Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.031947

ARTICLE

Binary Archimedes Optimization Algorithm for Computing Dominant Metric Dimension Problem

Basma Mohamed[1,*], **Linda Mohaisen**[2] **and Mohammed Amin**[1]

[1]Mathematics and Computer Science Department, Faculty of Science, Menoufia University, Shebin Elkom, 32511, Egypt

[2]Faculty of Computer and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia

*Corresponding Author: Basma Mohamed. Email: bosbos25jan@yahoo.com

Received: 01 May 2022 Accepted: 14 October 2022 Published: 26 January 2024

ABSTRACT

In this paper, we consider the NP-hard problem of finding the minimum dominant resolving set of graphs. A vertex set B of a connected graph G resolves G if every vertex of G is uniquely identified by its vector of distances to the vertices in B. A resolving set is dominating if every vertex of G that does not belong to B is a neighbor to some vertices in B. The dominant metric dimension of G is the cardinality number of the minimum dominant resolving set. The dominant metric dimension is computed by a binary version of the Archimedes optimization algorithm (BAOA). The objects of BAOA are binary encoded and used to represent which one of the vertices of the graph belongs to the dominant resolving set. The feasibility is enforced by repairing objects such that an additional vertex generated from vertices of G is added to B and this repairing process is iterated until B becomes the dominant resolving set. This is the first attempt to determine the dominant metric dimension problem heuristically. The proposed BAOA is compared to binary whale optimization (BWOA) and binary particle optimization (BPSO) algorithms. Computational results confirm the superiority of the BAOA for computing the dominant metric dimension.

KEYWORDS

Dominant metric dimension; archimedes optimization algorithm; binary optimization alternate snake graphs

1 Introduction

The primary metric dimension of graphs was just recently introduced in [1]. Based on domination theory and graph resolvability theory, the dominating metric dimension. Let $G = (V, E)$ be a connected graph and $d(u, v)$ be the shortest path between two vertices $u, v \in V(G)$. A resolving set of G is an ordered vertex set $B = \{x_1, x_2, \ldots, x_k\} \subseteq V(G)$ if the representation

$$r(v|B) = (d(v, x_1), d(v, x_2), \ldots, d(v, x_k))$$

is different for each $v \in V(G)$. A resolving set B is a dominating set of G if every vertex of $V \backslash B$ has at least one neighbor that belongs to B. The metric dimension of G, abbreviated dim (G), is the cardinality number of the smallest resolving set. The dominating metric dimension of G, abbreviated Ddim(G), is the cardinality number of the smallest dominating resolving set.

Example 1. The star graph S_7 is given in Fig. 1. The set $B = \{v_2, v_3, v_4, v_5, v_6\}$ is a minimal resolving set but not a dominating set of S_7 since v_7 is not adjacent to vertices in B. The set $\overline{B} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ is a minimal dominant resolving set of S_7. Thus, $dim(S_7) = 5$ and $Ddim(S_7) = 6$.

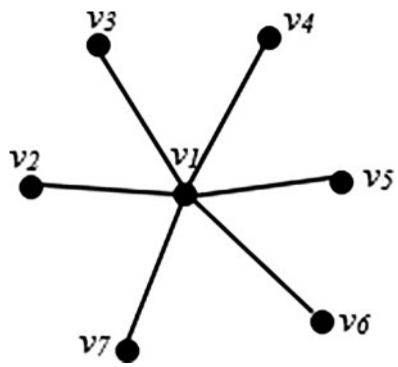

Figure 1: Star graph S_7

Both the metric dimension problem and the problem with the dominant set are NP-complete problems [2,3]. As a result, finding whether Ddim$(G) \leq K$ for a given graph G and input K is a typical NP-complete problem for the dominating metric dimension of G. Wireless communication networks, electrical networks, commercial networks, and chemical structures are all examples of networks that apply the dominance theory [4,5]. In order to overcome the problem of uniquely locating an intruder in a network, a minimal resolving set of a graph has been introduced in [6,7]. The concept of the smallest resolving set of a graph serving as the metric basis and its cardinality number serving as the metric dimension were independently introduced by the authors in [3].

The metric dimension is determined theoretically for several graphs in the literature [8–20]. On the other hand, a few algorithms have been proposed in the literature to compute heuristically the metric dimension. These are genetic algorithm [21], particle swarm optimization [22], and variable neighborhood search [23].

The dominant metric dimension is studied in [1,24]. In [1], the dominant metric dimension of path graph P_n, cycle graph C_n, star graph S_n, complete graph K_n, and complete bipartite graph $K_{m,n}$ are theoretically determined. It has been shown that $Ddim(P_n)$, $n = 1, 2$ is 1, $Ddim(P_n)$, $n > 4$ is $\left\lceil \frac{n}{3} \right\rceil$, $Ddim(C_n)$, $n \geq 7$ is $\left\lceil \frac{n}{3} \right\rceil$, $Ddim(S_n)$, $n \geq 2$ is $n-1$, $Ddim(K_n)$, $n \geq 2$ is $n-1$, and $Ddim\ (K_{m,n})$, m, $n \geq 2$ is $m + n - 2$. In [24], the dominant metric dimension of the corona product graph of G and H is investigated whenever H is a path graph P_n, cycle graph C_n, complete bipartite graph $K_{m,n}$, complete graph K_n and star graph S_n. Also see more details in the literature [25–30].

The smallest dominating resolving set of graphs is being calculated heuristically for the first time in this study. To resolve the problem, we modify the operations of a binary version of the Archimedes optimization algorithm (BAOA). The theoretically generated graph results are used to test the proposed BAOA. On various graphs and theoretically generated graphs, the proposed algorithm is compared with competing algorithms.

The paper is organized as follows: Section 2 gives an overview of Archimedes optimization algorithm (AOA). Section 3 gives the BAOA for computing the dominant metric dimension. Section 4 reports computational results. Finally, the conclusion is stated in Section 5.

