Equivalence of Two Linear Programming Relaxations for Broadcast Scheduling*

Samir Khuller and Yoo-Ah Kim

Abstract. A server needs to compute a broadcast schedule for n pages whose request times are known in advance. Outputting a page satisfies all outstanding requests for the page. The goal is to minimize the average waiting time of a client. In this paper we show the equivalence of two apparently different relaxations that have been considered for this problem.

Key words: scheduling, broadcasting, approximation algorithms, linear programming

1 Introduction

The informal 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 then waits. When page j has been transmitted, this request has been satisfied. Arrival times of requests for pages are known, and we wish to find a broadcast schedule that *minimizes the average waiting time.*

The work by Kalyanasundaram et al. [2] studies this problem. They showed that for any fixed $\epsilon, 0 < \epsilon \leq \frac{1}{3}$, it is possible to obtain a $\frac{1}{\epsilon}$-speed $\frac{1}{1-2\epsilon}$-approximation algorithm for minimizing the average response time, where a k-speed algorithm is one where the server is allowed to broadcast k pages in each time slot (we assume that $\frac{1}{\epsilon}$ is an integer). For example by setting $\epsilon = \frac{1}{3}$ they obtain a 3-speed, 3-approximation. The approximation factor bounds the cost of the k-speed solution compared to the cost of an optimal 1-speed solution. (This kind of approximation guarantee is also referred to as a "bicriteria" bound in many papers.) Note that they cannot set $\epsilon = \frac{1}{2}$ to get a 2-speed, constant approximation. Their algorithm is based on rounding a fractional solution for a "network-flow" like problem that is obtained from an integer programming formulation. The problem of minimizing the average response time has recently been shown to be NP-hard by Erlebach and Hall [1].

Subsequently, in [3] we considered a different Integer Program(IP) for this problem, and show that by relaxing this IP, for any fixed $\alpha \in (0, \frac{1}{2}]$, we can obtain a $\frac{1}{\alpha}$-speed solution that is a $\frac{1}{1-\alpha}$-approximation. For example, by setting

* Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Email:{samir,ykim}@cs.umd.edu.
Research supported by NSF Awards CCR-9820965 and NSF CCR-0113192.

$\alpha = \frac{1}{2}$ we obtain a 2-speed, 2-approximation. By setting $\alpha = \frac{1}{3}$ we obtain a 3-speed, 1.5-approximation. Note that our algorithm improves both the speed and approximation guarantee of the algorithm given in [2].

The main question motivating this work is as follows. Is the improvement due to a different IP formulation, or due to a different rounding procedure? Here we show that the two relaxations of the IP's are equivalent, and the improvement is simply due to a different rounding procedure.

2 Broadcast Scheduling Problem

The problem is formally stated in [2], but for the sake of completeness we will describe it here. There are n possible pages, $P = \{1, 2, \ldots, n\}$. We assume that time is discrete and at time t, any subset of pages can be requested. Let (p, t) represent a request for page p at time t. Let r_t^p denote the number of requests (p, t). Let T be the time of the last request for a page. Without loss of generality, we can assume that T is polynomially bounded as a function of n and the number of distinct times at which requests are made. This is because at each time when a request is made, we only need to broadcast as many pages after that as the number of distinct pages requested. A time slot t is the window of time between time $t-1$ and time t. We also have a k-speed server that can broadcast up to k pages during any time slot. We say that a request (p, t) is satisfied at time S_t^p, if S_t^p is the first time instance *after* t when page p is broadcast. In this paper, we work in the offline setting in which the server is aware of all the future requests. Our goal is to schedule the broadcast of pages in a way so as to minimize the total response time of all requests. The total response time is given by

$$\sum_p \sum_t r_t^p (S_t^p - t) \ .$$

The broadcast scheduling problem is described as an Integer Program (IP1) by [2]. This IP is then relaxed to obtain a Linear Program (LP1), which is then solved optimally in polynomial time. From the solution to the linear program, a solution is derived which is a 3-speed, 3-approximate integral solution.

For the same broadcast scheduling problem, an alternate Integer Program (IP2) is given in [3]. Again this is relaxed to obtain another Linear Program (LP2) which is solved optimally. From this solution to the linear program, a 2-speed, 2-approximate integral solution is derived.

