Volume 1, Issue 1, 2024

Intuitions & Insights

An Interdisciplinary Research Journal

Involvement of Mathematics in Examination Scheduling Problems and VideoGames

Sk Amanathulla[1 *] and Tarasankar Pramanik[2]

[1] Department of Mathematics, Raghunathpur College, Raghunathpur, West Bengal, India,

Email: samanathulla.math@raghunathpurcollege.ac.in

[2] Department of Mathematics, Khanpur Gangche High School, Khanpur, West Bengal, India,

Email: tarasankar.math07@gmail.com

Abstract : This paper explores the significant role of mathematics in two distinct realms: examination scheduling problems and video games. In the context of examination scheduling problems, the paper delves into the application of graph theory. A case study is presented, where in a university offers various course combinations, and the objective is to minimize the number of examination periods required while ensuring that students can take any combination without conflicts. Graph theory provides a systematic approach to represent the relationships between courses and derive optimal schedules, demonstrating its efficacy in solving real-life scheduling challenges. Shifting focus to video games, mathematics serves as a fundamental tool for designing game mechanics, graphics, and algorithms, contributing to the immersive and interactive gaming experience. The utilization of mathematical models, such as matrices and vectors, is discussed in the creation of realistic animations and simulations within video games. Through these discussions, the paper emphasizes the diverse and impactful applications of mathematics, particularly in the dynamic domains of examination scheduling and video games.

Keywords: vertex colouring, vector, matrix, graphs.

*Corresponding author: S. Amanathulla

Received: 02.11.2013; **Accepted**: 30.01.2024; **Published**: 03.04.2024

1. Introduction

As civilization has progressed, the significance of various scientific fields has become increasingly apparent throughout different eras. In the early stages of human existence, when society was uncivilized and coexisted with other animals, a transformative moment occurred with the discovery of fire. This development elevated humans above their counterparts, establishing them as the rulers of both living and nonliving entities. Subsequently, a curiosity forexploration and discovery emerged. Over time, a succession of inventions unfolded, including pivotal milestones such as the creation of tools, the wheel, advancements in agriculture, the harnessing of electricity, the creation of the electric motor, the industrial revolution, and the advent of computers. The integration of computers with the internet marked a new era, propelling the world into a dimension characterized by rapid technological progress.

Chemistry played a crucial role in the industrial revolution, while, in the current computer era, mathematics has assumed a significant role. Video games have emerged as one of the most popular and widespread applications of computer technology, with mathematics serving as a fundamental component in their development and design. The principal branches of mathematics employed in video game development encompass algebra, trigonometry, discrete mathematics, calculus, linear algebra, and more.

Integral topics in video game mathematics include the cross product, dot product, scaling vectors, reflections, matrices, scalar multiplication, trigonometric functions (cos, sin, tan), delta time, domain, range, and others [2, 3, 5]. Advanced video games often incorporate a combination of these mathematical concepts, whereas simpler games may only require trigonometry and algebra or a limited set of mathematics involving scalar multiplication.

The utilization of mathematical procedures is evident in various aspects of video game mechanics, enabling characters to perform actions such as walking at specific angles, sliding down surfaces, firing bullets from guns, and executing jumps. While the mathematics involved in video game scripting may be relatively straightforward, the field becomes more complex and mentally challenging in the context of game architecture. In game development, mathematical calculations are either simulated by developers or handled by the game engine at runtime to execute the required operations.

This chapter delves into the diverse applications of several mathematical topics in the context of video game design, highlighting the indispensable role that mathematics plays in shaping the virtual worlds we experience in modern gaming.

Graph Theory holds a pivotal position within the realm of Discrete Mathematics. Graphs play a crucial role as they serve as the exclusive means of visually representing information. Graphs have the unique ability to convey information that would otherwise require extensive textual descriptions. The principles of graph theory find extensive application in various computer science domains. Notable problems in graph theory include the Shortest Path Problem, Traveling Salesman Problem, Job Sequencing Problem, Conditional Covering Problem, Graph Colouring Problem, and more. Graph Coloring stands as a significant problem within graph theory, playing a crucial role in addressing a multitude of real-world challenges such as scheduling, traffic phasing, resource allocation, task assignment, and more [1, 20, 25]. The concept of labeling in graph theory serves as a generalization of the Graph Coloring problem and finds widespread application in solving issues like frequency assignment in wireless communication [10-19, 22, 24, 26, 27, 29, 30, 34, 37, 39]. Fuzzy graphs are the extension of graph theory, which have lot of applications in reality [4, 6-9, 21, 23, 28, 31-33, 35, 36, 38, 40-43].

The remaining part of this article are organized as follows: the concepts of vectors and its applications in video games are discussed in Section 2. In Section 3, graph colouring for various subclasses of graphs have studied. In Section 4, an application of graph colouring for scheduling examination in an university have presented. Section 5 is for brief conclusion.

3. Graphs with basic definitions:

A graph $G = (V, E)$ consisting of the set V together with the edge set E which is a subset of the Cartesian product $V \times V$ of the set V. Thus the mathematical representation of the graph shown in Figure 4 is given by $G = (V, E)$ where $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, c), (c, d), (b, d), (b, c)\}$. Graph theory has started its journey from a historically notable problem of seven bridges of Konigsberg solved by Leonhard Euler in 1736 in his literature.

The four lands of the city Konigsberg in Prussia (now Kaliningrad, Russia) including two large islands Kneiphof and Lompse were divided by Pregel river, which was connected each other by seven bridges (see Fig. 1).

