Dependent Rounding and its Applications to Approximation Algorithms*

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

Abstract

We 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 improved (approximation) algorithms for various problems. These include:

- low congestion multi-path routing;
- 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, as well as (iii) capacitated vertex cover; and
- fair scheduling of jobs on unrelated parallel machines.

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. One approach to accommodate such constraints is the elegant deterministic method of Ageev & Sviridenko [1]. In this work, we develop a dependent randomized rounding scheme to handle such constraints; the term "dependent" underscores the fact that various random choices we make are highly dependent. We then show applications to several problems by combining the scheme with new ideas; the rounding approach is also of independent interest. 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 provide 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 distribution. 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 $D_i \in$

*Earlier versions of this work were presented at the *IEEE Symp. on Foundations of Computer Science*, 2001 [35], and at the *IEEE Symp. on Foundations of Computer Science*, 2002 [22].

†Department of Computer Science, Rutgers University, Camden, NJ 08102. E-mail: rajivg@camden.rutgers.edu. Part of this work was done when the author was a student at the University of Maryland and was supported by NSF Award CCR-9820965.

‡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. Supported in part by NSF Award CCR-0208005 and NSF ITR Award CNS-0426683. 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 and NSF ITR Award CNS-0426683. E-mail: srin@cs.umd.edu.

$\{\lfloor d_i \rfloor, \lceil d_i \rceil\}$. 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 $X_{i,j}$. For any vertex i and any subset S of the edges incident on i:

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

An interesting and useful special case of our scheme occurs when the bipartite graph is a star (i.e., $|A| = 1$) and when the sum of the edge weights is an integer (i.e., $\sum_{i \in B} x_{1,i}$ is an integer). Simplifying the notation in this case and letting $[s]$ denote the set $\{1, 2, \ldots, s\}$, note that the basic problem here is the following. We are given a sequence $P = (x_1, x_2, \ldots, x_t)$ of t reals such that each x_i lies in $[0, 1]$, and such that $\sum_i x_i$ is an *integer*, say ℓ. We want to construct a distribution $D(t; P)$ on $\{0, 1\}^t$ such that if $(X_1, X_2, \ldots, X_t)$ denotes a vector sampled from $D(t; P)$, then properties (P1), (P2) and (P3) are satisfied. That is, $\Pr[X_i = 1] = x_i$ for each i, the X_i add up to ℓ with probability one, and

$$\forall b \in \{0,1\} \ \forall S \subseteq [t], \ \Pr[\bigwedge_{i \in S} (X_i = b)] \leq \prod_{i \in S} \Pr[X_i = b]. \tag{2}$$

By "constructing $D(t; P)$", we mean a randomized algorithm that takes P as input, and generates $(X_1, X_2, \ldots, X_t)$ in time polynomial in t. We show that for this special star-graph case, our dependent rounding scheme runs in time linear in t. Although the X_i's are not independent, the analog (2) of (P3) helps show that any non-negative linear combination of $X_1, X_2, \ldots, X_t$ is sharply concentrated around its mean: see Theorem 3.1, which in turn leads to our first application, Theorem 3.2. One may expect that (P2), which fixes the number of X_i that can equal b, will imply a negative correlation property such as (2). However, there exists a large class of distributions on $\{0, 1\}^t$ that satisfy (P1) and (P2), but not (P3) (i.e., they do not satisfy not (2)). Thus, extra care is needed in guaranteeing (P3). We illustrate this by an example: let $x_i = \frac{\ell}{t}$ for $i = 1 \ldots t$. Suppose $t = \ell \cdot c$ for some integer c. We group the variables into c groups, with ℓ variables in each group. We randomly pick one group and set all its variables to 1. We now have exactly ℓ variables that are set to 1, with each variable having probability $\frac{1}{c}$. However the variables in each group are positively corelated. We are not aware of any earlier proof of existence of distributions such as $D(t; P)$. (Several natural approaches do not work. For instance, consider generating the X_i independently with $\Pr[X_i = 1] = x_i$, and taking the distribution of $(X_1, X_2, \ldots, X_t)$ conditional on the event "$|\{i : X_i = 1\}| = \ell$". This can be seen to violate (P1) for $P = (0.75, 0.75, 0.5)$. See [37] for the relationship of such conditioning to some negative correlation–type results.) If $x_{1,i} = \ell/t$ for all i, then one choice for $D(t; P)$ is the hypergeometric distribution (sampling ℓ elements from $[t]$ *without* replacement). This, as well as the case where $\ell = 1$, are two instances of our problem – i.e., of showing that $D(t; P)$ exists, and sampling efficiently from it – with known solutions.

Our dependent rounding scheme draws on the ideas of [1, 35]. When combined with several other new ideas, it leads to various applications, especially in the context of routing and scheduling. All of these applications share the common feature that we solve some natural linear-programming (LP) relaxation of the problem at hand, and then use dependent rounding in conjunction with additional ideas to round the fractional solution. In each of these applications, the fact that (P2) is a probability-one guarantee, will prove to be important. One noteworthy feature of many 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. On the other hand, individual users may only require that they receive a certain level of service with a guaranteed probability, and may be less concerned with other users and with global objective functions. As we describe in Sections 4 and 5, our approach lets us provide guarantees of the form: each given user is satisfied with a certain guaranteed probability. Furthermore, some of our analyses are fairly involved; see, e.g., the analysis in Section 4.4. Given the many applications presented in this work, we believe that the dependent rounding technique and extensions thereof, will be of use in other settings also.

Organization of the paper. The basic dependent rounding scheme, application to random-graph models, and a strengthening of (P3) for star graphs, are presented in Section 2. The following three sections then use dependent rounding in conjunction with various new ideas, to develop improved (approximation) algorithms via LP-rounding. Section 3 considers low-congestion *multi-path* routing. Section 4 discusses a collection of problems in the area of *broadcast scheduling*, where the basic feature is that when a server broadcasts a topic, *all* users waiting for that topic get satisfied. Section 5 revisits the classical problem of "scheduling on unrelated parallel machines" from the viewpoint of per-user guarantees. Section 6 then compares our work with some related approaches, presents certain recent applications, and concludes with some open questions. Finally, we present an additional application, the capacitated vertex cover problem, in Appendix A.

A note to the reader. Our various applications involve several additional ideas. For the reader desiring a quick understanding of the basic method and of the ways of using it, we recommend Sections 2.1, 3, and the initial part of Section 4 up to the end of Section 4.2.

2 Dependent Rounding and Some Variants

We start by presenting our basic algorithm in Section 2.1. Next, we consider the special case of star-graphs in Section 2.2, and show in Theorem 2.4 that our algorithm guarantees a stronger version of (P3) in this case. We then present an extension of the basic scheme to the non-bipartite case, motivated for instance by models for massive graphs, in Section 2.3.

2.1 The Dependent Rounding Scheme

We now present the basic dependent rounding scheme, and summarize its main properties in Theorem 2.3. Suppose we are given a bipartite graph (A, B, E) with bipartition (A, B) and a value $x_{i,j} \in [0, 1]$ for each edge $(i, j) \in E$. 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 $O(|A|+|B|)$ steps via a depth-first-search (DFS), 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. The cycle/maximal path can actually be found in $O(|A| + |B|)$ time since the first back edge we encounter yields a cycle in the DFS.

Define

$$\alpha = \min\{\gamma > 0 : ((\exists(i, j) \in M_1 : y_{i,j} + \gamma = 1) \bigvee (\exists(i, j) \in M_2 : y_{i,j} - \gamma = 0))\};$$

$$\beta = \min\{\gamma > 0 : ((\exists(i, j) \in M_1 : y_{i,j} - \gamma = 0) \bigvee (\exists(i, j) \in M_2 : y_{i,j} + \gamma = 1))\}.$$

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.

We 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. So,

$$\text{the total running time is } O((|A| + |B|) \cdot |E|). \tag{3}$$

Let us prove next that properties (P1), (P2) and (P3) hold.

Lemma 2.1 *Property (P1) holds, i.e., For every edge* (i,j), $\Pr[X_{i,j} = 1] = x_{i,j}$.

Proof: Fix an edge $(p,q) \in E$; let $Y_{p,q,k}$ be the random variable denoting the value of $y_{p,q}$ at the beginning of iteration k. We will show that

$$\forall k \geq 1, \mathbf{E}[Y_{p,q,k+1}] = \mathbf{E}[Y_{p,q,k}] \tag{4}$$

We will then have, $\Pr[X_{p,q} = 1] = \mathbf{E}[Y_{p,q,|E|+1}] = \mathbf{E}[Y_{p,q,1}] = x_{p,q}$ and (P1) will hold. We now prove equation (4) for a fixed k.

One of the following two events could occur for the edge (p,q) in iteration k.

Event A: The edge (p,q) is not part of the cycle or maximal path and its value is not modified. Hence we have $\mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v) \wedge A] = v$.

Event B: The edge (p,q) is part of the cycle or maximal path during iteration k. In this case, w.l.o.g., let the edge (p,q) be part of the matching M_1. Then, there exist (α,β) such that $\alpha+\beta > 0$ and such that the edge value gets modified probabilistically as:

$$Y_{p,q,k+1} = \begin{cases} Y_{p,q,k} + \alpha & \text{with probability } \beta/(\alpha+\beta) \\ Y_{p,q,k} - \beta & \text{with probability } \alpha/(\alpha+\beta) \end{cases}$$

Let S be the set of all values of (α,β). We say that the event $B(\alpha_1,\beta_1)$ occurred if event B occurs and $(\alpha,\beta) = (\alpha_1,\beta_1)$ for a fixed $(\alpha_1,\beta_1) \in S$. We have

$$\mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v) \wedge B(\alpha_1,\beta_1)] = (v+\alpha_1)\left(\frac{\beta_1}{\alpha_1+\beta_1}\right) + (v-\beta_1)\left(\frac{\alpha_1}{\alpha_1+\beta_1}\right) = v$$

Since the above equation holds for all values of (α,β), it also holds unconditionally. Thus, $\mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v) \wedge B] = v$. Hence,

$$\begin{aligned} \mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v)] &= \mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v) \wedge B] \cdot \Pr[B] + \\ &\quad \mathbf{E}[Y_{p,q,k+1} | (Y_{p,q,k} = v) \wedge A] \cdot \Pr[A] \\ &= v(\Pr[A] + \Pr[B]) = v \end{aligned}$$

Let V be the set of all possible values of $Y_{p,q,k}$.

$$\begin{aligned}\mathbf{E}[Y_{p,q,k+1}] &= \sum_{v\in V}\mathbf{E}[Y_{p,q,k+1}|Y_{p,q,k}=v]\cdot\Pr[Y_{p,q,k}=v] \\ &= (\sum_{v\in V} v\cdot\Pr[Y_{p,q,k}=v]) = \mathbf{E}[Y_{p,q,k}]\end{aligned}$$

This completes our proof for Property (P1). □

Next, showing that (P2) holds is quite easy. Fix any vertex i, with fractional degree d_i. If i has 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 $y_{\cdot,\cdot}$ 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 see that (P2) holds.

We next prove:

Lemma 2.2 *Property (P3) holds.*

Proof: Fix a vertex i, and a subset S of edges incident on i, as in (1). Let $b=1$ (the proof for the case where $b=0$ is identical). If $f=(i,j)$, we let $Y_{f,k}\doteq Y_{i,j,k}$, where $Y_{i,j,k}$ denotes the value of $y_{i,j}$ at the beginning of iteration k. We will show that

$$\forall k, \mathbf{E}\left[\prod_{f\in S} Y_{f,k}\right] \le \mathbf{E}\left[\prod_{f\in S} Y_{f,k-1}\right] \tag{5}$$

Thus, we will have $\Pr[\bigwedge_{f\in S}(X_f=1)] = \mathbf{E}\left[\prod_{f\in S} Y_{f,|E|+1}\right] \le \mathbf{E}[\prod_{f\in S} Y_{f,1}] = \prod_{f\in S} x_{f,1} = \prod_{f\in S}\Pr[X_f=1]$ and (P3) will hold.

Let us now prove (5) for a fixed k. In iteration k, exactly one of the following three events occur:

Event A: Two edges in S have their values modified. Specifically, let $A(f_1,f_2,\alpha,\beta)$ denote the event that edges $\{f_1,f_2\}\subseteq S$ have their values changed in the following probabilistic way:

$$(Y_{f_1,k},Y_{f_2,k}) = \begin{cases} (Y_{f_1,k-1}+\alpha, Y_{f_2,k-1}-\alpha) & \text{with probability } \beta/(\alpha+\beta) \\ (Y_{f_1,k-1}-\beta, Y_{f_2,k-1}+\beta) & \text{with probability } \alpha/(\alpha+\beta)\end{cases}$$

Suppose, for each $f\in S$, $Y_{f,k-1}$ equals some fixed a_f. Let $S_1 = S-\{f_1,f_2\}$. Then,

$$\mathbf{E}[\prod_{f\in S} Y_{f,k}|(\forall f\in S,\, Y_{f,k-1}=a_f)\wedge A(f_1,f_2,\alpha,\beta)] =$$

$$\mathbf{E}[Y_{f_1,k}\cdot Y_{f_2,k}|(\forall f\in S,\, Y_{f,k-1}=a_f)\wedge A(f_1,f_2,\alpha,\beta)]\prod_{f\in S_1} a_f$$

The above expectation can be written as $(\psi + \Phi)\prod_{f \in S_1} a_f$, where

$$\begin{aligned}\psi &= (\beta/(\alpha+\beta)) \cdot (a_{f_1} + \alpha) \cdot (a_{f_2} - \alpha), \text{ and} \\ \Phi &= (\alpha/(\alpha+\beta)) \cdot (a_{f_1} - \beta) \cdot (a_{f_2} + \beta).\end{aligned}$$

It is easy to show that $\psi + \Phi \leq a_{f_1} a_{f_2}$. Thus, for any fixed $\{f_1, f_2\} \subseteq S$ and for any fixed (α, β), and for fixed values a_f, the following holds:

$$\mathbf{E}[\prod_{f \in S} Y_{f,k} | (\forall f \in S,\, Y_{f,k-1} = a_f) \wedge A(f_1, f_2, \alpha, \beta)] \leq \prod_{f \in S} a_f$$

Hence, $\mathbf{E}[\prod_{f \in S} Y_{f,k} | A] \leq \mathbf{E}[\prod_{f \in S} Y_{f,k-1} | A]$.

