Dependent Randomized Rounding: The Bipartite Case*

Benjamin Doerr[†‡] Marvin Künnemann[‡] Magnus Wahlström[†]

Abstract

We analyze the two existing algorithms to generate dependent randomized roundings for the bipartite edge weight rounding problem together with several newly proposed variants of these algorithms.

For both the edge-based approach of Gandhi, Khuller, Parthasarathy, Srinivasan (FOCS 2002) and the bit-wise approach of Doerr (STACS 2006) we give a simple derandomization (guaranteeing the same rounding errors as the randomized versions achieve with positive probability).

An experimental investigation on different types of random instances show that, contrary to the randomized rounding problem with disjoint cardinality constraints, the bit-wise approach is faster than the edge-based one, while the latter still achieves the best rounding errors. We propose a hybrid approach that, in terms of running time, combines advantages of the two previous approaches; in terms of rounding errors it seems a fair compromise. In all cases, the derandomized versions yield much better rounding errors than the randomized ones.

We also test how the algorithms compare when used to solve different broadcast scheduling problems (as suggested by Gandhi et al.). Since this needs more random decisions than just in the rounding process, we need to partially re-prove previous results and simplify the corresponding algorithms to finally derive a derandomized version. Again, the derandomized versions give significantly better approximations than the randomized versions.

We tested the algorithms on data taken from the Wikipedia access log. For the maximum throughput version of the problem, the derandomized algorithms compute solutions that are very close to the optimum of the linear relaxation. For the minimum average delay version, Gandhi et al. gave a $(2,1)$-bicriteria algorithm, i.e., an algorithm which produces a 2-speed schedule with an average delay which on expectation is no worse than that of the 1-speed optimum. For this problem variant, while the performance guarantee of the algorithms certainly holds, we find that a simple greedy heuristic generally produces superior solutions.

1 Introduction

Randomized rounding, introduced by Raghavan and Thompson [20, 21] is one of the key primitives in randomized algorithmics, see also [17]. Given a number $x \in [0,1]$ (which could be the fractional part of an arbitrary real number), we call a binary random variable y a *randomized rounding of* x, if $\Pr(y = 1) = x$. Algorithmically speaking, we round x up to one with probability equal to x.

This simple idea becomes very strong if we have many numbers to be rounded. Let $x \in [0,1]^n$ and let y be an independent randomized rounding of x (that is, y_i is a randomized rounding of x_i, mutually independent for all i). Let $a_1, \dots, a_n \in [0,1]$ be weights. Then not only does this way of rounding keep the expectation of weighted sums of the variables unchanged, that is $E(\sum_i a_i y_i) = \sum_i a_i x_i$, but in addition so-called Chernoff bounds show that with high probability the actual value of this weighted sum is close to its expectation. For example, for all $\lambda > 0$ we have

$$\Pr(|\sum_i a_i y_i - \sum_i a_i x_i| > \lambda\sqrt{n}) < e^{-2\lambda^2}.$$

1.1 Dependent Randomized Rounding

The last ten years saw a number of randomized rounding results where the variables were not rounded independently (e.g. [4, 8, 9, 11, 15, 16, 22]). Such ideas allow to better adapt the rounding procedure to the particular problem one is interested in. Of course, some care has to be taken if Chernoff-type large deviations bounds are needed, as they often assume independence of the variables.

The first step to use such ideas is to find suitable methods to generate randomized roundings that satisfy certain dependencies, called hard constraints. Here the ground-breaking paper of Srinivasan [22] provides a method to generate randomized roundings that satisfy a global *cardinality constraint*, that is, the requirement that the sum of all numbers remains unchanged (assuming that it is integral in the first place). Naturally,

*Supported by the German Science Foundation (DFG) via its priority program (SPP) 1307 "Algorithm Engineering", grant DO 749/4-2.

[†]Max-Planck-Institut für Informatik, Saarbrücken, Germany.

[‡]Universität des Saarlandes, Saarbrücken, Germany.

this easily extends to disjoint cardinality constraints. This was extended to the bipartite edge weight rounding problem, described below, by Gandhi et al. [14, 15]. A substantially different rounding method for both settings, was given in [9, 10].

Theoretical analyses show similar results for both approaches in many settings. This led the authors start analyzing these rounding methods from an algorithm engineering view-point [12, 13], starting with disjoint cardinality constraints. The experimental results show quite clearly that generally the rounding errors in soft constraints, that is expressions like $|\sum_i a_i y_i - \sum_i a_i x_i|$, are superior for the derandomized versions of the algorithms, and in particular better for the approach of Srinivasan [22], as derandomized by the authors, than for the approach of Doerr [10]. This holds both in terms of rounding errors and running time.

1.2 Our Work In this paper, we continue this line of research and regard randomized rounding for the bipartite edge weight problem. In this problem, defined properly in the next section, we are asked to round the weights of the edges of a bipartite graph, while maintaining the total weight incident to each vertex. We thus have two sets of disjoint cardinality constraints, which means that there is interaction between the constraints. Many problems of assignment-type lead to constraints like this, e.g., the broadcast scheduling problems regarded in Gandhi et al. [15].

What makes an experimental investigation interesting for the bipartite edge weight rounding problem, is the fact that the best known theoretical analyses of the two rounding algorithms give quite different running times. This stems from a pessimistic assumption in the analysis of the algorithm of Gandhi et al. [15], namely that the cycles or paths along which the weight is changed can have length up to n. So it is not clear if the resulting weaker running time guarantee for the algorithm of [15] really leads to worse running times in practice.