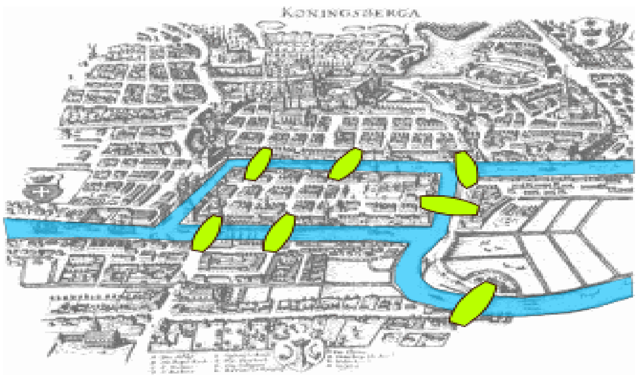

Fig. 1. Map of Konigsberg, Prussia (now Kaliningrad, Russia)

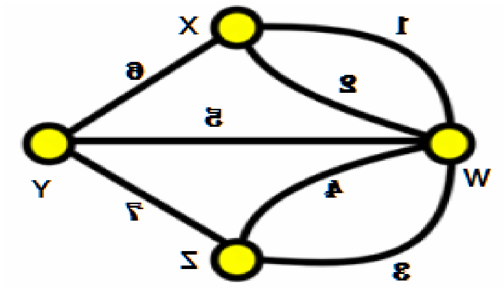

Fig. 2. Graph model of Konigsberg bridge

The problem is to devise a walk through those lands using all of those seven bridges once and only once. At that time, the problem was solved by Euler by considering its equivalent abstract model which was then named as Graph, and from then a new branch ofmathematics has opened for further development. He had considered every four lands as vertices and bridges between them were taken as edges of Graph (see Fig. 2). To understand better how graph theory helps to design some strategic games, thefirst step is to understand how the relationship between objects can be modeled as a graph and then finding paths between twospecified objects. For this study the following example:

Example: Consider a transportation strategy problem in a country among some cities of it. Consider the country has five cities A, B, C, , E among which one has to find a shortest transportation route or shortest path.The cities are connected by the routes, it is seen that there are 3 possible routes (visiting each city only once) by which transportation can be made from the city A to the city E. They are $A \rightarrow B \rightarrow C \rightarrow E$, $A \rightarrow C \rightarrow E$, $\rightarrow D \rightarrow C \rightarrow E$. It is very easy to decide that the shortest route from the city A to the city E is $A \rightarrow C \rightarrow E$. But, this decision becomes complicated when the number of vertices (nodes) and edges (lines) increases. Again, in these type of transportation problems, the most demanding question arises that "What is the minimum cost to transport certain things from source city to a destination city?" In these cases, the edges can be assigned cost to transport as edge weights. Now consider the costs of transportation .

In this case, the route $A \to D \to C \to E$ gives the minimum cost to transport certain things from the source city A to the destinationcity E. Hence this path will be the shortest path. For a complex graph with a huge number of vertices and edges, finding shortest path is a challenging problem for researchers. Many algorithms are designed to find the shortest path in several kinds of literature. Further graph theory has developed in several ways like fuzzy graph theory [4] where, edge weights and vertex weights are taken as fuzzy numbers. Since, fuzzy deals with several vague concepts such as tall man, about 5 k.m., within 2 hrs., etc., researches on fuzzy graph theory increases exponentially. Now, the question arises how this graph theory helps the computer to guide and take strategy? Suppose, in a shooting game a shooter has to shoot an object through the journey of several obstacles. In this case, when the computer plays an opponent, it has to decide the minimum number of objects to pass through. This problem is the same as finding the best shortest path in the equivalent graph model. Here the shortest path algorithm is fundamental to consider for programmers.

3.1 Graph Colouring

In the context of any simple graph G, graph coloring, or vertex coloring, involves assigning colors to the vertices of G in a manner that ensures no two adjacent vertices have the same color. The primary goal in graph coloring is to minimize the number of colors used. This number is called the chromatic number of the graph and is denoted by $\chi(G)$.

If a graph G is l -chromatic, then coloring all the vertices of G requires l colors. It cannot be colored using l or fewer colors. The graph has 6 vertices and it is coloured using only 3 different colors. (see Figure 3).

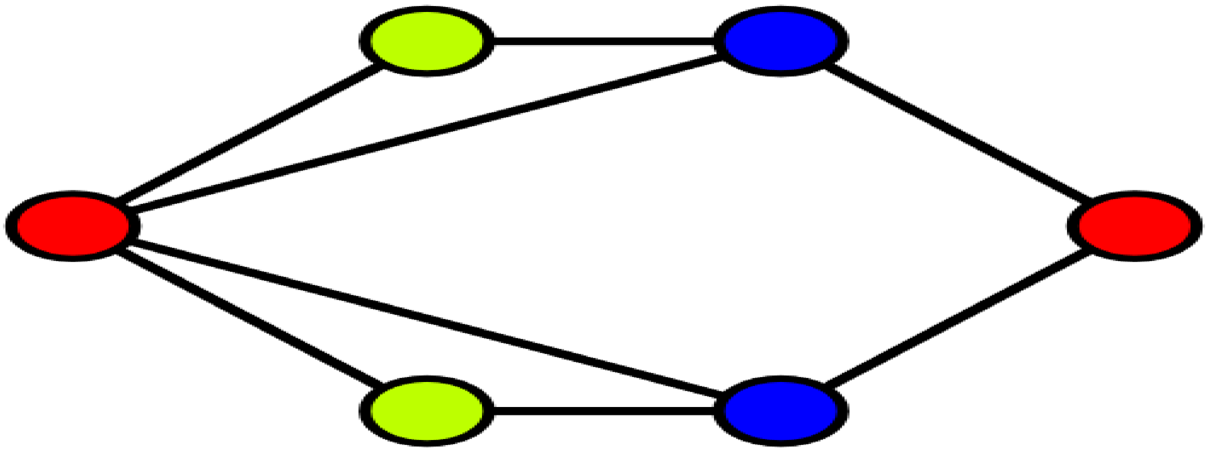

Fig. 3: A three chromatic graph

3.2 Vertex Colouring

Next, we will explore the coloring of certain special types of graphs.

Path P_n **:** Consider a path graph Pn with n (≥ 2)vertices. We can color the vertices with odd indices by blue(B) and the vertices with even indices by yellow(Y) (refer to Fig. 4). Consequently, Pn is 2-colorability.

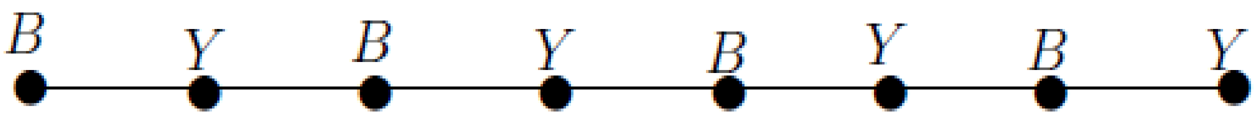

Fig. 4: P_n is 2 -colourable, $n \geq 2$

Cycle C_n **:** A cycle graph Cn with an even number of vertices can be colored using only two colors, such as blue and yellow (refer to Figure 10). However, if the number of vertices is odd, precisely three colors are required for coloring. Consider the vertices of Cn labeled as $V1, V2, \ldots, Vn$. Assign the color violet to $V1$ and yellow to $V2$. Since $V3$ is adjacent to both $V1$ (colored violet) and $V2$ (colored yellow), $V3$ requires an additional color, say blue. Thus, the vertices $V1, V2$, and $V3$ can be colored with violet, yellow, and blue, respectively (refer to Fig. 5).

