Exponential Algorithms

Mark Watson

***Abstract*—This document describes various NP-Complete algorithms.**

***Index Terms*—algorithms, np-complete**

I. Introduction

THIS document covers five different problems that are NP-complete.

II. Problems

I'll cover the following problems that have been shown to be classified as NP-complete problems.

1) Knapsack Problem
2) Traveling Salesman Problem
3) Subset Sum Problem
4) Subgraph Isomorphism Problem
5) Graph Coloring

A. Knapsack Problem

The knapsack problem is a simple optimization problem trying to maximize to value of items included in a storage container with limited space. Given a finite list of items, each item has a different value and size. The goal is to find the combination of items that has the highest value while still fitting in the given space allocation. For example, the storage space could be a bugler's bag, and the items could be the belongings of some person being robbed. The knapsack problem is related to the subset sum problem, which will be discussed shortly.

Given this overview, we can describe the problem in more concrete terms; given a bag of size W, a number of item values, v_i, and a number of item weights, w_i, we need to find a set of items, $\{p_1, p_2, ..., p_j\}$ that maximizes v_i while keeping sum of the items less than or equal to W. In the algorithm I'm discussing here there is only one of each item. The problem can also be generalizing to have an infinite amount of each item, which is known as the unbounded knapsack problem. The bounded knapsack problem can be solved using dynamic programming. The unbounded problem is more difficult. There are several approximation algorithms that can be used to solve the unbounded problem. One possible solution is a greedy algorithm proposed by [1]. This algorithm is far from optimal at solving the bounded problem.

Another way of optimizing the problem is to use dominance relations to throw away items we don't need. This applies only to the unbounded version of the problem.

B. Traveling Salesman Problem

The traveling salesman problem is a graph traversal problem. Given a non-directed, weighted graph the goal is to find the shortest Hamiltonian tour visiting each node once. The graph is fully connected with each node visiting every other node. The problem can be thought of as a set of cities with a traveling salesman trying to find the optimal tour between them. The problem has applications in several areas like planning and logistics[1].

The problem is NP-complete, so finding a solution using brute force doesn't work. Approximate solutions involve using iterative heuristics that start with a basic solution and then get better and better results. A starting algorithm could be a greedy nearest neighbor. Optimization involves looking at nodes that are close to each other and then swapping them. There are other approximation algorithms such as genetic algorithms or ant farms to try to obtain the optimal solution, but the Lin–Kernighan–Johnson or V-opt heuristics are state of the art.

C. Subset Sum Problem

The subset sum problem is simply given a set of numbers is there a subset that adds up to zero. The problem is NP-complete and requires finding all subsets to solve in a brute force manor. Using a divide and conquer method to solve for each half at once to change the algorithm into $O(2^{N/2})$, although it is still exponential.

There are two other ways to solve the problem as well. The first is a non-approximate solution using dynamic programming. It's order O(P - N), where P is the sum of the positive values and N is the sum of the negative values. This means the algorithm isn't really polynomial time [2]. The other algorithm is a polynomial time approximate algorithm. The algorithm doesn't always give the exact result, but it doesn't take forever to run since it's polynomial time.

D. Tetris

In the game of Tetris there are a few NP-complete problems [3]. One of which is maximizing the number of pieces played before a loss, given a preset sequence of pieces. To find the optimal solution every possibility must be enumerated and then tested. Since we're trying to fin a maximum we can use other methods to find approximate solutions such as branch and bound, etc. There aren't any dedicated algorithms for this problem as it isn't particularly important.

[1] http://en.wikipedia.org/wiki/Travelling_salesman_problem
[2] http://en.wikipedia.org/wiki/Subset_sum_problem
[3] http://en.wikipedia.org/wiki/Tetris

E. Graph Coloring

Given a graph G with n vertices and some integer k we need to find out if G admits a proper vertex coloring with k colors. A proper vertex coloring happens when no two adjacent nodes are colored the same thing. The act of finding a coloring for the graph takes k^n using brute force search. Using dynamic programming we can get a faster algorithm. The best algorithm is Yates algorithm, which runs in $O(2^n n)$.

REFERENCES

[1] G. B. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. pp. 266–277, 1957. [Online]. Available: http://www.jstor.org/stable/167356