A Novel Method for Optimal Sensor and Actuator Placements: The "Infinite Value Algorithm"

Mahdi Zavar[1],Mohammad Shahraeini[2*], Alireza Safa[3],Niki Manouchehri[4]

Abstract–Cyber-physical systems rely heavily on the functionality of sensors and actuators to operate effectively. Sensors transfer measurements from the physical part to the control and cyber part and while actuators that play the role of applying control signals to the physical part. The placement of sensors and actuators is an important issue for ensuring system's observability and controllability. This placement should be done in way that the system remains controllable and observable with fewest number of sensors and actuators.. Minimizing the number of sensors and actuators has a significant effect on cost and energy reduction. In this study, a new approach is introduced based on the existing paths and non-existent paths between the states of a system. In the proposed approach, the non-existent paths are defined as infinite paths. The best nodes are selected as the location of sensors and actuators based on the infinite paths and their numbers.The numerical simulations illustrate that the Infinite Value Algorithm has performed better in the placements in this way that it has consumed the unique solution in less time compared the Genetic Algorithms.

Keywords:Infinite value algorithm, Sensor placement, Actuator placement, Observability, Controllability.

1. Introduction

Cyber-physical systems (CPSs) integrate physical components with cyber capabilities, revolutionizing industries like electrical infrastructure and communication systems, to create a networked environment [1]. These systems also incorporate the concept of system dynamics, which means that physics and dynamics are intertwined. CPSs and networks form a pervasive computing platform that underlies many modern technologies. They are created by integrating and interconnecting physical equipment and systems from an engineering perspective and different algorithms, enabling them to influence each other. Since CPSs aim to control the dynamics of physical systems, they can be considered dynamic systems. Examples include medical devices [2, 3], communication peripherals [4], intelligent vehicles [5], transportation networks [6, 7], power generation networks [8, 9], and water distribution systems [10]. CPS is closely related to buzzwords such as the Internet of Things, Industry 4.0, Industrial Internet, and Machine-to-Machine. All of these reflect a vision of modern technology that connects the physical world with the world of information.

Graph-based algorithms can be used to analyze and model switched-mode systems [11]. A recent study proposed a new approach using graphs to model switching dynamical systems, capturing interactions between objects, and learning both intra-object and inter-object mode-switching behavior. This method has been successfully applied to model various real-world systems, from crowds of people to groups of immune cells and swarms. Additionally, graph models are gaining popularity in various fields such as social networks, knowledge graphs, and protein-protein interaction networks. Furthermore, a recent work explores the potential of graph models in the biomedical field and introduces a new framework for creating medical records as graphs while maintaining privacy [12].

1.1 CPS Challenges

CPSs pose numerous challenges, with modeling being the most crucial. Modeling helps to analyze how a system works, determine necessary criteria, and tackle other

1 Department of Electrical Engineering, Golestan University, Gorgan, Iran.
Email:m.zavar400@stu.gu.ac.ir

2* CorrespondingAuthor :Department of Electrical Engineering, Golestan University, Gorgan, Iran.
Email:m.shahr@gu.ac.ir

3Department of Electrical Engineering, Golestan University, Gorgan, Iran.
Email:a.safa@gu.ac.ir

4 Department of Electrical Engineering, Golestan University, Gorgan, Iran.
Email: n.manouchehri400@stu.gu.ac.ir

Received: 2024.01.12; Accepted: 2024.04.03

challenges confidently [13]. Other challenges include ensuring security, managing complexity, scalability, integration, and resilience of the system, which become more critical as the dimensions and size of the systems increase [14-18]. Another challenge is observability and controllability, which has recently gained attention from researchers. Observability and controllability allow for establishing a connection between the physical and cyber parts of the system through a series of equipment.

To address this challenge and control and observe a CPS, sensors and actuators must be placed correctly. Leitold and his colleagues have proposed an approach based on Tarjan's algorithm, which locates sensors by finding strongly connected components (SCC) in the system's graph. However, this approach has four significant flaws. Firstly, finding SCCs differs depending on the starting location of the algorithm. Secondly, the location of sensors in SCCs is not unique. Thirdly, the number of sensors is more than required for system observability. Finally, this approach is only used for placing sensors [19].

Using optimization algorithms such as the Genetic Algorithms (GAs) is another way to place sensors and actuators. However, these algorithms require defining the cost function and necessary conditions for execution, leading to many iterations that take a lot of time to execute. Additionally, the final answer is not unique for large systems. Moreover, these algorithms must be executed twice and separately for the placement of sensors and actuators, further complicating the placement process and increasing execution time [20, 21].

1.2Motivation and Contribution

The presented approach utilizes a graph-based method that leverages the system's structural model and adjacency matrix to identify both existing and non-existing paths between system states. While previous methods have limitations, the Infinite Value Algorithm (IVA) addresses these issues. The IVA offers several advancements, such as:

i. The simultaneous placement of sensors and actuators
ii. Achieving observability and controllability with a minimal number of sensors and actuators
iii. Unique results for sensor and actuator placement
iv. Significantly short time to reach the answers

1.3Paper Structure

To provide context for the proposed approach, Section 2 defines key concepts and terms used in the research, including state space model (SSM) and its two sets of equations: state equations and output equations. Section 3 outlines the proposed approach and provides pseudocode for its implementation. To improve the approach's effectiveness, Section 4 briefly discusses the use of GAs and related settings. To evaluate the effectiveness of the proposed approach, Section 5 presents simulations of two systems with 10 and 100 states, created using Erdős–Rényi random graphs. These simulations are evaluated by IVA and GAs, highlighting the advantages of IVA over previous methods. Finally, Section 6 presents conclusions based on the simulations.

2. Preliminaries

2.1 State Space Model

A SSM is a method for describing the behavior of a dynamic system using variables known as states. These states represent the internal state of the system, even if they cannot be observed directly, but can be inferred from available measurements. SSM consists of two sets of equations: state equations and output equations.

State equations illustrate how states change over time and are often expressed as first-order differential equations. They can be represented using matrices and vectors. For a linear time-invariant system with n state variables and m inputs, the state equations can be expressed as:

$$\dot{\mathrm{x}}(\mathrm{t}) = \mathbf{A}\,\mathrm{x}(\mathrm{t}) + \mathbf{B}\,\mathrm{u}(\mathrm{t}) \quad (1)$$

where**A** with $n \times n$ dimensions and **B** with $n \times m$ dimensions are state matrix and output matrix, respectively. Vector x with dimensions $n \times 1$, u with dimensions $m \times 1$ and $\dot{x}$ with dimensions $n \times 1$ are respectively referred to as state vector, input vector, and state vector derivative.

