Dependent Rounding in Bipartite Graphs

Rajiv Gandhi* Samir Khuller† Srinivasan Parthasarathy‡ Aravind Srinivasan§

Abstract

We combine the pipage rounding technique of Ageev & Sviridenko with a recent rounding method developed by Srinivasan, to develop a new randomized rounding approach for fractional vectors defined on the edge-sets of bipartite graphs. We show various ways of combining this technique with other ideas, leading to the following applications:

- *richer random-graph models for graphs with a given degree-sequence;*
- *improved approximation algorithms for: (i) throughput-maximization in broadcast scheduling, (ii) delay-minimization in broadcast scheduling, and (iii) capacitated vertex cover;*
- *fair scheduling of jobs on unrelated parallel machines.*

A useful feature of our method is that it lets us prove certain (probabilistic) per-user fairness properties.

1. Introduction

Various combinatorial optimization problems include hard *cardinality constraints*: e.g., a broadcast server may be able to broadcast at most one topic per time step. Two of the known approaches to accommodate such constraints are the deterministic method of Ageev & Sviridenko [1], and a probabilistic method developed by Srinivasan [17]. In this work, we combine these two approaches to develop a dependent randomized rounding scheme (the term "dependent" underscores the fact that various random choices we make here are highly dependent). We then show applications to various problems. We start by describing the dependent rounding scheme.

Suppose we are given a bipartite graph (A, B, E) with bipartition (A, B). We are also given a value $x_{i,j} \in [0, 1]$ for each edge $(i, j) \in E$. We show a randomized polynomial-time scheme that rounds each $x_{i,j}$ to a random variable $X_{i,j} \in \{0, 1\}$, in such a way that the following properties hold.

(P1): Marginal distributions. For every edge (i, j), $\Pr X_{i,j} = 1 = x_{i,j}$.

(P2): Degree-preservation. Consider any vertex $i \in A \cup B$. Define its fractional degree d_i to be $\sum_{j:(i,j)\in E} x_{i,j}$, and integral degree D_i to be the random variable $\sum_{j:(i,j)\in E} X_{i,j}$. Then, we have $\Pr D_i \in \{\lfloor d_i \rfloor, \lceil d_i \rceil\} = 1$. Note in particular that if d_i is an integer, then $D_i = d_i$ with probability 1; this will often model the cardinality constraints in our applications.

(P3): Negative correlation. If $f = (i, j)$ is an edge, let X_f denote $X_{i,j}$. Then for any vertex i and any subset S of the edges incident on i:

$$\forall b \in \{0, 1\}, \ \Pr \bigwedge_{f\in S} (X_f = b) \leq \prod_{f\in S} \Pr X_f = b. \qquad (1)$$

The elegant pipage rounding method [1] is basically a deterministic relative of this scheme wherein "$D_i \in \{\lfloor d_i \rfloor, \lceil d_i \rceil\}$" and some other useful properties hold; the work of Srinivasan [17] is for the case where $G = (A, B, E)$ is a star, i.e., where $|A| = 1$. As mentioned above, our dependent rounding scheme combines these two earlier approaches to guarantee (P1), (P2) and (P3). When combined with several other ideas, it leads to the following applications. One noteworthy feature of some of these applications is that they offer (probabilistic) *per-user* guarantees. In general, for optimization problems with multiple users where each user has her/his own objectives, it is difficult to simultaneously satisfy all users. For a class of applications, our approach lets us provide guarantees of the form: each given user is satisfied with a certain (reasonable) probability.

(a) Random-graph models for massive graphs. There has been considerable interest recently in modeling massive

*Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965. E-mail: gandhi@cs.umd.edu.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This research was supported by NSF Award CCR-9820965 and CCR-0113192. E-mail: samir@cs.umd.edu.

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

§Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Supported in part by NSF Award CCR-0208005. E-mail: srin@cs.umd.edu.

graphs like the Web graph and the Internet graph (see, e.g., the talks at [8]); in particular, there has been much study of the *power law* property of the degree-sequences of such graphs (see, e.g., [6]). Random graph models for such graphs can let us simulate various algorithms on such massive graphs [7], and hence there has been much renewed interest in generating (and studying the properties of) random graphs with a given degree sequence; see, e.g., [10]. However, since a graph is far more than its degree sequence, we ask: can we model additional features of such graphs? Concretely, suppose empirical measurements show, e.g., that vertices of degree $\sqrt{n}$ are more likely to be linked to vertices of degree $n^{1/3}$, than to those of degree $\log n$. Can such features be modeled in our random graphs, in addition to satisfying a given degree sequence? We consider this problem in Section 2.1 and show that the problem can be completely solved for bipartite graphs; for general graphs, we show that some deviation from the degree sequence is inevitable in general, and show a deviation that is smaller than what is achievable by other methods that we are aware of.

(b) Approximation algorithms in broadcast scheduling. Traditional scheduling problems require each job to receive its own chunk of processing time. The growth of (multimedia) broadcast technologies has led to situations where certain jobs can be *batched* and processed together: e.g., all users waiting to receive the same topic in a broadcast setting, get satisfied when that topic is broadcast. In such a broadcast scheduling scenario where the server can broadcast one topic per time-step, two problems have received much attention recently [13, 9, 11, 2, 3]. In the first problem, given a set of users that arrive over time, each with its own desired topic, the server needs to schedule its broadcasts in order to minimize the total waiting time of the users. Define an α-speed solution to be one in which the server can broadcast up to α topics per time-step. Improving upon [13, 9], in [11], α-speed β-approximations are developed for this problem, for the (α, β)-pairs $(2, 2)$, $(3, 3/2)$, and $(4, 1)$. We improve the last two of these, to present a 3-speed 1-approximation. The β-approximation is obtained by comparing with a 1-speed optimal solution. A dual maximization problem in this context is: given a set of time-windows in which each of the jobs (topic-requests) can be processed, come up with a (1-speed) schedule which maximizes the total weight of satisfied jobs. These problems (and several generalizations) have been shown approximable to within $1/2$ [3]. For the case where each time-window is an interval, we develop a $p = 3/4$-approximation; for arbitrary time-windows, we present a $p = (1 - 1/e) \sim 0.632$–approximation. In both these cases, we can also achieve a form of per-user fairness: we present algorithms which, on every given instance, will either prove that there is no schedule that satisfies all users, or present a randomized schedule where each user is satisfied with probability at least p.

(c) Capacitated vertex cover. This is a generalization of the classical vertex cover problem on graphs $G = (V, E)$. Given an integral capacity k_v and cost w_v for each vertex, we want a function $x : V \rightarrow I\!N_0$ and an orientation of the edges, such that $\sum_v w_v x_v$ is minimized subject to the constraints that the number of edges directed into each vertex v is at most $k_v x_v$. This problem arises in the context of a testing application in the process of drug design (for details see [12]). A primal-dual 2-approximation algorithm and an LP-rounding based 4-approximation have been given for this problem [12]. We present a simple LP-rounding based 2-approximation; furthermore, when generalized to the version of this problem with assignment costs, we get a better approximation guarantee. Recently, Chuzhoy and Naor [5] addressed a variant of the capacitated vertex cover problem in which there is a bound b_v on x_v. In other words, a vertex v can be used at most b_v times to cover edges. They gave a 3-approximation for the unweighted case and showed that the weighted case is at least as hard as the set-cover problem. As a corollary to our rounding procedure, for the weighted version, we obtain a 2-approximate solution in which a vertex v is used at most $2b_v$ times to cover edges.

