An Almost Linear–Time Algorithm for the Dense Subset–Sum Problem

Zvi Galil *
Department of Computer Science,
Tel–Aviv University and Columbia University.

Oded Margalit
Department of Computer Science,
Tel–Aviv University

Abstract

This paper describes a new approach for solving a large subproblem of the subset-sum problem. It is useful for solving other NP-hard integer programming problems. The limits and potential of this approach are investigated.

The approach yields an algorithm for solving the dense version of the subset-sum problem. It runs in time $O(\ell \log \ell)$, where ℓ is the bound on the size of the elements. But for dense enough inputs and target numbers near the middle sum it runs in time $O(m)$, where m is the number of elements. Consequently, it improves the previously best algorithms by at least one order of magnitude and sometimes by two.

The algorithm yields a characterization of the set of subset sums as a collection of arithmetic progressions with the same difference. This characterization is derived by elementary number theoretic and algorithmic techniques. Such a characterization was first obtained by using analytic number theory and yielded inferior algorithms.

1 Introduction

There are several ways to cope with the NP-hardness of optimization problems. One is to look for an approximate solution rather than the optimum. There is a vast literature about approximation algorithms and approximation schemes. For some problems there are very good approximation algorithms, for others, the problem is still NP-hard even if we settle for an approximate solution. Another way to cope with complexity is to settle for the average case or to allow probabilistic algorithms. There are cases where this approach has paid off and faster algorithms have been discovered. But this has not been the case with NP-hard optimization problems.

A third approach of coping with NP-hardness is to try to restrict the problem and design a polynomial-time algorithm. Here too, there have been mixed results. Some

*Work partially supported by NSF Grants CCR-8814977 and CCR-9014605.

problems remain NP-hard even when restricted quite severely, others become feasible. In this paper we follow and extend the third approach. A restriction of the problem that allows a polynomial-time algorithm is known. However, the resulting algorithm has cubic time bound. We impose an additional restriction that allows much better algorithms and consequently much larger instances can be solved by these algorithms.

A novelty of our algorithm is the use of elementary number theory to design algorithms for solving an integer programming problem. It might be expected that this would be the natural tool for solving such problems. But we do not know of other such examples.

We use the following notation given a set D of integers: $S_D = \sum_{a \in D} a$ is the sum of the elements of D and $D^* = \{S_E \mid E \subseteq D\}$ is the set of subset sums of D. The **subset-sum problem** is: Given a set A of m distinct integers in the interval $[1, \ell]$ and an integer N, find a subset $B \subseteq A$ such that $S_B \leq N$ and there is no $C \subseteq A$ such that $S_B < S_C \leq N$. We assume without loss of generality that $N \leq \frac{1}{2} S_A$.

This problem is known to be NP-hard [12] but not in the strong sense: there is a pseudo polynomial-time algorithm for solving it [6]. A simple m stage dynamic program solves the problem. The i-th stage finds all subset sums of the i-th prefix of A which are not larger than N. The time is $O(N)$ per stage, for a total time of $O(mN)$. The worst case is when N is $\Omega(\ell m)$; in this case the time is $\Theta(m^2 \ell)$. The space needed by this algorithm can be as large as $\Theta(\ell m)$ just to find S_B and $\Theta(m^2 \ell)$ for finding B itself.

We derive an algorithm that is two orders of magnitude better than the dynamic programming approach, but works in a slightly more restricted domain. We impose two restrictions: we consider only dense instances, and we consider only an interval of target numbers around $S_A/2$. Note that the target numbers near the middle sum are the hardest cases for the dynamic programming algorithm. In our case $m^2 > c\ell \log^2 \ell$ and $L < N < S_A - L$ where $L = cS_A \ell / m^2 = o(S_A)$. Our algorithm runs in time $O(\ell \log \ell)$, but for dense enough instances and target numbers near the middle sum its time bound is linear ($O(m)$).

The algorithm consists of a preprocessing stage which is followed by a stage that depends on N and can be repeated for different target numbers. The preprocessing stage takes between $O(m)$ and $O(\ell \log \ell)$ time. The second stage (for a given N) takes only constant time to find S_B and $O(\log^2 m)$ time to obtain a characterization of B (only $\log m$ for dense enough inputs). Of course, we may need $\Theta(m)$ time to list the elements of B.

Our paper essentially concludes an interplay between analytic number theory and algorithm design. In the Appendix we sketch this history of "back and forth" between analytic number theoretical methods and elementary ones. Freiman [8], using methods from analytical number theory, analyzed the structure of A^* for dense subset-sum problems. More recently, the structure was used to derive algorithms which improved the dynamic programming approach. The final algorithm derived using analytic number theory [5] requires $O(\ell^2 \log \ell)$ time. This bound is the same as for dynamic programming for the lowest density and as the density increases its improvement increases. Our new algorithm is faster by at least one, sometimes two, orders of magnitude. Moreover, our algorithm yields an elementary proof of the characterization of A^* by constructing the sets whose sums are the corresponding elements of A^*.

In Section 2 we give a sketch of our algorithm. In Sections 3 and 4 we sketch the two major steps of the algorithm. Our sketch is qualitative and leaves many of the parameters undefined. In Section 3 we describe an efficient algorithm that constructs a set A^1 with