Randomized Rounding in the Presence of a Cardinality Constraint[*]

Benjamin Doerr[**] Magnus Wahlström[**]

Abstract

We regard the problem of generating randomized roundings with a single cardinality constraint. This is motivated by recent results of Srinivasan (FOCS 2001), Gandhi et al. (FOCS 2002, J.ACM 2006) and the first author (STACS 2005, STACS 2006). Our work results in (a) an improved version of the bitwise derandomization given by the first author, (b) the first derandomization of Srinivasan's tree-based randomized approach, together with a proof of its correctness, and (c) an experimental comparison of the resulting algorithms.

Our experiments show that adding a single cardinality constraint typically reduces the rounding errors and not seriously increases the running times. In general, our derandomization of the tree-based approach is superior to the derandomized bitwise one, while the two randomized versions produce very similar rounding errors. When implementing the derandomized tree-based approach, however, the choice of the tree is important.

1 Introduction and Results

Randomized rounding is one of the core primitives in randomized computation. Hromkovič [Hro05] lists it as one of seven design paradigms for randomized algorithms. While proven to be very efficient in many theoretical analyses, virtually no experimental research has been done on randomized rounding. The survey article [MNR97], one of the few sources that mention this aspect, has a 10-line section on computational experience building on [NRT87], [DR89] and a personal communication by Pulleyblank and Raghavan. Since, clearly, this short-coming cannot be solved in a single paper, in this work we concentrate on a particular aspect that became very interesting recently, namely randomized roundings that respect disjoint cardinality constraints.

1.1 Rounding. Rounding is usually understood as replacing a non-integral number by one of the two neighboring integers. That is, y is called rounding of $x \in \mathbb{R}$ if $y \in \mathbb{Z}$ and $|y-x| < 1$. Note that the integral part of x is not important, so whenever convenient we may assume that the numbers to be rounded are in $[0,1]$.

Rounding comes up in many areas. Natural examples include rounding in statistics, where it is used to increase readability of data and for confidentiality protection reasons [CCE85], or the digital halftoning problem in image processing, which is to transform a gray-scale image into one having black and white pixels only (cf. [AKOT03, Doe04, LA08]).

In algorithmics, rounding also is a way of dealing with the fact that many problems in which integral solutions are sought, become much easier without the integrality constraint. Hence solving the problem without caring for integrality and then trying to round it to an integer one, is a natural and often successful approach.

1.2 Independent Randomized Rounding. Classical randomized rounding was introduced by Raghavan and Thompson [RT87, Rag88]. The key idea is to round x randomly with probabilities given by its fractional part. Hence y is called *randomized rounding* of $x \in [0,1]$ if

$$\begin{aligned} \Pr(y=1) &= x, \\ \Pr(y=0) &= 1-x. \end{aligned} \tag{1.1}$$

Often, randomized rounding is used in the context of integer linear programming. If A is an $m \times n$ matrix with entries in $[0,1]$ (this is just a normalization), $x \in [0,1]^n$ and $b \in \mathbb{R}^m$ such that $Ax \le b$, then a (random) vector y obtained from applying randomized rounding independently to the components of x satisfies

$$(Ay)_i \le b_i + O(\sqrt{\max\{b_i, \log(m)\}\log(m)}) \tag{1.2}$$

for all $i \in [m] := \{1,\dots,m\}$ with high probability. Such bounds stemming from so-called large deviations bounds on sums of independent random variables are the reason why randomized rounding found numerous applications in approximation algorithms [MNR97].

This approach was derandomized using the method of conditional probabilities and pessimistic estimators by Raghavan [Rag88]. This yields deterministic rounding algorithms computing roundings with bounds on the rounding errors comparable to (1.2).

1.3 Randomized Roundings Respecting Cardinality Constraints. Whereas the independence in randomly rounding the variables ensures that the *rounding errors* $|(Ax)_i - (Ay)_i|$ are small with high probability, it is very weak in guaranteeing that a constraint

[*]Partially supported by the German Research Foundation (DFG) via its priority programme "SPP 1307: Algorithm Engineering", grant DO 749/4-1.

[**]Max–Planck–Institut für Informatik, Saarbrücken, Germany

is satisfied without error. Being the simplest type of such constraints, in this paper we are only concerned with *cardinality constraints*. These are constraints requiring that for a set of variables, their sum does not change through the rounding. In other words, for a $J \subseteq [n]$ we need that the rounding y of $x \in [0,1]^n$ satisfies $\sum_{j \in J} y_j = \sum_{j \in J} x_j$. Clearly, this makes sense only if $\sum_{j \in J} x_j$ is an integer.

