An Introduction to Matroids and Greedy Algorithms

Woody March-Steinman

12/17/2018

Contents

1 Definitions and Background

This article is meant to be a review of matroid theory within the domain of the application of greedy algorithms.

1.1 Matroids

Across a number of domains, structures related to the idea of independence between members of a set is a fundamental part of evaluating a problem.

While initially constructed in 1935, the refinement of the definition of a matroid (from several basic definitions to the formulation of the finitary matroid to the construction of the definition of an infinite matroid in 2013) occurred over a relatively long period of time.[5]

Ultimately, a Matrix can defined in several ways, all of which are equivalent.

Definition 1. A finite Matroid is a pair (E, I) where E is a finite set and I is a collection of subsets of E called the independent sets. These two sets have the following properties [5]:

1. The empty set $\{\}$ is independent. That is, $\{\} \in I$ and $\{\} \subseteq E$
2. Every subset of an independent set is independent (The Hereditary Property)
3. if $A, B \in I$ and A has more elements than B, then there exists $x \in A$ such that $x \notin B$ such that $\{x\} \cup B \in I$ (The Augmentation Property)

As can be seen from this definition, a matroid is a finite set combined with a set of independent subsets of that set. The hereditary property implies that each subset of an independent set is independent (an intuition we can gather from the idea of linearly-independent lists). The Augmentation Property declares that any smaller independent set in I can be augmented with an element from a larger set in I such that it is still independent (given that the added element is not already in the smaller set).

As a side note, the first two conditions can represent an abstract simplicial complex (useful in modern data analysis), while the addition of the augmentation property allows us to consider sets of vectors as matroids. Indeed, we can also construct submatroids from any set in I. This can allow us to build solutions from considering independence relationships in submatroids using recursion, which seems akin to Kruskal's algorithm for finding the minimum spanning tree of a graph (more information below).

We can also construct the definition of a finite matroid from the concepts of basis, closure, and rank.

Definition 2. A basis for E is a maximal independent subset. That is, it is an independent set such that it becomes dependent with the addition of any other element from E.

Definition 3. A circuit of E is a minimal dependent subset. That is, it is a dependent subset of E such that the removal of any element from the set results in an independent set.

Conveniently, we can determine any of the independent sets, bases, dependent sets, and circuits of a matroid given one of the other sets. With the two definitions of circuits and bases, we most move forward to discuss graphs for an excellent visual representation of these concepts.

1.2 Graphs

In order to fully appreciate the utility of the matroid structure to the efficiency of greedy algorithms, it is important to first understand basic elements of graph theory. Graph theory has wide-ranging applications from the representation of computer networks to connectedness between neurons.

As a spark for graph theory, we come to an example of the seven bridges of Königsberg problem solved by Leonhard Euler in 1736 [8]. Suppose there exist several islands, all connected by bridges. We can treat a set of islands as destination points and the bridges as paths connecting one island to another. A question that might arise would be whether one can visit all islands while crossing each bridge between them exactly once. Euler's solution indicates that this is only possible for certain configurations of islands and bridges, which we'll call here nodes (islands) and edges (bridges). There must be exactly zero or two nodes with an odd number of edges built to them in order for the above walk to be possible. The bridges of Königsberg, represented by nodes and edges, demonstrates that certain problems can be solved through an analysis of connectivity alone, regardless of distance or cost or other considerations.

We can also consider graphs where each "bridge" between islands has some associated cost. We can consider the bridges and tunnels associated with Manhattan to be an example of such a weighted graph when we associate a toll. In fact, the cost of moving from one node to another in the graph differs depending on whether one is exiting or entering the island. One could additionally define a distance cost or time cost to travelling across certain bridges or tunnels. This type of graph is weighted, as each edge has additional cost or value to it, and it is also directed, as traffic can only flow in certain directions (and cost differs with direction).

Let us now come to a formal definition of a graph that will be useful for further discussion:

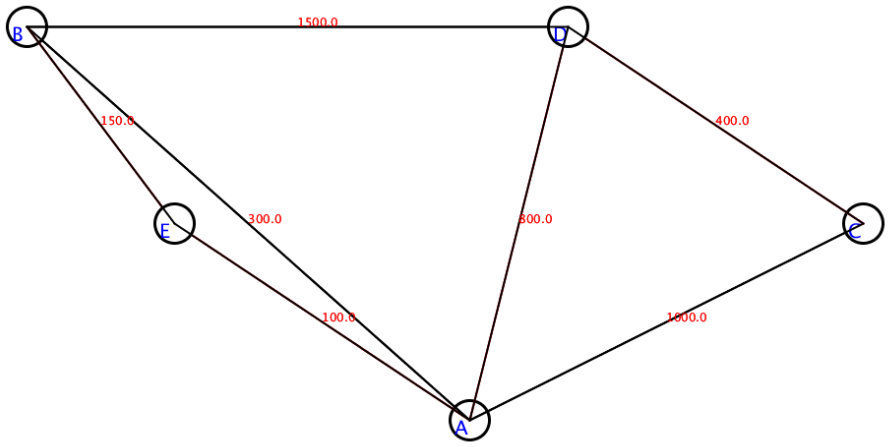
Definition 4. A graph is a collection of vertices (or nodes) V with associated edges E. A graph can be represented by a pair (V, E). An edge consists of a subset of two vertices. In an undirected graph with two vertices $v_1, v_2 \in V$, $v_1, v_2 \in E$ is equivalent to $v_2, v_1 \in E$. This is not the case for directed graphs, where the elements of E are ordered pairs (v_1, v_2). [10].

Remark 1. As can be seen by this definition, we have some set V of vertices

which define edges for simple graphs. If the set of edges defined (each necessarily of length two) is an independent set or a basis set, then the graph notation directly matches that of a matroid. This will be further expanded upon in the next section. It is important to note that directed graphs are outside of the scope of this discussion.

We can say that a walk is a finite list of edges traveled. A path along the edges of a graph is a walk where the same vertex is not "visited" twice. We can define a circuit to be a minimum path that returns to a vertex.

Below is a graph of cities and weights of paths between them (one can imagine weights to be generated by a flight cost function):



Labeling each one of the vertices in this fashion:

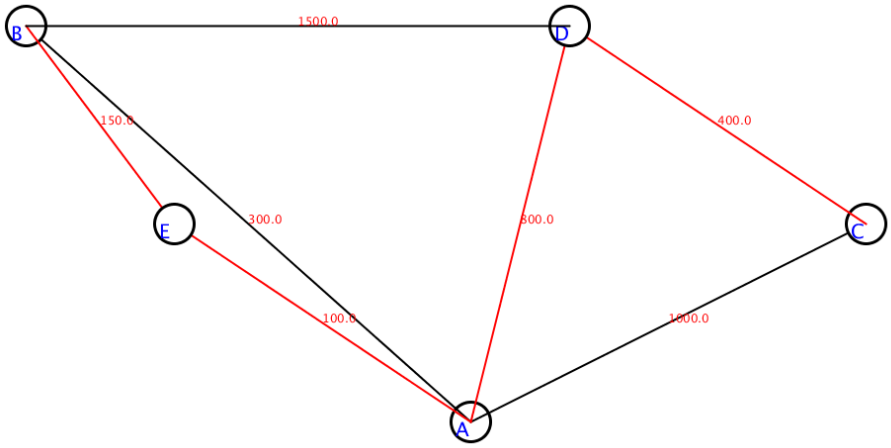
1. New York A
2. Chicago B
3. Rome C
4. Paris D
5. Pittsburgh E

We can create an adjacency matrix to represent the graph above, with each entry at i, j representing the edge weight between vertex i and vertex j. If an edge does not exist, the weight is zero:

$$\begin{bmatrix} 0 & 300.0 & 1000.0 & 800.0 & 100.0 \\ 300.0 & 0 & 0 & 1500.0 & 150.0 \\ 1000.0 & 0 & 0 & 400.0 & 0 \\ 800.0 & 1500.0 & 400.0 & 0 & 0 \\ 100.0 & 150.0 & 0 & 0 & 0 \end{bmatrix}$$

The fact that we can represent the graph above as a matrix strongly implies that there exists an independence relationship in the land of graphs, as there is a nice linear map from any graph to a matrix. This gives a sense that it should be possible to find independent edges or paths in a graph. Negative weights for some edges might imply a directed graph, which is easy to represent using this form of adjacency matrix.

Let's consider a subset M of connected edges of a weighted, undirected graph that contains no cycles (a tree) and connects all vertices of the graph (spanning). M is called a minimum spanning tree if is the subset of this kind with minimum weight, and a maximum spanning tree if it is the subset of this kind with maximum possible weight. Below is the minimum spanning tree for the graph above (edges marked in red):



An algorithm used to find this type of tree, Kruskal's algorithm [9], is a greedy algorithm that produces precisely a minimum spanning tree from any connected graph. Generally, this set of edges is a graph that is independent, as it contains no cycles. A minimum (or maximum) spanning tree fits in as a basis by the criterion of definition 2.

1.3 Greedy Algorithms

This paper largely concerns the applications of matroid theory to the utility of greedy algorithms. A greedy algorithm will take a locally-optimal choice at every point. In a graph, this can be represented by taking edges of lowest or of highest weight to traverse between two nodes, depending on a goal in question. Let us use the above weighted, directed graph to describe a transit to Queens from New Jersey. We make the choice based on time efficiency, not toll cost. By evaluating the best route using an algorithm like Dijkstra's Shortest Path algorithm [11] and choosing lowest-cost roads for each point, we can determine the optimal path for transit.