Event B: Exactly one edge in the set S has its value modified. Let $B(f_1, \alpha, \beta)$ denote the event that edge $f_1 \in S$ has its value changed in the following probabilistic way:

$$Y_{f_1,k} = \begin{cases} Y_{f_1,k-1} + \alpha & \text{with probability } \beta/(\alpha+\beta) \\ Y_{f_1,k-1} - \beta & \text{with probability } \alpha/(\alpha+\beta) \end{cases}$$

Thus, $\mathbf{E}[Y_{f_1,k} | (\forall f \in S,\, Y_{f,k-1} = a_f) \wedge B(f_1, \alpha, \beta)] = a_{f_1}$. Letting $S_1 = S - \{f_1\}$, we get that $\mathbf{E}[\prod_{f \in S} Y_{f,k} | (\forall f \in S,\, Y_{f,k-1} = a_f) \wedge B(f_1, \alpha, \beta)]$ equals

$$\mathbf{E}[Y_{f_1,k} | (\forall f \in S,\, Y_{f,k-1} = a_f) \wedge B(f_1, \alpha, \beta)] \prod_{f \in S_1} a_f = \prod_{f \in S} a_f.$$

Since this equation holds for any $f_1 \in S$, for any values a_f, and for any (α, β), we have $\mathbf{E}[\prod_{f \in S} Y_{f,k} | B] = \mathbf{E}[\prod_{f \in S} Y_{f,k-1}]$.

Event C: No edge has its value modified. Hence, $\mathbf{E}[\prod_{f \in S} Y_{f,k} | C] = \mathbf{E}[\prod_{f \in S} Y_{f,k-1}]$.

Thus by the above case-analysis that considers which of events A, B and C occurs, we get that $\mathbf{E}[\prod_{f \in S} Y_{f,k}] \leq \mathbf{E}[\prod_{f \in S} Y_{f,k-1}]$. This completes the proof. □

In the case of star-graphs, note that there are no cycles, and that any maximal path is of length 1 or 2. So, each iteration in this special case just needs constant time. Thus, recalling (3), we get our main theorem on dependent rounding:

Theorem 2.3 *Given an instance of dependent rounding on a bipartite graph (A, B, E), our algorithm runs in $O((|A| + |B|) \cdot |E|)$ time, and guarantees properties (P1), (P2), and (P3). Furthermore, if the bipartite graph is a star, our algorithm runs in linear time.*

2.2 An extension of (P3) for star-graphs

For star-graphs, we now show that our dependent-rounding approach of Section 2.1 has a much stronger negative correlation property than (P3):

Theorem 2.4 *Suppose we are given an instance of dependent rounding where (A, B, E) is a star-graph, with $A = \{u\}$ and u connected to all nodes in B. Then for any pair of disjoint subsets S_1 and S_2 of the edges incident on u, and any $b \in \{0, 1\}$, our dependent rounding satisfies*

$$\Pr[\bigwedge_{f \in S_2} (X_f = b) \mid (\bigwedge_{f' \in S_1} (X_{f'} = b))] \leq \Pr[\bigwedge_{f \in S_2} (X_f = b)].$$

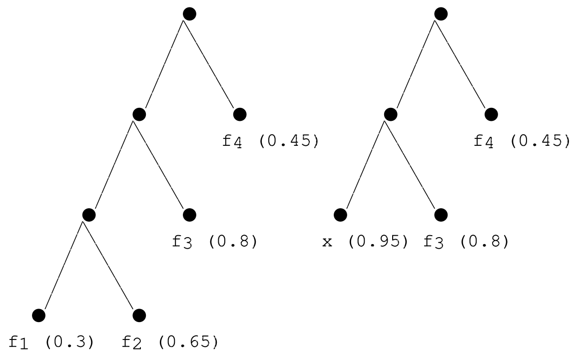

Figure 1: Pairing Trees: The tree on the left is the initial pairing tree. After one iteration, the second pairing tree is shown on the right. Note that the second pairing tree is the same irrespective of whether f_1 of f_2 gets rounded in the first tree.

It is easy to see that Theorem 2.4 implies (P3), but not *vice versa*. We anticipate that this strong correlation property will lead to some interesting applications.

Our proof of Theorem 2.4 is inspired by our proof of Lemma 2.2, 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. For ease of understanding and analysis, we will assume that the dependent rounding scheme chooses the maximal path in each iteration in the following way. Given a set L of labeled nodes, define a *pairing tree* of L to be any tree whose leaf-set is L, and in which each internal node has exactly two children. Consider any pairing tree whose leaf nodes correspond to the set of edges in the star-graph. We start with such a pairing tree. We update this tree with each iteration, so that the leaves will correspond to the current set of floating edges. During an iteration, if only one floating edge remains (this can happen only in the last iteration), we choose this edge as the maximal path. Otherwise, let two nodes at height zero in the pairing tree correspond to the edges f_1 and f_2 in the star. Our maximal path will consist of exactly these two edges. At the end of the iteration, the pairing tree is updated as follows. The two leaf nodes at height zero are deleted from the tree. The parent of the deleted nodes (which is now a leaf-node at height zero) will corresponded to the floating edge in the set $\{f_1, f_2\}$. If both the edges get rounded during the iteration, we create a new pairing tree whose leaves correspond to the current set of floating edges in the star-graph and use this tree for the next iteration. Figure 1 presents an illustration of the pairing tree. We fix such a pairing tree ahead of time, and then run dependent rounding.

Proof: Let $b = 0$ in Theorem 2.4 (the proof for the case where $b = 1$ is identical, and is omitted). Let T be the pairing tree at the beginning of the current iteration in the dependent rounding algorithm. Let l_T denote the number of leaf nodes in T. In the discussion below, for any $S \subseteq E$, let $\Pr[S] \doteq \Pr[\bigwedge_{s\in S} X_s = b]$. For any pairing tree T, let $\Pr[S \mid T] \doteq \Pr[S \mid T$ is the pairing tree in the current iteration$]$. Let S_1 and S_2 be any two disjoint subsets of E. We prove that

$$\forall \text{ pairing trees } T, \Pr[S_1 \cup S_2 \mid T] \leq \Pr[S_1 \mid T] \cdot \Pr[S_2 \mid T] \tag{6}$$

Setting T to be the initial pairing tree in (6) will then prove the theorem. The proof of (6) is by induction on l_T. The base case for induction is when $l_T \leq 2$, and this base case is easily seen to follow from (P3). Let us assume that (6) holds for all T' such that $l_{T'} \leq k$, where $k \geq 2$, and prove it for an arbitrary T such that $l_T = k + 1$. Let p and q, corresponding to edges p_1 and q_1, be the two nodes at height zero in T. At the beginning of the next iteration, we have an updated pairing tree T' where p and q have been deleted and their parent x in T is the new node at height zero (if both the edges got fixed in this iteration, we will have a new tree T'). In the discussion below, we let $\Pr'[S] \doteq \Pr[S \mid T']$. One of the following four cases occurs in the current iteration:

Case 1: $p_1 \in S_1$ **and** $q_1 \in S_2$**.** Let the edge values of p_1 and q_1 be α and β respectively. If $\alpha + \beta \geq 1$, one of the edges will get rounded to 1, making the L.H.S of (6) equal to zero. Hence, we will assume that $\alpha + \beta < 1$. At the end of this iteration, the edge values for (p_1, q_1) get modified to (α', β') respectively as follows:

$$(\alpha', \beta') = \begin{cases} (\alpha+\beta, 0) & \text{with probability } \alpha/(\alpha+\beta) \\ (0, \alpha+\beta) & \text{with probability } \beta/(\alpha+\beta) \end{cases}$$

Crucially, *irrespective* of which of the above events occur, T' has a new leaf node x with the corresponding edge value being equal to $\alpha + \beta$. Let $S_1' = S_1 \setminus \{p_1\}$ and $S_2' = S_2 \setminus \{q_1\}$. We thus have

$$\begin{aligned}
\Pr[S_1 \cup S_2 \mid T] &= \Pr{}'[S_1' \cup S_2' \cup \{x\}] \\
\Pr[S_1 \mid T] &= \frac{\beta}{\alpha+\beta}\Pr{}'[S_1'] + \frac{\alpha}{\alpha+\beta}\Pr{}'[S_1' \cup \{x\}] \\
\Pr[S_2 \mid T] &= \frac{\alpha}{\alpha+\beta}\Pr{}'[S_2'] + \frac{\beta}{\alpha+\beta}\Pr{}'[S_2' \cup \{x\}]
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Pr[S_1 \mid T] \cdot \Pr[S_2 \mid T] &= \left(\frac{\alpha}{\alpha+\beta}\right)^2 \Pr{}'[S_1' \cup \{x\}]\Pr{}'[S_2'] + \left(\frac{\beta}{\alpha+\beta}\right)^2 \Pr{}'[S_1']\Pr{}'[S_2' \cup \{x\}] + \\
&\quad \left(\frac{\alpha\beta}{(\alpha+\beta)^2}\right) \cdot (\Pr{}'[S_1']\Pr{}'[S_2'] + \Pr{}'[S_1' \cup \{x\}]\Pr{}'[S_2' \cup \{x\}]). \qquad (7)
\end{aligned}$$

So, in order to prove (6), we need to show that $\Pr{}'[S_1' \cup S_2' \cup \{x\}]$ is at most the r.h.s. of (7). To do so, we first use the induction hypothesis to get the following two bounds:

$$\begin{aligned}
\Pr{}'[S_1']\Pr{}'[S_2' \cup \{x\}] &\geq \Pr{}'[S_1' \cup S_2' \cup \{x\}] \text{ and} \qquad (8) \\
\Pr{}'[S_1' \cup \{x\}]\Pr{}'[S_2'] &\geq \Pr{}'[S_1' \cup S_2' \cup \{x\}] \qquad (9)
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
\Pr{}'[S_1']\Pr{}'[S_2'] + \Pr{}'[S_1' \cup \{x\}]\Pr{}'[S_2' \cup \{x\}] &\geq 2\sqrt{\Pr{}'[S_1'] \cdot \Pr{}'[S_2'] \cdot \Pr{}'[S_1' \cup \{x\}] \cdot \Pr{}'[S_2' \cup \{x\}]} \\
&= 2\sqrt{(\Pr{}'[S_1'] \cdot \Pr{}'[S_2' \cup \{x\}]) \cdot (\Pr{}'[S_2'] \cdot \Pr{}'[S_1' \cup \{x\}])} \\
&\geq 2\Pr{}'[S_1' \cup S_2' \cup \{x\}]. \qquad (10)
\end{aligned}$$

Plugging (8), (9) and (10) into (7), we get

$$\Pr[S_1 \mid T] \cdot \Pr[S_2 \mid T] \geq \left[\left(\frac{\alpha}{\alpha+\beta}\right)^2 + \left(\frac{\beta}{\alpha+\beta}\right)^2 + \frac{2\alpha\beta}{\alpha+\beta}\right] \cdot \Pr{}'[S_1' \cup S_2' \cup \{x\}] = \Pr{}'[S_1' \cup S_2' \cup \{x\}],$$

as required. This completes the proof for Case 1; the remaining three cases are simpler.

Case 2: $p_1, q_1 \in S_1$**.** Let $S_1' = S_1 \setminus \{p_1, q_1\}$. Again, we can assume that the sum of the edge-values $\alpha + \beta$ is less than one. In this case,

$$\begin{aligned}
\Pr[S_1 \mid T] &= \Pr{}'[S_1' \cup \{x\}] \\
\Pr[S_2 \mid T] &= \Pr{}'[S_2] \\
\Pr[S_1 \cup S_2 \mid T] &= \Pr{}'[S_1' \cup S_2 \cup \{x\}]
\end{aligned}$$

By the induction hypothesis, $\Pr'[S_1' \cup S_2 \cup \{x\}] \leq \Pr'[S_1' \cup \{x\}]\Pr'[S_2]$ and hence (6) follows. An identical argument holds for the case where $p_1, q_1 \in S_2$.

Case 3: $p_1 \in S_1$ **and** $q_1 \in E \setminus (S_1 \cup S_2)$**.** Let $S_1' = S_1 \setminus \{p_1\}$. Exactly one of two events happens during the iteration.

Event A: p_1 gets rounded to zero. In this case,

$$\begin{aligned}\Pr[S_1 \mid T] &= \Pr'[S_1'] \\ \Pr[S_2 \mid T] &= \Pr'[S_2] \\ \Pr[S_1 \cup S_2 \mid T] &= \Pr'[S_1' \cup S_2]\end{aligned}$$

By the induction hypothesis, $\Pr'[S_1' \cup S_2] \leq \Pr'[S_1']\Pr'[S_2]$ and hence (6) follows.

Event B: p_1 does not get rounded to zero. In this case,

$$\begin{aligned}\Pr[S_1 \mid T] &= \Pr'[S_1' \cup \{x\}] \\ \Pr[S_2 \mid T] &= \Pr'[S_2] \\ \Pr[S_1 \cup S_2 \mid T] &= \Pr'[S_1' \cup S_2 \cup \{x\}]\end{aligned}$$

By the induction hypothesis, $\Pr'[S_1' \cup S_2 \cup \{x\}] \leq \Pr'[S_1' \cup \{x\}]\Pr'[S_2]$ and hence (6) follows.

Case 4: $\{p_1, q_1\} \subseteq E \setminus (S_1 \cup S_2)$**.** In this case, $\Pr[S_1 \mid T] = \Pr'[S_1]$, $\Pr[S_2 \mid T] = \Pr'[S_2]$ and $\Pr[S_1 \cup S_2 \mid T] = \Pr'[S_1 \cup S_2]$; we are done by the induction hypothesis.

