A Polynomial Time Algorithm for the Subset Sum Problem Using Symmetrical Sets and Solution Spaces[1]

[Dr. Philip Brisk (?)], Johnathan Fiske

Abstract: The subset sum problem has avoided an efficient solution –a polynomial time algorithm- since its formal definition in David Karp's 1979 paper. As of writing this paper, all known algorithms the most efficient algorithm is exponential time. This paper introduces a polynomial time algorithm of O(n^2) for the subset sum problem. This was accomplished using a new data structure called a solution space and a special set called a symmetrical set.

1. INTRODUCTION

2. New Algorithmic approach

2.1 This Approach vs Other Approach

Other algorithms have concentrated on relying on a statistical model that reduces the computational complexity but at the expense of accuracy. This algorithm does neither. This algorithm is only concerned at looking at possible solutions themselves with a larger set than the given set S.

2.2 Explanation of Using Symmetrical Sets and Solution Spaces

A symmetrical set can be derived from any set S. The symmetrical set is interesting because it has the following properties:

1. The subsets that sum to zero can be predicted ahead of time. A function can return the positions of which subsets sum to zero.
2. Symmetrical set will have at least one subset that equals zero.

The symmetrical set shows that the worst case scenario is that a set S has an exponential amount of solutions.

In summary, we compare our set S to its symmetrical set.

Example: if S = {-3, 2, 4} the symmetrical set would be: {-4,-3, -2, -1, 0, 1, 2, 3, 4}
if S = {-1, 2, 3, 5} the symmetrical set would be: {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}

The solution space, all subsets which sum to integer i, consists of the following:

1. A function (f1) that given a position will generate a corresponding subset.
2. A function (f2) that creates deadspace or positions that will not generate a corresponding subset, because these subsets have been eliminated.
3. After going through the algorithm, f2 will update f1 to create a new positioning function called (f3). It is (f3) that will be the solution space for our non-symmetrical set S.

1

4. Check our solution space for set S by running it through an algorithm. Make sure all the solutions really do sum to 0 (or integer i once you get more advanced).

^ If we think about it in just terms of positions, then .
^ In the case that the entire solution space becomes a deadspace then the solution space is empty and returns false.

^In the end, the solution space will do the following:
-If solution space returns true: return f3. And the whole number of solutions.
-If solution space returns false: return false
