EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

Research in Computing Science

Vol. 154 No. 12
December 2025

Research in Computing Science

Research in Computing Science, Año 24, Volumen 154, No. 12, diciembre de 2025, es una publicación mensual, editada por el Instituto Politécnico Nacional, a través del Centro de Investigación en Computación. Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, Ciudad de México, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor responsable: Dr. Grigori Sidorov. Reserva de Derechos al Uso Exclusivo del Título No. 04-2019-082310242100-203. ISSN: en trámite, ambos otorgados por el Instituto Politécnico Nacional de Derecho de Autor. Responsable de la última actualización de este número: el Centro de Investigación en Computación, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Fecha de última modificación 01 de diciembre de 2025.

Research in Computing Science, year 24, Volume 154, No. 12, December 2025, is published monthly by the Center for Computing Research of IPN.

Volume 154(12)

Advances in Artificial Intelligence

Cesar Jesús Núñez-Prado (ed.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2025

ISSN: in process

Formerly ISSNs: 1870-4069, 1665-9899

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition

ISSN 1870-4069

Table of Contents

Page

ISSN 1870-4069

Significance of Algorithmic Approaches in Graph Theoretical Concepts and Searching Techniques

Muhammad Ateeb Ather

Department of Computer Sciences, Bahria University, Lahore, Pakistan

03-134211-022@student.bahria.edu.pk

Abstract. Perhaps no field or subfield of computer sciences is more intriguing and research about than algorithms, their significance and their application throughout different disciplines of sciences. Algorithms are fundamental foundations for many domains including modelling, testing, qualifying and employing in complex systems developed for different purposes. One of the biggest breakthrough in the history of science, Turing Machine also uses this very technique. Other disciplines also use different algorithms vastly to explain, express and implement theories and propositions. One such discipline is graph theory which deals with modeling, representing and expressing different propositions and empirical theories. It comes as no surprise that it also employs the algorithm technology. Having said that, it has not gained the emergence within research.

Keywords: Algorithms, graph theory, graph representation, shortest path problem, graph coloring.

1 Introduction

As the computers becomes more powerful and faster, on the same way the importance of algorithms increases. In computer science applications, development of algorithms plays a vital role. To build a large program with an appropriate time and space consumption, it is important to have well-organized solutions to the problem parts. The most important part of algorithm is that how much it is fast. Computer scientists usually talk about the runtime relative to the size of the input. However, many complex algorithms execution time can differ due to the number of factors other than the size of the input.

Algorithms plays an important role in our lives from various aspects in the field of computer science. Like in flying, making money, searching, weather forecasting, find shortest path to travel from one location to another and much more. Now everyone uses a computer so every time, when we uses the search engine and hit the search button for

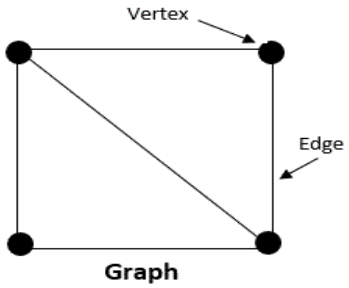


Fig. 1. Representation of Graph.

searching social networks and websites like google, twitter, Facebook etc. it provides result in a fraction of second. This is only possible because of algorithms. They have changed our lives by sorting through the vastness of the internet and giving us relevant, immediate results**.**

As it is known as the computer science field and knowledge is so vast that it comprises of many different sciences, graph theory is also a branch of computer science. Particularly in research domains of computer science, graph theoretical concepts are highly applied by computer science applications like image capturing, clustering and data mining etc. For example, in terms of trees by using vertices and edges data structures could be developed.

Using graph concepts, demonstration of network models can be implemented. Also routes and tracks in graph theoretical concepts are used in numerous applications such as resource networking, designing of database concepts etc. [1].

In computer applications, designing of graph algorithms is an essential part in graph theory. In the form of graph numerous algorithms have been designed to solve problems. These algorithms are being used in solving graph concepts and hence in solving the corresponding computer science application problems.

Searching algorithms(DFS, BFS) Shortest path algo's, Algorithms used to find cycles (rotations) in a graph, Algo's to find adjacency matrices and so on.

In 1736, Graph theory originated from Euler "Konigsberg bridge" problem. Euler's key insight was that by using simple mathematical structure called *graph*, the islands and bridges could be demonstrated. In CS (computer science) and mathematics, graph theory is expressed in terms of the study of graphs. To model pairwise associations between objects (entities), graphs are used as a mathematical structures.

Most of the people use the word "graph" roughly, you can't give any grantee what they are talking about and what type of graph. Because there are different types of graphs use in mathematics, computer science or any other field, with its own description. Section 2 describes the basic concepts of graphs and its types. Section 3 emphasis on the history of graphs. Section 4 focuses on the demonstration of graphs using matrices. Segment 5 describes basic algorithms of graphs.

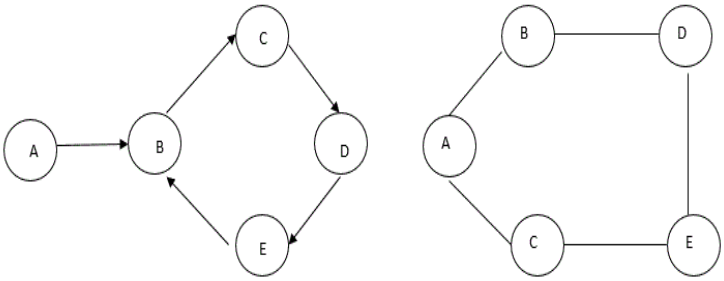


Fig. 2. Directed vs Un-Directed Graph.

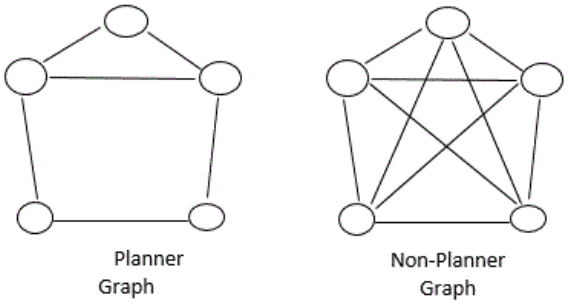


Fig. 3. Planner vs Non-Planner Graph.

2 Graph

According to authors, "Graph is a representative demonstration of a network and its connectivity" [2].

A special diagram called *graph* provides an efficient way of moving among different destinations in a routing problem.

Informally, graph is a map comprising of points called vertices, linked together by lines called edges.

In the above figure Black dots show destinations, while the lines used to join destination points called edge.

Graph is a set of (V, E), components of V are vertices (dots, nodes) and components of E are its edges (arcs, lines). Vertex: V is a joining or ending point. Edge: E is an association between nodes [3].

To analyze a graph it is important to study about degree of a graph. To find the degree of a graph, figure out all of the vertex degrees. The degree of the graph would be its largest vertex degree.

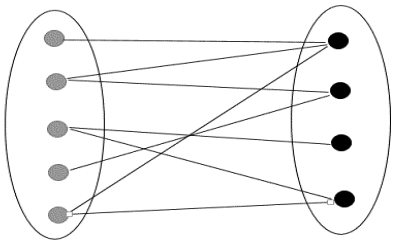


Fig. 4. Bipartite Graph.

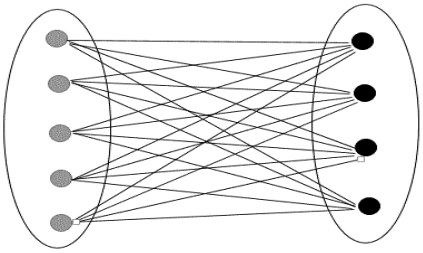
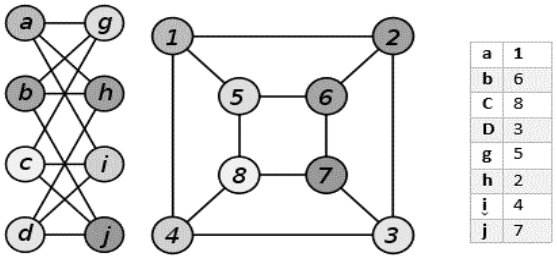


Fig. 5. Complete Bipartite Graph.



a	1
b	6
C	8
D	3
g	5
h	2
i	4
j	7

Fig. 6. Isomorphic Graph.

2.1 Types of Graphs

There are a number of graphs depending on the no of edges, no of vertices, interconnectivity and their overall structure. Here we will describe few important types of graphs.

– Directed and Un-Directed graph: A graph, each edge associates with other vertex must have a direction called DG. A graph contains edges and vertices but edges contains no direction called Un-DG.

– Planner and Non-planner Graphs: a graph will be planner if no link is overlapping with another. Otherwise it will be non-planner graph.

– Simple Graph: a graph does not contains loops and multiple edges.

- Complete Graph: Type of a simple graph in which every couple of dots joined by a link (edge).
- Regular Graph: a type of graph in which all vertices must have the same degree.
- Cyclic Graph: a graph comprises of minimum one cycle. At least one cycle is compulsory.
- Acyclic Graph: graph without cycles. A connected acyclic graph or diagram is known as *tree*, and disconnected acyclic map called *forest*.
- Cycle Graph: circular graph consists of only one cycle.
- Subgraph: graph contains some of the vertices and edges of another graph.
- Bouquet Graph: a graph comprises only one vertex.
- Bipartite Graph: special type of diagram in which set of nodes could be partitioned into two cells (v1, v2) or disjoint sets (two sets are said to be disjoint if have no common element) such that all edges go only between v1 and v2. Edges cannot move from v1 to v1 and from v2 to v2.
- Complete Bipartite Graph: is a special kind of graph, each vertex of one set must be joined to each vertex of another set of the graph.
- Isomorphic graphs: Two graphs that contain same number of edges & same number of vertices joined in the same way called isomorphic.

3 History

In recent past, researchers have focused towards investigating the potential of analyzing graph theory concepts. As discussed earlier graph theory came into existence with the problem of Konigsberg Bridge in 1736. Euler provide a solution of this problem by constructed a structure called Eulerian Graph, so Euler became the father of graph theory. A.F Mobius, introduce concept of bipartite & complete graph in 1840. In 1845, Gustav Kirchhoff implemented the idea of tree (connected graph does not contain cycles [4]) and developed graph theoretical concepts. Thomas found the 4 color problem in 1852. For more than a century, 4 color problem remained unsolved. Many invalid solutions were suggested. Thomas.P.kirkman & William R.Hamilton invented the concept of Hamiltonian graph in 1856. Sylvester introduced the term “graph” in 1878 [5]. Puzzle problem was first introduced in 1913 by H.Dudeney. Another section of graph theory called “Extremal Graph Theory”, were originated in 1941 by Ramsey. After a century, Heinrich provide a solution for 4 color problem by using computers in 1969. This conception of graph coloring is highly applied in the domains like resource allocation and scheduling.

In graph theory various concepts like paths, walks and circuits are used in various applications like TSP, networking etc. This leads to the advancements of new processes, new algorithms, approaches & procedures which are being used in enormous fields or applications [6].

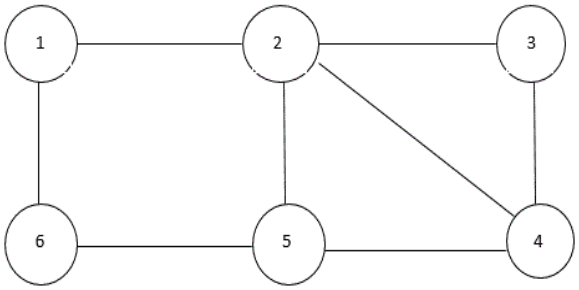


Fig. 7. Representation of Un-Directed Graph.

Node	Neighbors
1	2,6
2	1,3,4,5
3	2,4
4	2,3,5
5	2,4,6
6	1,5

Fig. 8. Adjacency List.

4 Graph Representation

There are different kinds of representations, which can be very beneficial in some situations and inoperable in others. Here we will see three methods to illustrate the graphs.

– Adjacency list.
– Adjacency matrix.
– Edge list.

Graph G= (V, E), might be either undirected or directed. Running time of an algorithm, could be demonstrated in the form of |E| & |V|. We will show the cardinality O (V+E), in asymptotic representation.

4.1 Edge List

One simple way to express a graph is just a list or array of |E| edges, which we call an edge list. To demonstrate an edge, we just have an array of two vertex numbers, or an array of objects comprising the vertex number of vertices that the edges are incident on. In an edge queue L, every edge e is kept as a tuple (vi, vj) in an arranged queue. The edge list of the graph described above is:

$$L = \{ (1,2) , (1,6) , (2,3) , (2,4) , (2,5) , (3,4) , (4,5) , (5,6) \}. \quad (1)$$

0	**1**	**0**	**0**	**0**	**1**
1	0	1	1	1	0
0	1	0	1	0	0
0	1	1	0	1	0
0	1	0	1	0	1
1	0	0	0	1	0

Fig. 9. Adjacency Matrix.

However being a space proficient technique to store the graph, an edge list access time is slower than the adjacency matrix access time (for most edges).

4.2 Adjacency List

It is related to an edge list but difference is, array is associated with a queue. Mass of an array is equal to the no of nodes. An entrance array (group) shows the linked lists of vertices neighboring to the i^{th} node. This demonstration could be used to express a weighted graph. In the nodes of linked list, the edge weights can be stored. Adjacency list demonstration of the graph is.

4.3 Adjacency Matrix

Matrix A, the AM of a graph is V x V, if there exists an edge, each entry is equal to 1 otherwise 0. Note that, graph we are using in the example is undirected, the resulting matrix is symmetric. Also no self-loops exists, each entry in the main diagonal is 0.

5 Graph Algorithms

There are number of algorithms that could be applied to graphs. Many of them are actually applied in real world like shortest path, topological sort, Dijkstra algorithm, DFS, BFS, graph coloring algorithms etc. Here, we will discuss few of them.

