Discrete Optimization 13 (2014) 1–15

Contents lists available at ScienceDirect

Discrete Optimization

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

# The online $k$-server problem with rejection[☆]

E. Bittner[a], Cs. Imreh[a,*], J. Nagy-György[b]

[a] *Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary*

[b] *Department of Mathematics, University of Szeged, Aradi Vértanúk tere 2, H-6720 Szeged, Hungary*

ARTICLE INFO

*Article history:*
Received 20 November 2011
Received in revised form 14 March 2014
Accepted 17 March 2014
Available online 16 April 2014

*Keywords:*
Online algorithms
Competitive analysis
$k$-server problems

ABSTRACT

In this work we investigate the online $k$-server problem where each request has a penalty and it is allowed to reject the requests. The goal is to minimize the sum of the total distance moved by the servers and the total penalty of the rejected requests. We extend the work function algorithm to this more general model and prove that it is $(4k-1)$-competitive. We also consider the problem for special cases: we prove that the work function algorithm is 5-competitive if $k=2$ and $(2k+1)$-competitive for any $k \geq 1$ if the metric space is the line. In the case of the line we also present the extension of the double-coverage algorithm and prove that it is $3k$-competitive. This algorithm has worse competitive ratio than the work function algorithm but it is much faster and memoryless. Moreover we prove that for any metric space containing at least $k+1$ points no online algorithm can have smaller competitive ratio than $2k+1$, and this shows that the work function algorithm has the smallest possible competitive ratio in the case of lines and also in the case $k=2$.


## 1. Introduction

The $k$-server problem can be formulated as follows. In the problem a metric space $\mathcal{M}$ is given with $k$ mobile servers that occupy distinct points of the space and a sequence of requests (points). Each of the requests has to be served, by moving one server from its current position to the requested point. The goal is to minimize the total cost, that is the sum of the distances covered by the $k$ servers. In the online version of the problem the requests arrive one by one and an online $k$-server algorithm serves each request immediately when it arrives, without any prior knowledge about the future requests. The online $k$-server problem has applications in planning maintenance service. The model can be also applied to upkeep or design of computer or sensor networks. In many of these applications it is a straightforward idea to allow the servers not to serve some of the requests. In this new model which is called $k$-server problem with rejection the $i$th request is a pair $\varrho_i = (r_i, p_i)$, where $r_i$ is a point of the space and $p_i > 0$ is the *penalty* for the rejection. Each request can be served the same way as in the classical $k$-server problem, or optionally it can also be rejected at the penalty given along with the request. The cost of an algorithm is the sum of the distances covered by the $k$ servers plus the sum of the penalties of the rejected requests. In this paper we study the online version of the $k$-server with rejection problem where the requests arrive one by one.

☆ This research was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency. Cs. Imreh was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. J. Nagy-György was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 National Excellence Program.

* Corresponding author. Tel.: +36 62 546715; fax: +36 62 546397.
*E-mail addresses:* bittner@inf.u-szeged.hu (E. Bittner), cimreh@inf.u-szeged.hu, imreh.csanad@gmail.com (Cs. Imreh), Nagy-Gyorgy@math.u-szeged.hu (J. Nagy-György).

http://dx.doi.org/10.1016/j.disopt.2014.03.002
1572-5286/

Typically, the quality of an online algorithm is judged using competitive analysis. An online algorithm for a minimization problem is asymptotically $c$-competitive if its cost is never more than $c$ times the optimal cost plus an additive absolute constant which is independent from the input. Without allowing the additive constant the algorithms are called competitive in the absolute sense. Here we use the asymptotic competitive ratio as it is usually done in the case of the online $k$-server problem.

*Related works:* The online $k$-server problem is one of the most known online problems. The problem is introduced in [1], where the first important results are presented. In [1] it is proved that $k$ is a lower bound for every metric space with at least $k+1$ points, and in [2] the work function algorithm is presented which is $(2k-1)$-competitive for every metric space. The $k$-server conjecture states that there exists an algorithm that is $k$-competitive for any metric space. The problem was also investigated for special metric spaces. In the case of uniform space, where the problem is equivalent to the paging problem, a $k$-competitive algorithm is given in [3]. If the metric space is a line then a $k$-competitive algorithm is given in [4].