(d) Scheduling on unrelated parallel machines. One of the early LP-rounding results in scheduling is as follows [15]. Suppose we have a set of jobs and a set of machines. Each job j must be processed on some machine; processing it on machine i involves a load of $p_{i,j}$ on machine i. Suppose we wish to find a schedule that minimizes the *makespan*: the maximum total load on any machine. A 2-approximation, as well as a proof that a 1.5–approximation would imply $P = NP$, are shown in [15]. Since then, the approximability of this problem has been open. Chekuri and Khanna [4] show that it is possible to obtain a schedule in which the makespan is at most T but each job gets scheduled with a probability $1/2$. Their algorithm uses the algorithm in [15]. However, it is not clear if we can further boost the probability of each job being scheduled and achieve a makespan of less than twice the optimal. We present the following result. Let $\epsilon \in (0, 1)$ be an arbitrary constant, and T be the optimal makespan. We present randomized schedules in which the makespan is at most $(2 - \Omega(1))T$ with probability 1, with each given job getting scheduled with probability at least $1 - 1/e - \epsilon$. We view such probabilistic per-user guarantees as a useful dimension in multi-criterion optimization. Consider scenarios where we cannot exceed the optimal makespan significantly. In such a setting, one approach could be to schedule some constant fraction of the jobs; our type of result is stronger than this, and also guarantees each user a certain probability of getting what the user wants.

(e). A strengthening of (P3) for star-graphs. Recall that [17] guarantees (P1), (P2) and (P3) for star-graphs. For star-graphs, we show that dependent rounding enjoys a much stronger negative-correlation property than (P3), in Section 2.2.

Our applications come about by combining the dependent rounding with various other ideas. The per-user guarantees that we are able to provide, as well as application (a), are some of the novel features of our results; in particular, to our knowledge, these are not achievable through currently known approaches.

2. The Dependent Rounding Scheme

Suppose we are given the bipartite graph (A, B, E) and the values $x_{i,j}$ as in the introduction. Our dependent randomized rounding scheme is as follows. Initialize $y_{i,j} = x_{i,j}$ for each $(i, j) \in E$. We will probabilistically modify the $y_{i,j}$ in several (at most $|E|$) iterations such that $y_{i,j} \in \{0, 1\}$ at the end (at which point we will set $X_{i,j} = y_{i,j}$ for all $(i, j) \in E$). Our iterations will satisfy the following two invariants:

(I1) For all $(i, j) \in E$, $y_{i,j} \in [0, 1]$.

(I2) Call $(i, j) \in E$ *rounded* if $y_{i,j} \in \{0, 1\}$, and *floating* if $y_{i,j} \in (0, 1)$. Once an edge (i, j) gets rounded, $y_{i,j}$ never changes.

An iteration proceeds as follows. Let $F \subseteq E$ be the current set of floating edges. If $F = \emptyset$, we are done. Otherwise, find (in linear time) a simple cycle or *maximal* path P in the subgraph (A, B, F), and partition the edge-set of P into two matchings M_1 and M_2. Note that such a partition exists since (A, B, E) is a bipartite graph. Define

$$\begin{aligned} \alpha &= \min\{\gamma > 0 : ((\exists (i,j) \in M_1 : y_{i,j} + \gamma = 1) \bigvee \\ &\qquad (\exists (i,j) \in M_2 : y_{i,j} - \gamma = 0)); \\ \beta &= \min\{\gamma > 0 : ((\exists (i,j) \in M_1 : y_{i,j} - \gamma = 0) \bigvee \\ &\qquad (\exists (i,j) \in M_2 : y_{i,j} + \gamma = 1)). \end{aligned}$$

Since the edges in $M_1 \cup M_2$ are currently floating, it is easy to see that the positive reals α and β exist. Now, independent of all random choices made so far, we execute the following randomized step:

> With probability $\beta/(\alpha + \beta)$, set $y_{i,j} := y_{i,j} + \alpha$ for all $(i, j) \in M_1$, and $y_{i,j} := y_{i,j} - \alpha$ for all $(i, j) \in M_2$; with the complementary probability of $\alpha/(\alpha+\beta)$, set $y_{i,j} := y_{i,j} - \beta$ for all $(i, j) \in M_1$, and $y_{i,j} := y_{i,j} + \beta$ for all $(i, j) \in M_2$.

This completes the description of an iteration. A simple check shows that the invariants (I1) and (I2) are maintained, and that at least one floating edge gets rounded in every iteration.

Let us now analyze the above randomized algorithm. First of all, since every iteration rounds at least one floating edge, we see from (I2) that we need at most $|E|$ iterations. Let us now prove (P2) for a vertex i. If i had at most one floating edge incident on it at the beginning of our dependent rounding, it is easy to verify that (P2) holds; so suppose i initially had at least two floating edges incident on it. We claim that as long as i has at least two floating edges incident on it, the value $D_i^{(y)} \doteq \sum_{j:(i,j) \in E} y_{i,j}$ remains at its initial value of d_i. To see this, first note that if i is not in the cycle/maximal path P chosen in an iteration, then $D_i^{(y)}$ is not altered in that iteration. Next, consider an iteration in which i had at least two floating edges incident on it, and in which i was in the cycle/path P. Then, i must have degree two in P, and so, it must have one edge in M_1 and one in M_2. Then, since edges $(i, j) \in M_1$ have their $y_{i,j}$ value increased/decreased by the same amount as edges in M_2 have their $y_{i,j}$ value decreased/increased, we see that $D_i^{(y)}$ does not change in this iteration. Now consider the last iteration at the beginning of which i had at least two floating edges incident on it. At the end of this iteration, we will have $D_i^{(y)} = d_i$, and i will have at most one floating edge incident on it. It is now easy to note that (P2) holds.

Properties (P1) and (P3) can be proved by induction on the number of floating edges, inspired by the approach of [17]; we just give a sketch of the proofs here. Fix a vertex i and an edge $(i, j) \in E$; let $Y_{i,j,k}$ be the random variable denoting the value of $y_{i,j}$ at the beginning of iteration k. We will show by backward induction on k that

$$\forall (k, v), \;\; \Pr X_{i,j} = 1 \bigm| (Y_{i,j,k} = v) = v; \qquad (2)$$

we may then substitute $(k, v) = (1, x_{i,j})$ to see that (P1) holds. The base case where $k \geq |E| + 1$, is trivial. Assuming (2) for k, let us prove it for $k - 1$. Suppose $Y_{i,j,k-1} = v$ for some v. In iteration $k - 1$, either $Y_{i,j,k}$ is set to $Y_{i,j,k-1}$ with probability 1, or there are values α and β such that: $Y_{i,j,k} := v+\alpha$ with probability $\beta/(\alpha+\beta)$, and $Y_{i,j,k} := v-\beta$ with probability $\alpha/(\alpha + \beta)$. So by the induction hypothesis, $\Pr X_{i,j} = 1 \bigm| (Y_{i,j,k-1} = v)$ equals

$$(\beta/(\alpha + \beta)) \cdot (v + \alpha) + (\alpha/(\alpha + \beta)) \cdot (v - \beta) = v.$$

This completes our sketch of the proof for (P1). Property (P3) can be proven by a similar, slightly more complicated, induction.