Output equations describe the relationship between observed system outputs and states. They specify how states and inputs affect outputs and are usually presented as algebraic equations. Assuming that the system has p outputs, the output equations can be written in the form:

$$y(t) = \mathbf{C}x(t) + \mathbf{D}u(t) \quad (2)$$

where**C** with dimensions $p \times n$ and **D** with dimensions $p \times m$ are the output matrix and transmission matrix, respectively [22, 23].

2.2Graph Theory

Graph theory is a significant branch of mathematics that deals with graphs and mathematical structures used to represent relationships between objects. This theory has wide applications for modeling distributed systems, large-scale systems, and multi-agent systems. Graph theory focuses on researching the structure and properties of graphs, including their connectivity, and developing algorithms to solve computational problems related to graphs. In recent years, graph theory has become increasingly important in the analysis of complex networks, such as social media platforms and biological networks. In general, graph theory is a valuable tool that presents the analysis of complex systems along with the characteristics of dynamic systems in a simpler and more comprehensible way. Since the stability of CPSs is of great importance, two concepts of controllability and visibility are defined for systems [24, 25].

In a graph, there are two main components: nodes (vertices) and edges. Nodes represent objects or entities, while edges represent connections between them. The mathematical notation of a graph is defined as:

$$\mathcal{G} = (\mathrm{N}, \mathrm{E}) \begin{cases} \mathrm{N} \coloneqq \text{Nodes} \\ \mathrm{E} \coloneqq \text{Edges} \end{cases} \tag{3}$$

A directed graph $\mathcal{G}$ is a graph whose edges define the connection between nodes one-way. In the sense that if there is an edge from node i to node$j$, it does not mean that there is an edge from node j to node i.

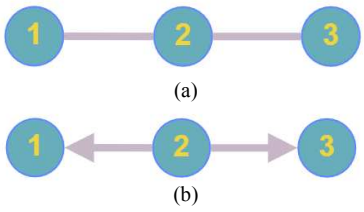

Fig. 1.(a) Undirected graph, (b) Directed Graph or digraph

A path is a sequence of connected edges that allows one to move from one node to another in a graph. The length of a path is determined by the number of edges it covers. A simple path does not repeat vertices or edges, except for the initial and final vertices. In Fig. 1(a), the path is formed by two edges. Figure 2 shows a path between 4 nodes, from node 1 to node 4 with three edges.

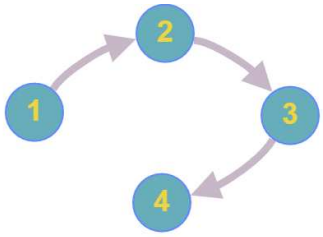

Fig. 2.A directed path with 4 nodes

2.3Adjacency Matrix

Each graph is drawn with a matrix called adjacency matrix. This matrix shows the relationship between the states of a CPS and is a square matrix whose dimensions are equal to the number of states or system nodes [26-28].

The adjacency matrix denoted by $\mathcal{A}$ in (4) is an$n \times n$ matrix where each row and column corresponds to a vertex. The value of a_{ij} in the matrix indicates whether there is an edge between vertices i and j. For a directed graph, if there is a directed edge from node i to node j, then a_{ij} has a value of one. If there is no directed edge from vertex i to vertex j, a_{ij} is set to zero. It is important to note that the diagonal entry a_{ii} in the adjacency matrix indicates whether there is a self-loop at node i or not. It can be set to a specified value if self-loops are allowed, or zero otherwise.

$$\mathcal{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{4}$$

The transpose of the adjacency matrix represents the state matrix of the system in the state space model as follows:

$$\mathcal{A} = \mathbf{A}^{\mathrm{T}} \tag{5}$$

2.4Controllability and Observability

Studying the observability and controllability of systems is one of the main applications of graph theory. For this purpose, a graph refers to the desired system in which the vertices correspond to the states, inputs, and outputs, and the edges express the relationships between the vertices. The graph can be assumed to be directed and weighted or without them, and the characteristics of dynamic systems can be informed using adjacency matrices. The main pillars of the systems are observability and controllability. By examining these two criteria, sensors and actuators can be placed in the systems [29, 30].

One of the widely used methods in evaluating the controllability and observability of CPSs is the use of

observability and controllability matrices and the Kalman rank check. Equations (6) and (7) show these matrices and the Kalman rank check of the system.

$$\mathcal{C} = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \cdots \quad \mathbf{A}^{n-1}\mathbf{B}] \tag{6}$$

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{7}$$

A necessary and sufficient condition for a CPS to be controllable and observable is that the rank of the controllability matrix $\mathcal{C}$ and the observability matrix $\mathcal{O}$ is equal to the number of states of the system, i.e. the dimensions of the state matrix **A** or the adjacency matrix $\mathcal{A}$. These conditions ensure that any state variable can be affected by appropriate choices of input and output vectors.

However,structuralcontrollability and observability occur based on the direction of the edges. In structural controllability, if an actuator is placed on a node that is connected to another node by a directed edge (output edge), both nodes are controlled. This process also transpires for observability with the input edge.

2.5 Structured Model of CPS

As mentioned earlier, modeling is one of the important challenges of CPS. Structural modeling is a new approach that has been used in recent years. The basic concept of the structured model is to store non-zero information in the state space matrices of the system. Zeros remain and non-zero entries are replaced by free parameters. Therefore, the state space matrices of the system as well as the adjacency matrix have only zero and one members. A system with this feature is known as a structured system, and in this way, modeling is called a structured model [31, 32].

This construct consists of several components, including certain zero-valued variables that do not have a direct effect on other variables. Some inputs, such as one, are constant and can help extract state variables effortlessly from other variables. In addition, the inputs that interact and communicate with each other are dependent on each other based on the definition of system parameters through algebraic equations. The symbol$\mathcal{A}$, which was previously used to represent the adjacency matrix, is also used as a structured model in the form of zeros and ones:

$$a_{ii}, a_{ij} := 0 \text{ or } 1 \tag{8}$$

To build a structuredmodel, you can create a directed graph with nodes that represent variables. These nodes include input nodes, state variables, and output nodes. If there are non-zero parameters for these variables in the state space equations, the associated nodes are connected by an edge.

By combining the techniques of structuredmodeling and graph theory, one can benefit from several advantages. First, for the extraction of additional structured knowledge by analyzing physical laws and decomposing the overall system into subsystems. Second, it provides the advantage of creating a clear visual representation of the structure,which is easier to understand. Third, it enables the possible of examine features that are purely structurally relevant without considering unknown parameter values. Typically, these unknown parameters refer to physical parameters in functions. Finally, it minimizes the computational load, making CPS manageable, especially for large systems.