This completes the proof of Theorem 2.4. □

2.3 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; see, e.g., [38, 17]. If we can model such graphs using appropriate random graphs, then we can sample multiple times from such a model and test candidate algorithms, such as Web-crawlers [9]. A particularly successful outcome 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., [10]). Hence, there has been much interest in generating (and studying) random graphs with a given degree-sequence (see, e.g., [23]). 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, values $\{x_{i,j} \in [0,1] : i < j\}$, and a degree-sequence $d_1, d_2, \ldots, d_n$ (realized by the values $x_{i,j}$), 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.

Our dependent rounding scheme solves this problem when G is bipartite. However, 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 $x_{i,j}$ 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. One method in this context is to construct a random graph where each edge (i, j) is put in *independently*, with probability $x_{i,j}$. This preserves (A2), but does not do well with (A1): the only (high-probability) guarantee we get is that for each i, $|D_i - d_i| \leq O(\max\{\sqrt{d_i \log n}, (\log n)^{1-o(1)}\})$. We now show that we can do much better than this:

Theorem 2.5 *Given a degree-sequence $d_1, d_2, \ldots, d_n$, and values $\{x_{i,j} \in [0, 1] : i < j\}$, we can efficiently generate an n-vertex random graph for which (A2) holds, and where the following relaxation of (A1) holds: with probability one for each vertex i, its (random) degree D_i satisfies $|D_i - d_i| \leq 2$. Letting m denote the number of nonzero $x_{i,j}$, the running time of our algorithm is $O(n + m^2)$.*

Thus, we get an essentially best-possible result. Recall that, in the bipartite rounding algorithm, if we encounter an even cycle, we "break" this cycle by probabilistically rounding (at least) one of the edges in the cycle. Our algorithm for non-bipartite graphs also proceeds by probabilistically breaking cycles in the graph. We now describe the details of the algorithm.

We start with a graph with vertices $1, 2, \ldots, n$; for each nonzero value $x_{i,j}$, we put an edge between i and j that has a value or label $x_{i,j}$. We will closely follow our algorithm of Section 2.1, and borrow notation such as "floating edges" from there. In the following description, we use the terms *simple cycle* and *linked odd cycles* in the following sense: each vertex in a simple cycle has degree two; a pair of linked odd cycles is a pair of odd cycles sharing a common vertex that has degree four. The algorithm proceeds in four phases as follows. Throughout the execution of the algorithm, G will denote the subgraph given by the *currently-floating* edges of F.

Phase 1: While there exists a simple even cycle in G, do:
Pick a simple cycle C from G. Partition the edges in C into matchings M_1 and M_2. Probabilistically modify the edge values of M_1 and M_2 as in the bipartite rounding algorithm.

Phase 2: While there exists a pair of linked odd cycles in G, do:
Pick a pair of linked odd cycles C from G. Partition the edges in C into two sets M_1 and M_2 such that for any given vertex, the number of edges incident upon it in M_1 is the same as that in M_2. (It is easy to see that such a partition exists since C is a linked pair of odd cycles). Probabilistically modify the edge values of M_1 and M_2 as in the bipartite rounding algorithm (M_1 and M_2 were matchings in the case of bipartite graphs.)

Phase 3: While there exists an odd cycle in G, do:
Pick an odd cycle C from G and pick an arbitrary edge e in C. Let Y_e be the random variable which denotes the value of e's edge-label. Let the current value of Y_e be y_e. Round Y_e to one with probability y_e and to zero with the complementary probability.

Phase 4: All cycles in G have been broken by the previous phases and G is now a forest. Apply the bipartite rounding algorithm on G.

We now argue that the two properties claimed by Theorem 2.5 hold. For any fixed edge, the expected value of the edge-label does not change in any of the phases. Hence we see that (A1) holds, by using the same simple argument as in the proof of Lemma 2.1. Phases **1** and **2** do not change the fractional degree of any vertex. Crucially, each vertex belongs to *at most one* odd cycle at the beginning of Phase **3**. Thus, Phases **3** and **4** change the degree of any vertex by at most one each. Hence, at the end of our algorithm, the integral degree of any vertex differs from its fractional degree by at most two.

We now discuss how to implement this algorithm. We first decompose the graph into its biconnected components [16]. Some biconnected components are *trivial*, if they consist of a single edge. Other biconnected components always contain a cycle. The following proposition shows that it is easy to find an even cycle in a non-trivial biconnected component. Before we understand the proof, we need to define the concept of *bridges* of a graph $G = (V, E)$ with respect to a cycle C. A trivial bridge is an edge of the graph that connects two nodes on C that are not adjacent in C. These are simply chords on the cycle. Consider the graph induced by the vertices in $V \setminus C$. Let $B_1, \ldots, B_k$ be the connected components in this graph. Let E_i be the set of edges that connect vertices in B_i to vertices on C. The edges E_i together with the component B_i form a bridge in the graph [16]. If an edge $(u, v) \in E_i$ with $u \in B_i$ and $v \in C$, then v is an attachment point of the bridge.

Proposition 2.6 *A non-trivial biconnected simple graph is either exactly an odd cycle, or must contain an even cycle.*

Proof: Assume that the biconnected component is not exactly an odd cycle. Find a cycle C in the biconnected component. Assume that the cycle is odd. Consider the bridges of the graph with respect to the cycle C. Since the graph is biconnected, each bridge has at least two distinct attachment points on C. This yields a path in the graph that is disjoint from C that connects two nodes u and v on C. Since C has odd length, the two paths between u and v using edges of C are of opposite parity. Using one of them along with the path that avoids C we obtain a simple cycle of even length. $\square$

Phase 1 is implemented as follows. If a biconnected component is trivial, or an odd cycle, we do not process it for now. We process each remaining component to identify even cycles (the proof of Proposition 2.6 suggests how to do this algorithmically in linear time). Once we remove an edge of the even cycle, we further decompose the graph into its biconnected components in linear time. We repeat this until each biconnected component is either trivial, or exactly an odd cycle.

In Phase 2 we find linked odd cycles. If two components are non-trivial and share a common cut vertex, they form a pair of linked odd cycles. We can perform the rounding as described above, and delete one edge to break a cycle. Eventually, all odd cycles are disjoint and we can perform the rounding as in Phase 3. Finally, when the graph is acyclic, it is bipartite and the rounding can be done as described in Section 2.1. The total running time of the algorithm is $O(n + m^2)$. Phase 1 is the most expensive since each time we delete one edge, we have to reconstruct the biconnected components, which takes time linear in the size of the component.

3 Low-congestion Multi-path Routing

For the rest of the paper, we see various applications of our dependent rounding algorithm of Section 2.1, in the context of approximation algorithms. We start with a routing problem in this section. In this problem, we are given a graph $G = (V, E)$ with a capacity $c_f > 0$ for each edge f, along with k pairs of vertices (s_i, t_i). For each $i \in [k]$, we are also given: (i) a demand $\rho_i > 0$, (ii) a collection $\mathcal{P}_i$ of (s_i, t_i)–paths, and (iii) an integer $1 \leq \ell_i \leq |\mathcal{P}_i|$. The objective is to choose ℓ_i paths from $\mathcal{P}_i$ for each i, in order to minimize the *relative congestion*: the maximum, over all edges f, of $(1/c_f)$ times the total demand of the chosen paths that pass through f. (We make the usual *balance assumption* [27]: if edge f lies in a path $P \in \mathcal{P}_i$, then $c_f \geq \rho_i$.) The case where $\ell_i = 1$ for all i is a classical problem, and is studied in [32, 27]. Our problem with arbitrary ℓ_i, in addition to its intrinsic interest, is motivated by the following optical networking problem. Given their high data rates (Gigabits/second and Terabits/second), a key requirement in optical networks is fast *restoration* from node/edge failures [29, 36, 13, 33, 11]. Many restoration strategies have been studied/deployed. A popular scheme that guarantees full recovery from single node/edge failures is variously called *1+1 Dedicated Protection*, *1+1 Restoration*, etc.: the same signal is transmitted over an active route and a *disjoint* backup route, hence being robust to single node/edge failures [13, 33, 11]. The obvious extension to protect against multiple node/edge failures has also been studied. To generate such paths in practice, min-cost max-flow is used to generate a large number of disjoint paths for each vertex pair, and a "suitable" subset of these paths is chosen in some heuristic way to minimize the congestion of the routing. We present an approximation algorithm for this problem with the same approximation ratio as for the case where $\ell_i = 1$ for all i [32, 27]; see Theorem 3.2. (In the above optical routing application, each $\mathcal{P}_i$ is a set of node/edge-disjoint paths; this does not seem to provide any improvement to the approximability, in our setting as well as in that of [32, 27].)

We start by recalling a result of [31], which shows that the negative correlation property (P3) has the following interesting large-deviations consequence. Recall from the introduction that $D(t;P)$ denotes the distributions obtained by our dependent rounding scheme for the special case of star-graphs. Suppose $P = (p_1, p_2, \ldots, p_t)$ is such that $\sum_{i \in [t]} p_i = l$ for some integer l, and that we sample a vector $(X_1, X_2, \ldots, X_t)$ from $D(t;P)$. Then, for any given sequence $a_1, a_2, \ldots, a_t \in [0,1]$, the random variable $\sum_i a_i X_i$ is sharply concentrated around its mean; in fact, the tail bounds on $\sum_i a_i X_i$ are at least as good as any bound obtainable by a Chernoff-type approach [31].

Theorem 3.1 ([31]) *Let $a_1, a_2, \ldots, a_t$ be reals in $[0,1]$, and $X_1, X_2, \ldots, X_t$ be random variables taking values in $\{0,1\}$.*

(i) Suppose $\forall S \subseteq [t] \quad \Pr[\bigwedge_{i \in S}(X_i = 1)] \leq \prod_{i \in S} \Pr[X_i = 1]$; also suppose $\mathbf{E}[\sum_i a_i X_i] \leq \mu_1$. Then, for any $\delta \geq 0$,

$$\Pr[\sum_i a_i X_i \geq \mu_1(1+\delta)] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu_1}.$$

(ii) Suppose $\forall S \subseteq [t] \quad \Pr[\bigwedge_{i \in S}(X_i = 0)] \leq \prod_{i \in S} \Pr[X_i = 0]$; also suppose $\mathbf{E}[\sum_i a_i X_i] \geq \mu_2$. Then, for any $\delta \in [0,1]$,

$$\Pr[\sum_i a_i X_i \leq \mu_2(1-\delta)] \leq e^{-\mu_2 \delta^2/2}.$$

Let us return to our low-congestion routing problem. Suppose $\mathcal{P}_i = \{P_{i,1}, P_{i,2}, \ldots\}$. There is a natural integer programming (IP) formulation for the problem, with a variable $z_{i,j}$ for each (i,j): $z_{i,j} = 1$ if $P_{i,j}$ is chosen, and is 0 otherwise. The LP relaxation lets each $z_{i,j}$ lie in $[0,1]$. Let the variable C^* denote the optimal fractional relative congestion. We have the constraints:

$$\forall i, \ \sum_j z_{i,j} = \ell_i; \qquad \forall f \in E, \ \sum_{(i,j):\ f \in P_{i,j}} \rho_i z_{i,j} \leq c_f C^*. \tag{11}$$

We now prove that we can apply dependent rounding to get a good approximation algorithm:

Theorem 3.2 *Let m denote the number of edges in G. We can efficiently round the fractional solution so that the resultant (integral) relative congestion C satisfies the following, with high probability:*

$$C \leq O\left(\frac{\log m}{\log(2 \log m / C^*)}\right) \ \textit{if } C^* \leq \log m; \tag{12}$$

$$C \leq C^* + O(\sqrt{C^* \log m}) \ \textit{if } C^* > \log m. \tag{13}$$

Proof: Let $|\mathcal{P}_i| = t_i$. Suppose $r_i = (z^*_{i,j} : j \in [t_i])$ is the vector of values for the $z_{i,j}$ in the optimal fractional solution. Independently for each $i \in [k]$, we sample from $D(t_i; r_i)$, and select the paths that are chosen (i.e., rounded to 1) by this process. By the first family of constraints in (11) and (P2), we have with probability 1 that ℓ_i paths are chosen for each i. Next, by (P1) and the second family of constraints in (11), the expected relative congestion on any given edge f is at most C^*. Suppose the constants implicit in the $O(\cdot)$ notation of (12) and (13) are large enough. Then, (P3) and Theorem 3.1(i) together show that for any given edge f, the probability that it gets relative congestion more than C is at most $1/(2m)$. (To see this, let $\mu_1 = C^*$ and $C = C^*(1+\delta)$. A standard calculation shows that if the constants in the $O(\cdot)$ notation of (12) and (13) are large enough, then the bound of Theorem 3.1(i) is at most $1/(2m)$.) Adding over all m edges, we get a relative congestion of at most C, with probability at least 1/2. As usual, this probability can be boosted by repetition. □

We are not aware of any other approach that will yield Theorem 3.2. If we round the values $z^*_{i,j}$ independently, the probability of choosing a sufficient number of paths can be as small as $2^{-\Theta(k)}$. If we instead try to bin-pack the $z^*_{i,j}$ for each given i and round independently from each bin, the approximation ratio can be about twice our approximation ratio (e.g., if most of the $z^*_{i,j}$ are just above 1/2, the optimal bin-packing will need roughly $2\ell_i$ bins). Note from (13) that we get $(1+o(1))$–approximations for families of instances where C^* grows faster than $\log m$; it appears difficult to get such results from any other approach that we are aware of.