2.1. Random-graph models for massive graphs

Recently, there has been growing interest in modeling the underlying graph of the Internet, WWW, and other such massive networks. If we can model such graphs using appropriate random graphs, then we can sample multiple times from such a model and test out candidate algorithms we have for such graphs, such as Web-crawlers. A particularly successful result of the study of such graphs has been the uncovering of the *power-law* behavior of the vertex-degrees of many such graphs (see, e.g., [6]); hence, there has been much interest in generating (and studying) random graphs with a given degree-sequence (see, e.g., [10]). Web/Internet measurements capture a lot of connectivity information in the graph, in addition to the distribution of the degrees of the nodes. In particular, through repeated sampling, these models capture the probability with which a node of a certain degree d_1 might share an edge with a node of degree d_2. Our question here is: since a network is much more than its degree sequence, can we model connectivity in addition to the degree-sequence? Concretely, given n, a degree-sequence $d_1, d_2, \ldots, d_n$, and values $\{x_{i,j} \in [0,1] : i < j\}$, we wish to generate an n-vertex random graph $G = (\{1, 2, \ldots, n\}, E)$ in which: (A1) vertex i has degree d_i with probability 1, and (A2) the probability of edge (i, j) occurring is $x_{i,j}$. (Note that we must have $d_i = \sum_j x_{i,j}$.) This is the problem we focus on, in order to take a step beyond degree-sequences.

It is easy to see that our dependent rounding scheme completely solves this problem, if G is bipartite. Next, can we get such a result for general graphs? Unfortunately, the answer is no: the reader can verify that no such distribution (i.e., random graph model) exists for the triangle with $x_{1,2} = x_{2,3} = x_{1,3} = 1/2$ (and hence with $d_1 = d_2 = d_3 = 1$). This example has $d_1 + d_2 + d_3$ being odd, which violates the basic property that the sum of the vertex-degrees should be even. However, even if the vertex-degrees add up to an even number, there are simple cases of non-bipartite graphs where there is no space of random graphs which satisfies (A1) and (A2). (Consider two vertex-disjoint triangles with all six $x_{i,j}$ values being $1/2$, and connect the two triangles by and edge whose $x_{i,j}$ value is 1.) Thus, we need to compromise – hopefully just a little – for general graphs. We show how to efficiently generate random graphs that preserve (A2), and relax (A1) to have *with probability one* that for each i, its (random) degree D_i satisfies $|D_i - d_i| \leq \lceil \log n \rceil$. The only other method we know of in this context is to construct a random graph where each edge (i, j) is put in *independently*, with probability $x_{i,j}$. This again preserves (A1), but does much worse for (A2): the high-probability guarantee we get here is that for each i, $|D_i - d_i| \leq O(\max\{\sqrt{d_i \log n}, \log n\})$.

Briefly, our algorithm for non-bipartite graphs is as follows. Arbitrarily partition the vertex set V into two sets A and B, whose sizes differ by at most one. Next, round all the edges $(i, j) \in A \times B$, using our bipartite rounding scheme. Finally, recursively (and independently) round the edges in the two subgraphs induced by A and B. (Also, in our constructions of both bipartite and non-bipartite graphs, the appropriate analog of (P3) holds.)

2.2. An extension of (P3) for star-graphs

Recall that the work of [17] shows how to guarantee (P1), (P2) and (P3) for star-graphs. For star-graphs, we now show that our dependent rounding in fact has a much stronger negative correlation property than (P3). For any vertex i, any pair of *disjoint* subsets S_1 and S_2 of the edges incident on i, and any $b \in \{0, 1\}$, we can show that $\Pr \bigwedge_{f \in S_2} (X_f = b) \mid (\bigwedge_{f \in S_1} (X_f = b))$ is upper-bounded by $\Pr \bigwedge_{f \in S_2} (X_f = b)$. It is easy to see that this implies (P3), but not vice versa. The proof is inspired by our proof for (P3), but is more involved and also uses the property that for star graphs, at least one floating edge incident on the center of the star gets rounded in every iteration. The proof is omitted from this version. We anticipate that this strong correlation property will lead to some interesting applications.

3. Broadcast Scheduling

In this section, we study two related scheduling problems in a broadcasting model, i.e., a model in which *all* users waiting to receive the same topic get satisfied when that topic is broadcast. The input is as follows. There is a set of 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 and let D_t^p be the deadline for request (p, t). Let r_t^p denote the number of requests (p, t). For clarity of exposition, we assume that all requests (p, t) have the same deadline D_t^p; our algorithm works for the case when different requests for page p at time t have different deadlines. A time slot t is the window of time between time $t - 1$ and time t. Let T be the last time of request for a page. Without loss of generality, we can assume that T is polynomially bounded. The server can broadcast one page at any time; an α-speed, β-approximation (or (α, β)-approximation) for a problem is one where α pages can be broadcast at any step, and where a β-approximation is achieved w.r.t. a 1-speed optimal solution. 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. We work in the off-line setting in which the server is aware of all the future requests.

3.1. Throughput Maximization

In this problem, our goal is to schedule the broadcast of pages in a way so as to maximize the total throughput, i.e., the total number of requests satisfied before their deadlines. (A simple extension also gives this result for the weighted case, where each request has a weight.) Bar-Noy *et al.* [3] present a 1–speed, $1/2$–approximation for this problem in a general setting; we now give an algorithm based on dependent rounding that gives a 1–speed, $3/4$–approximation for throughput maximization.

3.1.1 Integer Programming Formulation

We use the following integer programming (IP) formulation. The binary variable $y_{t'}^p = 1$ iff page p is broadcast at time

t'. The binary variable $x_t^p = 1$ iff request (p,t) is satisfied at some time $t', t < t' \le D_t^p$. The first set of constraints ensure that whenever a request (p,t) is satisfied, page p is broadcast at $t', t < t' \le D_t^p$. The second set of constraints ensure that at most one page is broadcast at any given time. The last two constraints ensure that the variables assume integral values. By letting the domain of x_t^p and $y_{t'}^p$ be $0 \le x_t^p, y_{t'}^p \le 1$, we obtain the LP relaxation for the problem.

$$
\begin{array}{ll}
\text{Maximize} \sum_{(p,t)} r_t^p \cdot x_t^p & \\
\sum_{t'=t+1}^{D_t^p} y_{t'}^p - x_t^p \ge 0 & \forall p, t, t' > t \\
\sum_p y_{t'}^p \le 1 & \forall t' \\
x_t^p \in \{0,1\} & \forall p, t, t' \\
y_{t'}^p \in \{0,1\} & \forall p, t'
\end{array}
\qquad (3)
$$