While the costs of the two IP's are clearly the same, it is not clear if the corresponding relaxations are equivalent or not. In fact, it might appear at first glance that the improvements crucially depend on the alternate IP formulation. However, we show that this is not the case and in fact the integrality gap of the two IP's is the same since the cost of the two linear programs is the same.

3 Formulation of KPV

In this section we describe the Integer Program (IP) for the formulation in [2]. We show that the relaxation obtained by this formulation is equivalent to the relaxation obtained by the formulation considered by [3].

Let T be the last time at which a page was requested. For each page p and each time slot $0 \leq t \leq T+n$ there is a vertex $v_p(t)$. There is also a source vertex s and a destination vertex d. For ease of notation, in [2], s is referred to as $v(-1)$ and d as $v(T+n+1)$. There is a directed edge $e_p(-1,0)$ from s to each $v_p(0)$ of unit capacity and 0 cost. For each $0 \leq t < t' \leq T+n$ there is a directed edge $e_p(t,t')$ from $v_p(t)$ to $v_p(t')$ of unit capacity and cost $w_p(t,t')$ where $w_p(t,t') = \sum_{j=t}^{t'-1}(t'-j)r_j^p$. For each $T+1 \leq t \leq T+n$ and for each p there is a directed edge $e_p(t,T+n+1)$ from $v_p(t)$ to d with unit capacity and 0 cost.

There are flow variables $f_p(t,t')$. A value of 1 for this variable indicates that we broadcast page p at time t and then again at time t' (and not in between).

IP1 is described as follows.

$$\min \quad \sum_{t=0}^{T+n} \sum_{t'=t+1}^{T+n} \sum_{p=1}^{n} w_p(t,t') f_p(t,t') \tag{1}$$

subject to

$$\sum_{p=1}^{n} f_p(-1,0) = n \tag{2}$$

$$\sum_{t'<t} f_p(t',t) = \sum_{t'>t} f_p(t,t') \quad 1 \leq p \leq n,\ 0 \leq t \leq T+n \tag{3}$$

$$\sum_{t<t'} \sum_{p=1}^{n} f_p(t,t') \leq 1 \qquad 1 \leq t' \leq T+n \tag{4}$$

$$f_p(t,t') \in \{0,1\} \quad 1 \leq p \leq n,\ 0 \leq t \leq t' \leq T+n \tag{5}$$

$$f_p(t,n+T+1) \in \{0,1\} \quad 1 \leq p \leq n,\ T+1 \leq t \leq T+n \tag{6}$$

They relax this IP to an LP, by replacing the integrality constraints by interval constraints (the variables are required to take values in the range $[0,1]$). (We cannot use min cost flows to solve this LP, as there are constraints on the total incoming flow into a set of vertices.)

4 Formulation of GKKW

The Broadcast Scheduling Problem can be formulated as an integer program as follows. The binary variable $y_{t'}^p = 1$ iff page p is broadcast at time t'. The binary variable $x_{tt'}^p = 1$ iff a request (p,t) is satisfied at time $t' > t$ i.e., $y_{t'}^p = 1$ and $y_{t''}^p = 0, t < t'' < t'$. The constraints (8) ensure that whenever a request (p,t) is satisfied at time t', page p is broadcast at t'. Constraints (9) ensure that every request (p,t) is satisfied at some time $t' > t$. Constraints (10) ensure that at most one page is broadcast at any given time.

IP2 is described as follows.

$$\min \sum_p \sum_t \sum_{t'=t+1}^{T+n} (t'-t) \cdot r_t^p \cdot x_{tt'}^p \tag{7}$$

subject to

$$y_{t'}^p - x_{tt'}^p \geq 0, \quad \forall p, t, t' > t \tag{8}$$

$$\sum_{t'=t+1}^{T+n} x_{tt'}^p = 1, \quad \forall p, t \tag{9}$$

$$\sum_p y_{t'}^p \leq 1, \quad \forall t' \tag{10}$$

$$x_{tt'}^p \in \{0,1\}, \quad \forall p, t, t' \tag{11}$$

$$y_{t'}^p \in \{0,1\}, \quad \forall p, t' \tag{12}$$

The corresponding linear programming (LP) relaxation can be obtained by letting the domain of $x_{tt'}^p$ and $y_{t'}^p$ be $0 \leq x_{tt'}^p, y_{t'}^p \leq 1$.

5 Equivalence of the two formulations

We first show that given a fractional solution to LP1, we can derive a fractional solution to LP2 with the same cost.