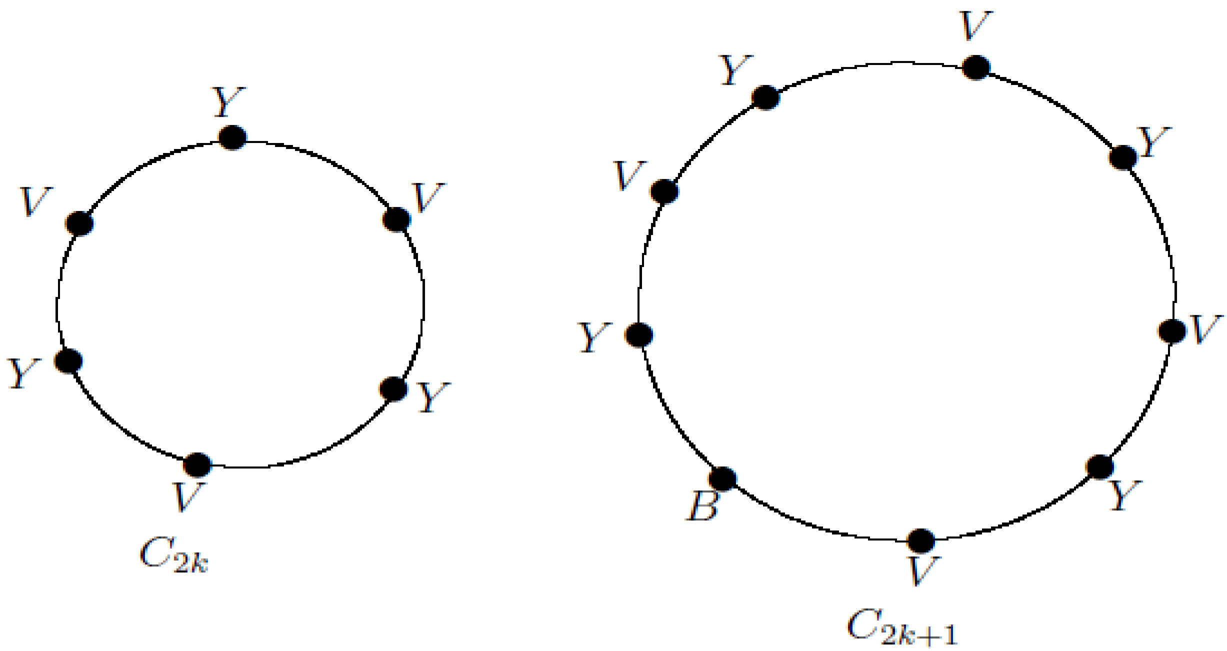

Fig. 5: Colouring of C_n

Tree: Coloring a tree is a straightforward process. Every tree is 22-colorable, as illustrated in Fig.6.

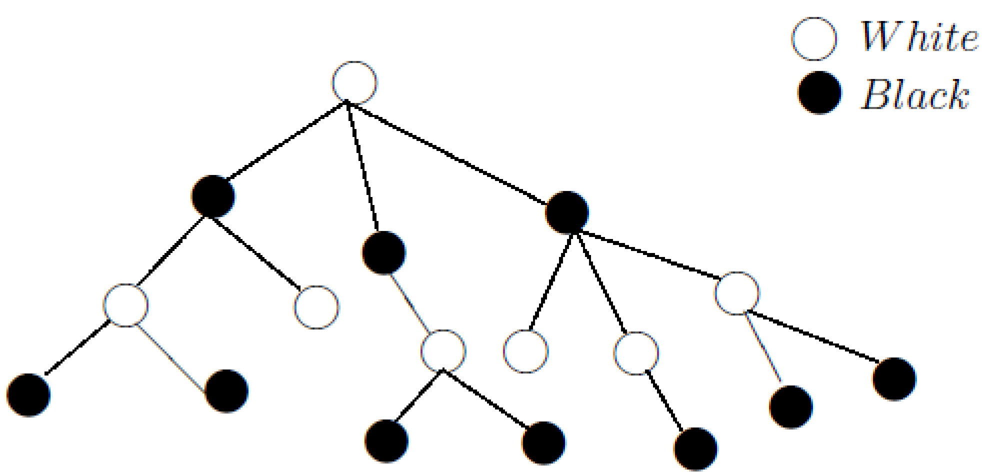

Fig. 6: Tree is 2 -colourable

Complete bipartite graph $K_{m,n}$**:** A complete bipartite graph $K_{3,4}$ is shown in Fig. 7.

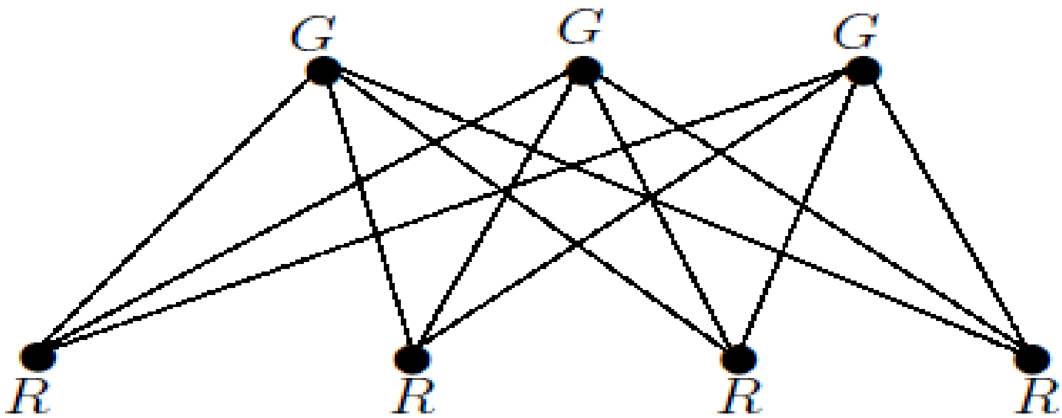

Fig. 7: $K_{3,4}$ is 2 -chromatic

In general, $\chi(K_{m,n}) = 2$.

