FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 126-130 | e-ISSN: 2347-470X

On the Connection of Matroids and Greedy Algorithms

Zaheer A. Siddiqui[1], Rati Shukla[2], Amrita Jyoti[3], Ayushi Prakash[4], Mayur Rahul[5]

[1]*Department of Software Engineering, NIT Durgapur, WB, India*
[2]*GIS Cell, MNNIT Prayagraj, UP, India*
[3]*Ajay Kumar Garg Engineering College, Ghaziabad, UP, India*
[4]*ABES Engineering College, Ghaziabad, UP, India*
[5]*Department of Computer Application, UIET, CSJM University Kanpur, India*

***Corresponding Author:** Zaheer A. Siddiqui; E-mail: zaheercs0115@gmail.com

ABSTRACT- Matroids are the combinatorial structure and Greedy algorithmic methods always produces optimal solutions for these mathematical models. A greedy method always selects the option that looks best at each step of process of finding optimal solution. In other words, it selects a choice which is optimal choice locally in such a strategy that this locally chosen option may direct to a solution that will be globally optimal. It is true that while selecting locally optimal solution at each stage, Greedy algorithms may not always yield optimal solutions, but if we can transform an unknown problem into matroid structure, then there must be a greedy algorithm that will always lead optimal solution for that unknown problem. The range of solutions provided by Greedy is large as compared to the applicability of the Matroid structure. In other words the problems that can be translated into Matroid structure is proper subset of set of all problems whether Greedy algorithm produces optimal solution. Matroid structure thus ensures the global optimal solution one can obtain with help of Greedy approach. We study various logarithmic and linear hierarchical based mathematical models from divergence sources to maximize our information for research purposes. We analyze the time complexity and provide constrains over the upper/lower bounds in correspondence with the optimal (maximum/minimum) solution. We try to establish the relationship between the maximization of information divergences, the optimal-likelihood theory, and classified sharing is instituted. We propose integration of unknown rough sets to matroids in this paper. Particularly, we devise methodically the upper and lower tightening bounds on rough matroids which may expand up to the generic combinatorial matroid structure. The relationships are established by the upper and lower tightening bounds approximations of generalized combinatorial rough sets based on different interdependent relation sets, respectively. As we define the generalized lower/upper bounds for rough matroid, we define a new structure for lower/upper greedoid leading to generalization of the greedoid. Additionally, based on the new established relation, the generalized rough set also provides a theory of poset matroid.

General Terms: Combinatorial structure, Greedy algorithms, Posets, Unit-time-processor, Activities, Graphical Data structure, Tree, Forests, Time complexity, Amortized analysis.

Keywords: Optimization, Cyclicity, Graphs, Tree, Greedy approach, Sets, Subsets, Sorting, Penalties, Vertex, Edge, Graphs.

ARTICLE INFORMATION

Author(s): Zaheer A. Siddiqui, Rati Shukla, Amrita Jyoti, Ayushi Prakash, Mayur Rahul

Special Issue Editor: Dr. Vikash Yadav
Received: 10/04/2022; **Accepted**: 25/04/2022; **Published**: 22/05/2022;
e-ISSN: 2347-470X;
Paper Id: 0422SI-IJEER-2022-02;
Citation: 10.37391/IJEER.100213
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-10/ijeer-100213

This article belongs to the Special Issue on **Recent Developments in Communication Technology using Machine Learning Techniques**.

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

The Greedy algorithms are very effective and widely acceptable algorithm that are applicable on wide range of real time problems to produce optimal (Maximum/minimum) solution. Each step of Greedy algorithm makes a locally best available choice at that particular moment of calculation. In other words, this method select a locally best (optimal) choice in such a hope that the selected choice may lead to a globally optimal solution. Although greedy methods may not produces optimal solutions always but for a range of problems, Greedy methods do yield optimal results much efficiently in comparison to solution provided by dynamic programming for the same problem. A greedy algorithm produces an optimal solution to a given problem by selecting a best choice among a range of choices. Decisions are made by selecting the best available choices at each stage. We will discuss about a theory that underlies a combinatorial structures known as matroids in this section. A greedy algorithm will always yield an optimal solution for these combinatorial structures (matroids).

