Matroids and Greedy Algorithms

Siddharth Mahendraker

June 17, 2013

Abstract

This is the abstract.

Contents

1.0 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 traveling salesman problem, or TSP. Assume a salesman has a list of cities he wants to vist, and knows the distance between each city. The TSP asks: What is shortest path which brings the salesman through each city and back to his starting point? Here, the optimal object is an optimal path which passes through each city 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 each path to see if it is optimal. If we fix a city of departure, there are $n-1$ cities left to choose from, where n is the total number of cities. Each of these choices indicates a different path which could have been taken. In the next city, there are $n-2$ cities to choose from, and so on. It turns out that given n cities, there are $\frac{(n-1)!}{2}$ 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 choices in enourmous. Consider for example the choice of $n = 64$. In this case, there are over 10^{86} possible paths to choose from! 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 succesive locally optimal decisions. For example, a greedy algorithm for the TSP could work as follows: Fix the city of departure. To get to another city, pick the shortest route out of the current 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 fact in certain situations, can give a worse path than a random algorithm which returns a random path.

However, greedy algorithms are still useful. If they can be proven to be optimal, 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.

2.0 Matroid Theory

3.0 Greedy Algorithms and Matroids

4.0 Optimality

5.0 Example: Kruskals Algorithm

References