The idea of allowing the algorithm to reject some parts of the input appeared in other online problems as well. The most closely related problem is online paging with rejection, which problem is the $k$-server problem with rejection for uniform space and studied in [5]. There a $(2k+1)$-competitive algorithm is presented and it is shown that no algorithm with smaller competitive ratio exists. The more general caching with rejection problem is also considered in [5]. A $(2k+1)$-competitive algorithm is given in the bit model, where the cost of loading a file is equal to its size and also in the fault model, where the cost of loading a file is equal to 1. Moreover, a $(2k+2)$-competitive algorithm is presented for the general caching problem.

The first online model with rejection appeared in online scheduling [6]. In this model the algorithm is allowed to reject the jobs and the objective is to minimize the sum of the penalties of the rejected jobs plus the makespan of the schedule of accepted jobs. This problem was further studied by introducing parameter learning online algorithm in [7]. After the first paper some further online scheduling models with rejection were investigated. In [8] the problem where it is allowed to preempt the jobs is considered, in [9] and [10] the online scheduling problem with rejection where the algorithm has to purchase the machines is investigated. Online bin packing with rejection, where it is allowed to reject the items and the cost is the sum of the number of used bins and the total penalty of the rejected items is investigated in [11] and [12]. Online graph colouring with rejection is studied in [13].

*Our results:* We extend the work function algorithm to this more general model where rejection is allowed and we prove that it is $(4k-1)$-competitive. We also consider the problem in special cases: we prove that the work function algorithm is 5-competitive if $k=2$ and $(2k+1)$-competitive for the line. We use similar techniques in the analysis of the work function algorithm to the techniques used in the case of the $k$-server problem without rejection. We prove the quasiconvexity of the work function and also a duality theorem. The difference comes from the fact that allowing rejection presents more possibilities in the changes of the work function and in the proofs of the lemmas and also during their application we have to handle these extra possibilities as well. The work function algorithm is very difficult and has large memory requirements, therefore we also analyse a simpler, memoryless algorithm in the case of the line. This is an extension of the double-coverage algorithm into the model with rejection, and we show that it is $3k$-competitive. Moreover we prove that for any metric space containing at least $k+1$ points no online algorithm can have a smaller competitive ratio than $2k+1$.

## 2. Notions and notations

For an arbitrary online algorithm $\mathcal{A}$ and an input sequence or subsequence $\varrho$ the cost of the solution given by $\mathcal{A}$ is denoted by $\mathcal{A}(\varrho)$. Moreover for an input sequence $\varrho$ let $OPT(\varrho)$ denote the cost of the optimal offline solution. Then an online algorithm $\mathcal{A}$ is called $c$-competitive if there exists a constant $b$ such that $\mathcal{A}(\varrho) \leq c \cdot OPT(\varrho) + b$ for any input sequence $\varrho$.

We will consider the online $k$-server problem with rejection problem as the special case of the metrical task system problem defined in [14]. Therefore first we recall the basic definitions and some fundamental results from the area of metrical task systems. In the metrical task system a finite metric space $\mathcal{S}$ of states is given. For notational convenience, if $z, y \in \mathcal{S}$, we denote the distance from $z$ to $y$ as $zy$. In the metrical task system problem we have to execute a sequence of tasks and the cost of executing a task depends on the state where we are. Our goal is to minimize the total cost which is the sum of the execution costs and the state transition costs defined by the distances in $\mathcal{S}$. Metrical task systems have many applications. They can model the functioning of machine which can be set to some states and its state influences its performance. Thus in a metrical task system problem we have an initial state $x_0 \in \mathcal{S}$ and a sequence of tasks $\overline{\tau} = \tau_1, \dots, \tau_m$ is given. Each task is a function from $\mathcal{S}$ to $R^+ \cup \infty$. We denote by $\bar{\tau}_i$ the prefix of the first $i$ tasks. Any sequence $\overline{x} = x_0, x_1, \dots, x_m$, where $x_i \in \mathcal{S}$, is called a schedule, or a service schedule for $x_0$. We use $\bar{x}_i$ to denote the prefix of the first $i+1$ elements of the schedule. We define

$$cost(x_0, \overline{\tau}, \overline{x}) = \sum_{i=1}^{m} (x_{i-1}x_i + \tau_i(x_i)).$$

Then the MTS problem is to find a service schedule with minimal cost. We will use $OPT(x_0, \overline{\tau})$ to denote this cost.

In the online MTS problem the tasks arrive one by one, and an online MTS algorithm has to execute each task changing into a selected state without any information about the further tasks. We will use the online work function algorithm which was developed for the solution of the online MTS problem. The work function is defined for every $i = 1, \dots, m$ for each