axioms

MDPI

Article

Constrained Binary Optimization Approach for Pinned Node Selection in Pinning Control of Complex Dynamical Networks

Alma Y. Alanis [1], Jesus Hernandez-Barragan [1], Daniel Ríos-Rivera [1], Oscar D. Sanchez [2] and Gabriel Martinez-Soltero [2,*]

[1] Departamento de Innovación Basada en la Información y el Conocimiento, Centro Universitario de Ciencias Exactas e Ingenierías, Universidad de Guadalajara, Guadalajara 44430, Mexico; alma.alanis@academicos.udg.mx (A.Y.A.); daniel.rios8936@alumnos.udg.mx (D.R.-R.)

[2] Departamento de Ciencias Computacionales, Centro Universitario de Ciencias Exactas e Ingenierías, Universidad de Guadalajara, Guadalajara 44430, Mexico; didier.sanchez@academicos.udg.mx

* Correspondence: erasmo.martinez@academicos.udg.mx

Abstract: In complex dynamical networks, pinning control techniques are often applied to control a small fraction of the nodes in order to stabilize the network with reduced control effort and energy, facilitating adequate development of the complex network. Selecting the controlled nodes is a key challenge to achieving optimal performance. Theoretical analysis of the network provides the minimum quantity of nodes to control but does not specify which ones should be controlled. Analytically, controllability analysis of the entire network would be ideal, but this becomes difficult for complex networks with a large number of nodes and non-linear dynamics. Another option is to evaluate all possible combinations with the minimum number of necessary nodes or the nodes that can be controlled, but this presents a computational challenge due to the large number of possible combinations. Therefore, the remaining option is the use of metaheuristic algorithms for the rapid and practical evaluation of these combinations. In this work, we propose to optimize the selection of nodes for pinning control based on binary optimization algorithms, subject to control and development constraints. The proposed approach involves finding a binary combination with a fixed number of controlled nodes that best stabilizes the network state to zero. This paper includes a comparative study among state-of-the-art binary optimization algorithms and modified classic optimization algorithms. The applicability of the proposed approach is validated through simulations considering a dynamical discrete-time complex network

Keywords: binary optimization; combinatorial optimization; pinning control; complex dynamical networks

MSC: 37N35; 37N40; 68T07; 93B70

Citation: Alanis, A.Y.; Hernandez-Barragan, J.; Ríos-Rivera, D.; Sanchez, O.D.; Martinez-Soltero, G. Constrained Binary Optimization Approach for Pinned Node Selection in Pinning Control of Complex Dynamical Networks. *Axioms* **2023**, *12*, 1088. https://doi.org/10.3390/axioms12121088

Academic Editors: Ivan Mauricio Amaya-Contreras and José Carlos Ortiz-Bayliss

Received: 21 October 2023
Revised: 22 November 2023
Accepted: 24 November 2023
Published: 28 November 2023

1. Introduction

The study of complex networks is an interesting topic due to its wide range of applications in sociology, biology, mathematics, physics, computer science, robotics, and other complex interconnected dynamic systems [1–4]. The control of complex networks is an important task for achieving adequate collective behavior. However, controlling all nodes is expensive and impractical, especially in large-scale complex networks [3].

In dynamical systems, a complex network can be viewed as a group of interconnected systems, where each system has its own dynamics and characteristics [3,4]. A control strategy can be designed to force a complex network to exhibit a desired behavior. Pinning control strategies, which involve controlling only a small fraction of nodes instead of all nodes in the network, are often used in complex dynamical networks [4]. The nodes selected for pinning control are referred to as pinned nodes. The choice of which nodes to pin remains a significant research topic. In this work, we propose the use of binary

optimization algorithms to determine the optimal pinned nodes for stabilizing the complex network state to zero.

Metaheuristic algorithms have been widely used to solve complex optimization problems in different research areas [5–7]. Binary optimization is a kind of combinatorial optimization problem, where decision variables are expressed as binary values $\{0, 1\}$. A binary version of the arithmetic optimization algorithm (AOA) [8] is proposed in [9]. The new algorithm, called BinAOA, outperformed other binary optimization algorithms such as the tree seed algorithm, Jaya, grey wolf optimization (GWO), particle swarm optimization (PSO), differential evolution (DE), and artificial bee colony. A hybrid binary optimization technique between GWO and PSO, called BGWOPSO, is presented in [10]. The proposed version of the hybrid particle swarm optimization and grey wolf optimizer (HPSOGWO) algorithm outperformed the binary version of GWO and PSO. Moreover, BPSOGWO performed better than other binary optimization techniques, such as genetic algorithms (GAs) and the whale optimization algorithm (WHO). In [11], the authors proposed a modified version of binary rat swarm optimization (BRSO) to overcome the local optimum stagnation and slow convergence problems. This version is called M-BRSO and it outperformed other algorithms such as PSO, WHO, GA, DE, and ant lion optimizer. In [12], the authors proposed a new binary grasshopper optimization algorithm (NBGOA), which was compared against other binary swarm-based optimization algorithms: BGOA, the binary dragonfly optimization, binary GWO, and binary PSO. This new proposed version, called NBGOA, proved to be superior to the compared algorithms in the feature selection problems. The drawback of these algorithms is that they have no restrictions on how many ones or zeros there can be in the solution. However, for the problem considered in this work, the number of pinned nodes is fixed by the theoretical analysis and cannot be modified without altering controller performance. In [13], the firefly algorithm (FA) [14] is modified to a discrete version (DFA), adding a mutation operator that just changes the position of the values, allowing the solution to not change the values of the solution.

Given the total number of pinned nodes, selecting which nodes in scale-free networks are convenient to use is a crucial problem. First, evaluating all possible combinations of pinned nodes to control is computationally expensive and impractical since a large number of possible combinations can be generated in a complex network. Moreover, selecting the pinned nodes randomly is not an option since the total possible combination for a total number of pinned nodes is extremely high, provoking a very low probability of randomly selecting an optimal solution. In [15,16], techniques for determining confidence intervals and exact computations for proportions and differences of proportions were discussed, shedding light on the importance of precise calculations in statistical analyses. Moreover, in [17], the authors reported that selecting nodes with a larger number of connections is more effective for scale-free networks. However, selecting nodes with the greatest number of connections is not always the best strategy, especially in complex networks with varying connection strengths between nodes. Evaluating all possible combinations is not feasible [14]. Due to these drawbacks, metaheuristic algorithms appear to be the best option for rapidly and practically evaluating these combinations, achieving an adequate balance between computational complexity and precision. In this paper, we address the problem of selecting the most suitable pinned nodes in a scale-free network, using binary optimization algorithms. Additionally, a comparative study of state-of-the-art binary optimization algorithms is conducted to determine the algorithm that most effectively identifies the pinned nodes, thereby achieving the best stabilization of the complex network state at zero.