Such constraints are quite common in the optimization literature, but they are also very useful in the design of randomized algorithms. Examples of this kind include routing applications ([RT87, RT91, Sri01]), flow problems ([RT87, RT91, GKR$^+$03]), partial and capacitated covering problems ([AS04, GKPS02, GHK$^+$03, CN06]) and the assignment problem with extra constraints ([AFK02]).

While clearly useful, it is not so easy to generate randomized roundings respecting such constraints. As said above, independent randomized rounding is not suitable. On the other hand, independence and the resulting large deviation bounds for the rounding errors were the reason why the classical randomized rounding became that successful.

The first to overcome this difficulty was Srinivasan [Sri01] (see also [GKPS06]). He presented a way to compute randomized roundings that respect the constraint that the sum of all variables remains unchanged (global cardinality constraint). Though not independent, they still fulfill some negative correlation properties, which imply the usual Chernoff bounds. Among other results, this yields a randomized algorithm for the integer splittable flow problem.

Srinivasan also shows that, surprisingly, the simple approach of generating independent randomized roundings until the hard constraint is satisfied, is not feasible. The resulting roundings may not be randomized roundings.

The deterministic "pipage rounding" algorithm of Ageev and Sviridenko [AS99, AS00, AS04] allows to round edge weights in a bipartite graph in such a way that the sum of weights incident with a vertex changes by less than one ("degree preservation"). This yields improved approximation algorithms for maximum coverage problems and max-cut problems with given sizes of parts. Ageev and Sviridenko note that their ideas could be used in a randomized way, but "the resulting algorithm will be too sophisticated to admit derandomization".

The ideas of [AS99, AS00, AS04] and [Sri01] were combined in Gandhi, Khuller, Parthasarathy and Srinivasan [GKPS02, GKPS06] to obtain randomized roundings of edge weights in bipartite graphs that are degree preserving and in addition fulfill negative correlation properties on sets of edges incident with a common vertex. This again yields improved randomized approximation algorithms for several problems as well as some nice per-user fairness properties.

Both in [Sri01] and [GKPS02, GKPS06], the derandomization problem was not regarded.

Independent of [Sri01, GKPS02, GKPS06] and using different methods, the first author in [Doe05] and [Doe06] (the latter is the paper we will refer to in the following) developed another way to generate randomized roundings that respect cardinality constraints. Due to the simpler, less sequential random experiment underlying the construction, this approach could be derandomized via a reduction to Raghavan's derandomization [Rag88] for the problem without hard constraints. Also, this approach proved to be superior for more complicated sets of cardinality constraints. For example, generating randomized roundings for the bipartite edge weight rounding problem can now be done in time $O(|E|\ell)$ instead of $O(|V||E|)$. Here (V, E) is the bipartite graph considered and ℓ is an upper bound for the (binary) coding length of each edge weight.

To ease the distinction, we call Srinivasan's approach *tree-based* and the one of [Doe06] *bitwise*, a naming motivated by underlying algorithms. Let us add that the approach of [Doe06] was extended to use other expansions than the binary one in [Doe07]. This gives superior results if all numbers to be rounded are rational and their denominators contain only small prime factors. While this is useful for certain problems, e.g., the controlled rounding problem from statistics, we shall not regard this aspect here.

1.4 Motivation. From the above, it is clear that for complex rounding problems, the new approach of [Doe05, Doe06] is the preferred choice. What is not clear, and hence is the starting point of this research, is how the tree-based and bitwise method compare for disjoint cardinality constraints; as the examples in [Sri01, Doe05] indicate, this is a highly useful special case. Since the problem for disjoint constraints is in both of these methods easily reduced to the one of one cardinality constraint involving all variables (by rounding the variables within each hard constraint in turn), we restrict ourselves to this problem.

Due to their very different nature, a comparison of the two rounding methods is not trivial (see below for a precise description of the methods). Also, such a comparison can probably not be done via theoretical means only, as both methods give the same asymptotic running times and error guarantees.

A second set of questions regarded in this work is how randomized rounding with hard constraints compares to the classical approach of Raghavan and Thompson (without constraints), again both in terms of running times and rounding errors.

1.5 Results. We implemented the algorithms introduced above. It is known that derandomized algorithms often are superior to their randomized counter-parts due to their greedy way of making some decisions. We observed the same for classical randomized rounding without constraints and, to a significantly lesser degree, for the approach in [Doe06]. This first observation made it highly desirable to also have a derandomization of Srinivasan's rounding approach. This seems difficult due to its highly sequential nature, but a careful analysis of Raghavan's paper [Rag88] shows that his pessimistic estimator can also be used in the more dependent random experiment underlying Srinivasan's randomized rounding. Hence a first, theoretical result stemming from this work is that Srinivasan's randomized roundings can be derandomized in time $O(mn)$, where $m \times n$ is the dimension of the matrix encoding the rounding errors.