2 Archimedes Optimization Algorithm (AOA)

AOA is a physics-inspired algorithm, specifically Archimedes' law. This algorithm, which is a member of the meta-heuristics class, was developed by Hashim et al. [31]. The uniqueness of this algorithm is found in the way the solution is encoded, which includes three auditory signals for the basic agents: volume (V), density (D) and acceleration (Γ). As a result, a random number generator creates the initial group of agents in Dim dimensions. As additive data, random values of V, D, and are shown. After that, each item is evaluated to determine which is the best (O_{best}).

During the AOA process, updates to density and volume change the acceleration based on the collision notion between objects. The general AOA steps are as follows:

The first step—Initialization: Initialize the positions of all objects using (1):

$$O_i = lb_i + rand \times (ub_i - lb_i);\ i = 1, 2, \ldots, N \tag{1}$$

where O_{i} represents the i^{th} object among N total objects. The terms lb_{i} and ub_{i}, respectively, stand for the lower and upper bounds of the search space.

Use (2) to specify the volume (vol) and density (den) for each i^{th} object:

$$den_i = rand$$

$$vol_i = rand \tag{2}$$

The acceleration (acc) of the i^{th} object is then initialized using the Eq. (3), where rand is a D-dimensional vector that creates a random number between 0 and 1.

$$\mathrm{acc_i} = lb_i + rand \times (ub_i - lb_i) \tag{3}$$

In this stage, evaluate the starting population and select the object with the best fitness value. x_{best}, den_{best}, vol_{best}, and acc_{best} should be assigned.

The second step—Update densities and volumes: The density and volume of object i for the iteration $t+1$ are modified by (4):

$$den_i^{t+1} = den_i^t + rand \times \left(den_{best} - den_i^t\right)$$

$$vol_i^{t+1} = vol_i^t + rand \times \left(vol_{best} - vol_i^t\right) \tag{4}$$

where $rand$ stands for a random number with a uniform distribution, and vol_{best} and den_{best} represent the volume and density associated with the best item discovered so far.

The third stage is the density scalar and transfer coefficient: In this phase, objects collide with one another until equilibrium is achieved. Switching from exploration to exploitation mode is the primary goal of the transfer function (T_{c}), according to Eq. (5):

$$T_c = \exp\left(\frac{t - T}{T}\right) \tag{5}$$

The maximum number of iterations is T, and T_c grows exponentially over time until it reaches 1. t stands for the current iteration. Additionally, the reduction in density scalar d_s in AOA enables the use of Eq. (6) to find the best solution:

$$d_s^{t+1} = \exp\left(\frac{t-T}{T}\right) - \left(\frac{t}{T}\right) \tag{6}$$

The fourth step-Exploration phase: uses a random selection of material (Mr) to bring agents into contact with one another. In cases when the transfer function value is less than or equal to 0.5, acceleration objects are updated using Eq. (7).

$$\Gamma_i^{t+1} = \frac{D_{Mr} + V_{Mr} \times \Gamma_{Mr}}{D_i^{t+1} \times V_i^{t+1}} \tag{7}$$

The fifth step-Phase of exploitation: This phase does not result in an agent collision. Eq. (8) is used to update acceleration objects when the transfer coefficient value is greater than 0.5.

$$\Gamma_i^{t+1} = \frac{D_{Best} + V_{Best} \times \Gamma_{Best}}{D_i^{t+1} \times V_i^{t+1}} \tag{8}$$

where Γ_{Best} denotes the acceleration of the optimal object O_{Best}.

The sixth phase-Normalization of acceleration: In this stage, normalize the acceleration in order to determine the rate of change using Eq. (9):

$$\Gamma_{i-norm}^{t+1} = \alpha \times \frac{\Gamma_i^{t+1} - \Gamma^{Min}}{\Gamma^{Max} - \Gamma^{Min}} + \beta \tag{9}$$

where α and β are constants of 0.9 and 0.1, respectively. The Γ_{i-norm}^{t+1} defines the percentage of steps that each agent will change. The higher value of acceleration means that the object realizes the operation of exploration; or else, the exploitation mode is active.

The seventh step—Update process: In the exploration phase ($T_c \leq 0.5$), Eq. (10) updates the position of the i^{th} object in iteration $t+1$, whereas in the exploitation phase ($T_c > 0.5$), Eq. (11) updates the position of the object.

$$O_i^{t+1} = O_i^t + c_1 \times r_5 \times \Gamma_{i-norm}^{t+1} \times d_s \times (O_{rand} - O_i^t) \tag{10}$$

where c_1 equals 2.

$$O_i^{t+1} = O_{Best}^t + F \times c_2 \times r_6 \times \Gamma_{i-norm}^{t+1} \times d_s \times (\delta \times O_{Best} - O_i^t) \tag{11}$$

where c_2 is equal to 6.

The parameter δ is positively correlated with time and this parameter is proportionally linked to the transfer coefficient T_c, i.e., $\delta = 2 \times T_c$. The main role of this parameter is to maintain a proper balance between exploration and exploitation operations. The margin between the best object and the other object is higher during the first iterations, resulting in a high random walk. However, in the final iterations, the margin will be decreased and provide a low random walk.

F is used for flagging, while Eq. (12) is utilized to determine the direction of the search:

$$F = \begin{cases} +1 & if\ P \leq 0.5 \\ -1 & if\ P \geq 0.5 \end{cases} \tag{12}$$

where $P = 2 \times \text{rand} - C_4$.

The eighth step is **evaluation,** where we utilize the score index Sc to assess the new population and other data such as D_{Best}, V_{Best}, and Γ_{Best} to identify the best objects.

3 Binary Archimedes Optimization Algorithm for Dominant Metric Dimension

Because it maintains a population of solutions and examines a vast area to find the best global solution, the Archimedes optimization algorithm can solve difficult optimization problems with numerous locally optimal solutions. This benefit enables the binary version of the algorithm to be applied to the dominant metric dimension problem. Using position vectors in the continuous real domain, objects can navigate the search space in the continuous version of AOA. By using an S-shaped transfer function to change the continuous variable AOA into a binary one, we may convert it into binary values. Position changes in discrete binary search space necessitate flipping between 0 and 1.

The following equation is used in the initialization step:

$$Obinary_{ij} = \begin{cases} 1 & rand() > 0.5 \\ 0, & else \end{cases} \tag{13}$$

where a rand is a random number between 0 and 1.

