Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 3396 – 3400

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineeringand Information Science

Dynamic Programming for NP-Hard Problems

Xiaodong Wang[a], Jun Tian[b]

[a]Quanzhou Normal University,Quanzhou, 362000,China
[b] Fujian Medical University, Fuzhou, 350005, China

Abstract

This paper presents the extremely simple algorithms for NP-hard subset-sum like problems with the bitset class. The presented algorithms decrease the time and space complexity of dynamic programming algorithms by exploiting word parallelism.
The computational experiments demonstrate that the achieved results are not only of theoretical interest, but also that the techniques developed may actually lead to considerably faster algorithms.

Keywords: subset sum; NP hard; word RAM; bitset

1. Introduction

In this paper, we consider the following NP-hard problems.

Subset-Sum Problem:

Given a set $S = \{a_1, a_2, \ldots, a_n\}$ of n nonnegative integers and an additional integer m, the subset-sum problem asks to choose a subset $\overline{S}$ of S such that $\sum_{\overline{S}} a_i$ is maximized without exceeding m. The integers $a_i \le m$ are frequently denoted the weights and m the capacity.

Bounded Subset-Sum Problem:

The bounded subset-sum problem is a generalization of the subset-sum problem where each weight may be chosen a bounded number of times.

1877-7058 © 2011 Published by Elsevier Ltd.
doi:10.1016/j.proeng.2011.08.636

Given a set $S = \{(a_1, d_1), (a_2, d_2), \ldots, (a_n, d_n)\}$ of nonnegative integer pairs, and an additional integer m, the problem asks to maximize the sum $\sum_S a_i x_i$ without exceeding m, respecting that $x_i \in \{0,1,\ldots,d_i\}$. The integers a_i denoted the weights, d_i denoted the bounds and m denoted the capacity.

Unbounded Subset-Sum Problem:

Given a set $S = \{a_1, a_2, \ldots, a_n\}$ of n nonnegative integers and an additional integer m, the unbounded subset-sum problem asks to maximize the sum $\sum_S a_i x_i$ without exceeding m, where $x_i \in \{0,1,\ldots\}$.

Two-Partition Problem:

The two-partition problem can be formulated as follows: Two sets of integers $S = \{a_1, a_2, \ldots, a_n\}$ and $T = \{a'_1, a'_2, \ldots, a'_n\}$ are given together with an m. Find two subsets $\overline{S}$ of S and $\overline{T}$ of T such that $\sum_{\overline{S}} a_i = \sum_{\overline{T}} a'_i$ is maximized without exceeding m.

A problem exhibits the property of optimal substructure can be solved through dynamic programming as this framework takes advantage of overlapping subproblems to decrease the computational effort[4].

Applying dynamic programming to *NP*-hard problems may lead to algorithms with pseudo polynomial running time. The dynamic programming algorithms generally have the additional benefit that we do not only obtain a single solution but a whole table of optimal sub-solutions corresponding to different values of the constraints. Solving the problem for all values of the constraints will change the problems considered to polynomial problems with respect to the input and output size.

In a word RAM, the operations like binary and, binary or, and bitwise shift with w bits can be implemented in constant time on words of size w. None of the developed algorithms make use of multiplication apart from the process of indexing multidimensional tables. The indexing can however be implemented by a single shift operation per index if the domain size of the index is extended to the nearest larger power of two. For an excellent survey on sorting and searching algorithms on the word RAM see[5].

The present paper exploits word parallelism for the *NP*-hard problems described above by using the bitset class. As the computational effort for solving these problems is huge, word parallelism is of great importance both from a theoretical and practical aspect.

2. The Algorithms Using Bitset Class

For the subset-sum problem, a straightforward dynamic programming algorithm can be designed as follows.

Let $t_{i,j}$ for $i = 0,\ldots,n, j = 0,\ldots,m$ be a solution to the subset-sum problem defined on items $S_i = \{a_1, \ldots, a_i\}$ with $m = j$.

To initialize the recursion we set $t_{0,j} = 0$ for $j = 0,\ldots,m$.

Subsequent values of t can be computed recursively as

$$t_{i,j} = \max\{t_{i-1,j}, t_{i-1,j-a_i} + a_i\} \qquad (1)$$

for $i = 1,\ldots,n, j = 0,\ldots,m$.