Minimizing Average Response Time in Broadcast Scheduling

Ravishankar Krishnaswamy[*]

Abstract

In this chapter, we survey the broadcast scheduling problem with the objective of minimizing the *average response time* of the requests. Formally, there is a single server that can hold n pages of unit size, and multiple requests for these pages arrive over time. At each time slot the server can broadcast one page which satisfies all the outstanding requests for this page at that time. The goal is to find a schedule to minimize the average response time of the requests, i.e. the duration since a request arrives until it is satisfied.

1 Introduction

In this chapter, we consider the classical broadcast scheduling problem and discuss some recent advances on this problem and also present some algorithmic ideas. The problem is formalized as follows: there is a server which has a collection of unit-sized *pages* $P = \{1, \ldots, n\}$. The server can broadcast pages in integer time slots in response to *requests*, which is given as the following sequence: at time t, the server receives $w_p(t) \in \mathbb{Z}_{\geq 0}$ requests for each page $p \in P$. We say that a request ρ for page p that arrives at time t is satisfied at time $c_p(t)$ if $c_p(t)$ is the first time after t by which the server has completely transmitted page p. The response time of the request ρ is defined to be $c_p(t) - t$, i.e., the time that elapses from its arrival till the time it is satisfied. Notice that by definition, the response time for any request is at least 1. The goal is to find a schedule for broadcasting pages to minimize the average response time, i.e. $(\sum_{t,p} w_p(t)(c_p(t) - t)) / \sum_{t,p} w_p(t)$. Recall that the problem we discuss here is an *offline problem*, where the entire request sequence is specified as part of the input. There has also been much research on the online version of the problem, and we briefly discuss this at the end of the chapter.

2 Algorithmic Ideas

Recall that in the offline broadcast scheduling problem, the algorithm knows the entire request sequence ahead of time, and needs to compute the best broadcast sequence to minimize the average response time of the requests. Erlebach and Hall [5] were the first to show complexity theoretic hardness for this problem by showing that it is NP-complete. We now survey the state-of-the-art algorithmic results and techniques for this problem.

All of the algorithmic results in [1, 3] and [2] are based on rounding the following natural LP relaxation for the problem. For each page $p \in [n]$ and each time t, there is a variable y_{pt} which indicates whether page p was transmitted at time t. We have another set of variables $x_{ptt'}$ s.t $t' > t$, which indicates whether a request for page p which arrives at time t is satisfied at t'. Let $w_p(t)$ denote the total weight of requests for page p

[*]Princeton University `ravishan@cs.cmu.edu`.

that arrive at time t.

$$\min \sum_{p,t,t'>t} (t'-t) \cdot w_p(t) \cdot x_{ptt'} \tag{1}$$
$$\text{s.t.} \sum_p y_{pt} \le 1 \qquad \forall t \tag{2}$$
$$\sum_{t'>t} x_{ptt'} \ge 1 \qquad \forall p,t \tag{3}$$
$$x_{ptt'} \le y_{pt'} \qquad \forall p,t,t' \ge t \tag{4}$$
$$x_{ptt'}, y_{pt'} \in [0,1] \qquad \forall p,t,t' \tag{5}$$

Constraint (2) ensures that only one page is transmitted in each time, (3) ensures that each request must be satisfied, and (4) ensures that a request for page p can be satisfied at time t only if p is transmitted at time t. Finally, a request arriving at time t that is satisfied at time t' contributes $(t'-t)$ to the objective. Now consider the linear program obtained by relaxing the integrality constraints on $x_{ptt'}$ and y_{pt}.

The techniques we describe below and the framework were introduced in [1, 3]. By fine-tuning these ideas, [2] shows the following bound on the integrality gap (and approximability) of the offline problem.

Theorem 1 ([2]) *Let $\gamma > 0$ be any arbitrary parameter. There is a polynomial time algorithm that finds a schedule with average response time $(2+\gamma) \cdot OPT + O((\sqrt{\log_{1+\gamma} n \cdot \log\log n}) \log n)$, where OPT denotes the value of the average response time in the optimum LP solution.*

By setting $\gamma = \Theta(\log n)$ above, we can get an approximation guarantee of $O(\log^{1.5} n)$. Also note that the $O(\log^{1.5} n)$ term in Theorem 1 is additive. As a result, for instances where OPT is large (say $\Omega(\log^{1.5+\epsilon} n)$ for some), we can set γ arbitrarily small to get an approximation ratio arbitrarily close to 2.

2.1 Rounding Techniques

The following points illustrate the main ideas that form the building blocks of the rounding algorithms in [1, 3, 2].

The Half-Integrality Assumption. In what follows, we discuss the techniques in the special case that the LP solution is *half-integral*, where all the $x_{ptt'} \in \{0, \frac{1}{2}\}$. The general case essentially builds upon this intuition, and all main technical ingredients are contained in this special case.

