total power generated by the generator units must meet or exceed the load demand for any given hour. Mathematically, this can be written as:

$$\sum_{i=1}^{N} P_{i,t} U_{i,t} \geq P_{D,t}, \quad for: \quad t = 1, 2, \ldots, T \tag{5}$$

Note that $P_{D,t}$ is the system load demand at time t.

3.2.2 System Spinning Reserve Constraint

While meeting the load demand is crucial, it is also important to keep some spinning reserve (usually 5–10% of the load demand) in case of power fluctuations. Mathematically, this is expressed as:

$$\sum_{i=1}^{N} P_{i,\max} U_{i,t} \geq P_{D,t} + P_{R,t} \tag{6}$$

Note that $P_{R,t}$ is the spinning reserve at time t.

3.2.3 Generator Capacity Constraint

The power generation units have constraints on how much power they can produce, and also they have a minimum power limit, below which they cannot function properly. These constraints can be expressed as:

$$P_{i,\min} \leq P_i \leq P_{i,\max}, \quad \text{where } \ i = 1, 2, \ldots, n \tag{7}$$

Note that $P_{i,\max}$ is the maximum power output, and $P_{i,\min}$ is the minimum power output of generator i.

3.2.4 Minimum Up and Down Time Constraints

Thermal power generation units contain boilers that take time to heat up when supplied with energy from fuel and cool down when the energy from fuel is cut off. This implies that once any unit is committed (ON state) it must be committed for a set period before being decommitted (OFF state). Similarly, the reverse is true: the decommitted units must complete a set period before being committed again. This can be mathematically expressed as:

$$U_{i,t} = \begin{cases} 1, & \text{if } \ 1 \leq T_{\text{ON},i,t-1} < M_{\text{UT},i} \\ 0, & \text{if } \ 1 \leq T_{\text{OFF},i,t-1} < M_{\text{DT},i} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Note that $M_{\text{DT},i}$ is the minimum downtime, and $M_{\text{UT},i}$ is the minimum up-time of unit i while $T_{\text{ON},i,t}$ is the number of hours for which unit i was previously committed at hour t.

3.2.5 Initial Status Constraint

Before planning the power schedule, it is important to consider the initial status of the generators I_i, i.e., whether they were in an ON or OFF state and if so for how many hours. This initial status will then constrain the generators' start-up and shut-down times.

4 Proposed Bald Eagle Search Algorithms for Solving UCP

In this paper, the BES has been utilized and adopted to tackle UCP, which is a problem with binary search space. This section describes the original version of BES as established by Alsattar et al. [20]. Thereafter, different transfer functions are introduced in the optimization loop of BES to map the continuous domain to the binary one. This modification yields a new version of BES called Binary Bald Eagle Search (BBES). To further boost the performance of BBES, two relevant operators for the binary domain are also introduced, and the new version is called Improved Binary Bald Eagle Search (IBBES). These stages of improvement are discussed in the following sections.

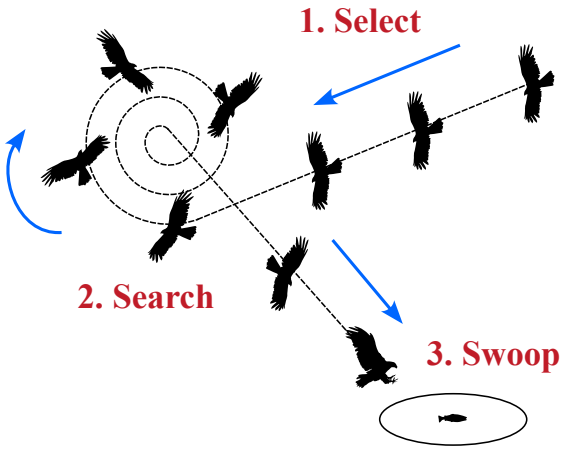


Fig. 3: Overview of the stages in the Bald Eagle Search Algorithm [20].

4.1 The Original Bald Eagle Search Algorithm

The BES algorithm belongs to the category of Swarm Intelligence Algorithms (SIA). The bald eagle, from which BES takes inspiration, is a large bird of prey with distinctive white-colored plumage covering its head area (hence the name 'bald' which stems from older English, meaning white-headed). It feeds primarily on fish, so it lives in regions alongside large water bodies. Blessed with very sharp eyesight, it can spot fish easily, and it strategically hunts fish from the water's surface. Its foraging strategy is interesting and can be segmented into three distinct stages: select, search, and swoop [20], as shown in Figure 3.

4.1.1 Select Stage

During the select stage, bald eagles survey their surroundings and strategically select an area with the most chances of obtaining food. Equation 9 provides a mathematical model for this behavior:

$$P_{\text{new},i} = P_{\text{best}} + \alpha * r(P_{\text{mean}} - P_i) \tag{9}$$

Note that, α controls the positioning changes of the bald eagle and ranges from 1.5 to 2, r is a randomly generated number with a continuous value between 0 and 1. P_{best} represents the best position discovered by the eagles in their prior searches. P_{mean} represents the average value of the positions in prior searches [20].

4.1.2 Search Stage

In the second stage, the bald eagles glide in an accelerated spiral fashion inwards, narrowing down the distance between them and the target. Equation 10 provides a mathematical model for this behavior:

$$P_{\text{new},i} = P_i + y(i) * (P_i - P_{i+1}) + x(i) * (P_i - P_{\text{mean}}) \tag{10}$$

$$x(i) = \frac{xr(i)}{\max(|xr|)} \tag{11}$$

$$y(i) = \frac{yr(i)}{\max(|yr|)} \tag{12}$$

$$xr(i) = r(i) * \sin(\theta(i)) \tag{13}$$

$$yr(i) = r(i) * \cos(\theta(i)) \tag{14}$$

$$\theta(i) = a * \pi * \text{rand} \tag{15}$$

$$r(i) = \theta(i) * R * \text{rand} \tag{16}$$

Note that a represents the corner value between the point search and the center, and ranges between 5 and 10. R represents the number of search cycles, and typically ranges from 0.5 to 2 [20].

4.1.3 Swoop Stage

In the final stage, the bald eagle swoops from its best-known location upon the target. Equation 17 shows this behavior mathematically as:

$$\begin{aligned} P_{\text{new},i} = \ & \text{rand} * P_{\text{best}} \\ & + x1(i) * (P_i - c1 * P_{\text{mean}}) \\ & + y1(i) * (P_i - c2 * P_{\text{best}}) \end{aligned} \tag{17}$$

$$x1(i) = \frac{xr(i)}{\max(|xr|)} \tag{18}$$

$$y1(i) = \frac{yr(i)}{\max(|yr|)} \tag{19}$$

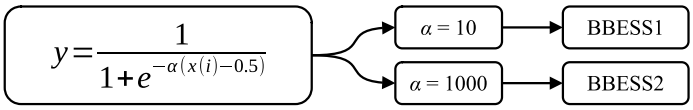
$$xr(i) = r(i) * \sinh[\theta(i)] \tag{20}$$

$$yr(i) = r(i) * \cosh[\theta(i)] \tag{21}$$

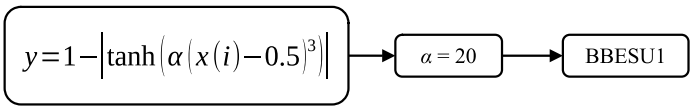
$$\theta(i) = a * \pi * \text{rand} \tag{22}$$

$$r(i) = \theta(i) \tag{23}$$

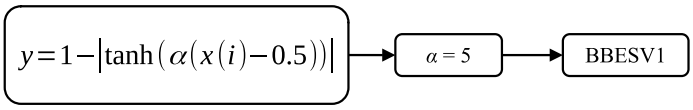
Note that $c1$ and $c2$ range from 1 to 2 [20].



(a) S-shape transfer function.



(b) U-shape transfer function.



(c) V-shape transfer function.

Fig. 4: Overview of the transfer functions tested in the binary bald eagle search algorithm.

4.2 Binary Bald Eagle Search (BBES)

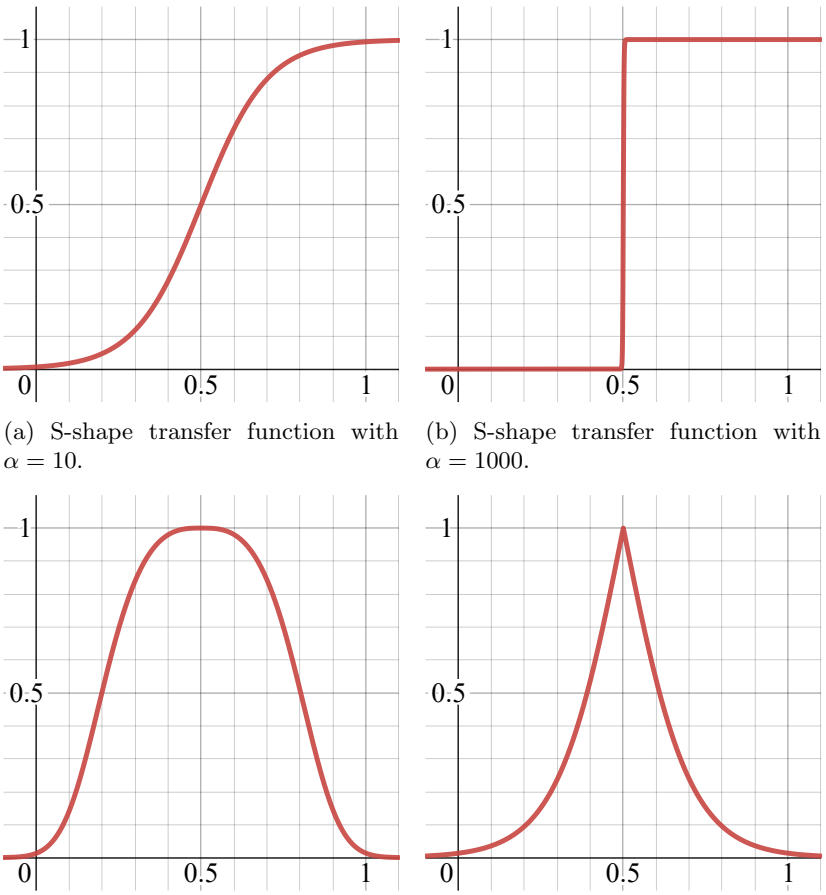
Since the fuel-based UCP has a binary search space, BES cannot be directly used to solve this problem. Therefore, to map the continuous values to binary, several different transfer functions were tested with the BES algorithm, and the best-performing transfer functions were selected for comparison: S-shape, U-shape, and V-shape transfer functions. Due to the NFL theorem [19], no single transfer function can provide the best results on all possible UCP cases, so to provide the audience with a broad range of possible configurations of the binary BES algorithm, four binary versions of the BES algorithm are presented: BBESS1, BBESS2, BBESU1, and BBESV1 as shown in Figure 4.