5.1 Breadth First Search (BFS)

BFS can be used to discover shortest path from maze problem according to E.F Moore [7]. BFS was first introduced in the late 1950's and exposed independently by C.Y.Lee as a wire routing problem. BFS is a general algorithm for searching or traversing a graph. Before moving to the next level neighbors, BFS discovers the starting node and its neighbor's node first. BFS search start working on the vertex which will be on level zero. There are two phase: In the 1^{st} step, go to all vertices which will be at the distance of one edge away. Go there, paint "visited" the vertices we have been visited adjacent to the starting node, and kept these vertices into the level one. In 2^{nd} step phase, visit

all new vertices that are at the space (distance) of 2 edges away from the starting node. All new vertices that do not previously assigned to a level kept into level two & so on, which are adjacent to level one. When every vertex has been visited, BFS traversal works end.

For traversing graph nodes, this is a very different methodology. The aim of BFS algorithm is to traverse the graph to the root node & to implement this, Queue is used. Now see, how BFS traversal done on the following graph.

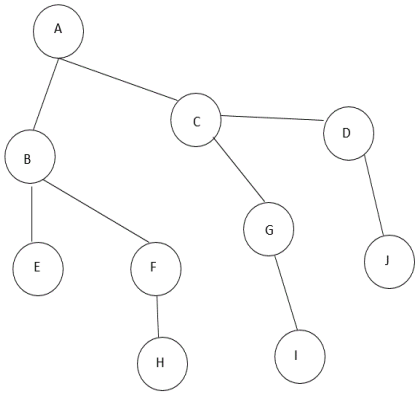


Fig. 10. Simple Graph with 10 vertices.

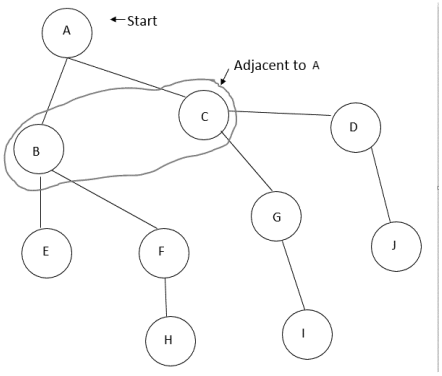


Fig. 11. First BFS Layer.

Pseudocode

BFS (G, s)

1. Q = queue comprising only s
2. while Q not empty
3. v = Q.front(); Q.remove front()
4. for w ∈ G.neighbors(v):
5. if w not seen:
6. mark w seen
7. Q.enqueue(w)

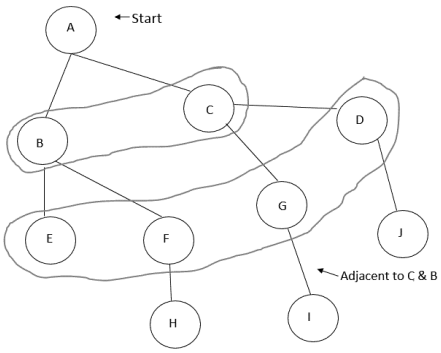


Fig. 12. Second BFS Layer.

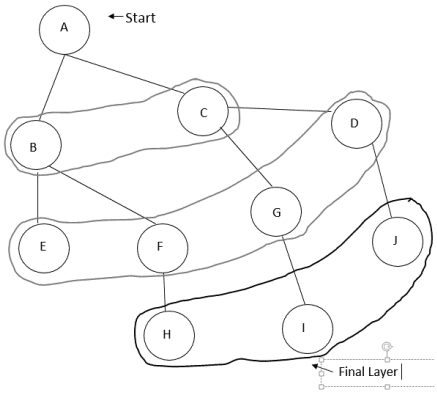


Fig. 13. Final BFS Layer.

For some explanation, this is an undirected graph. So you cannot see arrows. E. g., point from vertex F to vertex H, then the edge b/w vertex F to H can be traversed in either direction. Another point is that a vertex is said to be adjacent to another vertex if an edge joins the 2 vertices directly. When we apply BFS, we can only go from one adjacent vertex to another. You cannot move directly from vertex C to I.

In this algo, we will start from source vertex which is the root of the graph, put the source vertex into the queue. After visiting the root vertex, go to its neighbor nodes, if not visited yet. Queue contains all vertices that have been seen but not yet visited.

Now start BFS if we are at level 1. Put A into queue, unvisited nodes are H, I, G, J, D, F, E, C, B & visited node is A. Dqueue node (A). Now vertex (A) has 2 unvisited vertices C & B.

Now BFS algorithm looks at vertices B & C and mark as visited, so they have been taken placed in queue after vertex A listed its adjacent neighbors. At this stage only vertices A, B & C count as vertices which have been visited. Vertex A has no more unvisited children, Dqueue B. So insert vertices E & F into the queue, also put vertices G & D in the queue. Figure 12 depicts the next layer of BFS with gray wall.

Pseudocode

```
DFS(G, v){
1. mark v visited
2. set color of v to gray
3. for each successor v' of v {
4. if v' not yet visited {
a. DFS(v')}}
5. set color of v to black }
```

Stack Based DFS Pseudocode

```
DFS(G, s):
a. ST = stack containing only s
6. while ST not empty
a. v = ST.pop()
b. if v not visited
c. mark v visited
7. for w ∈ G.neighbors(v): S.push(w)
```

At this level, node E has no more unvisited vertices, Dqueue F, enqueue H into queue. Now we will Dqueue G & D and enqueue their vertices into queue, it includes I & J.

5.1.1 Complexity of BFS

Complexity of BFS could be explained in O (V+E) since each edge & node will be discovered in the worst case. Remind that O (E) may differ b/w O (1) & O (V^2) depending on how scarce the input graph is. If the graph is expressed by an adjacency list it obtains O (V+E) space in memory, while an adjacency matrix depiction occupies O (V^2) [8].

5.1.2 Applications of BFS

In graph theory, BFS could be used to resolve many problems, such as:

- Testing a graph for bipartiteness.
- Copying garbage collection, Cheney's algo.
- In a flow network, compute maximum flow- Ford Fulkerson method
- Finding shortest path b/w 2 nodes etc [9].

BFS can also be used to test bipartiteness, starting search from at any node, during the traverse give differ labels to the vertices visit. Starting node give the name 0, 1 to all its close or neighbors, 0 to those neighbors' neighbors and so on. Graph will not be bipartite, if a node has neighbors with the same label as itself at any phase. If traverse terminates and no situation occur as we said earlier then graph would be bipartite.

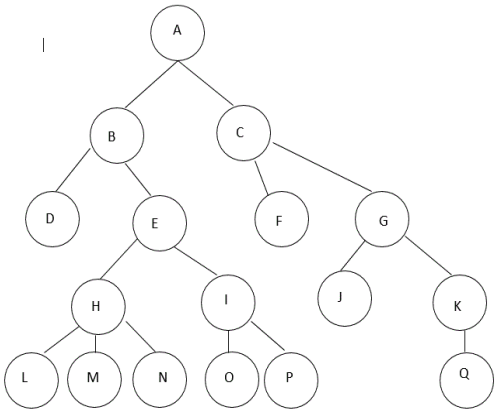


Fig. 14. 1st Depth First Search.

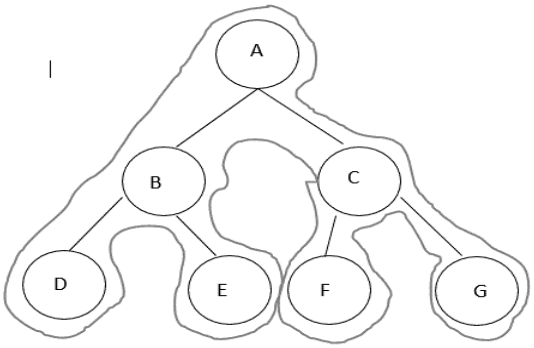


Fig. 15. Depth First Search (DFS).

5.2 Depth First Search (DFS)

French mathematician Charles Pierre Tremaux was discovered DFS in the 19th century as a policy for solving mazes [10-11].

DFS is another approach for traversing a graph. There are a number of algorithms which based on DFS, DFS allows visiting vertices of the graph only. Therefore, to move into the graph theory it's important to learn about the concepts of DFS. Algorithm is quite easy, move until there is a chance to go forward, otherwise go back.

In DFS, every vertex has 3 possible colors, such as:

Gray-vertex is in evolution

White- vertex is unvisited

Black- DFS has completed the processing

DFS can be finished with the help of Stack, LIFO execution. We start from node A, put into Stack, visited vertex is A. peek at the stack, vertex A has unvisited vertices B & C. Mark B as visited. At the top of the stack node B has D & E unvisited nodes. D marked as visited. Top of the stack D has no more unvisited nodes. Pop D from stack. Go to vertex E. visit E, unvisited vertices are E & I. Go to H visit its neighboring vertices, move on vertex I, visit it. I has unvisited vertices O & P. visit them. Continue this process when no vertices will be left unvisited.

Output generate will be in the form of,

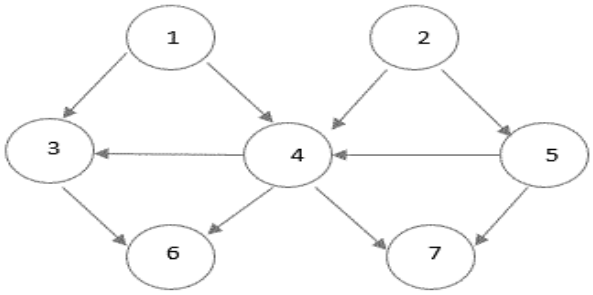


Fig. 16. Topological Sort.

A, B, D, E, H, L, M, N, I, O, P, C, F, G, J, K & Q

5.2.1 DFS Complexity

DFS complexity is O(V+E). For expressing the graph, if adjacency matrix is being used, than all edges adjacent to the vertex cannot find efficiently, the result will be in the form O (V^2) [8].

Another example:

Output: A B D E C F & G.

5.2.2 Ordering of Vertex

To generate linearly order of the vertices of the original graph, it's possible for us to use DFS. There are 3 collective methods we use to do this:

A pre-ordering generates order of vertices in the way that DFS algorithm first visited. To express the growth of the search, it's a natural & compressed technique. Preorder of the above example will be same of the output.

A post-ordering type of list of vertices in the way that they were last go through or visit by the algo. Post-order is D, E, B, F, G, C & A.

A reverse post-ordering generates list of vertices in an opposite order that they were visit last. Order is D, B, E, A, F, C & G. Parent node appear in the middle.

5.3 Topological Sort

In the early 1960's, topological sorting algorithms were 1st studied, (Jarnagin 1960) in the perspective of PERT approach for preparation in Project Management. In the

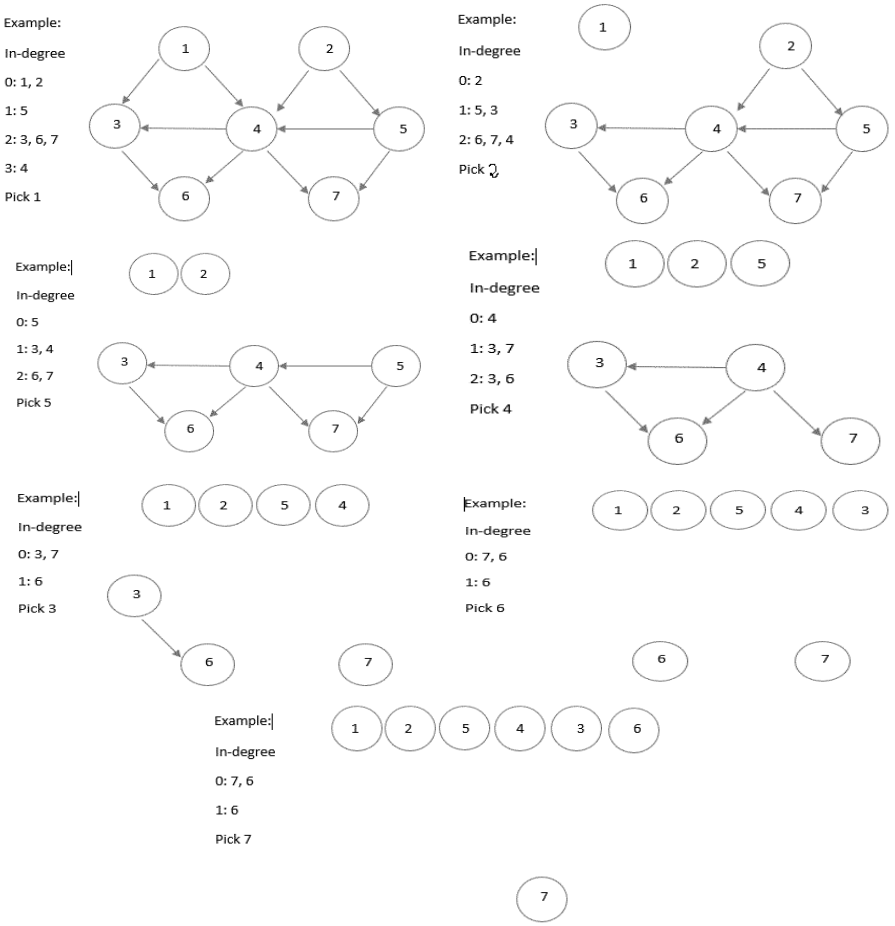


Fig. 17. Topo-Sort Example.

domain of computer science, sometimes topological sort of directed graph, called topological ordering or topo-sort, in which DG linear arrangement of the nodes is, parent must come before child, or we can say that from vertex u to vertex v, u must be come first before v. If graph contains no directed cycles, topological ordering will be possible only in this condition. To construct topological ordering of any DAG, algorithms are known will be in linear time for the development of topo-sorting. Through weighted directed acyclic graph to calculate shortest paths, topological ordering could be used. Using a serial algorithm, on a graph of m no of edges & n no of vertices, algo occupies linear time O (n+m) [8]. There are often many possible topological sorts of a DAG. For example

Topological order for this DAG will be in the form of:

– 1, 2, 5, 4, 3, 6, 7

– 2, 1, 5, 4, 7, 3, 6
– 2, 5, 1, 4, 7, 3, 6 etc.