To get a better understanding on the general working principles, we extensively test them on random instances of different types. We also propose a hybridization of the two previous algorithms. To have an indication on how they behave on 'real' instances, we implemented the broadcast scheduling algorithms of Gandhi et al. [15], derandomized them, and tested them on data stemming from the Wikipedia access log.

In a nut-shell, our findings are as follows. For the random instances, we observe that the superior running times suggested by the theoretical estimates for the bitwise approach are real. The difference of the observed running times, however, is significantly smaller than the corresponding difference of the running time guarantees. The hybrid approach manages to combine strengths of the two predecessors and is fastest on a broad range of instances. By not only tracking the running times, but also structural information like how often edges have to be visited by the algorithms, we gain insight in why these effects occur.

Concerning rounding errors in the random instances, the approach of Gandhi et al. [15] remains the strongest, similarly as previously observed also for disjoint cardinality constraints [12].

For the broadcasting setting, as Gandhi et al. [15], we consider two different objective functions, namely maximum throughput and minimum average delay. For both cases, we present complete derandomizations, that is, we derandomize both the rounding procedure and the other random decisions made in the algorithms. For the minimum average delay case, this requires giving a new proof of the approximation properties of the algorithm by Gandhi et al. For both cases, we also find that the rounding procedures, and in particular the derandomized versions, perform well compared to their guarantees. However, while the maximum throughput algorithm finds solutions very close to the optimum of the LP relaxation, the algorithm for minimum average delay is beaten clearly in our tests by a simple greedy heuristic.

Throughout for the broadcasting applications, we observe the new hybrid algorithm to be the fastest as well as producing the smallest errors.

2 Algorithms for the Bipartite Edge Weight Rounding Problem

Let $G = (V_1 \dot\cup V_2, E)$ be a bipartite (undirected) graph with edge weights $x := (x_e)_{e\in E}$. Since we are only interested in the rounding aspect, we shall assume that all weights are in $[0, 1]$. Let $V := V_1 \cup V_2$ and $n := |V|$. For $v \in V$, let $E_v := \{e \in E \mid v \in e\}$. We call $d_x(v) := \sum_{e\in E_v} x_e$ the *fractional degree* of v.

The *bipartite edge weight rounding problem* asks for a rounding (y_e) of (x_e) that changes the fractional degrees by less than one. In other words, we have $|y_e - x_e| < 1$ for all $e \in E$ and $|d_y(v) - d_x(v)| < 1$ for all $v \in V$. Solving such problems can be used to transform a fractional solution of a problem with assignment-type constraints into an integer one, as shown by Ageev and Sviridenko [1, 2, 3]. Examples include the maximum coverage problem, the maximum k-cut problem with given sizes of the parts, and several scheduling problems ([3, 15]).

Pipage rounding The key idea to solve such problems was given by Ageev and Sviridenko [3]. Denote by $E' :=$

$\{e \in E \mid x_e \notin \mathbb{Z}\}$ the set of all edges having non-integral weight. Let $G' = (V, E')$. Suppose that G' contains a simple cycle $C = (e_1, e_2, ..., e_{2k})$, here described via its sequence of edges. Then there is an $\varepsilon \in (-1, 1)$ such that alternatingly adding and subtracting ε to/from the edge weights in C keeps all edge weights in $[0, 1]$, but moves at least one edge weight to 0 or 1. Note that this does not change the fractional degrees. We may do the same for a maximal simple path. In this case, the fractional degrees of the end-vertices may change, however, they never go beyond their floors and ceilings. Consequently, repeating this *pipage rounding* step up to $m := |E|$ times, we end up with integral edge weights.

2.1 The Edge-based Approach Occasionally, in addition we aim at keeping further rounding errors small. By this we mean that for a given linear expression $f(x) = \sum_i a_x x_i$, we want that $|f(y) - f(x)|$ is small. It is easy to see that reasonable bounds for such expressions can only be guaranteed if $f(\cdot)$ only depends on variables with index in some E_v, $v \in V$. To simplify the notation, we shall always assume in this paper that the coefficients a_i of such an f are 0 and 1. This allows to identify f simply with a subset of E_v for some $v \in V$.

The first to show that solutions to the bipartite edge weight rounding problem exist keeping such additional rounding errors small, were Gandhi et al. [14]. What they showed is that there is a way to randomly round the edge weights (by this we always mean that $\Pr(y_e = 1) = x_e$ holds for all $e \in E$) in such a way that the variables $(x_e)_{e \in E_v}$ are negatively correlated for all $v \in V$. By a result of Panconesi and Srinivasan [19], this allows the use of classical Chernoff bounds on such sets of random variables.

The algorithm generating such randomized roundings is in fact quite easy—it is a randomized version of the deterministic pipage rounding algorithm given above. Note that for each cycle or path in G' there are two possible values $\varepsilon_1, \varepsilon_2$ that could be used. They have opposite sign, so w.l.o.g. let $\varepsilon_1 < 0$ and $\varepsilon_2 > 0$. Now choosing $\varepsilon = \varepsilon_1$ with probability $\varepsilon_2/(-\varepsilon_1 + \varepsilon_2)$ and $\varepsilon = \varepsilon_2$ with probability $-\varepsilon_1/(-\varepsilon_1 + \varepsilon_2)$ and performing the current pipage iteration with this ε, gives the required randomized roundings (as the involved proof in [15] shows). We shall call this the *edge-based approach* to generate randomized roundings for the bipartite edge weight rounding problem, since in each iteration (typically) exactly one edge becomes integral.