3.1.2 A $3/4$-approximation Algorithm

1. Solve optimally the LP relaxation.

2. Construct a bipartite graph $G = (U, V, E)$ as follows. U consists of vertices that represent time slots. Let u_t be a vertex in U corresponding to time t. Consider a page p and the time instances during which page p is broadcast fractionally in the LP solution. Let these time slots be $\{t_1, t_2, \ldots, t_k\}$ such that $t_i < t_{i+1}$. We will group these time slots into m windows, $W_j^p, 1 \le j \le m$, such that in each window except the first and the last, *exactly* one page p is broadcast fractionally. The first and last windows may broadcast a full page or less. The grouping of time slots into windows is done as follows. Choose $z \in (0,1]$ uniformly at random. z represents the amount of service provided by the first window. For each time instance t_h, we will associate a fraction $b_{t_h,j}^p$ that represents the amount of contribution made by time slot t_h towards the fractional broadcast of page p in W_j^p. For all h and for $j \ge 2$, define $b_{t_h,1}^p$ and $b_{t_h,j}^p$ as follows. If $\sum_{i=1}^{h-1} y_{t_i}^p < z$ then $b_{t_h,1}^p = \min\{y_{t_h}^p, z - \sum_{t'<t_h, t' \in W_1^p} b_{t',1}^p\}$, and 0 otherwise. For all $j \ge 2$, if $\sum_{i=1}^{h-1} y_{t_i}^p < j - 1 + z$ then $b_{t_h,j}^p = \min\{y_{t_h}^p - b_{t_h,j-1}^p, 1 - \sum_{t'<t_h, t' \in W_j^p} b_{t',j}^p\}$ and 0 otherwise.

A time slot t_h belongs to W_j^p iff $b_{t_h,j}^p > 0$. This implies that a window W_j^p consists of consecutive time slots and that a time slot can belong to at most two windows. Also, the total number of windows, $m \in \{\lceil \sum_{t'} y_{t'}^p \rceil, \lceil \sum_{t'} y_{t'}^p \rceil + 1\}$. The vertex set V consists of vertices that represent pages. For each page p, we have vertices $v_1^p, v_2^p, \ldots, v_m^p$ in V. For all p and j, v_j^p is connected to vertices corresponding to time-slots in W_j^p. The value of an edge (v_j^p, u_{t_h}) is equal to $b_{t_h,j}^p$. This construction is illustrated in Figure 1, in which a subgraph of G that is induced on the vertices and edges relevant to a particular page p are shown. $z = 1$ and $y_j^p, t_1 \le j \le t_7$, values are $0.3, 0.3, 0.5, 0.5, 0.2, 0.9, 0.8$.

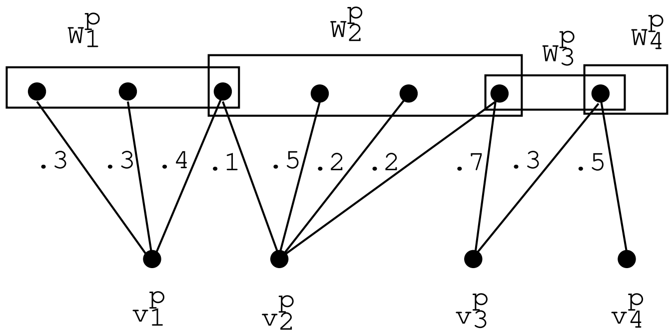

Figure 1. Subgraph of G relevant to page p whose $y_j^p, t_1 \le j \le t_7$, values are $0.3, 0.3, 0.5, 0.5, 0.2, 0.9, 0.8$. $z = 1$

3. Perform dependent rounding on G. In the rounded solution, if an edge (v_k^p, u_{t_h}) gets chosen then we broadcast page p at time t_h.

Analysis. Consider a request $r = (p,t)$. Let $t \in W_j^p$. Let l_r be the fraction of request r satisfied by the optimal LP solution before its deadline. Note that the fraction l_r can span at most two adjacent windows. Let $w = \sum_{t'>t, t' \in W_j^p} b_{t',j}^p$. Picking $z \in [0,1)$ uniformly at random is equivalent to picking $w \in [0,1]$ uniformly at random. Fix w. Let A_w and B_w be the set of time slots serving r in W_j^p and W_{j+1}^p respectively. Let R_i^p be the random variable that is 1 iff page p is broadcast during slot i in the rounded solution and is 0 otherwise. Note that $\sum_{i \in A_w} R_i^p$ and $\sum_{i \in B_w} R_i^p$ are at most 1, because of property $(P2)$ of dependent rounding.

Lemma 3.1 *For a fixed w, let $S_r(w)$ be the random variable that is 1 if r is satisfied in the rounded solution and is 0 otherwise. Then, $\mathbf{E}[S_r(w)] \ge \max\{w, l_r - w\}$ if $w \le l_r$, and $\mathbf{E}[S_r(w)] \ge l_r$ if $w > l_r$.*

Proof: $S_r(w) = \max(\sum_{i \in A_w} R_i^p, \sum_{i \in B_w} R_i^p)$ implies

$$
\begin{aligned}
\mathbf{E}[S_r(w)] &= \Pr S_r(w) = 1 \\
&= \Pr\,(\sum_{i \in A_w} R_i^p = 1) \bigvee (\sum_{i \in B_w} R_i^p = 1) \\
&\ge \max(\Pr \sum_{i \in A_w} R_i^p, \Pr \sum_{i \in B_w} R_i^p) \\
&= \left\{ \begin{array}{ll} \max\{w, l_r - w\} & \text{if } w \le l_r \\ l_r & \text{otherwise} \end{array} \right.
\end{aligned}
$$

Lemma 3.2 *Let S_r be the random variable that is 1 iff r is satisfied before D_t^p in the rounded solution and is 0 otherwise. Then, $\mathbf{E}[S_r] \ge \frac{3}{4} l_r$.*

Proof: $\mathbf{E}[S_r] \geq \int_0^1 \mathbf{E}[S_r(w)]\, dw = \int_0^{l_r} \max(w, l_r - w)\, dw + \int_{l_r}^1 l_r\, dw = \int_0^{l_r/2}(l_r - w)\, dw + \int_{l_r/2}^{l_r} w\, dw + \int_{l_r}^1 l_r\, dw = l_r - l_r^2/4 \geq \frac{3}{4} l_r$.

Lemma 3.3 *Our algorithm broadcasts at most one page at any time instance.*

Proof: The fractional values of the edges incident on any vertex in U sum up to at most 1. Hence, by property (P2) of dependent rounding, for any vertex $u_t \in U$, at most one edge incident on u_t gets chosen in the rounded solution. The claim follows.

Theorem 3.4 *Let OBJ be the random variable representing the total number of requests satisfied by our algorithm. Then,* $\mathbf{E}[OBJ] \geq \frac{3}{4}$*OPT.*

Proof: Since $OBJ = \sum_r r_t^p S_r$, $\mathbf{E}[OBJ] = \sum_r r_t^p \mathbf{E}[S_r]$. Using Lemma 3.2, we get $\mathbf{E}[OBJ] \geq \frac{3}{4} OPT$.