A transfer function is used to be able to map continuous values to binary ones. In this study, the sigmoid function (S) is used as follows:

$$S = \frac{1}{1 + e^{-10x^d}} \tag{14}$$

where x^d indicates the continuous-valued position at dimension d and S is the function output. The following equation is used to generate a binary value.

$$Obinary_{ij} = \begin{cases} 1 & rand() < S \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The proposed algorithm deals with the dominant resolving set problem as an optimization problem where it searches for the best solution, so each object can be represented as a one-dimensional vector $Obinary_{\mathrm{ij}} = (O_{i1}, O_{i2}, \ldots, O_{ij})$, $Obinary_{ij}$ is a binary-valued position vector if the j-th element of the vector has a value of 1, it means that vertex j belongs to B. If every $v \in V\ (G)$ has a distinct representation $r(v|B)$, then B is a dominant resolving set. The value of a binary-valued position vector is produced by computing the value of the S-shaped transfer function. In the BAOA algorithm, when an object is not feasible as a dominant resolving set, that object is repaired by adding a vertex from $V \backslash B$. This repair is applied until that object becomes a dominant resolving set.

Each solution in the population is represented by the algorithm as a string of binary values, where 1 indicates that the dominant resolving set will be chosen, in which case the corresponding value will be "1," and 0 indicates that the dominant resolving set will not be chosen, in which case the corresponding value will be "0".

Thus, the flowchart of the proposed BAOA algorithm is displayed in Fig. 2 and the pseudocode in Algorithm 1, respectively.

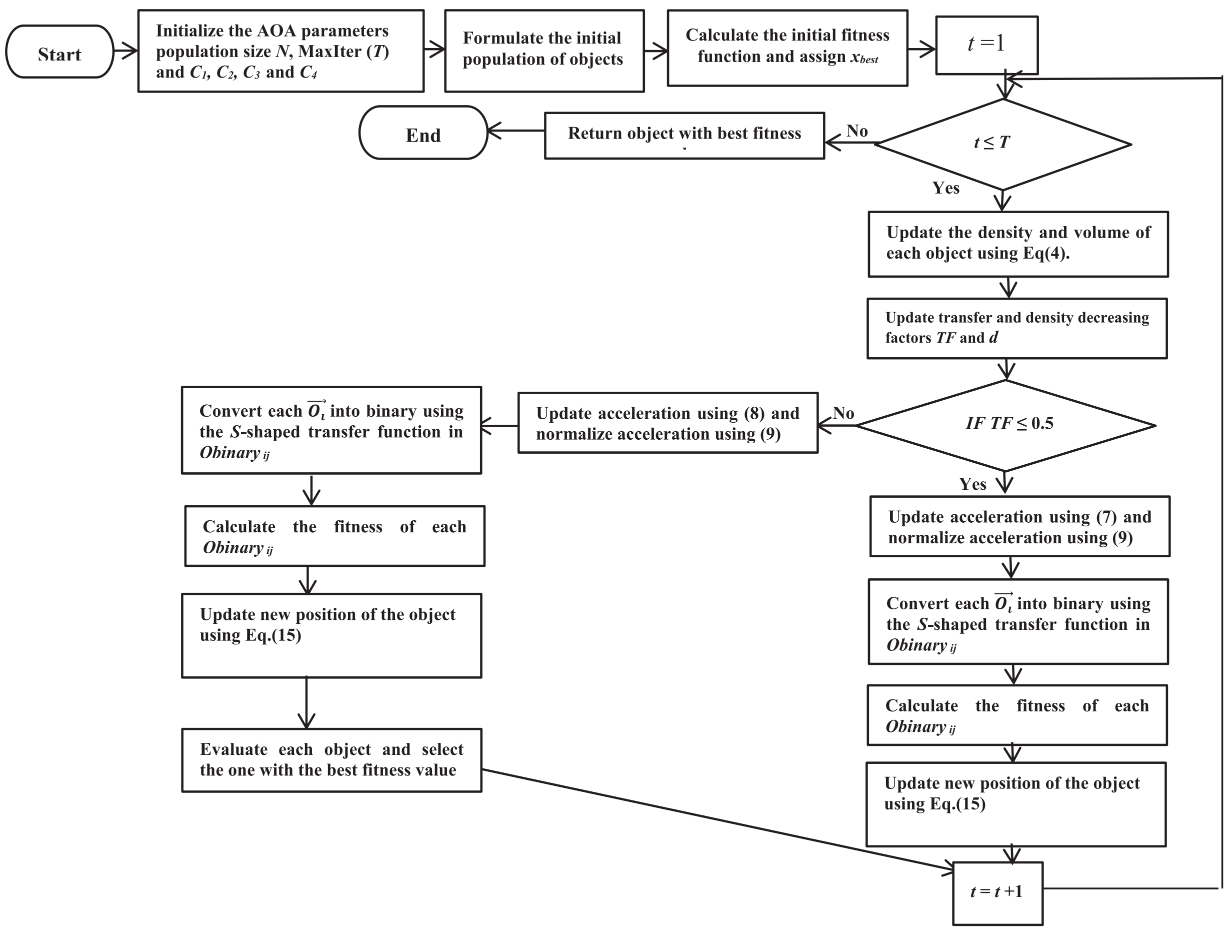

Figure 2: The flowchart of BAOA

Algorithm 1: Pseudocode of BAOA

Procedure AOA (population size N, maximum number of iterations T, C_1, C_2, C_3 and C_4)

  Initialize the positions of all objects O_i randomly, volumes (vol) and densities (den) as shown in (1)–(3), respectively.

  Evaluate the initial population and select the one with the best fitness value.

  Set iteration counter $t = 1$

While $t \leq T$

  for each object i **do**

    Update the density and volume of each object using (4).

    Update transfer and density decreasing factors TF and d as shown in (5) and (6), respectively.