2.2 The Bit-wise Approach A different approach to solve the bipartite edge weight rounding problem was presented in [10]. The crucial observation is that if all edge weights in the previous approach were equal to $1/2$, then a single pipage iteration would make all edge weights on the cycle or path integral. This can be exploited as follows.

Assume that all edge weights have a finite binary expansion of length ℓ, that is, for all $e \in E$ we have $2^\ell x_e \in \mathbb{Z}$. Write $x = x' + 2^{-\ell+1} x''$ with all x'_e having binary length $\ell - 1$ and $x'' \in \{0, 1/2\}^E$. Compute a randomized rounding y'' of x'' as above and set $x_e := x'_e + 2^{-\ell+1} y''_e$, which has binary length $\ell - 1$. Repeating this procedure ℓ times altogether also gives a randomized rounding for the bipartite edge weight rounding problem with negative correlation in the sets $E_v, v \in V$. In return for the slightly technical detour through the binary expansion, this approach has a (typically superior) running time of $O(m\ell)$ (the edge-based approach has a running time of $O(m\ell_c)$, where ℓ_c is the average cycle length, but in the absence of a bound on ℓ_c one has to assume $O(mn)$).

2.3 A Hybrid Approach We now propose a new algorithm for computing randomized roundings in the bipartite edge weight setting, which is a hybrid of the two previous ones. One iteration consists of the following. We now let G' be the graph consisting of all edges that have the least significant bit (among all edge weights) equal one. We compute a cycle or path as previously in G'. We choose the random ε as in the edge-based approach.

This can be seen as performing the edge-based approach, but only taking into account edges with the least significant bit equal to one, or using the bit-wise approach, but choosing the ε taking into account the true edge weights and not only their last bit.

It is easily seen that this approach combines advantages of both previous ones. In particular, in each iteration both one edge weight becomes integral and all edge weights being part of the rounding step have their binary length reduced. Consequently, both running time guarantees of $O(mn)$ and $O(m\ell)$ are valid.

Since the proof of Gandhi et al. [15] does not specify how to choose the cycle or path and since each cycle chosen by the above algorithm also is a cycle and treated identically in the setting of Gandhi et al. [15], one can easily see that this hybrid rounding algorithm computes randomized roundings for the bipartite edge weight rounding problem *if in each rounding iteration a cycle and not a path is used.*

The restriction to only cycles can be solved via a simple preprocessing. To this end, let us first note that all three rounding algorithms presented so far will always find cycles if the fractional degrees are all integral (simply because in this case G' never has vertices of odd degree). It is also straight-forward to see that by adding

two additional vertices and at most $2n + 1$ additional edges with appropriately chosen edge weights, we can make our bipartite Graph having only integral fractional degrees (we add one vertex to each bipartition class, an edge from each of these to each vertex in the opposite class having weight such that the 'old' vertex obtains an integral fractional degree, and finally an edge connecting the two new vertices and giving both (simultaneously, since their fractional degrees have equal fractional part) the desired fractional degree).

Note that this preprocessing can be done in time $O(m)$, so it does not affect the order of the total running time. We thus have the following.

THEOREM 2.1. *The hybrid rounding algorithm computes randomized roundings for the bipartite edge weight rounding problem in time $O(m \min\{n, \ell\})$ satisfying the same large deviation bounds as the edge-based and bit-wise approach.*

While typically this is not a gain in the order of magnitude of the running time, the hope is of course that the new algorithm proves to be faster in experiments.

2.4 Derandomization All randomized rounding algorithms proposed here can be derandomized using the classical pessimistic estimators. These are easily applied to the $\{0, 1/2\}$-case of the bit-wise approach. However, the rounding errors obtained via this derandomization are worse compared to classical randomized rounding, because the errors from the different iterations add up in a geometric series.

In [12] we showed that also the rounding approach of Srinivasan [22], a predecessor of the edge-based approach for disjoint cardinality constraints, can be derandomized via the classical pessimistic estimators. This argument (without proof given here) easily extends to the edge-based approach for the bipartite edge weight rounding problem. We thus obtain a more direct derandomization of the bit-wise approach, which in addition admits better bounds for the rounding errors.

THEOREM 2.2. *The edge-based rounding algorithm can be derandomized using the classical pessimistic estimators in an analogous way as done for the setting of disjoint constraints in [12]. The bit-wise approach can be derandomized via the same pessimistic estimators. Both derandomizations lead to rounding errors in the sets E_v as guaranteed by Chernoff bounds known for independent randomized rounding.*

By the second statement, we removed the previously inferior constant factors for the rounding errors in the bit-wise approach. In the following, when talking about derandomizations of the bit-wise approach, we shall always mean this improved version. Since the hybrid approach is a particular implementation of the edge-based approach, Theorem 2.2 immediately yields a derandomization for the hybrid approach as well.

3 Experiments on Random Instances

As a first set of test instances, we chose different kinds of random instances. As underlying graph, we took the following random bipartite graphs. In all cases, let us denote the graph by $G = (V_1 \dot\cup V_2, E)$ with $|V_1| = |V_2|$ and E being the random edge set. Let $V = V_1 \cup V_2$ and $n := |V|$.

Random bipartite graphs with m edges: In this kind, let E be a random subset of exactly m edges (subject to the bipartiteness requirement). This is a bipartite analogue of the classical random graph model $G_{n,m}$. In this random graph, naturally, each vertex has an expected degree of $d = 2m/n$. For d less than $\ln(N)$, the deviations of the actual degrees from the expected ones become quite significant—in particular, we typically see isolated vertices.