An example where a greedy algorithm is not always the right choice comes from chess. If we treat each move as weighted based on the value of each piece, a greedy algorithm might see an open queen and capture it immediately, possibly letting a player be lured into a clever checkmate due to the sacrifice. There are also trade-offs with efficiency, where a greedy algorithm might evaluate the correct path only after nearly-infinite computation time depending on the complexity of a transit system.

A greedy algorithm takes the best path at each opportunity. When applied to the traversal of a graph from node B to node C, the path taken by a greedy algorithm would be B, E, A, D, C as each edge along the way is the best choice for transit (an edge with the lowest cost at that point).

An example of a generic greedy algorithm acting on a graph to take the lowest-cost path:

```
current_node <- start_node
    while (current_node != node_goal)
        low_cost = edge_0_cost
        new_node = edge_0_destination
        for (each other edge n at current node)
            if (low_cost > edge_n_cost)
                low_cost <- edge_n_cost
                new_node <- edge_n_destination
        current_node <- new_node
```

2 Matroids and Greedy Algorithms

2.1 Relation to Greedy Algorithms

In this section, we aim to prove that there exists a relationship between the ability to represent a problem as a matroid and the effectiveness of a greedy algorithm. To be more explicit, a problem can be represented as a matroid if and only if the perfect solution is a greedy algorithm. In general, it is easy to see some obvious parallels between the language for matrices and the language for graphs. Graphs can contain cycles, or paths that lead back to a starting vertex. Graphs can also contain a sense of independence: a list of edges from one vertex to another that does not create a cycle is obviously independent. We can also generate some idea of a basis of a graph, in that the basis can be something like the minimum spanning tree. We can develop something like a basis through an algorithm like Kruskal's algorithm (or Prim's algorithm).

With reference to the previous problem to find a minimum spanning tree:

Example 1. Suppose we have a matroid $M = (E, \mathcal{I})$ and with a cost function $w : E \to \mathbb{R}$. Kruskal's algorithm asks for us to start with $I = \emptyset$, or a minimal submatroid of the given graph. Add the element from E with the lowest cost $e \in E$ such that $w(e) < w(f), f \in E, f \neq e$. The requirement here is that $I \cup e \in \mathcal{I}$. Repeat until we have a basis (via definition 2 above).

We can prove that for any cost function this algorithm produces a basis of minimum cost.

Proof. Let us have $I = i_1, ..., i_n$ with cost $w(i_1) + ... + w(i_n)$ and the minimum basis be $K = k_1, ...k_n$ with cost $w(k_i) + ... + w(k_n)$. Now suppose that the total weight of I is greater than the total weight of K. Assuming that I and K are both independent by definition, then there exists some edge in K such that we can switch some edge in I with that edge in K such that the list remains independent (this is possible for each vector in both sets). Suppose this vector is of lower cost than the one it replaces. But this is contradictory, as the algorithm is defined to have chosen the lowest cost edge. □

[12] A more general proof for matroids can be found on [12] on page 8.

References

[1] Maurer, S. Matroid Basis Graphs. JOURNAL OF COMBINATORIAL THEORY (B) 14, 216-240 (1973)

[2] Hillman, H. Matroid Theory. Accessed 11/26/2018 from https://www.whitman.edu/Documents/Academics/Mathematics/hillman.pdf

[3] Multiple Authors. Matroids. Accessed 11/26/2018. https://en.wikipedia.org/wiki/Matroid

[4] Locke, S. C. Greedy Algorithms. 1996. Accessed 11/26/2018. http://euler.math.fau.edu/locke/Greedy.htm

[5] Meretzky, D. Matroids Lecture, Hunter College. 11/21/2018.

[6] Edmonds, J. Mathematical Programming (1971) 1: 127. https://doi.org/10.1007/BF01584082

[7] Sporns, O. Graph Theory Methods for the Analysis of Neural Connectivity Patterns. Accessed 11/26/2018. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.201.6514&rep=rep1&type=pdf

[8] Multiple Authors. The Seven Bridges of Königsberg. Accessed 11/26/2018. https://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

[9] Kun, J. When Greedy Algorithms are Perfect: The Matroid. August 26, 2014. Accessed 11/26/2018. https://jeremykun.com/2014/08/26/when-greedy-algorithms-are-perfect-the-matroid/

[10] Multiple Authors. Graph Theory. https://en.wikipedia.org/wiki/Graph_theory

[11] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[12] http://math.sfsu.edu/federico/Clase/Matroids/LectureNotes/lectures1-25.pdf