Our proof approach also leads to the following type of algorithm, which offers a per-user guarantee: on every instance, the algorithm will either prove that there is no schedule that satisfies all users, or present a randomized schedule where each user is satisfied with probability at least $3/4$. Also, for the case where each request has an arbitrary time-window (instead of an interval) in which it must be processed, a simpler rounding scheme can be shown to yield the analog of Theorem 3.4 as well as of this per-user guarantee, with $1 - 1/e \sim 0.632$ replacing the value $3/4$.

3.2. Minimizing Response Time

In this problem, for each request (p, t), its deadline $D_t^p = \infty$. 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 $\sum_{(p,t)} r_t^p (S_t^p - t)$, where for a request (p, t), S_t^p is the first time instance *after* t when page p is broadcast. Here, we give a $(3, 1)$-approximation algorithm.

3.2.1 IP Formulation

The problem of minimizing response time can be formulated as an IP as follows as in [11]. 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 first set of constraints ensure that whenever a request (p, t) is satisfied at time t', page p is broadcast at t'. The second set of constraints ensure that every request (p, t) is satisfied at some time $t' > t$. The third set of constraints ensure that at most one page is broadcast at any given time. The last two sets of constraints ensure that the variables assume integral values. By letting the domain of $x_{tt'}^p$ and $y_{t'}^p$ be $0 \leq x_{tt'}^p, y_{t'}^p \leq 1$, we obtain the LP relaxation for this problem.

$$
\begin{array}{ll}
\text{Minimize} \displaystyle\sum_p \sum_t \sum_{t'=t+1}^{T+n} (t' - t) \cdot r_t^p \cdot x_{tt'}^p & \\
y_{t'}^p - x_{tt'}^p \geq 0 & \forall p, t, t' > t \\
\displaystyle\sum_{t'=t+1}^{T+n} x_{tt'}^p \geq 1 & \forall p, t \\
\displaystyle\sum_p y_{t'}^p \leq 1 & \forall t' \\
x_{tt'}^p \in \{0, 1\} & \forall p, t, t' \\
y_{t'}^p \in \{0, 1\} & \forall p, t'
\end{array}
\tag{4}
$$

3.2.2 Algorithm

Let an (α, β)-algorithm stand for an algorithm that outputs a solution in which α pages are broadcast at any time instance and the cost of the solution is β-factor away from the optimal. The current-best algorithm for this problem achieves bounds of $(2, 2)$, $(3, 1.5)$, and $(4, 1)$ [11]. In this Section, we give an algorithm that improves last two of these bounds. Our algorithm uses dependent rounding and gives bounds of $(2, 2)$ and $(3, 1)$. Our algorithm guarantees that the expected response time of *each* request is at most twice its response time in the LP solution.

1. Solve the LP relaxation optimally.

2. Let I be the given instance of the problem. For any page p, let $N_p = \{t_1, t_2, \ldots, t_{f_p}\}$ denote the times at which requests for page p are made in instance I. Let $ft(1/2, p, t)$ be the first time instance when $1/2$-fraction of request (p, t) gets satisfied in the LP solution, i.e., $ft(1/2, p, t) = \min\{t'' | \sum_{t'=t+1}^{t''} x_{tt'}^p \geq 1/2\}$. We transform instance I into a simplified instance I' which has the following property. If N_p' represents the times of positive requests for page p in I' then for any times $\{t_u', t_v'\} \subseteq N_p'$, such that $t_u' < t_v'$, we have $ft(1/2, p, t_u') \leq t_v'$. For every $t \in N_p \setminus N_p'$, we have we have $ft(1/2, p, t) > t'$, where $t' \in N_p'$. We drop all requests for page p made at each time $t \in N_p \setminus N_p'$. This transformation is done separately for each page. The pseudo-code is given in Figure 2.

3. Construct a bipartite graph $G = (U, V, E)$ as follows. In U, we have a vertex u_t for each time instance t. For each request $r' = (p, t)$ in I', we have a vertex $v_{r'}$ in V. Let $H_{r'}$ be the set of time slots that satisfy half of request r' in the LP solution at minimum cost. For each time slot $t' \in H_{r'}$, define $b_{t'}^p = \min\{y_{t'}^p, 1/2 - \sum_{t''<t', t'' \in H_{r'}} b_{t''}^p\}$. In other words, $b_{t'}^p$ is the contribution of time slot t' towards satisfying *exactly* half fraction of request r'. The edge set E consists of edges that connect each vertex $v_{r'}$ to the vertices corresponding to time slots in $H_{r'}$. The value associated with an edge

REQUEST CONSOLIDATION(page p, α)

$N'_p \leftarrow \{t_{f_p}\}$
$l \leftarrow f_p$
$r'^p_{t_l} \leftarrow r^p_{t_l}$
for $k \leftarrow l-1$ **down to** 1 **do**
  $ft(1/2, p, t_k) \leftarrow \min_{t'}\{\sum_{t=t_k+1}^{t'} x^p_{t_k t} \geq 1/2\}$
  if $(ft(1/2, p, t_k) \leq t_l)$ **then**
    $N'_p \leftarrow N'_p \cup \{t_k\}$
    $l \leftarrow k$
    $r'^p_{t_l} \leftarrow r^p_{t_l}$
  else
    $r'^p_{t_k} \leftarrow 0$
return N'_p

Figure 2. Algorithm to obtain I'

$(v_{r'}, u_{t'})$, where $t' \in H_{r'}$ is $2b^p_{t'}$. Note that for any two edges $(v_{r'}, u_{t'})$ and $(v_{r''}, u_{t'})$, r' and r'' correspond to request for two *different* pages.

4. Perform dependent rounding on G. For any request $r' = (p, t)$, if an edge $(v_{r'}, u_{t'})$ is selected then broadcast page p at t'.

3.2.3 Analysis

For any request $r = (p, t) \in I$, let f_r be the fraction of request satisfied in the LP solution at or before the first time instance t_k such that $t_k \in N'_p$ and $t_k \geq t$. Let C^f_r be the cost of satisfying this f_r fraction of request. Let C^h_r be the cost of satisfying the next half of request r, i.e, the cost of satisfying exactly half fraction of r after t_k. Let C^ϵ_r be the cost of satisfying the last $\epsilon = 1/2 - f_r$ fraction of request r. Let $C_r = C^f_r + C^h_r + C^\epsilon_r$ be the cost of satisfying r in the LP solution. Note that for a request $r' \in I'$, since $f = 0$ we have $C^f_{r'} = 0$ and $C^\epsilon_{r'} \geq C^h_{r'}$. For a request $r \in I \setminus I'$, $f < 1/2$. Let R^p_i be the random variable that is 1 iff page p is broadcast during slot i in the rounded solution and is 0 otherwise.