3. ProposedApproach: Infinite Value Algorithm

This algorithm is defined based on edges and paths between nodes so that the system becomes observable with the fewest number of sensors and controllable with the fewest number of actuators. In fact, in graph $\mathcal{G}$, a graph built based on the adjacency matrix, the i-th node with a directed edge has a path to the j-th node, by placing a sensor on the j-th node, both nodes can be viewed structurally. By examining the Kalman rank of the system confirms the correctness of this concept. The same concept is realized in the $\mathcal{G}$ graph transpose, i.e. the graph implemented based on the system state matrix, for system actuators and structural controllability. The IVA is named for the way it works, which treats the number of non-existent paths between nodes as an infinite value.

Algorithm 1. Shortest Path

ShortestPath():
Require:adj
Ensure: Sorted nodes paths
1.**for**each node**do**
2. $sp \leftarrow$ totalPath(adj)
3. $NoInfinite \leftarrow$ infPath(sp)
4. **end for**
5. $sortNoInf \leftarrow$ Sort($NoInfinite$)
return: $sortNoInf$

The way IVA works is that it first determines the shortest paths between each node and other nodes. If a node is not

connected to another node, it means that there is no input edge in the sensor placement and no output edge to that node in the actuator placement. In this algorithm, non-existent paths are specified with infinite value. After determining the shortest paths, the number of infinite paths is estimated and sorted in ascending order based on the node number. Then, if a node has no infinite path, that node is known as the sensor and operator location. If the number of nodes with zero infinite paths is more than one, then the node with the highest number of paths with the rest of the nodes is selected. If the minimum number of infinite paths is not zero, the same number of sensors and actuators are added.

Algorithm 2. Infinite Value Algorithm

Require: adj
Ensure: $driverNodes, SensorNodes$
1. $[NoNodes, NoEdges] \leftarrow$ findNum (adj)
2. Call ShortestPath(adj)
3. $driverNoInf \leftarrow sortNoInf(1)$
4. **while** $driverNoInf = 0$ **do**
5.**for**each node**do**
6.**if***i*th node hasn't path to jth node **then**
7.**for**each node**do**
8.**if***k*th node has path to jth node**then**
9. $NoPathToInf \leftarrow [k, shortestPath(k, j)]$
10.**end if**
11. **end for**
12. **for**each node**do**
13. **if***k*th node hasn't path to jth node**then**
14. $NoPathToInf \leftarrow [j, shortestPath(j, j)]$
15. **end if**
16.**end for**
17.**for**each $NoPathToInf$ **do**
18. $bestDriver \leftarrow$ selectBest($NoPathToInf$)
19.$minInf \leftarrow$ selectMinInf($bestDriver$)
20.**end for**
21.Set new $driverNodes$ from $minInf$
22.**end if**
23.**end for**
24. $driverNoInf \leftarrow driverNoInf - 1$
19. **end while**
20.The same process is done for sensor placement
return: $driverNodes, SensorNodes$

Equation (9) expresses the number of sensors and actuators based on infinite paths. If the number of infinite paths is non-zero in the lowest case, you should consider the paths between the nodes. If the first node and other sensor nodes are specified, the path between them should be checked, and if there is a round edge path between two nodes, choosing one of them is sufficient for the system's observability.This procedure is also true for the actuator placement.

$$\text{if}\begin{cases}\text{number of}\infty\ = 0\ \Rightarrow \text{Number of sensors or actuators}\ = 1\\ \text{number of}\infty = k\ \Rightarrow \text{Maximum number of sensors or actuators} = k+1\end{cases} \quad (9)$$

The main issue in sensor and actuator placement is determining what it means to minimize a matrix-valued function. Doubtlessly, the suitability of a sensor and actuator configuration depends on the evaluation criteria. Within the literature, various cost functions have been proposed, including maximizing the trace, the determinant, the rank, or the minimum eigenvalue of the observability and its dual, the controllability Gramian. For example, the authors in [33-36] used the observability Gramian and its dual, the controllability Gramian for sensor and actuator placement. More precisely, the observability of a deterministic linear system is given by the observability Gramian, which is defined as:

$$\mathrm{W}_0 = \int_{t_0}^{t_f} e^{\mathbf{A}^{\mathrm{T}}t}\mathbf{C}^{\mathrm{T}}\mathbf{C}e^{\mathbf{A}t}\,dt \quad (10)$$

It is proven that maximizing the observability Gramian will in some sense minimize the estimation error. Generally, the calculation of Gramian is computationally intractable. One intuitive way to handle this issue is to assume that systems are stable. The state transition matrix $e^{\mathbf{A}t}$ comprises decaying exponentials for stable systems, so a finite positive definite limit of the observability Gramian always exists and can be calculated by solving a Lyapunov equation in (11).

$$\mathbf{A}^{\mathrm{T}}\mathrm{W}_{\mathrm{o}} + \mathbf{C}^{\mathrm{T}}\mathbf{C} + \mathrm{W}_{\mathrm{o}}\mathbf{A} = 0 \quad (11)$$

However, this assumption restricts the applicability of these methods only for stable systems. In this light, we use the Kalman rank-check-based criterion to minimize for considering both stable and unstable linear systems.

4. Genetic Algorithm

A genetic algorithm is an evolutionary algorithm that is formulated based on biological methods such as mutation, inheritance, selection principles, etc. Figure 3 shows the

flowchart related to GA. This algorithm is used for prediction and mathematical modeling. Designed to replicate specific natural evolutionary processes, GA has become a highly effective stochastic search technique that relies on the principles of natural selection and genetics. A GA starts with a set of arbitrary solutions called a population, where each individual is coded as a chromosome that proposes a solution to the problem. These chromosomes evolve through several iterations or generations. During each generation, chromosomes are evaluated based on specific fitness criteria. As the process proceeds over several generations, the algorithm converges to the best chromosome, which represents the optimal solution [37].