Let $f_{(p,t)}(*,t')$ denote the sum of flows that cross request (p,t) and go into t'. That is, $f_{(p,t)}(*,t') = \sum_{i \leq t} f_p(i,t')$. Since request (p,t) contributes to $w_p(i,t')$ by $(t'-t) \cdot r_t^p$ when flow $f_p(i,t')$ crosses (p,t), we can rewrite the cost of LP1 as

$$\sum_p \sum_t \sum_{t'=t+1}^{T+n} (t'-t) \cdot r_t^p \cdot f_{(p,t)}(*,t').$$

We first prove the following two properties of an optimal solution of LP1, which are required to prove the equivalence.

Property 1. The sum of flows that cross request (p,t) equals one. In other words, $\sum_{t'=t+1}^{T+n} f_{(p,t)}(*,t') = 1$.

Property 2. Given two flows $f_p(i_1,j_1), f_p(i_2,j_2)$ with $i_1 < i_2$ in an optimal solution of LP1, we can assume that $j_1 \leq j_2$.

We have Property 1 since for each page p, the amount of flow coming out of the source is exactly one and flows are conserved. The following lemma will prove Property 2.

Lemma 1. *In an optimal solution of LP1, suppose that there are two flows $f_p(i_1,j_1), f_p(i_2,j_2)$ where $i_1 < i_2$ and $j_1 > j_2$. Without increasing the cost, we can convert it to a solution in which there are no such flows.*

Proof. When we select the nested pair of edges, we choose the earliest time i_1 and among flows starting from i_1, we select the edge with the latest time j_1. When we select the second edge among flows nested within (i_1, j_1), we choose the flow with the earliest time j_2. We pick one among all such edges.

We will argue that the change will not increase the number of flows nested by the flow starting from $i < i_1$. For the nested pairs starting at i_1, we will decrease the number of nested pairs of edges. That means that i_1 will never decrease and after a polynomial number of changes, i_1 will be increased and thus the procedure will terminate.

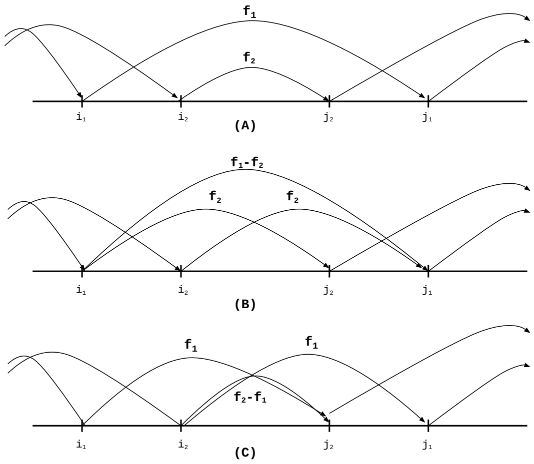

Fig. 1. Proof of Lemma 1

Let $f_1 = f_p(i_1, j_1), f_2 = f_p(i_2, j_2)$. For the case when $f_1 \geq f_2$, we split the flow $f_p(i_1, j_1)$ into two flows - send f_2 to j_2 and $f_1 - f_2$ to j_1. Instead of $f_p(i_2, j_2)$, we send a flow f_2 from i_2 to j_1. (see Figure 1(B)). For the case when $f_1 < f_2$, instead of $f_p(i_1, j_1)$, we send a flow f_1 from i_1 to j_2. We split the flow $f_p(i_2, j_2)$ into two flows - send $f_2 - f_1$ to j_2 and f_1 to j_1. See Figure 1(C). It is easy to see that the modified flow satisfies the constraints.

We show that this does not increase the cost. Request (p, t) where $t \geq i_2$ or $t < i_1$ are not affected by this change. For request (p, t) where $i_1 \leq t < i_2$, it was originally satisfied at time j_1 by f_1 but now it is satisfied at j_2 by f_2 (Case (B) in Figure 1) or by f_1 (Case (C) in Figure 1) and the remaining is satisfied at j_1 if any. This change only reduces the cost.