Lemma 3.5 *For any request $r' = (p, t) \in I'$, if $S_{r'}$ is the random variable denoting the cost of satisfying r' in the rounded solution then we have $\mathbf{E}[S_{r'}] \leq C_{r'}$.*

Proof: Since $S_{r'} = \min\{(t'-t)R^p_{t'} | t' \in H_{r'}\}$, we get $\mathbf{E}[S_{r'}] = \sum_{t' \in H_{r'}} (t'-t) \Pr R^p_{t'} = 1 = \sum_{t' \in H_{r'}} (t'-t) 2b^p_{t'} = 2C^h_{r'} \leq C^h_{r'} + C^\epsilon_{r'} = C_{r'}$

Lemma 3.6 *Consider any requests $r = (p, t) \in I \setminus I'$ and $r' = (p, t') \in I'$, where $t' \in N'_p$ is first such time instance after t. If S_r is the random variable denoting the cost of satisfying r in the rounded solution then we have $\mathbf{E}[S_r] \leq 2C_r$.*

Proof: Since $S_r = \min\{(t''-t)R^p_{t''} | t'' \in H_{r'}\}$, we have $\mathbf{E}[S_r] = \sum_{t'' \in H_{r'}} (t''-t) \Pr R^p_{t''} = 1 = \sum_{t'' \in H_{r'}} (t''-t) 2b^p_{t''} = 2C^h_r \leq 2C_r$.

Theorem 3.7 *The above rounding scheme yields a 2-speed, 2-approximate solution.*

3.2.4 A 3-speed, 1-factor Algorithm

In this section we give a $(3, 1)$-algorithm that builds on the $(2, 2)$-algorithm developed in Section 3.2.2. Note that in the 2-speed solution the requests in I' get satisfied optimally using the two channels as shown in Lemma 3.5. By using two channels we can only guarantee a factor of 2 for the requests in $I \setminus I'$. We now show that by using an extra channel we can satisfy the requests in $I \setminus I'$ optimally thus obtaining a 3-speed, 1-approximate solution. In addition to the four steps in Section 3.2.2, we perform steps 5 and 6 as described below.

5. Construct a bipartite graph $G' = (U', V', E')$ as follows. For each time instance t, we have a vertex u_t in U'. In V', for each time $t' \in N'_p$, we have one vertex $v^{t'}_p$ representing all requests $r = (p, t)$ such that $t < t'$ and $ft(1/2, p, t) > t'$. Each vertex $v^{t'}_p$ is connected to $u_{t''}$ where $t < t'' \leq t'$ and $y^p_{t''} > 0$. The value associated with each edge $(v^{t'}_p, u_{t''})$ is $y^p_{t''}$.

6. Perform dependent rounding on G'. For an edge $(v^{t'}_p, u_t)$ that gets chosen in the rounded solution, broadcast page p at time t. We will refer to the broadcast on this channel as the *third channel*.

Lemma 3.8 *For any request $r = (p, t) \in I \setminus I'$, if S_r is the random variable denoting the cost of satisfying r in the rounded solution then we have $\mathbf{E}[S_r] \leq C_r$.*

Proof: Recall the definition of f_r from Section 3.2.3. Let $f_r = 1/2 - \epsilon, \epsilon > 0$. By property (P2) of dependent rounding at most one edge incident on the vertex representing r gets chosen. By property (P1) of dependent rounding, request r gets satisfied on the third channel with a probability f_r with an expected cost of C^f_r. It gets satisfied on the first two channels (pages broadcast in step 4) with a probability $1 - f_r = 1/2 + \epsilon$.

$$\begin{aligned} \mathbf{E}[S_r] &= C^f_r + (1/2 + \epsilon)(2C^h_r) \\ &= C^f_r + C^h_r + 2\epsilon C^h_r \end{aligned} \tag{5}$$

Let t_h be the last time instance that contributes towards the cost C^h_r. $C^h_r \leq 1/2(t_h - t)$. Also, $C^\epsilon_r \geq \epsilon(t_h - t)$. Combining these facts with Equation (5), we get $\mathbf{E}[S_r] \leq C^f_r + C^h_r + \epsilon(t_h - t) \leq C^f_r + C^h_r + C^\epsilon_r = C_r$

Theorem 3.9 *The above rounding scheme yields a 3-speed, 1-approximate solution.*

Proof In steps 1-4, we broadcast at most two pages per time instance as argued in Theorem 3.7. In steps 5-6, we broadcast at most one page at any time instance since the fractional value of edges incident on any vertex in U' is at most one. Thus steps 1-6 ensure a 3-speed solution. The proof of optimality follows from Lemmas 3.5 and 3.8.

4. Capacitated Vertex Cover

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, ..., n\}$ and edge set E. Suppose that w_v denotes the weight of vertex v and k_v denotes the capacity of vertex v (we assume that k_v is an integer). A *capacitated vertex cover* is a function $x : V \rightarrow I\!N_0$ such that there exists an orientation of the edges of G in which the number of edges directed into vertex $v \in V$ is at most $k_v x_v$. (These edges are said to be *covered* by or *assigned* to v.) The *weight* of the cover is $\sum_{v \in V} x_v w_v$, where $x_v \equiv x(v)$. The MINIMUM CAPACITATED VERTEX COVER problem is that of computing a minimum weight capacitated cover. The problem generalizes the MINIMUM WEIGHT VERTEX COVER problem which can be obtained by setting $k_v = |V| - 1$ for every $v \in V$. The main difference is that in vertex cover, by picking a node v in the cover we can cover all edges incident to v, in this problem we can only cover a subset of at most k_v edges incident to node v. Clearly, the problem is NP-hard since it generalizes a well known NP-hard problem.

We denote by $\delta(v)$ the edges in E which are incident to v. We also denote by $d(v) = |\delta(v)|$ the degree of $v \in V$. For $S \subseteq V$ we denote by $G(S) = (V, E(S))$ the subgraph induced by S. We denote an edge with end-vertices i, j as a set $\{i, j\}$. We deal with orientation (direction) of edges. Orienting an edge $\{i, j\}$ from i to j is equivalent to replacing it by a directed edge (an arc) (i, j). A primal-dual factor 2 approximation was given by Guha, Hassin, Khuller and Or [12] and a factor 4 approximation was given using LP rounding. In this paper, we derive a 2 approximation by a simple LP rounding method, using dependent rounding. One advantage of this approach is the ease of implementation compared to a primal-dual algorithm (since very fast and easy to use LP solvers are available). In addition, more general objective functions can be easily captured in this framework, such as assignment costs (see Section 4.2). Recently, Chuzhoy and Naor [5] addressed a variant of capacitated vertex cover problem in which there is a bound b_v on x_v. In other words, a vertex v can be used at most b_v times to cover edges. They gave a 3-approximation for the unweighted case and showed that the weighted case is hard to approximate within a factor of $O(\log n)$. As a corollary to our rounding procedure, for the weighted version, we obtain a 2-approximate solution in which a vertex v is used at most $2b_v$ times.

4.1. IP formulation and rounding scheme

An IP formulation is as follows ([12]).

$$
\begin{array}{ll}
\text{Minimize } \sum_v w_v x_v & \\
y_{eu} + y_{ev} \geq 1 & e = \{u, v\} \in E, \\
k_v x_v - \sum_{e \in \delta(v)} y_{ev} \geq 0 & v \in V, \\
x_v \geq y_{ev} & v \in e \in E, \\
y_{ev} \in \{0, 1\} & v \in e \in E, \\
x_v \in I\!N_0 & v \in V.
\end{array}
\tag{6}
$$

In this formulation, $y_{ev} = 1$ denotes that the edge $e \in E$ is covered by vertex v. Clearly, the values of x in a feasible solution correspond to a capacitated cover. While we do not need the constraints $x_v \geq y_{ev}$ for the IP formulation, they will play an important role in the relaxation.

