Theoretical Computer Science 645 (2016) 41–47

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A deterministic fully polynomial time approximation scheme for counting integer knapsack solutions made easy ☆,☆☆

Nir Halman

Hebrew University of Jerusalem, Israel

ARTICLE INFO

Article history:
Received 2 December 2015
Received in revised form 9 May 2016
Accepted 10 June 2016
Available online 16 June 2016
Communicated by P. Krysta

Keywords:
Approximate counting
Integer knapsack
Dynamic programming
Binding constraints
K-approximating sets and functions

ABSTRACT

Given n elements with nonnegative integer weights $w = (w_1, \ldots, w_n)$, an integer capacity C and positive integer ranges $u = (u_1, \ldots, u_n)$, we consider the counting version of the classic integer knapsack problem: find the number of distinct multisets whose weights add up to at most C. We give a deterministic algorithm that estimates the number of solutions to within relative error ϵ in time polynomial in n, $\log U$ and $1/\epsilon$, where $U = \max_i u_i$. More precisely, our algorithm runs in $O(\frac{n^3 \log^2 U}{\epsilon} \log \frac{n \log U}{\epsilon})$ time. This is an improvement of n^2 and $1/\epsilon$ (up to log terms) over the best known deterministic algorithm by Gopalan et al. (2011) [5]. Our algorithm is relatively simple, and its analysis is rather elementary. Our results are achieved by means of a careful formulation of the problem as a dynamic program, using the notion of *binding constraints*.

1. Introduction

In this paper we target at designing a deterministic fully polynomial time approximation scheme (FPTAS) for one of the most basic #P-complete counting problems – counting the number of integer knapsack solutions. Given n elements with nonnegative integer weights $w = (w_1, \ldots, w_n)$, an integer capacity C, and positive integer ranges $u = (u_1, \ldots, u_n)$, we consider the counting version of the classic integer knapsack problem: find the size of the set of feasible solutions $\mathrm{KNAP}(w, C, u) = \{x \mid \sum_{i \le n} w_i x_i \le C,\ 0 \le x_i \le u_i\}$. (We assume, w.l.o.g., that $w_i u_i \le C$ for all i.) We give a deterministic FPTAS for this problem that for any tolerance $\epsilon > 0$ estimates the number of solutions within relative error ϵ in time polynomial in the (binary) input size and $1/\epsilon$.

Our result Our main result is the following theorem (the base of the logarithms in this paper are all 2 unless otherwise specified).

Theorem 1.1. *Given a knapsack instance $\mathrm{KNAP}(w, C, u)$ with $U = \max_i u_i$ and $\epsilon > 0$, there is a deterministic $O(\frac{n^3 \log^2 U}{\epsilon} \log \frac{n \log U}{\epsilon})$ algorithm that computes an ϵ-relative error approximation for $|\mathrm{KNAP}(w, C, u)|$.*

Relevance to existing literature The field of approximate counting is largely based on Markov Chain Monte Carlo Sampling [1], a technique that is inherently randomized, and has had remarkable success, see [2] and the references therein. The first

☆ A preliminary version of this paper appeared in APPROX/RANDOM 2016.

☆☆ Partial support for this research was provided by the Recanati Fund of the Jerusalem School of Business Administration.

E-mail address: halman@huji.ac.il.

http://dx.doi.org/10.1016/j.tcs.2016.06.015
0304-3975/

approximation schemes for counting integer knapsack solutions are fully polynomial *randomized* approximation schemes (FPRASs). Given parameters $\epsilon > 0$ for the error tolerance and $1 > \delta > 0$ for the failure probability, the FPRAS returns a solution which is correct with probability at least $1-\delta$, and the running time is required to be polynomial in the (binary) input size, $1/\epsilon$ and in $\log(1/\delta)$. To the best of our knowledge, the best FPRAS up to date is given by Dyer [3], and is achieved by combining dynamic programming with simple rejection sampling. The complexity of the algorithm is $O(n^5 + n^4/\epsilon^2)$, so in fact the algorithm is strongly polynomial (see, e.g., [4]), that is, the number of arithmetic operations is polynomial in n and independent of C, U.

To the best of our knowledge, the currently best (deterministic) FPTAS for this problem is given by Gopalan et al. [5], and has complexity $O(\frac{n^5}{\epsilon^2}\log^2 U \log W)$, where $W = \sum_i w_i u_i + C$ (see also [6]). We note that the real achievement of [6] is providing an FPTAS for the *multidimensional* version of the problem. Because of this reason they use a somewhat more sophisticated approach than ours, relying on read-once branching programs and insight from Meka and Zuckerman [7].