Firstly make in-degree array, list all vertices with degrees. After that, vertices which value is in-degree 0 put into the queue. Now that vertex will be dequeue and insert into the output. In-degree array will be updated after visiting all nodes with 0 in-degree.

5.3.1 Topo-Sort Complexity

Pseudocode

```
1.  L - Initialize ordered list to be empty
2.  S - all vertices having no incoming edge's
3.  While (S ≠ ∅)
4.        Eliminate vertex v from set S
5.        add v to the end of the list L
6.  For each (vertex v1 with edge e from v to v1) do
7.        E, eliminate from graph
8.  If (v1 has no more incoming edges) then
9.         In set S add v1
10. If (graph contains edges) then, return error
11. Else return L ( sorted order)
```

In DAG, if all sets of nodes in an organized and sorted way joined by edges, then all these edges create a (DHP) directed Hamiltonian path, it's a topo-sort property. Topological sort order will be unique in the form if Hamiltonian path occurs. The usual algorithms for topo-sort having linear running time in the form of vertices + edges (links), in an asymptotic notation O (|V| + |E|).

5.4 Kahn's Algorithm

A type of algorithm, works by picking vertices in the same order just like the eventual topo-sort, 1st pronounced by Kahn (1962). Identify list of vertices without no incoming edges & insert them into the set. Solution contained in the list, if graph is DAG & if there exist at least one cycle, topo-sorting will be impossible.

5.5 Tarjan's Algorithm

A type of algorithm which we may use as an alternative for topological ordering based on DFS, 1st described by Tarjan (1976). Algorithm executes in linear time & every node & edge is visited once, in this algorithm.

6 Shortest Path Problem

To search a path, in the graph G, b/w two nodes or vertices, SPP is the problem, in which edge values sum will be minimized. SPP can be described for graphs whether

Pseudocode

```
1.  Dis [sor] ← 0
2.  For all ve ∈ V-{sor}
3.  For all ve
4.  do Dis [ve]←∞
5.  S←∅
6.  Qu←V
7.  while ( Qu ≠ ∅ )  do
8.  u ← mini-dis (Qu, Dis)
9.  S ← S u {u}
10. For all (v ∈ neighbors[u]) do
11. if Dis[ve] > Dis[u]+we(u,ve)
12. then Dis[ve]← Dis[u] + we(u,ve)
13. return Dis
```

weighted or unweighted. In weighted graphs, find a lowest cost path is basically the main objective. Un-weighted graphs, goal is to identify a path with smallest number of hops. The problem sometimes called single pair SPP, to differentiate it for the following variations: The single source SPP, for weighted graph find minimum weighted path from starting node to the end of the graph. Here we use some algorithms which based on the type of the graph, if graph is weighted we use Dijkstra algo otherwise use simple Breadth First Search for an unweighted graph. The single destination SPP, identify shortest pathways from every node to a specific ending node, in a weighted graph. In all pairs SPP, we need to find smallest routs b/w each group of nodes.

Early History of Shortest Path Algorithms

- 1955-Shimbel, Information Networks
- 1956-Ford, worked on economics of transportation (RAND)
- 1957-Johnson, Ladew, Seitz, Gray, Meaker, Leyzorek, Petry, Combat development department of Army Electronic Proving Ground,
- 1958-Bellman, Dynamic programming, Simplex technique for linear programming-Dantzig
- 1959-Dijkstra, fast & simple form of Ford's algorithm & Moore worked for Bell Labs

Applications of Shortest Path Algorithms

It's a beneficial model to solve problem. To find directions b/w physical locations, such as Google Maps, we use SP Algorithms. For this application fast specialized algorithms exist [12]. Other applications:

- Texture mapping.
- Routing of telecommunications messages.
- Subroutine in advanced algorithms.
- Optimal pipelining of VLSI chip [13].

– Urban traffic planning.

– Network routing protocols (OSPF, BGP, RIP).

– Robot navigation.

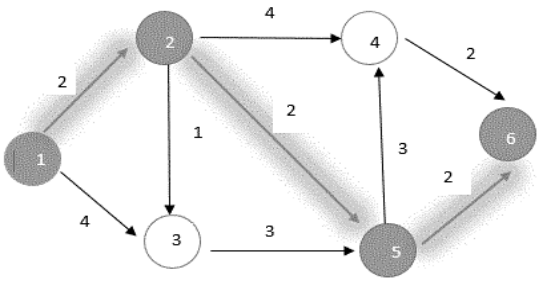


Fig. 18. Dijkstra Algorithm

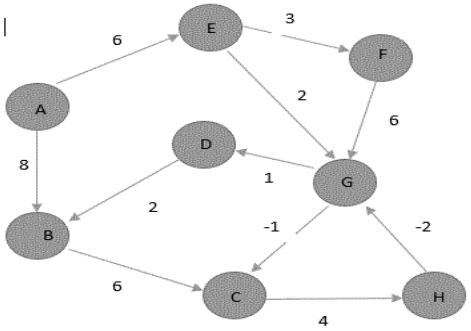


Fig. 19. Directed Graph for Bellman-Ford.

– Approximating piecewise linear functions.

6.1 Dijkstra Algorithm

In a graph, to search route from starting node to the destination node we use DA, with undirected edges & non-negative edge values. Connected graph is compulsory. It uses a greedy approach to find shortest path weights in the graph.

In line 1, distance to source vertex is 0. In line 3, declare all distances to ∞. Line 4, S, visited vertices set is initially {}. Line 5, queue initially comprises of all nodes. Line 6, while queue is ≠ {}. Line 7, choose variable from queue having min-distance. Line 8, in the list of visited nodes add u. Line 10, if new shortest track found, line 11- update value of shortest path.

Pseudocode

```
1. ds=0
2. dv = ∞ ∀ (v ≠ s)
3. j=1;
4. For  j to m-1
5. ∀ edge u → v of cost c
6. dv = minimum (dv, du + c)
   ∀ edge u → v of cost c
7. if (dv > du + c)
   return false
8. return true
```

In the following graph, if we select path {1, 2, 4 & 6} then its total cost will be 8. Now, go from another path {1, 3, 5 & 6} its total cost is equal to 9. If we go from {1, 2, 3, 5 & 6} its cost is equal to 8. Another path {1, 2, 3, 5, 4 & 6} cost is equal to 11. Another possibility {1, 2, 5, 4 & 6} cost will be 9. If we go from {1, 2, 5 & 6} its cost will be 6. This is the lowest cost from all costs, so Dijkstra algorithm choses way with lowest cost values and uses greedy approach to reach at the destination vertex.

6.2 Bellman-Ford Algorithm

In a weighted digraph, in which we start searching to identify lowest value route from a single start node to all of the other vertices of the graph [19]. Dijkstra algorithm is faster from BFA. But BFA is beneficial to control graphs with negative weights. There can be situations, where a graph can be in negative weight cycles, but we do not see these situations in a real life. But now there are some complex situations with negative edges. Few of them are detecting network failures or linear programming.

6.2.1 Bellman Ford Algorithm Description

Maintain list of unvisited nodes

Select source node & allocate maximum cost infinity to every other node

Cost of the source node remains 0 as it actually takes nothing to reach from the start node to itself

In every iteration of the algo it tries to minimize the cost of vertex

Repeat step 4 for |v|-1 times. From the last iteration we will have a shortest path from start to every vertex.

1st step

Suppose A is a starting node with cost 0, insert all vertices (A, E, F, D, G, B, C & H) in the list. Allocate cost infinity all vertices except starting vertex A.

2nd Step

– (A, E): cost of E [6].

– (A, B): cost of B [8].

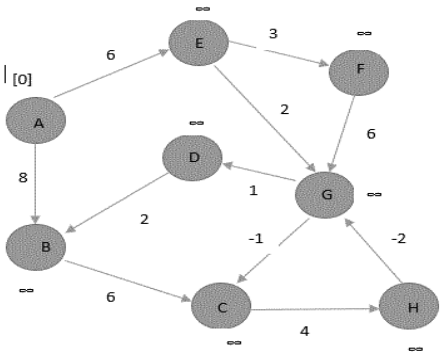


Fig. 20. 1st Step BFA.

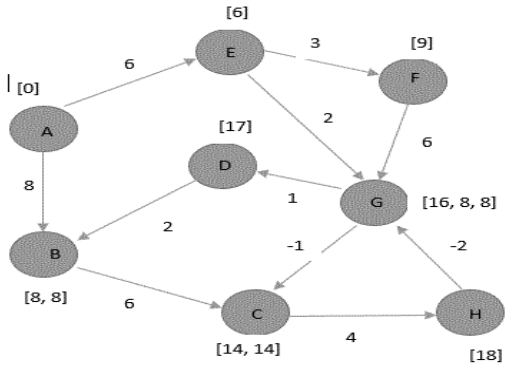


Fig. 21. 2nd Step BFA.

– (B, C): cost of C [14].
– (C, H): cost of H [18].
– (H, G): cost of G [16].
– (G, C): cost of C [14].
– (G, D): cost of D [17].
– (D, B): cost of B [8].
– (E, F): cost of F [9].
– (E, G): cost of G [8].
– (F, G): cost of G [8].

3rd Step

– (A, E): cost of E [6].

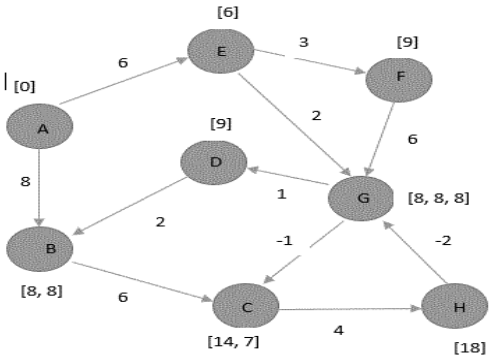


Fig. 22. 3rd Step BFA.

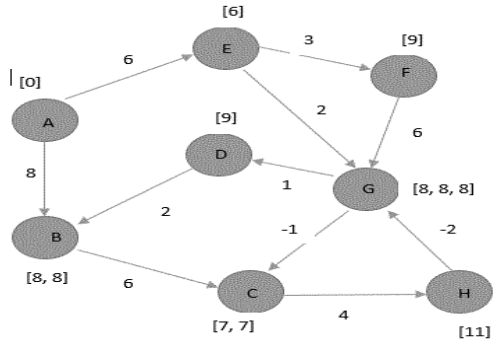


Fig. 23. 4th Step BFA.

- (A, B): cost of B [8].
- (B, C): cost of C [14].
- (C, H): cost of H [18].
- (H, G): cost of G [8].
- (G, C): cost of C [7].
- (G, D): cost of D [9].
- (D, B): cost of B [8].
- (E, F): cost of F [9].
- (E, G): cost of G [8].
- (F, G): cost of G [8].

4th Step

- (A, E): cost of E [6].

Pseudocode

```
1. for (m=1 to v, m++) do
2. for (n=1 to v, n++) do
3. Dis(m, n)= wei(m, n)
4. for (o=1 to v) { //o is an intermediate vertex
5. for (m=1 to v) do {
6. for (n=1 to v) do {
7. If (dist(m,o) + dist (o,n) < dist pass n,m )
8. Then dist pass n,m = dist pass m,o + dist pass o,n }}}
```

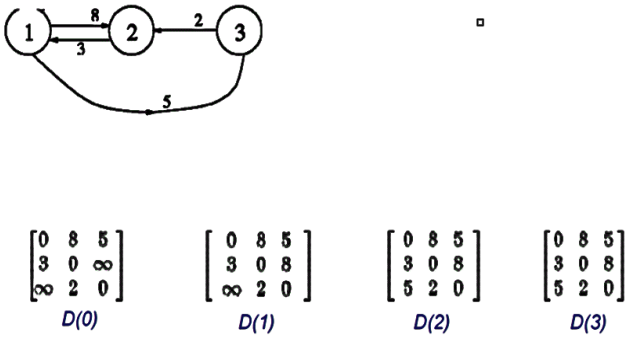


Fig. 24. Floyed Warshall Algorithm.

- (A, B): cost of B [8].
- (B, C): cost of C [7].
- (C, H): cost of H [11].
- (H, G): cost of G [8].
- (G, C): cost of C [7].
- (G, D): cost of D [9].
- (D, B): cost of B [8].
- (E, F): cost of F [9].
- (E, G): cost of G [8].
- (F, G): cost of G [8].

Complexity of Bellman ford could be represented in the form of O(VE) time, declaration will take O(V) time in 1st line, in the line 2 to 4 take O(E) time, V-1 passes over the edges & in line 5 to 7 for For loop takes O(E) time.

6.3 Floyd Warshall Algorithm

FWA is an algorithm in which, graph has –ve or +ve edge values but not containing –ve cycles. It finds only the length not the path. FWA is an example of dynamic

Pseudocode

```
JSS (V, E, s)
1.  begin
2.  Q=V
3.  for all (ver ∈ Q) do
4.          l[ver] = ∞
5.                  l[s]=0
6.  Do- while (Q ≠ ∅)
7.  begin:
8.       h= extr-min(Q)
9.  For ver ∈ adja [h] -- do
10. If codition  ver ∈ Q l[h] + we(h,ver) < l[ver]
11. Then l[ver] = l[h] + we(h,ver)  end while -- end of JSS
```

programming. This algorithm also known as Roy-Warshall Algorithm, Floyd's Algorithm, Roy-Floyd Algorithm & WFI Algorithm. FWA running time is O (V^3).

If (j=k), G[j][k] is 0,

If no edge b/w j to k then, G[j][k] is ∞

6.4 Johnson's Algorithm

JA is an algo to search shortest tracks b/w all pair of vertices in an edge-weighted, directed or sparse (a graph with few edges) graph. It permits some edge weights to be negative values, but no –ve cycles. It works by using bellman-ford & Dijkstra algorithm. Time complexity of this algo is O (V^2logV + VE).