A transfer function essentially restrains a continuous value in the range, $\mathbb{R}$, to a value between $[0, 1]$. Then, based on a threshold value in the range $[0, 1]$, the value is converted to either 0 or 1. In this study, the threshold value is set at 0.5, and any value equal to or above this threshold is converted to 1 and any value below 0.5 is converted to 0. Equation 24 expresses this conversion mathematically.

$$y = \begin{cases} 1, & \text{if } y \geq 0.5 \\ 0, & \text{if } y < 0.5 \end{cases} \tag{24}$$

4.2.1 S-shape Transfer Function

Equation 25 shows the S-shape transfer function, which has α as a coefficient. Two different variations of S-shape transfer functions were tested, where $\alpha = 10$ and $\alpha = 1000$ (see Figure 5). The former variation is henceforth referred to as S1 and the latter



(a) S-shape transfer function with $\alpha = 10$.

(b) S-shape transfer function with $\alpha = 1000$.

(c) Inverted U-shape transfer function with $\alpha = 20$.

(d) Inverse V-shape transfer function with $\alpha = 5$.

Fig. 5: Transfer functions used in BBES algorithm.

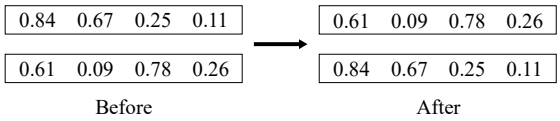
is referred to as S2. When these transfer functions are incorporated into BES, the resulting algorithms are referred to as BBESS1 and BBESS2, respectively.

$$y = \frac{1}{1 + e^{-\alpha(x(i)-0.5)}} \tag{25}$$

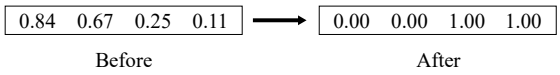
Note that, $\alpha = 10$ and $\alpha = 1000$ and $x(i)$ is the continuous value.

4.2.2 U-shape Transfer Function

Equation 26 shows the inverted U-shape transfer function which has α as a coefficient. One variation of this transfer function was tested, where $\alpha = 20$. This can be visualized using Figure 5. This variation is henceforth referred to as U1 and when incorporated into BES, the resulting algorithm is referred to as BBESU1.



(a) Swap-window operator.



(b) Window-mutation operator.

Fig. 6: Operators used to improve the performance of BBES.

$$y = 1 - \left|\tanh(\alpha(x(i) - 0.5)^3)\right| \tag{26}$$

Note that, $\alpha = 20$ and $x(i)$ is the continuous value.

4.2.3 V-shape Transfer Function

Equation 27 shows the inverted V-shape transfer function, which has α as a coefficient. One variation of this transfer function was tested, where $\alpha = 5$. This can be visualized using Figure 5. This variation is henceforth referred to as V1 and when incorporated into BES, the resulting algorithm is referred to as BBESV1.

$$y = 1 - |\tanh(\alpha(x(i) - 0.5))| \tag{27}$$

Note that, $\alpha = 5$ and $x(i)$ is the continuous value.

4.3 Improved Binary Bald Eagle Search (IBBES)

Although the binary version of BES is sufficient to solve the UCP, it is discovered that the BBES algorithm can get stuck in local solutions, especially during the latter half of the iteration process. Therefore, through extensive experimentation, it is observed that this issue can be alleviated through the incorporation of two operators into the algorithm: swap-window and window-mutation operators. Operators are sophisticated mutation mechanisms that enhance the exploratory behavior of meta-heuristic algorithms by increasing the diversity of individuals in the population. An example of the operators used to improve the performance of the BBES algorithm can be viewed in Figure 6.

4.3.1 Swap-window Operator

The swap-window operator [18] is applied to two randomly selected solutions in the population, p_1 and p_2, and a randomly selected range of positions between $1..D$ in the solutions as shown in Figure 6. The two selected ranges are then swapped with

Algorithm 1 Improved Binary Bald Eagle Search (IBBES) pseudocode.

Randomly initialize point P_i for n points.
Calculate the fitness values of initial Point: $f(P_i)$.
Let $k = 0$
while the termination conditions are not met **do**
  for each point i in the population **do** ▷ **Select stage**
    $P_{\text{new}} = P_{\text{best}} + \alpha * \text{rand} * (P_{\text{mean}} - P_i)$
    $P_{\text{new, binary}} = \text{transfer\_function}(P_{\text{new}})$ ▷ Convert to binary value
    if $f(P_{\text{new}}) < f(P_i)$ **then**
      $P_i = P_{\text{new}}$
    end if
    if $f(P_{\text{new}}) < f(P_{\text{best}})$ **then**
      $P_{\text{best}} = P_{\text{new}}$
    end if
  end for
  for each point i in the population **do** ▷ **Search stage**
    $P_{\text{new}} = P_i + y(i) * (P_i - P_{i+1}) + x(i) * (P_i - P_{\text{mean}})$
    $P_{\text{new, binary}} = transfer\_function(P_{\text{new}})$ ▷ Convert to binary value
    if $f(P_{\text{new}}) < f(P_i)$ **then**
      $P_i = P_{\text{new}}$
    end if
    if $f(P_{\text{new}}) < f(P_{\text{best}})$ **then**
      $P_{\text{best}} = P_{\text{new}}$
    end if
  end for
  for each point i in the population **do** ▷ **Swoop stage**
    $P_{\text{new}} = \text{rand} * P_{\text{best}} + x1(i) * (P_i - c1 * P_{\text{mean}}) + y1(i) * (P_i - c2 * P_{\text{best}})$
    $P_{\text{new, binary}} = transfer\_function(P_{\text{new}})$ ▷ Convert to binary value
    if $f(P_{\text{new}}) < f(P_i)$ **then**
      $P_i = P_{\text{new}}$
    end if
    if $f(P_{\text{new}}) < f(P_{\text{best}})$ **then**
      $P_{\text{best}} = P_{\text{new}}$
    end if
  end for
  $P_i = \text{swap\_window\_operator}(P_i)$ ▷ Swap-window operator
  $P_i = \text{window\_mutation\_operator}(P_i)$ ▷ Window-mutation operator
  Set $k = k + 1$
end while
Return P_{best}

each other. This process is repeated once every iteration of the BBES algorithm. See Figure 6a for visualization of this operator.

4.3.2 Window-mutation Operator

The window-mutation operator [18] is applied to one randomly selected solution in the population, p_1, and a randomly selected range of positions between $1..D$ in the solution as shown in Figure 6. The chosen range of positions in p_1 are then mutated according to the following rule: if $d_i < 0.5$ then $d_i = 1.0$ else $d_i = 0.0$, where d_i is the position value in the selected solution. See Figure 6b for visualization of this operator.

4.4 Time Complexity

The UCP contains two variables that influence the problem's time complexity: the number of operating hours T and the number of units N. Additionally, as seen from the proposed algorithm's pseudocode, IBBESS2 (see Algorithm 1), two variables affect the time complexity, the number of population P and the number of iterations I. As these variables increase in number, the time complexity increases as well. Therefore, the time complexity can be written as $\mathcal{O}(I \times P \times T \times N)$.

5 Implementation of IBBES on the Fuel-based UCP

The IBBES algorithm introduced in Section 4 is utilized to solve the fuel-based UCP. To do so, it is important to discuss the solution representation of the power generation schedule that is going to be modeled, which is discussed next.

5.1 Solution Representation

Power generation units in the system operate on an hourly basis. Therefore, at each given hour, several units are either in an ON or OFF state. The ON and OFF states can be modeled as a binary value, that is, 0 represents OFF and 1 represents ON states respectively. A sequence of these states can be represented in a string of 0s and 1s, and a combination of these strings for each hour can form a schedule of all the units at all hours. For example, consider the sequence of states in Table 8.

This schedule represents the daily schedule of the units to meet the hourly power demands. Therefore, this sequence can be transformed into a single string of 0s and 1s, as seen in Table 9, and we can have different populations or solutions to the UCP problem at hand.

5.2 Solution Initialization

The positions of bald eagles in the population in the proposed algorithms are initialized based on a randomly generated float value in the range $[0, 1]$. Based on the transfer function (see Section 4.2), these values are translated to a binary value of either 0 or 1. During the optimization process, if the position of the bald eagles exceeds 1 or goes below 0, the values are clipped to 1 and 0.

5.3 Repair Constraints

Constraint handling is crucial when solving the UCP. Various techniques exist in the literature to handle constraints in the optimization process [26]. Penalty methods are frequently used to differentiate between feasible and infeasible solutions. Infeasible solutions are discouraged in the population by adding either static or dynamic penalty weights to them, based on the number of constraint violations [27]. Alternatively, in some penalty methods, the sum of the constraint violations is used in a separate objective function such as ϵ-Constrained Approach [28] and Stochastic Ranking Method [29]. In essence, the process of constraint handling ensures that only the feasible solution is selected even if the infeasible solution has a better fitness, i.e., the process makes

Algorithm 2 Unit commitment problem pseudocode.

```
Input load, unit properties, and number of units.
Let constriants_satisfied = true
for each unit i do
    Sort units based on the lowest average fuel cost (Equation 28)    ▷ Create priority list
end for
for each hour t do
    for each unit i do
        U_{i,t} = random 0 or 1                                       ▷ Randomly generate ON/OFF schedule
        if initial status constraint is not satisfied then            ▷ Check constraints
            constriants_satisfied = false
        end if
        if system power constraint is not satisfied then
            constriants_satisfied = false
        end if
        if spinning reserve constraint is not satisfied then
            constriants_satisfied = false
        end if
        if minimum up and down-time constraint is not satisfied then
            constriants_satisfied = false
        end if
    end for
    if constriants_satisfied = false then                             ▷ Repair constraints
        for each unit do
            Repair initial status
            Repair spinning reserve
            Repair unit up and down-times
            Remove excess reserve
        end for
    end if
    Calculate the total cost of the schedule
end for
for each hour do                                                      ▷ Lambda iteration
    Let P_t = P_{R,t}
    for each unit do
        Let λ = 0
        if λ < b_i and U_{i,t} == 1 then
            λ = b_i
        end if
    end for
    while abs(P_t) > 0.001 do                                         ▷ Precision of up to three decimal places
        Let sum_p = 0
        Let sum_c = 0
        for each unit do
            if U_{i,t} == 1 then
                P_{i,t} = (λ − b_i)/(c_i * 2)
                P_{i,t} = min(P_{i,t}, P_{i(max)})
                P_{i,t} = max(P_{i,t}, P_{i(min)})
                sum_p += P_{i,t}
                sum_c += 1/c_i
            end if
        end for
        P_t = P_{R,t} − sum_p
        λ += P_t * 2/sum_c
    end while
end for
Return P_{i,t}
```

sure that there are no constraint violations [30]. Another constraint-handling approach involves repairing infeasible solutions to get feasible solutions, which is used in this paper.