We note in passing that the first (deterministic) FPTAS for counting 0/1 knapsack solutions (i.e., our problem restricted to the case where $u = (1, \ldots, 1)$) is given by Štefankovič et al. [2] and runs in $O(n^3\epsilon^{-1}\log(n/\epsilon))$ time. The currently best (deterministic) FPTAS runs in $O(n^3\epsilon^{-1}\log(1/\epsilon)/\log n)$ time [8].

Technique used In this paper we give two FPTASs that are based upon formulating the counting problem as a dynamic program. Instead of deciding at once how many copies of item i to put in the knapsack, we split the decision into a sequence of at most $\log u_i$ binary sub-decisions concerning (not necessarily all) the bundles of $1, 2, 4, \ldots, 2^{\lfloor \log u_i \rfloor}$ copies of the item. In order to "translate" this into a dynamic program, we use the idea of what we call *binding constraints*, as explained in detail below. The first FPTAS uses a "primal" DP formulation and approximates it via the recent technique of K-approximation sets and functions introduced by [9], which we overview in Section 2.1. The second FPTAS uses a "dual" DP formulation and approximates it in a similar way [2] approximate the 0/1 knapsack problem. We overview their solution in Section 3.1.

Our contribution While not strongly polynomial, the running time of our solutions are of order n and $1/\epsilon$ (up to log terms) faster than the (randomized, but strongly-polynomial) algorithm of Dyer [3]. The complexity of our solutions is also better by factors of n^2 and $1/\epsilon$ (up to log terms) than the (non strongly-polynomial, but deterministic) algorithm of Gopalan et al. [5]. Moreover, our algorithms are relatively simple and their analysis is rather elementary. A second contribution is our new DP technique – "binding constraints", which may be of independent interest.

Organization of the paper In Section 2 we present an FPTAS which is based upon a primal DP formulation of the problem. Our second FPTAS, based upon a dual DP formulation, is given in Section 3. In this way we showcase that the idea of binding constraints is useful for the primal as well as the dual DP formulation.

2. Algorithm via a primal DP formulation

A pseudo-polynomial algorithm is achieved using the following recurrence:

$$\begin{aligned} s_i(j) &= \textstyle\sum_{k=0}^{m_i(j)} s_{i-1}(j-kw_i) \qquad 2 \le i \le n,\ j = 1, \ldots, C, \\ s_1(j) &= m_1(j) + 1 \qquad\qquad\qquad\qquad\quad j = 1, \ldots, C, \end{aligned} \tag{1}$$

where function $m_i : [0, \ldots, C] \to \mathbb{Z}^+$ is defined as $m_i(j) := \max\{x \in \mathbb{Z}^+ \mid x \le u_i,\ xw_i \le j\}$ and returns the maximum number of copies of item i that can be placed in a knapsack with capacity j. Here $s_i(j)$ is the number of integer knapsack solutions that use a subset of the items $\{1, \ldots, i\}$ whose weights sum up to at most j. The solution of the counting problem is therefore $s_n(C)$. The complexity of this pseudo-polynomial algorithm is $O(nUC)$, i.e., exponential in both the (binary) sizes of U and C. We call such formulation *primal* because the range of the functions in (1) is the number of solutions.

In order to get our FPTAS we give in Section 2.2 a more careful DP formulation which is exponential only in the (binary) size of C. Before doing so, we briefly overview the technique of K-approximation sets and functions in Section 2.1. We use this technique in order to get our first FPTAS.

2.1. K-approximation sets and functions

Halman et al. [9] have introduced the technique of K-approximation sets and functions, and used it to develop an FPTAS for a certain stochastic inventory control problem. Halman et al. [10] have applied this tool to develop a framework for constructing FPTASs for a rather general class of stochastic dynamic programs. This technique has been used to yield FPTASs to various optimization problems, see [10] and the references therein. In this section we provide an overview of the technique of K-approximation sets and functions. In the next section we use this tool to construct FPTASs for counting the number of solutions of the integer knapsack problem. To simplify the discussion, we modify Halman et al.'s definition of the K-approximation function by restricting it to integer-valued nondecreasing functions.

Let $K \ge 1$, and let $\varphi : \{0, \ldots, B\} \to Z^+$ be an arbitrary function. We say that $\tilde{\varphi} : \{0, \ldots, B\} \to Z^+$ is a *K-approximation function* of φ if $\varphi(x) \le \tilde{\varphi}(x) \le K\varphi(x)$ for all $x = 0, \ldots, B$. The following property of K-approximation functions is extracted