7 Graph Coloring

Graph coloring has been using in many real time applications of computer science & it is the most significant part of graph theoretical concepts. Numerous labelling techniques exists & can be applied on the necessity bases. The appropriate graph coloring is the overall coloring of links & nodes (V & E), in a way no two nodes consists of the matching color, and must be used smallest no of colors. The usage of smallest no of colors in the graph called (CN) chromatic number & graph G called (PCG) properly colored graph. From 1972, Chromatic numbering problem we said one of the NP-complete problem. Register allocation in compilers is the major application of graph coloring and was introduced in 1982. Graph coloring has been using in theoretical concepts as well as practical applications, but there are different limitations that can be set on graphs, they may be color assigning or can be color itself. Due to limitations, it is still a vigorous domain of research [18].

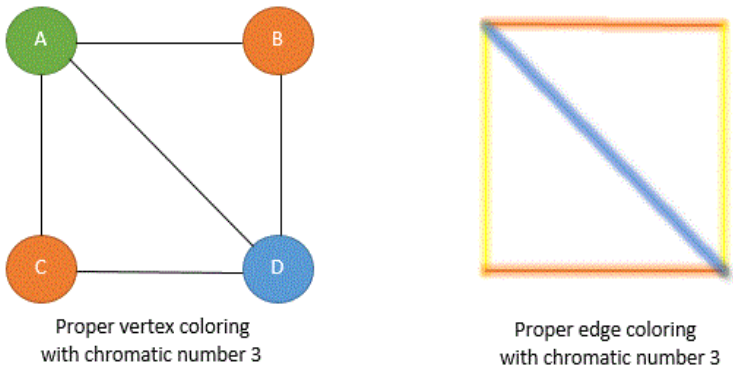


Fig. 25. Graph Coloring.

7.1 Vertex Coloring

Vertex coloring is the most famous problem in graph coloring. The problem is, color each vertex in a way that two adjacent vertices does not consist of the matching color.

The other graph filling complications like Link (Edge) and Map labelling could be converted into node coloring. Graph coloring G is basically a mapping (plotting) function & denoted as c: V (G)--S.

In set S, Variables of S are colors, assign color to the nodes, pick colors from a color class S. C is K-coloring, if |S|=K. if adjacent nodes have diverse colors then graph coloring will be appropriate . G called K-colorable, if G has appropriate K-coloring. For the representation of chromatic number, notation X(G) is used. G will be K-chromatic, If X (G) = K [14].

7.2 Edge Coloring

An EC is an appropriate filling of the edges, which means no two edges consists of the same color, when we allocate colors to edges. Edge CN or CI (chromatic index) X' (G), is lowest no of colors needed for EC. For (3-EC) of a cubic graph Tait coloring is basically used.

7.3 Total Coloring

No neighbor edges, no neighbor vertices, no edges & vertices allocated the same color, in this way total coloring will always be supposed to be proper. A kind of coloring applied on nodes of graph & links. X^n (G) notation represents the Total Chromatic Number (TCN) of a graph.

7.4 Chromatic Number

Pseudocode

```
1: function Main
2: D = ∅
3: for all v ∈ V do
4: D ∪ v. out degree
5: end for
6: while (D ≠ ∅) do
7: d ← max {d | d ∈ D}
8: schedule(vertices of degree(d))
9: D − {d}
10: end while
11: execute scheduled()
12: end function
14: return ∀v ∈ V : v → v. color
```

Finding a chromatic number is an NP-complete problem. Since we do not have any idea, how many colors will be used in the graph? For graph coloring, minimum no of colors represent Chromatic number.

8 Graph Coloring Algorithms

The problem of sequentially coloring an arbitrary graph has been studied widely. 4-coloring problem exist for planner graphs, but non-planner graphs may require large no of colors. To color a graph by using minimum number of colors, looks a very simple problem, but in reality it's not a simple one. To solve this problem, no of serially based polynomial time algorithms exists, which basically use limited no of colors. Here we present some sequential GC algorithms like LDF algorithm and smallest degree last algorithm and SDO and IDO algorithms but SDO and IDO are not appropriate to parallelization. [16]

For graph labeling, there exists a lot of heuristic techniques & one is Greedy Graph Coloring [17]. This type of approach emphasis on choosing the next vertex for coloring. In this heuristic based technique, once a node is filled with a color, then color cannot be changed. Here we present greedy algorithm for graph coloring, we cannot sure it will use minimum number of colors or not. In the graph, extreme degree of a node is d, so basic greedy coloring algorithm cannot uses more than d+1 colors.

– Assign color to the 1st vertex in the graph.
– Do step 3 when V-1 nodes remained.
– Suppose, when we choose a node and assign color, make sure that color has not been used previously labeled nodes, adjacent to that vertex. If any condition exist, then allocate new color.

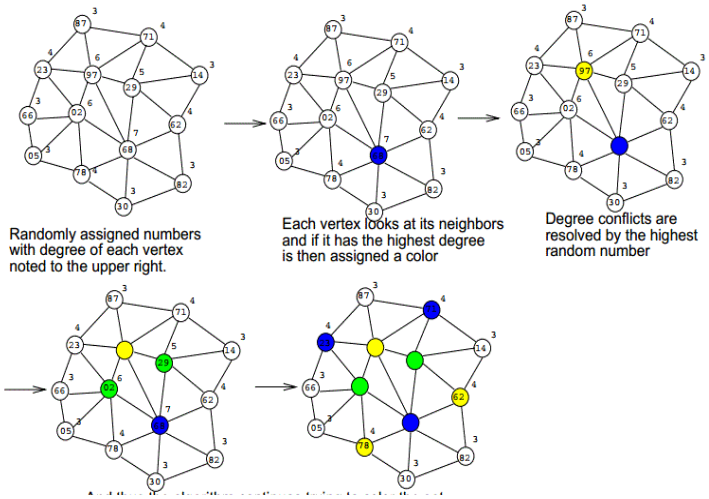


Fig. 26. Largest Degree First.

Pseudocode

1. L = 1
2. k = 1
3. U = V
4. while ([U] > 0) do
5. while { vertices v ∈ U with dU (v) <= L } do in parallel
6. S = { all vertices v with dU (v) <= L }
7. for all vertices v ∈ S , w (v) = k
8. U = U S
9. k = k + 1 end do
10. L = L + 1 end do

8.1 First Fit (FF)

FF algorithm is the fastest & easiest approach. In this algorithm, algorithm allocates each vertex smallest legal color in sequence. Complexity of this algo is expressed in linear time O (n) [14].

8.2 Degree Based Ordering (DBO)

DBO offers an enhanced tactic for graph coloring. For selecting the node to be labelled, it uses a specific selection norm. It is better than first fit because in which from random direction, we merely pick a node. For choosing a next node to be colored, few techniques have been suggested like:

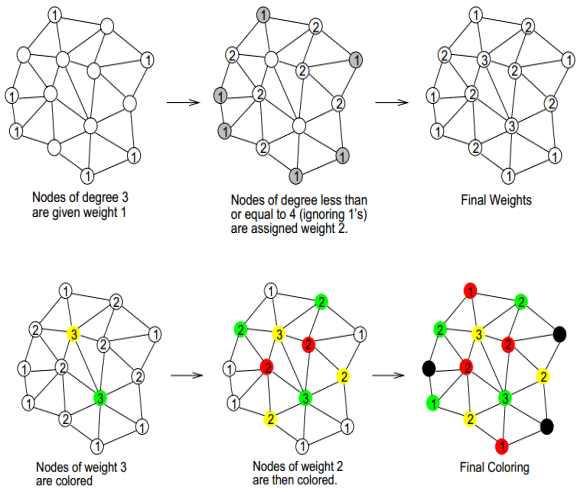


Fig. 27. Smallest Degree Last.

8.2.1 Largest Degree Ordering (LDO)

In this algorithm, vertices will never be colored in random order but vertices with the largest degree being the colored first. This heuristic offers better coloring then first fit and can be implemented in the notion O (n^2) [14].

In this example, we color those vertices having largest degree, we will never randomly choose the node. After selecting node, color it [15].

8.2.2 Smallest Degree Last Algorithm

This algorithm works in two steps, a weighting step and coloring step. The weighting step starts by searching small degree node. After assigning the weights, in coloring step, color the largest value node in the graph.

In the above example, in the 1st graph we will find the node having minimum no of edges. So nodes of degree 3 assign weight 1. In the 2nd figure, nodes of degree less than or equal to 4 neglect 1 assigned weight 2. Now we will search the highest value weight, color them [14-15].

8.2.3 Saturation Degree Ordering (SDO)

The SD of a node demonstrated by means of the no of its neighbors inversely shaded nodes. SDO heuristic offers superior shading then Largest Degree Ordering and could be denoted in the notion of O (n^3) [15].

8.2.4 Incident Degree Ordering (IDO)

IDO is the reformed form of SDO heuristic. Incident degree of a node demonstrated as like the no of its neighbor's painted nodes. Heuristic denoted in the notion of O (n^2) [15].

Pseudocode

1. while (no of shaded vertices < m) {
2. maximum = -1, I= 1
3. loop I to m {
4. if condition !colored(n_i) {
5. d equal to Degree(n_i)
6. if condition d greater than maximum{
7. maximum equal to d
8. ind equal to i}
9. if condition d equal to maximum
10. if condition ID(n_i) greater than ID(n_{index})
11. ind equal to i}
12. Color (n_{index})
13. no of colored nodes= no of colored nodes + 1 } }

Pseudocode

1. while (no of colored nodes < m) {
2. Maximum equal to -1, I equal to 1;
3. Loop I to m {
4. If condition !colored(n_i) {
5. d equal to SD(n_i)
6. if condition d greater than maximum {
7. maximum equal to d
8. ind equal to I }
9. if condition d equal to maximum
10. if condition Degree(n_i) greater than Degree(n_{index})
11. ind equal to I }
12. Color (n_{index})
13. No of colored nodes = no of colored nodes + 1}} [14]

8.2.5 New Heuristic Coloring Algorithms

In the first algorithm, (modification of LDO), we improved the LDO algorithm by merging it with an IDO algorithm. The algorithm operates like LDO, IDO was used to select nodes having same number of degree. For picking next nodes to be colored, there are two tactics.

– Number of nodes linked to the node Largest Degree Ordering.

– Number of colored nodes joined to the node Incident Degree Ordering.

In the second algorithm, (modification of SDO), we merge the both SDO algorithm and LDO. The algorithm functions like as SDO, when we initiate if there are 2 nodes having

same degree, then LDO we used to select nodes b/w them with the same degree. We applied 2 tactics for picking next nodes to be filled with colors.

– No of colors close to the node SDO.
– No of vertices nearby the node LDO.

8.2.6 Applications of Graph Coloring

The following applications of graph coloring can be mentioned.

– Making schedule: Problem is graph coloring in the form, assume, we have a list of students and subjects and we want to make an exam schedule for university. Many subjects would have common students. How we will manage that no two exams of the same student subject are scheduled at the same time? This problem we can model with the help of graph in which each subject will be the vertex and an edge b/w them we say there exists a common student.
– Bipartite Graphs: With the help of using colors we can model graph is bipartite or not. Graph will be bipartite if it is two-colorable [18].
– Sudoku: It is also a variation of graph coloring problem where each cell is denoted by a vertex. There is an edge b/w 2 vertices if they are in same column or same block or same row.
– Register allocation: We have already said, register allocation in compilers is a graph coloring problem. Problem is, it is a method of allocating huge number of objective program elements onto a minor number of Central Processing Unit records (registers).
– Map coloring: Geographical map of cities, states and countries where 2 adjacent cities cannot be represented with the same colors.

9 Conclusion

Algorithms are the foundation of modern sciences and empirical theories. Researches made so far significantly improve and insist upon further areas to be examined and dissected within algorithms, graph theory and their nexus.

The applications of algorithms go way beyond what we have though and researched about. Graph theory also plays a crucial part in expressing, authenticating and implementing these sciences and theories.

After analyzing the formalization of both disciplines in details and consulting researches performed on both the domains, it has been concluded that a combination of these two can open new gates to the world of sciences in terms of modelling and computational devices, communications, data organization and searching techniques.

References

1. Abdul, M., Ibtisam, R.: Graph Theory: A Comprehensive Survey about Graph Theory Applications in Computer Science and Social Networks. Inventions, 5(10) (2020). doi: 10.3390/inventions5010010.
2. Rodrigue, J.P.: The Geography of Transport Systems. Routledge (2020)
3. Gross, J.L., Yellen, J., Anderson, M.: Graph Theory and its Applications. Chapman and Hall/CRC (2018)
4. Deo, N.: Graph Theory with Applications to Engineering and Computer Science. Courier Dover Publications (2017)
5. Fernández, F.M., Castro, E.A.: Algebraic Methods in Quantum Chemistry and Physics. CRC Press (2020)
6. Mondal, B., De, K.: An Overview Applications of Graph Theory in Real Field. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2(5), pp. 751–759 (2017)
7. Tran, N.: Review of the Algorithm Design Manual. ACM SIGACT News, 53(3), pp. 21–23 (2022). doi: 10.1145/3561064.3561068.
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press (2022)
9. Bhargava, V.R., Assadi, P.: Hiring, Algorithms, and Choice: Why Interviews Still Matter. Business Ethics Quarterly, 34(2), pp. 201–230 (2024). doi: 10.1017/beq.2022.41.
10. Even, S.: Graph Algorithms. Cambridge University Press (2011). doi: 10.1017/CBO9781139015165.
11. Goodrich, M.T., Tamassia, R., Mount, D.M.: Data Structures and Algorithms in C++. John Wiley & Sons (2011)
12. Sanders, P., Schultes, D.: Engineering Fast Route Planning Algorithms. In: International Workshop on Experimental and Efficient Algorithms, pp. 23–36 (2007). doi: 10.1007/978-3-540-72845-0_2.
13. Chen, D.Z.: Developing Algorithms and Software for Geometric Path Planning Problems. ACM Computing Surveys (CSUR), 28(4), pp. 18 (1996)
14. Baghel, M., Agrawal, S., Silakari, S.: Recent Trends and Developments in Graph Coloring. In: Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), pp. 431–439 (2013). doi: 10.1007/978-3-642-35314-7_49
15. Allwright, J.R., Bordawekar, R., Coddington, P.D., Dincer, K., Martin, C.L.: A Comparison of Parallel Graph Coloring Algorithms. SCCS-666, pp. 1–19 (1995)
16. Husfeldt, T.: Graph Colouring Algorithms. arXiv Preprint (2015). doi: 10.1017/CBO9781139519793.016.
17. Shukla, A.N., Bharti, V., Garg, M.L.: A Greedy Technique Based Improved Approach to Solve Graph Colouring Problem. EAI Endorsed Transactions on Scalable Information Systems, 8(31) (2021). doi: 10.4108/eai.16-2-2021.168716.
18. Solomon, S., Wein, N.: Improved Dynamic Graph Coloring. ACM Transactions on Algorithms (TALG), 16(3), pp. 1–24 (2020). doi: 10.1145/3392724.

