Neurocomputing 149 (2015) 13–21

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An efficient algorithm for the subset sum problem based on finite-time convergent recurrent neural network

Shenshen Gu*, Rui Cui

School of Mechatronic Engineering and Automation, Shanghai University, 149 Yanchang Road, Shanghai 200072, PR China

ARTICLE INFO

Article history:
Received 30 June 2013
Received in revised form
1 December 2013
Accepted 16 December 2013
Available online 12 August 2014

Keywords:
Subset sum problem
L smallest *k*-subsets sum problem
L shortest paths
Convergence in finite time
Recurrent neural network

ABSTRACT

For a given set S of n real numbers, there are totally 2^n-1 different subsets excluding the empty set. The subset sum problem is defined as finding L subsets whose summation of subset elements are the L smallest among all possible subsets. This problem has many applications in operations research and real world. However, the problem is very computationally challenging. In this paper, a novel algorithm is proposed to solve this problem. Firstly, the L smallest k-subsets sum problem, a special case and sub-problem of the subset sum problem, is investigated. Given a positive integer k ($k \le n$), k-subset means the subset of k distinct elements of S. Obviously, there are totally $\binom{n}{k}$ k-subsets. By expressing all these k-subsets with a network, the L smallest k-subsets sum problem is converted to finding L shortest loopless paths in this network. By combining the L shortest paths algorithm and the finite-time convergent recurrent neural network, a new algorithm for the L smallest k-subsets sum problem is developed. Finally, the solution to the subset sum problem is obtained by combining the solutions to these sub-problems. And experimental results show that the proposed algorithm is very effective and efficient.

1. Introduction

For a given set S of n real numbers, a k-subset means a subset of S containing k distinct elements, where $k<n$ is a positive integer [1]. The number of k-subsets on n elements is therefore given by the binomial coefficient $\binom{n}{k}$. For example, there are $\binom{4}{2}=6$ 2-subsets of $\{2,4,6,9\}$, namely $\{2,4\}$, $\{2,6\}$, $\{2,9\}$, $\{4,6\}$, $\{4,9\}$ and $\{6,9\}$. The values of summation of elements for these 2-subsets are 6, 8, 11, 10, 13 and 15 respectively. The L smallest k-subsets sum problem is defined as finding L k-subsets whose summation of subset elements are the L smallest among all possible combinations. It is obvious that the three smallest 2-subsets of $\{2,4,6,9\}$ are $\{2,4\}$, $\{2,6\}$ and $\{4,6\}$.

It is obvious that the total number of distinct k-subsets on set S of n elements is given by $\sum_{k=1}^{n}\binom{n}{k}=2^n-1$. For the previous example of $\{2,4,6,9\}$, these subsets are $\{2\}$, $\{4\}$, $\{6\}$, $\{9\}$, $\{2,4\}$, $\{2,6\}$, $\{2,9\}$, $\{4,6\}$, $\{4,9\}$, $\{6,9\}$, $\{2,4,6\}$, $\{2,4,9\}$, $\{2,6,9\}$, $\{4,6,9\}$ and $\{2,4,6,9\}$. Then finding the L smallest subsets is known as a subset sum problem. For example, the five smallest subsets of $\{2,4,6,9\}$ is $\{2\}$, $\{4\}$, $\{6\}$, $\{2,4\}$ and $\{2,6\}$. The subset sum problem is proved to be NP-complete [2].

The subset sum problem is very important in operations research. In addition, the algorithm for solving the subset sum problem could be applied to solve other real world problem, as many real world problems can be formulated as a subset sum problem. For example, in computer science, it is widely applied to the optimal memory management in multiple programming [3]. In the field of telecommunication, it is used in allocating wireless resources to support multiple scalable video sequences [4]. For the application in the embedded system, it is used in generating application specific instructions for DSP applications to reduce the required code size and increasing performance in embedded DSP systems [5]. And in optimization, the subset sum problem can also be studied as a special case of the Knapsack problem [6].

