Information Processing Letters 115 (2015) 170–174

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

An approximation algorithm for the cutting-sticks problem

Jagadish M

Indian Institute of Technology Bombay, Powai, Mumbai 400076, India

ARTICLE INFO

Article history:
Received 9 December 2012
Received in revised form 5 May 2014
Accepted 6 September 2014
Available online 13 October 2014
Communicated by M. Chrobak

Keywords:
Approximation algorithms
Cutting-sticks

ABSTRACT

The *cuttings-sticks* problem is the following. We are given a bundle of sticks all having integer lengths. The total sum of their lengths is $n(n+1)/2$. Can we break the sticks so that the resulting sticks have lengths $1, 2, \dots, n$? The problem is known to be NP-hard. We consider an optimization version of the problem which involves cutting the sticks and placing them into boxes. The problem has a trivial polynomial time algorithm with an approximation ratio of 2. We present a greedy algorithm that achieves an approximation ratio of $\sqrt{2}$.

1. Introduction

Cutting-sticks problem. We are given k sticks all having integer lengths. Their lengths are $l_1, \dots, l_k$ and the total sum of their lengths is $n(n+1)/2$. Can we break the sticks to get sticks of lengths $1, 2, \dots, n$?

Notation. Let $[r]$ denote the set $\{1, 2, \dots, r\}$.

We propose an optimization version of the cutting-sticks problem and give an approximation algorithm for it. We first state the decision version of the problem slightly differently to resemble the optimization version.

Decision version. We are given k positive integers $l_1, \dots, l_k$ as input. Their total sum is $n(n+1)/2$. Can we partition the set $[n]$ into k subsets $B_1, \dots, B_k$ such that the sum of numbers in B_i equals l_i for all $1 \le i \le k$?

It is easy to see that the alternate formulation does not change the problem. For any positive instance, the ith stick can be broken down into sizes contained in the set B_i.

Optimization version. Given positive integers $l_1, \dots, l_k$ as input, find the smallest t for which we can partition the set $[t]$ into k subsets $B_1, \dots, B_k$ such that the sum of numbers in B_i is at least l_i for all $1 \le i \le k$. Output t and the corresponding partition in polynomial time.

Related work. The cutting-sticks problem is NP-hard since the 3-partition problem reduces to it [1]. For the special case when all the lengths are equal, there exists a polynomial time algorithm to solve the problem exactly [2]. We give a $\sqrt{2}$-approximation factor algorithm for the optimization version of the problem. Suppose we are given an instance of the problem that has $t = OPT$ as the output. Our algorithm outputs a number that is at most $\sqrt{2OPT}$ and gives the corresponding partitioning of the set $[\lceil\sqrt{2OPT}\rceil]$.

1.1. Physical interpretation

For ease of exposition, we give a physical interpretation of the optimization problem. Roughly, we can see the problem as breaking a set of sticks in order to fit them into a set of boxes.

Definition. The *box* b_i is of *size* i. Box b_i can *contain* a stick of length j if $j \le i$. Equivalently, a stick of length j can *fit* into a box whose size is at least j. We refer to boxes $b_1, \dots, b_t$ as '$[t]$ boxes'.

The set of sticks is said to *fit into* $[t]$ *boxes*, if the sticks can be broken into shorter sticks (*pieces*) such that:

1. Each piece fits into some box b_i.
2. One box contains at most one piece. (Some boxes could remain empty.)

E-mail address: jagadish@cse.iitb.ac.in.

http://dx.doi.org/10.1016/j.ipl.2014.09.007
0020-0190/© 2014 Elsevier B.V. All rights reserved.

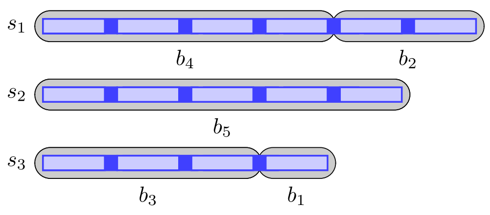

Fig. 1. An optimal way to fit sticks of lengths 6, 5 and 4 into [5] boxes. The first stick is broken into two pieces of lengths 4 and 2. The second stick is a piece by itself. The third stick is broken into pieces of lengths 3 and 1. The corresponding solution to the original optimization problem is $B_1 = \{4, 2\}$, $B_2 = \{5\}$ and $B_3 = \{3, 1\}$. If the sticks were of lengths 6, 5 and 3, an optimal solution would still use [5] boxes and could have the same partition (box b_1 would be left empty).