// Exploration phase //

  If $TF \leq 0.5$ then

(Continued)

Algorithm 1 (continued)

Update acceleration by using (7) and normalize acceleration by using (9)
Update position using (10)
Convert each $\overrightarrow{O_i}$ into binary using the S-shaped transfer function in $Obinary_{ij}$
Calculate the fitness of each $Obinary_{ij}$
Update the new position of the object using (15)
// Exploitation phase //
else
Update acceleration using (8) and normalize acceleration using (9)
Update direction flag F using (12)
Update position using (11)
Convert each $\overrightarrow{O_i}$ into binary using the S-shaped transfer function in $Obinary_{ij}$
Calculate the fitness of each $Obinary_{ij}$
Update the new position of the object using (15)
end if
end for
Evaluate each object and select the one with the best fitness value.
Set $t = t + 1$
end while
return object with best fitness value
end procedure

4 Experimental Results

In this section, the proposed BAOA is tested using graph results that are computed theoretically. The proposed BAOA is compared to the BWOA and BPSO on a complete graph, a star graph, a path graph, an alternate triangular snake with pendant edge graph, and an alternate quadrilateral snake graph.

The algorithm tests and comparisons were performed on a Windows 10 Ultimate 64-bit operating system; the processor was an Intel Core i7 running at 16 GB of RAM, the hard drive was 1TBHDD+1TBSSD, and the code was implemented in MATLAB 2021b. The parameter setting values are presented in Table 1.

Table 1: Parameter setting

Algorithms	Parameter name	Value
	Objects number	30
BAOA	Max iteration	500
	C_1 C_2 C_3 C_4	[2, 6, 2, 0.5]
	Number of runs	20
	Whales number	30
	a	Decrease from 2 to 0
BWOA	a_2	Decrease from −1 to −2

(Continued)

Table 1 (continued)

Algorithms	Parameter name	Value
	Max iteration	500
	Number of runs	20
BPSO	Swarm size	30
	C_1	Increases linearly from 0.5 to 2.5
	C_2	Decreases linearly from 2.5 to 0.5
	Inertia weight (w)	0.8
	Max iteration	500
	Number of runs	20

The BAOA, BWOA and BPSO have been run 20 times for each instance, and the results are summarized in Tables 2–5. The tables are organized as follows:

- The first three columns contain the test instance name, the number of nodes and edges, respectively.
- The fourth column contains the BAOA best solution (named BAOA $_{\text{best}}$) obtained in 20 runs;
- The average execution time (t) used to reach the final BAOA solution for the first time is given in the fifth column.
- The sixth column contains the average number of generations for finishing BAOA $_{\text{best}}$.
- The seven and the eighth column variance and standard deviation contain information on the average solution quality.

Table 2: Results on complete graph K_k

Instance	n	m	BAOA $_{\text{best}}$	t (s)	Iteration (generation)	BWOA	t	Iteration	BPSO	t	Iteration
K_1	3	3	2	**2.85**	1	2	9.28	1	2	13.71	1
K_2	4	6	3	**9.11**	1	3	32.65	4	3	29.28	3
K_3	5	10	4	**24.02**	1	4	78.92	8	4	42.13	17
K_4	6	15	5	73.68	3	5	95.24	9	5	58.16	5
K_5	7	21	6	104.21	5	6	147.83	13	6	93.04	22
K_6	8	28	7	**168.76**	8	7	192.39	5	7	184.21	14
K_7	9	36	8	**201.57**	2	8	209.45	21	8	256.09	39
K_8	10	45	9	316.39	9	9	**284.81**	15	9	338.05	7
K_9	11	55	10	**413.10**	3	10	525.32	18	10	480.19	125
K_{10}	12	66	11	**481.35**	7	11	742.99	141	11	623.53	10
K_{11}	13	78	12	**530.49**	13	12	881.13	27	12	735.28	21
K_{12}	14	91	13	**599.11**	6	13	1022.19	9	13	913.11	14
K_{13}	15	105	14	**681.03**	8	14	1273.44	31	14	1319.72	29
K_{14}	16	120	15	**746.34**	12	15	1379.15	116	15	1468.03	134
K_{15}	17	136	16	**563.12**	9	16	1428.23	54	16	1704.09	22
K_{16}	18	153	17	**725.71**	11	17	1592.71	12	17	1811.14	17
K_{17}	19	171	18	**914.93**	7	18	1714.83	37	18	1525.54	26
K_{18}	20	190	19	**1051.65**	21	19	1995.03	45	19	1761.28	32

(Continued)

Table 2 (continued)

Instance	n	m	BAOA $_{best}$	t (s)	Iteration (generation)	BWOA	t	Iteration	BPSO	t	Iteration
K_{19}	21	210	20	**1195.27**	19	20	2168.25	132	20	2056.87	71
K_{20}	22	231	21	**1034.09**	14	21	2409.12	96	21	2238.05	57

Table 3: Results on star graph S_k

Instance	n	m	BAOA $_{best}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
S_1	3	2	2	**6.23**	1	2	7.54	1	2	8.19	1
S_2	4	3	3	**25.18**	1	3	34.02	7	3	31.4	2
S_3	5	4	4	**69.01**	3	4	81.95	3	4	72.79	5
$S_{4.}$	6	5	5	127.39	2	5	**113.64**	9	5	146.71	11
S_5	7	6	6	**215.86**	9	6	239.36	17	6	267.25	8
S_6	8	7	7	**282.74**	1	7	418.14	22	7	429.41	15
S_7	9	8	8	**378.99**	4	8	547.58	36	8	514.99	27
S_8	10	9	9	**491.10**	2	9	623.72	14	9	602.15	31
S_9	11	10	10	**605.84**	3	10	735.91	8	10	699.73	23
S_{10}	12	11	11	762.46	3	11	870.73	27	11	**741.68**	109
S_{11}	13	12	12	836.84	8	12	954.23	10	12	**802.31**	4
S_{12}	14	13	13	**1009.17**	6	13	1063.15	79	13	1196.16	21
S_{13}	15	14	14	**1161.22**	11	14	1198.82	126	14	1278.52	117
S_{14}	16	15	15	**942.91**	9	15	1286.19	18	15	1193.04	28
S_{15}	17	16	16	**1157.23**	15	16	1335.08	22	16	1402.15	142
S_{16}	18	17	17	**1208.12**	13	17	1493.12	135	17	1519.82	32
S_{17}	19	18	18	**1379.44**	10	18	1586.31	28	18	1664.98	108
S_{18}	20	19	19	**1418.53**	17	19	1768.53	151	19	1575.34	83
S_{19}	21	20	20	**1246.37**	14	20	1913.22	87	20	1789.61	45
S_{20}	22	21	21	**1105.14**	15	21	2108.38	52	21	1902.27	36