ISSN 1870-4069

Sistema de clasificación de salud de hojas de plantas (SiCHoP)

Israel E. Nieto Granados[1], Gustavo Alonso Silverio[1], Antonio Alarcón Paredes[2]

[1] Universidad Autónoma de Guerrero, Unidad Académica de Ingeniería, México

[2] Instituto Politécnico Nacional Centro de Investigación en Computación, México

23500377@uagro.mx, gsilverio@uagro.mx, aalarcon@cic.ipn.mx

Resumen. Este trabajo presenta el desarrollo de un Sistema de Clasificación de Hojas de Plantas (SiCHoP) que utiliza imágenes RGB, y un sensor espectral basado en el Índice de Diferencia Normalizada (IDN) para abordar el problema de la clasificación de la salud de las hojas en la agricultura mediante un dispositivo portátil. Este sistema es capaz de clasificar las hojas en cinco estados de salud: excelente, bueno, regular, enfermo y muerto. La metodología utilizada se basa en un conjunto de datos propios, obteniendo una precisión del 75% en la clasificación. Los resultados indican una correlación significativa entre los valores de NDI y el estado de salud de las hojas, evidenciando la relación inversa entre el contenido de clorofila y el estrés de la planta. Esta innovación supone un avance en la agricultura de precisión, proporcionando una herramienta eficaz y precisa para el análisis y seguimiento de la sanidad vegetal, que podría mejorar la gestión y productividad agrícola.

Palabras clave: Sistema de clasificación de hojas de plantas (SiCHoP), imágenes RGB, sensor espectral, índice de diferencia normalizada (IDN), clasificación sanitaria, correlación significativa, contenido de clorofila, estrés vegetal, agricultura de precisión, monitorización de la salud de las plantas.

System for Plant Leaf Health Classification (SiCHoP)

Abstract. In this work, the development of a Plant Leaf Health Classification System (SiCHoP) that utilizes RGB images and a spectral sensor based on the Normalized Difference Index (NDI) to address the problem of leaf health classification in agriculture through a portable device is presented. This system is capable of classifying leaves into five health states: excellent, good, regular, diseased, and dead. The methodology employed is based on a proprietary dataset,

achieving an accuracy of 78% in classification. The results indicate a significant correlation between NDI values and leaf health status, evidencing the inverse relationship between chlorophyll content and plant stress. This innovative system represents a significant step forward in precision agriculture, offering farmers a reliable and efficient tool for plant health monitoring and management.

Keywords: Plant leaf classification system (SiCHoP), RGB imaging, spectral sensor, normalized difference index (NDI), health classification, significant correlation, chlorophyll content, plant stress, precision agriculture, plant health monitoring.

1. Introducción

La agricultura es la columna vertebral de la economía de muchos países del mundo, incluido México. Factores como el cambio climático, el crecimiento demográfico y la seguridad alimentaria impulsan la implementación de nuevas formas de mejorar la productividad agrícola. México cuenta con 32.4 millones de hectáreas destinadas a la cosecha, según la Encuesta Nacional Agropecuaria (ENA, 2017). De estas hectáreas, el 21 % están equipadas con sistemas de riego, mientras que el 79 % restante depende de las lluvias para mantener las cosechas (*La agricultura: el motor de nuestra economía*, s/f).

La clasificación de plantas es fundamental en campos como la agricultura, la ecología, la medicina y la conservación. Sin embargo, el proceso de secuenciación tradicional se realiza manualmente, requiere un alto grado de conocimiento y experiencia, y es lento y tedioso, especialmente para colecciones de muestras grandes. Este enfoque manual presenta un problema importante porque limita la capacidad de los agricultores y científicos para monitorear la salud de las plantas de manera efectiva y precisa.

La creciente necesidad de soluciones automatizadas para la clasificación de la salud de las hojas ha llevado al desarrollo de nuevas tecnologías, como el presente Sistema de Clasificación de Salud de Hojas de Plantas (SiCHoP), que utiliza imágenes RGB y un sensor espectral basado en el índice normalizado de la salud de las hojas en cinco categorías: excelente, bueno, regular, enfermo y muerto. Esta innovación tiene como objetivo abordar las limitaciones del método manual, proporcionando análisis más rápidos y precisos que pueden mejorar la gestión agrícola y la conservación de especies (Gama-Moreno et al., 2022a). Sin embargo, los recientes avances en las tecnologías de la información y la comunicación (TIC) han dado lugar a la agricultura de precisión, abriendo nuevas oportunidades para la clasificación automatizada de plantas (Sarma & Nidamanuri, 2023).

Los sistemas de clasificación de imágenes basados en el Índice de Vegetación de Diferencia Normalizada (NDVI) han demostrado ser altamente eficaces en la aplicación de métodos de percepción remota multiespectral. De acuerdo con (Sarma & Nidamanuri, 2023), estos sistemas han mostrado resultados prometedores, con una precisión que varía entre el 65 % y el 85 % en diferentes combinaciones de imágenes y

cultivos, lo que representa un avance significativo en la mejora de la productividad agrícola.

Esta investigación desarrolla un sistema de clasificación de hojas de plantas (SiCHoP) mediante imágenes RGB y la lectura de un sensor espectral, utilizando el Índice de Diferencia Normalizada (NDI). El sistema es capaz de identificar y clasificar las hojas de las plantas en las cinco categorías mencionadas previamente. Esta investigación representa una contribución importante al campo de la clasificación de hojas de plantas, usando un algoritmo de regresión para estimar el NDI y, posteriormente clasificando la salud de la hoja a partir de umbrales previamente establecidos, proporcionando una solución automatizada y precisa para mejorar el análisis y seguimiento de la salud de las plantas.

2. Estado del arte

La relación entre el Índice de Vegetación de Diferencia Normalizada (NDVI) y los niveles de estrés vegetal ha sido ampliamente estudiada, utilizando dispositivos como el SPAD-502 para evaluar clorofila y contenido de nitrógeno en plantas de fresa, arándano y aguacate. En (Gama-Moreno et al., 2022) se mostró que las mediciones de NDVI están correlacionadas con el contenido de nitrógeno en la hoja, revelando una relación directa entre la intensidad del color de la hoja, el contenido de clorofila y la absorción de nitrógeno. La investigación clasificó hojas en tres categorías: jóvenes, maduras y viejas, obteniendo un coeficiente de determinación R^2 de 0.97, lo que permite correlacionar con precisión los valores obtenidos. (Gama-Moreno et al., 2022) ha desarrollado un sistema de sensores inteligentes capaz de medir concentraciones de clorofila utilizando una cámara espectral que lee reflectancia en 12 longitudes de onda discretas en los espectros visible e infrarrojo cercano. Este sistema utiliza cuatro canales (R, G, B, NIR) para calcular diversos índices de vegetación, y ha sido probado en plántulas de tomate para medir el estado de clorofila.

(Gutiérrez-Soto et al., 2011) se centra en el desarrollo de un detector para estimar el estado de la clorofila del maíz mediante análisis espectral, utilizando instrumentos multiespectrales como el Greenseeker Handheld y el sensor OptRx. Este detector consta de una unidad microcontroladora, una cámara multiespectral de doble canal, un módulo de entrada/salida y un módulo de alimentación. El sistema ha demostrado ser eficaz en la detección no destructiva de clorofila.

En (Revelo Luna et al., 2021), se desarrolla un sistema de monitoreo de cultivos que capturó imágenes multiespectrales y calculó índices de vegetación como NDVI, RVI y NDGI para evaluar el estado de crecimiento de plantas de tomate. Un modelo de regresión lineal múltiple (MLR) se utilizó para estimar el contenido de clorofila basándose en los valores promedio de grises en diferentes canales e índices de vegetación, logrando un R^2 de 0.88. (Andrianto et al., 2020) probó un medidor de clorofila basado en IoT de bajo costo contra un espectrofotómetro (SP-3000nano) y un medidor comercial (SPAD-502) en hojas de Maniltoa Grandiflora, mostrando una fuerte correlación en las mediciones del contenido de clorofila. Este dispositivo mide exitosamente la clorofila, muestra los datos en una pantalla LCD, almacena la

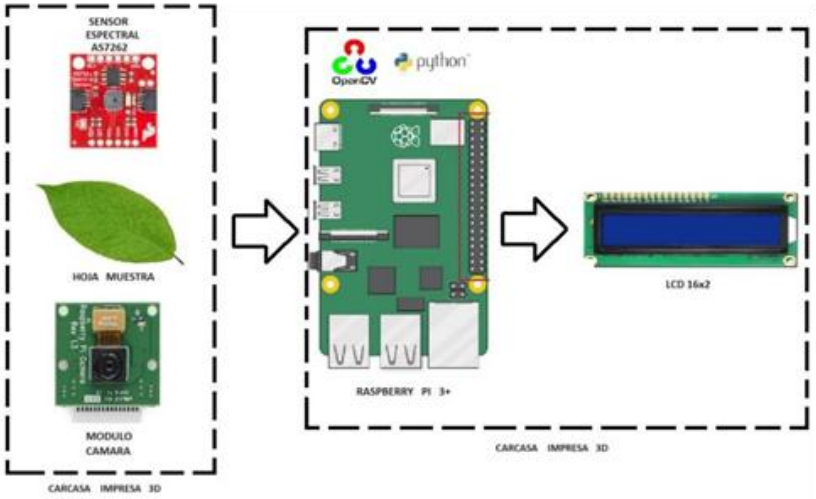


Fig. 1. Esquema de configuración para el sistema de clasificación de hojas de planta.

información en memoria y transmite los datos a una plataforma de servicio, sin mencionar el uso de una cámara.

Estos avances en la tecnología de sensores y análisis espectral han mejorado significativamente la precisión y eficiencia en la clasificación de plantas y la detección de clorofila, proporcionando herramientas cruciales para la agricultura de precisión y la gestión sostenible de cultivos.

3. Sistema propuesto

En la Figura 1 se muestra un esquema de configuración del sistema propuesto, capaz de realizar una clasificación de hojas de planta para determinar su estado de salud. En el presente trabajo se utilizó un conjunto de datos tomado desde la cámara del dispositivo, así como los datos recopilados del sensor espectral, hasta la creación de su correspondiente programación en Python, también se realizaron las pruebas, la reproducibilidad de los resultados y la validación de estos.

A continuación, se describen los módulos presentes en el sistema propuesto.

3.1. Conjunto de datos

Tras haber completado el análisis del estado del arte, se exploró en diversos repositorios en busca de bancos de datos que brindaran información sobre imágenes de hojas de plantas a partir de la medición de su NDI. Sin embargo, no se encontró ningún banco de datos que se adaptara al propósito de este trabajo, por ello, se optó por construir un banco de datos propio en involucra la toma de imágenes (Figura 2) y la medición de valores específicos que permiten un análisis de su salud.

Tabla 1. Conjunto de datos.

Categoría de Hoja	Número de Hojas	Imágenes por Hoja	Total de Imágenes	Mediciones Espectrales por Hoja	Total de Mediciones Espectrales
Excelente	10	10	100	10	100
Buena	10	10	100	10	100
Regular	10	10	100	10	100
Enferma	10	10	100	10	100
Muerta	10	10	100	10	100
Total	50	-	500	-	500

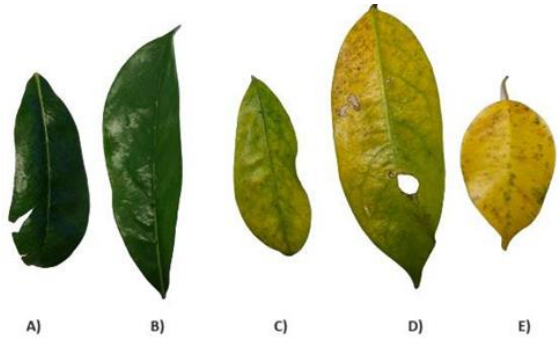


Fig. 2. Ejemplo de hojas del conjunto de datos para la toma de muestras. A) Excelente, B) Buena, C) Regular, D) Enferma, E) Muerta.

En este estudio, se utilizó un conjunto de datos propio que fue recopilado con el propósito de evaluar el rendimiento del Sistema de Clasificación de Hojas de Plantas (SiCHoP). El conjunto de datos se adquirió tomando en consideración dos enfoques: 1) imágenes de hojas por medio de una cámara RGB, y 2) mediciones obtenidas de las hojas a través de un sensor espectral. En ambos casos, se obtuvieron imágenes y datos de hojas en cinco categorías predefinidas: excelente, buena, regular, enferma y muerta.

Se tomaron en consideración 50 hojas distribuidas uniformemente en las cinco categorías de salud, es decir 10 hojas de cada categoría. Para cada hoja, se tomaron 10 imágenes utilizando una cámara RGB, lo que resultó en un total de 500 imágenes. Además, se obtuvieron 10 mediciones espectrales por hoja, generando un total de 500 mediciones espectrales.

Las muestras de la Tabla 1 fueron recolectadas en condiciones semicontroladas de luz para minimizar las variaciones debidas a factores externos. Se utilizó una cámara Raspberry Pi Rev 1.3 de 5MP para capturar imágenes RGB de cada hoja, y un sensor espectral SparkFun Breakout AS7262 para las mediciones espectrales. Las imágenes se tomaron desde diferentes ángulos para obtener una representación más completa y variada de las hojas, mientras que las mediciones espectrales se realizaron directamente

Tabla 2. Listado de Hardware y Software utilizados.