Note that because of the way we choose i_1, any flow from $i < i_1$ should go into $j \leq j_2$ Therefore, the modified flow cannot be nested by the flow coming from $i < i_1$. Apparently, the number of nested pairs within $f_p(i_1, j_1)$ is decreased by the change. Since the flows starting from i_1 (other than $f_p(i_1, j_1)$) go into $j < j_1$, the number of nested pairs does not increase. For the flows coming from i_1 *created by the change*, they cannot nest any flow since any flow starting from

i ($i_1 < i < j_2$) will go into $j > j_2$ (we chose the earliest time as j_2). Thus we have the lemma. □

For pages $p = 1 \ldots n$ and $t' = 1 \ldots T+n$ we define $y^p_{t'} = \sum_{t<t'} f_p(t,t')$. Once we define the y variables, the $x^p_{tt'}$ variables can be defined. Essentially, for any request (p,t) we can define $x^p_{tt'}$ as follows. Let $FT(p,t)$ be the first time instance when request (p,t) gets satisfied in the LP solution, i.e., $FT(p,t) = \min\{t'' | \sum_{t'=t+1}^{t''} y^p_{t'} \geq 1\}$. For $t < t' < FT(p,t)$ we have $x^p_{tt'} = y^p_{t'}$. For $t' = FT(p,t)$ we have $x^p_{tt'} = 1 - \sum_{t''=t+1}^{t'-1} y^p_{t''}$. For $t' > FT(p,t)$ we have $x^p_{tt'} = 0$.

We first show that all the contraints are satisfied with this solution and then argue that the cost of the solutions are preserved. Because of the way we define the x and y variables, Constraints (8) and (9) are satisfied. Constraints (10) are satisfied since $\sum_{t<t'} \sum_{p=1}^{n} f_p(t,t') = \sum_{p=1}^{n} \sum_{t<t'} f_p(t,t') = \sum_{p=1}^{n} y^p_{t'}$.

Lemma 2. *Given an optimal solution to LP1, we derive a fractional solution (x,y) to LP2 as described above. If we compute $f_{(p,t)}(*,t')$ from the solution of LP1, then we can prove that $x^p_{tt'} = f_{(p,t)}(*,t')$ for all requests (p,t) and $t' > t$.*

Proof. For $t' < FT(p,t)$,

$$f_{(p,t)}(*,t') = \sum_{i \leq t} f_p(i,t') = \sum_{i \leq t'} f_p(i,t') = y^p_{t'} = x^p_{tt'}$$

The second equality follows from the fact that $f_p(i,t') = 0$ for $t < i \leq t'$ since we have flow $f_p(t'', FT(p,t))$ for some $t'' \leq t$. (Otherwise, Property 2 is violated).

For $t' = FT(p,t)$, the remaining flow is $1 - \sum_{t''=t+1}^{t'-1} y^p_{t''}$ because of Property 1. All of this flow should go into t' since again it will violate Property 2 otherwise. □

We complete the discussion by showing how we can use a (fractional) solution for LP2 to derive a (fractional) solution for LP1. In fact we provide an algorithm to obtain the flow solution from the solution for the y^p_t variables.

First notice that our formulation stays the same even if we add variables $x^p_{tt'}$ for times t when no requests were made for a page p. In fact, we can simply make $r^p_t = 0$ so these variables do not contribute to the objective function.

Consider any particular page p and the y^p_t values. We show how to use these values to derive the flow variables $f_p(i,j)$. We define $excess_p(t)$ which is the excess flow at each time t for a page p. Define $f_p(-1,0) = 1$. We define the set $Excess_p$ as the (sorted) set of times t such that $excess_p(t) > 0$. Initialize $Excess_p = \{0\}$. The goal now is to move the excess flow to the right. At the same time we will ensure that the total incoming flow at time t does not exceed y^p_t, namely $\sum_{i<t} f_p(i,t) \leq y^p_t$. Pick up the first (smallest) time t in $Excess_p$. We move the flow out of t by saturating outgoing edges in order of increasing time to nodes which have $excess_p(t') < y^p_{t'}$. In other words, let t' be the first (smallest) time when the excess is smaller than the y value. We define $f_p(t,t')$ as the smaller of $excess_p(t)$ or $y^p_{t'} - excess_p(t')$. This moves (some of) the excess from t to t' and we continue the algorithm.

Note that the derived solution to LP1 also satisfies Property 1 and 2. We can see that Property 2 holds because in our algorithm we always pick up the first t in $Excess_p$ to send flows.