The contributions of this paper are as follows:

- We present an approach for finding the optimal pinned node selection for pinning control, aimed at stabilizing the complex network state to zero, given the total number of pinned nodes.
- We conduct a comparative study of state-of-the-art binary optimization algorithms to compute the optimal pinned nodes. The algorithms considered for comparison include the genetic algorithm (GA), arithmetic optimization algorithm (AOA), hybrid

algorithm of particle swarm optimization and grey wolf optimizer (HPSOGWO), rat swarm optimization (RSO), grasshopper optimization algorithm GOA, firefly algorithm FA, and its discrete version (DFA).

The paper is organized as follows: The problem formulation is presented in Section 2 along with the definition of the objective function. Section 3 presents the algorithms used in the comparison, and their respective modifications are shown. In Section 4, the results for pinned node selection are given. A brief analysis of the obtained results and the future research directions are presented in Section 5. Finally, conclusions are presented in Section 6.

2. Problem Formulation

The problem formulation is presented below. First, we described the complex network and the pinning control scheme. Then, the objective function formulation is described in detail.

2.1. Complex Network and Pinning Control

A dynamical complex network composed of N nodes can be described by the next model [4,18,19]

$$\dot{x}_\imath = f_\imath(x_\imath) + \sum_{\jmath=1}^{N} c_{\imath\jmath} a_{\imath\jmath} B x_\jmath + u_\imath, \tag{1}$$

where $x_\imath = \begin{bmatrix} x_{\imath 1} & x_{\imath 2} & \cdots & x_{\imath n} \end{bmatrix}^T$ is the state of the $\imath$-th node of dimension n. Function $f_\imath : \mathbb{R}^n \to \mathbb{R}^n$ represents the self-dynamics of node $\imath$. Constant $c_{\imath\jmath}$ is the connection strength between nodes $\imath$ and $\jmath$. Constant $a_{\imath\jmath} = 1$ if there is a connection between nodes $\imath$ and $\jmath$, otherwise $a_{\imath\jmath} = 0$. When $\imath = \jmath$, $a_{\imath\imath} = -k_\imath$, where $k_\imath$ is the degree (number of connections) of node $\imath$. Matrix $B \in \mathbb{R}^{n \times n}$ describes the connections between different components of the state vector. Finally, $u_\imath$ is a control input.

A common method to drive the system in (1) to stability is the pinning control technique [17], where only a few nodes of the network are controlled, even just one node [20]. A pinning control technique for complex networks can be defined as in [4]:

$$u_\imath(x_\imath) = -\kappa_\imath x_\imath, \tag{2}$$

where $\kappa_\imath$ is a real negative constant if node $\imath$ is selected as a pinned node, otherwise $\kappa_\imath = 0$. The value $\kappa_\imath$ is defined to yield the system converge to the 0 state. However, elements $\kappa_\imath$ can be chosen following a V-stability to warrant the system stability [4]. In this work, elements $\kappa_\imath$ are selected as $\kappa_\imath = 190$.

For coding purposes, we consider a discrete-time version of the model (1), defined as [4]

$$x_\imath(k+1) = x_\imath(k) + T\left(f_\imath(x_\imath(k)) + \sum_{\jmath=1}^{N} c_{\imath\jmath} a_{\imath\jmath} B x_\jmath(k) + u_\imath(x_\imath(k)) \right), \tag{3}$$

where T is the sampling time. Then, the discrete-time control input is given by

$$u_\imath(x_\imath(k)) = -\kappa_\imath x_\imath(k). \tag{4}$$

where $x_\imath(k)$ is the state of the $\imath$-th node at step time k.

2.2. Objective Function Formulation

A candidate binary solution ${}^{[B]}s \in \mathbb{Z}^{0+}$ is composed of a set of $\kappa_\imath$ values, defined as:

$${}^{[B]}s = \begin{bmatrix} \kappa_1 & \kappa_2 & \cdots & \kappa_N \end{bmatrix}^T, \tag{5}$$

where each decision variable κ_i is expressed with binary values $\{0,1\}$ based on the next scheme:

$$\kappa_\imath = \begin{cases} 1 & \text{if node } \imath \text{ represents a pinned node} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Note that the solution ${}^{[B]}s$ encodes which nodes in the network are considered as pinned nodes. For a total number of N_p pinned nodes analytically determined, we must ensure that $1 < N_p < N$ and

$$\sum_{\imath=1}^{N} \kappa_\imath = N_p, \tag{7}$$

otherwise, ${}^{[B]}s$ represent an infeasible solution.

Given a feasible candidate solution ${}^{[B]}s$, the pinning control (4) is performed to drive the system in (3) to stability. In order to find the solution ${}^{[B]}s$ that best stabilizes the network states at zero, the following cost function is proposed:

$$g = \sum_{\imath=1}^{N} \sum_{\jmath=1}^{n} |x_{\imath\jmath}(k_{end})|, \tag{8}$$

where $x_\imath(k_{end})$ is the state of the $\imath$-th node in the $\jmath$-th dimension at the final step time k_{end}. Since the main objective of this work is to select a subset of nodes to stabilize the network dynamics to zero, $x_{\imath\jmath}(k_{end})$ defines the stabilization error.

The aim of the proposed approach is to minimize the cost function (8) based on binary optimization algorithms.

3. Algorithms Used and Their Modifications

In the following subsections, the implemented algorithms are presented, showing their inspiration, representative equations, and modifications or additions to obtain binary solutions with a restricted number of ones. In this work, it was decided to unify the notation and the variables so that—regardless of the algorithm—they mean the same thing, with three specific ones: ${}^{[R]}s$, which represents the individual whose solution contains real values; ${}^{[I]}s$, which contains the solution in decimal integer form; and ${}^{[B]}s$, which is the binary solution of each individual. Regarding sub-indexes, i refers to the i-th individual in the population of each algorithm and d refers to the d-th position in the solution whose values go from 1 to the dimension (D).

3.1. Genetic Algorithms

Genetic algorithms are based on features of natural selection, such as a reproducing population, wherein every individual has a period of life, the population varies over time, and the individual's lifespan is related to its performance. The features mentioned earlier are encapsulated in three operations: selection, crossover, and mutation, which are widely used in combinatorial optimization problems. These problems include balancing the assembly line [21], descriptor selection [22,23], and planning of renewable energy communities [24]. The three operators can function in various ways, as described in [25,26]. However, the specific methods used in this study are presented in [14].

The selection is based on the probability p_i of the i-th individual ${}^{[B]}s_i$ and its probability is calculated with

$$p_i = \frac{fitness_i}{\sum_{k=1}^{M} fitness_k}, \tag{9}$$

the fitness is calculated by

$$fitness_i\left({}^{[B]}s_i\right) = \begin{cases} \dfrac{1}{1+f\left({}^{[B]}s_i\right)} & \text{if } f\left({}^{[B]}s_i\right) \geq 0 \\ 1+|f\left({}^{[B]}s_i\right)| & \text{if } f\left({}^{[B]}s_i\right) < 0. \end{cases} \tag{10}$$

Once the complete population has its probabilities calculated, the selection is done following Algorithm 1, where p_{sum} is the accumulated sum of p_i and M is the population size.