Meta-heuristic algorithms, by design, operate stochastically. This means that the solutions generated by the algorithm, may not satisfy all the constraints. Hence, a repair process must be included to make sure that any violations of the constraints can be fixed. To deal with violations of constraints, a four-step constraint repair process was embedded into the proposed algorithms [13]. However, before the repair process was started, two priority lists of all the units were constructed. The first list contained the units in order of the highest average fuel cost and the other contained the units in order of the lowest average fuel cost when functioning at maximum power output. Recalling Equation 2, the average fuel cost for each unit functioning at maximum power can be written as:

$$fc_i = \frac{a_i}{P_{i(\max)}} + b_i + c_i \cdot P_{i(\max)} \tag{28}$$

Note that fc_i represents the average fuel cost at full load for a unit i. The first step in the repair procedure is to check the spinning reserve constraint violations and based on the priority list, units that could be commissioned are turned on, and those that can be decommissioned are turned off (see Algorithm 4). The priority for commissioning the units is based on the lowest average fuel cost and the priority for decommissioning is the highest average fuel cost. Next, all the unit minimum up and down-time constraint violations are checked and repaired according to the priority list of units (see Algorithm 3). Lastly, any excess reserves were dealt with by decommissioning units (see Algorithm 5) [13]. It is important to note that after applying the swap-window and window-mutation operators, the constraints are re-checked and repaired if needed.

The decommissioning process involves three main steps (as seen in Algorithm 5):

- Sorting units in order of highest average fuel cost using Equation 28, i.e. units with the highest fuel cost will be decommissioned first.
- Checking excess spinning reserve for each hour.
- Decommissioning units based on the priority list calculated before, provided that the removal of the unit does not violate any constraints.

For example, if we take the unit schedule provided in Table 2 from the 4-unit problem and compare the demand to the maximum load for $t = 1$, it can be observed that the max load of 630 MW is more than the demand of 450 MW. To check if any units can be decommissioned, the units will first be sorted based on the highest average fuel cost (as calculated from Equation 28), i.e. units with the highest fuel cost will be decommissioned first. Next, the excess spinning reserve will be calculated which is the demand subtracted from the max load ($630 - 450 = 180$ MW). Finally, the first unit in the priority list that is ON will be selected (in this case it is unit 3 since it has the highest average fuel cost among the other units that are ON). However, before decommissioning the unit all constraints are checked to see if the unit schedule will be feasible (such as minimum up/downtime, initial status and spinning reserve) after the decommissioning process and if no constraints are violated then

Algorithm 3 Pseudocode for minimum up/down-time constraint handling and repair.

for each hour t **do**
  for each unit i **do**
    if $t == 0$ **then**
      ▷ If unit was previously on and is currently on.
      if $I_i > 0$ and $U_{i,t} == 1$ **then**
        $T_{i,t} = I_i + 1$
      ▷ If unit was previously on and is currently off.
      else if $I_i > 0$ and $U_{i,t} == 0$ **then**
        if $\text{abs}(I_i) < M_{\text{UT},i}$ **then**
          $U_{i,t} = 1$
          $T_{i,t} = I_i + 1$
        else
          $T_{i,t} = -1$
        end if
      ▷ If unit was previously off and is currently on.
      else if $I_i < 1$ and $U_{i,t} == 1$ **then**
        if $\text{abs}(I_i) < M_{\text{DT},i}$ **then**
          $U_{i,t} = 0$
          $T_{i,t} = I_i - 1$
        else
          $T_{i,t} = 1$
        end if
      ▷ If unit was previously off and is currently off.
      else if $I_i < 1$ and $U_{i,t} == 0$ **then**
        $T_{i,t} = I_i - 1$
      end if
    else
      ▷ If unit was previously on and is currently on.
      if $T_{i,t-1} > 0$ and $U_{i,t} == 1$ **then**
        $T_{i,t} = T_{i,t-1} + 1$
      ▷ If unit was previously on and is currently off.
      else if $T_{i,t-1} > 0$ and $U_{i,t} == 0$ **then**
        if $\text{abs}(T_{i,t-1}) < M_{\text{UT},i}$ **then**
          $U_{i,t} = 1$
          $T_{i,t} = T_{i,t-1} + 1$
        else
          $T_{i,t} = -1$
        end if
      ▷ If unit was previously off and is currently on.
      else if $T_{i,t-1} < 1$ and $U_{i,t} == 1$ **then**
        if $\text{abs}(T_{i,t-1}) < M_{\text{DT},i}$ **then**
          $U_{i,t} = 0$
          $T_{i,t} = T_{i,t-1} - 1$
        else
          $T_{i,t} = 1$
        end if
      ▷ If unit was previously off and is currently off.
      else if $T_{i,t-1} < 1$ and $U_{i,t} == 0$ **then**
        $T_{i,t} = T_{i,t-1} - 1$
      end if
    end if
  end for
end for

Algorithm 4 Pseudocode for spinning reserve constraint handling and repair.

```
for each unit i do
    Sort units based on the lowest average fuel cost (Equation 28)    ▷ Create priority list
end for
for each hour t do
    if P_{i,max} U_{i,t} < P_{D,t} + P_{R,t} then
        for each unit i do
            if t == 0 then
                                                    ▷ If unit was previously on and is currently off.
                if I_i > 0 and U_{i,t} == 0 then
                    U_{i,t} = 1
                    T_{i,t} = I_i + 1
                    P_{t,max} += P_{i,max}
                                                    ▷ If unit was previously off and is currently off.
                else if I_i < 1 and U_{i,t} == 0 then
                    if abs(I_i) >= M_{DT,i} then
                        U_{i,t} = 1
                        T_{i,t} = 1
                        P_{t,max} += P_{i,max}
                    end if
                end if
            else
                                                    ▷ If unit was previously on and is currently off.
                if T_{i,t-1} > 0 and U_{i,t} == 0 then
                    U_{i,t} = 1
                    T_{i,t} = T_{i,t-1} + 1
                    P_{t,max} += P_{i,max}
                                                    ▷ If unit was previously off and is currently off.
                else if T_{i,t-1} < 1 and U_{i,t} == 0 then
                    if abs(T_{i,t-1}) >= M_{DT,i} then
                        U_{i,t} = 1
                        T_{i,t} = 1
                        P_{t,max} += P_{i,max}
                    else
                        for each hour k from (i + 1 − abs(T_{i,t})) to i do
                            U_{i,k} = 1
                            if k == (i + 1 − abs(T_{i,t})) then
                                T_{i,k} = 1
                            else
                                T_{i,k} = T_{i,k-1} + 1
                            end if
                            P_{k,max} += P_{i,max}
                        end for
                    end if
                end if
            end if
            if P_{i,max} U_{i,t} >= P_{D,t} + P_{R,t} then
                break
            end if
        end for
    end if
end for
```

Algorithm 5 Pseudocode for decommissioning units.

```
for each unit i do
    Sort units based on the highest average fuel cost (Equation 28)   ▷ Create priority list
end for
for each hour t do
    if P_{i,max} U_{i,t} > P_{D,t} + P_{R,t} then
        for each unit i do
            if t == 0 then
                                                  ▷ If unit was previously on and is currently on.
                if I_i > 0 and U_{i,t} == 1 then
                    if abs(I_i) >= M_{UP,i} then
                        if (P_{t,max} − P_{i,max}) >= P_{D,t} + P_{R,t} then
                            U_{i,t} = 0
                            T_{i,t} = −1
                            P_{t,max} −= P_{i,max}
                        end if
                    end if
                                                  ▷ If unit was previously off and is currently on.
                else if I_i < 1 and U_{i,t} == 1 then
                    if (P_{t,max} − P_{i,max}) >= P_{D,t} + P_{R,t} then
                        U_{i,t} = 0
                        T_{i,t} = I_i − 1
                        P_{t,max} −= P_{i,max}
                    end if
                end if
            else
                                                  ▷ If unit was previously on and is currently on.
                if T_{i,t−1} > 0 and U_{i,t} == 1 then
                    if abs(T_{i,t−1}) >= M_{UP,i} then
                        if (P_{t,max} − P_{i,max}) >= P_{D,t} + P_{R,t} then
                            U_{i,t} = 0
                            T_{i,t} = −1
                            P_{t,max} −= P_{i,max}
                        end if
                    end if
                                                  ▷ If unit was previously off and is currently on.
                else if T_{i,t−1} < 1 and U_{i,t} == 1 then
                    if (P_{t,max} − P_{i,max}) >= P_{D,t} + P_{R,t} then
                        U_{i,t} = 0
                        T_{i,t} = T_{i,t−1} − 1
                        P_{t,max} −= P_{i,max}
                    end if
                end if
            end if
        end for
    end if
end for
```

the unit is decommissioned. In the example provided in Table 2, unit 3 can safely be decommissioned at $t = 1$ without violating any constraints while also meeting the spinning reserve requirement.

Table 2: Explaining the decommissioning process with an example from the 4-unit problem.

	Unit				Demand	Max load
	1	2	3	4	(MW)	(MW)
Initial unit status (h)	8	8	-5	-6		
Unit status (t = 1)	1	1	1	0	450	630
Unit status (t = 2)	1	1	1	0	530	630
Max power (MW)	300	250	80	60		
Average fuel cost ($/MWh)	19.62	20.34	23.54	28.00		
Priority	4	3	2	1		

5.4 Power Schedule Calculation

Once the ON / OFF schedule of the power units is determined by the proposed algorithm, then the Lambda-iteration technique [31] is utilized to get the optimal power schedule based on the ON / OFF timings of the generator units. The termination criteria for the lambda-iteration technique is that the difference between the power demand and the total power generated by the units in an hour is less than 0.001 (precision of up to three decimal places). Note that the priority method is used during the power schedule calculation, specifically during the lambda iteration method. The pseudocode detailing the steps in the fuel-based UCP along with the constraint repair process is provided in Algorithm 2.

6 Results and Discussion

Based on the proposed algorithms, BBESS1, BBESS2, BBESU1, BBESV1, and IBBESS2 in Section 4, the fuel-based UCP is solved based on seven distinct cases, 4-, 10-, 20-, 40-, 60-, 80-, 100-unit (Section 6.2) and the results from these experiments are presented and discussed in Section 6.3.

