Matroids and Greedy Algorithms

Siddharth Mahendraker

August 11, 2013

Abstract

In this paper, we examine the necessary and sufficient conditions for a greedy algorithm strategy to be successful for a given combinatorial optimization problem. This is done through the study of matroids and their properties. We show that a greedy algorithm is successful for a given problem if and only if the problem has can be formulated as a matroid.

Contents

1 Introduction

Combinatorial optimization is a mathematical framework for finding optimal objects amongst a large set of objects. A famous example of a combinatorial optimization problem is the travelling salesman problem, or TSP. Assume a salesman has a list of cities he wants to visit, and knows the distance between various pairs of cities. The TSP asks: What is the shortest path which takes the salesman through each city once and back to his starting point? Here, the optimal object is a shortest path which passes through each city once and returns the salesman to his point of departure, and the set of objects is the set of all paths which can be taken. To see why this problem could be difficult, assume that every city is connected to every other city and we want to check all the paths to find the shortest one. If we fix a city of departure, there are $n-1$ cities left to go to, where n is the total number of cities. Each of these choices represents a different path the salesman could take. In the next city, there are $n-2$ cities left to go to, and so on. It turns out, given n cities there are $(n-1)!/2$ unique paths which reach every city and return the salesman to his point of departure. Even for relatively small values of n, the number of possible paths is enormous; far larger than anything even a computer could hope to check. Consider for example the choice of $n = 64$. In this case, there are over 10^{86} possible paths to check! More paths than there are atoms in the universe! Using combinatorial optimization techniques, however, we can significantly reduce the amount of work needed to find an optimal path. Indeed, to date, instances of the TSP have been solved with tens of thousands of cities.

One particular strategy often used in combinatorial optimization is the greedy optimization strategy, or greedy algorithm. The greedy algorithm attempts to find the optimal object by making successive locally optimal decisions, in the hopes that this will lead to a globally optimal solution. For example, a greedy algorithm for the TSP could work as follows: Fix the city of departure. To reach another city, pick the shortest route out of the current city which leads to an unvisited city. Repeat until you return to the city of departure. Unfortunately, greedy strategies are often non-optimal. In the case of the TSP, this particular greedy algorithm will not yield the optimal path, and in certain situations, can in fact give the worst possible solution. More generally, it is known that greedy algorithms are far from optimal for most instances of the TSP [2].

However, for certain combinatorial optimization problems, greedy algorithms can be very useful. If they can be proven to be correct, greedy algorithms are often far simpler to implement than other combinatorial optimization strategies, such as dynamic programming. Furthermore, they often

consume less resources when implemented on a computer and thus run faster and more efficiently. Therefore, knowing when a greedy algorithm can be applied is very useful.

In this paper, we examine the necessary and sufficient conditions required for a greedy algorithm strategy to be successful for a given combinatorial optimization problem. This is done through the study of matroids and their properties, which as we shall soon see, lead naturally to greedy algorithm strategies.

The remainder of this paper will proceed as follows. First, we give an example of an successful greedy algorithm by examining the minimum spanning tree problem and Kruskal's algorithm. Next we introduce matroids and discuss their various forms and properties. Then we link certain properties of matroids to solutions to a general class of combinatorial optimization problems. Finally, we show that these solutions can be obtained through greedy stratagies.

2 The MSTP and Kruskal's Algorithm

Another well known problem in combinatorial optimization is the minimum spanning tree problem, or MSTP. Given a weighted graph $G = (V, E, w)$, the MSTP asks us to find an acyclic subgraph $H = (V, E', w)$ of G, such that H contains all the vertices of G and the weight of H, $w(H) = \sum_{e \in E'} w(e)$ is minimal.

Unlike the TSP, the MSTP can be solved with a simple and elegant algorithm, due to Kruskal [1].

Definition 2.1 (Kruskal's Algorithm). Given a graph $G = (V, E, w)$, the minimum spanning tree T^* of G can be computed as follows:

1. Begin with an empty set $T := \emptyset$.
2. Until $|T| = |V| - 1$, repeat steps 3 and 4 below.
3. Select a minimum cost edge e in $E - T$, breaking ties arbitrarily.
4. If $T \cup e$ is acyclic, set $T := T \cup e$. Otherwise discard the edge e and never consider it again.
5. Output a minimum spanning tree $T^* = (V, T, w)$.

Note that due to our frequent addition and removal of single elements from sets, we abbreviate $X \cup \{e\}$ to $X \cup e$ and $X - \{e\}$ to $X - e$.