Table 4: Results on path graph

Instance	n	m	BAOA $_{best}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
P_3	3	2	2	**1.17**	1	2	2.54	1	2	4.31	1
P_4	4	3	2	**3.99**	1	2	7.89	4	2	5.76	8
P_5	5	4	2	14.45	1	2	23.92	5	2	**12.04**	3
P_6	6	5	2	**48.86**	3	2	72.94	13	2	80.65	7
P_7	7	6	3	**83.91**	1	3	95.78	9	3	109.12	11
P_8	8	7	3	**107.29**	4	3	186.15	25	3	128.47	6
P_9	9	8	3	**154.35**	12	3	268.29	18	3	192.71	27
P_{10}	10	9	3	**232.13**	7	3	311.05	30	3	275.64	19
P_{11}	11	10	4	**309.11**	15	4	356.48	21	4	334.92	123
P_{12}	12	11	4	413.47	9	4	474.21	116	4	**401.13**	75
P_{13}	13	12	4	**509.16**	3	4	591.09	27	4	618.41	38
P_{14}	14	13	5	**484.78**	17	5	616.37	41	5	761.09	20
P_{15}	15	14	5	**659.36**	8	5	894.12	135	5	882.56	113

(Continued)

Table 4 (continued)

Instance	n	m	BAOA $_{\text{best}}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
P_{16}	16	15	5	**733.14**	14	5	1025.81	22	5	907.18	29
P_{17}	17	16	6	**898.07**	10	6	1136.45	117	6	1198.07	12
P_{18}	18	17	6	**951.23**	4	6	1417.93	29	6	1576.51	8
P_{19}	19	18	6	**874.59**	9	6	1641.18	26	6	1449.27	17
P_{20}	20	19	7	**1078.45**	2	7	1918.04	18	7	1685.93	24

Table 5: Results on alternate triangular snake with pendant edge

Instance	n	m	BAOA $_{\text{best}}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
$A(TS_1)$	5	5	2	**31.83**	1	2	55.27	4	2	76.12	2
$A(TS_2)$	8	9	2	**69.91**	1	2	93.12	6	2	108.45	5
$A(TS_3)$	11	13	3	135.78	4	3	**122.36**	5	3	157.81	9
$A(TS_4)$	14	17	5	**197.51**	3	5	241.35	17	5	212.38	12
$A(TS_5)$	17	21	6	**312.65**	7	6	349.02	9	6	375.95	27
$A(TS_6)$	20	25	7	501.14	9	7	**434.17**	16	7	487.34	10
$A(TS_7)$	23	29	8	**617.01**	5	8	694.62	29	8	643.29	22
$A(TS_8)$	26	33	9	**833.25**	8	9	908.53	34	9	871.43	15
$A(TS_9)$	29	37	10	1017.42	10	10	1213.44	18	10	**955.82**	42
$A(TS_{10})$	32	41	11	1182.61	10	11	1396.18	27	11	**1093.24**	19
$A(TS_{11})$	35	45	12	**1256.09**	15	12	1432.32	44	12	1327.91	32
$A(TS_{12})$	38	49	13	**1317.24**	3	13	1547.49	19	13	1466.80	11
$A(TS_{13})$	41	53	14	**1388.31**	11	14	1714.18	131	14	1534.11	17
$A(TS_{14})$	44	57	15	**1492.19**	14	15	1848.71	96	15	1675.86	35
$A(TS_{15})$	47	61	16	**1307.01**	12	16	1954.35	17	16	1811.99	122
$A(TS_{16})$	50	65	17	**1425.29**	18	17	2078.11	39	17	1927.52	18
$A(TS_{17})$	53	69	18	**1682.12**	11	18	2215.46	42	18	2083.28	29
$A(TS_{18})$	56	73	19	**1751.73**	16	19	2399.28	29	19	2204.39	24
$A(TS_{19})$	59	77	20	**1857.05**	2	20	2523.13	34	20	2391.15	141
$A(TS_{20})$	62	81	21	**1925.14**	11	21	2796.17	22	21	2503.57	27
$A(TS_{21})$	65	85	22	**2037.43**	19	22	2981.43	153	22	2655.29	36
$A(TS_{22})$	68	89	23	**2122.36**	26	23	3149.24	40	24	2810.92	48
$A(TS_{23})$	71	93	24	**2287.61**	19	24	3211.15	25	24	2673.42	21
$A(TS_{24})$	74	97	25	**1955.82**	1	25	3319.28	113	25	2716.19	19
$A(TS_{25})$	77	101	26	**1873.19**	17	26	3584.93	21	27	2931.43	32
$A(TS_{26})$	80	105	27	**2091.24**	19	27	3706.21	32	28	2815.95	120
$A(TS_{27})$	83	109	28	**2242.13**	19	29	3851.75	68	28	3099.22	29
$A(TS_{28})$	86	113	29	**2330.05**	19	30	3638.44	45	31	3186.36	23
$A(TS_{29})$	89	117	30	**2473.21**	19	31	3992.65	36	32	3319.74	31
$A(TS_{30})$	92	121	31	**2549.75**	19	34	4065.22	30	32	3479.15	22
$A(TS_{31})$	95	125	32	**2618.92**	18	35	4198.13	42	33	3611.89	38
$A(TS_{32})$	98	129	33	**2796.28**	20	36	4282.57	33	35	3843.54	25
$A(TS_{33})$	101	133	34	**2874.32**	2	37	4419.16	19	36	3954.93	16

Our stopping criterion is the cardinality of the dominant resolving set that reaches the known dominant metric dimension of the complete graph. BAOA takes 168.76 s on K_6, and it takes 8 iterations to complete BAOA to achieve the best solution.

Regarding BAOA results, Table 3 shows that for the star graph S_k,$1 \leq k \leq 20$, BAOA has reached an optimal solution. For example, in S_3, the time needed for BAOA is 69.01 s and reaches the best solution after 3 iteration. In [1], the dominant metric dimension for a path graph, a complete graph, and a star graph is theoretically determined.