Regular random bipartite graphs: For this reason, if we aim at graphs with small average degree, it is more appropriate to regard random regular bipartite graphs. Since these are difficult to generate uniformly at random, we regard the following type of regular random bipartite graphs. Let d be the degree we aim at. Starting with $E = \emptyset$, we repeat d times the following procedure. We repeat generating random matchings from V_1 to V_2 until we sample one that contains no edge already present in E and add this matching to E. Clearly, the resulting graph is regular with degree d.

Almost regular random bipartite graphs: Already for d being a large constant, the previous construction is too time-consuming. We therefore resort to the variant where we just take the union of d random matchings. By the birthday paradox, for $d = o(\sqrt{n})$, the average degree of G still is $d(1 - o(1))$. For $d \geq 3$, both variants of (almost) regular random bipartite graphs are connected with high probability.

As edge weights to be rounded we chose numbers $(x_e)_{e \in E}$ uniformly at random from $[0, 1]$. Given a rounding $(y_e)_{e \in E}$ of (x_e), we are interested in the rounding errors $|\sum_{e \in E_i}(y_e - x_e)|$ for some sets $E_1, E_2, \ldots$, each of which is a subset of the edges $E_v := \{e \in E \mid v \in E\}$ incident with a single vertex v. We determined the sets $E_1, E_2, \ldots$ by choosing for each $v \in V$ a fixed number of 10 random subsets of E_v.

3.1 Running Times As discussed in the introduction, one question of particular interest are the running

Algorithm	Error	Pipage rounding iterations	Edge visits	Path/Cycle lengths	Time (in s)
Edge-based	1.85	2,500	54,235	21.69	14.57
Bit-wise	2.01	7,341	35,642	4.86	10.21
Hybrid	1.86	2,487	22,316	8.97	7.95

(a) 5-regular random bipartite graphs on 1000 vertices. The randomized algorithms get rounding error 2.79–2.83.

Algorithm	Error	Pipage rounding iterations	Edge visits	Path/Cycle length	Time (in s)
Edge-based	3.13	9,812	323,354	32.96	108.52
Bit-wise	3.87	10,212	140,948	13.80	66.66
Hybrid	3.68	5,778	110,171	19.07	64.82

(b) Almost 20-regular random bipartite graphs on 1000 vertices (9,812 edges on average). The randomized algorithms get rounding error 5.27–5.37.

Algorithm	Error	Pipage rounding iterations	Edge visits	Path/Cycle length	Time (in s)
Edge-based	4.38	20,000	399,892	19.99	56.29
Bit-wise	6.09	31,008	286,581	9.24	45.66
Hybrid	5.43	14,827	161,257	10.88	33.35

(c) Random bipartite graphs with 20,000 edges on 400 vertices. The randomized algorithms get rounding error 10.81–11.00.

Table 1: Performance of the derandomized algorithms. The randomized algorithms have very similar structural data, running times 25–200 times lower, and rounding errors as shown in the captions.

times of the different algorithms, because it seems likely that the pessimistic estimates done in the theoretical analysis of the approach of [15] are not observed in an actual run.

We will present our data for the derandomized variants of the algorithms only. Throughout our experiments, the structural observations (number of path/cycles, number of edge visits, average cycle/path length) for the randomized variants are very similar to the corresponding values in the derandomized setting, while the running times are smaller by a factor of 25–200, with the relative comparison of the different approaches again similar as among the derandomized variants.

We measured the running times of the six rounding algorithms on different types of random instances. All algorithms have been implemented with equal care in C/C++. The running times were measured on AMD dual processor 2.4 GHz Opteron machines. All numbers are the averages of 100 runs, where all algorithms were run on exactly the same instances. The results are displayed in Table 1 and Figure 1. In addition to the actual running times, we also state the number of pipage rounding iterations and the total number of times an edge weight was changed. Since our implementation of the path and cycle finding procedure is such that equally often an edge is found by it as it has its weight changed, the latter is a good machine-independent estimator for the running time.

From data depicted in Figure 1 we indeed see a superlinear increase of both the running time and the number of edges visits for the edge-based approach, whereas the bit-wise and hybrid approach rather show a linear dependence on the number of variables. The reason is that the edge-based approach for general values of the variables needs m iterations to round all numbers and each iteration roughly needs time proportional to the length of the cycle/path used for the pipage rounding. Hence the edge-based approach suffers from an increasing path/cycle length. A simple regression looking for the best-fitting αx^β approximation to the curves of the running times and the number of edge visits indeed indicates an different order of magnitude. For the number of edge visits, the β-value is 1.00, 1.07 and 1.37 for bit-wise, hybrid and edge-based approach, respectively. For the actual running times, the corresponding β-values are 2.01, 2.11 and 2.35. Note that the extra factor of roughly n stems from the fact that computing the pessimistic estimators has a complexity roughly proportional to the number of rounding errors we want to keep small, which in our settings is linear in n.

In return, of course, the bit-wise approach suffers from the length of the binary expansion, whereas the length of the paths/cycles is irrelevant.