Viewing the LP solution as a convex combination of blocks. In half-integral solutions, note that every request is satisfied by the two earliest half-broadcasts of the corresponding page. For any page p, let $\tau_p = \{t_{p,1}, t_{p,2}, \ldots\}$ denote the times when the fractional solution broadcasts $\frac{1}{2}$ units of page p. Notice that the fractional solution can be entirely characterized by these sets τ_p for all pages p. The main intuition now is to view the fractional broadcast of each page as a *convex combination* of two different solutions, one which broadcasts the page p integrally at the odd times $\{t_{p,1}, t_{p,3}, \ldots\}$ and another which broadcasts the page p integrally at the even times $\{t_{p,2}, t_{p,4}, \ldots\}$. We call these the *odd schedule* and *even schedule* for page p.

Rounding the solution to minimize backlog: Attempt 1 [1]. The next step is to round the convex combination for each page into one of the odd or even schedules. Indeed, one such approach is to pick each schedule with probability $1/2$. Let us call this the *tentative schedule*. Note that on average, the tentative schedule broadcasts one page per time slot, and moreover, the expected response time of any request is equal to its fractional response time. The only issue however, is that different pages may broadcast at the same time slots. Indeed, there could some time interval $[t_1, t_2)$ where the tentative schedule makes many more than $t_2 - t_1$ broadcasts! A natural manner to resolve this issue is to broadcast conflicting pages in a first-come first-serve manner, breaking ties arbitrarily. Now, the typical request waits for at most its fractional cost (on average), plus the *backlog* due to conflicting broadcasts. Formally, the backlog is defined as $\max_{t_1,t_2>t_1} N_A(t_1,t_2) - (t_2 - t_1)$, where $N_A(t_1,t_2)$ is the number of broadcasts made in the interval $[t_1,t_2)$ by the tentative schedule. For this simple randomized algorithm, note that the backlog of any interval $[t_1,t_2)$ is at most $\widetilde{O}(\sqrt{n})$ w.h.p by a standard concentration bound. This can be formalized to give us the $\widetilde{O}(\sqrt{n})$ approximation algorithm of [1].

Rounding the solution to minimize backlog: Attempt 2 [3]. Our next attempt involves controlling the backlog by explicitly enforcing constraints which periodically reset the backlog to 0. For this, we write an auxiliary LP as follows: we divide the set τ_p for each page p into blocks of size $B = \Theta(\log n)$ units each, and have a variable for choosing the odd schedule or even schedule within each block[1]. Since each block has $B/2$ fractional transmissions (recall that the LP is half-integral), the total number of blocks is at most $2T/B$, where T is the time horizon of all broadcasts made by the LP solution. Therefore, the total number of variables is at most $4T/B$ (each block chooses either an odd schedule or even schedule). Now, instead of asking for the LP to choose schedules such that each time has at most one transmission, suppose we *group the time slots into intervals of size B and ask for B transmissions within each interval.* There are T/B such constraints, and there are $2T/B$ constraints which enforce that we pick one schedule for each block.

Therefore, in this relaxed LP, we start with a solution that has $4T/B$ variables each set to $1/2$ and a total of $3T/B$ constraints. But now, we can convert this into an *optimal basic feasible solution* (where the number of non-zero variables is at most the number of constraints). This implies that at least a constant fraction of the blocks chooses either the odd or even schedules integrally. It is easy to see that the backlog incurred in any time interval $[t_1,t_2)$ is at most $O(B)$ since we explicitly enforce 0 backlog for consecutive intervals of size B. Therefore, by repeating this process $O(\log T)$ time we get a fully integral schedule with backlog $O(\log TB) = O(\log^2 n)$. This then gives us the $2 \cdot OPT + O(\log^2 n)$ approximation guarantee of [3].

Rounding the solution to minimize backlog: Attempt 3 [2]. Our final attempt involves combining the ideas of attempts 1 and 2. Indeed, the main issue with approach 2 is that, when we solve for a basic feasible solution, we lose all control over how the solution looks! Therefore, we would ideally like for a rounding which enforces the constraints on time intervals of size B, but still *randomly selects the schedules within each block*. This way, we'll be able to argue that within each time interval of size B, the maximum backlog is $O(\sqrt{B})$. Moreover, if we look at a larger time interval I, we can decompose this into intervals of size B for which we have constraints in the LP, and a prefix and suffix of size at most B. Therefore, the backlog is constrained to be 0 by the LP for all intermediate intervals except the prefix and suffix which can have a backlog of $O(\sqrt{B})$. This will immediately give us the $O(\log^{1.5} n)$ approximation of [2].

Indeed, the main tool which lets us achieve this comes from a recent rounding technique of Lovett and Meka [7]. They prove the following result which they used as a subroutine for minimizing discrepancy of set-systems, but it turns out to be a general result applicable in our setting as well [2].