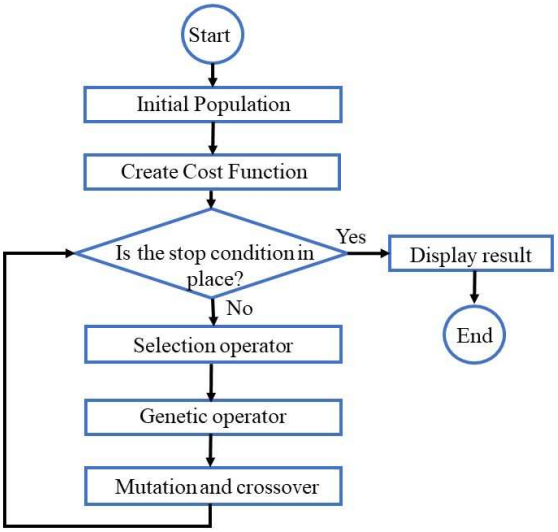

Fig. 3.A simple flowchart of GA

$$\text{Cost} = \text{observability or controllability} \tag{12}$$

The cost function considered for GA is system observability and controllability, which are performed in two separate steps.In (12) cost functions related to sensor and actuator placement are defined.Chromosome coding is binary and the number of genes is equal to the number of system states. The probability of crossoverand mutationequals 60% and 20%, respectively. The selection function is "TournamentSelection" and the population equals 25 individuals. The stop condition is defined as the number of executions equal to 200 iterations.

$$\begin{aligned} &\text{Minimize: Cost} = \text{Not}(OC) + Penalty \\ &OC = \text{Obsrvability or Controllability} \\ &Penalty = \frac{\text{Number of } Sensors}{\text{Number of } State} \end{aligned} \tag{13}$$

Another change has been made in the GA condition and cost function so that the number of sensors and actuators can be minimized. It can be seen in (13) cost functions and theirs conditions. However, the stopping condition this time is finding the minimum number of sensors and actuators to guarantee observability and controllability.

5. Simulation and Analysis

In this section, for 10-state and 100-state systems, the directed Erdős–Rényirandom graph model is used to compare IVA and GA. We have followed the approach presented in [38] to produce random directed graphs. All simulations and results are done in MATLAB software[1].The GA has been executed 10 times in such a way as to confirm the controllability and observability of the system.

5.1 The 10-state System

The first simulation is a system with 10 nodes, or a system that has 10 states in its state space model. The nodes or states of the 10-state system and their connections are expressed based on the adjacency matrix (14).

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

As noted earlier, from the adjacency matrix, it can be understood that each node has an output edge and an input edge. Therefore, according to (14), the graph of the system can be drawn and its connections can be observed. Figure 4 shows the graph associated with this 10×10 adjacency matrix, where blue circles refer to nodes and black arrows refer to edges.

[1]System info: Core(TM) i3 CPU, M370 @ 2.40GHz, RAM 4.00 GB

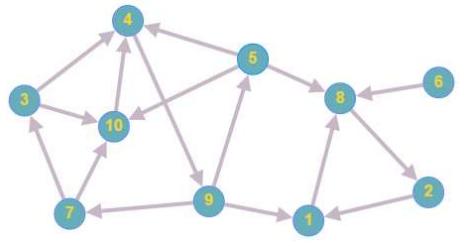

Fig. 4. Graph representation for 10-state system

In the first step, using the proximity matrix and Algorithm 1, it is possible to determine the existing paths and the infinite paths between the nodes. This tsakcan be easily in small dimensions of the drawing graph, but it becomes very difficult and sometimes impossible in large dimention.Therefore, utilizing the adjacency matrix will greatly assist in carrying out and implementing this algorithm. Table 1 and Table 2 show the paths between the nodes for the sensors positioning based on the proximity matrix and for the actuators positioning according to the state matrix. The sum of the infinite paths for each node is calculated and presented in ascending order in Table 2. Based on the IVA, the node with the fewest infinite paths is selected as candidate for both sensor and actuator. The system becomes observable with a sensor due to its structure, which allows access to all nodes of the system through one node.

Table 1. Existent paths and infinite paths in 10-state system for sensor placement

Node	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	0	1	3	2	3	3	4	2	1	3
x_2	2	0	5	4	2	2	6	1	3	5
x_3	∞	∞	0	3	4	∞	1	∞	2	4
x_4	∞	∞	1	0	1	∞	2	∞	2	1
x_5	∞	∞	3	2	0	∞	4	∞	1	3
x_6	∞	∞	∞	∞	∞	0	∞	∞	∞	∞
x_7	∞	∞	3	2	3	∞	0	∞	1	3
x_8	1	2	4	3	1	1	5	0	2	4
x_9	∞	∞	2	1	2	∞	3	∞	0	2
x_{10}	∞	∞	1	3	1	∞	1	∞	2	0

Table 2. Existent paths and infinite paths in 10-state system for actuator placement

Node	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	0	2	∞	∞	∞	∞	∞	1	∞	∞
x_2	1	0	∞	∞	∞	∞	∞	2	∞	∞
x_3	3	5	0	1	3	∞	3	4	2	1
x_4	2	4	3	0	2	∞	2	3	1	3
x_5	3	2	4	1	0	∞	3	1	2	1
x_6	3	2	∞	∞	∞	0	∞	1	∞	∞
x_7	4	6	1	2	4	∞	0	5	3	1
x_8	2	1	∞	∞	∞	∞	∞	0	∞	∞
x_9	1	3	2	2	1	∞	1	2	0	2
x_{10}	3	5	4	1	3	∞	3	4	2	0

According to Algorithm 1, Table 1 and Table 2, Table 3 is formed. Based on this table, 3 nodes are candidates for placing sensors and 6 nodes for actuators. But one must be chosen among them for sensor placement and two nodes be chosen for actuator placement. For this purpose, the sum of the shortest paths available for each node is calculated. In other words, the node that is connected to the largest node with the largest edge is selected. This table shows the number of infinite paths and the sum of the shortest paths available for placing sensors and actuators.

Now you can choose sensors and actuators according to Table 3. The number of infinite paths means that one node has no path to multiple nodes. In controllability, it means not controlling the node and in observability it means not seeing the desired node. In this table for actuators and sensors, the minimum number of infinite paths is zero and one, so this system is observable and controllable with one sensor and twoactuators. In the positioning of the sensor betweenx_1, $x_2$and x_8, node x_2 and the first positioning of the actuator betweenx_3, x_4, x_5, x_7, $x_9$andx_{10}, nodex_7 includes the most paths, so they are chosen as the right place for the placement of the sensor and the actuator.After that, to select the location of the second actuator, it should be determined which x_7 does not have a path to.then by referring to Table 2, another node that has a path to it should be selected. If several nodes have paths to that node, the node that has the highest sum of paths will be selected. Here, referring to Table 2 and the row related tox_7, it can be seen that this node does not have a path tox_6 and there is no other node that has a path to x_6. Therefore, $x_6$itself is also selected as the location of the actuator.