The data in Table 1 also shows that the new hybrid algorithm is generally faster than the other two. This effect is significant and too substantial to be caused by simply imitating the faster of the two previously known algorithms. We did run the other algorithms also with the same preprocessing as the hybrid, but this typically

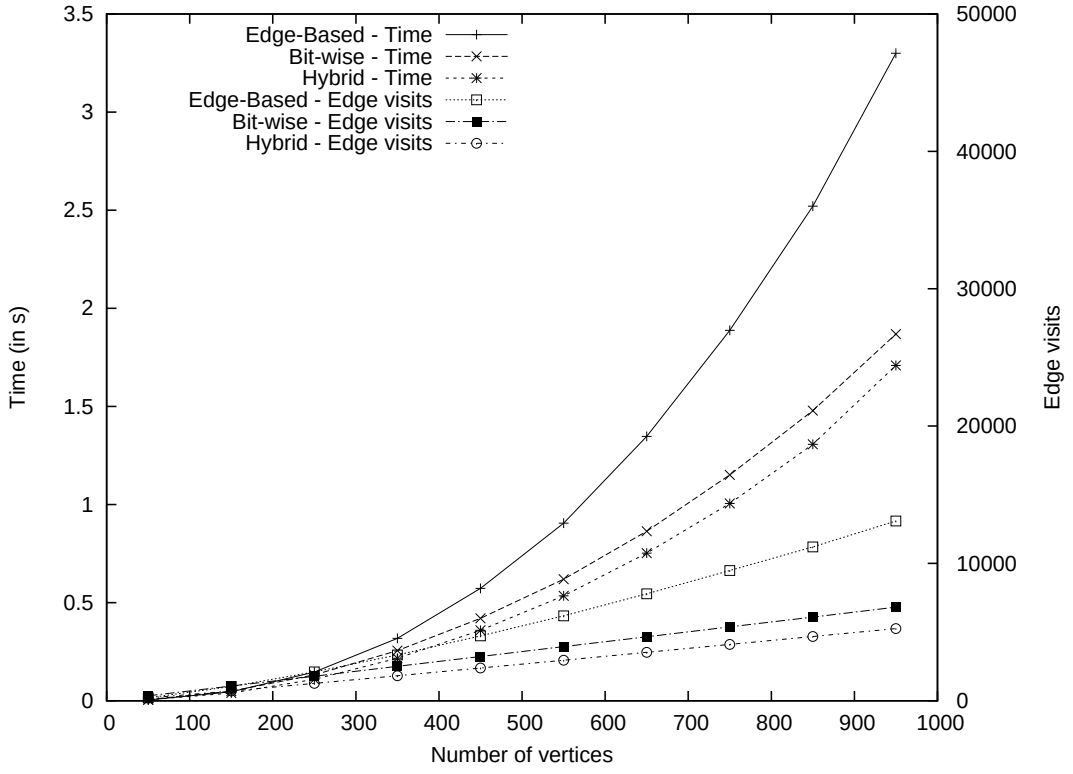

Figure 1: Running times and edge visits on 5-regular random bipartite graphs for different graph sizes. All algorithms shown are the derandomized variants.

decreased their performance. So in particular, this is not the reason why the hybrid algorithm is superior.

From the data given, we feel that we can identify two reasons for the good performance of the hybrid algorithm. For many instances the two ways of making progress (rounding edges and reducing bit-length) seem to combine nicely. This is visible from the reduced number of iterations needed, indicating that compared to the edge-based approach indeed the average number of edge weights fixed in one iteration is larger than one.

However, the hybrid approach is also significantly faster for the 5-matching instances, where both hybrid and edge-based approach need close to m iterations. Here we observe that the average path/cycle length is much smaller when running the hybrid approach. Since for this graph class typically only one edge is fully rounded per iteration, the running time is approximately proportional to the cycle length. The reason for the cycles being shorter seems to be that by only regarding edges with weight having a one in the least significant bit, the instance becomes even sparser, making shorter cycles and paths (or more appropriately, cycles going through the vertices added in the pre-processing of Section 2.3) more likely.

3.2 Rounding Errors Similar as for dependent randomized rounding with disjoint cardinality constraints, the edge-based rounding approach produces the lowest rounding errors among all derandomized algorithms. Since all three algorithms use the same pessimistic estimators, we find it hard to come up with a convincing explanation for this phenomenon.

All randomized algorithms produce very similar rounding errors (2.79-2.83 for the 5-regular random bipartite graphs, 5.27-5.37 for the almost 20-regular random bipartite graphs, 10.81-11.00 for the dense random bipartite graphs), all of which are significantly larger (up to around twice as large) than for the

derandomized algorithms.

4 Broadcast Scheduling

One setting where bipartite edge weight rounding has found application are broadcast scheduling problems [15]. In these problems, a server has a set of pages P, and receives requests for page broadcasts. Usually, time is divided into discrete time slots, and the server can only broadcast one page (or a bounded number of pages) per time slot. What makes the problem non-trivial is that when a particular page is broadcast, this simultaneously satisfies all requests for the page, not only one request. Possible optimization goals include maximizing the value of the satisfied requests, and minimizing the maximum or the average delay that any request experiences until it is satisfied.

Gandhi et al. [15] consider two of these goals, the maximum throughput and the minimum average delay versions. In the former setting, requests have deadlines after which they can no longer be satisfied; for this setting, Gandhi et al. present a 3/4-approximation. For the latter setting, they give a $(2, 1)$-bicriteria approximation, that is, a 2-speed schedule (a schedule broadcasting two pages per time slot) which on expectation incurs an average delay no greater than the smallest possible delay of a 1-speed schedule. Both these results remain the strongest in their respective direction. However, several other variants have been considered in the literature, see, e.g., [5–7, 18].

In this section, we present fully derandomized versions of the algorithms of Gandhi et al. [15], and report on the results of experiments where we apply them to broadcasting instances derived from Wikipedia access logs. The log files were downloaded from [23].

4.1 Brief Algorithm Description The algorithms of this section all follow the same basic pattern in producing a rounding problem from a broadcasting problem instance. We here sketch an outline of the algorithms, and subsequently provide proofs of our derandomizations. In general, for details and complete descriptions, we refer to Gandhi et al. [15].