Due to the importance of the subset sum problem, many algorithms have been proposed. Lagarias and Odlyzko proposed a polynomial time algorithm for this problem in 1983 [7]. However, the algorithm is only suitable to the cases when the density of the problem, $(d(s)=n/\log_2(\max_i s_i))$, is less than $1/n$. In 1990, Lobstein proved that there is no polynomial-time algorithm solving the general subset sum problem [8]. Several heuristic algorithms have been proposed such as the quantum computation method [9], the space–time tradeoff method [10] and the penalty function method [3]. However, these algorithms do not always find a solution when one exists.

Since the subset sum problem can be thought of as the extension of the L smallest k-subsets sum problem. The solution to the subset sum problem can be found by solving a series of L smallest k-subsets sum problems according to some strategies that will be stated in detail in Section 5. And the time spent in solving

* Corresponding author.
E-mail address: gushenshen@shu.edu.cn (S. Gu).

http://dx.doi.org/10.1016/j.neucom.2013.12.063
0925-2312/

the subset sum problem is equivalent to the summation of time spent in solving these L smallest k-subsets sum problems. Then, it is clear that the subset sum problem can be solved very efficiently if there is a fast and exact algorithm for the L smallest k-subsets sum problem. For this reason, the L smallest k-subsets sum problem is first studied in detail in this paper. A specified structure network was creatively proposed to express all the $\binom{n}{k}$ k-subsets. Then, based on the relationship between the $\binom{n}{k}$ k-subsets and the proposed network, finding the L smallest k-subsets sum is equivalent to searching the L shortest loopless paths in this network. Furthermore, by combining the L shortest paths algorithm and the finite-time convergent recurrent neural network, a fast and exact algorithm for the L smallest k-subsets sum problem is developed. And finally, a complete algorithm to the subset sum problem is proposed by combining the results of these k-subsets sum problems.

The remainder of this paper is organized as follows. In Section 2, the L smallest k-subsets sum problem formulation is presented and the expression of the problem with a network is illustrated. Then the procedure for finding the L shortest loopless paths in this network is described in Section 3. In Section 4, by combining the L shortest loopless paths algorithm with the finite-time convergent recurrent neural network, a new algorithm is developed for the L smallest k-subsets sum problem. In Section 5, a complete algorithm to the subset sum problem by combining the results of these k-subsets sum problems is illustrated. Next, in Section 6, experimental results are given to verify the efficiency and effectiveness of the proposed algorithms for the L smallest k-subsets sum problem and the subset sum problem by comparing with the dynamic programming based algorithm [17]. Finally, Section 7 concludes this paper.

2. Problem formulation and model description of the L smallest k-subsets problem

To solve the L smallest k-subsets problem with optimization method, an appropriate mathematical model is needed. It is obvious that finding the L smallest k-subsets is equivalent to determine the 1st, 2nd, 3rd, …, $(L-1)$th and Lth smallest k-subsets step by step. Mathematically, the lth $(0<l\le L)$ smallest k-subset problem can be formulated as a function

$$x_i=\begin{cases}1 & \text{if } v_i\in\{\text{the } l\text{th smallest } k\text{-subset}\};\\ 0 & \text{otherwise};\end{cases} \tag{1}$$

for $i=1,\ldots,n$; where $v\in R^n$ is the given set, $x\in\{0,1\}^n$ indicates the elements of the given set belonging to the lth smallest k-subset, and $k\in\{1,\ldots,n-1\}$. Fig. 1 shows the operation graphically.