Theorem 2 (Constructive partial coloring theorem [7]) *Let $\mathbf{y} \in [0,1]^m$ be any starting point, $\delta > 0$ be an arbitrary error parameter, $v_1, \ldots, v_n \in \mathbf{R}^n$ vectors and $\lambda_1, \ldots, \lambda_n \geq 0$ parameters with*

$$\sum_{i=1}^{n} e^{-\lambda_i^2/16} \leq \frac{m}{16}\,. \tag{6}$$

Then there is a randomized $\widetilde{O}((m+n)^3/\delta^2)$-time algorithm to compute a vector $\mathbf{z} \in [0,1]^m$ with

[1]If the first block chooses an odd schedule and the second block chooses an even schedule, then the requests which arrived at the boundary may incur greater costs, but [3] argue that these can be bounded for $B \geq \Omega(\log n)$.

(i) $z_j \in [0,\delta] \cup [1-\delta, 1]$ *for at least* $m/32$ *of the indices* $j \in [m]$,

(ii) $|v_i \cdot \mathbf{z} - v_i \cdot \mathbf{y}| \leq \lambda_i ||v_i||_2$, *for each* $i \in [n]$.

3 Hardness Results

The authors [2] also complement the above algorithmic result with the following negative results.

Theorem 3 *The natural LP relaxation for the broadcast problem has an integrality gap of* $\Omega(\log n)$.

Interestingly, Theorem 3 is based on establishing a new connection with the problem of minimizing the discrepancy of 3-permutations. In the 3-permutation problem, we are given 3-permutations π_1, π_2, π_3 of $[n]$. The goal of the problem is to find a coloring χ that minimizes the discrepancy. The discrepancy of $\Pi = (\pi_1, \pi_2, \pi_3)$ w.r.t a ± 1 coloring χ is the worst case discrepancy of all prefixes. That is, $\max_{i=1}^{3} \max_{k=1}^{n} \left| \sum_{j=1}^{k} \chi(\pi_{i,j}) \right|$, where $\pi_{i,j}$ is the j^{th} element in π_i. Newman and Nikolov [8] showed a tight $\Omega(\log n)$ lower bound on the discrepancy of 3-permutations, resolving a long standing conjecture. The authors [2] note that this can be used to give an integrality gap for the broadcast scheduling problem as well. Then, by generalizing the connection to the discrepancy of ℓ-permutations, [2] shows the following hardness results (prior to this, only NP-hardness was known).

Theorem 4 *There is no* $O(\log^{1/2-\epsilon} n)$ *approximation algorithm for the problem of minimizing average response time, for any* $\epsilon > 0$, *unless* $\mathsf{NP} \subseteq \bigcup_{t>0} \mathsf{BPTIME}(2^{\log^t n})$. *Moreover, for any sufficiently large* ℓ, *there is no* $O(\ell^{1/2})$ *approximation algorithm for the* ℓ*-permutation problem, unless* $\mathsf{NP} = \mathsf{RP}$.

4 Online Broadcast Scheduling

The broadcast scheduling problem has also been studied in the *online scheduling model*, where the algorithm is made aware of requests *only when they arrive*, and it has to make the broadcast choices without knowledge of the future requests. Naturally, the performance of our algorithms degrade when compared to the offline model, but remarkably, we can get non-trivial algorithms even in the online model! The only additional assumption we need in the online model is that our scheduling algorithm may broadcast two pages (instead of one) every $1/\epsilon$ time slots, in order to get approximation ratios that depend on $1/\epsilon$. In particular, several $(1+\epsilon)$-speed, $O(\text{poly}(1/\epsilon))$ are now known [6, 4], and it is also known that extra speed is necessary to obtain $n^{o(1)}$ competitive algorithms.

References

[1] N. Bansal, M. Charikar, S. Khanna and J. Naor. *Approximating the average response time in broadcast scheduling.* Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005.

[2] N. Bansal, M. Charikar, R. Krishnaswamy and S. Li. *Better algorithms and hardness for broadcast scheduling via a discrepancy approach.* Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, 2014.

[3] N. Bansal, D. Coppersmith and M. Sviridenko. *Improved approximation algorithms for broadcast scheduling.* SIAM Journal on Computing, 38(3), pp. 1157-1174, 2008.

[4] N. Bansal, R. Krishnaswamy, and V. Nagarajan. *Better scalable algorithms for broadcast scheduling.* In Automata, Languages and Programming (ICALP), pages 324-335, 2010.

[5] T. Erlebach and A. Hall. *NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow.* Proc. 13th ACM-SIAM Symp. on Disc. Algorithms, pages 194-202, 2002.

[6] S. Im, and B. Moseley, *An online scalable algorithm for average flow time in broadcast scheduling.* ACM Transactions on Algorithms (TALG), 8(4), 39, 2012.

[7] S. Lovett, and R. Meka, *Constructive discrepancy minimization by walking on the edges*, In 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 61-67, 2012.

[8] A. Newman, and A. Nikolov, *A counterexample to Beck's conjecture on the discrepancy of three permutations.* arXiv preprint arXiv:1104.2922.