Table 3. Sum of the infinite and existent paths for 10-state system

Node	Number of Infinite Paths		Sum of Existent Paths	
	Sensors	Actuators	Sensors	Actuators
x_1	0	7	22	3
x_2	0	7	30	3
x_3	4	1	14	22
x_4	4	1	7	20
x_5	4	1	13	17
x_6	9	6	0	6
x_7	4	1	12	26
x_8	0	7	23	3
x_9	4	1	10	14
x_{10}	4	1	8	25

Figure 5 shows the comparison of the execution time of IVA and GAin 10runs. It should be noted that in IVA sensor and actuator placement are determined at the same time, so the displayed execution time is the sum of the execution time of the actuator and sensor placement. GA_{oc-sen}andGA_{oc-act} mean sensor placement and actuator placement respectively by GA along with (12). Also,

$GA_{min-sen}$ and $GA_{min-act}$ refer to sensor placement and actuator placement by GA along with (13). As can be seen, the running times of GA_{oc} and GA_{min} each alone are significantly different from IVA. However, Fig. 6 shows the execution time of GA_{oc} and GA_{min} as the total execution time of sensor placement and actuator placement. Based on this figure, it can be seen that the execution time of GAs is very different from the execution time of the proposed IVA.

Figure 7 and Fig. 8 respectively show the output results of the algorithms during 10 separate runs for sensor and actuator placements. The IVA had a unique answer, while the GA_oc and GA_min had different answers in the output. As mentioned earlier, one of the disadvantages of GA is the creation of the cost function and its necessary conditions. This has caused the failure to define the necessary conditions for sensor and actuator placement, resulting lack of a unique output. In addition, in Fig. 8, the performance of GA_{oc} in the ninth run is specified and it has selected one more actuator that is not minimized.

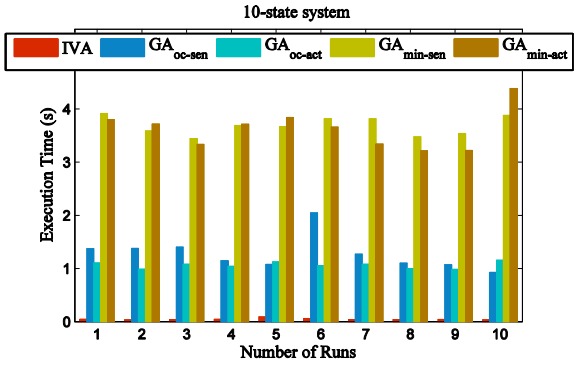

Fig. 5. Comparison of execution time of IVA and two separate cost functions ofGA_{oc} and GA_{min} for sensor and actuator in 10-state system

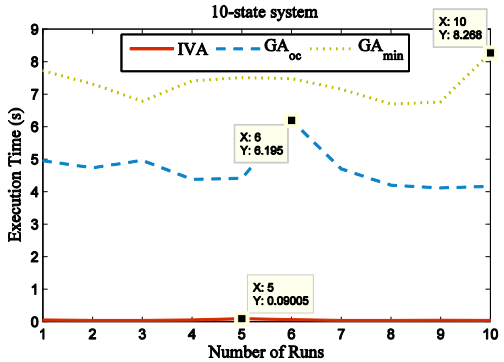

Fig. 6. Comparison of execution time of IVA, GA_{oc} and GA_{min} for sensor and actuator placement in 10-state system

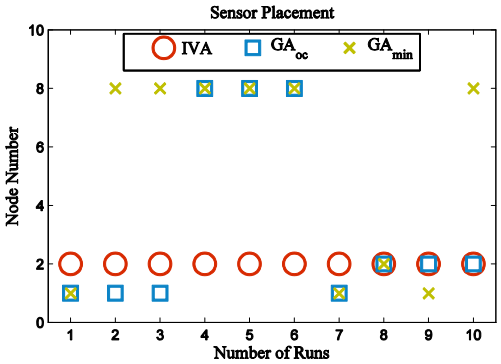

Fig. 7. Comparsionof sensorplacements in 10- state system

Table 4 clearly compares the performance of these three algorithms for placements. As can be seen, if the number of sensors and actuators is considered almost equal, their execution time is very defferent. Approximately, the execution time of GA_{oc} was 100 times and the execution time of GA_{min} was 150 times that of IVA.

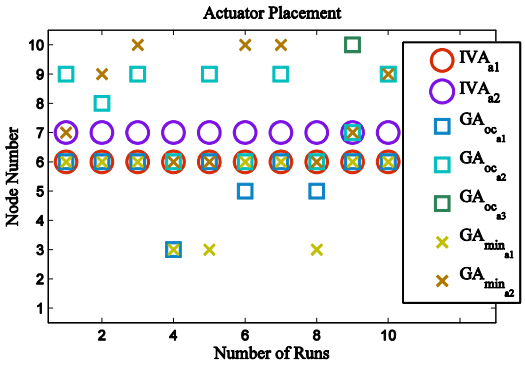

Fig. 8. Comparsionof actuatorplacements in 10- state system

Table 4. Compare algorithm results in 10-state system

IVA	GA_{min}	GA_{oc}		
1	1	1	Avg. Number of Sensors	
2	2	2.1	Avg. Number of Actuators	
0.048	3.684	2.122	Sensors	Avg. Execution Time
	3.624	2.558	Actuators	
0.048	7.308	4.680	Avg. Total Execution Time	

5.3The 100-state System

Now a larger system with 100 nodes or states is considered, on which the same processes as before are performed. The related graph is shown in Fig. 9. Now, using the tables related to IVA, you can start placing the sensor and actuator. The calculation of the shortest existing paths and non-existing paths or infinite paths between nodes is done in the first step and can be seen in Table 5.

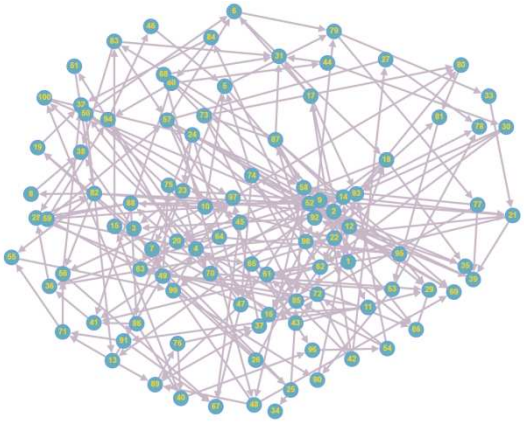

Fig. 9. Graph representation for 100-state system

As can be seen in Table 5, one node for the sensor and 8 nodes for the actuators are identified as having the fewest infinite paths. For positioning the sensors, only x_{78}has the fewest infinite paths, and for positioning the actuators, x_{84} has the most paths by 590. Therefore, these two nodes are selected as the first sensor and actuator nodes, but according to (9), because each of these nodes has an infinite path, another node for the sensor and another node for the actuator must be selected. To determine the other sensors and actuators, it should be checked which node is the infinite path of the nodes.