A matroid is defined as an ordered pair P = (S, ξ) that satisfies the following conditions-

1. The set S is a Finite.
2. The set ξ is a collection of nonempty subsets of S, called the ***independent subsets***, satisfying the property as if Y ∈ ξ and X ⊆ Y, Then X ∈ ξ. This property is called ***hereditary***.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 126-130 | e-ISSN: 2347-470X

3. If X ∈ ξ, Y ∈ ξ, and |X| ⊂ |Y|, then there must be some element a ∈ X – Y, then we will have X ∪ {a} ∈ ξ. And we say that P satisfies the ***exchange property***.

The sets in P are generally known as independent sets. We can state that if A is subset of any independent set of P, then A will itself be an independent set. The union of all sets in P is known as ground set. If an independent set X is not a proper subset of another independent set, then X is called a basis. The exchange property implies that every basis of a matroid will have the same cardinality for every independent subset of A [1-2].

2. BASIC CONCEPTS AND DEFINITIONS

2.1 Graph

A graph G = (Φ, E), consist of set of objects Φ = {v_1, v_2, v_3. . . v_n}, denoted as set of vertices and E = {e_1, e_2, e_3, . . . , e_m}, denoted as set of edges such that e_k is identified by an unordered pair (v_i , v_j) of vertices [3].

2.2 Subgraph

A graph G′ = (Φ′, E′) is said to be subgraph of a given graph G = (Φ, E), if all the vertices and edges in G′ are also in G. G′ is obtained by deleting some edges or some vertices or both from G. In In other words (Φ′, E′) ⊆ (Φ, E) [3].

2.3 Tree

A tree is defined as connected graph and does not have any cycles, or a tree is a connected acyclic graph. The edges of a tree are known as branches. It may be assure that a tree must to be a simple graph without self-loops and parallel branches forming loops [3]. One of the important classes of graphs is the trees. The importance of trees is evident from their applications in various areas, especially theoretical computer science and molecular evolution. The various kinds of data structures referred to as trees in computer science are similar to trees in graph theory, except that computer science trees have directed edge [3].

2.4 Properties

- A tree with n vertices has n edges.
- A minimally connected graph is a graph that gets disconnected if we remove any one its edge. Clearly, there is no cycles in a minimally connected graph
- A graph is a tree if only if it is minimally connected.
- A simple graph with n vertices, n-1 edges and no cycles is connected must be a tree.
- Any tree with at least two vertices has at least two pendant vertices.

3. GRAPHICAL MATROID: A DEMONSTRATION

Let us consider a graph G = (Φ, E), where Φ denotes the no of vertices and E denotes the no of edges is shown in *Figure 1*.

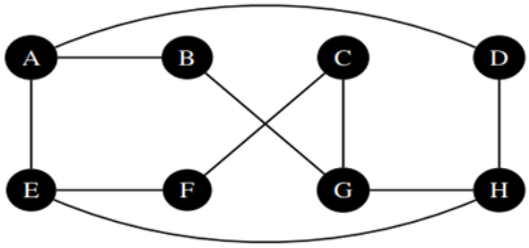

Figure 1: Sample Graph

The following are the specifications of the graph:

- The graph is Simple graph.
- It does not contain any self-loop and parallel edges.
- The Graph is strongly connected graph.
- It contains several circuits so it is not a tree.

The acyclicity and independent property of the matroid requires the given graph to be Tree. The graphical matroid is said to be independent if the graph does not contain any cycle. If there exits any cycle in the graph then independent property of the matroid is no longer preserved. A graph can be a tree if is connected. That is, if each of the nodes is connected with a link to at least one other node. If a node is not connected to some other node, then the assembly is not a tree so all the operation that involves the graphical matroid structure must be in acyclic and connected only form.