Algorithm 1: Parent selection algorithm.

```
r ← random() ;                          /* generates a real number r ∈ [0,1) */
p_sum ← 0;
for i ← 1 to M do
    p_sum ← p_sum + p_i;
    if p_sum <= r then
        i-th parent selected;
    end
end
```

In the crossover, from the two parents p_1, p_2 selected with Algorithm 1, two children, y_1 and y_2, are generated according to Algorithm 2, where a crossover point p_c is an integer number generated randomly in the range from 1 to the dimension of problem D, to know which characteristics are inherited from each parent.

Algorithm 2: Crossover.

```
p_c ← randInt(1,D);        /* generates an integer number ∈ {1,2,...,D} */
for j ← 1 to D do
    if j < p_c then
        y_2j ← p_1j;   y_1j ← p_2j;
    else
        y_1j ← p_1j;   y_2j ← p_2j;
    end
end
```

In the mutation, a probability p_m is fixed to decide if a chromosome is mutated or not following Algorithm 3, where r_a and r_b are random real numbers generated in the range of [0, 1), and LB_d and UB_d are the lower and upper bounds in the d-th position, respectively.

Algorithm 3: Mutation.

```
for i ← 1 to M do
    for d ← 1 to D do
        r_a ← random() ;                /* generates a real number ∈ [0,1) */
        if r_a < p_m then
            r_b ← random() ;            /* generates a real number ∈ [0,1) */
            y_id ← LB_d + (UB_d − LB_d) r_b;
        else
            y_id ← y_id;
        end
    end
end
```

Since there is a possibility that during crossover, children may be generated with more or fewer bits than N_p, a new mutation operator has been proposed; see Algorithm 4. This algorithm counts the number of times the solution has been mutated.

Algorithm 4: Binary-constrained mutation.

```
for i ← 1 to M do
    c ← 0;
    while sum(y_i) < N_p or c < 2 do
        d ← randInt(1,D); /* generates an integer number ∈ {1,2,...,D} */
        v ← randInt(0,1) ;        /* generates an integer number ∈ {0,1} */
        if y_id ≠ v then
            c ← c + 1;
            y_id ← v;
        end
    end
end
```

3.2. Arithmetic Optimization Algorithm

The arithmetic optimization algorithm [8] is based on the elementary arithmetic operators, addition, subtraction, multiplication, and division. It is a population-based algorithm in which individuals are generated randomly, it introduces two terms, math optimizer accelerated (MOA) and math optimization probability (MOP) function calculated by Equations (11) and (12) as follows:

$$\mathrm{MOA}(C_{iter}) = Min + C_{iter} \times \left(\frac{Max - Min}{M_{iter}}\right), \tag{11}$$

$$\mathrm{MOP}(C_{iter}) = 1 - \left(\frac{{C_{iter}}^{1/\alpha}}{{M_{iter}}^{1/\alpha}}\right), \tag{12}$$

where $\mathrm{MOA}(C_{Iter})$ and $\mathrm{MOP}(C_{Iter})$ denote the function values at the current iteration (C_{iter}), which ranges between 1 and the maximum number of iterations (M_{iter}). Min and Max denote the minimum and maximum values of the MOA, respectively, α is a sensitive parameter that defines exploitation accuracy over iterations. The balance between exploration and exploitation is determined by r_1 a random real value generated between 0 and 1. When r_1 is lower than the MOA value, the algorithm follows an exploitation strategy; otherwise, it will perform an exploration strategy. The exploration aspect is facilitated by the multiplication and division operators, which have an equal probability of being applied using r_2, which is a random real number in the range of [0, 1). Exploration is given by

$${}^{[R]}s_{id}(C_{iter}+1) = \begin{cases} best({}^{[R]}s_{id}) \div (\mathrm{MOP} + \varepsilon) \times ((\mathrm{UB}_d - \mathrm{LB}_d) \times \mu + \mathrm{LB}_d) & r_2 < 0.5 \\ best({}^{[R]}s_{id}) \times (\mathrm{MOP}) \times ((\mathrm{UB}_d - \mathrm{LB}_d) \times \mu + \mathrm{LB}_d) & \text{otherwise,} \end{cases} \tag{13}$$

where ${}^{[R]}s_{id}$ denotes the i-th solution in the population that stores real values in the d-th position, and $best({}^{[R]}s_{id})$ is the d-th position in the best solution so far. ϵ is a small integer number, and UB_d and LB_d are upper and lower bounds of the d-th position, respectively. The μ term is a control parameter used to adjust the search process, which is fixed to 0.5, according to the authors. For the exploitation characteristic, addition and subtraction operators are designed, allowing the random term r_3 to decide which operator is applied, according to the following equation

$${}^{[R]}s_{id}(C_{iter}+1) = \begin{cases} best({}^{[R]}s_{id}) - (\mathrm{MOP}) \times ((\mathrm{UB}_d - \mathrm{LB}_d) \times \mu + \mathrm{LB}_d) & r_3 < 0.5 \\ best({}^{[R]}s_{id}) + (\mathrm{MOP}) \times ((\mathrm{UB}_d - \mathrm{LB}_d) \times \mu + \mathrm{LB}_d) & \text{otherwise.} \end{cases} \tag{14}$$

In its binary version, the elements of every individual pass through a transfer function followed by a threshold to convert continuous values to binary values [9], but this process does not guarantee that the solution will end with a fixed number of ones. Therefore, to transform the continuous values to binary values, the binary-constrained transform is

applied to each ${}^{[R]}s$ in the population, ensuring that the number of elements set to one remains equal to N_p. This transform is presented in Algorithm 5.

Algorithm 5: Binary-constrained transform.

```
[R]s_i = delOutBounds([R]s_i) ;   /* pops all the index out of the bounds */
[I]s_i ← floor([R]s_i);
[I]s_i ← unique([I]s_i);                    /* pops all the index repeated */
while len([I]s_i) < N_p do
    [R]s_r = randInt(1, D) ; /* generates an integer number ∈ {1,2,...,D} */
    [R]s_i ← {[R]s_i,[R] s_r} ;        /* add a random number to the solution */
    [I]s_i ← floor([R]s_i);
    [I]s_i ← unique([I]s_i)
end
[B]s_i ← {0_1, 0_2, 0_3, ..., 0_D}; /* create a vector of zeros of dimension D */
for d ∈ [I]s_i do
    [B]s_id ← 1 ;     /* set to 1 the d-th position in the i-th solution */
end
```

In Algorithm 5, the delOutBounds(·) function works to delete all values that are out of bounds. In this case, since the solution contains real numbers that are used as indices where the binary string has ones, the function will eliminate values lower than zero and higher than the dimension of the problem. The function floor (·) is used to convert the real values to integers, enabling their use as indices. The unique (·) function will return the solution without repeated indices. The len (·) function returns the length of the solution, allowing us to check if it contains as many elements as required, which is N_p. If the solution is missing values, it will be filled with random indices until they are all distinct. Once all the indices are different, a binary solution ${}^{[B]}s_i$ filled with zeros is generated. After that, positions included in the solution are changed to ones.

