PSEUDO-POLYNOMIAL TIME SOLUTION OF A SUBSET-SUM PROBLEM USING A SINUSOIDAL BASIS

J. KEITH KENEMER

Abstract. The subset-sum problem is an NP-complete problem with no known polynomial-time solution. A method for solving a variant of this problem is presented which uses a sinusoidal basis to encode integers. The advantage of using a basis to encode numbers is that multiple numbers and computational results can be held in superposition, allowing an exponential number of candidate solutions to be explored simultaneously. A current limitation of this approach is that the precision of an encoded number is limited by the computational resources required to represent the basis functions and superpositions and this limitation results in pseudo-polynomial complexity.

1. Introduction

The proposed method is inspired by quantum computing and other waveform-based logic approaches such as Sinusoidal Logic and Noise-Based Logic (NBL). Computation using basis functions can be viewed as an exploration of the boundary between classical, analog, and quantum computation. In Sinusoidal and Noise-Based Logic, each bit is encoded as a basis function which is used to construct numbers in the waveform basis. In the proposed method, numbers are directly encoded as the frequency of a complex sinusoid. The disadvantage of this approach is a limitation of precision which is achievable due to the computational resources required for the representation. The advantage is that arithmetic operations, such as addition, are easier to compute in the waveform basis. Although other pseudo-polynomial time algorithms exist for this class of problems (e.g. dynamic programming), the use of basis functions to encode and process candidate solutions may provide an interesting new way to approach some optimization problems.

2. Subset Sum Problem With Repetition (SRP)

In the subset-sum problem (SSP), we are given a set of N integers and asked whether there is a subset which sums to a given target value. There are 2^N subsets to consider. A variant of this problem allows for repetition of the values chosen for the subsets. Here we will consider an SRP instance where the maximum number of repetitions is fixed but can be allocated across values in the subset. In other words, each possible subset is constructed by making k independent selections ($1 \leq k \leq N$) with replacement from the given set. Subsets of size 1 to N can be chosen and elements can be repeated up to N times. This problem contains SSP within it and so is also NP-complete. The problem can be stated as follows:

Given a set $A = \{a_1, a_2, ..., a_N\}$ of N non-zero elements, are there integers r_i in the range $0 \leq r_i \leq N$ which satisfy $\sum_i^N r_i \leq N$ such that $\sum_{i=1}^N a_i r_i = T$? This problem can also be interpreted as attempting to find a vector R with elements r_i such that the inner product exactly matches the target value: $\langle A, R \rangle = T$ subject to a constraint on the L_1-norm of R and the range of elements in R.

This problem can be encoded into a complex sinusoidal superposition using the following method. We will encode numbers as the frequency of a complex sinusoid, which represents the number of cycles per unit time. The corresponding complex sinusoid for element a_k is therefore $\Phi_k = e^{i\omega_k n}$, where $\omega_k = \frac{2\pi a_k}{L}$, n is the time index (sample number) and L is the number of samples used to represent a unit of time. We need to potentially encode numbers with magnitude ranging from 1 to $M = N \cdot max\{|a_i|\}$, which is the maximum possible sum magnitude. In order to satisfy the Nyquist sampling rate, we require at least 2 samples/cycle and so $L = 2M$. We can encode the set of all possible subset-sum choices in the following superposition:

$$\Psi(n) = \sum_{k=1}^{N} e^{i\omega_k n} \left(1 + \sum_{m=1}^{N} e^{i\omega_m n}\right)^{N-1} \tag{1}$$

The first product-term represents the fact that at least one choice a_k from set A must be made. The second (exponentiated) product-term represents the freedom of the remaining $N-1$ choices to be either a 'real' choice a_m from A, or a 'non-choice' of 0, which translates to $e^0 = 1$ allowing subsets of various sizes to be created. The multiplication/modulation of these terms generates sinusoids whose frequencies represent all the possible sums for all of the possible subset choices and these candidate solutions all coexist simultaneously in the output superposition. We can obtain a Yes/No answer to the SRP statement by checking for the presence of the sinusoid associated with the target value T by checking to see if the following inner product is non-zero:

$$\varphi = \left\langle \Psi(n), e^{\frac{2\pi i T}{L} n} \right\rangle \tag{2}$$

If the inner product is zero (or effectively zero relative to the working precision of the computation), then there is no solution. If the inner product is non-zero, then a solution exists. This inner product can be thought of as an oracle function–a function which provides guidance on how to reduce the search space for finding a solution by indicating when a solution is present. There are many variations on how a solution can be found but they would have the same general structure for iterating. Compute the oracle function $\varphi = \langle \Psi, \Phi_T \rangle$. Then iteratively restrict the search space and recompute the oracle function until a solution is found. This is described in more detail in the following section.