4 Applications of Graph Theory in Examination Scheduling

Graph theory is a powerful tool for solving a wide range of real-life problems. Below, we highlight one important application of graph theory in practical scenarios.

Consider a university that provides the following combinations of courses for students, with each student required to choose one group course:

Group 1: English, Computer Science , Biology.

Group 2: Physics, Chemistry, Mathematics.

Group 3: Mathematics, Physics , Computer Science.

Group 4: English, Chemistry, Physics.

Group 5: Chemistry, Biology , Mathematics.

Group 6: Psychology, Chemistry, Biology.

Group 7: Biology, Mathematics, Psychology.

Group 8: English, Chemistry, Computer Science.

The objective is to determine the minimum number of examination periods needed for the seven specified courses, ensuring that students can take any combination without conflicts. Additionally, we aim to identify a feasible schedule utilizing this minimal number of periods.

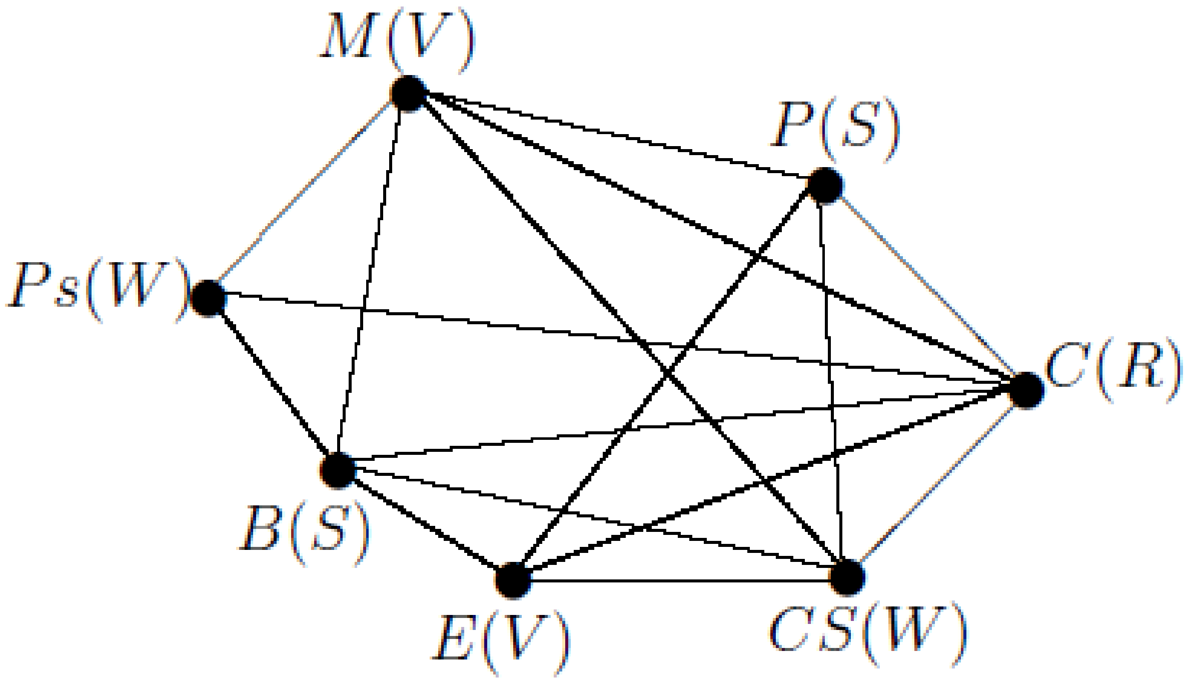

Fig. 8: Graph corresponding to 8 groups of course. The alphabets within the parenthesis represent colours.

A graph is constructed with seven vertices labeled as English, Biology, Computer Science, Physics, Mathematics, Chemistry, and Psychology, denoted by E, B, CS, P, M, C and Ps. Edges are drawn between vertices if the corresponding subjects belong to the same group. For instance, English, Biology, and Computer Science form Group 1, so vertices E, B and CS are connected by edges. Similarly, Physics, Mathematics, and Chemistry form Group 2, connecting vertices P, M and C. The graph for this representation is depicted in Fig. 8. We colour the vertices M by V (violet), (P, C, CS, B and Ps can not be coloured by V), P by S (silver) the colour of C can not be V and S), C by R (red), CS by W (white), E by V, B by S and PS by W (see Fig. 8). Note that only four colors are required to color this graph. Vertices sharing the same color are not adjacent, allowing for the scheduling of corresponding subjects in the same examination period. For example, since Mathematics (M) and English (E) share the same color, their examinations can be held simultaneously. The computer schedule is presented below.

Period	Subjects
$1(V)$	English, Mathematics
$2(S)$	Biology, Physics
$3(R)$	Chemistry
$4(W)$	Psychology, Computer Science

2. Vectors and Video Games

The initial phase of video game design hinges on the application of vector [3,6] concepts, governing the character's movement in various directions—upward, downward, leftward, rightward, and even jumping. A vector serves as a mathematical tool capable of representing both magnitudes and the direction of movement for a body or any other entity. In the realm of advanced video game development, vectors are frequently employed to define changes in position and can be manipulated

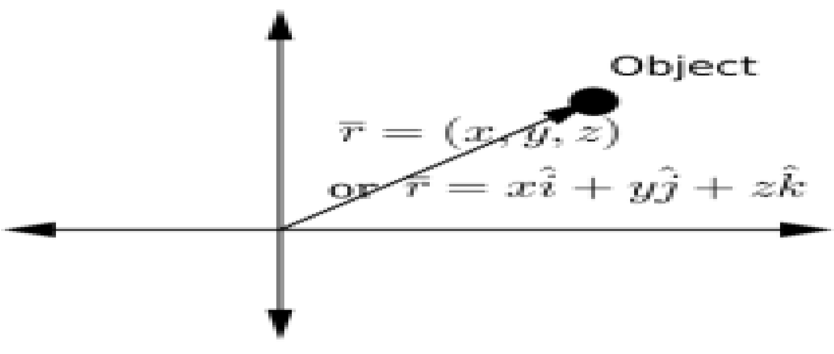

Fig. 9: Vector representation of an object

While the physical world exists in up to three-dimensional space, mathematically, vectors can manifest in one-dimensional, two-dimensional, three-dimensional, four-dimensional, and beyond. Several terms associated with the use of vector concepts include:

Direction: A vector resembles an arrow indicating a specific direction, and indeed, this is its practical application. For instance, if an individual possesses a vector pointing south, they can direct all their units to move in the southern direction.