As usual, we start with an optimal fractional solution to the natural LP-relaxation of the problem. In particular, this will contain a fractional schedule y_t^p, denoting the fractional broadcast of page p at time t, such that in every time slot pages with a total weight of at most one are broadcast. This fractional schedule is then converted into a bipartite edge weight rounding problem instance in the following manner. We create a bipartite graph $G = (V_1 \dot\cup V_2, E)$, where $V_1 = \{t_1, \ldots, t_k\}$ represents the time slots, and an edge incident to vertex t_i represents a page being broadcast at time i. The vertices V_2 will represent *windows*, grouping for each page p the fractional broadcasts of p consecutively into sets W_j^p, such that (with the possible exception of the first and the last windows) in each window, p will be broadcast exactly once; that is, the edges incident to a window represent fractional broadcasts y_t^p summing to exactly one. Note that a variable y_t^p may be split into more than one window to attain this. Thus, rounding an edge (t_i, W_j^p) to one represents a decision that the transmission of page p associated with its j:th window will occur at time i.

For the first window, a random value $z \in (0, 1]$, the *shift value*, is chosen, this being the sum of the values of its incident edges (again, splitting the last fractional transmission y_t^p of the window into two if necessary). In [15], the random selection of this shift value was essential to their approximation guarantees.

4.2 Derandomizing Max-Throughput Broadcasting As mentioned, Gandhi et al. [15] give a 3/4-approximation to the max-throughput broadcasting problem. The algorithm follows the scheme described in the previous section. We will give a derandomization, including for the selection of shift values z; we begin by sketching the argument for the probabilistic case.

Consider a single request r for a page p, with first possible broadcasting time t and deadline D. The variables y_t^p relevant to r are spread over at most two windows W_j^p and W_{j+1}^p, and the request is satisfied by the integral solution if and only if one of the corresponding edges is chosen. Assume for simplicity that the relaxed solution satisfies r completely. Then, the total value of the relevant edges in window W_j^p is a random variable y which depends on z, and the remaining edges, to a value of $1 - y$, are incident to window W_{j+1}^p. Because of the rounding procedure, we can make no statement of the joint probability that either of these windows ends up satisfying the request. However, the probability that the first window satisfies the request, by the exclusiveness of the events in a window, is exactly y, and the (unconditional) probability that the second window satisfies it is $1 - y$. Thus we estimate the probability that the request is satisfied by $\max(y, 1-y)$. By the linearity of expectation, the expected weight of the satisfied requests in the final broadcasting schedule is at least 3/4 of the total weight of all requests. Again, we refer to [15] for proof. (If a request is only fractionally satisfied by the fractional solution, then likewise, the probability over z and over the randomized rounding that the request is satisfied is at least 3/4 of its fractional satisfaction.)

We now show how these arguments can be turned into a derandomized form of the result.

THEOREM 4.1. *The weighted max-throughput broad-*

casting problem has a deterministic 3/4*-approximation.*

Proof. Let y_p^t be an optimal solution to the LP-relaxation. For a request r, with weight $w(r)$, let $A_r(z)$ be the set of edges relevant to r incident to its first associated window as a function of the shift z, and let $B_r(z)$ be its remaining relevant edges. Let $F_r(z) = w(r) \cdot \max(\sum_{e \in A_r(z)} x_e, \sum_{e \in B_r(z)} x_e)$, where x_e is the weight of edge e, be the estimation of the expected contribution by r to the value of the rounded schedule. Finally, let $F^p(z)$ be the sum of $F_r(z)$ for all requests r for page p. It is not hard to show, first, that we can make a deterministic choice to optimize the value of $F^p(z)$ for each page, and second, that we can use the sum over F^p for all pages p to guide the rounding algorithm.

For the first, we simply observe that $F^p(z)$ is a piecewise linear function, with events occurring at every point where some request r gets associated to exactly one window. Thus, it suffices to compute $F^p(z)$ at most $2R$ points, where R is the number of requests (which becomes R points if the LP satisfies all constraints).

For the second, imagine that we would replace each function $F_r(z)$ by a linear function $F'_r(z)$ matching its value, specifically by either $w(r) \sum A_r(z)$ or $w(r) \sum B_r(z)$, and let F' be the sum of these replacement functions. Then F' is a completely linear function, and since a pipage rounding is a linear adjustment one of its two candidate points x' will satisfy $F'(x') \geq F'(x) = F(x)$. Clearly, $F(x') \geq F'(x')$.

4.3 Derandomizing Minimum Average Delay Broadcasting We now turn to the derandomization of the minimum average delay broadcasting problem. Recall that in this version, requests have no deadlines, every request must eventually be satisfied, and the objective value we optimize is the average (or total) delay incurred for all requests. For our result, we first sketch an alternative proof of the analysis for the randomized algorithm for the minimum average delay broadcasting problem given by Gandhi et al. [15]. We then use the results and tools of this proof to give a complete derandomized version of the algorithm.

The randomized algorithm for the minimum average delay version given in [15] is a slight modification of the previously described one. The modification consists in that the fractional solution given by the LP-relaxation is doubled (i.e., every variable y_t^p is multiplied by two) before the division into windows and the construction of the bipartite graph begins. The critical property that this guarantees is that for every request r, if you consider the variables y_t^p that contribute to its incurred delay in the fractional solution, then there is one window in the bipartite graph that consists exclusively of such edges.

We now argue that we do not need the randomly chosen shifting value of z—we can take any value for z (including the convenient choice of one) and still obtain an expected average delay which is at most as large as the value of the LP. Besides simplifying the algorithm, this will be the basis of our derandomization.