Lemma 3. *Given an optimal solution (x, y) to LP2, we derive a fractional solution to LP1 using the above algorithm. We also compute $f_{(p,t)}(*, t')$ from the derived solution of LP1. We claim that $x^p_{tt'} = f_{(p,t)}(*, t')$ for all requests (p, t) and $t' > t$.*

Proof. Consider the situation when $t < t' < FT(p, t)$ (In this case, $x^p_{tt'} = y^p_{t'}$). Since we send flow at most $y^p_{t'}$ to t', $f_{(p,t)}(*, t') \leq y^p_{t'}$. If $f_{(p,t)}(*, t') < y^p_{t'}$, it will violate Property 2.

Now let us consider $t' = FT(p, t)$. The remaining flow is $1 - \sum_{t''=t+1}^{t'-1} f_{(p,t)}(*, t'') = 1 - \sum_{t''=t+1}^{t'-1} y^p_{t''}$ (by Property 1). It should go into t' because of Property 2. □

Theorem 4. *The costs of LP1 and LP2 are the same.*

Proof. As mentioned earlier we can rewrite the cost of LP1 as

$$\sum_p \sum_t \sum_{t'=t+1}^{T+n} (t' - t) \cdot r^p_t \cdot f_{(p,t)}(*, t').$$

Given this formulation and the relationship with $x^p_{tt'}$ it is easy to see that this cost is the same as the cost of LP2. □

6 Algorithm to obtain a fractional solution

In this section we describe a fast combinatorial algorithm to obtain a fractional solution which is nearly optimal. Young [4] developed an algorithm that approximately solves linear programs with no negative coefficients. We can formulate the broadcast scheduling problem as linear program with no negative coefficients and apply the algorithm in [4].

Our formulation is similar to LP1 in the sense that it is considered as a type of min-cost max-flow problem. However, here we have a variable f_l for each valid path l from source to sink. For a set of paths F_t which visit time t (broadcast the page at time t), we have capacity constraints $\sum_{l \in F_t} f_l \leq 1$. We also have demand constraints $\sum_{l \in F_p} f_l \geq 1$ where F_p represents paths for each page p. We assume that the set of paths is $\cup_p F_p$. The cost w_l of f_l is the sum of edge costs in the path l where the cost of an edge is the same as in LP1. Therefore we want to minimize $\sum_p \sum_{l \in F_p} w_l f_l$.

As described in [4], we first reduce the optimization version to the feasibility problem and then solve the following problem by relaxing the constraints. We assume we are given a budget W.

$$\sum_{l \in F_t} f_l \leq 1 \quad \forall t \tag{13}$$

$$\sum_{l \in F_p} f_l \geq 1 - \epsilon \quad \forall \; page \; p \tag{14}$$

$$\sum_p \sum_{l \in F_p} w_l f_l \leq (1 + O(\epsilon))W \tag{15}$$

In this LP formulation, we have exponentially many paths but the algorithm in [4] only requires to compute the shortest path with edge weights given by a certain function in each step. Thus the time the algorithm takes is bounded by these shortest path computations. We need $O((T+n)\log(T+n)(\log\log(T+n)+1/\epsilon^2))$ shortest path computations for this problem [4].

After obtaining a fractional solution, we convert it to an integral solution using the rounding scheme in [3]. However, since we obtained the fractional solution by relaxing demand constraints (for each page, we broadcast only a $(1-\epsilon)$-fraction), we cannot obtain a 2-speed solution. Therefore we can obtain a $\frac{1}{\alpha}$-speed, $\frac{1}{1-\alpha}(1+O(\epsilon))$-approximation solution where $\alpha \leq \frac{1}{3}$. Specifically, when $\alpha = \frac{1}{3}$ we have 3-speed, $(1.5+O(\epsilon))$-approximation.

References

1. T. Erlebach, and A. Hall. NP-hardness of Broadcast Scheduling and Inapproximability of Single-source Unsplittable Min-cost Flow. In *Proc. of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 194-202, 2002.
2. B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling Broadcasts in Wireless Networks. In *European Symposium of Algorithms*, LNCS 1879, Springer-Verlag, 290-301, 2000.
3. R. Gandhi, S. Khuller, Y. Kim, and Y-C. Wan. Algorithms for Minimizing Response Time in Broadcast Scheduling. In *9th Integer Programming and Combinatorial Optimization Conference*, LNCS 2337, Springer-Verlag, 425-438, 2002.
4. N. Young. Sequential and Parallel Algorithms for Mixed Packing and Covering. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 538-546, 2001.