Algorithm Threshold and Round:

1. Solve the above LP to obtain an optimal fractional solution.

2. Pick a value α uniformly at random in the interval $[\frac{1}{2}, 1]$.

3. For each edge $e = (u, v)$ if $y_{eu} \geq \alpha$ then set $y^*_{eu} = 1$, else if $y_{ev} \geq \alpha$ then set $y^*_{ev} = 1$.

4. For the remaining edges (where $y^*_{eu} < \alpha$ and $y^*_{ev} < \alpha$) we will use dependent rounding. Let the set of remaining edges be E'.

5. Create a bipartite graph as follows. Let one side contain a vertex corresponding to each edge in E'. The other side contains a vertex corresponding to each vertex in V. There is an edge from $e \in E'$ to $u \in V$ if e is incident on u. The weight of this edge is y_{eu}. W.l.o.g, $y_{eu} + y_{ev} = 1$. We now use dependent rounding to round the y_{eu} values to integers y^*_{eu}. We define $x^*_u = \lceil \frac{\sum_{e \in E(u)} y^*_{eu}}{k_u} \rceil$. In other words, after rounding the y_{eu} values to $\{0, 1\}$ we simply define the x^*_u value to be the number of copies of u that are required to cover all the edges assigned to it.

Theorem 4.1 *The expected cost of the integral solution produced by Algorithm Threshold and Round, is at most $2OPT_{LP}$, where OPT_{LP} is the cost of the minimum fractional solution to the relaxation. Moreover,* $\mathbf{E}[x^*_v] \leq 2x_v$.

Corollary 4.2 *If b_v is an upper bound on x_v, then Algorithm Threshold and Round produces a 2-approximate solution in which $x^*_v \leq 2x_v$.*

The proof of Theorem 4.1 is quite complicated and omitted. We will show that if we pick α as $\frac{2}{3}$ then we can obtain a 3-approximation.

Theorem 4.3 *If $\alpha = \frac{2}{3}$ then the expected cost of the algorithm is at most $3OPT_{LP}$.*

Proof: We will show that for any vertex v, the expected cost of x_v^* is at most $3x_v$. Since the cost of the solution produced is $\sum_v w_v x_v^*$, the bound follows.

Let d_v be the number of edges assigned to v by edges that have their y_{ev} value at least $\frac{2}{3}$. Some edges (v, u) are assigned to the other end u since $y_{eu} \geq \frac{2}{3}$. Let $a_v + f_v = \sum_{e \in \delta(v) \cap E'} y_{ev}$, where $0 \leq f_v < 1$ and a_v is an integer.

Notice that

$$\mathbf{E}(x_v^*) = f_v \cdot (\lceil \frac{d_v + a_v + 1}{k_v} \rceil) + (1 - f_v) \cdot (\lceil \frac{d_v + a_v}{k_v} \rceil).$$

We would like to prove that this is at most $3x_v$.

We consider two cases: If $d_v + a_v < k_v$ then the LHS is simply 1. If $x_v \geq \frac{1}{3}$, we are done. We now show that this is the case. If $d_v \geq 1$, since $x_v \geq y_{ev} \geq \frac{2}{3}$, the claim follows. If $d_v = 0$ and $E' \cap \delta(v) \neq \emptyset$ then for the unassigned edges we have $y_{ev} \geq \frac{1}{3}$ (otherwise they would have been assigned to the other end point). Since $x_v \geq y_{ev}$, the claim follows.

We now consider the other case where $d_v + a_v \geq k_v$. We can upper bound $\lceil x \rceil$ by $x + 1$. Using this bound, we get $\mathbf{E}[x_v^*] = f_v(\frac{d_v + a_v + 1 + k_v}{k_v}) + (1 - f_v)\frac{d_v + a_v + k_v}{k_v} = \frac{d_v + a_v + k_v}{k_v} + \frac{f_v}{k_v} \leq \frac{2(d_v + a_v)}{k_v} + \frac{f_v}{k_v} \leq \frac{2(d_v + a_v + f_v)}{k_v} \leq \frac{3(\frac{2}{3}(d_v + a_v + f_v))}{k_v}$, i.e., at most three times the cost of the optimal LP solution.

4.2. Assignment Costs

We can generalize this problem in the following way: in addition to having weights on the vertices, we have an assignment cost c_{eu} for assigning an edge e to vertex u. In the IP, the only change that we make is to add $\sum_{e \in E} \sum_{u \in e} c_{eu} y_{eu}$ to the objective function. We now solve the resulting relaxation of the IP to the natural LP and run Algorithm Threshold and Round by choosing α in the same way.

Suppose that in the optimal solution of the linear program, the optimum cost is $OPT_{LP} = OPT_{LP}^v + OPT_{LP}^e$ which represents the optimum fractional cost due to the weighted sum of chosen vertices and the assignment cost of edges.

Theorem 4.4 *Algorithm Threshold and Round finds a solution y^* such that the expected assignment cost is at most $(4 - 2\sqrt{2})OPT_{LP}^e$.*

Combining this with Theorem 4.1 where we bound the expected cost of the weighted sum of the chosen vertices by $2OPT_v$, we clearly obtain an integral solution that is a 2 approximation for the problem where our objective is the sum of the weighted cost of vertices and the assignment cost of edges.

Theorem 4.5 *Algorithm Threshold and Round finds a solution x^*, y^* such that the expected weight of vertices is at most $2OPT_{LP}^v$ and the expected assignment cost is at most $(4 - 2\sqrt{2})OPT_{LP}^e$. Thus this gives a 2 approximation for the problem with vertex weights and assignment costs (since the total cost is at most $2OPT_{LP}$).*

5. Scheduling on Unrelated Parallel Machines

Given a set J of jobs, a set M of machines, and (for each $j \in J$ and $i \in M$) the time p_{ij} required to process job j on machine i, the problem is to schedule the jobs on the machines so as to minimize the makespan. Lenstra, Shmoys and Tardos [15] give a 2-approximate solution for this problem; their result is as follows. Define a linear program LP(t) with a variable $x_{i,j}$ for each machine i and job j; x_{ij} is 1 iff job j is scheduled on machine i. Let $S_t = \{(i,j) | \, p_{ij} \leq t\}$. LP($t$) asks for a feasible solution for the following system of linear constraints: (i) $\sum_i x_{ij} = 1$ for all j; (ii) $\sum_j x_{ij} p_{ij} \leq t$ for all i; (iii) $x_{ij} \geq 0$ for all $(i,j) \in S_t$; and (iv) $x_{ij} = 0$ for all $(i,j) \notin S_t$. It is easy to see that the given instance has makespan at most t only if LP(t) has a feasible solution; thus, the infimum of all such "feasible" t is a lower bound on the optimal makespan. We will throughout let T be a lower bound on this infimum which is close to the infimum with high precision; such a T can be computed efficiently via a bisection search. Then, the work of [15] presents a rounding method to produce a schedule of makespan at most $2T$, which is a 2-approximation. Since then, it has been an open question if we can do better in polynomial time.