Proof that any shifting value z leads to a randomized (2,1)-approximation algorithm: Let the shifting parameter z be chosen arbitrarily. Consider a single request r. The edges relevant to r are the edges from its starting time, until a point such that the edges sum up to a total value of at least one in the (unscaled) LP solution. These edges will be spread over up to three windows in the graph; call these W_A, W_B, and W_C, with corresponding sets of relevant edges A, B, and C. Let y be the total value of A.

Since the total value of W_B is one, r will be satisfied via an edge from $A \cup B$. For $e \in A \cup B$, let $p(e)$ be the probability that e is the earliest edge satisfying r. For edges in A, $p(e)$ equals x_e, and consequently with probability y an edge from A satisfies r.

The remaining mass $1 - y$ lies in B and has an unknown distribution, except that $p(e) \leq x_e$, since the latter is the probability that e corresponds to a broadcast of the requested page. Let $q(e) := x_e - p(e)$.

Now let $d(e)$ be the delay caused by the event that the edge e is the earliest edge chosen to satisfy r (defining $d(e) = 1$ if e lies in the first time slot that can satisfy the request r). Then the expected delay for r is $\sum_{e \in A \cup B} p(e)d(e) = \sum_{e \in A} x_e d(e) + \sum_{e \in B} p(e)d(e)$.

Recall that the contribution of r to the value of the LP is $\sum_{e \in A \cup B \cup C} (x_e/2)d(e)$.

Using elementary observations like (i) $\sum_{e \in B} p(e) = 1 - y$, (ii) $\sum_{e \in B} x_e = 1$ and (iii) the fact that for $e \in A$, $f \in B$ we always have $d(e) \leq d(f)$, we derive $\sum_{e \in A} x_e d(e) \leq \sum_{e \in A} (x_e/2)d(e) + \sum_{e \in B} (q(e)/2)d(e)$.

Similarly, using $\sum_{e \in C} x_e \geq 1 - y$, we derive $\sum_{e \in B} p(e)d(e) \leq \sum_{e \in B} (p(e)/2)d(e) + \sum_{e \in C} (x_e/2)d(e)$. Since, by definition, $q(e) + p(e) = x_e$ for all $e \in B$, the last two inequalities show that the expected delay caused by r is at most its contribution to the LP optimum.

A derandomized algorithm for the minimum average delay problem: We use the above argumentation to show a derandomization also for the minimum average delay problem.

THEOREM 4.2. *The weighted minimum average delay broadcasting problem has a deterministic* $(2, 1)$*-approximation.*

Proof. As explained above, there is no need to derandomize the choice of z, since it can be chosen arbitrarily.

Instance	Method	Random z		Optimal z	
		Time (in s)	Value ($\cdot 10^5$)	Time (in s)	Value ($\cdot 10^5$)
Small	Greedy	< 0.01	7.77	< 0.01	7.77
Small	Randomized (all)	≤ 0.01	8.02–8.06	0.045–0.05	8.44–8.50
Small	Bitwise, derand.	0.57	8.53	0.34	8.74
Small	Edge-based, derand.	0.1	8.65	0.06	8.75
Small	Hybrid, derand.	0.07	8.59	0.08	8.81
Small	LP-relaxation	0.05	8.85	0.05	8.85
Large	Greedy	< 0.01	24.6	< 0.01	24.6
Large	Randomized (all)	0.02–0.04	24.5–24.7	0.49–0.51	25.8
Large	Bitwise, derand.	6	26.0	5.4	26.6
Large	Edge-based, derand.	2.5	26.5	2.4	27.0
Large	Hybrid, derand.	2.5	26.5	2.1	27.3
Large	LP-relaxation	0.5	27.5	0.5	27.5

(a) Results for the max-throughput experiments

Method	Instance: small		Instance: large	
	Time (in s)	Value ($\cdot 10^6$)	Time (in s)	Value ($\cdot 10^6$)
LP-relaxation	24.6	6.10	1151	18.8
Randomized (all)	≤ 0.2	5.19–5.22	≤ 0.05	15.3
Bitwise, derand.	0.77	4.95	13	14.4
Edge-based, derand.	0.18	4.86	6.4	14
Hybrid, derand.	0.12	4.90	3.9	14
Greedy	< 0.01	3.76	< 0.01	11.5

(b) Results for the minimum average delay experiments. The LP-relaxation is 1-speed, all other algorithms 2-speed.

Table 2: Broadcast scheduling results.

It thus suffices to present a function that can serve as a guide for the rounding process. For this, consider the worst-case bound on the distribution $p(e)$ as implicit in the above paragraphs. This distribution will match x_e for edges A, and will assign mass $1 - w(A)$ to edges B, without exceeding x_e for any individual edge in B. Thus, the weight of $p(e)$ in B constitutes a "packing" of the mass at the end of window W_B; let $f_r(x)$ be the corresponding function value for a request r. Although f_r is not linear, depending in a non-trivial way on the edges in A, it can, as in the max-throughput case, be bounded by a linear function at any fractional assignment x.

Consider thus a particular fractional assignment x, and let $\hat{p}(e) = p(e)/x_e$, with $p(e)$ as described. Let e_0 be the unique edge in B for which $\hat{p}(e)$ is not integral (if no such edge exists, let e_0 be the last edge in B for which $\hat{p}(e) = 0$). Let $\hat{B}$ be those edges in B for which $\hat{p}(e) = 1$. Then it can be seen that $f'_r(x') = \sum_{e \in A \cup \hat{B}} d(e)x'_e + \delta d(e_0)$, with δ depending linearly on the weight of edges in A and $\hat{B}$, is an upper bound on $f_r(x')$.