6.1 Experimental Design

The proposed algorithms were programmed in Rust language (Version 1.65.0) [32] and tested on different UCPs (4-, 10-, 20-, 40-, 60-, 80-, and 100-unit) on different operating schedules (8 hours for the 4-unit problem and 24 hours for others). The simulation is carried out on a Ubuntu Linux system (22.04 LTS) with an Intel Core i7 processor and 16 GB of RAM. Additionally, two other algorithms were programmed and tested, Improved Binary Grey Wolf Optimizer S2 (IBGWOS2) and Improved Binary Whale Optimization Algorithm S2 (IBWOAS2), for comparison with the proposed algorithms using the same system and programming language. IBGWOS2 and IBWOAS2 are the improved binary versions of Grey Wolf Optimizer (GWO) [33] and Whale Optimization Algorithm (WOA) [34] using the same S2 transfer function and operators as the proposed IBBESS2. Table 3 shows the individual parameter settings for all the tested algorithms in this paper. Note that the parameter settings listed in Table 3 are the ones that achieved the best results, as suggested by intensive experimentation. Parameter settings for the proposed algorithms BBESS1, BBESS2, BBESU1, BBESV1, and

IBBESS2 are adapted from those suggested by Alsattar et al. [20]. Additionally, the parameter settings for each compared algorithm in the literature provide the best results as suggested by the authors respectively.

Table 3: Parameter settings for the tested algorithms.

Algorithm	**Parameters**
BBESS1, BBESS2, BBESU1, BBESV1 and IBBESS2	$\alpha = 2$ $a = 10$ $R = 2$ Population size = 80 Number of iterations = 300
IBGWOS2	Population size = 80 Number of iterations = 300
IBWOAS2	Population size = 80 Number of iterations = 300

Table 4: 4-unit properties.

Properties	**Unit 1**	**Unit 2**	**Unit 3**	**Unit 4**
$P_{\max}$ (MW)	300	250	80	60
$P_{\min}$ (MW)	75	60	25	20
a (\$/h)	648.74	585.62	213.0	252.0
b (\$/MWh)	16.83	16.95	20.74	23.6
c (\$/MW^{2}h)	0.0021	0.0042	0.0018	0.0034
Min Uptime (h)	4	3	2	1
Min Downtime (h)	5	5	4	1
Hot Start Cost (\$)	500	170	150	0
Cold Start Cost (\$)	1100	400	350	0.02
Cold Start Hours (h)	5	5	4	0
Initial Status (h)	8	8	-5	-6

Table 5: 4-unit hourly demand.

Hour	**Demand**	**Hour**	**Demand**	**Hour**	**Demand**	**Hour**	**Demand**
1	450	3	600	5	400	7	290
2	530	4	540	6	280	8	500

6.2 UCP Datasets Used

To test the proposed algorithm's performance, existing UCP datasets are selected based on the number of units, i.e. 4-, 10-, 20-, 40-, 60-, 80-, and 100 units [11]. The

Table 6: 10-unit properties.

Properties	**Unit 1**	**Unit 2**	**Unit 3**	**Unit 4**	**Unit 5**
$P_{\max}$ (MW)	455	455	130	130	162
$P_{\min}$ (MW)	150	150	20	20	25
a (\$/h)	1000.0	970.0	700.0	680.0	450.0
b (\$/MWh)	16.19	17.26	16.6	16.5	19.7
c (\$/MW^{2}h)	0.00048	0.00031	0.002	0.00211	0.00398
Min Uptime (h)	8	8	5	5	6
Min Downtime (h)	8	8	5	5	6
Hot Start Cost (\$)	4500	5000	550	560	900
Cold Start Cost (\$)	9000	10000	1100	1120	1800
Cold Start Hours (h)	5	5	4	4	4
Initial Status (h)	8	8	-5	-5	-6
Properties	**Unit 6**	**Unit 7**	**Unit 8**	**Unit 9**	**Unit 10**
$P_{\max}$ (MW)	80	85	55	55	55
$P_{\min}$ (MW)	20	25	10	10	10
a (\$/h)	370.0	480.0	660.0	665.0	670.0
b (\$/MWh)	22.26	27.74	25.92	27.27	27.79
c (\$/MW^{2}h)	0.00712	0.00079	0.00413	0.00222	0.00173
Min Uptime (h)	3	3	1	1	1
Min Downtime (h)	3	3	1	1	1
Hot Start Cost (\$)	170	260	30	30	30
Cold Start Cost (\$)	340	520	60	60	60
Cold Start Hours (h)	2	2	0	0	0
Initial Status (h)	-3	-3	-1	-1	-1

Table 7: 10-unit hourly demand.

Hour	**Demand**	**Hour**	**Demand**	**Hour**	**Demand**	**Hour**	**Demand**
1	700	7	1150	13	1400	19	1200
2	750	8	1200	14	1300	20	1400
3	850	9	1300	15	1200	21	1300
4	950	10	1400	16	1050	22	1100
5	1000	11	1450	17	1000	23	900
6	1100	12	1500	18	1100	24	800

Table 8: Unit schedule representation

	Unit 1	**Unit 2**	**Unit 3**	**Unit 4**
Hour 1	0	1	1	1
Hour 2	1	1	0	0
Hour 3	1	1	0	1

Table 9: Solution representation of the UCP

	Hour 1				**Hour 2**				**Hour 3**			
Solution 1	0	1	1	1	1	1	0	0	1	1	0	1
Solution 2	1	0	0	0	1	0	0	0	0	1	0	0
Solution 3	1	1	0	1	0	1	0	1	1	1	1	1
Solution 4	0	0	1	0	1	1	0	0	1	0	0	0

properties of units for the 4-unit problem are given in Table 4 along with the power demand per hour in Table 5, while the properties of the units in the 10-unit problem are given in Table 6 along with the power demand per hour in Table 7. For problems larger than the 10-unit problem, such as the 20-unit problem, the unit properties remain the same, except that the units were now repeated twice. Similarly, for the 40-unit problem, the 10-units were repeated four times, for the 60-unit problem, they were repeated six times, and so forth. For problems larger than the 10-unit problem, the operating schedule window remained unchanged at twenty-four hours as compared with the 10-unit problem except that the demand per hour is multiplied by the increase in the number of units (eg: twice for 20 units, four times for 40-units and so forth).

In addition, the 20-, 40-, 60-, 80- and 100-unit problems utilize the same 10-unit dataset which is duplicated based on the number of required units, i.e. 20-unit dataset contains two 10-unit datasets, the 40-unit dataset contains four 10-unit datasets and so forth. Likewise, the hourly power demand for 20-, 40-, 60-, 80- and 100-unit problems is just a multiplied version of the 10-unit problem based on the number of 10-unit, i.e. 20-unit problem has an hourly power demand that is twice that of the 10-unit problem, while the 40-unit problem has an hourly power demand that is four times that of the 10-unit problem and so forth. Lastly, we assume a 10% spinning reserve in all cases.

6.3 Experimental Results and Discussion

The results from the simulation of the fuel-based UCP on seven different cases, 4-, 10-, 20-, 40-, 60-, 80- and 100-unit, and five tested algorithms, BBESS1, BBESS2, BBESU1, BBESV1 and IBBESS2 and two additional algorithms IBGWOS2, IBWOAS2 (for comparison purpose) are presented in Tables 10-12. Scores from existing techniques are also included for comparison with the proposed algorithms.

Table 10: Comparison of results for the 4-unit problem.

Method	**Best ($)**	**Average ($)**	**Worst ($)**
ILR [35]	75,232	-	-
A.SMP [36]	74,812	74,877	75,166
LRPSO [35]	74,808	-	-
BDE [37]	74,676	-	-
GA [38]	74,675	-	-
HHSRSA [39]	74,476	74,476	74,476
BGWO [11]	**73,933**	**73,933**	**73,933**
BBESS1	74,241	74,241	74,241
BBESS2	74,241	74,241	74,241
BBESU1	74,241	74,241	74,241
BBESV1	74,241	74,241	74,241
IBGWOS2	74,241	74,241	74,241
IBWOAS2	74,241	74,241	74,241
IBBESS2	74,241	74,241	74,241

The 4-unit problem is the easiest to solve, compared with other larger-scale UCPs. As expected, there is an almost negligible difference in the results between

Table 11: Comparison of results for the 10-, 20- and 40-unit problems.

	10-unit problem			20-unit problem			40-unit problem		
Method	**Best ($)**	**Average ($)**	**Worst ($)**	**Best ($)**	**Average ($)**	**Worst ($)**	**Best ($)**	**Average ($)**	**Worst ($)**
LR [18]	565,825	-	-	1,130,660	-	-	2,258,503	-	-
GA [38]	565,825	-	570,032	1,126,243	-	1,132,059	2,251,911	-	2,259,706
EP [40]	564,551	565,352	566,231	1,125,494	1,127,257	1,129,793	2,249,093	2,252,612	2,256,085
LRGA [41]	564,800	-	-	**1,122,622**	-	-	**2,242,178**	-	-
SA [42]	565,828	565,988	566,260	1,126,251	1,127,955	1,129,112	2,250,063	2,252,125	2,254,539
ICGA [43]	566,404	-	-	1,127,244	-	-	2,254,123	-	-
MA [38]	566,686	566,787	567,022	1,128,192	1,128,213	1,128,403	2,249,589	2,249,589	2,249,589
IPPDTM [44]	563,977	-	-	-	-	-	2,247,162	-	-
GRASP [45]	565,825	-	-	1,128,160	-	-	2,259,340	-	-
LMBSI [46]	563,977	-	-	1,123,990	-	-	2,243,708	-	-
DPLR [47]	564,049	-	-	1,128,098	-	-	2,256,195	-	-
GACC [48]	563,977	564,792	565,606	1,125,516	1,127,153	1,128,790	2,249,715	2,253,270	2,256,824
LRPSO [35]	565,869	-	-	1,128,072	-	-	2,251,116	-	-
ELR [47]	563,977	-	-	1,130,660	-	-	-	-	-
EPL [49]	563,977	-	-	1,124,369	-	-	2,246,508	-	-
IPSO [50]	563,954	564,162	564,579	1,125,279	-	1,127,643	2,248,163	-	2,252,117
BFWA [51]	563,977	564,018	564,855	1,124,858	1,124,941	1,125,087	2,248,228	2,248,572	2,248,645
IBPSO [52]	563,977	564,155	565,312	1,125,216	1,125,448	1,125,730	2,248,581	2,248,875	2,249,302
SFLA [53]	564,769	-	-	1,123,261	-	-	2,246,005	-	-
ICA [54]	563,938	-	-	1,124,274	-	-	2,247,078	-	-
IQEA [37]	563,977	563,977	563,977	1,123,890	1,124,320	1,124,504	2,245,151	2,246,026	2,246,701
QEA [55]	563,938	563,969	564,672	1,123,607	1,124,689	1,125,715	2,245,557	2,246,728	2,248,296
SDP [56]	563,938	-	-	1,124,357	-	-	2,243,328	-	-
QBPSO [57]	563,977	563,977	563,977	1,123,297	1,123,981	1,124,294	-	-	-
HHSRSA [39]	563,938	563,965	563,995	1,124,889	1,124,913	1,124,952	2,248,508	2,248,653	2,248,757
RM [58]	563,977	-	-	1,123,990	-	-	-	-	-
GVNS [59]	563,938	-	-	1,123,297	-	-	-	-	-
hGADE/r1 [60]	563,938	564,044	564,283	1,123,386	1,124,436	1,125,045	2,243,724	2,245,582	2,247,130
hGADE/cur1 [60]	563,959	564,088	564,350	1,123,426	1,124,502	1,125,076	2,243,522	2,245,321	2,246,540
Enh-hGADE [60]	563,938	563,997	564,261	1,123,386	1,124,262	1,124,939	2,243,522	2,245,020	2,246,487
BGWO1 [11]	563,977	564,379	565,518	1,125,546	1,126,126	1,127,393	2,252,475	2,257,866	2,263,333
BGWO2 [11]	**563,937**	**563,937**	**563,937**	1,123,297	1,124,215	1,124,379	2,244,701	2,245,145	**2,246,021**
BBESS1	563,977	564,020	564,406	1,123,812	1,124,279	1,124,711	2,246,031	2,247,370	2,248,548
BBESS2	563,977	564,082	564,731	1,124,294	1,124,436	1,125,194	2,246,660	2,247,202	2,248,312
BBESU1	563,977	563,977	563,977	1,123,417	1,124,267	1,124,839	2,245,523	2,247,070	2,247,905
BBESV1	563,977	563,977	563,977	1,123,794	1,124,260	1,124,296	2,246,807	2,247,261	2,248,207
IBGWOS2	563,977	563,977	563,977	1,123,298	1,124,111	1,124,791	2,242,967	2,245,046	2,247,280
IBWOAS2	563,977	563,977	563,977	1,123,316	1,124,165	1,124,711	2,243,203	2,245,640	2,248,235
IBBESS2	563,977	563,977	563,977	1,123,298	**1,123,592**	**1,124,294**	2,242,967	**2,243,755**	2,246,463