3.3. Optimization Hybrid Grey Wolf Optimization

The original version of this algorithm is a combination of grey wolf optimization and particle swarm optimization [27], creating a hybridization of the specific equations of each. This results in three steps: searching for prey, attacking the prey, and moving as in PSO. The first step is defined by

$$\begin{aligned} D_\alpha &= |C_1 \cdot {}^{[R]}\, s_\alpha - w \cdot {}^{[R]}\, s_i|, \\ D_\beta &= |C_2 \cdot {}^{[R]}\, s_\beta - w \cdot {}^{[R]}\, s_i|, \\ D_\delta &= |C_3 \cdot {}^{[R]}\, s_\delta - w \cdot {}^{[R]}\, s_i|, \end{aligned} \tag{15}$$

where each value D_α, D_β, and D_δ represents the movement of each wolf to the three best wolves (α, β, δ), the values of C_1, C_2, C_3 are random values in the range of [0, 2], and w is the inertia factor generated in each iteration, randomly, in the range of [0.5, 1].

The 'attacking the prey' step is defined as

$$\begin{aligned} \omega_\alpha &= {}^{[R]}\, s_\alpha - A_1 D_\alpha, \\ \omega_\beta &= {}^{[R]}\, s_\beta - A_2 D_\beta, \\ \omega_\delta &= {}^{[R]}\, s_\delta - A_3 D_\delta, \end{aligned} \tag{16}$$

where A is a random value in the range of $[-a, a]$, where the value of a decreases from two to zero as the iterations pass.

After the 'attacking the prey' step, the movements of individuals are carried out according to a velocity that is calculated in a similar way to PSO. The moving step is given by

$$v_i(C_{iter}+1) = wv_i(C_{iter}) + c_1r_1(\omega_\alpha - {}^{[R]}s_i) + c_2r_2(\omega_\beta - {}^{[R]}s_i) + c_3r_3(\omega_\delta - {}^{[R]}s_i), \quad (17)$$

$${}^{[R]}s_i(C_{iter}+1) = {}^{[R]}s_i(C_{iter}) + v_i(C_{iter}+1), \quad (18)$$

where c_1, c_2, c_3 are constants that ponder which wolf follows more, and r_1, r_2, r_3 are random values in the range of [0, 1].

This algorithm works in a continuous domain. To make a binary solution, the authors of [10] use a transform function, defined as

$${}^{[R]}s(C_{iter}) = \begin{cases} 1 & \text{if } sigmoid\left(\frac{{}^{[R]}s_1 + {}^{[R]}s_2 + {}^{[R]}s_3}{3}\right) > random() \\ 0 & \text{otherwise,} \end{cases} \quad (19)$$

$$sigmoid(a) = \frac{1}{1+e^{-10(a-0.5)}}, \quad (20)$$

where ${}^{[R]}s_1$, ${}^{[R]}s_2$, and ${}^{[R]}s_3$ are positions generated from every individual with knowledge of the best three individuals (wolves α, β, and δ). But instead of using this algorithm, the continuous version and the transform Algorithm 5 presented before were applied.

3.4. Grasshopper Optimization Algorithm

The grasshopper optimization algorithm (GOA) [28] is also a nature-inspired population algorithm. It balances exploration and exploitation with the lifespan of grasshoppers. While they are larvae, they have slow movements. In childhood, they perform small steps. During adulthood, they perform long-range steps.

In this work, the authors introduced an interaction operator, S, defined by

$$S(r) = \beta e^{\frac{-r}{l}} - e^r, \quad (21)$$

where β and l indicate the intensity of attraction and the attractive length scale, respectively. Then, the movement of the grasshopper is given by

$${}^{[R]}s_{id} = \left(\sum_{j=1}^{M} c\frac{UB_d - LB_d}{2} S(|{}^{[R]}s_{jd} - {}^{[R]}s_{id}|)\frac{{}^{[R]}s_j - {}^{[R]}s_i}{h_{ij}}\right) + T_d, \quad (22)$$

where T_d is the value in the d-th dimension in the best solution found, and the coefficient, c, is reduced as the iterations pass in the following way:

$$c = Max - C_{iter}\frac{Max - Min}{M_{iter}} \quad (23)$$

where Min and Max denote the minimum and maximum values of c, respectively, through the iterations. In the original work, $Max = 1$ and $Min = 0.00001$. This algorithm also has its binary version presented in [12], but it can provide a solution with a variable number of ones; for this experiment, the continuous version is passed by the constrained transform algorithm.

3.5. Rat Swarm Optimization Algorithm

Inspired by the behavior of rats in nature, the rat swarm optimization (RSO) [29] emulates the aggressiveness and social intelligence of rats against other animals, with two main phases: chasing and fighting.

Chasing the prey is represented by

$$P = A \cdot^{[R]} s_i + C \cdot (P_r -^{[R]} s_i), \tag{24}$$

where $^{[R]}s_i$ is the rat position and P_r is the position of the best rat in the population, A is a calculated parameter, and C is a random number in the interval [0, 2]. Moreover,

$$A = r - C_{iter}\left(\frac{r}{M_{iter}}\right), \tag{25}$$

where r is a vector with random numbers in the interval $[1, 5]$.

For the fighting phase, the following equation models the behavior

$$^{[R]}s_i(C_{iter} + 1) = |P_r - P|. \tag{26}$$

The previous equations are suitable for continuous problems, and there are binary versions, such as in [11], as previously mentioned. However, once again, this algorithm is not constrained to produce a combination of binary values with a specific number of ones. Therefore, to make it capable of generating a solution of interest for this work, the algorithm that transforms the continuous solution into a binary solution is applied before the call to the objective function.

3.6. Firefly Algorithm

The firefly algorithm is based on the attraction of fireflies to each other [14], being attracted to those whose brightness is greater, and the perception of light decreases exponentially with distance

The following equation represents the attractiveness β from firefly $^{[R]}s_i$ to firefly $^{[R]}s_j$

$$\beta = \frac{I_0}{1 + \gamma \cdot (h_{ij})^2}, \text{where } I_0 = 0.8, \tag{27}$$

$$h_{ij} = \|^{[R]}s_i -^{[R]} s_j\|, \tag{28}$$

where $\gamma \in [0, \infty)$ is the absorption coefficient and I_0 is the light intensity at the source, normally as $I_0 = 1$.

