Employing Graph Theory to Solve Optimization Problems

Dipali Waghulde, Dr. Madhukar Palve, Dr. Narendra Swami
Research Scholar, Mathematics Department, JJT University, India.
Prof. Ramkrishna More Arts, Commerce and Science College Pune, India.
Department of Mathematics, SJJT University, Rajasthan, India
Corresponding Author: Dipali Waghulde, Dr. Madhukar Palve, Dr. Narendra Swami

Abstract

Graph theory is a mathematical branch that focuses on study of graphs. Graphs are mathematical structures that are used to represent pair wise relations between objects. Graphs consist of vertices (nodes) which are connected by edges (lines). Graph theory explores the properties and applications of graphs in various fields including computer science, operations research and many more. This paper helps to find how graph theory can be applied to solve optimization problems, specifically the Traveling Salesman Problem (TSP), resource allocation and scheduling. The TSP is an NP-hard problem, that involves finding the shortest possible route for visiting a set of cities. Some of the algorithms like Dynamic Programming, Nearest Neighbor, and Genetic Algorithms are used to discourse TSP. Additionally, resource allocation problems can be solved using techniques such as the Hungarian Algorithm, Ford-Fulkerson algorithm. The scheduling problems can be solved using Critical Path Method (CPM) and graph coloring. These problems are explored for their impact in real-world optimization framework. Using graph representations, problem formulations and the applications of advanced algorithms, this paper determines the impact of Graph Theory on optimizing solutions in different fields.

Keywords: Optimization problems, TSP, Resource allocation, Scheduling

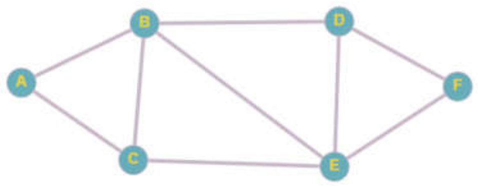
1. Introduction

Optimization problems are wide spread in various fields that include logistics [1], operations management [2] and computer science. The best solution can be obtained from a set of feasible solutions with the help of these optimization problems. Here the "best" solution refers to an outcome that optimizes a specific objective [3]. Graph theory formulates a powerful model and solves the problems by transforming complex relationships into simpler graph structures. Graph theory has numerous applications in solving real-world problems. One of the classic problems in graph theory is the Traveling Salesman Problem (TSP), an NP-hard problem in combinatorial optimization [4]. The objective of TSP is finding the shortest possible route in which the salesman must visit a number of cities exactly once and returns to the starting city. Various algorithms, including Dynamic Programming [6], Nearest Neighbour and Genetic Algorithm [7], are used to solve TSP. Resource allocation and scheduling are other significant applications of graph theory. Resource allocation involves distributing resources among tasks or agents to optimize specific objectives. [5]. Scheduling, on the other hand, deals with the time-based assignment of activities to resources and can be represented using directed acyclic graphs or as a graph coloring problem [9].

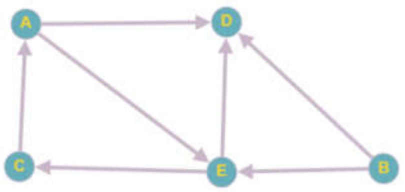
2. Graph Theory Basics:

Graph theory involves the study of graphs, which are mathematical structures used to model pair wise relations between objects. A graph G is defined as a pair (V, E) where V is a set of vertices (nodes) and E is a set of edges (links) connecting pairs of vertices. Graphs can be undirected or directed, weighted or unweighted and can represent a wide range of real-world problems.

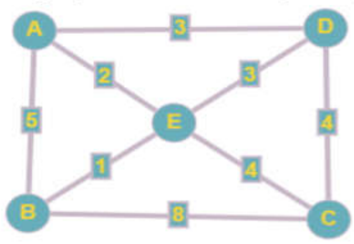
i) **Undirected Graph:** A graph in which no edge has a specific direction assigned to it



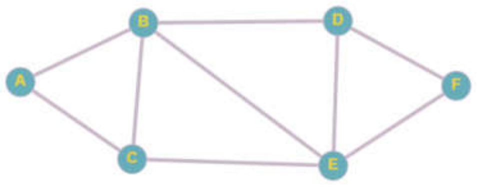
ii) **Directed Graph:** A graph in which every edge has a specific direction assigned to it.



iii) **Weighted Graph:** A graph in which every edge is assigned with a real number is called weighted edge and such graph is called weighted graph.



iv) **Unweighted Graph:** A graph in which no edge is assigned with a real number is called unweighted graph.