Unit Vector: A specific instance of a directional vector is a vector of length of 1. This type of vector is referred to as a normal vector, or a unit vector.

Velocity: Velocity is a vector quantity that describes the rate of motion of an object. It includes the speed of the object's and the direction in which it is moving.

Operations of vectors: There are many algebraic operations on vectors such as addition, subtraction, cross product, dot product, and so forth. In the programming realm, developers have employed the n-tuple coordinate system to depict a vector.

In Fig. 9, the object's position is denoted by the vector $\boldsymbol{r} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ or $\boldsymbol{r} = x\hat{\imath} + y\hat{\jmath} + zk$ with its direction along the line joining the object point and the origin. Vectors can consider as variables in mathematics, can undergo operations such as addition, subtraction, and other algebraic expressions. Vector addition combines the displacements described by individual vectors into a resultant vector. For instance, if an object moves along vector A and subsequently along vector B, the cumulative effect is equivalent to moving along vector C. Fig. 10 illustrates the pictorial representation of vector addition, where vectors are depicted along lines.

In case of vector subtraction $(A - B)$, the vector B is inverted (i.e., its direction is changed) and then added to the first one. This scenario is depicted in Fig. 11.

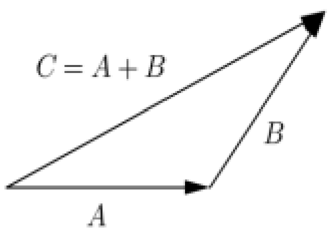

Fig. 10: Addition of two vectors

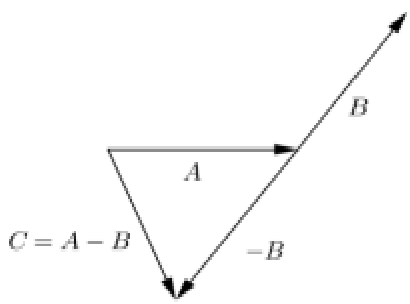

Fig. 11: Subtraction of two vectors

This resultant vector is termed the velocity vector. Hence, vectors can be categorized into two representations: the position vector and the velocity vector.

The dot product of vectors: *The dot product is a mathematical operation that takes two vectors and produces a scalar. It is denoted by a dot (·) between the vectors. For two vectors* ***a*** *and* ***b****, the dot product is calculated as follows:*

a· **b** = |**a**||**b**| $cos\ \theta$, where θ stands for the angle between the vectors. In component form, if

a=(a_1, a_2, a_3 **)** and **b=(** b_1, b_2, b_3 **),** then **a**.**b**= $a_1b_1 + a_2b_2 + a_3b_3$ **.**

Cross product of vectors: The cross product or vector product is a mathematical operation that takes two vectors and produces a third vector which is perpendicular to the plane containing the original vectors. The cross product of the vectors **a** and **b** is defined by

a × **b** = |**a**||**b**| $sin\ \theta\, \hat{n}$

where θ stands for the angle between **a** and **b** and $\hat{n}$ is the unit vector perpendicular to the plane containing **a** and **b** in the direction given by right-hand rule.

Uses of vectors in video games:

In the preceding section, we familiarized ourselves with vectors. In this section, we explore how vectors find application in video games. Let's consider a scenario in a video game where cannonballs are thrown towards a target, and the computer needs to track the position of the cannonball every second following its initial trajectory. In object-oriented programming, if we define a vector as an object, equipped with fundamental operations like addition, subtraction, cross product, dot product, velocity, etc., we can efficiently obtain the desired results using the principles of vector mathematics.

Algorithm Move AlongX:

//This coding moves the cannonball along the horizontal direction from aninitial point.

Read *a,b,c,v;*

position=(*a,b,c*); //The cannonball is initially at a point in 3-dimensional space which is at a distance *a* units from x-axis, *b* units from y-axis and *c* units from z-axis.

velocity=(*v*,0,0); //Here we set the velocity of the cannonball at *v* points per second along x-axis.

*position=position+velocity; //*This operation moves the cannonball to the position (*a+v,b,c*) along horizontal direction.

//Here we have used the vector addition.

End MoveAlongX.

Similarly, to move the cannonball along the y-axis and z-axis (vertical direction) the following algorithms can be used.

Algorithm MoveAlongY:

//This coding moves the cannonball along the y-axis from an initial point. Read *a,b,c,v;*

position=(*a*,*b*,*c*); //The cannonball is initially at a point in 3-dimensional space which is at a distance *a* units from x-axis, *b* units from y-axis and *c* units from z-axis.

velocity=(*0*,*v*,0); //Here we set the velocity of the cannonball at *v* points per second along y-axis.

*position=position+velocity; //*This operation moves the cannonball to the position (*a*,*b*+*v*,*c*) along y-axis.

//Here we have used the vector addition.

End MoveAlongY.

Algorithm MoveAlongZ:

//This coding moves the cannonball along the z-axis from an initial point. Read *a,b,c,v;*

position=(*a*,*b*,*c*); //The cannonball is initially at a point in 3-dimensional space which is at a distance *a* units from x-axis, *b* units from y-axis and *c* units from z-axis.

velocity=(*0,0,v*); //Here we set the velocity of the cannonball at *v* points per second along z-axis.

*position=position+velocity; //*This operation moves the cannonball to the position (*a*,*b*,*c*+*v*) along z-axis.

//Here we have used the vector addition.

End MoveAlongZ.

2.1 A matrix in video games

Matrix [10] is another important topic to use in video game programming. Matrix is a rectangular array elements arranged in rows and columns. A matrix of order 3 × 3 can be written as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

A matrix can serve as an alternative representation of vectors. Specifically, the columns of a matrix can be interpreted as column vectors, while the rows of the matrix are referred to as row vectors. Before going to use the matrix in video games, some basic matrix operations are described.