The movement of fireflies is defined by

$$^{[R]}s_i =^{[R]} s_i + \beta \cdot (^{[R]}s_i -^{[R]} s_j) + \alpha \cdot (r - 0.5), \tag{29}$$

where α is the size of the random step since the term $(r - 0.5)$ will generate a value between $[-0.5, 0.5]$. The continuous version of this algorithm is applied with the binary-constrained transform algorithm 5. In addition to the experiments, the discrete version presented in [13] is implemented for the comparison. Since it introduces the mutation operator, it is feasible to use in this problem, constraining it to values of zero or one. Since its mutation just moves the values of place, it keeps a fixed number of ones in the solution.

3.7. Summary

In summary, seven algorithms are presented with different modifications—some work natively with binary solutions and others work with real values that are later changed to a binary solution through a transformation function. The change applied to each algorithm is shown in Table 1.

Table 1. Modifications made to every algorithm.

Algorithm	Modification
GA	Binary-Constrained Mutation
AOA	Binary-Constrained Transform
HPSOGWO	Binary-Constrained Transform
GOA	Binary-Constrained Transform
RSO	Binary-Constrained Transform
FA	Binary-Constrained Transform
DFA	None

4. Results

To demonstrate the applicability of the proposed constrained binary optimization for the pinned node selection, we consider a scale-free network of 50 nodes. The topology was generated randomly, taking into account the goal of having approximately 20% of the possible connections. The resulting network is illustrated in Figure 1, which has a node connection degree mean value of 10.28. Furthermore, to better appreciate the connection degree of each node, Figure 2 is presented. The number of pin nodes was set to $N_p = 10$.

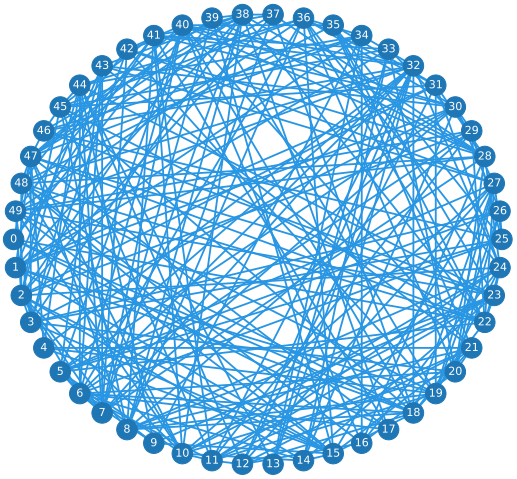

Figure 1. The 50-node network used for the simulations.

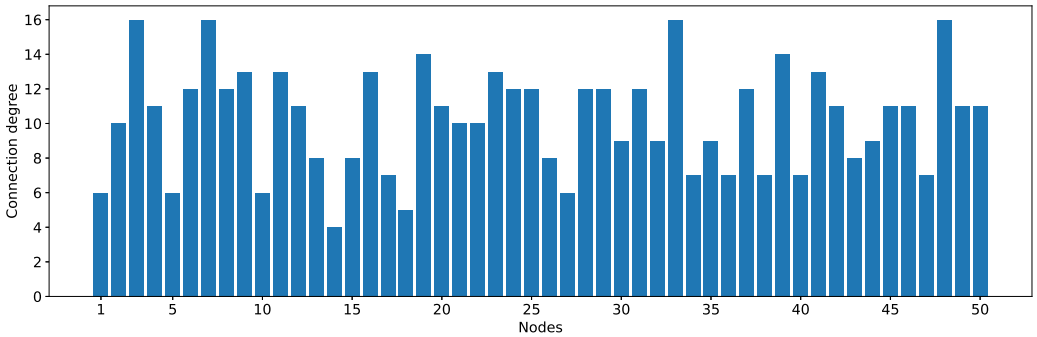

Figure 2. Connection degrees in the considered complex network for the simulations.

The parameters of the network are given below. The connection strength is set to $c_{ij} = 100$. Moreover, matrix B is composed as:

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{30}$$

Matrix B is defined as an identity since no connection among the states x_ι is required.

The node's self-dynamics are determined by the following chaotic attractor:

$$f_\iota(x_\iota(k)) = \begin{bmatrix} 10(x_{\iota 2}(k) - x_{\iota 1}(k)) \\ 28x_{\iota 1}(k) - x_{\iota 1}(k)x_{\iota 3}(k) - x_{\iota 2}(k) \\ x_{\iota 1}(k)x_{\iota 2}(k) - (8/3)x_{\iota 3}(k) \end{bmatrix}, \tag{31}$$

which is the Lorenz system [30]; chaotic attractors are commonly used in the literature to evaluate the performance of controllers in complex networks [31–34]. Stability is preserved by controlling N_p nodes according to the analysis presented in [4,20]. It is important to note that this analysis does not provide information on which nodes to control; hence, it is the main objective of this research.

All algorithms were tested with 30 iterations with a population size of $M = 20$, except for the firefly algorithm, which has $M = 10$ since it performs a comparison of all the individuals. Particular parameters for each algorithm are listed in Table 2.

Table 2. Parameters of each algorithm.

Algorithm	Parameters
GA	$M = 20, p_m = 0.7$
AOA	$M = 20, \alpha = 5, \mu = 0.5, \varepsilon = 0.000001, Max = 0.2, Min = 0.05$
HPSOGWO	$M = 20, c_1 = c_2 = c_3 = 0.5$
GOA	$M = 20, \beta = 0.5, l = 1.5$
RSO	$M = 20,$
FA	$M = 10, \gamma = 0.4, \alpha = 0.01$
DFA	$M = 10, \gamma = 0.4, \alpha = 0.01$

Other selection strategies for pinning the nodes involve choosing them randomly or first picking the node with the highest connection degree, and then continuing to select and pin the other nodes sequentially in decreasing order of connection degrees. With this in mind, three strategies to select the pin nodes are presented. First, 100 attempts with the random strategy are made; in Table 3, the best five results of this strategy are shown.

Table 3. Results of random pinning nodes.

Selected Nodes (ι)	Stabilization Error
1,3,4,6,7,10,32,33,48,50	**0.0105133261**
3,4,13,18,19,20,24,37,44,50	0.0107734755
1,9,19,20,21,25,34,45,46,48	0.0108550197
1,6,11,16,24,28,30,31,44,50	0.0142530794
1,3,19,28,32,36,38,42,48,50	0.0150178204

Second, with the strategy of choosing the ones with more connections and picking N_p nodes, there are five possible combinations that return the results shown in Table 4.

Table 4. Results of the five combinations with nodes with more connections.