3. Optimization Problems:

3.1. The Traveling Salesman Problem (TSP)

Problem Description: The Traveling sales man problem (TSP) is an NP-hard problem in combinatorial optimization [4]. It is an optimization problem in which the objective is to find the shortest possible route by visiting number of cities exactly once with minimum distance travelled and return to the city from where the sales man started to travel. In other words, given a set of cities and distances between them, the objective is to find the Hamiltonian cycle with the minimum total distance. The Hamiltonian cycle is a cycle which covers every vertex exactly once.

Example: Consider 4 cities A(1), B(2), C(3), D(4). Distance between every two cities is given. Representing the cities as vertices and paths between them as edges with weights in kilometers, corresponding to distances in the following matrix.

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 10 & 5 & 12 \\ 2 & 7 & 0 & 11 & 8 \\ 3 & 3 & 5 & 0 & 9 \\ 4 & 13 & 6 & 15 & 0 \end{array}$$

Algorithms used to solve Traveling Salesman Problems:

3.1.1 Dynamic Programming: The Held-Karp algorithm uses a recursive approach with memorization to solve the problem in $O(n^2 2^n)$ time. Dynamic Programming is a classic algorithm strategy to solve optimization problems. It sides as to decompose the original problem into several sub-problems and solve each sub-problem separately, and the current decision is affected by the previous decision. Since the optimal solution of the original problems derived from the optimal solution of the sub-problem, the solution table of the sub-problem needs to be stored in the calculation process to avoid repeated calculations. [6]

Solution of given TSP by Dynamic Programming [20]:

Here we use the general formula:

$g(I, S) = \min \{C_{ik} + g(k, S-\{k\})\}$, $k \in S$ and $S=\{$set of vertices $1,2,3,4\}$

$\therefore g(1, \{2,3,4\}) = \min(C_{1k} + g(k, \{2,3,4\}-\{k\}))$ $\qquad k = 2,3,4$

Now, for 0 vertex: $g(2, \phi) = 7$, $g(3, \phi) = 3$, $g(4, \phi), = 13$,

For 1 vertex:

$g(2,\{3\}) = C_{23} + g(3, \phi) = 11 + 3 = 14$

$g(2,\{4\}) = C_{24} + g(4, \phi) = 8 + 13 = 21$

$g(3,\{2\}) = C_{32} + g(2, \phi) = 5 + 7 = 12$

$g(3,\{4\}) = C_{34} + g(4, \phi) = 9 + 13 = 22$

$g(4,\{2\}) = C_{42} + g(2, \phi) = 6 + 7 = 13$

$g(4,\{3\}) = C_{43} + g(3, \phi) = 15 + 3 = 18$

For 2 vertices:

$g(2,\{3,4\}) = \min\{ C_{23} + g(3,\{4\}) , C_{24} + g(4, \{3\})\} = \min\{11+22, 8+18\} = \min\{33, 26\} = 26$

$g(3,\{2,4\}) = \min\{(C_{32} + g(2, \{4\}), C_{34} + g(4,\{2\})\} = \min\{5+21, 9+13\} = \min\{26, 22\} = 22$

$g(4,\{3,2\})\ \min\{ C_{43} + g(3, \{2\}), C_{42} + g(2,\{3\})\} = \min\{15+12, 6+14\} = \min\{27, 20\} = 20$

$\therefore$ The optimal Solution,

$g(1,\{2,3,4\}) = \min\{g(2,\{3,4\}) + C_{12}, g(3,\{2,4\}) + C_{13}, g(4,\{3,2\}) + C_{14}\}$

$= \min\{26 + 10, 22 + 5, 20 + 12\} = \min\{36, 27, 32\}= 27$

The optimal tour is: 1→3→4→2→1 and length of tour is = 5+9+6+7 = 27

3.1.2 Nearest Neighbour Algorithm: One of the simplest heuristic algorithms to solve the Traveling Salesman Problem is the nearest neighbour algorithm. In this algorithm [7]:
i) Select any city as a starting city.
ii) Find the unvisited city with minimum distance and traverse it.
iii) Now the city is marked as visited.
iv) Continue in this way. If any unvisited city is remaining, then go to step ii)
v) Return to the starting city.