Hardware	Software
Raspberry Pi 3b+	Python + Thonny IDE
Sensor Espectral SparkFun Breakout - AS7262 Visible	
Cámara Raspberry Pi Rev 1.3 5MP	
Pantalla de Cristal Líquido de 16x2 (LCD)	
Resistencia variable 10KΩ	
Cables de conexión (Jumpers)	
Carcasa en 3D (diseño e impresión)	

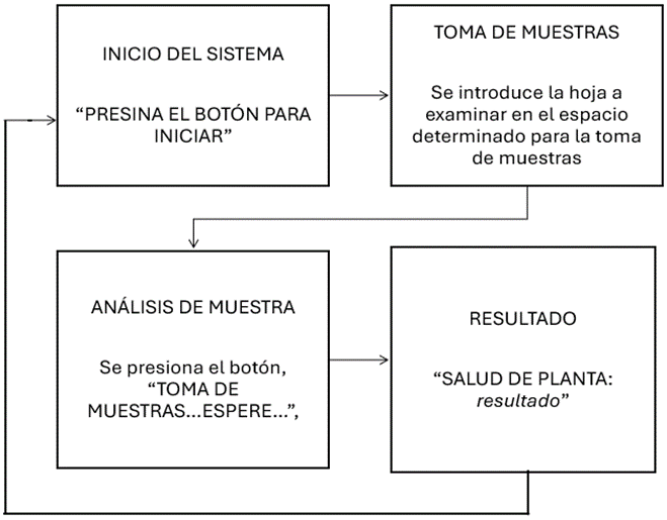


Fig. 3. Diagrama a bloques del sistema para la clasificación de salud de hojas de planta.

sobre la superficie de las hojas en varias posiciones para garantizar la precisión de los datos.

Este conjunto de datos proporciona una base sólida para el desarrollo y la validación del Sistema de Clasificación de Hojas de Plantas (SiCHoP). La distribución equilibrada de las muestras y la replicación de mediciones aseguran la robustez y fiabilidad del modelo, permitiendo una evaluación precisa del estado de salud de las hojas.

3.2. Hardware y software

Para esta sección, se llevaron a cabo pruebas utilizando los sensores, así como ensayos en campo, seguidos de un análisis detallado de los resultados obtenidos. Sírvase observar la Tabla 2 para obtener el listado de hardware y software utilizado para el sistema.

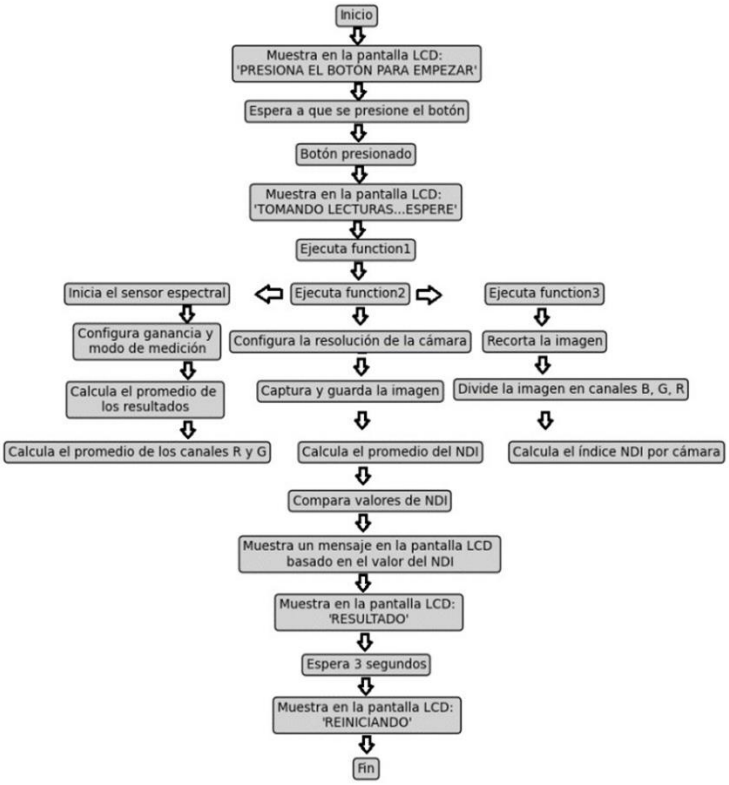


Fig. 4. Diagrama de flujo del sistema propuesto.

3.3. Funcionamiento del sistema propuesto

En esta sección se describe la forma en cómo opera el sistema propuesto. Para ello, en la Fig. 3 se ilustra el procedimiento metodológico que se siguió en esta etapa del estudio, y en la Fig. 4 se muestra el diagrama de flujo de éste.

3.4. Diseño del dispositivo

Para el presente proyecto de investigación se desarrollaron dos carcasas impresas en 3D utilizando Tinkercad® (Figura 5), la primera de ellas contiene la tarjeta Raspberry Pi 3 conectada a un display de 16x2 (izquierda), también se diseñó otra carcasa dividida en tres partes (derecha) para colocar el módulo de cámara y el módulo de sensor espectral.

Una vez impreso, programado y armado el sistema para la toma de muestras, en la Figura 6 se muestra el sistema tomando muestras a diferentes tipos de hojas arrojando dos resultados de acuerdo con la clasificación correspondiente a la toma de valores.

3.5. Modelo propuesto

El método de experimentación se basa en la recopilación de datos a través del sistema de adquisición que utiliza la cámara y el sensor espectral referidos con antelación. Para este propósito se utilizó el Índice de Diferencia Normalizada (NDI, Normalized Difference Index) para analizar los datos de salud de las hojas. El valor

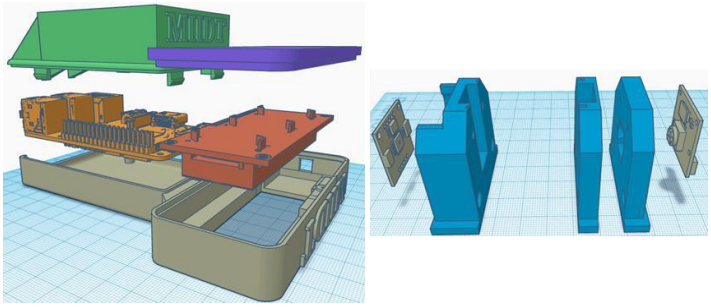


Fig. 5. Carcasa para LCD y Raspberry (izquierda), carcasa para AS7262 y cámara (derecha).

Fig. 6. Toma de muestras en campo con el SiCHoP.

NDI se obtiene a partir de la siguiente expresión:

$$NDI = (G - R) / (G + R), \quad (1)$$

donde G y R indican los canales verde y rojo, tanto de la imagen, como del espectro visible leídos por la cámara RGB y por el sensor espectral, respectivamente.

El NDI permite determinar qué tan saludable es una planta, basado en cómo refleja la energía y la luz. Para el ojo humano, una planta es verde porque el pigmento de clorofila que contiene refleja las ondas verdes y absorbe las rojas. Esto significa que una planta sana absorbe activamente la luz roja, en una planta no saludable ocurre exactamente lo contrario (Revelo Luna et al., 2021). En la Tabla 3 se muestran las mediciones promedio tras analizar diez veces la misma muestra, en donde se obtiene

Tabla 3. Promedios de NDI por imagen y sensor espectral en diferentes tipos de hoja.

Salud de la hoja	No. imágenes	Promedio de NDI por cámara	Promedio del NDI por sensor espectral
Excelente	10	279.366801±10%	51.771777±10%
Buena	10	247.232234±10%	34.460867±10%
Regular	10	186.614082±10%	9.149693±10%
Enferma	10	172.858158±10%	0.909121±10%
Muerta	10	164.151569±10%	-6.344167±10%

un promedio del NDI calculado por cada sensor en cada una de las cinco categorías propuestas.

Una vez que se calculó el NDI a partir de las 500 imágenes y las 500 mediciones del sensor espectral, los datos fueron utilizados para ajustar un modelo de regresión polinómica de segundo grado. La idea es que el modelo entrenado pueda reflejar la relación no lineal entre los valores del NDI y el estado de salud de las hojas.

La ecuación del modelo de regresión utilizado es la que se muestra a continuación:

$$y = a \cdot (NDI)^2 + b \cdot NDI + c, \tag{2}$$

donde:

– a, b y c son los coeficientes determinados durante el proceso de ajuste del modelo de regresión sobre los datos observados.
– NDI es el Índice de Diferencia Normalizada calculado para cada muestra de hoja a partir de la ecuación (1).

Hasta este momento sólo se ha estimado un valor de NDI que puede ser cercano a alguna de las categorías de salud propuestas, pero será necesario establecer umbrales de decisión para poder estimar la salud de cada hoja, tomando como base el NDI.

4. Resultados

El modelo polinómico mostró un ajuste robusto a los datos, con coeficientes de determinación (R^2) de 0.9698 y 0.9867, para las imágenes y las mediciones espectrales, respectivamente, lo que indica que el modelo es viable para la predicción de la salud de las hojas en función de los valores de NDI obtenidos tanto por la cámara RGB como por el sensor espectral. Estos resultados confirman que el uso de un modelo polinómico permite establecer una relación entre el NDI y la condición fisiológica de las plantas.

Aun así, para clasificar las hojas en las cinco categorías de salud (excelente, buena, regular, enferma y muerta) en función de los valores del NDI, es necesario establecer umbrales específicos. Estos umbrales determinan los rangos de NDI que corresponden a cada categoría, permitiendo que el sistema clasifique automáticamente la salud de las hojas basándose en los valores obtenidos por la cámara RGB y el sensor espectral. Los umbrales seleccionados se muestran en la Tabla 4.

Tabla 4. Umbrales de clasificación por categoría.

Categoría de Salud	Rango de NDI (Cámara RGB)	Rango de NDI (Sensor Espectral)
Excelente	NDI > 250	NDI > 45
Buena	200 < NDI ≤ 250	30 < NDI ≤ 45
Regular	150 < NDI ≤ 200	5 < NDI ≤ 30
Enferma	100 < NDI ≤ 150	0 < NDI ≤ 5
Muerta	NDI ≤ 100	NDI ≤ 0

Los umbrales se seleccionaron de manera que reflejen con precisión la degradación progresiva de la salud de las hojas. Por ejemplo, hojas con un NDI mayor a 250 en la cámara RGB mostraron un alto contenido de clorofila y se clasificaron como "Excelentes", mientras que hojas con un NDI por debajo de 100 indicaron una falta significativa de clorofila, clasificándolas como "Muertas". De manera similar, los valores de NDI obtenidos por el sensor espectral mostraron un patrón decreciente consistente con la disminución en la salud de las hojas, justificando la asignación de rangos específicos para cada categoría.

La correcta definición de los umbrales de NDI es crucial para garantizar la precisión del Sistema de Clasificación de Hojas de Plantas (SiCHoP). Estos umbrales permiten que el sistema clasifique de manera efectiva la salud de las hojas en las categorías apropiadas, proporcionando un método automatizado y confiable para evaluar la salud de las plantas y mejorar la gestión agrícola.

El análisis cuantitativo se centra en la comparación de los promedios de NDI obtenidos por el sistema propuesto y su correlación con las categorías de salud. Se registraron promedios de NDI para cada categoría:

– *Excelente*: NDI promedio de 279.37 (cámara) y 51.77 (sensor espectral)

– *Buena*: NDI promedio de 247.23 (cámara) y 34.46 (sensor espectral)

– *Regular*: NDI promedio de 186.61 (cámara) y 9.15 (sensor espectral)

– *Enferma*: NDI promedio de 172.86 (cámara) y -0.91 (sensor espectral)

– *Muerta*: NDI promedio de 164.15 (cámara) y -6.34 (sensor espectral)

La tendencia decreciente del NDI es evidente, mostrando una reducción significativa desde la categoría Excelente hasta Muerta. Los coeficientes de determinación R^2 de 0.9698 y R^2 de 0.9867 indican un fuerte ajuste del modelo de regresión a los datos observados.

Estos resultados reflejan una disminución constante en los valores de NDI a medida que se avanza de una categoría de salud a otra, lo que respalda la efectividad del sistema para evaluar el estado fisiológico de las plantas. Además, la alta correlación entre los valores de NDI y el estado de salud de las hojas sugiere que el sistema puede ser una herramienta valiosa para la gestión sostenible de cultivos.

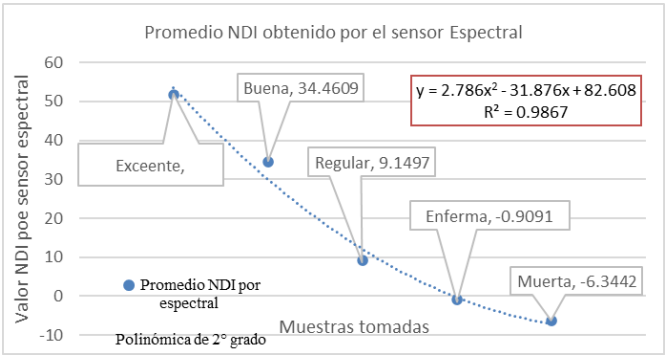


Fig. 7. Gráfico de promedio del NDI por sensor espectral.

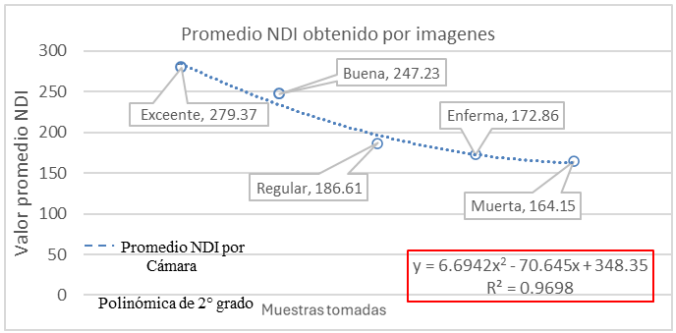


Fig. 8. Gráfico de promedio del NDI por imagen RGB.