4 Broadcast Scheduling

In this section, we study three related scheduling problems in a *broadcasting* model. 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., users waiting to receive the same topic in a broadcast setting. For example, all waiting users get satisfied when that topic is broadcast [6, 26, 14, 21, 4, 5]. The basic features of the model, common to all three problems, are as follows. There is a set of pages or topics, $P = \{1, 2, \ldots, n\}$, that can be broadcast by a broadcast server. We assume that time is discrete; for an integer t, the time-slot (or simply *time*) t is the window of time $(t-1, t]$. Any subset of the pages can be requested at time t. All users receive every page that is broadcast; the main problem in all three variants is to construct a good broadcast-schedule. The default assumption is that the server can broadcast at most one page at any time; in Section 4.4, we will also consider "2-speed" solutions where the server is allowed to broadcast up to two pages per time-slot. We work in the offline setting in which the server is aware of all future requests. How does a user-request get satisfied? Suppose a user requests page p at time t. Then, there is a parameter k such that this request gets satisfied when page p has been broadcast at k different time-slots t' that are larger than t. The "natural" choice for k, as in [26, 14, 21, 4, 5] (and as in the preliminary version of this work [22]), is 1; this is the choice considered in Section 4.4. However, recent advances in broadcast under lossy conditions (see, e.g., [30]) motivate us to consider the case of general k, as follows. Suppose the broadcast medium is lossy, and that packets can get dropped with some probability. A class of interesting "universal erasure codes" have been presented recently [30]; in our context, they work as follows. Suppose a page p is composed of some s input symbols. The broadcast server is *rateless* and can generate an arbitrary number of encoding symbols; the s required symbols can be recovered from *any* $s + O(\sqrt{s}\log^2 s)$ of the symbols received from the server, with high probability [30]. Thus, this is a general scheme that works with a variety of loss models; it naturally motivates our consideration that k broadcasts of page p suffice to recover page p with high probability. (We will also study a variant of this in Section 4.3.)

Further problem-specific details will be presented in Sections 4.2, 4.3, and 4.4. Informally, Sections 4.2 and 4.3 deal with maximizing the number of "satisfied" users, while Section 4.4 requires all users to be satisfied and to have "short" waiting times on the average. All three of these sections make use of a generic "random offsetting plus dependent-rounding" algorithm, which is described next.

4.1 A Generic Algorithm

This algorithm takes as input a fractional solution $\mathcal{S}$, where fractional quantities of pages are broadcast at each time unit. As far as this section is considered, $\mathcal{S}$ is a collection of non-negative values $\{y_t^p\}$ where p indexes pages and t indexes time-slots, such that: (i) in the setting of Sections 4.2 and 4.3, $\sum_p y_t^p \leq 1$ for all t; (ii) in the setting of Section 4.4, $\sum_p y_t^p \leq 2$ for all t.

We now give the reader a sense of what the variables y_t^p will mean, when we later apply the generic algorithm of this section. Each of Sections 4.2, 4.3, and 4.4 will start with an IP formulation for the problem at hand, wherein a binary variable y_t^p is 1 iff page p is broadcast at time slot t. There will be the further constraint that at most one page can be broadcast at any time: i.e., $\forall t, \sum_p y_t^p \leq 1$; additional constraints (which do not matter for now) will also be present. The LP relaxation will then be solved; in particular, y_t^p will be allowed to be a real in the range $[0, 1]$. A "fractional solution" $\mathcal{S}$ will then be fed as input to the "generic algorithm" that we are going to present next. In Sections 4.2 and 4.3, $\mathcal{S}$ will be the set of values $\{y_t^p\}$ in an optimal solution to the LP relaxation; in Section 4.4, $\mathcal{S}$ will be the set of values $\{y_t^p\}$ that are *twice* the values in an optimal solution to the LP relaxation.

Given $\mathcal{S}$, the generic algorithm proceeds in two steps as described below.

Step 1. Construct a bipartite graph $G = (U, V, E)$ as follows. U consists of vertices that represent time slots. Let u_t denote the vertex in U corresponding to time t. Consider a page p and the time instances during which page p is broadcast fractionally in $\mathcal{S}$. 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 some number $m = m(p)$ of windows, $W_j^p, 1 \leq j \leq m$, such that in each window except the first and the last, *exactly* one page p is broadcast fractionally. More formally, we will define non-negative values $b_{t,j}^p$ for each time-slot t and window W_j^p, such that for each j: $b_{t,j}^p$ is derived from y_t^p in a natural way, the values t for which $b_{t,j}^p \neq 0$ form an interval, and $\sum_t b_{t,j}^p = 1$ for $2 \leq j \leq m-1$. (The first and last windows, W_1^p and W_m^p, 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. (It suffices to use the same z for all pages p.) Intuitively, the windows represent contiguous chunks of page p broadcast in $\mathcal{S}$. The first chunk is of size z, the last chunk is of size at most one, and all other intermediate chunks are of size exactly one. Formally, 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 toward the fractional broadcast of page p in W_j^p. For all h, define $b_{t_h,1}^p$ and $b_{t_h,j}^p$, for $j \geq 2$ 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 $j \geq 2$, 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 the total number of windows $m_p \in \{\lceil \sum_{t'} y_{t'}^p \rceil, \lceil \sum_{t'} y_{t'}^p \rceil + 1\}$. The vertex set V consists of vertices that represent pages. For each page p, we have vertices $v_1^p, v_2^p, \ldots, v_{m_p}^p$ in V. For all p and j, v_j^p is connected to vertices corresponding to timeslots in W_j^p. The value of an edge (v_j^p, u_{t_h}) is equal to $b_{t_h,j}^p$. The above is repeated for all pages p, using the same random value z. This construction is illustrated in Figure 2, in which a subgraph of G that is induced on the vertices and edges relevant to a particular page p are shown. For this example we choose $z = 1$ and $y_j^p, t_1 \leq j \leq t_7$, values are $0.3, 0.3, 0.5, 0.5, 0.2, 0.9, 0.8$.

Step 2. Perform dependent rounding in G. If an edge (v_k^p, u_{t_h}) gets chosen in the rounded solution, then we broadcast page p at time t_h.

This concludes the description of the generic algorithm. As we shall see, the use of the "random offset" z is critical in guaranteeing the performance of the algorithms that follow.

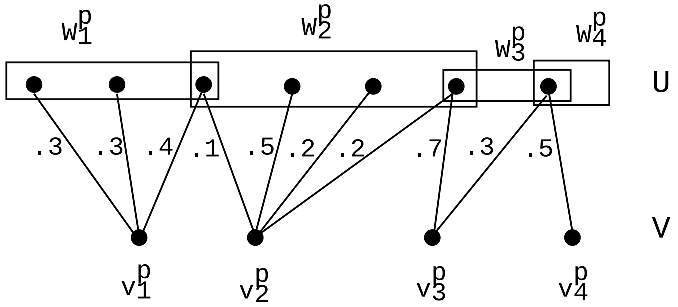

Figure 2: Subgraph of G relevant to page p whose $y_j^p, t_1 \leq j \leq t_7$, values are $0.3, 0.3, 0.5, 0.5, 0.2, 0.9, 0.8$. We set $z = 1$ here.

4.2 Throughput Maximization

This maximization problem is informally as follows. The server can broadcast at most one page at any time. Our goal is to schedule the broadcast of pages so that a "large" number of the requests are satisfied.

More formally, a request for page p that arrives at time t is denoted (p, t); the request is for k units of page p, as mentioned at the beginning of Section 4. Each request also has a weight, and all our results for the unweighted case also hold for the weighted case. The general problem is: given a set of time-windows in which each of the requests can be processed, to come up with a broadcast-schedule which maximizes the total weight of satisfied requests. These problems (and several generalizations) have been shown approximable to within 1/2 [5]. We consider the special case where each request (p, t) comes with a *deadline* $D_t^p > t$. This problem has been shown to be NP-hard by Gailis and Khuller [19]. A time slot t' serves (p, t) iff $t < t' \leq D_t^p$; thus, (p, t) is satisfied iff there are at least k such distinct time-slots t'. We now present an algorithm based on dependent rounding with the following guarantee: the algorithm either proves that there is no solution where every user is satisfied, or constructs a schedule in which each user has a probability at least 3/4 of being satisfied. (In particular, in the latter case, the expected number of requests satisfied is at least 3/4th of the total.)

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. Let T be the time of last request for any page. We use the following IP formulation. The binary variable $y_{t'}^p$ is 1 iff page p is broadcast at time slot t'; the binary variable x_t^p is 1 iff request (p, t) is satisfied at time $t', t < t' \leq D_t^p$. The first set of constraints ensure that whenever a request (p, t) is satisfied, k units of page p is broadcast at times $t', t < t' \leq D_t^p$. 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 $0 \leq x_t^p, y_{t'}^p \leq 1$, we obtain the LP relaxation for the problem.

$$
\begin{array}{ll}
\text{Maximize} \sum_{(p,t)} r_t^p \cdot x_t^p & \\
\left(\sum_{t'=t+1}^{D_t^p} y_{t'}^p \right) - k x_t^p \geq 0 & \forall p, t, t' > t \\
\sum_p y_{t'}^p \leq 1 & \forall t' \\
x_t^p \in \{0, 1\} & \forall p, t \\
y_{t'}^p \in \{0, 1\} & \forall p, t'
\end{array}
\qquad (14)
$$

We solve the LP relaxation of the IP to get an optimal fractional solution. If $x_t^p < 1$ for some request (p,t), we announce that not all requests can be satisfied simultaneously, and halt. Otherwise, we feed the optimal fractional values $\{y_t^p\}$ as input to the generic algorithm of Section 4.1. We now analyze this algorithm, starting with a simple lemma:

Lemma 4.1 *Our algorithm broadcasts at most one page at any time slot.*

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 lemma follows. □

The main issue left is to see how many of the requests get satisfied. We may assume that $x_t^p = 1$ for all requests (p,t); if not, we would have announced "infeasible problem" and halted. Consider a request $r = (p,t)$; we will now lower-bound the probability that there are k broadcasts within its deadline. The k units of fractional solution received by r can span at most $k+1$ adjacent windows. Specifically, the first and the last window together fractionally provide one unit of page p and the intermediate windows provide $k-1$ units of page p to r. The fraction of page p broadcast at time t' does not serve r if $t' \leq t$ and belongs to the first of these windows, or if $t' > D_t^p$ and belongs to the last of these windows. For example, in Figure 2, suppose the request $r = (p,t)$ arrives at time $t = t_4$. Then, W_2^p is the first window that serves r; the fraction of page p broadcast at times t_3 and t_4 (which are 0.1 and 0.5 respectively) do not serve r. Let $Y(r)$ be the following random variable that is determined by the first step of our generic algorithm: $Y(r)$ is the fraction of page p broadcast in the first window that serves r. (Hence, the last window serves a fraction of $1 - Y(r)$ of page p to r.) As just seen, $Y(r) = 0.2 + 0.2 = 0.4$, if $t = t_4$ in Figure 2.

Let S_r be the indicator random variable that denotes r getting satisfied in the rounded solution; let $S_r(y)$ denote this random variable when we condition on the event "$Y(r) = y$". We now prove a useful lower-bound on $\mathbf{E}[S_r(y)] = \mathbf{E}[S_r \mid Y(r) = y]$:

Lemma 4.2 *For any y, $\mathbf{E}[S_r(y)] \geq \max\{y, 1-y\}$.*

Proof: Let us condition on the event "$Y(r) = y$". Let A_y and B_y denote the set of time slots in the first and last windows respectively, which serve request r. Let R_i^p be the indicator random variable which denotes if page p was broadcast at time instant i in the rounded solution. Since the number of units of p which serve r in the rounded solution is $k-1+\max(\sum_{i\in A_y} R_i^p, \sum_{i\in B_y} R_i^p)$, we have $S_r(y) = \max\{\sum_{i\in A_y} R_i^p, \sum_{i\in B_y} R_i^p\}$. Thus,

$$\mathbf{E}[S_r(y)] = \Pr[(\sum_{i\in A_y} R_i^p = 1) \bigvee (\sum_{i\in B_y} R_i^p = 1)] \geq \max\{\Pr[(\sum_{i\in A_y} R_i^p) = 1], \Pr[(\sum_{i\in B_y} R_i^p) = 1]\} = \max\{y, 1-y\}.$$

□

We now show that each request r has probability at least 3/4 of getting satisfied:

Lemma 4.3 $\mathbf{E}[S_r] \geq 3/4$.

Proof: Observe that $Y(r)$ is uniformly distributed in $(0,1]$. Thus,

$$\mathbf{E}[S_r] = \int_0^1 \mathbf{E}[S_r(y)]\, dy = \int_0^1 \max\{y, 1-y\}\, dy = \int_0^{1/2} (1-y)\, dy + \int_{1/2}^1 y\, dy = \frac{3}{4}.$$

□

Thus we get the following theorem, via the linearity of expectation.

Theorem 4.4 *Given an instance of the throughput-maximization problem, our algorithm: (i) either proves that not all requests are satisfiable, or (ii) constructs a random schedule in which the expected fraction of requests satisfied is at least* $3/4$. *Furthermore, in case (ii), any given request is satisfied with probability at least* $3/4$.

4.3 Proportional Throughput Maximization

The "proportional throughput maximization" problem is defined as follows. Each request (p,t) requires k units of page p by its deadline D_t^p. If $k' \leq k$ units of page p are broadcast from time $t+1$ to D_t^p, then (p,t) has a benefit of k'/k. At most one page may be broadcast at any time and our goal is to maximize the sum of the benefits of all requests. (We also consider a "per-user guarantee" variant in Section 4.3.3.) We consider this model for two reasons. First, it becomes the clean problem of approximating maximum throughput, in the special case of $k=1$. Second, if we have a loss model where k' broadcasts of page p will help us recover page p with probability at least k'/k, our current problem serves as a useful model. We use the generic algorithm of Section 4.1 for this problem and obtain an approximation ratio of $(1-1/(4k))$. In particular, we get an approximation ratio of $3/4$ when $k=1$. Bar-Noy *et al.* [5] consider the case where $k=1$, and present a $1/2$–approximation in a general setting.

Two variants are also considered, in Sections 4.3.3 and 4.3.4. Theorem 4.8 summarizes the three main results we obtain in Section 4.3.