Solution of given TSP by Nearest Neighbour:

Vertex	1	2	3	4
1	-	10	5	12
2	7	-	11	8
3	3	5	-	9
4	13	6	15	-

We start from vertex 1. Nearest vertex of 1 is 3 having distance 5. So visit vertex 3. We cannot visit the remaining vertices in column 3. Nearest vertex of 3 is 1, but we cannot visit back 1. So visit the vertex 2 with minimum distance 5. Nearest vertex of 2 is now 4 with distance 8. So visit 4. Then nearest vertex of 4 is 6, but we cannot visit it. So visit vertex 1 back as all the vertices are visited.
∴ The rout is 1→3→2→4→1 and length of rout is 5 + 5+ 8 + 13 = 31

3.1.3 Genetic Algorithm: This algorithm is widely recognized for tackling difficult problems where it is more challenging to find the best solution. It works on natural selection by evaluating the fitness of each member in a population. The algorithm then generates new individuals through processes like alteration, which introduces randomness, similar to how genetic alterations occur in nature. Finally, it selects the individual with the highest fitness score. When applying a genetic algorithm to the Traveling Salesman Problem (TSP), certain constraints are necessary. For example, each route must visit each city only once to avoid loops, and only valid routes are considered in the algorithm. [7]

Solution of given TSP by Genetic Algorithm
Step 1: Consider all possible paths that visit every city exactly once.
[1, 2, 3, 4], [1, 2, 4, 3], [1, 3, 2, 4], [1, 3, 4, 2], [1, 4, 2, 3], [1, 4, 3, 2]

Step 2: Population Generation
Let the initial population size is 4.
∴ Population is [1, 2, 3, 4], [1, 3, 2, 4], [1, 4, 3, 2], [1, 4, 2, 3].

Step 3: Calculate Distance
Path 1: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 = 10 + 11 + 9 + 13 = \mathbf{43}$
Path 2: $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1 = 5 + 5 + 8 + 13 = \mathbf{31}$
Path 3: $1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 = 12 + 15 + 5 + 7 = \mathbf{39}$

Path 4: $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1 = 12 + 6 + 11 + 3 = \mathbf{32}$

Step 4: Selection of shorter path
Path 2 and Path 4 have shorter length. So they are selected for crossover.

Step 5: Recombination of paths
Parent 1: [1, 3, 2, 4] Parent 2: [1, 4, 2, 3]
After recombination: Child 1: [1, 3, 2, 4] Child 2: [1, 4, 3, 2]
Child 1 and 2 are obtained by taking first half of the parent and remaining cities from other parent.

Step 6: Mutation: Swapping two cities randomly.
Let the mutation be done on child 2 by swapping 3rd and 4th cities.
∴ Child 2 before mutation is [1, 4, 3, 2] and after mutation is: [1, 4, 2, 3]

Step 7: Continuing step 1 to 6 of selection, crossover, and mutation for multiple generations, we find a near-optimal solution $[1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1]$ with minimum distance = 31

Comparing the above three algorithms, Dynamic programming gives the optimal solution.

4. Resource Allocation

Problem Description

Resource allocation includes determining and distributing a set of resources among various tasks or agents to optimize specific objectives, such as cost, efficiency or utility. Properly allocating resources confirms that project resources are used efficiently to achieve maximum impact while bringing into line with team objectives [5].