Table 5. Sum of the paths for 100-state system

Placement	Node	Number of Infinite Paths	Sum of Existent Paths
Sensors	x_{78}	5	172
Actoators	x_{30}	1	439
	x_{44}	1	530
	x_{60}	1	525
	x_{68}	1	421
	x_{73}	1	438
	x_{84}	1	590
	x_{87}	1	371
	x_{90}	1	15

Table 6 shows the nodes that do not have a path to x_{78}. Among them, two nodes x_{66} and x_{29}have a round trip path, so by choosing one of them as a sensor location, the other node will also be seen. Considering the total number of paths of each node in Table 6, $x_{29}$is placed in more paths than x_{66}, so $x_{29}$is chosen as the location of the sensor.In addition, for the other nodes listed in Table 6, there is no path to other nodes, so those nodes themselves are selected as the location of the sensors.

Table 6. Existent paths and infinite paths in 100-state system for sensor placement

Node	x_{29}	x_{31}	x_{34}	x_{66}	x_{75}	Sum of Existent Paths
x_{29}	0	∞	∞	1	∞	357
x_{31}	∞	0	∞	∞	∞	583
x_{34}	∞	∞	0	∞	∞	742
x_{66}	1	∞	∞	0	∞	351
x_{75}	∞	∞	∞	∞	0	519
x_{78}	∞	∞	∞	∞	∞	172

To select the second actuator node, the node to which x_{84}does not have a path must be specified first. Node x_{26} is the mentioned node and is not related to any other node, so it is chosen as the location of the actuator.

Table 7. Existent paths and infinite paths in 100-state system for actuator placement

Node	x_{26}	x_{84}	Sum of Existent Paths
x_{26}	0	∞	560
x_{84}	∞	∞	590

Figure10 and Fig. 11 show the execution time of IVA and the execution time of GAs. Figure 10 is the comparison of IVA,$GA_{oc}$and $GA_{min}$for sensor and actuator placement with two separate cost functions, and Fig.11 is the comparison of IVA, $GA_{oc}$and GA_{min}in general.

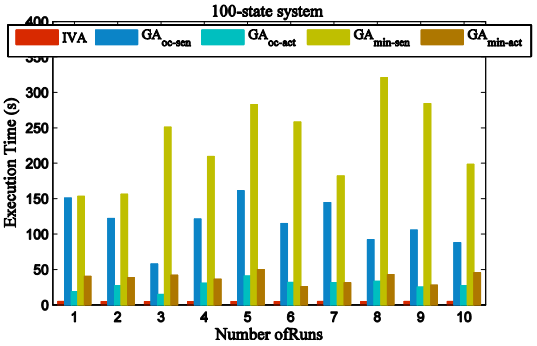

Fig. 10. Comparison of execution time of IVA and two separate cost functions of GAs for sensor and actuator in 100-state system

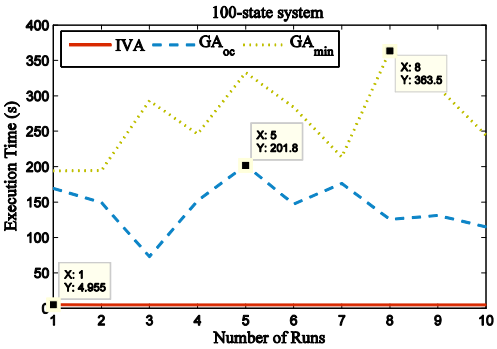

Fig. 11. Comparison of execution time of IVA and GAs for placements in 100-state system

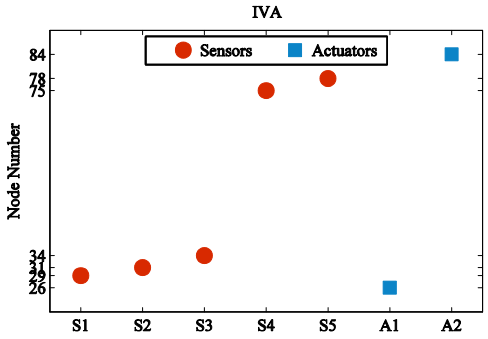

Fig. 12. Sensor and actuator placements in 100- state system by IVA

Figure 12 is related to the placement of the sensor and actuator in IVA, which can be observed and controlled by checking the Kalman rank of this 100-state system with 5 sensors and 2 actuators. Additionally, Fig. 13 displays the number of sensors and actuators in this system for 10 different runs of GA_{oc}. It is evident that the minimization of the number of sensors and actuators did not occur in this method. Furthermore, Fig. 14 illustrates the sensor and actuator placement by GA_{min}. However, this algorithm was unable to minimize the number of sensors in its fifth run within 200 iterations.

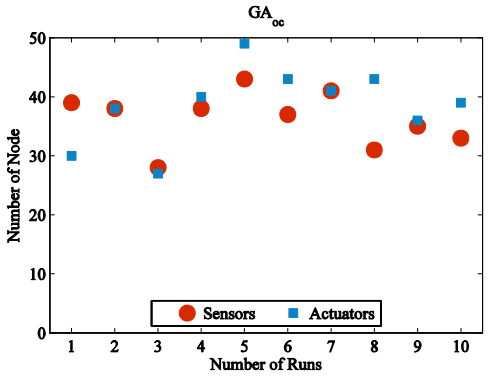

Fig. 13. Number of Sensors and actuators in 100- state system by GA_{oc}

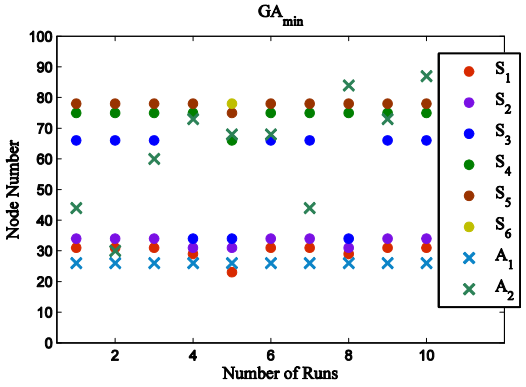

Fig. 14. Sensor and actuator placements in 100- state system by GA_{min}

Another comparison between the algorithms for the 100-state system is shown in Table 8. Here, the GA_{oc}considers a very large number of nodes for sensors and actuators, while in $GA_{min}$and IVA, their number is very small. On the other hand, the difference in execution time of GAs has been significantly higher than IVA. The total execution time of $GA_{min}$was more than 1.5 times that of $GA_{oc}$and more than 50 times that of IVA. Also, the total execution time of $GA_{oc}$was more than 30 times that of IVA.