4.3.1 Algorithm

We solve the LP relaxation of the following IP formulation, and input the fractional solution $\mathcal{S}$ to the generic algorithm. Note that this IP is essentially a scaling of IP (14).

$$\begin{array}{ll}
\text{Maximize} \sum_{(p,t)} r_t^p \cdot x_t^p/k & \\
\left(\sum_{t'=t+1}^{D_t^p} y_{t'}^p\right) - x_t^p \geq 0 & \forall p,t,t'>t \\
\sum_p y_{t'}^p \leq 1 & \forall t' \\
x_t^p \in \{0,1,\ldots,k\} & \forall p,t \\
y_{t'}^p \in \{0,1\} & \forall p,t'
\end{array} \qquad (15)$$

The LP relaxation lets all the x_t^p and y_t^p to be reals, subject to the constraints $0 \leq x_t^p \leq k$ and $0 \leq y_t^p \leq 1$.

4.3.2 Analysis

Consider a request $r=(p,t)$. Let $0 \leq x_r \leq k$ be the fractional amount of service received by r in the fractional solution. The x_r units of service span at most $\lceil x_r \rceil + 1$ windows. As in Section 4.2, let the amount of service provided by the first window to r be denoted $Y = Y(r)$. As in Section 4.2, Y is distributed uniformly at random in the range $(0,1]$. Let $0 \leq X_r \leq k$ be the random variable which denotes the number of integral units of service received by r in the rounded solution. Let R_i^p be the indicator random variable which is one iff page p is broadcast at time i in the rounded solution.

Lemma 4.5 *Let* $x_r \leq k-1$. *Then,* $\mathbf{E}[X_r] = x_r$.

Proof: The x_r fractional units of page p which serve r in the fractional solution, span at most $\lceil x_r \rceil + 1 \leq k$ windows. Hence, $X_r \leq k$. Therefore, $X_r = \sum_{i=t+1}^{D_t^p} R_i^p$. Hence $\mathbf{E}[X_r] = \sum_{i=t+1}^{D_t^p} \mathbf{E}[R_i^p] = \sum_{i=t+1}^{D_t^p} x_i^p = x_r$. $\square$

Lemma 4.6 *Let $x_r = k - 1 + f$, where $f \in (0, 1]$. Then, $\mathbf{E}[X_r] \geq x_r - \frac{f^2}{4}$.*

Proof: Suppose Y equals some value y. There are two cases:

Case 1: $f < y \leq 1$**.** In this case, the x_r units of service in the fractional solution span exactly k windows. By the same arguments as in the proof of Lemma 4.5, we have $\mathbf{E}[X_r \mid (Y = y)] = x_r$ in this case.

Case 2: $0 < y \leq f$**.** In this case, the x_r units of service in the fractional solution span exactly $k+1$ windows. In addition, the first window provides a service of y, the last window provides a service of $f - y$ and the intermediate $k - 1$ windows provide $k - 1$ units of service. Let A_y and B_y denote the set of time slots in the first and last windows respectively which serve request r. $X_r = k$ if one of the slots in A_y or B_y broadcasts p in the rounded solution and $X_r = k - 1$ otherwise. Thus, $X_r = k - 1 + \max(\sum_{i \in A_y} R_i^p, \sum_{i \in B_y} R_i^p)$. So in this case,

$$\begin{aligned}
\mathbf{E}[X_r \mid (Y = y)] &= k - 1 + \Pr[(\sum_{i \in A_y} R_i^p = 1) \bigvee (\sum_{i \in B_y} R_i^p)] \\
&\geq k - 1 + \max\{\Pr[(\sum_{i \in A_y} R_i^p) = 1], \Pr[(\sum_{i \in B_y} R_i^p) = 1]\} \\
&= k - 1 + \max\{y, f - y\}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\mathbf{E}[X_r] &\geq \int_0^f (k - 1 + \max\{y, f - y\})\, dy + \int_f^1 x_r\, dy \\
&= \int_0^f (k - 1 + \max\{y, f - y\})\, dy + (1 - f)x_r \\
&= (k - 1)f + \frac{3f^2}{4} + (1 - f)x_r \\
&= (k - 1)f + x_r - f(k - 1 + f) + \frac{3f^2}{4} \\
&= x_r - \frac{f^2}{4}.
\end{aligned}$$

$\square$

Lemma 4.7 *Let $\alpha_r = \mathbf{E}[X_r]/x_r$. Then, $\alpha_r \geq 1 - 1/(4k)$. Also, if $k = 1$, then $\alpha_r \geq 1 - x_r/4 \geq 3/4$.*

Proof: If $x_r \leq k - 1$, then Lemma 4.5 implies this claim. If $x_r = k - 1 + f$ for some $f \in (0, 1]$, then Lemma 4.6 yields

$$\alpha_r \geq 1 - \frac{f^2}{4x_r} \geq 1 - \frac{f}{4(k - 1 + f)} \geq 1 - \frac{1}{4k}.$$

If $k = 1$, then $x_r = f$, and $\alpha_r \geq 1 - \frac{x_r^2}{4x_r} = 1 - x_r/4 \geq 3/4$. $\square$

Lemma 4.1 shows that the solution is feasible since at most one page is broadcast at any time. Thus by the linearity of expectation, we have an $(1 - 1/(4k))$–approximation in expectation, for the problem of proportional throughput maximization.

4.3.3 A "per-user guarantee" variant

The discussion above can easily be extended to the following "per-user guarantee" setting. Consider the case where $k = 1$; throughput and proportional throughput coincide in this case. Instead of maximizing (proportional) throughput, suppose we aim for a "fair" schedule, where each individual has a reasonably large probability of getting its request met within its deadline. More precisely, suppose we have the "maxmin" problem of maximizing the minimum probability of satisfaction, over all users. Our IP can be modified to the following LP that models this problem. We now let each x_t^p and y_t^p be a real in $[0,1]$; we add a new variable q and add the constraint "$x_t^p \geq q$" for all q, and the objective function is to maximize q. The semantics of these variables with respect to a randomized schedule, is as follows. x_t^p denotes the probability that the request (p,t) gets satisfied, and y_t^p is the probability that page p is broadcast at time t. Also, q is the minimum of all the x_t^p values. It is easy to see that this new LP is a valid relaxation of our "per-user" problem: if there is a randomized schedule where each (p,t) has a probability at least q' of being satisfied, then this new LP indeed has a feasible solution with objective function value at least q'.

We now proceed with the same algorithm as in Section 4.3.1, except that the LP solved now is the one of the previous paragraph. Let q^* be the optimal value of the LP. Then, Lemma 4.7 shows that in our final randomized schedule, each request has probability at least $q_r^* - (q_r^*)^2/4$ of getting satisfied. So, we have a $3/4$–approximation to the problem of maxmin fairness; furthermore, under "heavy traffic" conditions where q_r^* can be small, our approximation ratio of $1 - q_r^*/4$ actually approaches one.

4.3.4 Arbitrary time-windows

We now briefly consider the following throughput-maximization problem in the case $k = 1$. When a request for a page p arrives at time t, it also specifies a set $S(p,t)$ of time slots larger than t; this request is satisfied iff page p is broadcast in one of the time-slots that lies in $S(p,t)$. The throughput-maximization problem here has been shown to be approximable to within $1/2$ in a general setting [5]; we now present an $(1-1/e)$–approximation algorithm. The LP is the same as that of Section 4.3.1, except that the first family of constraints "$\left(\sum_{t'=t+1}^{D_t^p} y_{t'}^p\right) - x_t^p \geq 0$" is replaced by the family of constraints "$(\sum_{t' \in S(p,t)} y_{t'}^p) - x_t^p \geq 0$". We solve the LP relaxation, and the randomized rounding is now the following very special case of dependent rounding. Independently for each time-slot t', we choose at most one page p: the probability of choosing page p equals $y_{t'}^p$. (This is possible since $\sum_p y_{t'}^p \leq 1$.) Now consider a request for a page p that arrives at time t. The probability of this request being satisfied is

$$1 - \prod_{t' \in S(p,t)} (1 - y_{t'}^p) \geq 1 - e^{-\sum_{t' \in S(p,t)} y_{t'}^p} \geq 1 - e^{-x_t^p} \geq (1 - 1/e) \cdot x_t^p.$$

Thus we get an $(1-1/e)$–approximation.

A summary of the main results of Section 4.3 is as follows.

Theorem 4.8 *The proportional throughput maximization problem can be approximated to within* $1 - 1/(4k)$, *where* k *is the number of times the requested page needs to be broadcast, in order to satisfy any given request. Also, for the per-user-guarantee variant of this problem, if* q^* *denotes the optimal (i.e., maximum) minimum probability of satisfaction for any request, then we can construct a random schedule in which the probability of satisfaction for each request is at least* $q_r^* - (q_r^*)^2/4 \geq (3/4) \cdot q^*$.

In the case where $k = 1$ *and where each request specifies an arbitrary set of time-slots for it to be satisfied, we can approximate the number of satisfied requests to within* $(1 - 1/e)$.

4.4 Minimizing Average Response Time

We now move on to the case where each request must be satisfied. We are mainly concerned with the average (equivalently, total) response time for the requests, and deadlines are not important. Thus, for each request (p,t), its deadline D_t^p equals infinity. The parameter k equals 1, and our goal is to schedule the broadcast of pages in a way that minimizes 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. As before, r_t^p denotes the number of requests for page p that arrive at time t. An α-*speed* broadcast schedule is one in which at most α pages are broadcast at any time instance. Let an (α,β)-algorithm stand for an algorithm that constructs an α-speed schedule whose expected cost is at most β times the cost of an optimal 1-speed solution. Gandhi *et al.* provided approximation algorithms for this problem which achieved the bounds of $(2,2), (3,1.5)$, and $(4,1)$ [21]. The latter bounds were improved to $(3,1)$ via dependent rounding, in a preliminary version of this work [22]. We now improve all these by providing a bound of $(2,1)$ using the dependent rounding technique (in the form of the generic algorithm of Section 4.1). A "per-user fairness" version also holds; see Theorem 4.14. The analysis of the algorithm is much more involved than in Sections 4.2 and 4.3.

An IP formulation for the problem is given below. The binary variable $y_{t'}^p = 1$ iff page p is broadcast at time slot t'. The binary variable $x_{tt'}^p = 1$ iff a request (p,t) is satisfied at time $t' > t$; i.e., if $y_{t'}^p = 1$ and $y_{t''}^p = 0, t < t'' < t'$. Also, T is the time of last request for any page. It is easy to check that this is a valid IP formulation.

$$
\begin{array}{lll}
\text{Minimize} \displaystyle\sum_p \sum_t \sum_{t'=t+1}^{T+n} (t'-t) \cdot r_t^p \cdot x_{tt'}^p & & \\
y_{t'}^p - x_{tt'}^p \geq 0 & \forall p, t, t' > t & \\
\displaystyle\sum_{t'=t+1}^{T+n} x_{tt'}^p \geq 1 & \forall p, t & (16) \\
\displaystyle\sum_p y_{t'}^p \leq 1 & \forall t' & \\
x_{tt'}^p \in \{0,1\} & \forall p, t, t' & \\
y_{t'}^p \in \{0,1\} & \forall p, t' &
\end{array}
$$

Algorithm. We use the generic broadcast scheduling algorithm described in Section 4.1. The fractional solution $\mathcal{S}$ in this algorithm is a 2-speed solution obtained as follows. We first solve the LP relaxation optimally. $\mathcal{S}$ is obtained by doubling the fraction of each page broadcast at each time slot by the LP solution. $\mathcal{S}$ is then rounded using the generic algorithm.

4.4.1 Analysis

Consider a request r for page p. We assume w.l.o.g. that this request arrives at time 0. Let the cost incurred by the LP solution to satisfy this request be $O_r = \sum_{i=1}^{l} x_i t_i$, where $x_i > 0$ is the fractional amount of service r receives at time t_i and $\sum_{i=1}^{l} x_i = 1$. Let $1 \leq t_1 \leq t_2 \leq \cdots \leq t_l$. We also assume w.l.o.g. that there exists a $\lambda \in \{1,\ldots,l\}$ such that $\sum_{i=1}^{\lambda} x_i = 1/2$ (otherwise we can "split" an appropriate x_i into two fractions x_λ and $x_{\lambda'}$ to achieve this). In general, the time slots $t_1,\ldots,t_l$ will span three consecutive windows in $\mathcal{S}$; since r arrives at time 0, these are the first three windows in $\mathcal{S}$. Let the random variable Y denote the fraction of service which r receives from the first window in $\mathcal{S}$. Note that Y is distributed uniformly in the interval $(0,1]$. Let $A_r(Y)$ and $B_r(Y)$ be the set of time-slots which belong to the first and second windows respectively that serve r. Define X_i to be the indicator random variable which is one

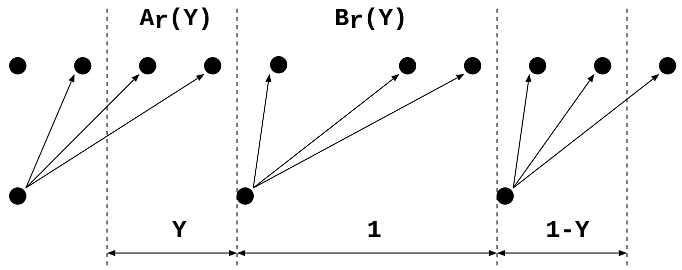

Figure 3: Variables relevant to request r: vertices on top represent time-slots, and vertices below represent windows. Values Y and 1 are the amount of service r receives from time slots in $A_r(Y)$ and $B_r(Y)$ respectively.

iff p is broadcast by the rounded solution at time t_i. Let C_r be the random variable which denotes the cost incurred by the request r in the rounded solution. Define $s_i = \sum_{j=1}^{i} x_j$, with $s_0 = 0$. The variables relevant to request r are illustrated in Figure 3.