Selected Nodes (ι)	Stabilization Error
3,7,33,48,19,39,8,11,16,23	0.0012275025
3,7,33,48,19,39,8,11,16,41	0.0023177425
3,7,33,48,19,39,8,11,23,41	0.0031266226
3,7,33,48,19,39,8,11,23,41	0.0013908266
3,7,33,48,19,39,11,16,23,41	**0.0004301837**

Finally, the results of each algorithm are presented in Table 5, showing its respective combination and error obtained. Figure 3 shows how the evaluation fitness function is minimized through the iterations of each algorithm.

Table 5. Results obtained by each algorithm.

Algorithm	Selected Nodes (ι)	Stabilization Error
GA	3,14,22,33,34,35,37,38,43,48	0.0001228105
AOA	1,3,4,6,33,38,39,43,48,50	0.0037335172
HPSOGWO	1,3,16,18,20,29,33,34,48,50	$7.0052533573 \times 10^{-6}$
GOA	3,10,18,20,22,28,30,33,37,48	$1.2757280761 \times 10^{-6}$
RSO	3,5,10,12,13,19,24,33,37,48	$1.5527264707 \times 10^{-5}$
FA	1,3,19,20,26,31,37,42,44,48	$4.681870615 \times 10^{-7}$
DFA	3,16,18,20,29,31,33,44,47,48	$\mathbf{2.0350233065} \times 10^{-7}$

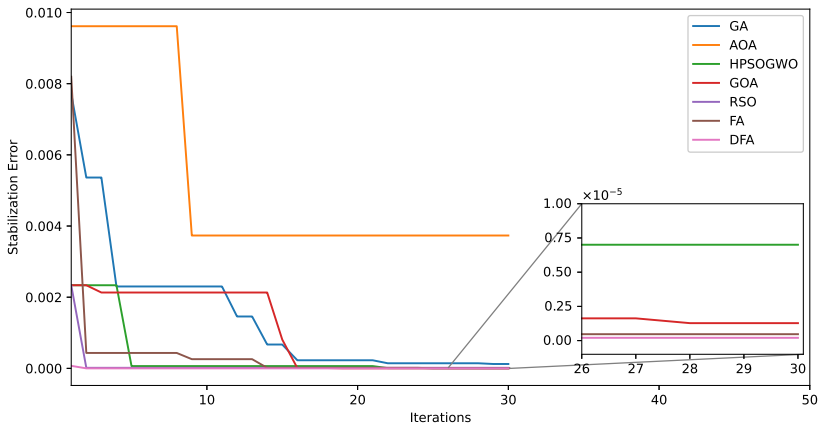

Figure 3. Stabilization error of each algorithm in every iteration.

The second experiment uses the same topology of the network, but four chaotic attractors are added. They are defined by the following equations:

$$f_\iota(x_\iota(k)) = \begin{bmatrix} 35(x_{\iota 2}(k) - x_{\iota 1}(k)) \\ -7x_{\iota 1}(k) - x_{\iota 1}(k)x_{\iota 3}(k) + 28x_{\iota 2}(k) \\ x_{\iota 1}(k)x_{\iota 2}(k) - 3x_{\iota 3}(k) \end{bmatrix}, \tag{32}$$

which is the Chen system, with its parameters as they appear in [35].

$$f_\iota(x_\iota(k)) = \begin{bmatrix} 36(x_{\iota 2}(k) - x_{\iota 1}(k)) \\ 15x_{\iota 1}(k) - x_{\iota 1}(k)x_{\iota 3}(k) \\ x_{\iota 1}(k)x_{\iota 2}(k) - 3x_{\iota 3}(k) \end{bmatrix}, \tag{33}$$

which is the Lü system, where its parameters are obtained from [36].

$$f_\iota(x_\iota(k)) = \begin{bmatrix} 10(x_{\iota 2}(k) - x_{\iota 1}(k)) + x_{\iota 2}(k)x_{\iota 3}(k) \\ 25x_{\iota 1}(k) - x_{\iota 1}(k)x_{\iota 3}(k) - x_{\iota 2}(k) \\ x_{\iota 1}(k)x_{\iota 3}(k) - 7x_{\iota 3}(k) \end{bmatrix}, \tag{34}$$

which is the Qi system, with the parameters set as in [37].

$$f_\iota(x_\iota(k)) = \begin{bmatrix} 9.35(x_{\iota 2}(k) - h(x_{\iota 1}(k))) \\ x_{\iota 1}(k) - x_{i 2}(k) + x_{\iota 3}(k) \\ -14.79x_{\iota 3}(k) \end{bmatrix}, \tag{35}$$

which is the Chua system, where $h(x_{i1(k)})$ is defined by

$$h(x_{\iota 1}(k)) = \frac{2}{7}x_{\iota 1}(k) - \frac{3}{14}(|x_{\iota 1}(k) + 1| - |x_{\iota 1}(k) - 1|) \tag{36}$$

as it appears in [38]. All the chaotic attractors are interspersed within the network, according to Table 6.

Table 6. Systems assigned to each node.

System	Nodes(ι)
Lorenz	1,6,11,16,21,26,31,36,41,46
Chen	2,7,12,17,22,27,32,37,42,47
Lü	3,8,13,18,23,28,33,38,43,48
Qi	4,9,14,19,24,29,34,39,44,49
Chua	5,10,15,20,25,30,35,40,45,50

As in the previous experiment, random combinations of pin nodes were selected, attempting to obtain the combination with the best performance. Table 7 shows the best five combinations obtained from 100 tries.

Table 7. The top five results of random pinning nodes for experiment two.

Selected Nodes (ι)	Stabilization Error
1,3,14,26,29,34,36,42,47,50	**0.0107664230**
3,6,10,11,16,24,26,33,37,44	0.0693599189
1,2,3,7,24,27,31,36,42,50	0.1874461797
3,4,8,19,20,23,29,35,38,44	0.2147937241
2,7,8,9,13,21,25,31,34,50	0.2685484577

Table 8 shows the results of setting the nodes with more connections as pinned nodes. It can be seen that some combinations of pinned nodes failed to converge, as indicated by the results computed as a 'not applicable' (N/A) value. Selecting the nodes with the most connections as pinned nodes does not guarantee the stability of the system. Table 9 shows the stabilization error obtained by all the algorithms, being AOA the one that obtained the lowest error.

Table 8. Results of the five combinations with nodes with more connections for experiment two.

Selected Nodes (ι)	Stabilization Error
3,7,33,48,19,39,8,11,16,23	N/A
3,7,33,48,19,39,8,11,16,41	22.495094562
3,7,33,48,19,39,8,11,23,41	25.991754941
3,7,33,48,19,39,8,11,23,41	N/A
3,7,33,48,19,39,11,16,23,41	**5.0256082799**

Table 9. Results obtained by each algorithm for experiment two.