Tabla 5. Matriz de confusión.

PREDICCIÓN / REAL	Excelente	Buena	Regular	Enferma	Muerta
Excelente	9	1	0	0	0
Buena	0	8	2	0	0
Regular	0	1	7	2	0
Enferma	0	0	1	6	3
Muerta	0	0	0	1	9

Este análisis resalta la relación inversa entre el NDI y la degradación de la condición de salud, subrayando la precisión del modelo. De acuerdo con (Gama-Moreno et al., 2022b), la intensidad del color de la hoja está relacionada con el contenido de clorofila y la cantidad de nitrógeno en la hoja.

La matriz de confusión Tabla 5, muestra un desempeño aceptable del modelo en la clasificación de la salud de las hojas basado en los promedios de NDI proporcionados,

Tabla 6. Ejemplo de imagen y comparativa de NDI por foto y por sensor espectral.

Imagen	Resultado	NDI por cámara	NDI por sensor espectral
	Excelente	265.398461	49.18318815
	Buena	259.5938457	36.18391035
	Regular	177.233779	8.6920835
	Enferma	181.5010659	0.95457705
	Muerta	155.9439906	-6.02695865

con una precisión en todas las categorías y una correlación aceptables entre los valores de NDI y el estado fisiológico de las plantas. Estos resultados son consistentes con los coeficientes de determinación elevados, indicando un ajuste óptimo del modelo a los datos observados.

A partir de la matriz de confusión es posible ver que, tras llevar a cabo pruebas en hojas de plantas de varias especies, aunque las hojas parecen saludables, los resultados proporcionados por el sistema no son consistentes con esta opinión. Esto se puede deber a varias razones, entre ellas, por la luz que incide en el sensor espectral durante la lectura, o bien debido a que para las imágenes la región de interés (ROI) influye en la evaluación de la salud de las hojas.

En la Tabla 6 se enlistan los resultados tomados a muestras aleatorias de hojas de plantas de diferentes especies. Así como los valores NDI obtenidos por cada sensor utilizado.

5. Conclusiones

El SiCHoP representa una mejora sobre tecnologías existentes al ofrecer una solución más accesible y económica en comparación con dispositivos comerciales como el SPAD-502. A diferencia de sistemas que centran su análisis a un número limitado de categorías o requerían equipos costosos, este sistema permite clasificar las hojas en cinco categorías de salud utilizando un algoritmo sencillo implementado en una computadora embebida. Esto facilita el monitoreo de la salud de las plantas de manera más eficiente y precisa, contribuyendo a la agricultura de precisión.

Este avance no solo optimiza el análisis de la condición de las hojas, sino que también proporciona una base sólida para futuras investigaciones en la agricultura de precisión donde se requieren soluciones prácticas y accesibles. La implementación de SiCHoP representa un paso significativo hacia la mejora de la productividad agrícola y la gestión sostenible de cultivos mediante el uso de tecnologías avanzadas.

Esta tecnología ofrece una alternativa económica y fiable a dispositivos comerciales como el SPAD-502. Sin embargo, la omisión del análisis de textura limita su capacidad para detectar patrones más complejos, señalando la necesidad de mejoras en esta área para aumentar la precisión y robustez del sistema. Las investigaciones futuras deberían centrarse en superar estas limitaciones e incorporar el análisis de textura para incrementar la precisión y eficacia del sistema en aplicaciones agrícolas. En resumen, a pesar de sus limitaciones, el SiCHoP es un indicador importante para la clasificación de hojas de plantas y proporciona a los expertos una herramienta sencilla y confiable.

Referencias

1. Andrianto, H., Suhardi, S., Faizal, A.: Performance Evaluation of Low-Cost Iot Based Chlorophyll Meter. Bulletin of Electrical Engineering and Informatics, 9(3), pp. 956–963 (2020). doi: 10.11591/eei.v9i3.2014.
2. Byun, N., Lee, J., Won, J.Y., Kang, Y.J.: Structural Responses Estimation of Cable-Stayed Bridge from Limited Number of Multi-Response Data. Sensors, 22(10), pp. 3745 (2022). doi: 10.3390/s22103745.
3. Chen, D., Hu, H., Liao, C., Ye, J., Bao, W., Mo, J., Wu, Y., Dong, T., Fan, H., Pei, J.: Crop NDVI Time Series Construction by Fusing Sentinel-1, Sentinel-2, and Environmental Data with an Ensemble-Based Framework. Computers and Electronics in Agriculture, 215, pp. 108–388 (2023). doi: 10.1016/j.compag.2023.108388.
4. Flores Rodríguez, A.G., Flores-Garnica, J.G., González-Eguiarte, D.R., Gallegos-Rodríguez, A., Zarazúa-Villaseñor, P., Mena-Munguía, S.: Análisis comparativo de índices espectrales para ubicar y dimensionar niveles de severidad de incendios forestales. Investigaciones geográficas, 106 (2021). doi: 10.14350/rig.60396.
5. Gama-Moreno, L.A., Plazola Soltero, V.H., Murguia Vadillo, C.G., Martínez Hernández, C., López Carrillo, E.: Prototipo de cámara infrarroja para obtener el índice NDVI en agricultura de precisión. Programación matemática y software, 14(1) (2022). doi: 10.30973/progmat/2022.14.1/2.
6. Gutiérrez-Soto, M.V., Cadet-Piedra, E., Rodríguez-Montero, W., Araya-Alfaro, J.M.: El GreenSeekerTM y el diagnóstico del estado de salud de los cultivos. Agronomía mesoamericana, 22(2), pp. 397 (2011). doi: 10.15517/am.v22i2.11799.

7. La agricultura: El motor de nuestra economía. https://tecscience.tec.mx/es/divulgacion-ciencia/la-agricultura-el-motor-de-nuestra-economia/Girimonte-Garcia-Fronti.
8. NDVI: Fórmula y uso del índice de vegetacion NA agricultura. https://eos.com/es/make-an-analysis/ndvi/NDVI_clasificacion_Kmeans.
9. Revelo Luna, D.A., Mejía Manzano, J., Montoya Bonilla, B., Hoyos García, J.: Analysis of the Vegetation Indices NDVI, GNDVI, and NDRE for the Characterization of Coffee Crops (Coffea arabica). Ingeniería y Desarrollo, 38(2), pp. 298–312 (2021). DOI:10.14482/inde.38.2.628
10. Sarma, A.S., Nidamanuri, R.R.: Transfer Learning for Plant-level Crop Classification using Drone-based Hyperspectral Imagery. In: 2023 International Conference on Machine Intelligence for GeoAnalytics and Remote Sensing (MIGARS), pp. 1–4 (2023). doi: 10.1109/MIGARS57353.2023.10064501
11. Thonny.: Python IDE for beginners. https://thonny.org/
12. Tinkercad.: Cree diseños digitales 3D con CAD en línea. https://www.tinkercad.com/
Welcome to Python.org. https://www.python.org/

ISSN 1870-4069

Effects of Preprocessing on Microscopic Images for the Giardia Lamblia Detection

Alberto García-Ochoa, Luis Pellegrin

Universidad Autónoma de Baja California,
Faculty of Sciences,
Mexico

{lgarcia98,luis.pellegrin}@uabc.edu.mx

Abstract. This paper presents a comparison of deep learning architectures, MobileNetV2 and Inception v3, using different datasets to analyze the efficiency of Giardia lamblia parasite detection in bright field microscopy images. The datasets were preprocessed with techniques such as image cropping, and then used to train both architectures. The comparison was carried out using metrics such as accuracy, precision, and recall. The experimental results demonstrate that preprocessing techniques, particularly image cropping, significantly improved the models' performance. This comparative approach underscores the importance of preprocessing strategies in optimizing deep learning models for specific detection tasks.

Keywords: Giardia lamblia, deep learning, detection.

1 Introduction

In recent years, deep learning has significantly impacted computer vision tasks such as image classification, object detection, face recognition, and semantic segmentation, almost replacing traditional feature extraction methods [2]. Specifically on the detection of the waterborne parasite *Giardia lamblia*, there have been some advancements. However, some of these approaches are not up-to-date with recent deep learning architectures and more importantly, some still use traditional feature extraction methods because of the way the images were collected. These two factors can introduce significant variability in the deep learning models; thus the importance of exploring simple preprocessing approaches such as image cropping in order to improve the performance of deep learning models.

G. lamblia is a protozoan parasite that reproduces in the small intestine after being ingested, causing giardiasis in both animals and humans [7]. Symptoms of *giardiasis* include diarrhea, mild fever, fatigue, and myalgia. In rare cases, *G. lamblia* parasites can infect the lungs and trachea, resulting in coughing, dehydration, and extreme weight loss in humans [7]. The detection process of parasite involves sampling, isolation, staining, and microscopy. Under the

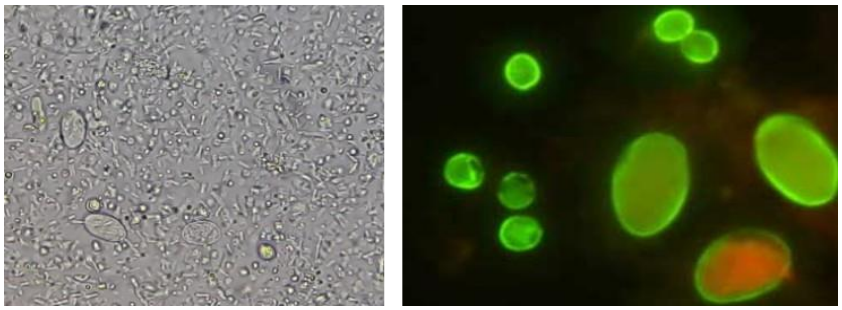


Fig. 1. *G. lamblia* parasites through clear microscopy; through immunofluorescence technique.

microscope, direct visual inspection is performed by an expert to detect the presence of the parasite, requiring a considerable amount of time.

In environmental samples, the gold standard is the immunofluorescence technique, as this parasite can be easily confused with other microorganisms [7,8]. Although the immunofluorescence technique offers advantages, it requires specialized equipment and trained personnel. On the other hand, detection of *G. lamblia* with brightfield microscopy has the advantages of being easy to perform and being cost-effective. However, it also has some disadvantages such as being time-consuming, low detection accuracy and difficulty for uncommon microorganisms detection [5]. For instance, in Fig. 1, *G. lamblia* parasites are shown in microscopic images under the two aforementioned techniques.

Although the references used in this section might appear dated, they provide foundational insights into the methods still relevant today. For example, Methods based on neural networks [11]; applying digital image processing techniques that use shape and color features [3]; or through the application of filters that seek to remove noise and preserve edges [1]. Nevertheless, in environmental water samples, low contrast and noise in microscopic images can significantly affect segmentation accuracy, as water samples may contain a low number of cysts and contamination components that hinder strain differentiation and multilocus molecular analysis of G. lamblia strains [7].

In the past two decades, the development of computational methods for the detection of microorganisms such as G. lamblia has relied on the use of deep learning-based architectures due to their high effectiveness [5]. Different approaches have been made, for instance, one study developed a field-portable, cost-effective imaging flow cytometer using lens-free holographic imaging accurately detected G. lamblia cysts in water samples at 100 mL/h, identifying cysts in real-time without labels [4].

Another approach involves microscopic images from fruit samples using a smartphone microscope, they used different deep learning architectures and compared them [6]. Although these proposals obtained favorable results using deep learning, both rely on specialized hardware with certain limitations such as cost, accessibility and quality issues.

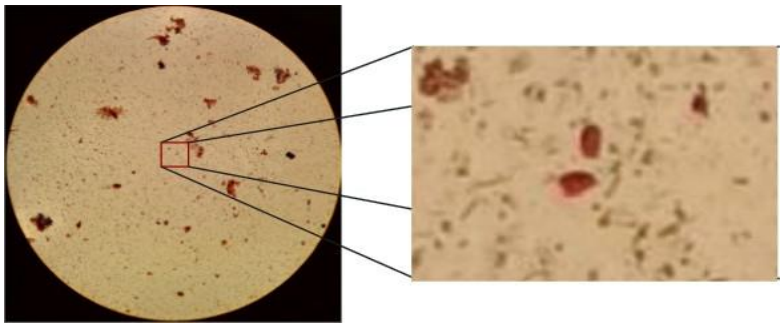


Fig. 2. *G. lamblia* in idoine seen through 10x.

Considering the above, in this work a comparison between two deep neural network methods was done, the models were trained using bright field microscopic images taken from a cheap and accessible way of sampling. The images were preprocessed in three different ways resulting in three datasets. The preproccesing consisted in different image cropping sizes. The models were trained on these datasets in order to analyze the effects of each cropping size on the detection of the G. lamblia parasite in microscopic images.

2 Datasets

The images were collected using a Xiaomi Redmi Pro smartphone camera and a ZEISS Primo Star commercial microscope. To ensure the stability of the smartphone and reduce image blurring caused by movement, a phone holder was employed to attach the phone to the microscope. This setup minimized vibrations and maintained consistent focus during image capture.

For the preparation of the samples, a 15-micrometer micropipets was used. This allowed precise handling and placement of the samples onto microscope slides, ensuring clear and detailed imaging under the microscope. The careful preparation and mounting of the samples were crucial to obtaining high-quality images for the dataset.

The photos captured included both positive and negative G. lamblia samples stained with iodine, with the microscope set to 10x magnification (e.g. see Fig.2).

No additional filters or zoom were applied to the smartphone camera during the process.

Subsequently, different processes were performed on the images to facilitate their analysis and use. First, the images were processed to meet specific criteria necessary for subsequent tests. Each image had a substantial black spot that occupied more than half of the frame (e.g. see Fig.3), which rendered those portions unusable for analysis. To enhance the quality and usability of the images, these black spots were meticulously removed. This preprocessing step ensured that the remaining parts of the images were suitable for accurate and reliable testing.