Comparing randomized and derandomized versions again, we find that both Raghavan's and Srinivasan's approach show a similar advantage of the derandomized version over the randomized one, whereas derandomization was found to be less superior for the bitwise approach. We find two possible reasons. (i) The bit-levels are rounded independently, and hence there is no way of exploiting a greedy behaviour interrelating the bit-levels. (ii) In [Doe06], the problem is reduced to a derandomization problem for $\{-1, 0, 1\}$ matrices. Since Raghavan's derandomization is formulated for $\{0, 1\}$ matrices only, in [Doe06] a simple union bound type approach is taken (writing $A = A_1 - A_2$ for $\{0, 1\}$ matrices A_1, A_2 and derandomizing with respect to them). Extending Raghavan's derandomization to $\{-1, 0, 1\}$ matrices (and thus being able to derandomize with respect to A directly), we observe a much weaker effect, concluding that (ii) was the main reason for the derandomization being less superior to the randomized version than expected.

We compare the different ways of computing roundings for two classes of instances, namely random instances and structured instances stemming from formulating the problem of generating low-discrepancy point sets for numerical integration as a rounding problem.

Summarized, the details given in Section 4 show the following. In most cases, the algorithms respecting the cardinality constraint on all variables produce smaller rounding errors on other constraints than classical randomized rounding and its derandomization. In such, there is no "price of hard constraints" except a moderate increase in running times. In most cases, the bitwise derandomization of [Doe06] suffers from regarding bit-levels independently, both in terms of run-time and rounding errors. On the other hand, both randomized versions produce almost identical rounding errors. We also observe that when implementing the derandomized tree-based approach, the choice of the tree is important. In particular, we observe the natural choice of a tree that results in rounding the variables in a (again natural) linear order, to lead to five times larger rounding errors in certain instances. Even when using a random linear order, we still see much larger rounding errors.

2 Description of the Rounding Algorithms

We shall now describe the three randomized rounding algorithms and their derandomizations. As discussed earlier, we lose nothing by assuming that the numbers $x_1, \ldots, x_n$ to be rounded are all in $[0, 1]$. We shall assume that we have only one constraint involving all variables. To make the cardinality constraint feasible, we assume that the sum of all variables is an integer.

2.1 Randomized Rounding. As discussed in the introduction, classical randomized rounding (without hard constraints) was introduced by Raghavan and Thompson. It consists of simply rounding each variable independently with probabilities as in (1.1).

Srinivasan's approach to maintain a single cardinality constraint involving all variables is repeatedly rounding as follows. In arbitrary order, take two variables x_{j_1} and x_{j_2}, add a random ε to one and reduce the other by ε. The random ε is chosen from $[-1, 1]$ in such a way that one variables becomes an integer, the other variable stays in $[0, 1]$ and $E(x_{j_k}) = x_{j_k}$ for $k = 1, 2$. Note that this precisely defines the distribution of ε. The procedure that performs this operation on two variables is called `simplify`.

While Srinivasan's approach builds on the fact that rounding with cardinality constraints is easy for two variables, the approach in [Doe06] exploits the fact that this problem is easy if all variables have values in $\{0, \frac{1}{2}\}$. Let us describe this special case first and then the reduction to the special case. If all variables are in $\{0, \frac{1}{2}\}$ and add up to an integer, we can partition the variables that are equal to $\frac{1}{2}$ into pairs. For each pair, independently and uniformly at random, we pick one variable to be rounded up to one and round the other down to zero. Clearly, this maintains the cardinality constraint and satisfies (1.1).

If all variables have a finite binary representation each of length at most ℓ, we can use the above to reduce the binary length. We write $x = x' + 2^{-\ell+1}x''$ with $x' \in [0, 1]^n$ having binary length $\ell - 1$ and $x'' \in \{0, \frac{1}{2}\}^n$. Let y'' be the outcome of the above rounding procedure applied to x''. Then $y := x' + y''$ has binary length $\ell - 1$ and satisfies $\sum_{j=1}^n y_j = \sum_{j=1}^n x_j$. Hence repeating this for another $\ell - 1$ iterations, we obtain a random binary vector that respects the cardinality constraint with probability one.