Regarding BAOA results, Table 4 shows that for path graph P_n, $3 \leq n \leq 20$, BAOA has reached an optimal solution. For example, in P_6, the time required for BAOA is 48.86 s, with 3 iterations required to achieve the best solution.

BAOA found an optimal solution for the alternate triangular snake with pendant edge $A(TS_k)$, $1 \leq k \leq 33$, as shown in Table 5. For example, in $A(TS_4)$, the time required for BAOA is 197.51 s, with 3 iterations required to achieve the best solution.

Regarding BAOA results, Table 6 shows that alternate quadrilateral snake $A(QS_k)$, $1 \leq k \leq 33$, BAOA has reached an optimal solution. For example, $A(QS_3)$, the time needed for BAOA is 205.23 s and reaches the best solution after 2 iterations.

Table 6: Results on alternate quadrilateral snake

Instance	n	m	BAOA $_{\text{best}}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
$A(QS_1)$	5	5	3	**71.49**	1	3	108.63	1	3	95.12	1
$A(QS_2)$	9	10	4	154.17	1	4	191.27	6	4	**128.54**	3
$A(QS_3)$	13	15	5	**205.23**	2	5	287.15	4	5	292.36	9
$A(QS_4)$	17	20	7	**298.38**	5	7	561.12	9	7	523.49	21
$A(QS_5)$	21	25	**7**	**412.64**	8	7	643.38	22	7	678.22	14
$A(QS_6)$	25	30	8	**509.81**	17	8	801.79	35	8	835.69	18
$A(QS_7)$	29	35	9	**647.15**	44	9	982.46	19	9	923.15	30
$A(QS_8)$	33	40	10	803.59	18	10	**737.12**	52	10	1174.38	23
$A(QS_9)$	37	45	11	**917.42**	36	11	1276.84	43	11	1305.47	19
$A(QS_{10})$	41	50	12	**1043.18**	25	12	1421.75	28	12	1398.71	6
$A(QS_{11})$	45	55	13	**1205.49**	51	13	1538.52	152	13	1512.87	35
$A(QS_{12})$	49	60	14	**1316.32**	37	14	1673.21	63	14	1609.53	41
$A(QS_{13})$	53	65	15	**1450.76**	46	15	1825.43	49	15	1715.19	64
$A(QS_{14})$	57	70	16	**1561.12**	34	16	1913.92	38	16	1873.14	112
$A(QS_{15})$	61	75	17	**1696.85**	42	17	2056.27	106	17	2119.25	88
$A(QS_{16})$	65	80	18	**1783.09**	45	18	2172.49	47	18	2205.82	95
$A(QS_{17})$	69	85	19	**1832.24**	42	20	2328.72	134	19	2289.07	61
$A(QS_{18})$	73	90	20	**1920.11**	46	20	2416.41	52	20	2397.28	49
$A(QS_{19})$	77	95	21	**1752.89**	1	21	2673.12	29	21	2473.16	18
$A(QS_{20})$	81	100	22	**1898.13**	42	22	2768.95	46	22	2583.55	105
$A(QS_{21})$	85	105	23	**2016.47**	17	23	2995.64	37	23	2839.31	22
$A(QS_{22})$	89	110	24	**2158.22**	44	24	3082.18	55	24	2765.24	53
$A(QS_{23})$	93	115	25	**2311.54**	40	25	3176.27	61	25	2932.89	47
$A(QS_{24})$	97	120	26	**2483.77**	43	26	3287.34	42	26	3139.27	19
$A(QS_{25})$	101	125	27	**2613.10**	3	27	3375.16	74	27	3290.58	12
$A(QS_{26})$	105	130	28	**2791.64**	11	28	3592.28	25	28	3367.12	28

(Continued)

Table 6 (continued)

Instance	n	m	BAOA $_{best}$	t (s)	Iteration	BWOA	t	Iteration	BPSO	t	Iteration
$A(QS_{27})$	109	135	29	**2528.19**	2	29	3787.12	36	29	3515.44	17
$A(QS_{28})$	113	140	30	**2816.15**	39	30	3971.47	58	30	3636.78	144
$A(QS_{29})$	117	145	31	**2943.37**	26	31	4218.38	22	31	3849.25	25
$A(QS_{30})$	121	150	32	**3096.12**	43	32	4416.22	29	32	3698.47	33
$A(QS_{31})$	125	155	33	**3179.65**	1	33	4594.19	84	33	3953.13	40
$A(QS_{32})$	129	160	34	**3202.94**	2	34	4683.85	23	34	4165.73	25
$A(QS_{33})$	133	165	35	**3341.59**	4	35	4925.03	49	35	4379.24	32

Tables 2–6 display the results for various graphs, which show that the proposed BAOA can achieve the best optimal solution (known dominant metric dimension) in a reasonable amount of time, especially for the path graph, complete graph and star graph. It proves the correctness and superiority of the proposed BAOA.

Experiments in this paper are performed on a subset of complete graph instances with $n \le 22$ and $m \le 231$ in Table 2, star graph instances with $n \le 22$ and $m \le 21$ in Table 3, path graph instances with $n \le 20$ and $m \le 19$ in Table 4, alternate triangular snake with pendant edge graph instances with $n \le 101$ and $m \le 133$ in Table 5, and alternate quadrilateral snake graph instances with $n \le 133$ and $m \le 165$ in Table 6.

Figs. 3–7 show the superiority of the proposed BAOA according to the dominant metric dimension. For example, the dominant metric dimension by BAOA for K_2 is 3 and reaches 9.11 s. P_7 is 3 and reaches 83.91 s. All figures show the superiority of the proposed BAOA.