Addition and subtraction of matrices:

Let

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \ldots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \ldots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \ldots & b_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{m1} & b_{m2} & b_{m3} & \ldots & b_{mn} \end{bmatrix}$$

be two matrices each of order $m \times n$. Here ij-th or (i, j)-th element of the matrix A is a_{ij}. Similarly, ij-th or (i, j)-th element of the matrix B is b_{ij}. The addition of these two matrices can be obtained by adding the elements of their respective positions. Notice that addends in the addition of matrices are of same order. Thus,

$$A + B = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & \ldots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & \ldots & a_{2n}+b_{2n} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} & \ldots & a_{3n}+b_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & a_{m3}+b_{m3} & \ldots & a_{mn}+b_{mn} \end{bmatrix}$$

Similarly,

$$A - B = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} & a_{13} - b_{13} & \dots & a_{1n} - b_{1n} \\ a_{21} - b_{21} & a_{22} - b_{22} & a_{23} - b_{23} & \dots & a_{2n} - b_{2n} \\ a_{31} - b_{31} & a_{32} - b_{32} & a_{33} - b_{33} & \dots & a_{3n} - b_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & a_{m3} - b_{m3} & \dots & a_{mn} - b_{mn} \end{bmatrix}$$

Multiplication of matrices*:*

Multiplication of matrices is a bit different. In general, multiplication of matrices is non-commutative. That is, $AB \neq BA$. Two matrices A and B can be multiplied and the result AB is obtained iff the numbers of column of A is equal to the number of rows of B.

A matrix is multiplied by another matrix by the vector dot product. The dot productof i-th row vector of A and j-th column vector of B is the ij-th element of AB. Thusfor any given two matrices

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1p} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2p} \\ b_{31} & b_{32} & b_{33} & \dots & b_{3p} \\ \dots & \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{np} \end{bmatrix}$$

we have

$$AB = \begin{bmatrix} \sum_{k=1}^{n} a_{1k}b_{k1} & \sum_{k=1}^{n} a_{1k}b_{k2} & \sum_{k=1}^{n} a_{1k}b_{k3} & \dots & \sum_{k=1}^{n} a_{1k}b_{kp} \\ \sum_{k=1}^{n} a_{2k}b_{k1} & \sum_{k=1}^{n} a_{2k}b_{k2} & \sum_{k=1}^{n} a_{2k}b_{k3} & \dots & \sum_{k=1}^{n} a_{2k}b_{kp} \\ \sum_{k=1}^{n} a_{3k}b_{k1} & \sum_{k=1}^{n} a_{3k}b_{k2} & \sum_{k=1}^{n} a_{3k}b_{k3} & \dots & \sum_{k=1}^{n} a_{3k}b_{kp} \\ \dots & \dots & \dots & \dots & \dots \\ \sum_{k=1}^{n} a_{mk}b_{k1} & \sum_{k=1}^{n} a_{mk}b_{k2} & \sum_{k=1}^{n} a_{mk}b_{k3} & \dots & \sum_{k=1}^{n} a_{mk}b_{kp} \end{bmatrix}$$

Now, rotating of an object in space is another essential task of a programmer in video game programming. For rotating an object in a 2-dimensional plane, we use a rotation matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

In order to rotate an object, its position is represented by a position vector, often treated as a column vector **v**. The resulting vector after rotation can be acquired through matrix multiplication, denoted as ***R*v**. In a three-dimensional system, a fundamental rotation involves rotating vectors by an θ about the x-,- or

z- axis using the right-hand rule. Three basic rotation matrices are utilized for this purpose.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

, $$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \text{ and}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Each of these rotation matrices has the capability to rotate an object, represented by a position vector treated as a column vector, in a counterclockwise direction. This occurs when the axis about which the rotation takes place points towards the observer, the coordinate system is right-handed, and the angle θ is positive. For instance, to perform a rotation of the x-axis by an angle of 90 degrees about the z-axis, the following steps are executed:

$$R_z(90^\circ)\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Other rotational matrices can be obtained from these three using matrix multiplication. This basic rotation methodology can be used in cannonball game also. Here is an algorithm to rotate a cannonball which is moving in any plane of a 3- dimensional space.

Algorithm TurnObject

//This algorithm turns the object in a certain angle specified by the user. Read θ,

$v=(v_x, v_y, v_z)$; //Here θ is the angle by which the vector v is to be rotated.

```
        if v_x = 0 then
        rotatingVector = R_x(θ)v;
        else if v_y = 0 then
        rotatingVector = R_y(θ)v;
        else if v_z = 0 then
        rotatingVector = R_z(θ)v;
else
Print "I can rotate only the vectors when they are only on one of the
planes xy, yz or zx.
end if;
```

End TurnObject

2.2 Probability and Statistics in Video Games

Human beings have harbored a deep fondness for gaming throughout the various stages of civilization. Across the ages, numerous types of games and sports have been invented. Many games hinge on predictive skills, while sports serve as demonstrations of physical prowess. Gambling stands out as one of the most popular games, demanding a keen sense of prediction. Gamblers have perpetually sought to grasp the odds in games of chance, leading to the evolution of the scientific study of probabilitywithin mathematics.

The foundations of probability theory were laid in the sixteenth century by the Italian polymath Gerolamo Cardano. Subsequently, in the seventeenth century, mathematicians Pierre de Fermat and Blaise Pascal further pioneered the field of probability. Jakob Bernoulli and Abraham de Moivre, in later periods, treated probability as a distinct branch of mathematics. Since then, probability has blossomed into a fully-fledged and essential discipline within the realm of mathematics.

In the domain of probability theory [5], discussions revolve around the likelihood of events occurring, quantified as a numerical value between 0 and 1 (equivalently, 0% to 100%). If the chance of an event happening is 100%, it is certain to occur. A 50% chance indicates equal likelihood of occurrence or non-occurrence.

Consider a dice game where the die has six faces numbered 1 through 6. Can anyone predict which number will appear face-up before the die is thrown?