the algorithms, as seen in Table 10. Binary Grey Wolf Optimizer (BGWO) algorithm by Panwar et al. [11] displayed the best results out of all the other algorithms, in the categories of best, average, and worst costs. All the proposed variants of BBES along with IBBESS2 came in second place with about 0.41% worse scores. Overall, this test case proved to be insufficient in distinguishing the performance of individual algorithms.

The 10-unit problem is only slightly more difficult to solve than the 4-unit version and likewise, the results for all the algorithms are very similar to each other (Table 10). Again, BGWO1 and BGWO2 algorithms by Panwar et al. [11] scored the best, in the categories of best, average, and worst costs, with the IBBESS2 variant scoring only 0.007% worse, which is an almost negligible difference. Overall, the 10-unit case proved to be as undemanding as the 4-unit case, since all algorithms achieved nearly similar scores.

The 20-unit problem is more interesting than previous cases because it is more challenging. As seen in Table 11, the proposed algorithm, IBBESS2, displayed the best scores, in the categories of average and worst costs. Lagrangian Relaxation Genetic Algorithm (LRGA) by Cheng et al. [41] achieved the best score in the category of the

Table 12: Comparison of results for the 60-, 80- and 100-unit problem.

	60-unit problem			80-unit problem			100-unit problem		
Method	**Best ($)**	**Average ($)**	**Worst ($)**	**Best ($)**	**Average ($)**	**Worst ($)**	**Best ($)**	**Average ($)**	**Worst ($)**
MA [38]	3,370,595	3,370,820	3,371,272	4,494,214	4,494,378	4,494,439	5,616,314	5,616,699	5,616,900
LR [18]	3,394,066	-	-	4,526,022	-	-	5,657,277	-	-
GA [38]	3,376,625	-	3,384,252	4,504,933	-	4,510,129	5,627,437	-	5,637,914
EP [40]	3,371,611	3,376,255	3,381,012	4,498,479	4,505,536	4,512,739	5,623,885	5,633,800	5,639,148
LRGA [41]	3,371,079	-	-	4,501,844	-	-	5,613,127	-	-
SA [42]	-	-	-	4,498,076	4,501,156	4,503,987	5,617,876	5,624,301	5,628,506
ICGA [43]	3,378,108	-	-	4,498,943	-	-	5,630,838	-	-
IPPDTM [44]	3,366,874	-	-	4,490,208	-	-	5,609,782	-	-
GRASP [45]	3,383,184	-	-	4,525,934	-	-	5,668,870	-	-
LMBSI [46]	3,362,918	-	-	4,483,593	-	-	5,602,844	-	-
DPLR [47]	3,384,293	-	-	4,512,391	-	-	5,640,488	-	-
GACC [48]	3,375,065	3,378,976	3,382,886	4,505,614	4,516,731	4,527,847	5,626,514	5,636,522	5,646,529
LRPSO [35]	3,376,407	-	-	4,496,717	-	-	5,623,607	-	-
EPL [49]	3,366,210	-	-	4,489,322	-	-	5,608,440	-	-
IPSO [50]	3,370,979	-	3,379,125	4,495,032	-	4,508,943	5,619,284	-	5,628,506
BFWA [51]	3,367,445	3,367,828	3,367,974	4,491,284	4,492,550	4,493,036	5,610,954	5,612,422	5,612,790
IBPSO [52]	3,367,865	3,368,278	3,368,779	4,491,083	4,491,681	4,492,686	5,610,293	5,611,181	5,612,265
SFLA [53]	3,368,257	-	-	4,503,928	-	-	5,624,526	-	-
ICA [54]	3,371,722	-	-	4,497,919	-	-	5,617,913	-	-
IQEA [37]	3,365,003	3,365,667	**3,366,223**	4,486,963	4,487,985	4,489,286	5,606,022	5,607,561	5,608,525
QEA [55]	3,366,676	3,368,220	3,372,007	4,488,470	4,490,128	4,492,839	5,609,550	5,611,797	5,613,220
SDP [56]	3,363,031	-	-	4,484,365	-	-	**5,602,538**	-	-
hGADE/r1 [60]	3,363,470	3,365,587	3,368,196	4,486,180	4,489,500	4,498,651	5,604,787	5,610,074	5,620,236
hGADE/cur1 [60]	3,362,908	3,364,841	3,367,820	4,485,160	4,487,968	4,494,013	5,606,075	5,610,336	5,620,839
Enh-hGADE [60]	3,362,908	3,364,538	3,367,820	4,485,160	4,487,293	4,489,114	5,604,787	5,607,487	5,612,131
BGWO1 [11]	3,368,934	3,375,221	3,384,306	4,495,173	4,506,362	4,513,026	5,628,975	5,637,659	5,643,899
BGWO2 [11]	3,362,515	3,366,488	3,367,144	**4,483,381**	4,486,676	**4,488,568**	5,604,146	5,607,031	**5,607,723**
BBESS1	3,367,014	3,368,917	3,369,838	4,489,312	4,491,944	4,493,196	5,610,936	5,614,068	5,614,674
BBESS2	3,366,818	3,367,750	3,369,024	4,489,235	4,490,646	4,492,272	5,609,502	5,612,433	5,616,309
BBESU1	3,366,843	3,367,791	3,369,120	4,489,414	4,491,037	4,492,367	5,610,984	5,612,587	5,614,643
BBESV1	3,367,064	3,368,289	3,369,535	4,490,232	4,491,724	4,492,751	5,612,284	5,613,433	5,614,398
IBGWOS2	3,363,703	3,366,513	3,368,526	4,484,754	4,488,301	4,490,400	5,603,036	5,608,283	5,612,895
IBWOAS2	3,363,327	3,365,710	3,368,278	4,484,772	4,488,156	4,491,821	5,604,347	5,609,009	5,614,363
IBBESS2	**3,362,295**	**3,364,034**	3,367,987	4,483,894	**4,485,816**	4,490,283	5,603,584	**5,606,697**	5,610,858

best cost. As expected, the IBBESS2 variant outperformed the non-improved variants, BBESS1, BBESS2, BBESU1, and BBESV1.

The 40-unit problem proved more challenging and although the differences between algorithm performance are still small, it is much more apparent as compared with previous cases (see Table 11). IBBESS2 achieved the best score in the average cost category, while BGWO2 achieved the best score in the category of worst cost, and LRGA [41] achieved the best score in the category of the best cost. The difference in performance between IBBESS2 and the non-improved variants is now noticeably larger and it can be seen that the former easily surpassed the latter variants.

In the 60-unit problem, IBBESS2 scored best in the categories of best and average costs, clearly surpassing non-improved BBES variants (see Table 12). Additionally, BGWO2, IBGWOS2, and IBWOAS2 demonstrated great scores as well.

The 80-unit problem is one of the large-scale UCP and the difference in the performance of the compared algorithms is now very clearly noticeable (see Table 12). In terms of the average cost, IBBESS2 scored the best again, closely followed by BGWO2 and the tested variants, IBGWOS2, and IBWOAS2. However, in terms of the best cost, BGWO2 took first place with IBBESS2 very close to second place.

The largest tested problem was the 100-unit problem and as expected, this case shows the largest difference between the performance of the compared algorithms (see Table 12). In terms of the average cost, IBBESS2 was able to surpass BGWO2, though by a small margin. Unsurprisingly, the non-improved variants, BBESS1, BBESS2,

BBESU1, and BBESV1 can no longer compete with the improved variant, IBBESS2, displaying worse scores in terms of best, average, and worst costs. IBGWOS2 and IBWOAS2 proved to be quite competitive in their performance as well, often coming in close to the scores by IBBESS2.