Clearly, Kruskal's algorithm is greedy. At every step, the algorithm selects a minimum weight edge in G, and attempts to add it to the edge set of the output spanning tree.

Definition 2.2. An algorithm is called a *greedy algorithm* if it makes a locally optimal decision at each step.

Kruskal's algorithm shows how efficient greedy algorithms can be, given the correct problem structure. Intuitively, Kruskal's algorithm works because given any subset S of the vertex set of a graph G, an MST of the subgraph induced by S will always be a part of an MST of G. Furthermore, if an edge e is a minimum weight edge across all edges, it must be part of some MST of G.

In the next section we introduce matroids, which generalize certain properties of the MSTP mentioned above. We will revisit Kruskal's algorithm in subsequent sections and provide a more rigorous proof of correctness using matroid theory.

3 Matroid Theory

Intuitively, matroids consolidate various ideas of dependence which arises naturally in certain fields of mathematics, particularly linear algebra and graph theory. In linear algebra, a set of vectors U of a vector space V is called dependent if there is a linear of combination of these vectors which sum to 0. In graph theory, a set of edges S of a graph G is called dependent if the edges of S contain a cycle.

An interesting property of matroids is that there are several equivalent axiom systems which can be used to reason about them. In this essay, we will only explore two such axiom systems: the independent set system, and the bases system. For a more thorough overview of matroid theory, see [3].

Definition 3.1. A *matroid* M is an ordered pair $(E, \mathcal{I})$, where E is a finite set and $\mathcal{I}$ is a collection of subsets of E satisfying the following three conditions:

(I1) $\emptyset$ is in $\mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.

(I3) If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there exists an element $e \in I_2 - I_1$ such that $I_1 \cup e \in \mathcal{I}$.

We call E the *ground set* of a matroid $M = (E, \mathcal{I})$ and we call the members of $\mathcal{I}$ the *independent sets* of M. A subset S of E which is not in $\mathcal{I}$ is called a *dependent set*. We also call (I2) the *hereditary property* and (I3) the *exchange property*.

From this definition, we can already intuitively see that matroids encapsulate lots of structure which is important in solving combinatorial optimization problems. Specifically, (I2) and (I3) are reminiscent of the overlapping subproblem structure and optimal substructure exploited by dynamic programming algorithms [1]. That is, we are guaranteed that all subsets of an independent set are themselves independent, and that we can always build larger independent sets from smaller ones.

Proposition 3.2. *Let $G = (V, E)$ be a graph and let $\mathcal{I}$ be the set of subsets of E which do not contain a cycle. Then $M = (E, \mathcal{I})$ is a matroid.*

Proof. By definition, $\emptyset \in \mathcal{I}$ and thus $\mathcal{I}$ satisfies (I1). It is evident that if $H \in \mathcal{I}$ and $K \subseteq H$, then $K \in \mathcal{I}$. Thus $\mathcal{I}$ satisfies (I2). We show that $\mathcal{I}$ satisfies (I3) by contradiction. Suppose I_1 and I_2 are elements of $\mathcal{I}$, and $|I_1| < |I_2|$. Assume that for all $e \in I_2 - I_1$, the set $I_1 \cup e \notin \mathcal{I}$. This means that the set $I_1 \cup e$ contains a cycle for all e. Since I_1 is itself acyclic, there is only one cycle in $I_1 \cup e$. Now, consider $I_1 \cup I_2$. This set has exactly $|I_2 - I_1|$ cycles, one for each $e \in I_2 - I_1$. Let I_3 be a maximal independent set in $I_1 \cup I_2$, obtained by removing an edge from each of the $|I_2 - I_1|$ cycles in $I_1 \cup I_2$. Now,

$$\begin{aligned} |I_3| &= |I_1 \cup I_2| - |I_2 - I_1| \\ &= |I_1| + |I_2| - |I_1 \cap I_2| - |I_2 - I_1| \\ &= |I_1| + |I_2| - |I_1 \cap I_2| - (|I_2| + |I_1 \cap I_2|) \\ &= |I_1|. \end{aligned}$$

It follows that $|I_3| < |I_2|$, contradicting the maximality of I_3. Therefore, (I3) holds, and M is indeed a matroid. □

A matroid derived from a graph G is called a *cycle matroid* and is denoted $M(G)$. Furthermore, given a subset S of the vertex set of a graph G, we write $G[S]$ to denote the subgraph induced by S.

