Computers and Mathematics with Applications 58 (2009) 619–631

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

A graphical realization of the dynamic programming method for solving *NP*-hard combinatorial problems

Alexander A. Lazarev [a,*], Frank Werner [b]

[a] *Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya street 65, 117997 Moscow, Russia*
[b] *Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany*

ARTICLE INFO

Keywords:
Dynamic programming
Exact algorithm
Graphical algorithm
Partition problem
Knapsack problem

ABSTRACT

In this paper, we consider a graphical realization of dynamic programming. The concept is discussed on the partition and knapsack problems. In contrast to dynamic programming, the new algorithm can also treat problems with non-integer data without necessary transformations of the corresponding problem. We compare the proposed method with existing algorithms for these problems on small-size instances of the partition problem with $n \leq 10$ numbers. For almost all instances, the new algorithm considers on average substantially less "stages" than the dynamic programming algorithm.

1. Introduction

Dynamic programming is a general optimization technique developed by Bellman. It can be considered as a recursive optimization procedure which interprets the optimization problem as a multi-stage decision process. This means that the problem is decomposed into a number of stages. At each stage, a decision has to be made which has an impact on the decision to be made in later stages. By means of Bellman's optimization principle [1], a recursive equation is set up which describes the optimal criterion value at a given stage in terms of the optimal criterion values of the previously considered stage. Bellman's optimality principle can be briefly formulated as follows: Starting from any current stage, an optimal policy for the subsequent stages is independent of the policy adopted in the previous stages. In the case of a combinatorial problem, at some stage α sets of a particular size α are considered. To determine the optimal criterion value for a particular subset of size α, one has to know the optimal values for all necessary subsets of size $\alpha - 1$. If the problem includes n elements, the number of subsets to be considered is equal to $O(2^n)$. Therefore, dynamic programming usually results in an exponential complexity. However, if the problem considered is *NP*-hard in the ordinary sense, it is possible to derive pseudopolynomial algorithms. The application of dynamic programming requires special separability properties.

In this paper, we give a graphical realization of the dynamic programming method which is based on a property originally given for the single machine total tardiness problem by Lazarev and Werner (see [2], Property B-1). This approach can be considered as a generalization of dynamic programming. The new approach often reduces the number of "states" to be considered in each stage. Moreover, in contrast to dynamic programming, it can also treat problems with non-integer data without necessary transformations of the corresponding problem. However, the complexity remains the same as for a problem with integer data in terms of the "states" to be considered.

In the following, we consider the partition and knapsack problems for illustrating the graphical approach. Both these combinatorial optimization problems are *NP*-hard in the ordinary sense (see, e.g. [3–6]). Here, we consider the following formulations of these problems.

Partition problem: Given is an ordered set $B = \{b_1, b_2, \ldots, b_n\}$ of n positive numbers with $b_1 \geq b_2 \geq \cdots \geq b_n$. We wish to determine a partition of the set B into two subsets B^1 and B^2 such that

* Corresponding author.
E-mail addresses: jobmath@mail.ru (A.A. Lazarev), frank.werner@mathematik.uni-magdeburg.de (F. Werner).

0898-1221/$ – see front matter Crown Copyright © 2009 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.06.008

$$\left|\sum_{b_i\in B^1} b_i - \sum_{b_i\in B^2} b_i\right| \to \min, \tag{1}$$

where $B^1 \cup B^2 = B$ and $B^1 \cap B^2 = \oslash$.

One-dimensional knapsack problem: One wishes to fill a knapsack of capacity A with items having the largest possible total utility. If any item can be put at most once into the knapsack, we get the binary or $0 - 1$ knapsack problem. This problem can be written as the following integer linear programming problem:

$$\begin{cases} f(x) = \sum_{i=1}^{n} c_i x_i \to \max \\ \sum_{i=1}^{n} a_i x_i \le A; \\ 0 < c_i,\ 0 < a_i \le A,\ i = 1, 2, \ldots, n; \\ \sum_{i=1}^{n} a_i > A; \\ x_i \in \{0, 1\},\ i = 1, 2, \ldots, n. \end{cases} \tag{2}$$

Here, c_i gives the utility and a_i the required capacity of item i, $i = 1, 2, \ldots, n$. The variable x_i characterizes whether item i is put into the knapsack or not.

We note that problems (1) and (2) are equivalent if

$$c_i = a_i = b_i \quad \text{for } i = 1, 2, \ldots, n \quad \text{and} \quad A = \frac{1}{2}\sum_{j=1}^{n} b_j.$$

For the application of the graphical algorithm, the problem data may be arbitrary non-negative real numbers.

This paper is organized as follows. In Section 2, we consider the partition problem. First we explain the concept of the graphical algorithm for this problem. Then we describe in Section 2.2 how the number of intervals (or points) to be considered by the graphical algorithm can be reduced. The algorithm is illustrated by an example in Section 2.3. Then we prove the optimality of the given algorithm and discuss some complexity aspects in Section 2.4. Computational results for small-size instances are given in Section 2.5. In Section 3, we consider the knapsack problem. In Section 3.1, a brief illustration of the application of dynamic programming to the knapsack problem is given. In Section 3.2, the graphical algorithm is applied to the knapsack problem. An illustrative example for the graphical algorithm is given in Section 3.3. Some complexity aspects are discussed in Section 3.4. Finally, we give some concluding remarks in Section 4.

2. Partition problem

2.1. Graphical algorithm

First, we explain the graphical realization of the dynamic programming method for the partition problem. We describe the approach for an arbitrary step (or stage) α: first for $\alpha = 1$ and then for $\alpha = 2, 3, \ldots, n$. In each step, we determine function $F_\alpha(t)$ and best partitions $(B^1_\alpha(t); B^2_\alpha(t))$ for the set $\{b_1, b_2, \ldots, b_\alpha\}$ in dependence on parameter t by means of the results of the previous step $\alpha - 1$. The value $F_\alpha(t)$ describes the minimal function value (1) for including $b_1, b_2, \ldots, b_\alpha$ into one of the current subsets B^1 or B^2 subject to the constraint that in steps $\alpha + 1, \alpha + 2, \ldots, n$, altogether t more units of the numbers $b_{\alpha+1}, b_{\alpha+2}, \ldots, b_n$ are included into the corresponding set B^1 than into the corresponding set B^2. If t is negative, it means that t more units are included into set B^2 in the next steps.

In the initial step $\alpha = 1$, we have

$$F_1(t) = \begin{cases} -(t + b_1), & \text{if } -\sum_{j=1}^{n} b_j \le t < -b_1; \\ t + b_1, & \text{if } -b_1 \le t < 0; \\ -(t - b_1), & \text{if } 0 \le t < b_1; \\ t - b_1, & \text{if } b_1 \le t \le \sum_{j=1}^{n} b_j \end{cases}$$

and

$$(B^1_1(t); B^2_1(t)) = \begin{cases} (b_1; \oslash), & \text{if } -\sum_{j=1}^{n} b_j \le t < 0; \\ (\oslash; b_1), & \text{if } 0 \le t \le \sum_{j=1}^{n} b_j. \end{cases}$$