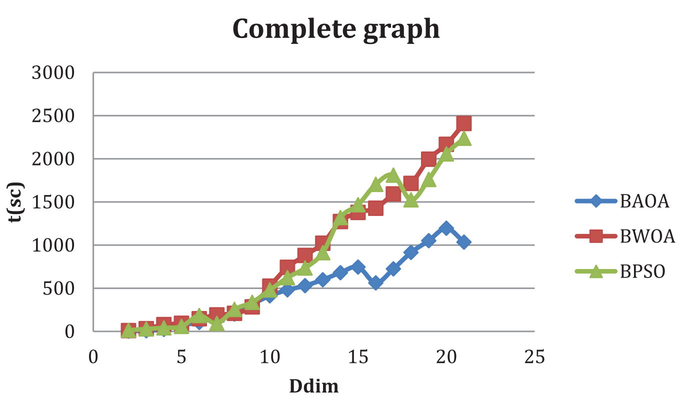

Figure 3: Comparison between BAOA $_{best}$ and t (s) for computing the dominant metric dimension of a complete graph

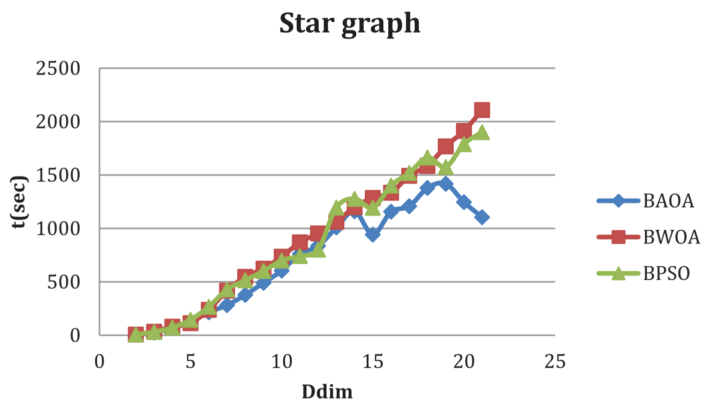

Figure 4: Comparison between BAOA $_{best}$ and t (s) for computing the dominant metric dimension of a star graph

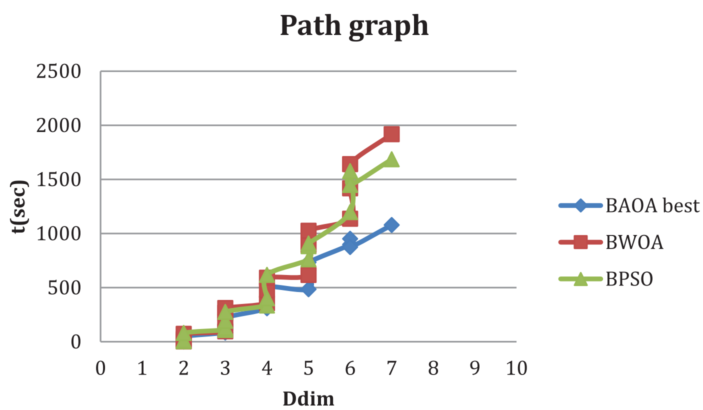

Figure 5: Comparison between BAOA $_{best}$ and t (s) for computing the dominant metric dimension of a path graph

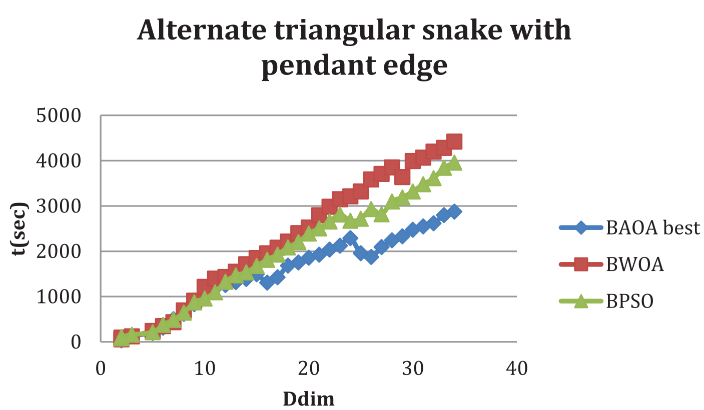

Figure 6: Comparison between BAOA $_{best}$ and t (s) for computing the dominant metric dimension of an alternate triangular snake with pendant edge graph

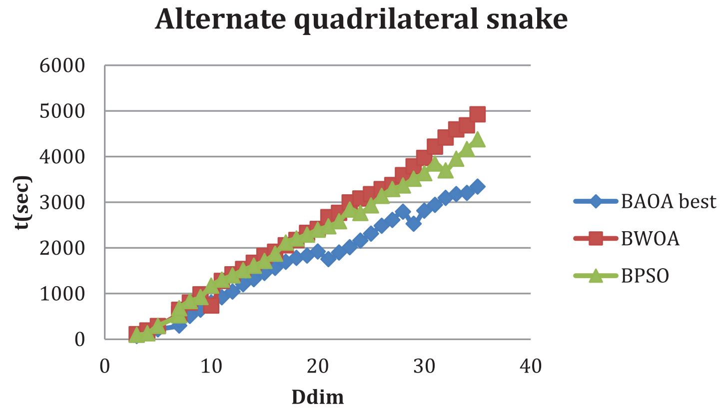

Figure 7: Comparison between BAOA $_{\text{best}}$ and *t* (s) for computing the dominant metric dimension of an alternate quadrilateral snake graph

5 Conclusion

In this paper, the operations of a binary version of the Archimedes optimization algorithm BAOA are adapted to solve the dominant metric dimension problem. The proposed BAOA is tested using graph results that are computed theoretically. The proposed algorithm is compared to competitive algorithms on graphs that are computed theoretically and other graphs. The performance of the proposed BAOA outperforms that of the BWOA and BPSO.

An Open Problem. Other efficient metaheuristic algorithms for determining any variant of metric dimension that does not compute the previous heuristically for any regular graph or planar graph, as well as comparing them to competitive algorithms.

Acknowledgement: This work was not supported by any funding.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Conceptualization, methodology, writing review and formal analysis, Basma Mohamed; investigation, resources, Linda Mohaisen; writing–original draught preparation, validation, editing and visualization, Mohammed Amin. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials: The data underlying the results presented in the study are available within the manuscript.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Susilowati, I. Sa'adah, R. Z. Fauziyyah and A. Erfanian, "The dominant metric dimension of graphs," *Heliyon*, vol. 6, no. 3, pp. e03633, 2020.
[2] M. R. Garey, "A guide to the theory of NP-completeness," in *Computers and Intractability*, 1979.
[3] F. Harary and R. A. Melter, "On the metric dimension of a graph," *ARS Combinatoria*, vol. 2, pp. 191–195, 1976.