In light of the above definitions, the optimization problem we want to solve can be stated as follows. We are given k sticks $s_1, \dots, s_k$ having lengths $l_1, \dots, l_k$, respectively. What is the smallest number t for which we can we fit the sticks into $[t]$ boxes?

For example, if the stick lengths are 6, 5 and 4, then smallest t for which we can fit the sticks into $[t]$ boxes is 5 (see Fig. 1).

If we can solve the above problem exactly, we can solve the decision version of the cutting-sticks problem too. Since the latter problem is NP-hard, we will look for an approximation algorithm. We give a polynomial time algorithm with approximation factor $\sqrt{2}$ for the above problem. In other words, suppose OPT is the smallest t for which we can fit the given sticks into $[t]$ boxes. Our algorithm outputs a number that is at most $\sqrt{2}OPT$ and gives a way to fit the sticks into $[\sqrt{2}OPT]$ boxes.

2. Greedy algorithm

Assumption. We make a simplifying assumption that we know the value of OPT prior to the start of the algorithm. This assumption can be removed by binary searching for the minimum value for which the algorithm returns a solution. We defer the details to Section 4.5.

Informally, our algorithm works as follows. At each step, we pick the longest stick and either cut from one end of stick and place the piece into the largest empty box or place the stick itself into the box. The exact algorithm is given below. The parameter α denotes the approximation factor of the algorithm whose value will be fixed to $\sqrt{2}$ with hindsight.

Fig. 2 shows the run of the algorithm on the input $l_1 = 6$, $l_2 = 5$, $l_3 = 4$ with $OPT = 5$. The example shows a bad case when the algorithm fails when run with $\alpha = 1$. The algorithm considers boxes in decreasing order of their sizes. In the first step, box b_5 is picked and a portion equal to length 5 is cut from the stick s_1. The remaining portion of the stick s_1 has length 1. In the second step, the box b_4 is picked and a piece of size 4 is cut from s_2 and placed in b_4. In the next step b_3 is picked and so on. The greedy algorithm fails to fit all the sticks if it starts with $\alpha = 1$. In the next section, we prove that if $\alpha \geq \sqrt{2}$, then the algorithm succeeds for any input.

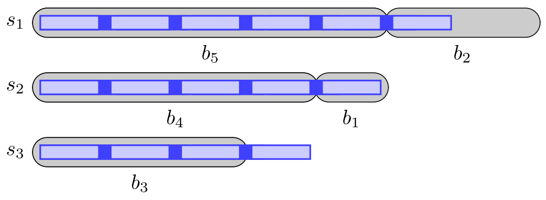

Fig. 2. An execution of the greedy algorithm with [5] boxes. Although the sticks fit into [5] boxes optimally, the greedy algorithm fails to do so: a portion of the stick s_3 is not fit into any box. It can be easily checked that if the greedy algorithm started with [6] boxes, then it would succeed in fitting all the sticks.

```
Initially, we have $[\alpha OPT]$ empty boxes and the sticks labelled $s_1, \dots, s_k$.

1. Pick the largest empty box available (say $b_i$).
   Note that box $b_i$ has size $i$. Pick the stick with
   the longest remaining length (say $s_j$). Let $len(s_j)$
   denote the current length of $s_j$.
   (a) If $len(s_j) \leq i$, then fit the remaining portion
       of the stick $s_j$ into $b_i$.
   (b) Otherwise, cut the stick $s_j$ from one end to
       get a piece of length $i$ and place this piece
       into the box $b_i$. The stick $s_j$ now has length
       equal to $len(s_j) - i$.
   In either case, box $b_i$ becomes non-empty after
   this step.
2. Repeat Step 1 until no empty box is available or
   until all the sticks are fit into boxes.
```

3. A lower bound on *OPT*

We are given an input with the guarantee that all the sticks will fit into $[OPT]$ boxes. To prove an approximation ratio, we first need a lower bound on the value of OPT. We express lower bounds on OPT in terms of two quantities of the input: number of sticks and the total length of the sticks.

Claim 3.1. *The number of sticks is at most OPT.*

Proof. Every stick requires at least one box and there are only OPT boxes. □

Notation. Let $\langle r \rangle$ denote the quantity $r(r+1)/2$.

Claim 3.2. *The total length of all the sticks is at most $\langle OPT \rangle$.*

Proof. All the sticks fit into $[OPT]$ blocks. The total size of boxes is $\langle OPT \rangle$. □

4. Analysis

We prove that the above algorithm has an approximation ratio of $\sqrt{2}$. We first discuss a weaker result that brings out a few aspects of our actual analysis.

Definitions. A *partially filled box* is one which contains a piece of stick that is *strictly* smaller than its size (the