No one can tell exactly but definitely can tell that the chance of coming any one of the faces is $\left(\frac{100}{6}\right)\%$. In a probabilistic approach, this value becomes $\frac{1}{6}$.

$$\text{Prob(event)} = \frac{\text{Number of favorable outcomes}}{\text{Number of possible outcomes}}$$

Probability and statistics play a crucial role in the realm of video game design. Consider a card-playing game scenario where a user opts to play against a computer opponent. In such instances, the programmer needs to establish a winning probability corresponding to the chosen difficulty level. This probability is contingent on factors such as the distribution of cards and the user's gaming techniques. These calculations often rely on classical probabilistic theories, outlined as follows:

In a TETRIS game, various shapes of objects appear randomly, and players aim to arrange them to form a horizontal line without any gaps. The programmer's responsibility includes the effective randomization of these objects, a task that can be tackled by calculating the probabilities associated with different shapes.

2.3 Graphs and Strategic Games:

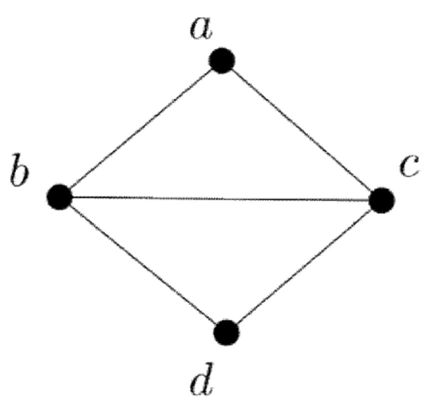

Fig. 12: Graph with 4 vertices and 5edges.

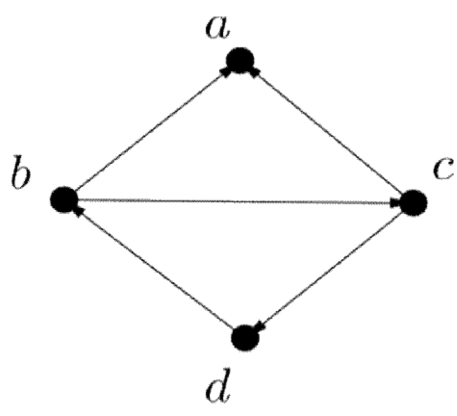

Fig. 13: Directed graph with 4 vertices

Due to the advancement of computers, various strategic games are invented and become popular among youngsters as well as adults. There are so many strategic games such as Dominoes, Checkers or Draughts, Monopoly, Scrabble,Kensington, Backgammon, Risk, etc. In each of these games, the human player competes with a computer as their opponent. These games do not require too much graphics, but it demands strategy to decide what will happen upon a decision taken by the player and this should be programmed so that the computer can take right strategy against a decision taken by a real player. Because computers cannot think for themselves yet! They need to be told correctly whatto do. In most of the strategic games such as Checkers, Chess, Monopoly, Diplomacy, etc., computers have to findthe best path to move the object from one point to another point. In this case, the task of a programmer is to give exact instructions to the computer at every stage of the journey of the object. These type of problems are called *pathfinding*. The problems can be nicely solved by using Graph Theory.

Graphs are finished up of *vertices (nodes or points)*, and the relations between them are represented by straight or curved lines which are called *edges (arcs or lines)*. If the edges have directions then the correspondinggraph is called a directed graph (see Fig. 13) otherwise the graph is undirected (see Fig. 12). Graphs find application in representing diverse relationships and processes within various fields. In computer science, graphs are secondhand to represent very complicated networks, data structure handling and also in theprogramming of video games.

5. Conclusion

In this article, the exploration of mathematics in video games and examination scheduling underscores its diverse applications. In the gaming industry, mathematical models enhance graphics, animations, and algorithms, enriching the gaming experience. On the academic front, graph theory proves instrumental in optimizing examination schedules, demonstrating the practical significance of mathematics in real-world problem-solving. The pervasive impact of mathematics in these domains emphasizes its ongoing role in addressing complex challenges and enhancing both entertainment and educational sectors.

Acknowledgement: The author give a sincere thanks to the respected Editors and Reviewers for their valuable suggestions

to improve the quality of the work.

References:

[1] Biggs, N., Norman L., B., & Biggs N. (1993). Algebraic graph theory. 67, Cambridge university press.

[2] Brand, L. (1947). Vector and tensor analysis. New York: John Wiley & Sons.

[3] De Groot, Morris H., & Mark J. (2012). Schervish. Probability and statistics. Pearson Education.

[4] Mathew, S., John N. M., & Davender S. M. (2018). Fuzzy graph theory. Springer International Publishing.

[5] Meyer, C. D. (2000). Matrix analysis and applied linear algebra. Siam 71, 1-15..

[6] Pramanik, T., Pal, M., & Mondal, S. (2016). Interval- valued fuzzy threshold graph. Pacific Science Review A: Natural Science and Engineering18(1) , 66-71.

[7] Pramanik, T., Samanta, S., Sarkar, B., Pal, M. (2017). Fuzzy ϕ-tolerance competition graphs. Soft Computing 21(13), 3723-3734.

[8] Pramanik, T., Samanta, S., & Pal, M. (2016). Interval-valued fuzzy planar graphs. International journal of machine learning and cybernetics , 7(4), 653-664.

[9] Pramanik, T., Samanta, S., Pal, M., Mondal, S., & Sarkar, B. (2016). Interval-valued fuzzy ϕ-tolerance competition graphs. SpringerPlus, 5(1), 19-36.

[10] Amanathulla, S., & Pal, M. (2016). L(3,2,1)- and L(4,3,2,1)-labeling problems on circular-arc graphs. International Journal of Control Theory and Applications, 9(34), 869-884.

[11] Amanathulla, S., & Pal, M. (2016). L(0,1)- and L(1,1)-labeling problems on circular-arc graphs. International Journal of Soft Computing, 11(6), 343-350.

[12] Amanathulla, S., & Pal, M. (2017). L(3,2,1)- and L(4,3,2,1)-labeling problems on interval graphs. AKCE International Journal of Graphs and Combinatorics, 14 (3), 205-215.

[13] Amanathulla, S., & Pal, M. (2017). L(3,2,1)-labeling problems on permutation graphs. Transylvanian Review, 25(14), 3939-3953.

[14] Amanathulla, S., & Pal, M. (2017). $L(h_1, h_2, ..., h_m)$ -labeling problems on interval graphs. International Journal of Control Theory and Applications, 10(01), 467-479.

[15] Amanathulla, S., & Pal, M. (2017). $L(h_1, h_2, ..., h_m)$ -labeling problems on circular-arc graphs. Far East Journal of Mathematical Sciences, 102(6), 1279-1300.

