Operations Research Letters 44 (2016) 784–789

Contents lists available at ScienceDirect

Operations Research Letters

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

Minimizing the maximum flow time in batch scheduling

Sungjin Im [a,*], Hoon Oh [b], Maryam Shadloo [a]

[a] University of California at Merced, Merced, CA, United States
[b] Rutgers University-Camden, Camden, NJ, United States

ARTICLE INFO

Article history:
Received 27 May 2016
Received in revised form
29 September 2016
Accepted 29 September 2016
Available online 6 October 2016

Keywords:
Batch scheduling
Broadcast scheduling
Maximum flow time
Approximation
Resource augmentation

ABSTRACT

We consider the maximum flow time minimization problem in batch scheduling, which is a capacitated version of broadcast scheduling. In this setting, n different pages of information are available at the server which receives requests from clients over time for specific pages. The server can transmit at most one page p at each time to satisfy a batch of requests for the same page p, up to a certain capacity B_p. In this paper we give the first $(1+\epsilon)$-approximations for this problem with arbitrarily small resource augmentation, using either more capacity or more speed.

1. Introduction

In batch scheduling, there is a server that stores n different unit-sized pages of information. Each client submits to the server a request ρ at time r_ρ asking for a specific page p_ρ. The server can transmit at most one page p at each time to satisfy up to B_p outstanding requests of the same page p simultaneously; the capacity B_p can be different for each page. Processing a batch of requests together is a popular method to increase the server's throughput. Not surprisingly, batch scheduling appears in various forms in numerous applications, not only in server–client scheduling, but also in manufacturing lines; for pointers of the applications, see [7].

Broadcast scheduling is a special case of batch scheduling which has received considerable attention in theoretical computer science. The only difference is that in broadcast scheduling there is no limit on the number of requests the server can aggregate at a time, i.e. $B_p = \infty$ for all p. In other words, batch scheduling is a capacitated version of broadcast scheduling. However, as discussed in [2], capacities are often present in practice. For example, there could be a limit on the number of clients a server can serve at a time.

What makes batch/broadcast scheduling algorithmically challenging is that the scheduler could aggregate more requests by waiting for other requests arriving in the future for the same page. While the server can increase throughput by doing so, it makes earlier arriving requests wait longer, thereby making the clients submitting those requests unhappy. Such a tradeoff becomes more challenging in batch scheduling since the scheduler also has to factor in batch sizes. A request ρ's flow time is defined as its completion time C_ρ minus its arrival time r_ρ and measures how long the request waits since its arrival until its completion time. When requests compete to get served earlier, a popular way of combining the flow time of individual requests is to consider flow time objectives such as total flow time or the maximum flow time.

In this paper, we study the objective of minimizing the maximum flow time, i.e. $\max_\rho(C_\rho - r_\rho)$ in the batch scheduling setting. In broadcast scheduling, First-In-First-Out (FIFO) is known to be a 2-approximation [3,5]. At each time, the algorithm FIFO transmits the page of an outstanding request with the earliest arrival time; notice that FIFO is in fact an online algorithm since it does not need to know requests arriving in the future. It was subsequently shown that no online algorithms can be better than 2-competitive [3,4]. The open question whether there exists a better than 2-approximation was recently answered in [7], which gave a PTAS using a variant of α-point rounding and dynamic programming (DP). The work in [7] essentially closed the complexity of the problem in broadcast scheduling since the problem was already known to be strongly NP-hard [3].

The main goal of this paper is to understand the complexity of the maximum flow time objective in batch scheduling, which captures capacity constraints commonly appearing in practice. A recent work shows that FIFO is still 2-competitive in batch

* Corresponding author.
E-mail addresses: sim3@ucmerced.edu (S. Im), ho62@rutgers.edu (H. Oh), mshadloo@ucmerced.edu (M. Shadloo).

http://dx.doi.org/10.1016/j.orl.2016.09.016
0167-6377/

scheduling [6] as it is in broadcast scheduling. Our work started from the question if there exists a better than 2-approximation in batch scheduling.

1.1. Our result

Our main result is the first $(1+\epsilon)$-approximations with arbitrarily small resource augmentation. We consider two types of resources augmented, capacity and speed. In the capacity augmentation model, the algorithm is allowed to satisfy up to $(1+\delta)B_p$ requests of page p by one transmission of page p and is compared against the optimal scheduler subject to the original capacity B_p for every p. In this model, if the algorithm's objective is at most c times the optimum for all inputs, we say that the algorithm is a $(1+\delta)$-capacity c-approximation. We believe that capacity augmentation model is reasonable since capacities are specified only approximately in practice when capacities are large—if all capacities are constants, we obtain a PTAS without any resource augmentation; see Section 2.2.

In the speed augmentation model, both the algorithm and the optimal scheduler are subject to the same capacities, but the algorithm is given an extra speed. If it is given $1+\delta$ speed, it is allowed to make one additional transmission than the optimal scheduler in every $\lfloor 1/\delta \rfloor$ time steps. In this model, we say the algorithm is a $(1+\delta)$-speed c-approximation if the algorithm's objective is at most c times the optimum for all inputs. Speed augmentation is widely considered in the scheduling literature [8].