As is obvious from the description or proven in [RT87, Sri01, Doe06], all three types of roundings (i) can be computed in time $O(n)$ (assuming constant length binary expansions in the third case), (ii) are randomized roundings in the sense of (1.1), (iii) respect the hard constraint (except for Raghavan's approach), and (iv) satisfy large deviation bounds like (1.2). More precisely, for all $\lambda \geq 1$, we have $|(Ay)_i - (Ax)_i| \leq (e-1)\sqrt{\max\{(Ax)_i, \ln(2m)\}\ln(2\lambda m)}$ for all $i \in [m]$ with probability at least $1 - 1/\lambda$ (for all three ways of generating randomized roundings).

2.2 Derandomizations. The derandomization problem is to (typically deterministically) compute a rounding y of x that satisfies bounds like (1.2) not only with high probability, but surely. This is done via the method of conditional probabilities. We round the variables one after the other, where the current variable is rounded to that value which, assuming the remaining variables are rounded in the randomized rounding fashion, minimizes the probability of violating the large deviation bounds (or a suitable estimate thereof). Typically, this leads to a run-time of $O(nm)$.

The crucial problem here is estimating the conditional probabilities involved. Raghavan [Rag88] gives so-called pessimistic estimators that fulfill this purpose under reasonable restrictions. This leads to the following result.

Theorem 2.1. (Raghavan (1988)) *Let $x \in ([0,1] \cap \mathbb{Q})^n$ and $A \in \{0,1\}^{m\times n}$. In the RAM model of computation, a $y \in \{0,1\}^n$ can be computed in time $O(mn)$ such that for all $i \in [m]$,*

$$|(Ay)_i - (Ax)_i| \leq (e-1)\sqrt{\max\{(Ax)_i, \ln(2m)\}\ln(2m)}.$$

Building on this derandomization, the following derandomization for the case of cardinality constraints was derived in [Doe06].

Theorem 2.2. *Let $A \in [0,1]^{m\times n}$. Let $x \in [0,1]^n$ such that $\sum_{j=1}^n x_j \in \mathbb{Z}$. Then for all $\ell \in \mathbb{N}$, in time $O(mn\ell)$ a rounding $y \in \{0,1\}^n$ of x can be computed such that $\sum_{j=1}^n y_j = \sum_{j=1}^n x_j$ and*

$$|(Ax)_i - (Ay)_i| \leq 90\sqrt{\max\{(Ax)_i, \ln(4m)\}\ln(4m)} + n2^{-\ell}$$

for all $i \in [m]$.

It is noted in [Doe06], the proof there is not optimized to yield the best possible constant. After adding a small improvement in Section 3, we obtain a derandomization that in our experiments in Section 4 produces rounding errors that are between 50% and 150% of those produced by Raghavan's derandomization.

For Srinivasan's way of generating randomized roundings no derandomization was known previously. We shall fix this in the following section.

3 New Derandomizations

In this section, we sketch the new derandomizations we developed while experimenting. Most noteworthy is the derandomization of Srinivasan's randomized roundings with cardinality constraints.

3.1 Derandomizing Srinivasan's Randomized Roundings. As discussed in the introduction, Srinivasan did not provide a derandomization for his randomized roundings with cardinality constraints. Due to the sequential, highly dependent nature of the underlying random experiment (see the previous section), it may seem unlikely that a derandomization for this approach exists (at least the first author thought so and therefore developed an alternative approach). In this subsection, we show that not only does a derandomization exist; Raghavan's method works directly (i.e. the pessimistic estimators can be used to select between the two outcomes of an application of `simplify`).

Theorem 3.1. *Let $A \in [0,1]^{m\times n}$ and $x \in [0,1]^n$, such that $\sum_{j=1}^n x_j \in \mathbb{Z}$. Then, a $y \in \{0,1\}^n$ can be computed in time $O(mn)$ such that $\sum_{j=1}^n y_j = \sum_{j=1}^n x_j$ and for all $i \in [m]$,*

$$|(Ay)_i - (Ax)_i| \leq (e-1)\sqrt{\max\{(Ax)_i, \ln(2m)\}\ln(2m)}.$$

Proof. We will show that for each binary decision the algorithm has to make (in this case, assigning x_a or x_b out of a pair of variables x_a, x_b), the estimated conditional probability of a large deviation will decrease with at least one of these assignments. The rest of the result follows from Raghavan's original paper [Rag88].