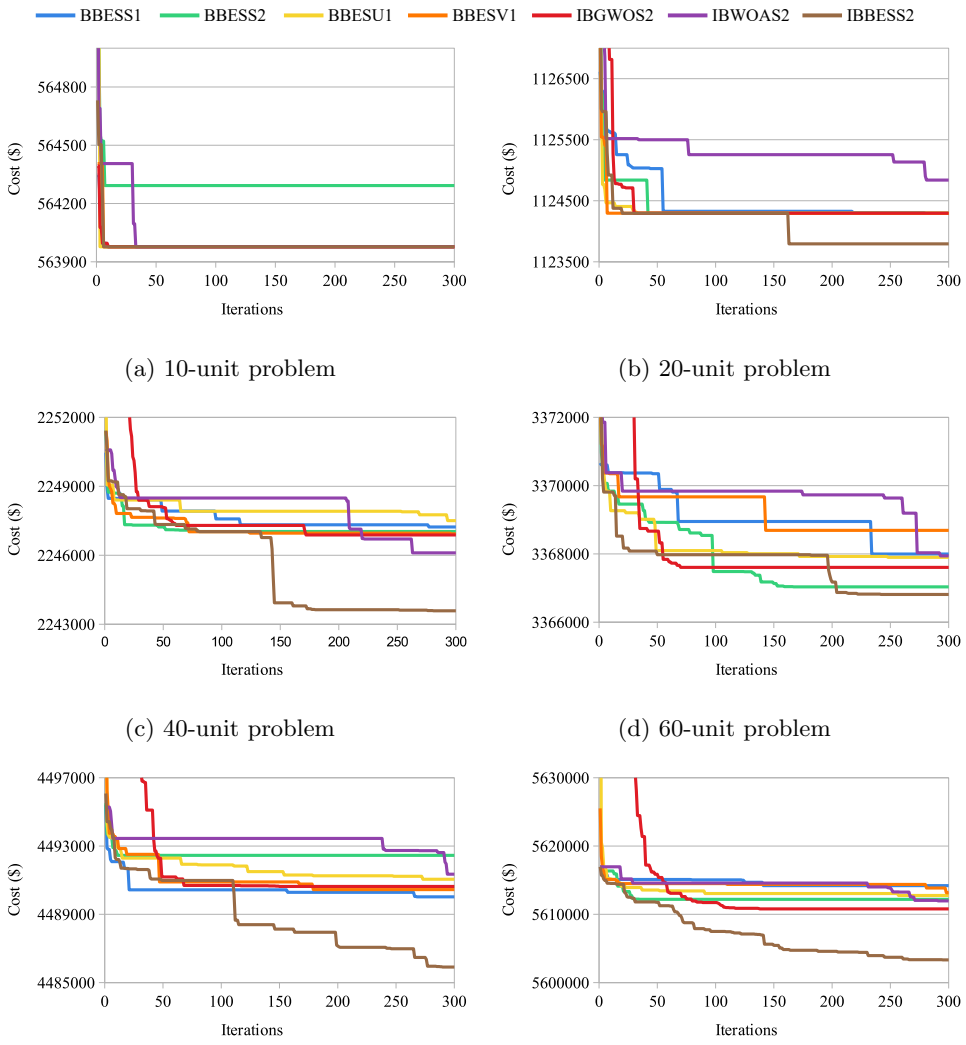
Additionally, the convergence of the proposed algorithms is also recorded and displayed in Figure 7 for all test cases. It can be seen clearly that IBBESS2 demonstrated quicker convergence to an optimal solution as compared with other algorithms, especially in the large-scale UCP. Figure 8 shows box plots for the proposed algorithms as tested on 10-, 20-, 40-, 60-, 80-, and 100-unit problems using 30 different runs. In addition to converging faster, IBBESS2 also records better average scores compared to other tested algorithms. This shows the true potential of using IBBESS2 in solving large-scale UCP as it provides competitive results with faster convergence and better average scores. The best power generation schedules, as generated by IBBESS2, are provided for reference as additional supplementary materials.

Table 13: Friedman's test rankings for IBBESS2 and comparative methods based on average scores.

Algorithm	**Ranking**	**Algorithm**	**Ranking**
IBBESS2	1.75	BBESV1	10.58
BGWO2	3.33	BBESS2	10.92
Enh-hGADE	4.17	BFWA	12.50
IBGWOS2	5.08	BBESS1	12.67
IBWOAS2	5.58	IBPSO	13.17
IQEA	6.08	MA	17.00
hGADE/cur1	7.67	EP	17.50
hGADE/r1	7.92	BGWO1	17.50
QEA	9.17	GACC	18.00
BBESU1	9.42		

6.4 Impact of Proposed Operators on IBBESS2 Computational Times

The proposed algorithm, IBBESS2, incorporates two additional operators as compared with the base algorithm BBESS2. Therefore, to measure the impact of these operators, the average execution times over 30 runs were recorded for the base algorithm, BBESS2, and the improved version, IBBESS2 (Figure 9). Interestingly, the average execution times of IBBESS2 in three test cases (20-, 60- and 80-unit problems) were less than those by BBESS2 while in other test cases (10-, 40- and 100-unit problems) they were more than those by BBESS2. This suggests that the impact of including the operators is minimal and does not significantly increase the average computational time of the proposed algorithm.



(a) 10-unit problem

(b) 20-unit problem

(c) 40-unit problem

(d) 60-unit problem

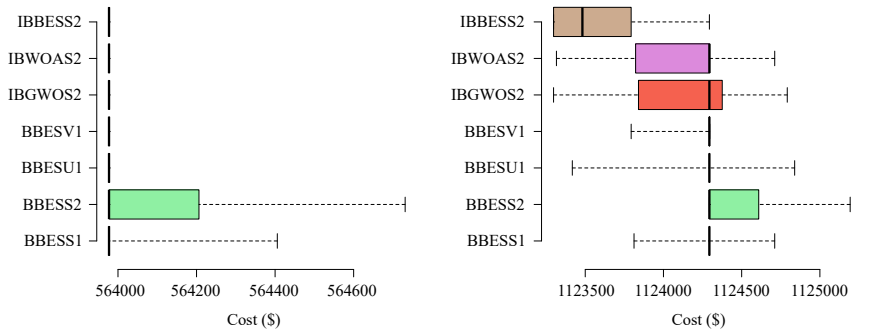
(e) 80-unit problem

(f) 100-unit problem

Fig. 7: Convergence curves for the proposed algorithms for 10-, 20-, 40-, 60-, 80- and 100-unit problems.

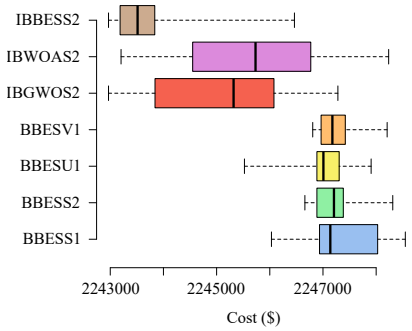
6.5 Sensitivity Analysis

As mentioned in Section 4, two operators were utilized to enhance the performance of the proposed algorithm: swap-window and window-mutation operators. These operators are modified versions of the ones used by Kazarlis et al. [18]. The operator probability was changed from 0 to 1 in increments of 0.1 to observe the effect of the operators. Figure 10 shows the results of this experiment in the form of line graphs. It

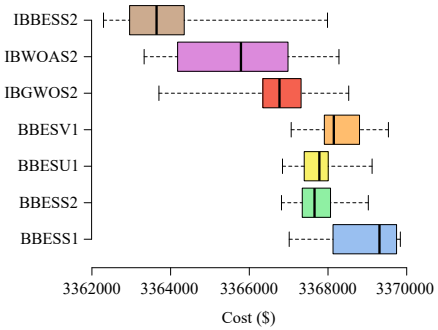


(a) 10-unit problem

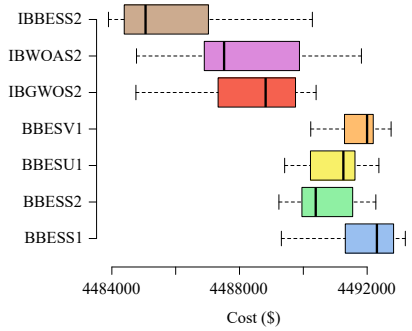
(b) 20-unit problem



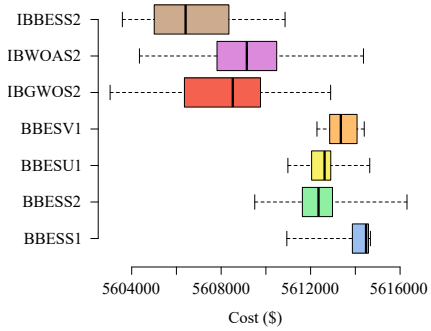
(c) 40-unit problem



(d) 60-unit problem



(e) 80-unit problem



(f) 100-unit problem

Fig. 8: Box plot comparison of the proposed algorithms for 10-, 20-, 40-, 60-, 80- and 100-unit problems after 30 runs.

Table 14: Holm's test results between the control method (IBBESS2) and other comparative methods based on average scores.

Rank	Algorithm	p-value	α/Rank	Hypothesis
18	GACC	5.68E-07	0.002778	Reject
17	EP	1.25E-06	0.002941	Reject
16	BGWO1	1.25E-06	0.003125	Reject
15	MA	2.68E-06	0.003333	Reject
14	IBPSO	4.41E-04	0.003571	Reject
13	BBESS1	7.79E-04	0.003846	Reject
12	BFWA	9.37E-04	0.004167	Reject
11	BBESS2	0.004781	0.004545	Not Reject
10	BBESV1	0.006551	0.005000	Not Reject
9	BBESU1	0.018287	0.005556	Not Reject
8	QEA	0.022442	0.006250	Not Reject
7	hGADE/r1	0.057688	0.007143	Not Reject
6	hGADE/cur1	0.068590	0.008333	Not Reject
5	IQEA	0.182279	0.010000	Not Reject
4	IBWOAS2	0.238050	0.012500	Not Reject
3	IBGWOS2	0.304902	0.016667	Not Reject
2	Enh-hGADE	0.456977	0.025000	Not Reject
1	BGWO2	0.626018	0.050000	Not Reject

Table 15: Friedman's test rankings for IBBESS2 and comparative methods based on best scores.

Algorithm	Ranking	Algorithm	Ranking
BGWO2	3.67	LRGA	16.00
IBBESS2	5.08	BFWA	17.92
Enh-hGADE	6.08	IBPSO	17.92
SDP	6.33	ICA	18.00
IBGWOS2	6.58	SFLA	18.67
IBWOAS2	7.42	IPSO	18.83
hGADE/r1	7.50	BGWO1	22.92
hGADE/cur1	7.58	EP	23.67
LMBSI	7.92	GACC	24.25
QEA	10.67	MA	24.50
IQEA	11.58	LRPSO	26.00
BBESU1	13.75	GA	27.33
EPL	14.42	ICGA	28.33
BBESS1	14.42	DPLR	28.83
BBESS2	14.42	GRASP	30.50
BBESV1	15.92	LR	31.00

can be seen clearly from the figure that as the probability of the operators increased, the performance of the IBBESS2 algorithm improved as well, with the change being sharper between 0 and 0.3 and then a more gradual change from 0.3 to 1 probability. This shows that the operators used, proved advantageous in lowering the overall operating cost for the 40-unit problem, and thus the probability of 1 was selected, i.e., the operators are always used in the improved binary version, IBBESS2.

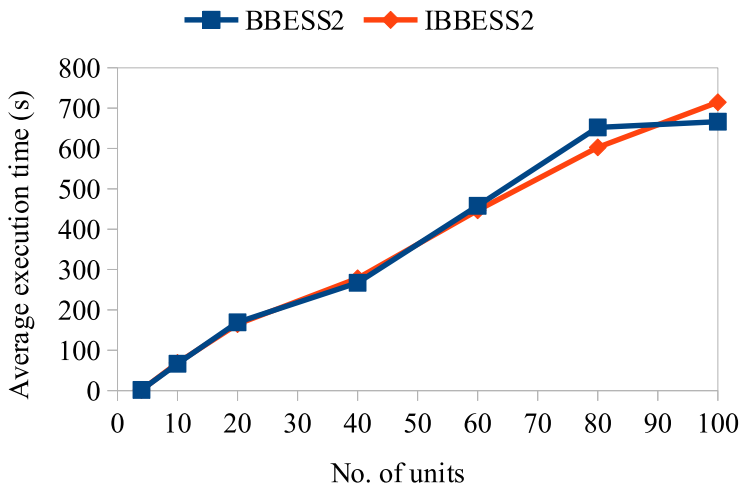


Fig. 9: Variation of the proposed algorithm's average execution time (30 runs) with problem complexity.