When $l=1$ and k is a nonnegative integer less than n, the above operation is almost the same with the k-Winners-Take-All (k WTA) operation [11]. The only difference is that k WTA operation finds k largest elements from n candidates instead of the smallest ones. However, solving the L smallest k-subsets sum problem is much more complicated than solving the k WTA problem. On one hand, when the $(l-1)$th smallest k-subset is already known as $\hat{x}_{l-1}$, the (l)th smallest k-subset can be obtained by solving the following integer optimization:

$$\begin{aligned}&\text{minimize}\quad v^Tx,\\ &\text{subject to}\quad e^Tx=k\\ &\quad x\neq\hat{x}_{l-1}\quad\text{(C1)}\\ &\quad v^Tx\ge v^T\hat{x}_{l-1}\quad\text{(C2)}\\ &\quad x_i\in\{0,1\},\quad i=1,2,\ldots,n.\end{aligned} \tag{2}$$

where $v=[v_1,\ldots,v_n]^T\in R^n$, $e=[1,\ldots,1]^T\in R^n$, $x=[x_1,\ldots,x_n]^T\in R^n$, l is an integer greater than one and k is a nonnegative integer less than n. Compared with the optimization formulation of k WTA problem [12], conditions (C1) and (C2) are added in finding the (l)th smallest k-subset, which increases the difficulty greatly. In addition, it is proved in [12] that condition $x_i\in\{0,1\}$ in solving k WTA can be relaxed to $x_i\in[0,1]$. However, this integer condition in optimization problem (2) cannot be relaxed. On the other hand, for the k WTA problem, only one round of optimization should be solved. However, for the L smallest k-subsets sum problem, totally L round of optimization (2) should be solved one by one from $l=1$ to $l=L$.

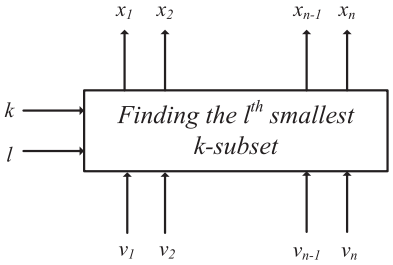

Fig. 1. Diagram of finding the lth $(0<l\le L)$ smallest k-subset operation.

Since integer optimization (2) is difficult to solve due to the inequality condition (C1) and the integer restriction for x, it is not wise to solve the original problem by integer programming. Here, a specific network structure is created to represent all k-subsets of set S. As shown in Fig. 2, each network has one source node n_{start} (i.e. start point), one sink node n_{end} (i.e. ending point) and several intermediate nodes such as $n_{1.1}$ and $n_{2.1}$. There are some links connecting different nodes, and for each link one weight is assigned. For example, in Fig. 2(b), the weight for the link connecting nodes $n_{1.1}$ and $n_{2.1}$ is four. For the network of Fig. 2 (b), it can be easily enumerated that there are totally six different paths from the source node to the sink node. Each path represents a 2-subset by the weights of the links constituting this path. For instance, path $n_{start}\to n_{1.1}\to n_{2.1}\to n_{end}$ represents subset $\{2,4\}$. And all these six paths represent all 2-subsets of set $\{2,4,6,9\}$. When constructing a network for representing k-subsets of n elements, the network is composed of a source node, a sink node and k layers of intermediate nodes. For all the layers $(i=1,\ldots,k)$, it has $(n-k+1)$ intermediate nodes. It can be proved that the upper bound for the number of nodes is nk. For the first intermediate layer $(i=1)$, these $(n-k+1)$ nodes are connected with the source node. And the weights on these links are assigned by the $(n-k+1)$ smallest elements of the given set. For the intermediate layers $(i=2,\ldots,k)$, connect these nodes with the $(i-1)$th layer nodes properly and assigned the ith to the $(i-n-k-1)$th smallest elements of the given set to these links appropriately. For the last intermediate layer $(i=k)$, these $(n-k+1)$ nodes are connected with the sink node. And the weights on these links are all assigned by 0. By applying these procedures, a network representing all k-subsets of n elements can be constructed. Therefore, it is clear that finding the L smallest k-subsets sum is equivalent to searching L shortest paths in a network.

3. Finding L shortest paths in a network

Finding L shortest paths in a network is a well-known optimization problem. Many real world applications such as transportation system can be formulated by this problem [14]. There are several algorithms available for finding L shortest paths in a network, among which the algorithm proposed by Yen is one of the most efficient in term of the number of operations and the number of memory addresses [13]. Here the algorithm is adapted to find L shortest paths in the specific structure network. The first