4.4 Experimental Results Since we were not able to find truly real-world instances for the two broadcast scheduling problems discussed above, we generated two instances of different size using data from the Wikipedia access statistics [23], which we expect to show similar characteristics as instances showing up in broadcast scheduling problems.

A random subset of frequently accessed pages serves as our set of allowed pages. The time events are every third (small instance) or second (large instance) hour in a 6-day period in which each of the allowed pages is requested with weight equal to the number of accesses in that hour. The deadlines for each request are chosen equally as $D = 17$.

The results for the max-throughput experiments are given in Table 2a. Recall from the proof of Theorem 4.1 that there are two aspects, or levels, to the derandomization. One is the choice of the page shift z, which can be done randomly or deterministically, the other is the use of a derandomized rounding process. We report in the table all combinations of these aspects. Since the randomized rounding algorithms behave very similarly, the data for these is collected in one line.

We find that, perhaps surprisingly, the proper choice of z has almost as powerful an effect on the outcome of the experiment as using a derandomized rounding process. We also find that the combination, when both aspects are derandomized, produces very good results, being very close to the LP-optimum.

For comparison, we also include a simple greedy heuristic, which in every time slot broadcasts that page which would satisfy the largest weight of unsatisfied requests. For the max-throughput problem, this algorithm is comparable to the randomized rounding approaches, but is clearly inferior to all derandomizations.

The results for the minimum average delay problem are given in Table 2b; remember that here, a smaller value is better. The LP-relaxation shows the 1-speed optimum, while the remaining data is for 2-speed schedules.

As can be seen from the table, all (de)randomized algorithms find a resonable approximation to the LP optimum (clearly better than the approximation guarantee). Also, again derandomization improves over the randomized solutions. However, all these solutions are not very competitive, if compared to the 2-speed schedule produced by the simple greedy heuristic. So it seems that for this problem (or at least this type of instance), the approach via bipartite edge weight rounding is not a good idea.

References

[1] A. A. Ageev and M. Sviridenko. An approximation algorithm for hypergraph Max-Cut with given sizes of parts. In M. Paterson, editor, *Algorithms — ESA 2000*, Vol. 1879 of *Lecture Notes in Computer Science*, pp. 32–41. Springer, 2000.

[2] A. A. Ageev and M. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In G. Cornuéjols, R. E. Burkard, and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization (IPCO)*, Vol. 1610 of *Lecture Notes in Computer Science*, pp. 17–30. Springer, 1999.

[3] A. A. Ageev and M. I. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.*, 8:307–328, 2004.

[4] S. Arora, A. Frieze, and H. Kaplan. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Math. Program.*, 92:1–36, 2002.

[5] N. Bansal, M. Charikar, S. Khanna, and J. Naor. Approximating the average response time in broadcast scheduling. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 215–221, 2005.

[6] N. Bansal, D. Coppersmith, and M. Sviridenko. Improved approximation algorithms for broadcast scheduling. *SIAM J. Comput.*, 38:1157–1174, 2008.

[7] J. Chang, T. Erlebach, R. Gailis, and S. Khuller. Broadcast scheduling: algorithms and complexity. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 473–482, 2008.

[8] B. Doerr. Non-independent randomized rounding. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 506–507, 2003.

[9] B. Doerr. Roundings respecting hard constraints. In V. Diekert and B. Durand, editors, *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, Vol. 3404 of *Lecture Notes in Computer Science*, pp. 617–628, Berlin–Heidelberg, 2005. Springer-Verlag.

[10] B. Doerr. Generating randomized roundings with cardinality constraints and derandomizations. In B. Durand and W. Thomas, editors, *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS'06)*, Vol. 3884 of *Lecture Notes in Computer Science*, pp. 571–583, Berlin–Heidelberg, 2006. Springer-Verlag.

[11] B. Doerr and H. Schnieder. Non-independent randomized rounding and an application to digital halftoning. In R. Möhring and R. Raman, editors, *Algorithms—ESA 2002*, Vol. 2461 of *Lecture Notes in Computer Science*, pp. 399–410, Berlin–Heidelberg, 2002. Springer-Verlag.

[12] B. Doerr and M. Wahlström. Randomized rounding in the presence of a cardinality constraint. In *10th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 162–174, 2009.

[13] B. Doerr, M. Künnemann, and M. Wahlström. Randomized rounding for routing and covering problems: Experiments and improvements. In P. Festa, editor, *Experimental Algorithms*, Vol. 6049 of *Lecture Notes in Computer Science*, pp. 190–201, Berlin–Heidelberg, 2010. Springer-Verlag.

[14] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 323–332, 2002.

[15] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53: 324–360, 2006.

[16] E. Halperin and A. Srinivasan. Improved approximation algorithms for the partial vertex cover problem. In K. Jansen, S. Leonardi, and V. Vazirani, editors, *Approximation Algorithms for Combinatorial Optimization*, Vol. 2462 of *Lecture Notes in Computer Science*, pp. 161–174, Berlin–Heidelberg, 2002. Springer-Verlag.

[17] J. Hromkovič. *Design and analysis of randomized algorithms. Introduction to design paradigms.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2005.

[18] S. Im and B. Moseley. An online scalable algorithm for average flow time in broadcast scheduling. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1322–1333, 2010.

[19] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff–Hoeffding bounds. *SIAM J. Comput.*, 26: 350–368, 1997.

[20] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.*, 37:130–143, 1988.

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

[22] A. Srinivasan. Distributions on level-sets with applications to approximations algorithms. In *Proc. 42nd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 588–597, 2001.

[23] Wikistats. URL `http://dammit.lt/wikistats/`.