Our Approach. Our main goal now is to prove Lemma 4.13, which shows that $\mathbf{E}[C_r] \leq O_r$; as we shall see, property (P2) will play a critical role. To prove Lemma 4.13, we bound $\mathbf{E}[C_r]$ in two different ways. First, Lemma 4.9 uses the property that whatever the value of Y is, the set of time-slots $B_r(Y)$ broadcast page p with probability 1; thus, it suffices to bound this cost. This bound alone does not suffice for our purposes; we can only show that $\mathbf{E}[C_r] \leq (4 - 2\sqrt{2})O_r$ in this manner. So, as a second approach to bound $\mathbf{E}[C_r]$, Lemma 4.10 starts with the observation that r needs to wait for a broadcast of p from $B_r(Y)$ only if the event

$$\mathcal{E} \equiv \text{ (page } p \text{ was not broadcast in } A_r(Y))$$

happens. Now, conditional on $\mathcal{E}$, the distribution of broadcasts in $B_r(Y)$ could be quite arbitrary, but (P2) still ensures that there will be a broadcast of p in $B_r(Y)$! Thus, the worst case is that conditional on $\mathcal{E}$, p is broadcast in the *last* time-slot of $B_r(Y)$. Lemma 4.10 bounds $\mathbf{E}[C_r]$ using this idea. The average of these two bounds is also an upper-bound on $\mathbf{E}[C_r]$; Lemma 4.12 then shows that in the resulting optimization problem with the x_i as variables, the maximum possible value of $\mathbf{E}[C_r]/O_r$ is 1.

Lemma 4.9 $\mathbf{E}[C_r] \leq 2\sum_{i=1}^{\lambda}(2s_{i-1} + x_i)x_i t_i + 2\sum_{i=\lambda+1}^{l}(2 - 2s_i + x_i)x_i t_i$.

Proof: Let $f(y) = \sum_{t_i \in B_r(y)} \mathbf{E}[X_i \mid (Y = y)]t_i$. Since page p will be transmitted in at least one of the slots in $B_r(Y)$ by (P2), we have $C_r \leq \sum_{t_i \in B_r(Y)} X_i t_i$ with probability 1. Hence $\mathbf{E}[C_r \mid (Y = y)] \leq f(y)$ and $\mathbf{E}[C_r] \leq \int_0^1 f(y)\,dy$. We now calculate the contribution of each t_i to this integral. There are two cases:
Case 1: $i \leq \lambda$. If $Y \leq 2s_{i-1}$, then t_i fractionally broadcasts $2x_i$ units of p in the second window. If $2s_{i-1} < Y \leq 2s_i$, then t_i fractionally broadcasts $(2s_i - Y)$ units of p in the second window. If $Y > 2s_i$, then t_i does not belong to the second window. All three scenarios are illustrated in the Figure 4.
Case 2: $\lambda < i \leq l$. If $Y \geq 2s_i - 1$, then t_i fractionally broadcasts $2x_i$ units of p in the second window. If $2s_{i-1} - 1 \leq Y < 2s_i - 1$, then t_i fractionally broadcasts $(Y + 1 - 2s_{i-1})$ units of p in the second window. Otherwise, t_i does not belong to the second window.

By property (P1) of dependent rounding, if t_i fractionally broadcasts some a units of p in the second window, then the probability that page p is broadcast at time t_i by our generic algorithm, is exactly a. Thus,

$$\mathbf{E}[C_r] \quad \leq \quad \int_0^1 f(y)\,dy$$

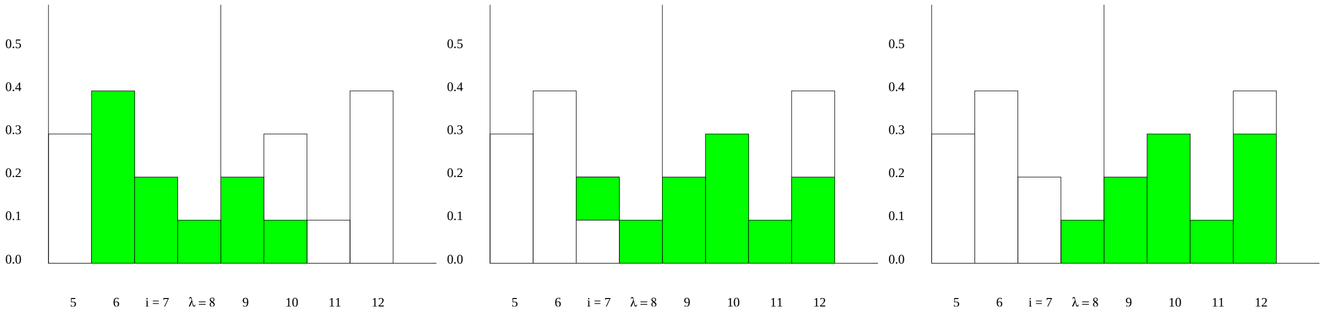

Figure 4: Contribution of slot i to the integral $\int_0^1 f(y)\,dy$: slots $5,\ldots,12$ fractionally serve request r in the LP solution. For each slot, the height of the bar indicates the fraction of page p broadcast in that slot in the 2-speed fractional solution. The request receives exactly one unit of page p from slots $5,\ldots,\lambda = 8$. The fractional broadcasts of page p within the time window $B_r(Y)$ is denoted by the shaded regions. The contribution of slot i to the integral depends on its position relative the start of the shaded region Y. (a) $Y \le 2s_{i-1}$; slot i is completely included in the shaded region. (b) $2s_{i-1} < Y \le 2s_i$; slot i is partially included in the shaded region. (c) $Y > 2s_i$; slot i is completely omitted from the shaded region.

$$= \sum_{i=1}^{\lambda}\left(2s_{i-1}2x_it_i + \int_{2s_{i-1}}^{2s_i}((2s_i-y)t_i)\,dy\right) + \sum_{i=\lambda+1}^{l}\left((2-2s_i)2x_it_i + \int_{2s_{i-1}-1}^{2s_i-1}(y+1-2s_{i-1})\,dy\right)$$

$$= 2\sum_{i=1}^{\lambda}(2s_i+x_i)x_it_i + 2\sum_{i=\lambda+1}^{l}(2-2s_i+x_i)x_it_i.$$

□

Lemma 4.10 $\mathbf{E}[C_r] \le 2\sum_{i=1}^{\lambda}(1-2s_i+x_i)x_it_i + 2\sum_{i=\lambda+1}^{l}(2-2s_i+x_i)x_it_i$.

Proof: Suppose we condition on the event "$Y = y$". Let $Z_r(y)$ be the indicator random variable which is one iff $A_r(y)$ does not serve r in the rounded solution. Let $T_r(y)$ be the random variable denoting the last time slot in $B_r(y)$. Since r is served by either the first or the second window in the rounded solution, the following holds with probability 1:

$$C_r \le \left(\sum_{t_i\in A_r(y)} X_it_i\right) + Z_r(y)T_r(y).$$

So we have

$$\mathbf{E}[C_r \mid (Y=y)] \le \left(\sum_{t_i\in A_r(y)} \mathbf{E}[X_i \mid (Y=y)]t_i\right) + \mathbf{E}[Z_r(y)]\mathbf{E}[T_r(y)].$$

Since $\mathbf{E}[Z_r(y)] = (1-y)$,

$$\mathbf{E}[C_r] \le \int_{y=0}^{y=1}\left(\sum_{t_i\in A_r(y)} \mathbf{E}[X_i \mid (Y=y)]t_i\right)dy + \int_{y=0}^{y=1}(1-y)T_r(y)\,dy.$$

We now compute the contribution of each t_i to each of the two integrals. There are two cases:
Case 1 $i \le \lambda$: t_i never contributes to the second integral since it is never the last time slot in the second window. If $2s_i < y \le 1$ then t_i contributes $2x_i$ to the first integral. If $2s_{i-1} < y \le 2s_i$ then t_i contributes

$2s_i - y$ to the first integral. If $0 < y \leq 2s_{i-1}$ it contributes nothing to either of the two integrals.
Case 2 $i \geq \lambda + 1$: t_i never contributes to the first integral since it is never part of the first window. $T_r(y) = t_i$ iff $2s_{i-1} - 1 < y \leq 2s_i - 1$.

Thus, we have

$$\mathbf{E}[C_r] = \sum_{i=1}^{\lambda} \left((1 - 2s_i)2x_i t_i + \int_{y=2s_{i-1}}^{2s_i} (2s_i - y)t_i \, dy \right) + \sum_{i=\lambda+1}^{l} \int_{y=2s_{i-1}-1}^{2s_i-1} (1 - y)t_i$$

Simplifying the above expression yields the lemma. □

Lemma 4.11 $\mathbf{E}[C_r] \leq \sum_{i=1}^{\lambda} x_i t_i + 2\sum_{i=\lambda+1}^{l} (2 - 2s_i + x_i)x_i t_i$.

Proof: The lemma follows by averaging the bounds given by Lemmas 4.9 and 4.10. □

The term "$2\sum_{i=\lambda+1}^{l}(2 - 2s_i + x_i)x_i t_i$" is next upper-bounded by Lemma 4.12. For convenience, Lemma 4.12 relabels the values $t_{\lambda+1}, t_{\lambda+2}, \ldots t_l$ as $v_1, v_2, \ldots, v_j$, and the values $x_{\lambda+1}, x_{\lambda+2}, \ldots x_l$ as $z_1, z_2, \ldots, z_j$.

Lemma 4.12 *Let values $1 \leq v_1 \leq v_2 \leq \cdots \leq v_j$ be given, and let $z_1, \ldots, z_j$ be real-valued variables. Consider the problem of maximizing the value f subject to the following constraints:*

$$\begin{aligned} f &= \frac{2\sum_{i=1}^{j}(2 - 2s_i + z_i)z_i v_i}{\sum_{i=1}^{j} z_i v_i} \\ s_i &= 1/2 + \sum_{u=1}^{i} z_u \qquad \forall i \\ \textstyle\sum_{i=1}^{j} z_i &= 1/2 \\ z_i &\geq 0 \qquad \forall i \end{aligned}$$

The maximum value of f subject to these constraints is at most 1.

Proof: The problem has a maximum, since we have a continuous objective function defined on a compact domain. Let f^* be the maximum value, and let the values of the variables in a maximizing solution be z_i^* and s_i^*. We start by making some observations about the z_i^* values, which hold w.l.o.g. Note first that those z_i^* that are zero can be eliminated from the problem. Next, if $v_i = v_{i+1}$ for some i, it is easy to see that the objective function does not change if we increment z_{i+1}^* by z_i^*, and reset z_i^* to 0; so, we can assume that $1 \leq v_1 < v_2 < \cdots < v_j$. If exactly one of the z_i^* values is non-zero, then $f^* = 1$. Hence, assume w.l.o.g. that all z_i^* values are non-zero, that there are at least two of these, and that $1 \leq v_1 < v_2 < \cdots < v_j$. Let $N = 2\sum_{i=1}^{j}(2 - 2s_i^* + z_i^*)z_i^* v_i$ and $D = \sum_{i=1}^{j} z_i v_i$, so that $f^* = N/D$. We now examine the structure of this solution by perturbing z_1^* and z_2^*. Specifically, increase and decrease the values of z_1^* and z_2^* respectively by an infinitesimal value ϵ. Clearly, the new solution is still feasible. The value of N changes by $\Delta N + O(\epsilon^2)$, where $\Delta N = 2\epsilon(2 - 2s_1)(v_1 - v_2)$; the value of D changes by $\Delta D = \epsilon(v_1 - v_2)$. Observe that for f^* to be the maximum value of the optimization problem, the following is a necessary condition: $f^* = N/D = \Delta N/\Delta D = 2(2 - 2s_1)$.

Repeating the above arguments for different z_i^* leads to the following: $f^* = 2(2-2s_1) = 2(2-2s_2) = \cdots = 2(2-2s_{j-1})$ and hence, $s_1 = s_2 = \ldots = s_{j-1}$. So, there are at most two non-zero z_i^* values, which we take to be $z_1^* = 1/2 - z$ and $z_2^* = z$. We now have

$$\begin{aligned} f^* &= \frac{2(2-2s_1+z_1^*)z_1^*v_1 + 2(2-2s_2+z_2^*)z_2^*v_2}{z_1^*v_1 + z_2^*v_2} \\ &= \frac{2(1/4 - z^2)v_1 + 2z^2v_2}{v_1/2 + z(v_2 - v_1)} \\ &= \frac{v_1/2 + 2z^2(v_2 - v_1)}{v_1/2 + z(v_2 - v_1)} \\ &\leq \max\{1, 2z\} \\ &\leq 1. \end{aligned}$$

□

Recall that O_r denotes the cost incurred by the LP solution to serve r. Then, our key lemma is:

Lemma 4.13 $\mathbf{E}[C_r] \leq O_r$.

Proof: Lemma 4.11 implies that

$$\mathbf{E}[C_r]/O_r \leq \frac{\sum_{i=1}^{\lambda} x_i t_i + 2\sum_{i=\lambda+1}^{l}(2-2s_i+x_i)x_i t_i}{\sum_{i=1}^{l} x_i t_i} \leq \max\left\{1, \frac{2\sum_{i=\lambda+1}^{l}(2-2s_i+x_i)x_i t_i}{\sum_{i=\lambda+1}^{l} x_i t_i}\right\}.$$

The lemma now follows from Lemma 4.12. □

Theorem 4.14 *Our rounding scheme yields a 2-speed 1-approximate solution. Furthermore, it leads to the following per-user guarantee. Suppose each user-request $r = (p,t)$ comes with a delay (response-time) bound D_r. Then, there is an efficient algorithm that does the following: (i) it either proves that there is no 1-speed solution that satisfies each request within its response-time bound, or (ii) it constructs a randomized 2-speed schedule such that for each request r,*

- *the expected response time of r is at most D_r, and*
- *with probability 1, the response time of r is at most $2 \cdot D_r$.*

Proof: Our algorithm constructs a 2-speed solution since the fractional value of edges incident on a vertex in U is at most 2: by property (P2) of dependent rounding, at most two edges will be incident on any vertex in U in the rounded solution. The claim that we have an 1-approximate solution in expectation, follows from Lemma 4.13 and the linearity of expectation.