3.1 Matroid Structure

Theorem: If G = (Φ, E) is an undirected graph, then P_G= (S_G, I_G) is a matroid.

Proof: Lets Define P_G = (S_G, I_G) as a subset of original graph G = (Φ, E) as shown below in *Figure 2*.

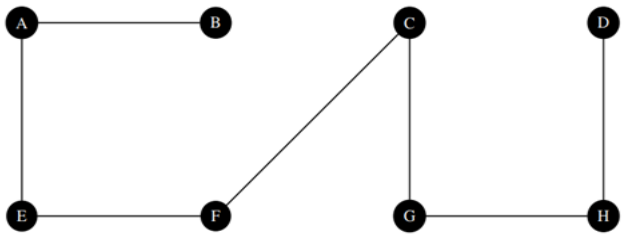

Figure 3: Reference Graph

1. Set S_G, we defined it on set of edges of graph, i.e. E.
2. If X is a proper subset of E, then X ∈ I_G iff X is acyclic. That is, a set of edges X will be independent iff the sub graph G_x = (Φ, X) produces a forest.
3. I_G is the family of all independent subset of S_G.

3.2 Verification of Matroid Structure

To prove that the given Structure satisfies the matroid Structure we have to the following terms

- Independent Structure
- Finiteness
- Exchange Property
- Hereditary Property

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 126-130 | e-ISSN: 2347-470X

3.2.1 Independent Structure

This is the fundamental and essential property that any matroid structure must have to satisfy. In case of Graphical matroid, the independent property is the acyclic graph. The reference graph $M_G = (S_G, I_G)$ is the acyclic graph. Whenever the acylicity of the graph is preserved, then matroid structure is also remain preserved.

3.2.2 Finiteness

- Graph G = (Φ, E) has six no. of vertices (finite).
- No. of edges is five (finite). Thus, it is a finite graph.
- We have defined S_G = E, so S_G will be finite.
- All the family of subsets of edges i.e. I_G will also be finite hence Independent.

3.2.3 Exchange Property

Let us consider the reference graph again in *Figure 3:*

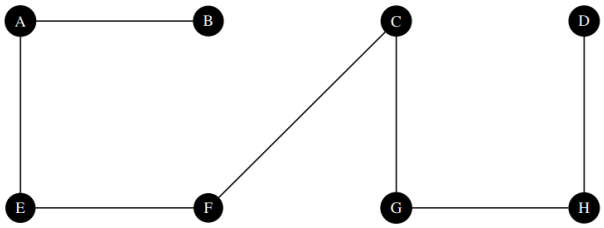

Figure 3: Reference Graph G_R = (Φ, R)

Let G_X = (Φ, X) (shown in *Figure 4*), and G_Y = (Φ, Y) are two independent graph of the reference graph. Consider |Y| > |X|.

1. The graph G_Y = (Φ, Y) is shown in *Figure 5*.
2. Number of components in this graph is three.
3. The cardinality of this graph is larger than the Graph G_X
4. Number of vertices in this graph is V same as reference graph.

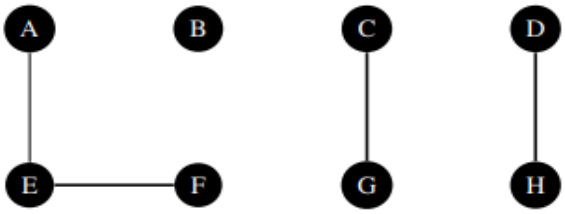

Figure 4: Graph G_X = (Φ, X)

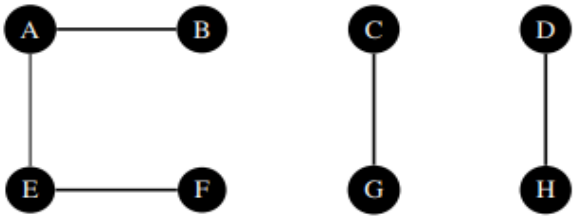

Figure 5: Graph G_Y = (Φ, Y)'