[16] Amanathulla, S., & Pal, M. (2018). Surjective L(2,1)-labeling of cycles and circular-arc graphs. Journal of Intelligent and Fuzzy Systems, 35, 739-748.

[17] Amanathulla, S., & Pal, M. (2018). L(1,1,1)- and L(1,1,1,1)-labeling problems of square of paths, complete graphs and complete bipartite graphs. Far East Journal of Mathematical Sciences, 106(2), 515-527.

[18] Amanathulla, S., Sahoo, S., & Pal, M. (2019). L(3,1,1)-labeling numbers of squares of paths, complete graphs and complete bipartite graphs. Journal of Intelligent and Fuzzy Systems, 36, 1917–1925.

[19] Amanathulla, S., & Pal, M. (2020). Pal, L (h, k)-Labeling of Intersection Graphs, in Book, Handbook of Research on Advanced Applications of Graph Theory in Modern Society. IGI Global, 135-170.

[20] Amanathulla, S., & Pal, M. (2020). Involvement of Graph Theory in Reality, in Book, Mathematics and Its Connections to Real World. Kindle Direct Publishing, (2020) 113-131**.**

[21] Amanathulla, S., Bera, B., & Pal, M. (2021). Balanced picture fuzzy graph with application. Artificial Intelligence Review, 54 , 5255-5281.

[22] Amanathulla, S., Muhiuddin, G., Al-Kadi, D., & Pal, M. (2021). Distance two surjective labeling of paths and interval graphs. Discrete Dynamics in Nature and Society, doi.org/10.1155/2021/9958077.

[23] Hossein, R., Muhiuddin, G., Amanathulla**, S.,** Borzooei, R. A., & Pal, M. (2021). A Study On Cubic Graphs With Novel Application. Journal of Intelligent and Fuzzy Systems, 40, 89–101.

[24] Amanathulla, S., & Pal, M. (2021). L(3,2,1)-labeling problems on trapezoid graphs. Discrete Mathematics Algorithms and Applications, 13(5) , doi.org/10.1142/S1793830921500683.

[25] Amanathulla, S., Bera, B., & Pal, M. (2021). Real world applications of discrete mathematics. Malaya Journal of Matematik, 9(1), 152-158.

[26] Amanathulla, S., Bera, B., & Pal, M. (2021). L(2,1,1)-labeling of circular-arc graphs. Journal of Scientific Research, 13(2), 537-544.

[27] Amanathulla, S., Bera, B., & Pal, M. (2021). Surjective L(3,1)-labeling of circular-arc graphs. Malaya Journal of Matematik, 9(1), 925-929.

[28] Amanathulla, S., Muhiuddin, G., Al Kadi, D., & Pal, M. (2021). Multiple attribute decision-making problem using picture fuzzy graph. Mathematical Problems in Engineering, 9937828.

[29] Amanathulla, S., Bera, B., & Pal, M. (2022). L(2,1,1)-labeling of interval graphs. International Journal of Mathematics for Industry, 14(1), 22500031-22500038.

[30] Amanathulla, S., & Pal, M. (2022). L(2,2,1)- labelling problems on square of paths. International Journal of Advanced Intelligence Paradigms, doi**:** 10.1504/IJAIP.2022.10034134.

[31] Amanathulla, S., & Pal, M. (2021). An introduction to picture fuzzy graph and its application to select best routes in an airlines network. In Book, Advanced and application of fuzzy sets and logic, IGI Global, 385-411.

[32] Banerjee, A., & Amanathulla, S. (2022). Application of Picture Fuzzy Bondage Set to Find Least Crowded Passenger-Friendly Railway Division in India. Research Square.

[33] Khatun, J., Amanathulla, S., & Pal, M. (2023). Picture Fuzzy Cubic Graphs with Applications. Journal of Intelligent and Fuzzy Systems, doi.org: 10.3233/JIFS-232523.

[34] Amanathulla, S., Khatun, J., Pal, M. (2023). L(3,2,1)-labeling problem of square of path. International Journal of Mathematics for Industry, doi.org/ 10.1142/S2661335223500077.

[35] Das Adhikari, B., Banerjee, A., Amanathulla, S., & Mondal, S. (2023). International Journal of Mathematics for Industry, doi.org/10.1142/S2661335223500089 .

[36] Bera, B., Amanathulla, S., Mahato, S. K. (2023). A Comprehensive Study of Picture Fuzzy Planar Graphs with Real-World Applications. Journal of Uncertain Systems, doi.org/10.1142/S1752890923500095.

[37] Khatun, J., Amanathulla, S., Peng, S. L. (2023). Minimization of Span in L (3, 1)-Labeling for a Particular Type of Intersection Graphs, in Book Fuzzy Optimization, Decision-making and Operations Research: Theory and Applications, Springer International Publishing, 207-217.

[38] Khatun, J., & Amanathulla, S. (2023). An Application of Neutrosophic Graph in Decision-Making Problem for Alliances of Companies, in Book Fuzzy Optimization,

Decision-making and Operations Research: Theory and Applications. Springer International Publishing, 241-255.

[39] Patra, N., Amanathulla, S., Pal, M., Mondal, S. (2023). A restricted L (2, 1)-labelling problem on interval graphs. TWMS Journal of Applied And Engineering Mathematics, 13(2), 635-648.

[40] Bhattacharya, A., & Pal, M. (2021). Optimization in business strategy as a part of sustainable economic growth using clique covering of fuzzy graphs. Soft Computing; doi: 10.1007/s00500-021-05670-z.

[41] Bhattacharya, A., & Pal, M. (2021). Vertex covering problems of fuzzy graphs and their application in CCTV installation. Neural Computing and Applications; doi:10.1007/s00521-020-05324-5.

[42] Bhattacharya, A., & Pal, M. (2021). Fifth Sustainable Development Goal, Gender equality in India: Analysis by Mathematics of Uncertainty and Covering of Fuzzy Graphs. Neural Computing and Applications; doi.org/10.1007/s00521-021-06136-x.

[43] Bhattacharya, A., & Pal, M. (2021). Fuzzy covering problem of fuzzy graphs and its application to investigate the Indian economy in new normal. Journal of Applied Mathematics and Computing; doi.org/10.1007/s12190-021-01539-4.