As for the per-user guarantee, we proceed as follows. Given a delay bound D_r for each user r, we start by modifying our IP. We add the extra constraint

$$\forall r = (p,t), \quad \sum_{t'=t+1}^{T+n} (t'-t) \cdot r_t^p \cdot x_{tt'}^p \leq D_r.$$

We also remove the objective function, and simply ask for a feasible solution. We next solve the LP relaxation. If it has no feasible solution, then we halt, declaring that there is no 1-speed solution that satisfies each request within its response-time bound. Otherwise, suppose the LP-solver returns a feasible solution. Then, we proceed as above (doubling the y_t^p values and running the generic algorithm). Consider any request $r = (p,t)$. Lemma 4.13 shows that the expected response time of r in the randomized schedule constructed, is at most D_r. Finally, since at least one of the time-slots in $B_r(Y)$ will transmit page p with probability 1, it is easily seen that with probability 1, the response time of r is at most $2 \cdot D_r$. □

5 Scheduling on Unrelated Parallel Machines

We now consider a more traditional type of scheduling problem, from the standpoint of per-user guarantees. Given a set J of jobs, a set M of machines, and for each $j \in J$ and $i \in M$, the time $p_{ij} \in Z^+$ required to process job j on machine i, the problem considered in this section is to schedule the jobs on the machines so as to minimize the makespan. In their breakthrough work that has had several applications, Lenstra, Shmoys & Tardos present a 2-approximate solution for this problem [28]; they also show that if the problem can be approximated better than $3/2$, then $P = NP$. Since their work, the approximation threshold of the problem has been an intriguing open question; in particular, it has proven difficult to improve the approximation guarantee of 2. Continuing with our theme of probabilistic per-user guarantees, we are thus motivated to ask: if we only require a certain guaranteed probability of getting scheduled for each individual job, can we achieve a better approximation ratio for the makespan? We answer this in the affirmative, as described next.

Let OPT denote the optimal makespan of a given instance of the problem. A natural "per-user guarantee"–type question is as follows. Suppose a parameter p, $0 < p \leq 1$, is given. How small a λ can we exhibit, so that the following two properties are satisfied algorithmically?

(Per-user guarantee) Every individual job gets scheduled with probability at least p, and

(Low makespan) with probability 1, the makespan is at most $\lambda \cdot OPT$.

As mentioned above, the work of [28] exhibits the (p, λ)-pair of $(1, 2)$; as pointed out by Chandra Chekuri (personal communication), an elegant probabilistic modification of [28] yields the pair $(1/2, 1)$ [7]. Given these two pairs, it is natural to conjecture that the pair $(p, 2p)$ is achievable for all $p \in [1/2, 1]$. However, going beyond the two pairs $(1, 2)$ and $(1/2, 1)$ – in particular, achieving any pair of the form $(1/2 + \Omega(1), 2 - \Omega(1))$ – seems to require further work and possibly a different approach. By employing a variant of dependent rounding, we now prove the following:

Theorem 5.1 *For any $p \in (0, 1)$, the pair $(1 - e^{-p}, 1 + p)$ is achievable for the per-user/makespan tradeoff: i.e., we can construct a random schedule in which any given job gets scheduled with probability at least $1 - e^{-p}$, and in which the makespan is at most $(1 + p) \cdot OPT$ with probability 1. In particular, the pair $(1 - 1/e - \epsilon, 2 - \Omega(1))$ is achievable, for any positive constant ϵ.*

Note that Theorem 5.1 is incomparable with the $(1/2, 1)$ result of [7], if $p > \ln 2$. To prove Theorem 5.1, we need a weighted notion of dependent rounding, which is described next in Section 5.1; the scheduling algorithm that underlies Theorem 5.1 is then presented in Section 5.2.

5.1 Weighted dependent rounding

We will use the following notion of *weighted* dependent rounding (WDR). Fix a machine i. WDR involves a star graph G_i, where the central vertex of G_i is machine i, with leaves corresponding to the jobs assigned to i. Recall that $p_{i,j}$ is the processing time for job j, if j gets scheduled on machine i. Suppose we have a dependent rounding instance on G_i; we modify our dependent rounding as follows. Note that there is no cycle in G_i, and that a maximal simple path has length at most 2. 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); thus, (i,j) is the only floating edge incident on i at this point. We then randomly round $y_{i,j}$: round to 1 with probability $y_{i,j}$ and round to 0 with the complementary probability of $1 - y_{i,j}$.

We call the above scheme "weighted" as it is guided by the "weights" $p_{i,j}$. The scheme is easily seen to have the following two properties:

(B1) Let $\sum_j p_{i,j} \cdot y_{i,j}$ be the "load" on i. As long as G_i has at least one yet-unrounded edge, the load on i remains constant.

(B2) Fix an edge f (f is of the form (i,j) for some j). Let $Y_{f,k}$ denote the value of y_f at the end of iteration k. Then, $\forall k,\ \mathbf{E}[Y_{f,k}] = Y_{f,1}$; in other words, the expected value of y_f remains constant. A simple induction on k similar to the proof of Lemma 2.1, shows this.

5.2 Algorithm and analysis

We define a family of linear programs LP(T), one for each value of T, where T is a "guess" of the optimal makespan. Variable x_{ij} is 1 iff job j is scheduled on machine i. Let $S_T = \{(i,j) |\ p_{ij} \leq T\}$. LP$(T)$ has a variable x_{ij} for each machine i and job j. 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 LP(T) has a feasible solution if T is an upper bound on the makespan. By conducting a bisection (i.e., binary or doubling) search, we can find the smallest value of T (or a value sufficiently close to optimal) for which LP(T) is feasible. For any T for which LP(T) is feasible, the work of [28] presents a rounding scheme to construct a schedule of makespan at most $2T$; in conjunction with the above bisection-search idea, this yields a 2-approximation. We will now present a different rounding scheme. Let T^* denote the smallest value of T for which LP(T) is feasible; also let the values $\{x_{i,j}\}$ denote an optimal solution to LP(T^*).

Given a parameter $p \in (0,1)$, the algorithm proceeds as follows. It first finds the values T^* and $\{x_{i,j}\}$. We then set $x'_{i,j} = p \cdot x_{i,j}$ for each (i,j). We now have $|M|$ disjoint instances of WDR, each centered on some machine i; the initial values $y_{i,j}$ of these instance equal $x'_{i,j}$. WDR is run independently on these $|M|$ instances. Property (B1) and the fact "$x_{i,j} > 0$ implies $p_{i,j} \leq T$", together imply that with probability 1, the final load on any machine i is at most $T^*p + T^* = T^*(1+p)$. Since the $|M|$ different instances are run independently, property (B2) shows that for any job j, the probability that it does not get scheduled, is

$$\prod_i (1 - px_{i,j}) \leq e^{-\sum px_{i,j}} = e^{-p}.$$

Thus we have Theorem 5.1.

6 Discussion

We have presented a dependent rounding scheme, and seen several applications of it. We now briefly survey some related work. The deterministic pipage rounding method of Ageev & Sviridenko [1] is closely related to our work. Due to its probabilistic nature, our method can also yield per-user guarantees, which do not seem possible from deterministic approaches such as pipage rounding. Also, our application of Section 3 does not appear to follow from the work of [1]. Next, following the publication of the preliminary versions of this work [35, 22], a method of obtaining (P1) and (P2) in the general setting of unimodular matrices, has been presented in [12]. This approach does not appear to guarantee (P3); in particular, it does not seem to yield our application of Section 3. Recall from Section 3 that one main use for (P3) is in showing large-deviation bounds; in work unrelated to ours, novel randomized rounding schemes for bipartite graphs that keep many different linear objective functions close to their mean (with high probability), have been presented in [18].

The dependent rounding scheme has been employed in approximation algorithms for the partial vertex cover problem [25], and in efficient random-graph models for large social networks [15]. We view the scheme as a useful tool that will have further applications in future.

We conclude with some open problems. First, it would be interesting to know how best we can approximate the problem of minimizing average response-time, if we are constrained to use only 1-speed solutions. Some recent exciting progress has been made by Bansal et al [2, 3]. Regarding scheduling on unrelated parallel machines, it seems reasonable to conjecture that our pairs $(1-e^{-p}, 1+p)$ can be improved to those of the form $(p', 2p')$. Finally, it would be very nice to understand the approximability threshold for scheduling on unrelated parallel machines.

Acknowledgments. This work evolved over a period of time, and benefited much from discussions with several people. We thank Moses Charikar, Chandra Chekuri, Sanjeev Khanna, Seffi Naor, Iraj Saniee, Peter Winkler, and Leonid Zosin for helpful discussions. We also 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 [28], and Alan Frieze for his encouragement. Our thanks also to Julia Chuzhoy and Seffi Naor for sending us an early version of [8]. Finally, we thank the referees for their valuable suggestions.

References

[1] Ageev, A., and Sviridenko, M. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization 8*, 3 (2004), 307–328.

[2] Bansal, N., Charikar, M., Khanna, S., and Naor, J. S. Approximating the average response time in broadcast scheduling. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2005), Society for Industrial and Applied Mathematics, pp. 215–221.

[3] Bansal, N., Coppersmith, D., and Sviridenko, M. Improved approximation algorithms for broadcast scheduling, 2004. IBM Tech Report RC23468.

[4] Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., and Schieber, B. A unified approach to approximating resource allocation and scheduling. *J. ACM 48*, 5 (2001), 1069–1090.

[5] Bar-Noy, A., Guha, S., Katz, Y., Naor, J., Schieber, B., and Shachnai, H. Throughput maximization of real-time scheduling with batching. In *Proc. Thirteenth annual ACM-SIAM Symposium on Discrete Algorithms* (2002), pp. 742–751.

[6] Bartal, Y., and Muthukrishnan, S. Minimizing maximum response time in scheduling broadcasts. In *Proc. Eleventh annual ACM-SIAM Symposium on Discrete Algorithms* (2000), pp. 558–559.

[7] Chekuri, C., and Khanna, S. A PTAS for the multiple knapsack problem. In *Proc. Eleventh annual ACM-SIAM Symposium on Discrete Algorithms* (2000), pp. 213–222.

[8] Chuzhoy, J., and Naor, J. Covering problems with hard capacities. In *Proc. IEEE Symposium on Foundations of Computer Science* (2002), pp. 481–489.

[9] Cooper, C., and Frieze, A. Crawling on simple models of web graphs. *Internet Mathematics 1* (2003), 57–90.

[10] Cooper, C., and Frieze, A. A general model of web graphs. *Random Struct. Algorithms 22*, 3 (2003), 311–335.

[11] Davis, R. D., Kumaran, K., Liu, G., and Saniee, I. Spider: a simple and flexible tool for design and provisioning of protected lightpaths in optical networks. *Bell Labs Technical Journal 6* (January–June 2001).

[12] Doerr, B. Non-independent randomized rounding. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (2003), pp. 506–507.

[13] Doshi, B. T., Dravida, S., Harshavardhana, P., Hauser, O., and Wang, Y. Optical network design and restoration. *Bell Labs Technical Journal, Issue on Optical Networking 4* (January–March 1999).

[14] Erlebach, T., and Hall, A. Np-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *Journal of Scheduling 7* (may 2004), 223 – 241.

[15] Eubank, S., Kumar, V. S. A., Marathe, M. V., Srinivasan, A., and Wang, N. Structural and algorithmic aspects of massive social networks. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (2004), pp. 711–720.

[16] Even, S. *Graph Algorithms.* Computer Science Press, 1979.

[17] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1999), ACM Press, pp. 251–262.

[18] Frieze, S. A. A., and Kaplan, H. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming* (2002), 1–36.

[19] Gailis, R., and Khuller, S. Broadcast scheduling with deadlines. http://www.cs.umd.edu/users/samir/grant/renars.ps.

[20] Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G., and Srinivasan, A. An improved approximation algorithm for vertex cover with hard capacities. In *Proc. International Colloquium on Automata, Languages, and Programming* (2003), pp. 164–175.

[21] Gandhi, R., Khuller, S., Kim, Y., and Wan, Y. C. Algorithms for minimizing response time in broadcast scheduling. *Algorithmica*, 4 (January 2004), 597–608.

[22] Gandhi, R., Khuller, S., Parthasarathy, S., and Srinivasan, A. Dependent rounding in bipartite graphs. In *Proc. IEEE Symposium on Foundations of Computer Science* (2002), pp. 323–332.

[23] Graham, F. C., and Lu, L. Connected components in random graphs with given degree sequences. *Annals of Combinatorics 6* (2002), 125–145.

[24] Guha, S., Hassin, R., Khuller, S., and Or, E. Capacitated vertex covering. *J. of Algorithms* (2003), 257–270.

[25] Halperin, E., and Srinivasan, A. Improved approximation algorithms for the partial vertex cover problem. In *Proc. Fifth International Workshop on Approximation Algorithms for Combinatorial Optimization Problems* (2002), pp. 161–174.

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

[27] Kleinberg, J. Approximation algorithms for disjoint paths problems. *Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT* (1996).

[28] Lenstra, J. K., Shmoys, D. B., and Tardos, E. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming 46* (1990), 259–271.

[29] Logothetis, D., and Trivedi, K. The effect of detection and restoration times for error recovery in communication networks. *Journal of Network and Systems Management 5* (1997), 173–195.

[30] Luby, M. LT codes. In *Proc. IEEE Symposium on Foundations of Computer Science* (2002), pp. 271–282.

[31] Panconesi, A., and Srinivasan, A. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. Comput. 26* (1997), 350–368.

[32] Raghavan, P., and Thompson, C. D. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica 7* (1987), 365–374.

[33] SDH frequently asked questions. http://www1.biz.biglobe.ne.jp/~worldnet/faq/sdh.html.

[34] Shmoys, D. B., and Tardos, E. An approximation algorithm for the generalized assignment problem. *Mathematical Programming* (1993), 461–474.

[35] Srinivasan, A. Distributions on level-sets with applications to approximation algorithms. In *Proc. IEEE Symposium on Foundations of Computer Science* (2001), pp. 588–597.

[36] Stern, T. E., and Bala, K. *Multiwavelength optical networks: a layered approach.* Prentice Hall, 1999.

[37] Trotter, W. T., and Winkler, P. Ramsey theory and sequences of random variables. *Combinatorics, Probability and Computing 7* (1998), 221–238.

[38] Yook, S., Jeong, H., and Barabasi, A. Modeling the internet's large-scale topology. *Proceedings of the National Academy of Sciences* (2002), 13382–13386.

