Broadcast Scheduling with Deadlines

Renars Gailis* Samir Khuller †

1 Introduction.

Broadcasting is an appropriate mechanism to disseminate data since multiple clients can have their requests satisfied simultaneously. A very large amount of work in the database and algorithms literature has focused on scheduling problems based on a broadcasting model [4, 3, 1, 2, 13, 6, 14](including several PhD theses from Maryland and Brown).Broadcasting is used in commercial systems, including the Intel Intercast System, and the Hughes DirecPC.We focus our attention on *pull-based* schemes, where clients request the data that they need (for example via phone lines) and the data is delivered on a fast broadcast medium (often using satellites).

A key consideration is the design of a good broadcast schedule. The challenge is in designing an algorithm that *guarantees good response time* [6]. While the practical problem is clearly online [3, 7], it is interesting to study the complexity of the offline problem as well. In fact, a lot of recent algorithmic work has focussed on minimizing the sum of completion times [11, 8, 10, 9] as well as for minimizing the maximum response time [6].

In trying to evaluate the performance of online algorithms, it is useful to compare them to an optimal offline solution. In addition, when the demands are known for a small window of time into the future (also called the look-ahead model in online algorithms) being able to quickly compute an optimal offline solution can be extremely useful. Many kinds of demands for data (e.g., web traffic) exhibit good predictability over the short term, and thus knowledge of requests in the immediate future leads to a situation where one is trying to compute a good offline solution.

The specific problem we study is that of *broadcast scheduling with deadlines* inspired by the work in [6]. The description of the problem is as follows. There are n data items, $1, \ldots, n$, called pages. Time is broken into "slots". A time slot is defined as the unit of time to transmit one page on the wireless channel. A request for a page j arrives at time t and specifies a deadline D_t^j. If page j is transmitted by the deadline, this request is satisfied (in other words, the job is scheduled within its "window"). Arrival times and deadlines of requests for pages are *all known in advance*. We wish to find a broadcast schedule that meets all deadlines. The algorithm proposed in [6] runs in polynomial time but it produces a fractional schedule; since we do not allow fractional schedules, the problems become quite different computationally. In fact, almost all the work in this area focuses on the version where the server has to make a decision as to which page to broadcast in each time slot.

The broadcast scheduling with deadlines problem is a decision problem that naturally leads to the following optimization problems: (a) Find a broadcast schedule that maximizes the number of requests that meet their deadlines (b) Find a minimum *delay factor* α such that there is a broadcast schedule in which every request (j, t) is satisfied by time $t + \alpha(D_t^j - t)$. For version (a) a $\frac{3}{4}$ approximation was given recently [9] (improving the bound of $\frac{1}{2}$ in [5]), however it was not known if this problem is NP-hard or not. In fact, for most broadcast scheduling problems, NP-hardness has been open (a notable exception is the recent result of [8]).

*Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : renars@cs.umd.edu.

†Research supported by NSF Award CCR-0113192. Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : samir@cs.umd.edu.

The problem of broadcast scheduling with deadlines arose as a sub-problem in trying to minimize the maximum response time [6]. By guessing the maximum response time (approximately), an instance of the deadline problem is created, for which a fractional solution can be obtained in polynomial time, by formulating it as an LP. This yields a polynomial approximation scheme for minimizing max response time when pre-emption is permitted.

We show the following results.

THEOREM 1.1. *The broadcast scheduling problem with deadlines is NP-complete.*

From this it follows that the broadcast scheduling problem with hard deadlines for which we wish to maximize the throughput, is NP-hard answering a question from [9]. This also clarifies the misconception that this problem can be solved in polynomial time, a result that some researchers thought was implied by the result in the paper by Bartal and Muthukrishnan [6] for the case when service times are unit. (The result in [6] is for arbitrary service times, but the schedule involves pre-emption at arbitrary times.)

For version (b) of the deadline scheduling problem, we can show the following:

THEOREM 1.2. *If there is a $(2-\epsilon)$ approximation for the broadcast scheduling problem with deadlines, then $P = NP$.*

Finally, we show the following algorithmic result.

THEOREM 1.3. *If there is a fractional solution which meets all the deadlines, then there is a 2-speed integral solution that meets all the deadlines.*

2 Outline of Proofs

Our proof of Theorem 1.1 is via a reduction from the well known vertex cover problem. Let $n = |V|$ and $m = |E|$. We create a data item corresponding to each vertex. For each edge $e_i \in E$ we create a block $B_i, (i = 0, \dots, m-1)$ of n consecutive time slots. We make n requests at the beginning of each block B_i at time $i \cdot n$ requesting *all* data items with deadline of $(i+1) \cdot n$. Notice, that to satisfy all these requests each data item will appear exactly once within each block. Suppose $e_i = (u, v)$. Then we create two more requests for items u and v with arrival time $i \cdot n + k - 2$ and deadline $i \cdot n + k + 1$.

Place all these blocks together, one after another. For each block at time $i \cdot n + k$ we make n more requests requesting all data items with deadline $(i+1) \cdot n + k$.

We claim that graph G has a vertex cover of size k if and only if the broadcasting scheduling instance described above has a feasible solution in which all deadlines are met.

Notice the following properties. To satisfy all n requests that are made at the beginning of any block, all blocks must contain a permutation of items, where each item appears exactly once. To satisfy all n requests that are made at time $i \cdot n + k$ the data items that appear in time interval $i \cdot n$ to $i \cdot n + k - 1$ must also appear in the same interval in next block.