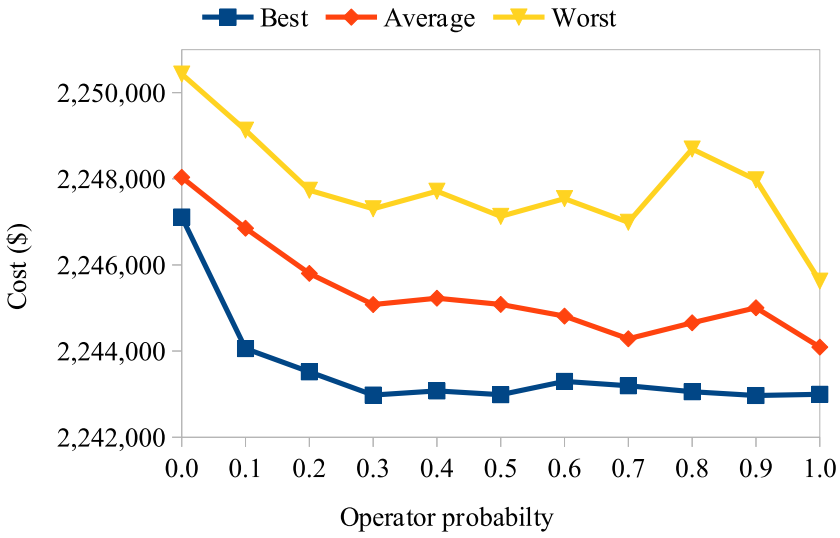


Fig. 10: Variation of best, average, and worst values with operator probability for the 40-unit UCP using IBBESS2.

Table 16: Holm's test results between the control method (BGWO2) and other comparative methods based on best scores.

Rank	Algorithm	p-value	α/Rank	Hypothesis
31	LR	4.49E-07	0.001613	Reject
30	GRASP	7.25E-07	0.001667	Reject
29	DPLR	3.37E-06	0.001724	Reject
28	ICGA	5.25E-06	0.001786	Reject
27	GA	1.24E-05	0.001852	Reject
26	LRPSO	3.73E-05	0.001923	Reject
25	MA	1.20E-04	0.002000	Reject
24	GACC	1.44E-04	0.002083	Reject
23	EP	2.22E-04	0.002174	Reject
22	BGWO1	3.79E-04	0.002273	Reject
21	IPSO	0.005105	0.002381	Not Reject
20	SFLA	0.005613	0.002500	Not Reject
19	ICA	0.008134	0.002632	Not Reject
18	BFWA	0.008511	0.002778	Not Reject
17	IBPSO	0.008511	0.002941	Not Reject
16	LRGA	0.022775	0.003125	Not Reject
15	BBESV1	0.023709	0.003333	Not Reject
14	EPL	0.047161	0.003571	Not Reject
13	BBESS1	0.047161	0.003846	Not Reject
12	BBESS2	0.047161	0.004167	Not Reject
11	BBESU1	0.062637	0.004545	Not Reject
10	IQEA	0.143820	0.005000	Not Reject
9	QEA	0.196198	0.005556	Not Reject
8	LMBSI	0.432625	0.006250	Not Reject
7	hGADE/cur1	0.469580	0.007143	Not Reject
6	hGADE/r1	0.479084	0.008333	Not Reject
5	IBWOAS2	0.488693	0.010000	Not Reject
4	IBGWOS2	0.590214	0.012500	Not Reject
3	SDP	0.622461	0.016667	Not Reject
2	Enh-hGADE	0.655448	0.025000	Not Reject
1	IBBESS2	0.793653	0.050000	Not Reject

6.6 Statistical Analysis

To fully understand the effectiveness of the proposed algorithms and highlight their significant results, two statistical tests, namely Friedman's test and Holm's test, were employed to assess whether the results were statistically significant in solving 10-, 20-, 40-, 60-, 80- and 100-unit problems. Friedman's test determines if there is a significant difference between the algorithm results, assuming the null hypothesis that there is no inconsistency in the accuracy of the compared algorithms. If the p-values obtained from Friedman's test were less than or equal to the predetermined significance level ($\alpha = 0.05$), then the null hypothesis was rejected, indicating significant differences in accuracy among the compared methods. The algorithm with the lowest rank determined by Friedman's test was used as the control for further analysis. Holm's test is often conducted as a post-hoc analysis following Friedman's test to investigate pairwise comparisons between the control method and other algorithms. This test determines

if there are statistically significant differences between the control method and the others.

Table 13 provides a summary of the average ranking results for the proposed IBBESS2 algorithm compared to other methods, obtained through Friedman's test with a significance level of $\alpha = 0.05$ using the results from Tables 11 and 12. The p value resulting from Friedman's test, based on the average scores of IBBESS2 and other algorithms, is computed as $7.38E-11$. The null hypothesis of similar scores is rejected, confirming the presence of statistically significant differences in performance among all compared algorithms. According to the rankings presented in Tables 13 and 15, the proposed IBBESS2 algorithm (the control method) achieved the highest rank of 1.75 in the average scores and the second highest rank of 5.08 in the best scores at $\alpha = 0.05$. IBBESS2 is followed by BGWO2 and Enh-hGADE in second and third place, respectively, outperforming all other competing algorithms when comparing the average scores. This indicates that the proposed algorithm, IBBESS2, ranks among the top two positions compared to the others, highlighting its robustness compared to other promising algorithms in the literature.

Further steps are necessary to determine which algorithms significantly differ from IBBESS2 and which ones are similar in terms of accuracy. To address this, a post-hoc statistical method called Holm's test is utilized to identify the algorithms that perform better or worse than IBBESS2. Holm's test ranks all compared algorithms based on their p-values and compares them to $\alpha/(k-i)$, where k and i represent the degrees of freedom and method number, respectively. The method stops rejecting hypotheses once it reaches a point where all remaining hypotheses are considered acceptable. Holm's test was employed, and the results indicated statistically significant differences between IBBESS2 and the other algorithms. Tables 14 and 16 display the results obtained through Holm's test as a post-hoc statistical analysis for both average and best scores. The results in Tables 13 and 15 reject hypotheses with p-values ≤ 0.00555555. Examining the ranking outcomes in those tables reveals that IBBESS2 consistently achieves better results than other algorithms across the 10-, 20-, 40-, 60-, 80-, and 100-unit problems examined in this study. These findings demonstrate the effectiveness of IBBESS2 in avoiding local optimum solutions and showcase its exploitation capabilities.

7 Conclusion

The Unit Commitment Problem (UCP) involves planning power generation schedules to meet demand while minimizing costs and adhering to constraints. This paper introduced four binary variants of the Bald Eagle Search (BES) algorithm, with S-, U-, and V-shape transfer functions, and improved the best-performing variant with swap-window and window-mutation operators, creating Improved Binary Bald Eagle Search (IBBESS2). These variants were applied to solve fuel-based UCP using 4-, 10-, 20-, 40-, 60-, 80-, and 100-unit cases, with IBBESS2 outperforming others and achieving competitive scores against previous techniques like Binary Grey Wolf Optimizer. IBBESS2 shows quicker convergence, especially in large-scale UCPs, and outperforms other algorithms significantly according to the Friedman statistical test where the

proposed IBBESS2 achieved a rank of 14.98, which is about 32% better than the next highest rank of 22.12 by IBGWOS2. However, IBBESS2 is limited to binary search spaces and does not apply to problems with continuous search spaces. Additionally, the performance of IBBESS2 can be further validated by testing on other well-known systems such as the IEEE 118-bus and the Taiwan Power 38-unit system [61]. Future research could explore its performance on different types of UCP such as the stochastic-based and profit-based types or multi-objective UCP, and its performance could also be improved with hybridization with local-search algorithms such as β-hill climbing [62]. Additionally, IBBESS2 could be applied to various optimization problems beyond UCP such as feature selection [63], university timetabling [64], and nurse restoring [65].

Statements and Declarations

- **Competing Interests.** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- **Ethical and Informed Consent for Data Used.** The data used in this study did not involve ethical and informed consent.
- **Data Availability and Access.** The datasets generated or analyzed during the current study are available from the corresponding author upon reasonable request.

References

[1] Montero, L., Bello, A., & Reneses, J. (2022). A review on the unit commitment problem: approaches, techniques, and resolution methods. *Energies*, *15*(4), 1296.

[2] Conejo, A. J., & Baringo, L. (2018). Unit commitment and economic dispatch. In *Power system operations* (1st ed., pp. 197–232). Springer, Cham, Switzerland.

[3] Ning, C., & You, F. (2019). Data-driven adaptive robust unit commitment under wind power uncertainty: a bayesian nonparametric approach. *IEEE Transactions on Power Systems*, *34*(3), 2409–2418.

[4] Zhao, J., Liu, S., Zhou, M., Guo, X., & Qi, L. (2018). An improved binary cuckoo search algorithm for solving unit commitment problems: methodological description. *IEEE Access*, *6*, 43535–43545.

[5] Bendotti, P., Fouilhoux, P., & Rottner, C. (2019). On the complexity of the unit commitment problem. *Annals of Operations Research*, *274*(1), 119–130.

[6] Roughgarden, T. (2020). What is np-hardness? In *Algorithms illuminated: algorithms for np-hard problems* (1st ed., pp. 1–37). Soundlikeyourself Publishing LLC, New York, NY, USA.

[7] Reddy K., S., Saad Al-Sumaiti, A., Gupta, V., Kumar, R., & Saxena, A. (2019). An improved binary grey wolf optimizer (ibgwo) for unit commitment problem in thermal generation. In *8th international conference on power systems (icps)* (pp. 1–6). IEEE, Jaipur, India.

[8] Abdi, H. (2021). Profit-based unit commitment problem: a review of models, methods , challenges, and future directions. *Renewable and Sustainable Energy Reviews*, *138*, 110504.

[9] El-Shorbagy, M., Mousa, A., & Farag, M. (2019). An intelligent computing technique based on a dynamic-size subpopulations for unit commitment problem. *OPSEARCH*, *56*(3), 911–944.

[10] Ul Haq, S. S., Ahmad, A., Ikram, F., Nawaz, T., Majeed, A., Saddique, M. S., & Sattar, M. K. (2021). A novel binary variant model of swarm inspired polar bear optimization algorithm employed for scalable unit commitment. *International Transactions on Electrical Energy Systems*, *31*(2), e12711.