4.1 Algorithms used to solve Resource Allocation Problems:
Maximum Bipartite Matching:

4.1.1 Hungarian Algorithm: The Hungarian Algorithm is an effective method to find the best possible allocations, by making it ideal for optimal allocation tasks. This optimization technique solves the resource allocation problem for the jobs available. An assignment problem is a specific type of transportation problem where the aim is to allocate a set of resources to an equal number of tasks or activities that minimizes the overall cost or maximizes the total profit of the allocation. [12]. Here, the Hungarian method is employed as a solution technique for determining the optimal resource-task assignments efficiently that reduces the total time required. It is the general Hungarian algorithm used to solve the Assignment Problem.

Example: A company has 4 workers, and it needs to assign each worker to 4 different tasks. The goal is to allocate workers to tasks in a way that minimizes the total time required to complete all tasks. The time each worker takes to complete each task is given in a cost matrix, where the entry C(i, j) represents the time worker i takes to complete Task j.

Cost Matrix (Time Taken in Hours):

	Task 1	**Task 2**	**Task 3**	**Task 4**
Worker 1	**9**	**2**	**7**	**8**
Worker 2	**6**	**4**	**3**	**7**
Worker 3	**5**	**8**	**1**	**8**
Worker 4	**7**	**6**	**9**	**4**

Row min.

$$\begin{bmatrix} 10 & 3 & 8 & 9 \\ 7 & 5 & 4 & 8 \\ 6 & 9 & 2 & 9 \\ 8 & 7 & 10 & 5 \end{bmatrix} \begin{array}{c} 3 \\ 4 \\ 2 \\ 5 \end{array} \quad \text{Subtracting row min}$$

$$\begin{bmatrix} 7 & 0 & 5 & 6 \\ 3 & 1 & 0 & 4 \\ 4 & 7 & 0 & 7 \\ 5 & 2 & 5 & 0 \end{bmatrix} \quad \text{Subtracting column min}$$

Col min 3 0 0 0

Making assignments

$$\begin{bmatrix} 4 & \boxdot & 5 & 6 \\ \boxdot & 1 & \otimes & 4 \\ 1 & 7 & \boxdot & 7 \\ 2 & 2 & 5 & \boxdot \end{bmatrix}$$

As the number of assignments $\boxdot$ are equal to order of the matrix. So these are optimal assignments given by:

Worker 1 →Task 2 (3 hrs),

Worker 2 → Task 1(7 hrs),

Worker 3 → Task 3(2 hrs),

Worker 4 → Task 4 (5 hrs)

∴ The total time for completing all tasks = 3 + 7 + 2 + 5 = 17 hrs.

- **Network Flow:**

4.1.2 Ford-Fulkerson Algorithm: The Ford-Fulkerson Algorithm is mainly used to solve various problems involving network flow. In this algorithm, we assume that G is a finite directed graph where each edge (u, v) has a capacity c(u, v) ≥ 0, representing the maximum flow that edge can support. Each edge also has a flow f(u, v) that must be less than or equal to its capacity. The graph G includes two special vertices: a source (s), which is the only node that can generate flow, and a destination (t), which is the only node that can absorb flow. All other nodes in the network can neither create nor consume flow. The implementation of Ford-Fulkerson involves several functions, that starts with the generation of the residual network. The residual network indicates the additional flow that can be accommodated by each link. This is achieved by subtracting the current flow from the capacity of each link, resulting in the residual network that reflects the current state of the network and the potential flow that can be added to each link.[8]

Example: Consider the following network problem. There is directed edge from one node to another. Each edge is having certain capacity. We can find the maximum flow using Ford-Fulkerson algorithm as follows:

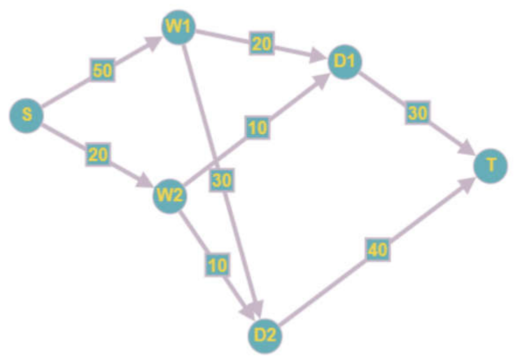


Fig. 1

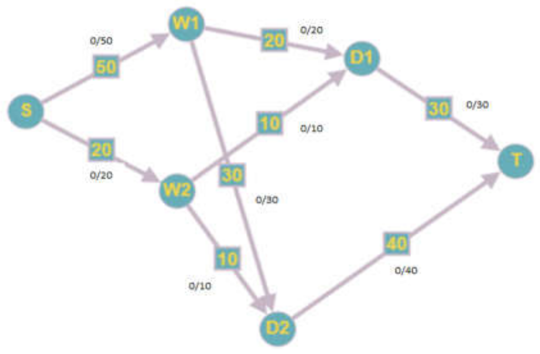


Fig. 2

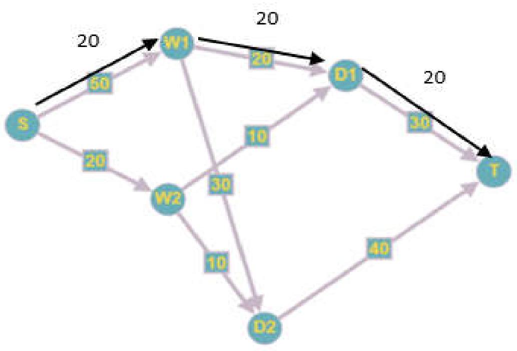


Fig. 3

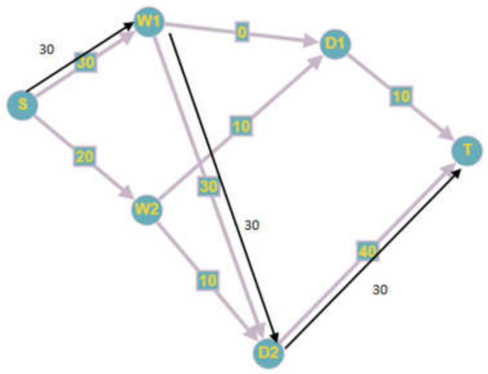


Fig. 4

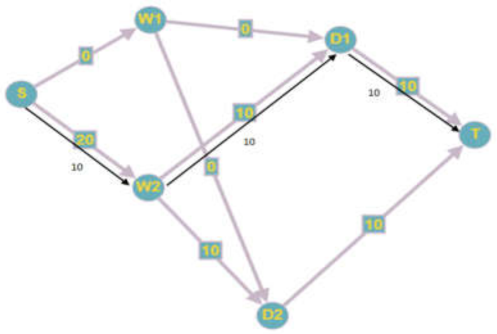


Fig. 5

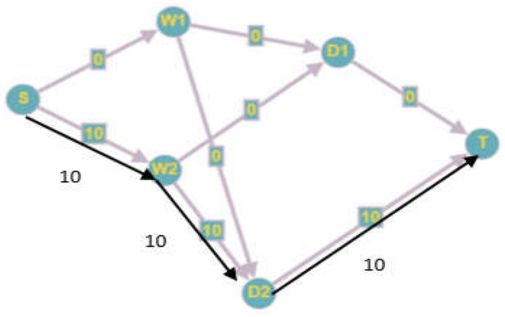


Fig. 6

Description:

Initially the flow of each edge will be zero. (Fig. 2)

i) Consider the first augmented path (using BFS) : $S \rightarrow W_1 \rightarrow D_1 \rightarrow T$. (Fig. 3)

The bottleneck capacity (minimum capacity) is 20 at $W_1 \rightarrow D_1$.

We can push 20 units of flow along this path.

∴ Updated flow in network is 20 units from $S \rightarrow T$.

ii) Second augmenting path : $S \rightarrow W_1 \rightarrow D_2 \rightarrow T$. (Fig. 4) with bottleneck capacity 30 at $W_2 \rightarrow D_2$.

We can push 30 units of flow along this path.

∴ Updated flow in network is 50 units total.

iii) Third augmenting path: $S \rightarrow W_2 \rightarrow D_1 \rightarrow T$. (Fig. 5) with bottleneck capacity 30 at $W_2 \rightarrow D_1$.