In the spirit of offering per-user guarantees, we can also ask the following question. Are there constants $p > 0$ and $q < 2$ for which we can construct a randomized schedule that has makespan at most qT with probability 1, and where each given job gets scheduled with probability at least p? (The algorithm of [15] provides $p = 1$ and $q = 2$.) As pointed out by Chandra Chekuri to us, this is indeed possible for $p = 1/2$ and $q = 1$ [4]. It is not clear how to extend this approach to achieve $p > 1/2$ and $q < 2$; we show such a result here.

Let T be as above, and let the vector x denote an optimal solution to LP(T). We need the following notion of (ϕ, ψ)-*weighted* dependent rounding. Briefly, we will work with a star graph G_i, whose center is some machine i and whose leaves are the jobs j. We have a dependent rounding instance on this graph given by the vector x; we modify our dependent rounding scheme as follows. Note that there is no cycle in G_i. In dependent rounding, suppose we pick a path $P = \langle j_1, i, j_2 \rangle$; let the values $y_{i,\cdot}$ be the probabilistically changing values as in Section 2. We now define α to be the smallest positive γ for which either $y_{i,j_1} + \gamma = 1$ or $y_{i,j_2} - (p_{ij_1}/p_{ij_2}) \cdot \gamma = 0$; define β to be the smallest positive γ for which either $y_{i,j_1} - \gamma = 0$ or $y_{i,j_2} + (p_{ij_1}/p_{ij_2}) \cdot \gamma = 1$. With probability $\beta/(\alpha + \beta)$, increment y_{i,j_1} by α and decrement y_{i,j_2} by $(p_{ij_1}/p_{ij_2}) \cdot \alpha$; with probability $\alpha/(\alpha + \beta)$, decrement y_{i,j_1} by β and increment y_{i,j_2} by $(p_{ij_1}/p_{ij_2}) \cdot \beta$. Finally, suppose the maximal path P is just an edge (i, j). If $p_{i,j} > (1 - \phi)T$ and $y_{i,j} \leq \psi$, set $y_{i,j}$ to be zero (this is crucial); else (i.e., if $p_{i,j} \leq (1 - \phi)T$ or if $y_{i,j} > \psi$), set $y_{i,j} := 1$ with probability $y_{i,j}$ and set $y_{i,j} := 0$ with probability $1 - y_{i,j}$. This scheme has some useful properties which we exploit below.

Algorithm. We are given a constant $\epsilon > 0$, and proceed as follows.

1. Find the lower bound T by bisection search, and solve LP(T) optimally to get a solution vector x.

2. Consider a bipartite graph $G = (J, M, E)$, where J is the set of jobs, M is the set of machines and an edge $(i, j) \in E$ iff $x_{ij} > 0$. Repeatedly applying a "scaling-rounding-rescaling" technique of [14] a certain number of times, we obtain the following modified instance. The load on each machine is at most T. Moreover, every job is assigned at least $1/(1 + \delta)$ fractionally, where $\delta = \Theta(\epsilon)$. Crucially, there are constants $d = d(\epsilon)$ and $u = u(\epsilon)$ (with $\lim_{\epsilon \to 0} u(\epsilon) = 0$) such that for all j, one of the following two conditions holds: (a) $\sum_{i:\ p_{i,j} > T/2} x_{ij} \leq u(\epsilon)$, or (b) for all i such that $x_{ij} > 0$, we have $x_{ij} \geq 1/(d(1 + \delta))$.

3. For each $i \in M$, let G_i be the subgraph of G induced on vertex i and its neighbors. Perform $(1/2, \psi)$-weighted dependent rounding on all the G_i independently, for a certain $\psi = \Theta(1/d)$. Schedule job j on an arbitrary machine i for which x_{ij} gets rounded to 1, if such an i exists.

Theorem 5.1 *The above algorithm yields a $(2 - \Omega(1))$-approximate solution in which each job is scheduled with a probability of at least $(1 - \frac{1}{e} - \epsilon)$.*

Acknowledgments. We thank Eran Halperin and Maxim Sviridenko for pointing out the work of [1] to us. We thank Chandra Chekuri and David Shmoys for helpful discussions on [15], and Alan Frieze for his encouragement. We thank Julia Chuzhoy and Seffi Naor for sending us a copy of [5].

References

[1] A. Ageev and M. Sviridenko. *Pipage rounding: a new method of constructing algorithms with proven performance guarantee*. Submitted for publication; available at *www.research.ibm.com/people/s/sviri/sv_publ.html*

[2] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber. *A Unified Approach to Approximating Resource Allocation and Scheduling.* Proc. ACM Symposium on Theory of Computing, 735-744, 2000.

[3] A. Bar-Noy, S. Guha, Y. Katz, J. Naor, B. Schieber and H. Shachnai. *Throughput Maximization of Real-Time Scheduling with Batching.* Proc. ACM-SIAM Symposium on Discrete Algorithms, 742-751, 2002.

[4] C. Chekuri and S. Khanna. *A PTAS for the Multiple Knapsack Problem.* Proc. ACM-SIAM Symposium on Discrete Algorithms, 213–222, 2000.

[5] J. Chuzhoy and J. Naor. *Covering Problems with Hard Capacities.* Proc. IEEE Symposium on Foundations of Computer Science, 2002 (to appear).

[6] C. Cooper and A. Frieze. *On a general model of web graphs*. Proc. European Symposium on Algorithms, 500–511, 2001.

[7] C. Cooper and A. Frieze. *Crawling on web graphs*. Proc. ACM Symposium on Theory of Computing, 419–427, 2002.

[8] DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling, 2002. See *http://dimacs.rutgers.edu/Workshops/Internet/program.html.*

[9] T. Erlebach, A. Hall. *NP-Hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow.* Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms, 194-202, 2002.

[10] F. Chung Graham and L. Lu. *Connected components in random graphs with given degree sequences.* Manuscript.

[11] R. Gandhi, S. Khuller, Y. Kim and Y. C. Wan. *Algorithms for Minimizing Response Time in Broadcast Scheduling*. Proc. Ninth Conference on Integer Programming and Combinatorial Optimization, 425–438, 2002.

[12] S. Guha, R. Hassin, S. Khuller and E. Or. *Capacitated Vertex Covering with Applications*. Proc. ACM-SIAM Symposium on Discrete Algorithms, 858-865, 2002.

[13] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. *Scheduling broadcasts in wireless networks*. In Proc. European Symposium of Algorithms, LNCS 1879, Springer-Verlag, 290-301, 2000.

[14] F. T. Leighton, C. J. Liu, S. B. Rao and A. Srinivasan. *New Algorithmic Aspects of the Local Lemma with Applications to Routing and Partitioning*. SIAM Journal on Computing, Vol. 31, 626-641, 2001.

[15] J. K. Lenstra, D. B. Shmoys and É. Tardos. *Approximation algorithms for scheduling unrelated parallel machines*. Mathematical Programming, 46:259–271, 1990.

[16] D. B. Shmoys, É. Tardos and K. Aardal. *Approximation Algorithms for Facility Location Problems*. Proc. ACM Symposium on Theory of Computing, 265–274, 1997.

[17] A. Srinivasan. *Distributions on Level-Sets with Applications to Approximation Algorithms*. Proc. IEEE Symposium on Foundations of Computer Science, 588–597, 2001.