[11] Panwar, L. K., Reddy, S., Verma, A., Panigrahi, B. K., & Kumar, R. (2018). Binary grey wolf optimizer for large scale unit commitment problem. *Swarm and Evolutionary Computation*, *38*, 251–266.

[12] Pan, J.-S., Hu, P., & Chu, S.-C. (2021). Binary fish migration optimization for solving unit commitment. *Energy*, *226*, 120329.

[13] Kumar, V., & Kumar, D. (2020). Binary whale optimization algorithm and its application to unit commitment problem. *Neural Computing and Applications*, *32*(7), 2095–2123.

[14] Dhaliwal, J. S., & Dhillon, J. S. (2018). Modified binary differential evolution algorithm to solve unit commitment problem. *Electric Power Components and Systems*, *46*(8), 900–918.

[15] Zhai, Y., Liao, X., Mu, N., & Le, J. (2020). A two-layer algorithm based on pso for solving unit commitment problem. *Soft Computing*, *24*(12), 9161–9178.

[16] Dhaliwal, J. S., & Dhillon, J. (2021). A synergy of binary differential evolution and binary local search optimizer to solve multi-objective profit based unit commitment problem. *Applied Soft Computing*, *107*, 107387.

[17] Zhu, Y., & Gao, H. (2020). Improved binary artificial fish swarm algorithm and fast constraint processing for large scale unit commitment. *IEEE Access*, *8*, 152081–152092.

[18] Kazarlis, S. A., Bakirtzis, A., & Petridis, V. (1996). A genetic algorithm solution to the unit commitment problem. *IEEE Transactions on Power Systems*, *11*(1), 83–92.

[19] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

[20] Alsattar, H., Zaidan, A., & Zaidan, B. (2020). Novel meta-heuristic bald eagle search optimisation algorithm. *Artificial Intelligence Review*, *53*(3), 2237–2264.

[21] Ferahtia, S., Rezk, H., Abdelkareem, M. A., & Olabi, A. (2022). Optimal techno-economic energy management strategy for building's microgrids based bald eagle search optimization algorithm. *Applied Energy*, *306*, 118069.

[22] Ferahtia, S., Rezk, H., Djerioui, A., Houari, A., Motahhir, S., & Zeghlache, S. (2023). Modified bald eagle search algorithm for lithium-ion battery model parameters extraction. *ISA Transactions*, *134*, 357–379.

[23] Rezk, H., Ferahtia, S., Sayed, E. T., Abdelkareem, M. A., & Olabi, A. (2022). Robust parameter identification strategy of solid oxide fuel cells using bald eagle search optimization algorithm. *International Journal of Energy Research*, *46*(8), 10535–10552.

[24] Angayarkanni, S., Sivakumar, R., & Ramana Rao, Y. (2021). Hybrid grey wolf: bald eagle search optimized support vector regression for traffic flow forecasting. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 1293–1304.

[25] Muralikrishnan, N., Jebaraj, L., & Rajan, C. C. A. (2020). A comprehensive review on evolutionary optimization techniques applied for unit commitment problem. *IEEE Access*, *8*, 132980–133014.

[26] He, X.-S., Fan, Q.-W., Karamanoglu, M., & Yang, X.-S. (2019). Comparison of constraint-handling techniques for metaheuristic optimization. In *19th international conference on computational science* (pp. 357–366). Springer, Cham, Switzerland.

[27] Powell, D., & Skolnick, M. M. (1993). Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proceedings of the fifth international conference on genetic agorithms* (pp. 424–431). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[28] Takahama, T., & Sakai, S. (2006). Solving constrained optimization problems by the ε constrained particle swarm optimizer with adaptive velocity limit control. In *2006 IEEE conference on cybernetics and intelligent systems* (pp. 1–7). IEEE, Bangkok, Thailand.

[29] Runarsson, T. P., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, *4*(3), 284–294.

[30] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, *186*(2-4), 311–338.

[31] Wood, A. J., Wollenberg, B. F., & Sheblé, G. B. (2013). Economic dispatch of thermal units and methods of solution. In *Power generation, operation, and control* (3rd ed., pp. 63–146). John Wiley & Sons, Hoboken, New Jersey, USA.

[32] Matsakis, N. D., & Klock, F. S. (2014). The rust language. *ACM SIGAda Ada Letters*, *34*(3), 103–104.

[33] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.

[34] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

[35] Sriyanyong, P., & Song, Y. (2005). Unit commitment using particle swarm optimization combined with lagrange relaxation. In *2005 IEEE power engineering society general meeting* (pp. 2752–2759). IEEE, San Francisco, CA, USA.

[36] Khanmohammadi, S., Amiri, M., & Haque, M. T. (2010). A new three-stage method for solving unit commitment problem. *Energy*, *35*(7), 3072–3080.

[37] Jeong, Y.-W., Lee, W.-N., Kim, H.-H., Park, J.-B., & Shin, J.-R. (2009). Thermal unit commitment using binary differential evolution. *Journal of Electrical Engineering and Technology*, *4*(3), 323–329.

[38] Valenzuela, J., & Smith, A. E. (2002). A seeded memetic algorithm for large unit commitment problems. *Journal of Heuristics*, *8*(2), 173–195.

[39] Kamboj, V. K., Bath, S., & Dhillon, J. (2016). Implementation of hybrid harmony search/random search algorithm for single area unit commitment problem. *International Journal of Electrical Power & Energy Systems*, *77*, 228–249.

[40] Juste, K., Kita, H., Tanaka, E., & Hasegawa, J. (1999). An evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power systems*, *14*(4), 1452–1459.

[41] Cheng, C.-P., Liu, C.-W., & Liu, C.-C. (2000). Unit commitment by lagrangian relaxation and genetic algorithms. *IEEE transactions on power systems*, *15*(2), 707–714.

[42] Simopoulos, D. N., Kavatza, S. D., & Vournas, C. D. (2006). Unit commitment by an enhanced simulated annealing algorithm. *IEEE Transactions on Power Systems*, *21*(1), 68–76.

[43] Damousis, I. G., Bakirtzis, A. G., & Dokopoulos, P. S. (2004). A solution to the unit-commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power systems*, *19*(2), 1165–1172.

[44] Chandram, K., Subrahmanyam, N., & Sydulu, M. (2011). Unit commitment by improved pre-prepared power demand table and muller method. *International Journal of Electrical Power & Energy Systems*, *33*(1), 106–114.

[45] Viana, A., de Sousa, J. P., & Matos, M. (2003). Using grasp to solve the unit commitment problem. *Annals of Operations Research*, *120*(1), 117–132.

[46] Silva Jr, I. C., Carneiro Jr, S., de Oliveira, E. J., Pereira, J., Garcia, P. A., & Marcato, A. L. (2008). A lagrangian multiplier based sensitive index to determine the unit commitment of thermal units. *International Journal of Electrical Power & Energy Systems*, *30*(9), 504–510.

[47] Ongsakul, W., & Petcharaks, N. (2004). Unit commitment by enhanced adaptive lagrangian relaxation. *IEEE Transactions on Power Systems*, *19*(1), 620–628.

[48] Senjyu, T., Yamashiro, H., Uezato, K., & Funabashi, T. (2002). A unit commitment problem by using genetic algorithm based on unit characteristic classification. In *2002 IEEE power engineering society winter meeting* (pp. 58–63, Vol. 1). IEEE, New York, NY, USA.

[49] Srinivasan, D., & Chazelas, J. (2004). A priority list-based evolutionary algorithm to solve large scale unit commitment problem. In *2004 international conference on power system technology* (pp. 1746–1751, Vol. 2). IEEE, Singapore.

[50] Zhao, B., Guo, C., Bai, B., & Cao, Y. (2006). An improved particle swarm optimization algorithm for unit commitment. *International Journal of Electrical Power & Energy Systems*, *28*(7), 482–490.

[51] Panwar, L. K., Reddy, S., & Kumar, R. (2015). Binary fireworks algorithm based thermal unit commitment. *International Journal of Swarm Intelligence Research (IJSIR)*, *6*(2), 87–101.

[52] Yuan, X., Nie, H., Su, A., Wang, L., & Yuan, Y. (2009). An improved binary particle swarm optimization for unit commitment problem. *Expert Systems with applications*, *36*(4), 8049–8055.

[53] Ebrahimi, J., Hosseinian, S. H., & Gharehpetian, G. B. (2010). Unit commitment problem solution using shuffled frog leaping algorithm. *IEEE Transactions on Power Systems*, *26*(2), 573–581.

[54] Hadji, M. M., & Vahidi, B. (2011). A solution to the unit commitment problem using imperialistic competition algorithm. *IEEE Transactions on Power Systems*, *27*(1), 117–124.

[55] Lau, T., Chung, C., Wong, K., Chung, T., & Ho, S. L. (2009). Quantum-inspired evolutionary algorithm approach for unit commitment. *IEEE Transactions on Power Systems*, *24*(3), 1503–1512.

[56] Jabr, R. (2013). Rank-constrained semidefinite program for unit commitment. *International Journal of Electrical Power & Energy Systems*, *47*, 13–20.

[57] Jeong, Y.-W., Park, J.-B., Jang, S.-H., & Lee, K. Y. (2010). A new quantum-inspired binary pso: application to unit commitment problems for power systems. *IEEE Transactions on Power Systems*, *25*(3), 1486–1495.

[58] Quan, R., Jian, J.-b., & Mu, Y.-d. (2014). Tighter relaxation method for unit commitment based on second-order cone programming and valid inequalities. *International Journal of Electrical Power & Energy Systems*, *55*, 82–90.

[59] Todosijević, R., Mladenović, M., Hanafi, S., Mladenović, N., & Crévits, I. (2016). Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. *International Journal of Electrical Power & Energy Systems*, *78*, 873–883.

[60] Trivedi, A., Srinivasan, D., Biswas, S., & Reindl, T. (2015). Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, *23*, 50–64.

[61] Bai, X., & Wei, H. (2009). Semi-definite programming-based method for security-constrained unit commitment with operational and optimal power flow constraints. *IET Generation, Transmission & Distribution*, *3*(2), 182–197.

[62] Al-Betar, M. A. (2017). β-hill climbing: an exploratory local search. *Neural Computing and Applications*, *28*(1), 153–168.

[63] Awadallah, M. A., Al-Betar, M. A., Hammouri, A. I., & Alomari, O. A. (2020). Binary jaya algorithm with adaptive mutation for feature selection. *Arabian Journal for Science and Engineering*, *45*(12), 10875–10890.

[64] Al-Betar, M. A., Khader, A. T., & Zaman, M. (2012). University course timetabling using a hybrid harmony search metaheuristic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(5), 664–681.

[65] Awadallah, M. A., Bolaji, A. L., & Al-Betar, M. A. (2015). A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing*, *35*, 726–739.