Table 8. Results of proposed algotithm for 100-state system

IVA	GA_{min}	GA_{oc}		
5	5.1	36.30	Avg. Number of Sensors	
2	2	38.60	Avg. Number of Actuators	
4.783	229.668	115.828	Sensors	Avg. Execution Time
	38.154	28.136	Actuators	
4.783	267.822	143.964	Total Avg.Execution Time	

5. Conclusion

In this study, a new approach introduced,which simultaneously and in parallel advances the placement of sensors and actuators. This graph-based algorithm is based on the paths between nodes, particularly the paths that are introduced as infinite paths. After addressing the basic and necessary definitions, IVA is presented and a comparison with GAs has been made in the placements to evaluate it. Additionally, two methods have been used for GA, one of which only works with system controllability and observability as GA_{oc}, and the other determines the minimum number of sensors and actuators required to obtain controllability and observability as GA_{min}.

The simulation results indicate three main and fundamental advantages of the IVA algorithm, which can be an important reason for its use. The most important factor in IVA is the simultaneous sensor and actuator placement. This process prevents the algorithm from repeating itself and both placements occur with a single execution. This is done separately in the GA optimization algorithms. Another advantage is that IVA guarantees the uniqueness of the solution, while GA_{min}, although it is designed to find the minimum value, was not able to minimize in 200 iterations. The third benefitof IVA is the much lower execution time compared to GAs.The aforementioned advantages have been shown in simulations related to 10-state and 100-state systems. However, execution time will take a significant amount in large-scale systems, and the time difference between these two algorithms (i.e. the proposed one and GAs) will be more visible. In general, IVA as a graph-based method has performed better than GAsas optimization-based method and can be a suitable option for sensor and

actuator placement.

In future research, there will be a focus on combining IVA and centrality measures for placements. This will be particularly useful when infinite paths in IVA are equal, as the use of centrality measures will allow for more efficient selection of placement candidates. Additionally, important real systems will be studied to determine the impact of various factors and to assess the effectiveness of the proposed algorithm. Furthermore, future work should extend the proposed methodology to incorporate more sophisticated models. In particular, we optimized sensor and actuator placement with the assumption that the understudied models are linear time invariant systems. However, real-world systems are often influenced by time varying and nonlinear processes. Neglecting these dynamics may impair the optimality of the sensor and actuator placement. Further work should also investigate ways to extend our proposed sensor and actuator placement algorithm to time-varying and nonlinear systems.

References

[1] Bakirtzis, Georgios, Christina Vasilakopoulou, and Cody H Fleming. "Compositional Cyber-Physical Systems Modeling." arXiv preprint arXiv:2101.10484 (2021).

[2] Ho, Nicholas, Pooi-Mun Wong, Ngoc-Son Hoang, Dun-Kai Koh, Matthew Chin Heng Chua, and Chee-Kong Chui. "Cps-Based Manufacturing Workcell for the Production of Hybrid Medical Devices." Journal of Ambient Intelligence and Humanized Computing 12 (2021): 10865-79.

[3] Tavčar, Jože, Jože Duhovnik, and Imre Horváth. "From Validation of Medical Devices Towards Validation of Adaptive Cyber-Physical Systems." Journal of Integrated Design and Process Science 23, no. 1 (2019): 37-59.

[4] Castillo-Martínez, Diego Hilario, Adolfo Josué Rodríguez-Rodríguez, Adrian Soto, Alberto Berrueta, David Tomás Vargas-Requena, Ignacio R Matias, Pablo Sanchis, Alfredo Ursúa, and Wenceslao Eduardo Rodríguez-Rodríguez. "Design and on-Field Validation of an Embedded System for Monitoring Second-Life Electric Vehicle Lithium-Ion Batteries." Sensors 22, no. 17 (2022): 6376.

[5] Hu, Zhongxu, Shanhe Lou, Yang Xing, Xiao Wang, Dongpu Cao, and Chen Lv. "Review and Perspectives on Driver Digital Twin and Its Enabling Technologies for Intelligent Vehicles." IEEE Transactions on Intelligent Vehicles (2022).

[6] Li, Ning, Haiyi Sun, Xin Jing, and Zhongtang Chen. "Dynamic Modeling and Aperiodically Intermittent Strategy for Adaptive Finite-Time Synchronization Control of the Multi-Weighted Complex Transportation Networks with Multiple Delays." Chinese Physics B 30, no. 9 (2021): 090507.

[7] Wu, Jianping, and Dongping Fang. "Role of Cps in Smart Cities." Cyber-Physical Systems in the Built Environment (2020): 255-72.

[8] Cui, Yi, Feifei Bai, Tapan Saha, and Jalil Yaghoobi. "Authenticating Source Information of Distribution Synchrophasors at Intra-State Locations for Cyber-Physical Resilient Power Networks." International Journal of Electrical Power & Energy Systems 139 (2022): 108009.

[9] Khan, Izhar Ahmed, Dechang Pi, Nasrullah Khan, Zaheer Ullah Khan, Yasir Hussain, Asif Nawaz, and Farman Ali. "A Privacy-Conserving Framework Based Intrusion Detection Method for Detecting and Recognizing Malicious Behaviours in Cyber-Physical Power Networks." Applied Intelligence (2021): 1-16.

[10] Cui, Xinyue. "Cyber-Physical System (Cps) Architecture for Real-Time Water Sustainability Management in Manufacturing Industry." Procedia CIRP 99 (2021): 543-48.

[11] Liu, Yongtuo, Sara Magliacane, Miltiadis Kofinas, and Efstratios Gavves. "Graph Switching Dynamical Systems." arXiv preprint arXiv:2306.00370 (2023).

[12] Vazirgiannis, Michalis. "Gnns and Graph Generative Models for Biomedical Applications." Proceedings of the ACM Web Conference 2023, Austin, TX, USA, Association for Computing Machinery, 2023.

[13] Adamos, Konstantinos, George Stergiopoulos, Michalis Karamousadakis, and Dimitris Gritzalis. "Enhancing Attack Resilience of Cyber-Physical Systems through State Dependency Graph Models." International Journal of Information Security (2023): 1-12.

[14] Bodkhe, Umesh, Dhyey Mehta, Sudeep Tanwar, Pronaya Bhattacharya, Pradeep Kumar Singh, and Wei-Chiang Hong. "A Survey on Decentralized Consensus Mechanisms for Cyber Physical Systems." IEEE Access 8 (2020): 54371-401.