3. Search Algorithm

If we allow the possibility of an empty subset in the search space (which is legitimate), we can re-write Eq (1) as follows:

$$\Psi(n) = \left(\sum_{k=0}^{N} e^{i\omega_k n}\right)^N = \coprod_{m=1}^{N} \left(\sum_{k=0}^{N} e^{i\omega_k n}\right)_m \tag{3}$$

Here we have absorbed the non-choice into the set of possibilities by starting the indexing at 0. The degree of freedom we have in reducing the search space is to limit the index range in each summation. Since we have an oracle function which indicates when a solution is present in the superposition, we can perform a binary search of the index limits, ultimately resulting in a single frequency for each factor. This result describes which elements of A are contained within a satisfying subset, along with the number of occurrences which effectively gives the desired R vector. For example, the search process may conclude with Ψ in the state:

$$\Psi(n) = \left(e^{i\omega_1 n}\right)\left(e^{i\omega_1 n}\right)\left(e^{i\omega_3 n}\right)\left(e^{i\omega_5 n}\right) \tag{4}$$

which indicates that $A' = \{a_1, a_3, a_5\}$ is a satisfying subset with $R = \{2, 0, 1, 0, 1, 0, 0, 0, ...\}$. In other words, $2 * a_1 + a_3 + a_5 = T$. If we have a hypothetical subroutine called $search()$ which computes Ψ and the oracle function $\langle \Psi, \Phi_T \rangle$, then the entire search process can be completed using $N \cdot log(N)$ calls to $search()$.

4. Computational Complexity

The complexity will be estimated by counting the number of complex-valued operations, assuming all operations are approximately the same with respect to performance (as may be the cause for a native GPU implementation). For each sample n, The first product-term in Equation (1)requires N complex exponentiations and $N - 1$ complex additions. The second product term can re-use the sum-of-exponentials result from the first term, so it only requires one additional addition operation per sample (L operations). To complete the superposition requires the exponentiation of the second term (L operations) and then a multiplication of the two product terms (L operations). Therefore, construction of the superposition in Equation (1) requires $N_1 = (2N - 1) \cdot L + 3L$ complex-valued operations. Calculating the inner product from Equation (2) requires L complex multiplications and $L - 1$ complex additions for a total of $N_p = 2L - 1$ operations. The total number of complex-valued operations required to construct the initial superposition and compute the inner product against the target sinusoid represents the number of operations required for a Yes/No decision of satisifiability:

$$N_i = (2N - 1)L + 3L + (2L - 1) = 2NL + 4L - 1 = O(N \cdot L) \tag{5}$$

The $seach()$ routine requires independent control of each factor, however the overall product can be updated one factor at a time (see Eq 3). There are N complex exponentiations and N complex additions for each factor. This updated factor must be integrated into the overall product by a point-wise (sample-by-sample) multiplication which requires L complex multiplications. The factor representing the unrestricted choices must be modified by updating the exponent for each sample (L operations) and integrating this into the overall result by performing point-wise multiplication (L operations). Therefore the superposition update requires

$2NL + 3L$ operations. The oracle function inner product requires an additional $2L - 1$ operations. The complexity of $search()$ is therefore:

$$N_s = 2NL + 5L - 1 = O(N \cdot L) \tag{6}$$

In section 3, we observed that the entire search process can be completed in $N \cdot log(N)$ calls to $search()$. Therefore the total number of operations for entire search process, i.e the solution of the problem in on the order of:

$$N_T = O(L \cdot N \cdot N \cdot log(N)) = O(L \cdot N^2 \cdot log(N)) \tag{7}$$

The complexity is polynomial in N, the number of set elements and L, the number of samples used to represent the basis functions. However, $L = 2M$ and so depends on the maximum sum magnitude M, and this makes it dependent on the size of the input elements in set A. The input values are exponential in the number of bits, and so the overall search algorithm has pseudo-polynomial complexity.

5. Summary

A new method for solving a subset-sum problem variant in pseudo-polynomial time was presented which was inspired by quantum computing and other waveform-based logic schemes. The interesting aspect of the approach is mainly in how it solves the optimization problem, by holding all possible solutions in superposition, which allows a systematic reduction of the search space. This method also relies heavily on vector operations such as inner product, which can easily be performed in parallel. This type of approach may find applications for optimization problems where a minimum set of assumptions can be made for the input space and a high degree of parallelism is desired.