Since the proof of Theorem 1.2 is more complex, it is omitted. It is also via a reduction from vertex cover.

We now outline the proof of Theorem 1.3. Consider the following IP formulation. We define a variable y_t^p that is 1 iff and only if page p is broadcast in time slot t. We add two sets of constraints. For every time slot t we have $\sum_p y_t^p \leq 1$. In addition, for each request (p, t) we have $\sum_{t'=t+1}^{D_t^p} y_{t'}^p \geq 1$. The constraints simply ensure that each at most one page is broadcast in each time slot and each request is satisfied by its deadline. Suppose that the relaxation has a feasible solution. It is possible to create instances when a fractional feasible solution exists, but no integral solution exists (S. Guha, personal communication). We show that we can derive an integral solution from a fractional solution where at most two pages are output in a time slot. (The version with max throughput ensures that at least a $\frac{3}{4}$ fraction of the requests are satisifed, while outputting only one page in each time slot [9].)

The basic idea is to compute for each request the first time when at least half the request has been satisfied fractionally. Let $ft(p,t)$ be the first time instance when $\frac{1}{2}$-fraction of request (p,t) get satisfied in the LP solution, i.e. $ft(p,t) = \min\{t'' | \sum_{t'=t+1}^{t''} y_{t'}^p \geq \frac{1}{2}\}$. We filter requests for page p as follows, transforming the instance I into a simpler instance $\mathcal{I}$ which has the following property. If $\mathcal{N}_p$ represents the times of requests for page p in $\mathcal{I}$ then for any times $\{t'_u, t'_v\} \subseteq \mathcal{N}_p$, such that $t'_u < t'_v$, we have $ft(p, t'_u) \leq t'_v$. This transformation is done separately for each page p.

This filtering is done by sorting all requests for page p based on arrival time and scanning them in decreasing order of arrival time. The first one is added to $\mathcal{I}$. When considering request (p,t), if $ft(p,t)$ overlaps the last added request in $\mathcal{I}$, we *delete* this request.

The effect is to produce a collection of requests in $\mathcal{I}$ for each page that have the property that they are able to get $\frac{1}{2}$ amount of page p without any sharing. By doubling the value of y_t^p (in each time slot) we are able to provide 1 unit of page p (fractionally) to each request. Finally, we can use network flows to convert this to a 2 speed solution where all requests in $\mathcal{I}$ are satisifed integrally. Clearly, all the requests in $\mathcal{I}$ are satisfied within their deadlines. For the remaining requests we can show that they too are satisfied. The main reason for this is that for a request that was dropped from $\mathcal{I}$, at least half the fractional service comes after the closest next request in $\mathcal{I}$. As a result, the closest request in $\mathcal{I}$, must get at least half its service before the deadline of the request that was dropped. Thus when we double the service times of all the requests in $\mathcal{I}$, the request that was dropped certainly gets one unit of service (fractionally) before its deadline. A network flow approach such as the one used in [10] can be be used to convert a fractional solution to an integral feasible solution.

3 Conclusion

Of interest is the special case of the broadcast scheduling with deadlines problem, in which all deadlines are "uniform", that is for every request (j,t), $D_t^j - t$ is identical. This special case is polynomially equivalent to problem of minimizing the maximum response time. The main interesting open problem is to obtain a 1-speed approximation algorithm with a factor < 2 for the problem of minimizing the maximum response time. (In [6] a PTAS was given via a reduction to the deadline problem, but this produces a fractional schedule. At the same time it was shown that a 2 approximation exists for the non-preemptive version.) We also do not know if minimizing the max response time is NP-hard or not.
Acknowledgments. We would like to thank Sudipto Guha and Sriram Pemmaraju for helpful discussions.

References

[1] S. Acharya, M. Franklin, and S. Zdonik. Dissemination-based data delivery using broadcast disks. In *IEEE Personal Communications*, 2(6), 1995.
[2] S. Acharya and S. Muthukrishnan. On demand broadcasts: new metrics and algorithms. In MOBICOM 1998.
[3] D. Aksoy, and M. Franklin. RxW: A scheduling approach for large-scale on-demand data broadcast. In *IEEE/ACM Transactions On Networking*, Volume 7, Number 6, 486-860, 1999.
[4] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. In SODA 1998.
[5] A. Bar-Noy, S. Guha, Y. Katz, J. Naor, B. Schieber and H. Shachnai. Throughput Maximization of Real-Time Scheduling with Batching. In SODA 2002.
[6] Y. Bartal and S. Muthukrishnan. Minimizing maximum response time in scheduling broadcasts. In SODA 2000.
[7] J. Edmonds and K. Pruhs. Multicast pull scheduling: when fairness is fine. In SODA 2002.
[8] T. Erlebach, A. Hall. NP-Hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. In SODA 2002.
[9] R. Gandhi, S. Khuller, S. Parthasarthy and A. Srinivasan. Dependent rounding in bipartite graphs. In FOCS 2002.
[10] R. Gandhi, S. Khuller, Y. Kim and Y-C. Wan. Algorithms for minimizing response time in broadcast scheduling. In IPCO 2002.
[11] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. In ESA 2000.
[12] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. In FOCS 1995.
[13] H. Kaplan, R. E. Tarjan, and K. Tsioutsiouliklis. Faster kinetic heaps and their use in broadcast scheduling. In SODA 2001.
[14] J. Wong. Broadcast Delivery. In *Proc. of the IEEE*, 76(12), 1988.