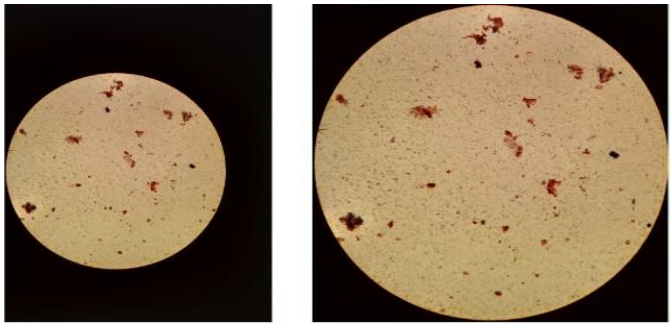


Fig. 3. Image before taking out the black spot (3120×4160), and after taking out the black spot (2580×2580).

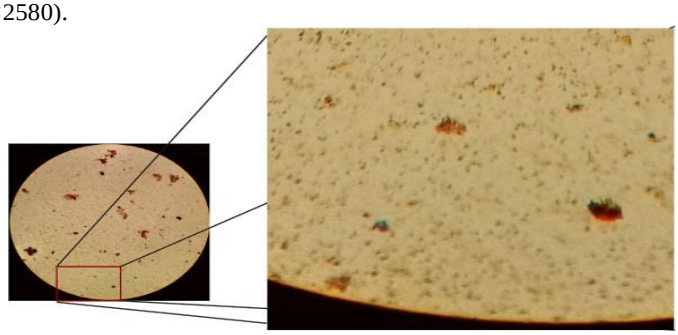


Fig. 4. Blurry parts from the images.

Another issue that arose was the blurriness around the outer parts of the field of view (e.g. see Fig.4). To address this problem, the images were cropped by approximately 18% from the edges. This cropping significantly improved the clarity of the images, ensuring that the central portion, which remained in focus, was used for analysis. By eliminating the blurry edges, the overall quality and reliability of the images were enhanced, making them more suitable for detailed examination.

Next, we created two additional datasets with different image sizes where each image was divided into quadrants as shown in Fig. 4, i.e. from the original we created the second one and the third one. The first dataset did not have any cropping and consisted of 940 images with a size of 1650×1650 pixels. The second dataset had a four quadrant crop and consisted of 3,760 images with a size of 825×825 pixels. And the third dataset had a sixteen quadrant crop and consisted of 15,040 images with a size of 412×412 pixels. Two classes were considered: one with the presence of G. lamblia and one without. No data augmentation techniques were applied.

It is important to note that there was a considerable class imbalance in the datasets, with a significantly higher number of images in the class with the presence of G. lamblia compared to the class without its presence. It is expected that this imbalance could influence the performance of the models, leading them

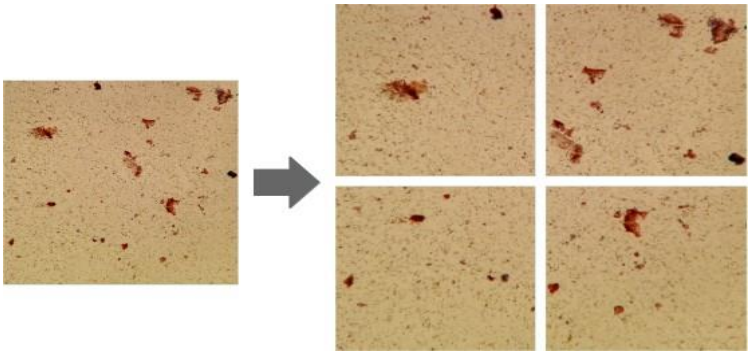


Fig. 5. Images divided into quadrants.

to favor the class with Giardia, and thus impacting the accuracy and reliability of the results.

3 Experiments and Results

The main objective of this work is to understand the effects of image cropping in a deep learning model's performance for identifying microorganisms, i.e., G. lamblia parasite. To achieve this, three datasets were used: one without cropping, a second with images cropped into four quadrants, and a third with images cropped into sixteen quadrants. Each dataset was split into the same images with 80% for training and 20% for testing the architectures, although these images are fed with different sizes in the evaluated architectures. A comparison was performed to analyze the resulting effects.

3.1 Deep Learning Architectures and Settings

Two deep learning architectures were selected: MobileNetV2 [9] and InceptionV3 [10]. MobileNetV2 is highly efficient and simple, it uses depth wise separable convolutions and inverted residual blocks to reduce parameters and operations. It has an excellent balance between precision and efficiency, making it ideal for resource-constrained devices like mobile phones [9].

On the other hand, InceptionV3 is able to extract complex features, depth, and it has proven performance in benchmarks. Its architecture allows for capturing detailed image information while reducing the risk of overfitting, making it ideal for demanding image classification tasks [10].

The experiments were conducted using an Nvidia 1050ti 4GB GPU and 24 GB of RAM. For a fair comparison both models were trained for 20 epochs with a learning rate of 0.001 and compiled using the Adam optimizer. The MobileNetV2 model used a GlobalAveragePooling2D layer, followed by Dense layers with regularization, while InceptionV3 was set up similarly with dropout to prevent

Table 1. Performance metrics for different experiments.

Datasets	Architecture	Accuracy	Precision	Recall
without crop	MobileNetV2	89.9	100	88.3
without crop	InceptionV3	86.2	100	84.7
Four quadrant	MobileNetV2	93.6	100	92.3
Four quadrant	InceptionV3	90.4	100	88.8
Sixteen quadrant	MobileNetV2	94.7	100	93.5
Sixteen quadrant	InceptionV3	93.6	80.4	98.5

overfitting. Key metrics such as precision, recall and accuracy were used to evaluate the model's performance.

3.2 Evaluating Deep Learning Architectures

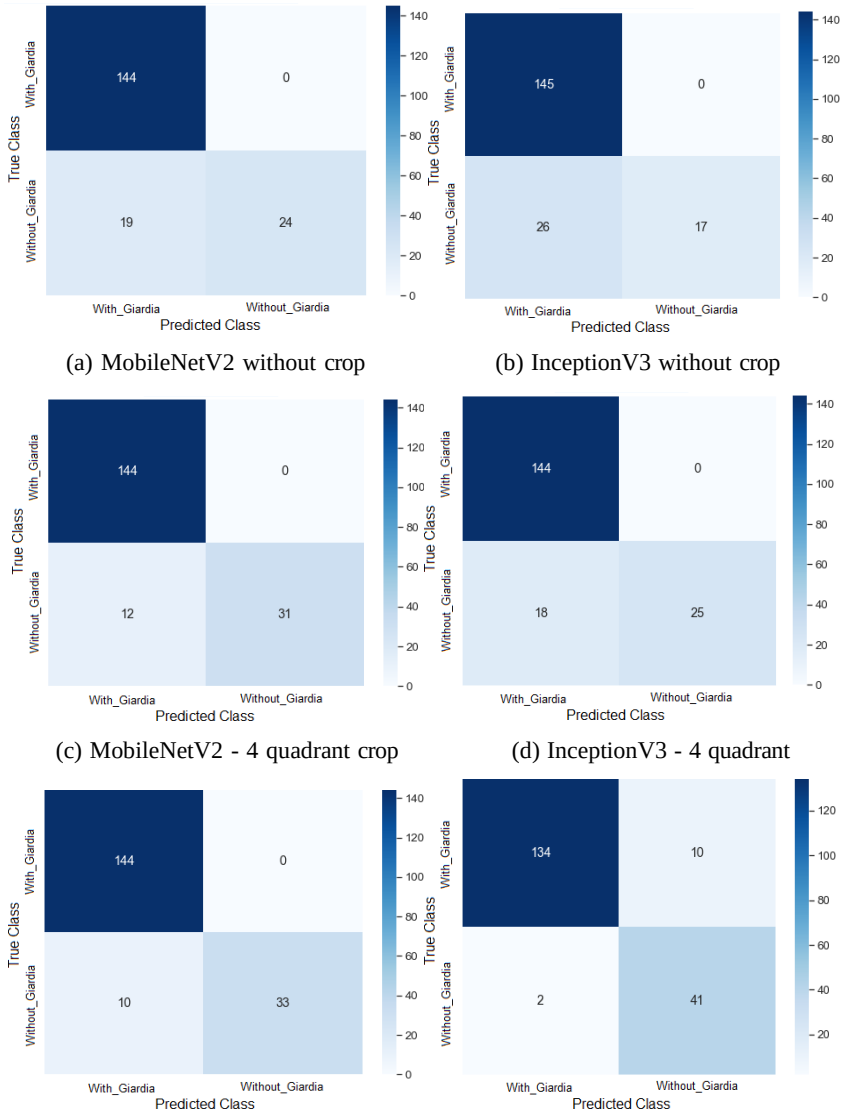
Both models showed better results with the sixteen quadrant crop dataset and among the two architectures, MobileNetV2 had the best results achieving an accuracy of 89.9% on the without crop dataset, 96.6% on the four quadrant dataset and 94.7% on the sixteen quadrant dataset (Table 1).

In Fig. 6, the confusion matrices show that MobileNetV2 had better results than InceptionV3 in terms of balancing true positives and minimizing false negatives on all datasets. MobileNetV2 handled the class imbalance better, achieving higher true negative rates. However, InceptionV3, while slightly less consistent, had a significant reduction in false negatives with the four and sixteen quadrant datasets, this suggests that image cropping into quadrants helped the model to focus on relevant features. Furthermore, given the class imbalance, the models still managed to perform well on the minority class, suggesting a robustness that warrants further exploration. The incidence of false positives was acceptable, considering the nature of the problem, which suggests it's preferable to classify a sample as contaminated rather than incorrectly stating otherwise, prioritizing caution.

4 Conclusions

The findings of this study indicate that image cropping has better results on the two architectures for the detection of G. lamblia. In fact, the sixteen quadrant crop achieved the best results. This suggests that selectively cutting relevant portions of an image enhances the model's ability to capture essential features, leading to improved performance in identifying G. lamblia cysts.

MobileNetV2 was the top-performing architecture, consistently achieving superior results in the three proposed datasets. This results show the effectiveness of MobileNetV2 in handling image variations and extracting relevant information



(a) MobileNetV2 without crop (b) InceptionV3 without crop

(c) MobileNetV2 - 4 quadrant crop (d) InceptionV3 - 4 quadrant

(e) MobileNetV2 - 16 quadrant crop (f) InceptionV3 - 16 quadrant crop

Fig. 6. MobileNetV2 and InceptionV3 confusion matrices.

for the G. lamblia detection task. On the other hand, InceptionV3 also had good results but it did not surpass MobileNetV2.

The observed trend suggests that smaller crop sizes result in better outcomes for G. lamblia detection. This aligns with the intuition that finer granularity in selecting relevant image regions allows the model to focus on more

discriminative features, leading to enhanced detection accuracy. However, the presented results between cropping sizes were not remarkable, so it can also be said that making smaller images means a bigger effort in manual classification which is time consuming specially for G. lamblia detection, which may not be worth it even if the results were better slightly better.

References

1. Badsha, S., Mokhtar, N., Arof, H., Lim, Y.A.L., Mubin, M., Ibrahim, Z.: Utomatic Cryptosporidium and Giardia Viability Detection in Treated Water, 56 (2013). doi: 10.1186/1687-5281-2013-56.
2. Bayoudh, K.: A Survey of Multimodal Hybrid Deep Learning for Computer Vision: Architectures, Applications, Trends, and Challenges. Information Fusion, 105, pp. 102217 (2024). doi: 10.1016/j.inffus.2023.102217.
3. Fernandez-Canque, H., Beggs, B., Clafferty, E., Boutaleb, T., Smith, H., Hintea, S.: Microorganisms Detection in Drinking Water Using Image Processing. In: 6th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine (ITAB) Conference (2006)
4. Göröcs, Z., Baum, D., Song, F., de Haan, K., Koydemir, H., Qiu, Y., Cai, Z., Skandakumar, T., Peterman, S., Tamamitsu, M., Ozcan, A.: Deep Learning-Enabled Holographic Imaging Flow-Cytometry for Label-Free Detection of Giardia Lamblia in Water Samples. In: OSA Imaging and Applied Optics Congress 2021 (3D, COSI, DH, ISA, pcAOP) (2021). doi: 10.1364/DH.2021.DF4C.1.
5. Ma, P., Li, C., Rahaman, M., Yao, Y., Zhang, J., Zou, S., Zhao, X., Grzegorzek, M.: A State-of-the-Art Survey of Object Detection Techniques in Microorganism Image Analysis: from Traditional Image Processing and Classical Machine Learning to Current Deep Convolutional Neural Networks and Potential Visual Transformers. Artificial Intelligence Review, pp. 1627–1698 (2023)
6. Nakarmi, S., Pudasaini, S., Thapaliya, S., Upretee, P., Shrestha, R., Giri, B., Neupane, B.B., Khanal, B.: Deep-Learning Assisted Detection and Quantification of (oo)cysts of Giardia and Cryptosporidium on Smartphone Microscopy Images. Machine Learning for Biomedical Imaging, 2, pp. 956–976 (2024). doi: 10.59275/j.melba.2024-a333.
7. Plutzer, J., Ongerth, J., Karanis, P.: Giardia Taxonomy, Phylogeny and Epidemiology: Facts and Open Questions. International Journal of Hygiene and Environmental Health, 213, pp. 321–33 (2010). doi: 10.1016/j.ijheh.2010.06.005.
8. Rosado García, F., Guerrero, M., Karanis, G., Alvarez, M.D.C., Karanis, P.: Water-Borne Protozoa Parasites: The Latin American Perspective. International Journal of Hygiene and Environmental Health, 220 (2017). doi: 10.1016/j.ijheh.2017.03.008
9. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted Residuals and Linear Bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018)
10. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016)
11. Widmer, K., Srikumar, D., Pillai, S.: Use of Artificial Neural Networks to Accurately Identify Cryptosporidium Oocyst and Giardia Cyst Images. Applied and Environmental Microbiology, 71, pp. 80–4 (2005). doi: 10.1128/AEM.71.1.80-84.

Electronic edition
Available online: http://www.rcs.cic.ipn.mx

http://rcs.cic.ipn.mx

Centro de Investigación
en Computación