We can push 10 units of flow along this path.

∴ Updated flow in network is 60 units total.

iv) Fourth augmenting path: $S \rightarrow W_2 \rightarrow D_2 \rightarrow T$. (Fig. 6) with bottleneck capacity 10 at $W_2 \rightarrow D_1$.

Also at $D_2 \rightarrow T$.

We can push 10 units of flow along this path.

∴ Updated flow in network is 70 units total.

v) No augmenting path is remaining as all capacities on potential path are fully utilized. So the algorithm is terminated.

5. Scheduling

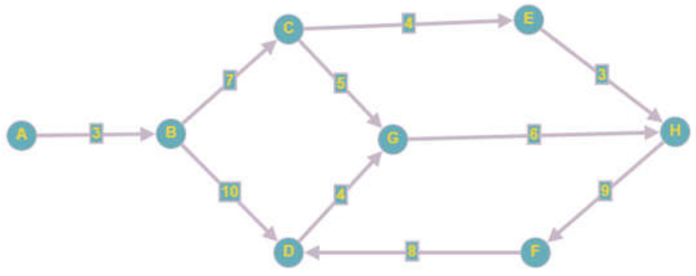
Problem Description

The main objective of scheduling is creating schedules, which involves time-based assignments of activities to resources. This process undergoes various goals and constraints. Scheduling problems can have several application areas, e.g. the scheduling of production operations in manufacturing industry, computer processes in operating systems, truck movements in transportation, aircraft crews, etc. [9]

5.1 Algorithms to solve Scheduling Problems:

5.1.1 Critical Path Method (CPM): To plan and schedule projects, several methods can be used, one of which is the Critical Path Method (CPM). CPM is an integrated network of activities designed to achieve maximum work efficiency. It simplifies determining the total project time by summing the durations of each activity and taking the latest finish time. The critical path do not allow any delay. It ensures the project stays on schedule. CPM is more commonly used method than other traditional methods since it focuses on the most important tasks that keeps the project on track and completed within the planned timeline. [10]

Example: A company is managing a project with activities A, B, C, D, E, F, G, H. Duration of the activities is given by means of the following weighted graph. The problem is to find the critical path that will determine the total project time.



The earliest start (Es), latest start (Ls), earliest finish (Ef) and latest finish (Lf) times are calculated in the following table:

Activity (i-j)	Duration (Dij)	Es (Ei)	Ls=Lj-Dij	Ef=Ei-Dij	Lf (Lj)	Float
A-B	3	0	0	3	3	0 ←
B-C	7	3	12	10	19	9
B-D	10	3	3	13	13	0 ←
C-E	4	10	23	14	27	13
C-G	5	10	19	15	24	9
D-F	8	13	13	21	21	0 ←
D-G	4	13	20	17	24	7
E-H	3	14	27	17	30	13
G-H	6	17	24	23	30	7
F-H	9	21	21	30	30	0 ←

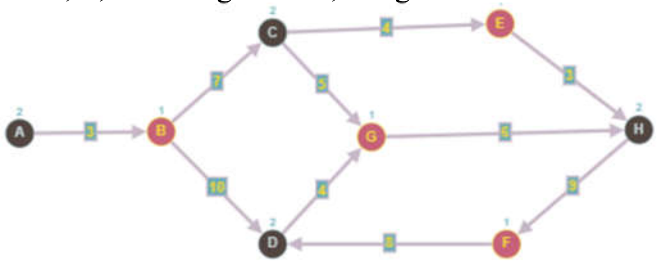
Float: The amount of time an activity can be delayed without affecting the project's finish date (float = LS – ES)

Here the critical path is : A→B→D→F→H and the project will take 30 days to complete. Activities C, E, G can be delayed without delaying the entire project.