Algorithm	Selected Nodes (ι)	Stabilization Error
GA	9,19,27,29,30,34,35,37,46,48	0.0018476030
AOA	1,3,5,6,12,19,35,39,48,50	**0.0001338463**
HPSOGWO	10,11,12,17,19,21,26,33,35,47	0.0045804490
GOA	1,5,9,10,15,22,24,32,35,44	0.0099802442
RSO	5,9,15,22,23,26,32,34,44,47	0.0051811245
FA	1,3,19,20,26,31,37,42,44,48	0.0008024141
DFA	5,7,14,19,26,29,31,34,35,46	0.0180924252

5. Discussion

Given the scale-free network in Figure 1, the number of possible combinations with $N_P = 10$ pinned nodes are 10,272,278,170. Evaluating all of these combinations is not feasible; therefore, if there was a single optimal solution, the probability of finding it randomly would be 9.73493×10^{-11}. Moreover, there could be more than one optimal solution. However, all the combinations should be evaluated in order to find them, returning to the challenge that it is not feasible to evaluate all possible solutions.

The reported results from the experiments show that randomly picking the pinned nodes is not an option if users are interested in finding an optimal solution. On the other hand, the selection of the pinned nodes with the most connections (as can be seen in Figure 2) is a better option than picking them randomly. However, the results presented in the simulations section show that they are not optimal solutions. Moreover, some node selections may yield the system to instability, as can be seen in the results of Table 8. The simulation results demonstrated that binary optimization algorithms, such as GA, HPSOGWO, GOA, RSO, FA, and DFA, outperformed the random selection of pinned nodes and the selection of nodes with the most connection approaches. The use of binary optimization algorithms is a suitable option to provide an optimal pinned node selection in a scale-free network.

This work compared the performances of the following binary optimization algorithms: GA, AOA, HPSOGWO, GOA, RSO, FA, and DFA for pinned node selections in the pinning control of complex dynamical networks. Convergence curves in Figure 3 show that the AOA algorithm performed poorly. Although the performances of GA, HPSOGWO, GOA, RSO, FA, and DFA algorithms are quite similar, DFA outperformed the others. DFA demonstrated the lowest error results and fastest convergence rates within 30 iterations.

Figure 4 shows the convergence curves of the algorithms for experiment two. In this experiment, AOA outperformed all of the compared algorithms, followed by FA, which—as in the first experiment—obtained the second-best results. All the algorithms outperformed the selection of combinations with the highest connection degree. Except for DFA, the rest of the algorithms performed better than the random selection of pin nodes. It is remarkable that, out of one hundred random selections, only one obtained a better result than the DFA, which was the worst of the algorithms tested in experiment two.

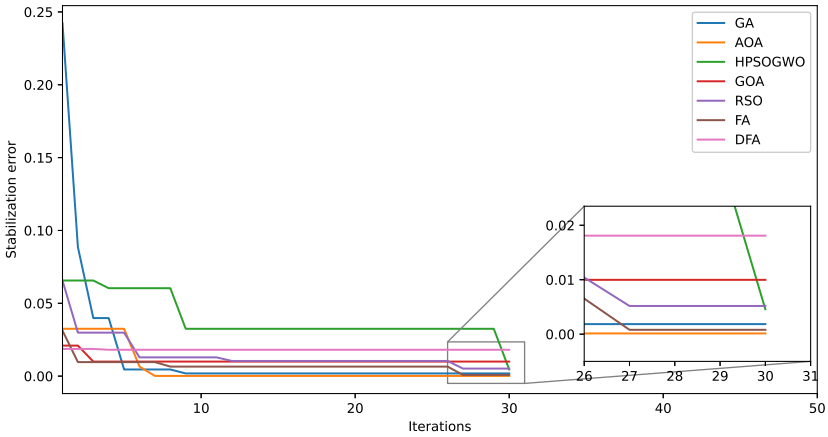

Figure 4. Stabilization errors for each algorithm for every iteration in the experiment, with 5 different systems.

This work aimed to find the optimal selection of pinned nodes to control a dynamical complex network, rather than determine the total number of pinned nodes. Indeed, the proposed approach requires a specific number of pinned nodes. Moreover, since controlling all nodes in a complex network is expensive and impractical, defining an adequate number of pinned nodes is a crucial task to solve. In future work, we plan to implement an optimization approach to determine the best nodes for pin control, along with the number of nodes necessary to effectively stabilize the complex network state at zero. On the other hand, it is also appealing to propose a global optimization approach to obtain optimal controller gains for improving convergence rates at zero.

6. Conclusions

This paper addresses the problem of pinned node selection in the pinning control of scale-free complex networks using binary optimization algorithms.

Simulation experiments with a 50-node complex dynamical network were performed to show that binary optimization algorithms are more convenient for pinned selection than conventional selection methods, such as picking pinned nodes randomly and selecting those nodes with a high degree of connections.

Moreover, a comparison was performed to compare the behaviors of AOA, HPSOGWO, GOA, RSO, FA, and DFA binary algorithms for pinned node selection. Simulation results demonstrated that DFA outperformed other algorithms when all nodes were homogeneous (all the nodes had the same self-dynamics). DFA exhibited the fastest convergence rates with the lowest optimization errors. Based on the results, we found that AOA was the most suitable algorithm for pinned node selection in the pinning control of scale-free complex dynamical networks when the nodes are heterogeneous. FA had the same performance in both experiments.

Author Contributions: A.Y.A.: Supervision, project administration, review, editing, funding acquisition. J.H.-B.: Writing, review, and editing. D.R.-R.: Methodology and implementation of the simulation results. O.D.S.: Supervision, review, and editing. G.M.-S.: Studied the state-of-the-art, methodology, implementation of the simulation results, and writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONAHCYT Mexico, through project PCC-2022-319619.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors also thank Universidad de Guadalajara for their support in this research.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AOA	arithmetic optimization algorithm
BinAOA	binary arithmetic optimization algorithm
BGOA	binary grasshopper optimization algorithm
BGWOPSO	binary hybrid grey wolf optimization-particle swarm optimization
BRSO	binary rat swarm optimization
DE	differential evolution
DFA	discrete firefly algorithm
FA	firefly algorithm
GA	genetic algorithm
GOA	grasshopper optimization algorithm
GWO	grey wolf optimization
HPSOGWO	hybrid algorithm of particle swarm optimization and grey wolf optimizer
MOA	math optimizer accelerated
MOP	math optimization probability
NBGOA	New binary grasshopper optimization algorithm
PSO	particle swarm optimization
RSO	rat swarm optimization
WHO	whale optimization algorithm