Consider an edge $e_k \in (Y - X)$ as shown in *Figure 6*

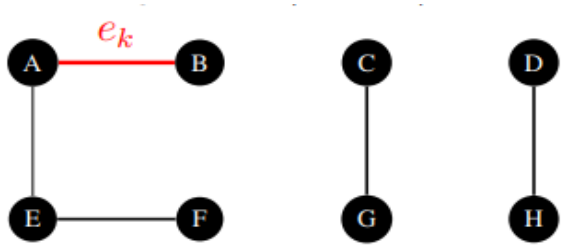

Figure 6: Graph G_Y = (Φ, Y)

[\Let us remove edge e_k from graph G_Y forming a new graph G'_Y = (Φ, Y′) as shown in *Figure 7*.

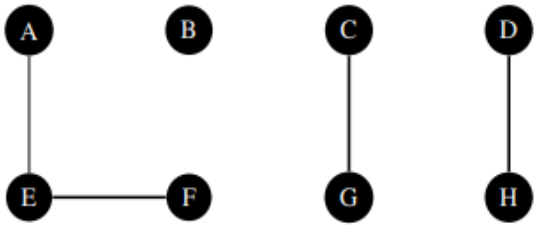

Figure 7: Graph G'_Y = (Φ, Y′)

Let us join the separated edge e_k in graph G_X = (Φ, X) forming a new Graph G'_X = (Φ, X′) shown in *Figure 8*.

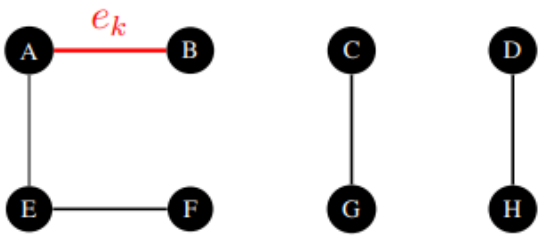

Figure 8: Graph G'_X = (Φ, X′)

The resultant graph G'_X is nothing but the original graph G_Y = (Φ, Y) itself. A forest is an undirected graph, all of whose connected components are trees. Now it is clear from the figure that if we connect two disjoint a cyclic component, and then the resulting graph will remain acyclic.

1. The resultant graph is still acyclic thus preserve the independent property.
2. Adding an edge between two different components is safe and does not create a cycle as long as components are itself acyclic.
3. Thus, two components are merged into single component producing a new component by preserving the original property.
4. Hence proving the ***Exchange Property.***

3.2.4 Hereditary Property

- Let A∈ IG is a subset of forest.
- Let B⊂A, which means no. of tree in B is less than that of A.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 126-130 | e-ISSN: 2347-470X

- Intuitively, removing of trees from a forest will leave it as forest again.
- Therefore, it satisfies ***independent property***.

Consider the reference graph $M_G = (S_G, I_G)$ as in *Figure 9*.

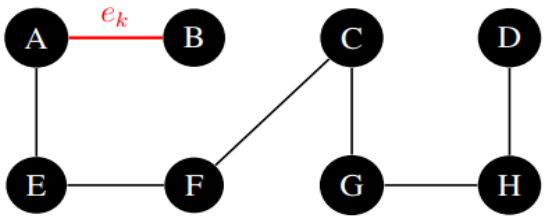

Figure 9: Connected Graph

Consider the edge e_k, let's remove the edge e_k from this graph. The resultant graph will look like as in *Figure 10*.