[15] Mittal, Saurabh, and Andreas Tolk. Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy. John Wiley & Sons, 2019.

[16] Abbas, Arbab Waseem, and Safdar Nawaz Khan Marwat. "Scalable Emulated Framework for Iot

Devices in Smart Logistics Based Cyber-Physical Systems: Bonded Coverage and Connectivity Analysis." IEEE Access 8 (2020): 138350-72.

[17] Fataliyev, Tahmasib Kh, and Shakir A Mehdiyev. "Integration of Cyber-Physical Systems in E-Science Environment: State-of-the-Art, Problems and Effective Solutions." International Journal of Modern Education and Computer Science 11, no. 9 (2019): 35.

[18] Colabianchi, Silvia, Francesco Costantino, Giulio Di Gravio, Fabio Nonino, and Riccardo Patriarca. "Discussing Resilience in the Context of Cyber Physical Systems." Computers & Industrial Engineering 160 (2021): 107534.

[19] Leitold, Dániel, Ágnes Vathy-Fogarassy, and János Abonyi. "Network-Based Observability and Controllability Analysis of Dynamical Systems: The Nocad Toolbox." F1000Research 8 (2019).

[20] Civera, Marco, Marica Leonarda Pecorelli, Rosario Ceravolo, Cecilia Surace, and Luca Zanotti Fragonara. "A Multi□Objective Genetic Algorithm Strategy for Robust Optimal Sensor Placement." Computer□Aided Civil and Infrastructure Engineering 36, no. 9 (2021): 1185-202.

[21] Dhuri, KD, and P Seshu. "Multi-Objective Optimization of Piezo Actuator Placement and Sizing Using Genetic Algorithm." Journal of sound and vibration 323, no. 3-5 (2009): 495-514.

[22] Peng, Yuhuai, Alireza Jolfaei, Qiaozhi Hua, Wen-Long Shang, and Keping Yu. "Real-Time Transmission Optimization for Edge Computing in Industrial Cyber-Physical Systems." IEEE Transactions on Industrial Informatics 18, no. 12 (2022): 9292-301.

[23] Zhou, Xin, Xiaodong Gou, Tingting Huang, and Shunkun Yang. "Review on Testing of Cyber Physical Systems: Methods and Testbeds." IEEE Access 6 (2018): 52179-94.

[24] Tutte, William Thomas. Graph Theory. Vol. 21: Cambridge university press, 2001.

[25] Shahraeini, Mohammad, Panayiotis Kotzanikolaou, and Mehrab Nasrolahi. "Communication Resilience for Smart Grids Based on Dependence Graphs and Eigenspectral Analysis." IEEE Systems Journal 16, no. 4 (2022): 6558-68.

[26] Bhunia, Pintu, Santanu Bag, and Kallol Paul. "Bounds for Eigenvalues of the Adjacency Matrix of a Graph." Journal of Interdisciplinary Mathematics 22, no. 4 (2019): 415-31.

[27] Shahraeini, Mohammad, Shahla Khormali, and Ahad Alvandi. "Optimal Pmu Placement Considering Reliability of Measurement System in Smart Grids." Paper presented at the 2022 12th International Conference on Computer and Knowledge Engineering (ICCKE), 2022.

[28] Xie, Jun, Qiguang Miao, Ruyi Liu, Wentian Xin, Lei Tang, Sheng Zhong, and Xuesong Gao. "Attention Adjacency Matrix Based Graph Convolutional Networks for Skeleton-Based Action Recognition." Neurocomputing 440 (2021): 230-39.

[29] Jungers, Raphael M, Atreyee Kundu, and WPMH Heemels. "Observability and Controllability Analysis of Linear Systems Subject to Data Losses." IEEE Transactions on Automatic Control 63, no. 10 (2017): 3361-76.

[30] Yan, Jiayuan, Bin Hu, Zhi-Hong Guan, Tao Li, and Ding-Xue Zhang. "On Controllability and Observability of a Class of Fractional-Order Switched Systems with Impulse." Nonlinear Analysis: Hybrid Systems 50 (2023): 101378.

[31] Chen, Chen, Ruiyue Peng, Lei Ying, and Hanghang Tong. "Fast Connectivity Minimization on Large-Scale Networks." ACM Transactions on Knowledge Discovery from Data (TKDD) 15, no. 3 (2021): 1-25.

[32] Sun, Wen, Junxia Guan, Jinhu Lü, Zhigang Zheng, Xinghuo Yu, and Shihua Chen. "Synchronization of the Networked System with Continuous and Impulsive Hybrid Communications." IEEE Transactions on Neural Networks and Learning Systems 31, no. 3 (2019): 960-71.

[33] Bopardikar, Shaunak D. "A Randomized Approach to Sensor Placement with Observability Assurance." Automatica 123 (2021): 109340. https://doi.org/https://doi.org/10.1016/j.automatica.2020.109340.

[34] Manohar, Krithika, J Nathan Kutz, and Steven L Brunton. "Optimal Sensor and Actuator Selection Using Balanced Model Reduction." IEEE Transactions on Automatic Control 67, no. 4 (2021): 2108-15. https://doi.org/https://doi.org/10.1109/TAC.2021.3082502.

[35] Takahashi, Shun, Yasuo Sasaki, Takayuki Nagata, Keigo Yamada, Kumi Nakai, Yuji Saito, and Taku Nonomura. "Sensor Selection by Greedy Method for Linear Dynamical Systems: Comparative Study on Fisher-Information-Matrix, Observability-Gramian and Kalman-Filter-Based Indices." IEEE Access (2023). https://doi.org/https://doi.org/10.1109/ACCESS.2023.3291415.

[36] Yamada, Keigo, Yasuo Sasaki, Takayuki Nagata, Kumi

Nakai, Daisuke Tsubakino, and Taku Nonomura. "Efficient Sensor Node Selection for Observability Gramian Optimization." Sensors 23, no. 13 (2023): 5961. https://doi.org/https://doi.org/10.3390/s23135961.

[37] Shirajuddin, Talhah Mohamad, Nur Shazwani Muhammad, and Jazuri Abdullah. "Optimization Problems in Water Distribution Systems Using Non-Dominated Sorting Genetic Algorithm Ii: An Overview." Ain Shams Engineering Journal 14, no. 4 (2023): 101932.

[38] Shahraeini, Mohammad. "Modified Erdős–Rényi Random Graph Model for Generating Synthetic Power Grids." IEEE Systems Journal. Institute of Electrical and Electronics Engineers (IEEE), 2023. https://doi.org/10.1109/jsyst.2023.3339664.