5.1.2 Graph Coloring : Graph coloring is widely used in scheduling problems in different area of applications, time slots allocation in time table, allocation of register, assignment of frequency, application in communication network, Sudoku, pattern matching and applications in parallel computing [16]. Graph coloring process involves coloring all the nodes in such a way that no two adjacent nodes have the same color. A properly colored graph is one where this rule is followed; ensuring adjacent nodes have different colours. [11]

Example: Consider the graph in CPM. Here 8 activities A, B, A, C, D, E, F, G, H are to be managed by a company. The aim is to schedule the activities so that they cannot overlap each other. This can be done with the graph coloring.
Consider the activities as vertices.
Assign color 1 to vertex A, since A is not dependent on any other vertex.
Since B is adjacent to A, assign color 2 to B.
Verties C and D are adjacent to B, assign color 1 to vertice C and D, since they do not depend upon each other.

E is adjacent to C, assign color 2 to E.
G is adjacent to vertices C and D having color 1, assign color 2 to G.
F is adjacent to D, assign color 2 to F.
H is adjacent to vertices E, F, G having color 2, assign color 1 to H.



Here activities A, C, D, H can be scheduled in same time period and activities B, G, E, F can be scheduled in same time period, such that no two conflicting activities overlap.

6. Conclusion

Optimization problems play a vibrant role to address real-world challenges along various fields and graph theory provides a robust framework for modeling and solving optimization problems. The Traveling Salesman Problem (TSP), resource allocation and scheduling are leading examples of how graph theory can make simpler, the complex optimization jobs. Applying various algorithms, such as the Nearest Neighbor and Genetic Algorithm for TSP, the Hungarian and Ford-Fulkerson Algorithms for resource allocation and CPM and graph coloring for scheduling, the optimization problems can be solved efficiently. It contributes the advancement in logistics, operations and beyond. The various algorithms used, demonstrate the wide applications and importance of optimization methods in improving decision-making processes.

7. References

[1] Castaneda, J.; Ghorbani, E.; Ammouriova, M.; Panadero, J.; Juan, A. A. Optimizing Transport Logistics under Uncertainty with Simheuristics: Concepts, Review and Trends. Logistics 2022, 6, 42. https://doi.org/10.3390/logistics6030042

[2] Artificial Intelligence Review, 2023, Volume 56, Number 1, Page 253 Idrees Alsolbi, Fahimeh Hosseinnia Shavaki, Renu Agarwal, Gnana K Bharathy, Shiv Prakash, Mukesh Prasad

[3] Gunantara, N., & Ai, Q. (2018). A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, *5*(1). https://doi.org/10.1080/23311916.2018.1502242

[4] Jan Scholz. 2018. Genetic Algorithms and the Traveling Salesman Problem a historical Review. ACM Trans. Web0, 0, Article0 (2018), 11pages. https://doi.org/10.13140/RG.2.2.22632.78088

[5]https://asana.com/resources/resource-allocation

[6] Zhang, J. (2021, November). Comparison of various algorithms based on TSP solving. Journal of Physics: Conference Series, 2083(3). doi:10.1088/1742-6596/2083/3/032007

[7] Haider A Abdulkarim, I.F. (2015, August). Comparision of Algorithms for solving Traveling Salseman Problem. International Journal of Engineering and Advanced Technology (IJEAT), 4(6).

[8] E. P. Neto and G. Callou, "An Approach Based on Ford-Fulkerson Algorithm to Optimize Network Bandwidth Usage," 2015 Brazilian Symposium on Computing Systems Engineering (SBESC), Foz do Iguacu, Brazil, 2015, pp. 76-79, doi: 10.1109/SBESC.2015.21.

[9] Sauer, J. (2003, January). Planning and Scheduling An Overview. ResearchGate.

[10] S. Atin and R. Lubis 2019. Implementation of Critical Path Method in Project Planning and Scheduling IOP Conf. Ser.: Mater. Sci. Eng. 662 022031

[11] Suman De, An efficient technique of resource scheduling in cloud using graph coloring algorithm,Global Transitions Proceedings,Volume 3, Issue 1,2022,Pages 169-176,ISSN 2666-285X,https://doi.org/10.1016/j.gltp.2022.03.005.

[12] Disha Patel, Hungarian Method Based Resource Scheduling algorithm in Cloud Computing, 2016