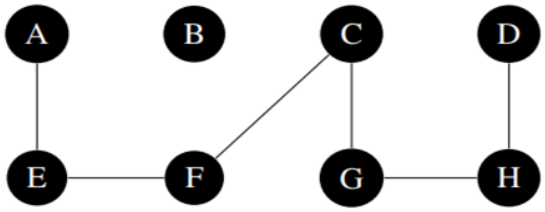

Figure 10: Disconnected Graph

Removal of e_k does not create a cycle in graph [3], rather it disconnect the graph, hence proving ***Hereditary Property***.

4. METHODOLOGY & RESULT ANALYSIS

4.1 A task-scheduling problem

In task scheduling problem, we optimally schedule single-unit tasks (activities) executing on a single threaded processor [1]. Here every unit-time task is provided with its deadline and an incurred penalty if the task fails to meet its mentioned deadline. A single-unit task is a process that required single time unit to complete when allocated on a single processor machine.

More formally, we are provided with following information

- A set of n unit time activities $S = \{a_1, a_2. \ . \ . \ a_n\}$.
- *n* deadline $d_1, d_2, \ . \ . \ . \ , d_n$, corresponding to each activity a_i.
- Inequality $1 \leq di \leq n$, is satisfied for each activity.
- A set of non-negative penalties $w_1, w_2. \ . \ . \ w_n$ incurred with each activity a_i, If it does not end by its deadline.

We can define the execution of activities of S in any of |S|! The sequence thus obtained is called schedule. An activity a_i finishes after its deadline then it will be late in a schedule. An activity a_i will be early in a schedule if it is finished before its deadline. We can always transform an arbitrary schedule into early-first form, in which the early tasks followed by the late tasks. Such schedule where all early task are followed by all late tasks are in form called ***canonical structure.*** A canonical structure can be translated in a Matroid structure and by this virtue; a canonical form of schedule will always generate an optimal solution [1]. The activities needed to be sorted in increasing order of their deadlines or decreasing order of their penalties. In either case, the theme of presented algorithm will not change. Here our goal in to minimize the penalties for the late tasks.

4.2 Algorithm

1. The task of scheduling the activities are highly dependent on verifying the independence of a give set A of activity.
2. For n activities, any schedule can take maximum time n unit.
3. The activities are sorted in non-decreasing penalties.
4. Activities can also be sorted in increasing of their deadlines, without affecting the algorithm and running time.
5. $N_t(A)$ denotes the number of activities in A that are finished by time t or before. For t =0, 1, 2 … n, we have $N_t(A) \leq$ t. This is call independent check.
6. By definition then $N_0(A) = 0$.

4.3 Time complexity

1. Sorting time = ***O(n log n)***
2. Performing 'n' independent check, each check takes ***O(n),*** total time for check = ***$O(n^2)$***
3. Total time taken (Time complexity) = ***O (n log n) + $O(n^2)$ = $O(n^2)$***

4.4 Task-scheduling problem with matroid and fast disjoint- set forests

(a)

1. The operation OPR-MAKE-SET(x) returns a pointer to the element 'x' that is newly made set of its own [6].
2. The operation OPR-FIND-SET(x) tracks the sequence of parent pointers starting from a queried node 'x' extended up to an element on top (root node).
3. The operation Union(x, y) replaces the set containing 'x' and the set 'y' resulting a new set with their union.
4. The height of the tree and linear time complexity can be achieved by executing two operations in parallel named as **OPR_RANK_UNION** and **OPR_COMPRESSION_PATH.**

(b)

1. Each of these algorithm will take ***O(m log n)*** times when executed individually. Where 'n' is number of operation and 'm' is the height of the forest without path compression.
2. The Graphical data structures always performs well when several operations are carried out in parallel.
3. In our algorithm, we calculate a different version time complexity called ***Amortized analysis.***
4. The amortizes analysis performs well when the input size is large.
5. In the algorithm depicted below, we executed ***Rank Union and Path compression*** operation parallel.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 126-130 | e-ISSN: 2347-470X