Appendix

A Capacitated Vertex Cover

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, ..., n\}$ and edge set E. Recall that an *orientation* of the edges of G is the process of replacing each edge $\{i, j\}$ by one of the two directed edges (i, j) or (j, i). In the problem at hand, each vertex v comes with three non-negative values: a *weight* w_v, a *capacity* k_v, and a *maximum number of allowed-copies* b_v. The values k_v and b_v will be integers. A *capacitated vertex cover* is a collection of values $x_v \in \{0, 1, \ldots, b_v\}$ for each v and 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. So, the problem is to choose an integral number of copies (bounded by b_v) of each vertex v and to assign each edge to one of its two end-points, such that each vertex has sufficient capacity for all of its assigned edges. The *weight* of the cover is $\sum_{v\in V} x_v w_v$. 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, while 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.

This problem has received a good deal of attention recently; in particular, the case where all the b_v are infinite is quite different from the case where some of the b_v could be finite. Let us call these the "unbounded" and "bounded" case respectively. The unbounded case has been studied by Guha, Hassin, Khuller & Or [24]: they present a 4-approximation using LP rounding, and a primal-dual 2-approximation. The bounded case has been studied by [8], who showed that this case is: (i) as hard to approximate as set cover if the vertex-weights w_v are arbitrary, and (ii) can be approximated to within 3 if the vertex-weights w_v are all 1. (To our knowledge, this is one of the very few problems demonstrating such a difference between the weighted and unweighted scenarios.) This result (ii) has been improved by [20] to a 2-approximation.

Thus, we ask: if we do have general vertex-weights w_v and still desire a 2-approximation for the objective function, how much of a violation of the "$x_v \leq b_v$" constraints is sufficient? We show that we can obtain a 2-approximation wherein $x_v \leq 2 \cdot b_v$ for all v. (Our algorithm is simple to describe, but somewhat nontrivial to analyze; it involves a random thresholding followed by dependent rounding.) Even for the unbounded case, we obtain an improvement over [24] as follows. Analogous to the generalization of [28] that is investigated in [34] by considering assignment-costs, we consider a generalization in Section A.2, and present a 2-approximation for this generalization also.

A.1 IP formulation and rounding scheme

We denote by $\delta(v)$ the edges in E which are incident to v. An IP formulation of the minimum capacitated vertex cover problem is to minimize $\sum_v w_v x_v$ subject to:

$$\begin{aligned} y_{eu} + y_{ev} &\geq 1 \quad \forall e = \{u,v\} \in E \\ k_v x_v - \sum_{e\in\delta(v)} y_{ev} &\geq 0 \quad \forall v \in V \\ x_v &\geq y_{ev} \quad \forall v \in e \in E \\ y_{ev} &\in \{0,1\} \quad \forall v \in e \in E \\ x_v &\in \{0,1,\ldots,b_v\} \quad \forall v \in V \end{aligned} \tag{17}$$

In this formulation, $y_{ev} = 1$ denotes that the edge $e \in E$ is covered by vertex v. Clearly, the values (x,y) 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 LP relaxation: this relaxation lets each y_{ev} to be a real in $[0,1]$, and each x_v to be a real in $[0,b_v]$. We present our algorithm next. In the algorithm, we first do a (random) thresholding: we choose a "random" α, and for all pairs (e,v) such that $y_{ev} \geq \alpha$, we assign e to v. Informally, this takes care of pairs (e,v) where y_{ev} was "large", to start with. The remaining unassigned edges are then handled by dependent rounding.

Algorithm Threshold-and-Round:

1. Solve the LP relaxation of the above IP to obtain an optimal fractional solution (x,y). We will assume w.l.o.g. that for all $e = (u,v)$, $y_{eu} + y_{ev} = 1$.

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 $y^*_{ev} = 1$. (Note that with probability one, at most one of these two assignments will happen for an edge e in this step, since $y_{eu} + y_{ev} = 1$.)

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}. As mentioned above, we have w.l.o.g. that $y_{eu} + y_{ev} = 1$. We now use dependent rounding to round the y_{eu} values to integers y^*_{eu}. We define $x^*_u = \left\lceil \frac{\sum_{e \in E(u)} y^*_{eu}}{k_u} \right\rceil$. 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.

This completes the description of the algorithm. As a first step in our analysis, we estimate the probability of an edge getting assigned to a particular end-point:

Lemma A.1 *For any edge e and for each endpoint v of e, $\mathbf{E}[y^*_{ev}] \leq (4 - 2\sqrt{2})y_{ev}$.*

Proof: If $y_{ev} < 1/2$, then the only way for e to get assigned to v is in the dependent rounding step; since dependent rounding will do such an assignment with probability y_{ev}, we get $\mathbf{E}[y^*_{ev}] \leq y_{ev} \leq (4 - 2\sqrt{2})y_{ev}$.

Next suppose $y_{ev} \geq 1/2$. We have

$$\begin{aligned}
\mathbf{E}[y^*_{ev}] &= \mathbf{E}[y^*_{ev} \mid \alpha \leq y_{ev}] \cdot \Pr[\alpha \leq y_{ev}] + \mathbf{E}[y^*_{ev} \mid \alpha > y_{ev}] \cdot \Pr[\alpha > y_{ev}] \\
&= 2(y_{ev} - \frac{1}{2}) + 2y_{ev}(1 - y_{ev}) \\
&= (4 - 2\sqrt{2})y_{ev} - (\sqrt{2}y_{ev} - 1)^2 \\
&\leq (4 - 2\sqrt{2})y_{ev}.
\end{aligned}$$

□

Our two main goals are to show that no x^*_v is much more than b_v, and to bound the weight of the cover constructed (i.e., to bound the approximation ratio). Recall that the results of [8] show that if we desire any constant-factor approximation, some blowup in the values b_v is necessary. We start by showing that $x^*_v \leq 2b_v$ with probability 1:

Lemma A.2 *For each vertex v, $x^*_v \leq 2\lceil x_v \rceil$ with probability 1. In particular, $\Pr[x^*_v \leq 2b_v] = 1$.*

Proof: We define some quantities relative to v. Let L_v denote the set of edges e incident on v such that $y_{ev} \geq 1/2$, and let S_v denote the set of remaining edges incident on v. (L_v and S_v stand for "large" and "small" respectively.) In the worst case, all edges in L_v can get assigned to v in Step 3, and all edges in S_v will participate in the dependent rounding. It is sufficient to show that in such a case, $2\lceil x_v \rceil$ copies of v suffice. Let $\ell_v = |L_v|$, and $s_v = \sum_{e \in S_v} y_{ev}$. Note that in this worst case, the number of edges assigned to v is $\ell_v + \lceil s_v \rceil$. We aim to show that

$$\ell_v + \lceil s_v \rceil \leq 2k_v \lceil x_v \rceil. \tag{18}$$

We have by (17) that $\ell_v/2 + s_v \leq k_v x_v$, i.e.,

$$\ell_v + 2s_v \leq 2k_v \lceil x_v \rceil. \tag{19}$$

Our desired bound (18) holds as a consequence of (19), as follows. If $s_v \geq 1/2$, then we are done since $\lceil s_v \rceil \leq 2s_v$. If $0 < s_v < 1/2$, then since ℓ_v and $2k_v\lceil x_v \rceil$ are integers, bound (19) implies that $\ell_v \leq 2k_v\lceil x_v \rceil - 1$, which establishes (18). $\square$

Fix any vertex v. Our next main goal is to prove that $\mathbf{E}[x_v^*] \leq 2x_v$, which will then establish an expected approximation ratio of 2. We now show that $\mathbf{E}[x_v^*] \leq 2x_v$ through Lemmas A.3, A.4 and A.5, each of which works for a certain set of values of x_v. We start with the first case, where x_v is either quite small or quite large:

Lemma A.3 *If $x_v < 1/2$ or if $x_v \geq \frac{1+\sqrt{2}}{2}$, then $\mathbf{E}[x_v^*] \leq 2x_v$.*

Proof: First suppose $x_v < 1/2$. Consider any edge $e = (u, v)$ incident on v. Since $y_{ev} \leq x_v < 1/2$, e cannot get assigned to v in Step 3. Furthermore, if $\alpha \leq y_{eu}$, then e will get assigned to u in Step 3. Note that $y_{eu} = 1 - y_{ev} \geq 1 - x_v$. So, if $\alpha \leq 1 - x_v$, then *all* edges incident on v will get assigned to their other end-points in Step 3. Thus, a necessary condition for some edges to get assigned to v is that $\alpha > 1 - x_v$, which happens with probability $2x_v$. Now, Lemma A.2 shows that x_v^* takes values only in $\{0, 1\}$. Thus, $\mathbf{E}[x_v^*] \leq (2x_v) \cdot 1 = 2x_v$.

Next suppose $x_v \geq \frac{1+\sqrt{2}}{2}$. Let the random variable R_v denote the number of edges assigned to v. Lemma A.1 and the linearity of expectation show that

$$\mathbf{E}[R_v] \leq \sum_{e=(v,u)} (4 - 2\sqrt{2}) y_{ev} \leq (4 - 2\sqrt{2}) k_v x_v,$$

where the second inequality is a consequence of (17). Therefore,

$$\mathbf{E}[x_v^*] = \mathbf{E}\left[\left\lceil \frac{R_v}{k_v} \right\rceil\right] \leq \frac{\mathbf{E}[R_v]}{k_v} + 1 \leq (4 - 2\sqrt{2}) x_v + 1 \leq 2x_v,$$

where the last inequality holds since $x_v \geq \frac{1+\sqrt{2}}{2}$. $\square$

Lemma A.4 *If $\frac{1}{2} \leq x_v < 1$ then $\mathbf{E}[x_v^*] \leq 2x_v$.*

Proof: Case (1): $\alpha > x_v$. No edge e can have $y_{ev} \geq \alpha$. So, if an edge gets assigned to v, this has to happen in the dependent rounding step. Let $a_v + f_v = \sum_{e \in \delta(v)} y_{ev}$, where $0 \leq f_v < 1$ and a_v is an integer. The maximum number of edges assigned to v is $a_v + 1$. Note from (17) that $a_v + f_v \leq k_v x_v < k_v$. So, $a_v \leq k - 1$, and at most one copy of vertex v in needed. Thus, $x_v^* \leq 1$ in this case.

Case (2): $\alpha \leq x_v$. Here, we get from Lemma A.2 that $x_v^* \leq 2$. So,

$$\begin{aligned} \mathbf{E}[x_v^*] &= \mathbf{E}[x_v^* \mid \alpha > x_v] \cdot \Pr[\alpha > x_v] + \mathbf{E}[x_v^* \mid \alpha \leq x_v] \cdot \Pr[\alpha \leq x_v] \\ &\leq 2(1 - x_v) + 2(x_v - \frac{1}{2})2 \leq 2x_v. \end{aligned}$$

$\square$

The remaining range of values for x_v is covered by:

Lemma A.5 *If $1 \leq x_v \leq 1.5$, then $\mathbf{E}[x_v^*] \leq 2x_v$.*

Proof: Case (1): $\alpha \leq \frac{x_v}{2}$. In this case, we use the bound $x_v^* \leq 3$ which follows from Lemma A.2.

Case (2): $\alpha > \frac{x_v}{2}$. Consider any such fixed value for α. Let L_v denote the set of edges e incident on v such that $y_{ev} \geq \alpha$, and let S_v denote the set of remaining edges incident on v. All edges in L_v will get assigned to v in Step 3, and in the worst case, all edges in S_v will participate in the dependent rounding. We will now prove that in such a case, 2 copies of v suffice. Let $\ell_v = |L_v|$, and write $\sum_{e \in S_v} y_{ev}$ as $a_v + f_v$, where a_v is an integer and $0 \leq f_v < 1$. Note that in this worst case,

$$x_v^* \leq \lceil \frac{\ell_v + a_v + 1}{k_v} \rceil. \tag{20}$$

Since each $e \in L_v$ has $y_{ev} \geq \alpha > \frac{x_v}{2}$, bound (17) shows that $\frac{x_v}{2}\ell_v + a_v + f_v < k_v x_v$. This implies that $\ell_v + \frac{2}{x_v}(a_v + f_v) < 2k_v$; so, $\ell_v + a_v < 2k_v$. Since ℓ_v, a_v and k_v are integers, we get from (20) that $x_v^* \leq 2$.

Combining the above two cases,

$$\begin{aligned} \mathbf{E}[x_v^*] &= \mathbf{E}[x_v^* \mid \alpha \leq \frac{x_v}{2}] \cdot \Pr[\alpha \leq \frac{x_v}{2}] + \mathbf{E}[x_v^* \mid \alpha > \frac{x_v}{2}] \cdot \Pr[\alpha > \frac{x_v}{2}] \\ &\leq 2(\frac{x_v}{2} - \frac{1}{2})3 + 2(1 - \frac{x_v}{2})2 \\ &= 1 + x_v \leq 2x_v. \end{aligned}$$

□

Lemmas A.3, A.4 and A.5 collectively prove that that for any value of x_v, $\mathbf{E}[x_v^*] \leq 2x_v$. This fact, along with the linearity of expectation and Lemma A.2, yields our result for capacitated vertex cover:

Theorem A.6 *The expected cost of the integral solution produced by Algorithm Threshold-and-Round is at most $2 \cdot OPT_{LP}$, where OPT_{LP} is the cost of the minimum fractional solution to the relaxation. Furthermore, the integral solution has $x_v^* \leq 2b_v$ for all vertices v, with probability one.*

A.2 Assignment Costs

We can generalize the 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. Our rounding algorithm remains the same. 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. Now, Lemma A.1 shows that the expected assignment cost is at most $(4 - 2\sqrt{2})OPT_{LP}^e$. Combining this with Theorem A.6 where we bound the expected cost of the weighted sum of the chosen vertices by $2 \cdot OPT_{LP}^v$, we 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 costs.

Theorem A.7 *Algorithm Threshold-and-Round finds a solution (x^*, y^*) such that the expected weight of vertices is at most $2 \cdot OPT_{LP}^v$ 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 $2 \cdot OPT_{LP}$).*