References

1. Strogatz, S.H. Exploring Complex Networks. *Nature* **2001**, *410*, 268–276. [CrossRef] [PubMed]
2. Chen, G.; Wang, X.; Li, X. *Fundamentals oF Complex Networks: Models, Structures and Dynamics*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
3. Sun, W.; Lü, J.; Chen, S.; Yu, X. Pinning impulsive control algorithms for complex network. *Chaos Interdiscip. J. Nonlinear Sci.* **2014**, *24*, 013141. [CrossRef] [PubMed]
4. Ríos-Rivera, D.; Alanis, A.Y.; Sanchez, E.N. Neural-Impulsive Pinning Control for Complex Networks Based on V-Stability. *Mathematics* **2020**, *8*, 1388. [CrossRef]
5. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
6. Peres, F.; Castelli, M. Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design, and Development. *Appl. Sci.* **2021**, *11*, 6449. [CrossRef]
7. Pan, J.S.; Hu, P.; Snášel, V.; Chu, S.C. A survey on binary metaheuristic algorithms and their engineering applications. *Artif. Intell. Rev.* **2023**, *56*, 6101–6167. [CrossRef] [PubMed]
8. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
9. Emine, B.; Yildizdan, G. A New Binary Arithmetic Optimization Algorithm for Uncapacitated Facility Location Problem. 2022. Available online: https://www.researchgate.net/publication/363863370_A_New_Binary_Arithmetic_Optimization_Algorithm_For_Uncapacitated_Facility_Location_Problem (accessed on 1 October 2023).
10. Al-Tashi, Q.; Kadir, S.J.A.; Rais, H.M.; Mirjalili, S.; Alhussian, H. Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access* **2019**, *7*, 39496–39508. [CrossRef]
11. Rahab, H.; Haouassi, H.; Souidi, M.E.H.; Bakhouche, A.; Mahdaoui, R.; Bekhouche, M. A modified binary rat swarm optimization algorithm for feature selection in Arabic sentiment analysis. *Arab. J. Sci. Eng.* **2023**, *48*, 10125–10152. [CrossRef]
12. Hichem, H.; Elkamel, M.; Rafik, M.; Mesaaoud, M.T.; Ouahiba, C. A new binary grasshopper optimization algorithm for feature selection problem. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 316–328. [CrossRef]
13. Bidar, M.; Mouhoub, M.; Sadaoui, S. Discrete firefly algorithm: A new metaheuristic approach for solving constraint satisfaction problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Cambridge, MA, USA, 2018; pp. 1–8.
14. Simon, D. *Evolutionary Optimization Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
15. Agresti, A. Dealing with discreteness: Makingexact'confidence intervals for proportions, differences of proportions, and odds ratios more exact. *Stat. Methods Med. Res.* **2003**, *12*, 3–21. [CrossRef] [PubMed]
16. Jäntschi, L. Binomial distributed data confidence interval calculation: Formulas, algorithms and examples. *Symmetry* **2022**, *14*, 1104. [CrossRef]

17. Li, X.; Wang, X.; Chen, G. Pinning a complex dynamical network to its equilibrium. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2004**, *51*, 2074–2087. [CrossRef]
18. Vega, C.J.; Sanchez, E.N.; Alzate, R. Inverse optimal pinning control for synchronization of complex networks with nonidentical chaotic nodes. *IFAC-PapersOnLine* **2018**, *51*, 235–239. [CrossRef]
19. Rios, J.D.; Ríos-Rivera, D.; Hernandez-Barragan, J.; Pérez-Cisneros, M.; Alanis, A.Y. Formation Control of Mobile Robots Based on Pin Control of Complex Networks. *Machines* **2022**, *10*, 898. [CrossRef]
20. Xiang, J.; Chen, G. On the V-stability of complex dynamical networks. *Automatica* **2007**, *43*, 1049–1057. [CrossRef]
21. Anderson, E.J.; Ferris, M.C. Genetic algorithms for combinatorial optimization: The assemble line balancing problem. *ORSA J. Comput.* **1994**, *6*, 161–173. [CrossRef]
22. Shahlaei, M. Descriptor selection methods in quantitative structure–activity relationship studies: A review study. *Chem. Rev.* **2013**, *113*, 8093–8103. [CrossRef]
23. Wegner, J.K.; Fröhlich, H.; Zell, A. Feature selection for descriptor based classification models. 1. Theory and GA-SEC algorithm. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 921–930. [CrossRef]
24. Lazzari, F.; Mor, G.; Cipriano, J.; Solsona, F.; Chemisana, D.; Guericke, D. Optimizing planning and operation of renewable energy communities with genetic algorithms. *Appl. Energy* **2023**, *338*, 120906. [CrossRef]
25. Umbarkar, A.J.; Sheth, P.D. Crossover operators in genetic algorithms: A review. *ICTACT J. Soft Comput.* **2015**, *6*. [CrossRef]
26. Yao, X. An empirical study of genetic operators in genetic algorithms. *Microprocess. Microprogram.* **1993**, *38*, 707–714. [CrossRef]
27. Singh, N.; Singh, S. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *J. Appl. Math.* **2017**, *2017*, 2030489. [CrossRef]
28. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
29. Dhiman, G.; Garg, M.; Nagar, A.; Kumar, V.; Dehghani, M. A novel algorithm for global optimization: Rat swarm optimizer. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 8457–8482. [CrossRef]
30. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
31. Mai, X.H.; Wei, D.Q.; Zhang, B.; Luo, X.S. Controlling chaos in complex motor networks by environment. *IEEE Trans. Circuits Syst. II Express Briefs* **2015**, *62*, 603–607. [CrossRef]
32. Solís-Perales, G.; Ruiz-Velázquez, E.; Valle-Rodríguez, D. Synchronization in complex networks with distinct chaotic nodes. *Commun. Nonlinear Sci. Numer. Simul.* **2009**, *14*, 2528–2535. [CrossRef]
33. Lai, Q.; Chen, C.; Zhao, X.W.; Kengne, J.; Volos, C. Constructing chaotic system with multiple coexisting attractors. *IEEE Access* **2019**, *7*, 24051–24056. [CrossRef]
34. Soriano-Sánchez, A.G.; Posadas-Castillo, C.; Platas-Garza, M.A.; Cruz-Hernández, C.; López-Gutiérrez, R.M. Coupling strength computation for chaotic synchronization of complex networks with multi-scroll attractors. *Appl. Math. Comput.* **2016**, *275*, 305–316. [CrossRef]
35. Chen, G.; Ueta, T. Yet another chaotic attractor. *Int. J. Bifurc. Chaos* **1999**, *9*, 1465–1466. [CrossRef]
36. Lü, J.; Chen, G. A new chaotic attractor coined. *Int. J. Bifurc. Chaos* **2002**, *12*, 659–661. [CrossRef]
37. Qi, G.; Chen, G.; Du, S.; Chen, Z.; Yuan, Z. Analysis of a new chaotic system. *Phys. A Stat. Mech. Its Appl.* **2005**, *352*, 295–308. [CrossRef]
38. Elhadj, Z.; Sprott, C.J. The unified chaotic system describing the Lorenz and Chua systems. *Facta Univ.-Ser. Electron. Energetics* **2010**, *23*, 345–355. [CrossRef]