4.5 Pseudocode for the Faster Algorithm

```
FAST-DISJOINT-SCHEDULING(A)
    let D[1..n] be a newly created array
    excecute k (from 1 till n)
     ak.ET = ak.DL
     if D[ai. DL] != φ
        p = OP-FIND-SET(D[ak.DL])
        q = OP-FIND-SET(D[ak.ET])
        a[k]. ET = q.down - 1
     p = OP-MAKE-SET{ak}
     D[ak.ET] = p
     p.down = p.up = ak.ET
     if D[ak.ET - 1] != φ
        UNION{D[ak.ET-1, D[ak.ET]}
     if D[ak.ET+1] != φ
        UNION{D[ak.ET], D[ak.ET+1]}
```

4.6 Time complexity Analysis

1. Sorting time = ***O(n log n)***
2. The constant factors and overhead for initializing and maintaining Graph data structure is high in comparison with linear data structures implemented via arrays.
3. Performing OPR_MAKE_SET ***O(n)***, UNION ***O(n)*** & OPR_FIND_SET ***O(n), total 3n*** operations, where each may occur at most ***n*** times will be ***O(n* α (n)), where α (n) is a very slowly growing function.***
4. The function ***O(α(n))*** is practically highly slow growing function and for all practical purposes ***O(α (n))*** ≤ 4.
5. Time complexity =***O(n α (n)) +O(n log n) = O(nlogn).***

5. CONCLUSION

The result of Task-scheduling algorithm over Modified Task-scheduling algorithm is depicted by achieving lower time complexity by the later algorithm. The proposed algorithm will perform well and more effective in comparison finding solution via canonical form when the input size of the problem is fairly large. Although the proposed algorithm may not work well for small size input due hidden constants and overheads, but it will outrun the other algorithm once the size in fairly enough. The time comparison is depicted in the graph chart below in *Figure 11*.

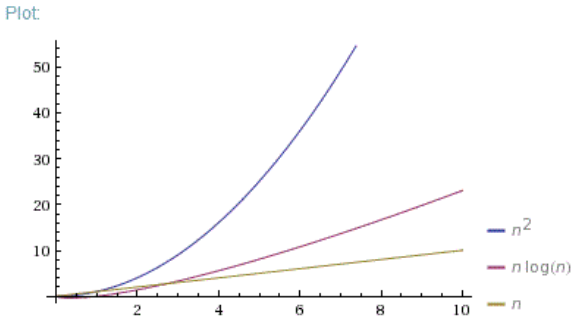

Figure 11: Cost Comparison of different Algorithms

6. ACKNOWLEDGMENTS

Our thanks to the experts, who have contributed towards development, provided guidance and reviewed the Paper.

REFERENCES

[1] Introduction to Algorithm: Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest and Clifford Stein.
[2] Ghanshyam Prasad Dubey, Dr. Rakesh Kumar Bhujade (2021), Investigating the Impact of Feature Reduction Through Information Gain and Correlation on the Performance of Error Back Propagation Based IDS. IJEER 9(3), 27-34. DOI: 10.37391/IJEER.090302.https://ijeer.forexjournal.co.in/archive/volume-9/ijeer-090302.html
[3] Avadhesh Kumar Dixit, Rakesh Kumar Yadav and Ramapati Mishra (2021), Contrast Enhancement of Colour Images by Optimized Fuzzy Intensification. IJEER 9(4), 143-149. DOI: 10.37391/IJEER.090408.
[4] Fundamental of Computer Algorithm: Ellis Horowitz and Sartaj Sahani.
[5] Introduction to Graph Theory: N. Narsingh Deo.
[6] Rough matroid, IEEE Coference paper 2011: William Zhu, and Shiping Wang, Lab of Granular Computing Zhangzhou normal University, Zhangzhou 363000, China.
[7] Reducible Matroid and Reducible Element of Covering-based Rough Sets, IEEE Conference 2010 : Chengyi Yu, Fan Min, William Zhu, Lab of Granular Computing, Zhangzhou Normal University, Zhangzhou 363000, China.
[8] "http://en.wikipedia.or