Theorem 1. *Let m denote the number of requests. For minimizing the maximum flow time in batch scheduling, for any $\epsilon > 0$ and $\delta > 0$, we have the following approximations:*

1. *[Section 2.3] a $(1+\delta)$-capacity $(1+\epsilon)$-approximation with running time $m^{O\left(\frac{1}{\epsilon^4\delta^2}\right)}$; and*
2. *[Section 2.4] a $(1+\delta)$-speed $(1+\epsilon)$-approximation with running time $m^{O\left(\frac{1}{\epsilon^3\delta}\cdot\log(1/(\epsilon\delta))\right)}$.*

We also show how to obtain a quasi-polynomial time approximation scheme (QPTAS) without using any extra resources in Section 2.1. Currently, we do not know how to obtain a true PTAS, which we leave as an open problem.

1.2. Overview of our approach

At a high level, we closely follow the PTAS framework used in [7] for broadcast scheduling, which combines DP and a variant of α-point rounding. We discuss how we modify each part to obtain our result in batch scheduling. We first discuss the rounding part. The rounding part is used when the optimum, opt is large, say $\text{opt} \geq \Omega(\log m)$ where m is the number of requests. A standard linear programming (LP) used in broadcast scheduling is the following: variable $x_{p,t}$ denotes how much page p is transmitted at time t, and we need constraints that (i) page p must be transmitted within opt time steps after every time the page is requested, and (ii) at most one page can be transmitted at each time. In batch scheduling, we need to add more constraints to factor in capacity constraints. For every page p and every interval $I = [t_1, t_2]$, we ensure that (i′) at least $\lceil m_{p,I}/B_p \rceil$ transmissions are made for page p during $[t_1, t_2 + \text{opt}]$, where $m_{p,I}$ is the number of requests made during I for page p. This new constraint, together with (ii), turns out to be necessary and sufficient conditions for a feasible integral solution to correspond to a schedule with the maximum flow time at most opt.

We use the same variant of the α-point rounding used in [7]. We give a quick overview of the rounding scheme explaining how it works well with the new LP constraint. After solving the LP, we obtain a fractional solution $\{x_{p,t}\}$ and would like to round it. A standard α-point rounding picks a random value α_p from [0, 1] uniformly and independently for each page p, then attempts to transmit page p at the first time t when the accumulative transmission of page p, i.e. $\sum_{t' \leq t} x_{t'p}$ becomes greater than α_p plus each non-negative integer. Note that a new transmission of page p is made before the LP solution accumulates another unit of transmissions of page p. Hence due to the constraint (i), we obtain a temporary schedule with the maximum flow time at most opt. However, the temporary schedule may be infeasible since it may make too many transmissions during a short interval, which translates into a large increase of the objective when it is converted into a feasible schedule. Roughly speaking, a large number of pages/random variables result in a large variance in congestion. To overcome this issue, [7] partitioned pages into $O(\text{opt})$ groups and used only one random variable of the maximum value at most 1 for each group, therefore was able to have a small congestion over all intervals w.h.p. The new rounding kept the key property that a new transmission of page p is made before the LP solution accumulates another unit of transmissions of page p. Thus, the new constraint (i′) we use for batch scheduling ensures that the rounding scheme makes enough transmissions in the temporary schedule while using a small number of random variables. The rounding can be de-randomized using the method of pessimistic estimators [9].

We now discuss the DP part which is used when $\text{opt} = O(\log m)$. It is easy to see that in an optimal solution, the number of distinct pages of requests alive at a time is at most opt. Using this observation, if $B_p = O(1)$ for all p, one can obtain an optimal schedule via a DP that keeps track of the number of outstanding requests for each page. In the capacity augmentation model, by adding some dummy requests and using an appropriate scaling, we reduce the general problem to the case where all capacities are constants. In the speed augmentation model, we use a different idea. We make an extra transmission of page p before we have too many possibilities for the number of alive requests of the page p—the transmission is used to simplify the number. Here we carefully decide which pages to transmit using extra speed since we are allowed to make only one extra transmission in every $1/\delta$ time steps.

1.3. Related work

In batch scheduling, [6] studied online algorithms for flow time objectives. Specifically, [6] showed $O(1)$-speed $O(1)$-capacity $O(1)$-competitive algorithms for the total flow time objective and the more general ℓ_k-norms of flow time. As mentioned before, [6] also showed that FIFO is 2-competitive for the maximum flow time objective. For the best offline results on flow time objectives in broadcast scheduling, see [1,7].

1.4. Notation and organization

Let $T := \max_\rho r_\rho + m$ be the last time we need to consider in our schedule. In other words, any 'reasonable' algorithm can complete all requests by time T. As observed in [7], one can assume w.l.o.g. that $T = O(m^2)$. This is because if there is an idle time period of length more than m, one can break the instance into two disjoint instances since the earlier arriving requests can be satisfied by any reasonable algorithm before the other requests arrive. We will show algorithms with running time polynomial (or quasi-polynomial) in m and T, which will imply the desired results.

For notational convenience, we will use a model that is slightly different from but equivalent to the previously studied models. At each time, first a set of requests arrive, and then a page is transmitted to satisfy requests that have arrived but have not been