Consider a collection $\mathcal{B}$ of maximal independent sets of a matroid M. Given the ground set E of M we can easily reconstruct the independents sets of M as follows: The set X is in $\mathcal{I}$ if and only if $X \subseteq B \in \mathcal{B}$. We call a maximal independent set of a matroid a *base* or *basis*.

From our discussion of MSTs and Proposition 3.2 we can deduce that any spanning tree of a graph G is a basis of the cycle matroid $M(G)$. Notice

that this idea of a basis coincides with our idea of a basis from linear algebra. Like bases of a vector space, all spanning trees are equicardinal. Furthermore, if one "fills in the cycles" we can imagine that a spanning tree "spans" the graph, just as the span of a basis spans a vector space.

Proposition 3.3. *Let $\mathcal{B}$ be a collection of bases of a matroid $M = (E, \mathcal{I})$. Then the elements of $\mathcal{B}$ are equicardinal.*

Proof. Let B_1 and B_2 be elements of $\mathcal{B}$. Suppose $|B_1| < |B_2|$. Then by (I3), there exists a element $e \in B_2 - B_1$ such that $B_1 \cup e \in \mathcal{I}$. However, this contradicts the maximality of B_1. Therefore, $|B_1| \geq |B_2|$. Similarly, $|B_2| \geq |B_1|$. Thus $|B_1| = |B_2|$. Since B_1 and B_2 were chosen arbitrarily, all elements of $\mathcal{B}$ are equicardinal. □

The bases of a matroid M also have a very attractive property, similar to the exchange property for independent sets, called the *basis exchange property.*

Proposition 3.4. *Let $\mathcal{B}$ be a collection of bases of a matroid $M = (E, \mathcal{I})$. If B_1 and B_2 are elements of $\mathcal{B}$ and $x \in B_1 - B_2$, then there exists an element $y \in B_2 - B_1$ such that $(B_1 - x) \cup y \in \mathcal{B}$.*

Proof. Consider $B_1 - x \subset B_1$. Then $B_1 - x$ is independent by (I2). Now, $|B_1 - x| < |B_2|$ by Proposition 3.3. Thus, there exists an element $y \in B_2 - B_1$ such that $(B_1 - x) \cup y \in \mathcal{I}$ by (I3). Since $|(B_1 - x) \cup y| = |B_1|$ and all bases are equicardinal, there cannot exist an independent set containing $(B_1 - x) \cup y$. Thus $(B_1 - x) \cup y \in \mathcal{B}$. □

In the next section, we link the bases of matroids to solutions to a general class of combinatorial optimization problemes. We then explore solutions to to these problems using greedy strategies.

4 Greedy Algorithms and Matroids

Let us now define more precisely what it means for a problem to be a combinatorial optimization problem. As we mentioned earlier, combinatorial optimization problems deal with finding an optimal object among a large set of objets.

Definition 4.1. Let $\mathcal{I}$ be a collection of subsets of a set E. Define w as a function from E to $\mathbb{R}$, where

$$w(X) = \sum_{x \in X} w(x)$$

for all $X \subseteq E$. We call $w(X)$ the *weight* of X.

An *optimization problem* for a pair $(\mathcal{I}, w)$ is the problem of finding a maximal element S of $\mathcal{I}$ such that S has minimal weight.

We call the a maximal element with minimum weight a *solution* to an optimization problem.

It is easy to see how this definition encapsulates our intuition about combinatorial optimization problems. Because $\mathcal{I}$ is a collection of subsets of a set, it is plausible that $\mathcal{I}$ could become exponential in E, and thus very large. Specifically, in the worst case, if $\mathcal{I} = \mathcal{P}(E)$ then $|\mathcal{I}| = 2^{|E|}$, which grows exponentially in the size of E.

For example, the TSP corresponds to the optimization problem where $\mathcal{I}$ is a collection of subsets of the edge set of a graph $G = (V, E, w)$ which form a circuit, and w is simply the weight function of G. In this optimization problem, a maximal element of $\mathcal{I}$ would be Hamiltonian circuit, and a solution to $(\mathcal{I}, w)$ would be a Hamiltonian circuit of minimal weight.

5 Conclusions and Further Discussion

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 55 Hayward Street, Cambridge, Massachusetts, third edition, 2009.

[2] G. Gutin, A. Yeo, and A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for TSP. *Discrete Applied Mathematics*, (117):81–86, 2002.

[3] James G. Oxley. *Matroid Theory*. Oxford University Press, New York, 1992.