Information Sciences 180 (2010) 3974–3979

Contents lists available at ScienceDirect

Information Sciences

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

A dynamic programming algorithm for tree-like weighted set packing problem

Mehmet Gulek, Ismail Hakki Toroslu *

Middle East Technical University, Department of Computer Engineering, Ankara, Turkey

ARTICLE INFO

Article history:
Received 7 July 2009
Received in revised form 19 May 2010
Accepted 23 June 2010

Keywords:
Assignment problem
Dynamic programming
Set packing problem
Tree-like structure
Complexity of algorithm

ABSTRACT

In hierarchical organizations, hierarchical structures naturally correspond to nested sets. That is, we have a collection of sets such that for any two sets, either one of them is a subset of the other, or they are disjoint. In other words, a nested set system forms a hierarchy in the form of a tree structure. The task assignment problem on such hierarchical organizations is a real life problem. In this paper, we introduce the tree-like weighted set packing problem, which is a weighted set packing problem restricted to sets forming tree-like hierarchical structure. We propose a dynamic programming algorithm with cubic time complexity.

1. Introduction

The set packing problem is an NP-Complete problem, it is one of 21 problems of Karp [8]. The set packing problem asks whether there are some k pairwise disjoint subsets in a family S of subsets of a universal set U.

Bounding the cardinalities of the subsets by a number m results in a similar problem called m-set packing problem. That is, all the subsets should have cardinality at most m. This problem is also NP-Complete for $m \geqslant 3$ [2]. Algorithms for the parameterized version of m-Set Packing can be found in Fellows et al. [1], Jia et al. [7] and Liu et al. [11].

Below we define the weighted set packing problem as a decision problem.

Definition. Let U be a set. Let S be a family of subsets of U. Let w: $S \to N$ be a weight function assigning (possibly negative) weights to the subsets. Let $k \in N$ and $h \in N$. The *weighted set packing problem* asks whether there are some k pairwise disjoint subsets K in S such that $h \leqslant \sum_{D \in K} w(D)$.

Clearly, the weighted set packing problem is NP-Complete due to the NP-Completeness of the unweighted case.

Problems similar to the weighted set packing problem can be found in the literature. For example, in Halldórsson and Chandra [5] the Weighted m-set packing problem was defined. This problem does not contain the constraint k, so, only the total weight is maximized. The optimization version of Weighted m-Set Packing is NP-Hard [5]. In Halldórsson and Chandra [5], an approximation algorithm for the problem was given with an approximation ratio $2(m + 1)/3$. Also, a parameterized algorithm for Weighted m-Set Packing was introduced in Liu et al. [10].

In hierarchical organizations, hierarchical structures naturally correspond to nested sets. That is, we have a collection of sets such that for any two sets X and Y, either X is a subset of Y, or Y is a subset of X, or X and Y are disjoint. In other words, a nested set system forms a hierarchy in the form of a tree structure.

* Corresponding author. Tel.: +90 312 210 5585; fax: +90 312 210 5544.
E-mail address: toroslu@ceng.metu.edu.tr (I.H. Toroslu).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.06.035

The task assignment problem on such hierarchical organizations is a real life problem. For example, in military applications, the engagement rules for artillery correspond to this kind of target assignment problem [3]. This target assignment problem is equivalent to the weighted set packing problem in which the sets are nested. Other variations of hierarchical task assignment problems can be found in Toroslu and Arslanoglu [13].

Various forms of assignment problems have been studied in the literature [9,6,12], but, none of them are similar to our work.

In the following section we define the tree-like weighted set packing problem, which is a weighted set packing problem restricted to sets forming a tree-like hierarchical structure. Then, we describe an efficient solution using dynamic programming. To the best of our knowledge this is the first introduction of this problem.

2. Tree-like weighted set packing problem

In this section we introduce the Tree-like weighted set packing problem, which turns out to be in P, by imposing a condition on the internal structure of the family of subsets at hand. The condition is stated using the notion of a *tree*, showing why the selected title makes sense (we actually use the notion of *forest* instead of *tree* to obtain a more general problem).

Definition. We say that a family S of nonempty subsets is *tree-like*, if the subsets in S can be organized as a forest F of rooted trees such that,

1. the nodes in F are exactly the subsets in S,
2. if a node Y is a child of another node X in F, then we have $Y \subset X$,
3. if Z is another child of X, then Y and Z are disjoint,
4. the roots are pairwise disjoint (i.e. they have no common elements).

There may be other definitions of the notion "*tree-like*", as in Guo and Niedermeier [4]. Guo and Niedermeier [4] deal with the problem *Tree-Like Weighted Set Cover* (*TWSC*). In that work, a subset collection C is called *tree-like* if the subsets in C can be organized in a tree T such that every subset one-to-one corresponds to a node of T and, for each element s in C, the nodes of T corresponding to the subsets containing s induce a subtree of T. Our *tree-like* definition is very similar to the one in Guo and Niedermeier [4], except that we also allow forests.

Below we will define the Tree-like weighted set packing problem as an optimization problem. We will avoid the notion of disjoint subsets, since two disjoint sets in this setting correspond to two independent nodes. First we define **independency** in an arbitrary forest:

Definition. We say two distinct nodes x and y in a forest are *dependent*, if they are on the same path from a root to a leaf in the forest. Otherwise they are *independent*. We can extend the definition to set of nodes. We say that a node set N is *independent* if any two distinct nodes x and y in N are *independent*.

Definition. Let F be a forest of rooted and weighted trees with n nodes. The weights of the nodes are (possibly negative) integers. For a node d, let $w(d)$ denote the weight of d. Let also $k \in N$ $(k \leqslant n)$ be given. The problem is to find a set $K = \{d_1, d_2, \ldots, d_k\}$ of k nodes in F such that, the sum of the weights of these nodes $\sum_{d \in K} w(d)$ is maximal among all such sets, subject to the **independency** constraint, which states that any two distinct nodes $x, y \in K$ should be independent.

We will call this problem the *tree-like weighted set packing problem*.

In what follows, for simplicity, we will assume that F is actually a tree. We do not lose any generality doing so, because, otherwise, F can be converted to a tree by adding a new node r as root, and making all the old roots children of r. This will not affect the result, provided that the artificial node r has a weight less than all other nodes.

3. A recurrence relation for the problem

For a nonempty tree T, a weight function w, and a number $k \geqslant 0$, let $Q_k(T)$ denote the maximum sum of weights of k pairwise independent nodes in T, that is, what the defined problem asks. Let n denote the number of nodes in T, and t denote the number of leaves in T. We also want to achieve $Q_k(T) = -\infty$ if $k > t$.

Here is a recurrence relation for this problem. Let r denote the root of a nonempty tree T. Let r have $m \geqslant 0$ children (subtrees) $c_i (1 \leqslant i \leqslant m)$ and $0 \leqslant k$. Again, let t denote the number of leaves in T (note that $t > 0$ since we assume T is nonempty):

$$\mathtt{Q_k(T)} = \begin{cases} 0 & \text{if } k = 0, \\ max(\mathtt{Q_1(c_i)}) max\, w(r) & \text{if } k = 1, \\ -\infty & \text{if } k > t, \\ max\left(\sum_{i=1}^{m} \mathtt{Q_{k_i}(c_i)} \ \text{ such that } \sum_{i=1}^{m} k_i = k \right) & \text{otherwise}. \end{cases}$$