[4] J. L. Hurink and T. Nieberg, “Approximating minimum independent dominating sets in wireless networks,” *Information Processing Letters*, vol. 109, no. 2, pp. 155–160, 2008.
[5] A. H. Karbasi and R. E. Atani, “Application of dominating sets in wireless sensor networks,” *International Journal of Security and its Applications*, vol. 7, no. 4, pp. 185–202, 2013.
[6] D. Vukičević and A. Klobučar, “K-dominating sets on linear benzenoids and on the infinite hexagonal grid,” *Croatica Chemica Acta*, vol. 80, no. 2, pp. 187–191, 2007.
[7] P. J. Slater, “Leaves of trees,” *Congressus Numerantium*, vol. 14, pp. 549–559, 1975.
[8] S. Akhter and R. Farooq, “Metric dimension of fullerene graphs,” *Electron, Journal of Graph Theory and Applications*, vol. 7, no. 1, pp. 91–103, 2019.
[9] T. Vetrík, “On the metric dimension of directed and undirected circulant graphs,” *Discussiones Mathematicae: Graph Theory*, vol. 40, no. 1, pp. 67–76, 2020.
[10] S. Nawaz, M. Ali, M. A. Khan and S. Khan, “Computing metric dimension of power of total graph,” *IEEE Access*, vol. 9, pp. 74550–74561, 2021.
[11] S. Nazeer, M. Hussain, F. A. Alrawajeh and S. Almotairi, “Metric dimension on path-related graphs,” *Mathematical Problems in Engineering*, vol. 2021, pp. 1–12, 2021.
[12] M. Mulyono and W. Wulandari, “The metric dimension of friendship graph F_n, lollipop graph $L_{m,n}$ and petersen graph $P_{n,m}$,” *Bulletin of Mathematics*, vol. 8, no. 2, pp. 117–124, 2016.
[13] M. F. Nadeem, S. Qu, A. Ahmad and M. Azeem, “Metric dimension of some generalized families of toeplitz graphs,” *Mathematical Problems in Engineering*, vol. 2022, pp. 1–10, 2022.
[14] A. N. Koam, A. Ahmad, M. S. Alatawi, M. F. Nadeem and M. Azeem, “Computation of metric-based resolvability of quartz without pendant nodes,” *IEEE Access*, vol. 9, pp. 151834–151840, 2021.
[15] H. Fernau, P. Heggernes, P. van’t Hof, D. Meister and R. Saei, “Computing the metric dimension for chain graphs,” *Information Processing Letters*, vol. 115, no. 9, pp. 671–676, 2015.
[16] I. Tomescu and M. Imran, “R-sets and metric dimension of necklace graphs,” *Applied Mathematics and Information Sciences*, vol. 9, no. 1, pp. 63–67, 2015.
[17] I. J. L. Garces and J. B. Rosario, “Computing the metric dimension of truncated wheels,” *Applied Mathematical Sciences*, vol. 9, no. 56, pp. 2761–2767, 2015.
[18] M. Imran, F. Bashir, A. Q. Baig, A. U. H. Bokhary, A. Riasat *et al.,* “On metric dimension of flower graphs and convex polytopes,” *Utilitas Mathematica*, vol. 92, pp. 389–409, 2013.
[19] H. Iswadi, E. T. Baskoro and R. Simanjuntak, “On the metric dimension of corona product of graphs,” *Far East Journal of Mathematical Sciences*, vol. 52, no. 2, pp. 155–170, 2011.
[20] A. T. Shahida and M. S. Sunitha, “On the metric dimension of joins of two graphs,” *Global Journal of Pure and Applied Mathematics*, vol. 5, no. 9, pp. 33–38, 2014.
[21] J. Kratica, V. Kovačević-Vujčić and M. Čangalović, “Computing the metric dimension of graphs by genetic algorithms,” *Computational Optimization and Applications*, vol. 44, no. 2, pp. 343–361, 2009.
[22] D. T. Murdiansyah, “Computing the metric dimension of hypercube graphs by particle swarm optimization algorithms,” in *Int. Conf. on Soft Computing and Data Mining*, Cham, Springer, pp. 171–178, 2016.
[23] N. Mladenović, J. Kratica, V. Kovačević-Vujčić and M. Čangalović, “Variable neighborhood search for metric dimension and minimal doubly resolving set problems,” *European Journal of Operational Research*, vol. 220, no. 2, pp. 328–337, 2012.
[24] R. P. Adirasari, H. Suprajitno and L. Susilowati, “The dominant metric dimension of corona product graphs,” *Baghdad Science Journal*, vol. 18, no. 2, pp. 0349, 2021.
[25] B. Mohamed and M. Amin, “The metric dimension of subdivisions of lilly graph, tadpole graph and special trees,” *Applied and Computational Mathematics*, vol. 12, no. 1, pp. 9–14, 2023.
[26] B. Mohamed, L. Mohaisen and M. Amin, “Computing connected resolvability of graphs using binary enhanced harris hawks optimization,” *Intelligent Automation & Soft Computing*, vol. 36, no. 2, pp. 2349–2361, 2023.
[27] B. Mohamed, L. Mohaisen and M. Amin, “Binary equilibrium optimization algorithm for computing connected domination metric dimension problem,” *Scientific Programming*, vol. 2022, pp. 1–15, 2022.

[28] B. Mohamed and M. Amin, “Domination number and secure resolving sets in cyclic networks,” *Applied and Computational Mathematics*, vol. 12, no. 2, pp. 42–45, 2023.

[29] B. Mohamed, “Metric dimension of graphs and its application to robotic navigation,” *International Journal of Computer Applications*, vol. 184, no. 15, pp. 1–3, 2022.

[30] B. Mohamed and M. Amin, “A hybrid optimization algorithms for solving metric dimension problem,” *International Journal on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks*, vol. 15, no. 1, pp. 1–10, 2023.

[31] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk and W. Al-Atabany, “Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems,” *Applied Intelligence*, vol. 51, no. 3, pp. 1531–1551, 2021.