The formula for Raghavan's pessimistic estimators is

$$C_i^+ \prod_{j=1}^n ((e^{a_{i,j}t_i} - 1)p_j - 1)$$

for the probability that $(Ay)_i$ grows too big, and

$$C_i^- \prod_{j=1}^n ((e^{-a_{i,j}t_i} - 1)p_j - 1)$$

for the probability that it grows too small, where C_i^+, C_i^- and t_i are row-specific constants, p_j is the expected value of x_j, and $a_{i,j}$ is the entry of the matrix

A in position (i,j). In particular, note that for a fix i, the constant in front of p_j depends only on $a_{i,j}$, which is in our setting either 0 or 1, and if $a_{i,j}=0$, then this constant is naturally 0.

Consider thus one estimator, with an estimated probability of failure of P, and assume that $a_{i,a}=a_{i,b}=1$. Letting P_a resp. P_b be the result of adjusting for x_a resp. x_b being assigned, we will show $\min(P_a,P_b)\le P$ by showing $P_a+P_b\le 2P$. Abstracting away the common constants, we see that this depends on the value of

$$\begin{aligned}(p_a+\delta_a+\tfrac{1}{c})(p_b-\delta_a+\tfrac{1}{c})+\\(p_a-\delta_b+\tfrac{1}{c})(p_b+\delta_b+\tfrac{1}{c})-2(p_a+\tfrac{1}{c})(p_b+\tfrac{1}{c})\\=(p_b-p_a)(\delta_a-\delta_b)-\delta_a^2-\delta_b^2\end{aligned}$$

for a constant c. Now we only have to consider the cases of the `simplify` procedure of [Sri01]: If $p_a+p_b\le 1$, then $\delta_a=p_b$ and $\delta_b=p_a$, and the expression reduces to $-2p_ap_b<0$; otherwise, $\delta_a=p_b-1$ and $\delta_b=p_a-1$, and the expression reduces to $2p_a+2p_b-2p_ap_b-2=2(p_a-1)(1-p_b)<0$ (since $p_a,p_b<1$). The theorem, including the constant, now follows from Raghavan [Rag88].

Remarkably, in this case the proof of the derandomization turns out to be easier than the proof of the randomization, which uses more complicated machinery of negative correlation (see [Sri01]).

3.2 Extending Raghavan's Derandomization to $\{-1,0,1\}$ Matrices. As discussed in the introduction, Raghavan formulated his derandomization for $\{0,1\}$ matrices only, but the bitwise derandomization needs a derandomization for $\{-1,0,1\}$ matrices.

The simple way (taken in [Doe06]) is to decompose the $\{-1,0,1\}$ matrix A into two $\{0,1\}$ matrices A_1,A_2 such that $A=A_1-A_2$, solve the derandomization problem for the matrix $\binom{A_1}{A_2}$ and note that the rounding errors for A are at most twice the ones for $\binom{A_1}{A_2}$.

For randomized rounding, such tricks may increase the theoretical bounds, but clearly, the true rounding errors would be those given by the Chernoff bounds using the matrix A, simply because we do independent randomized rounding without taking the matrix into account in the rounding procedure.

Surprisingly, things are different for derandomizations as first experiments showed (cf. Subsection 4.6). For this reason, we extended Raghavan's derandomization to $\{-1,0,+1\}$ matrices.

THEOREM 3.2. *Let $x\in([0,1]\cap\mathbb{Q})^n$ and $A\in\{-1,0,1\}^{m\times n}$. Denote by $|A|$ the matrix obtained from A by taking absolute values in each entry. Then in the RAM model of computation, a $y\in\{0,1\}^n$ can be computed in time $O(mn)$ such that for all $i\in[m]$,*

$$|(Ay)_i-(Ax)_i|\le(e-1)\sqrt{\max\{(|A|x)_i,\ln(2m)\}\ln(2m)}.$$

Proof. Since we use Theorem 3.2 only in the case that $x\in\{0,\frac{1}{2}\}^n$, we shall prove the result only for this setting. So let us assume that $x\in\{0,\frac{1}{2}\}$.

We claim that we can now use Raghavan's pessimistic estimators. To see this, note first that they bound the probability that the rounding error in one of the m linear constraints exceeds the given limit, by the sum of these probabilities (cf. e.g. equation (2.8) in [Rag88]). Hence it suffices that we show that the estimators for each of these individual events also work if some a_{ij} have the value -1.

Such an estimator bounds the probability that a weighted sum $\sum_{j=1}^n a_jX_j$, $a_j\in\{0,1\}$, of independent $\{0,1\}$–valued random variables X_j deviates from its expectation by more than given limits. Now assume that some of the a_j are -1 and that all X_j are randomized roundings of numbers $x_j\in\{0,\frac{1}{2}\}$. Then we define a_j^* by $a_j^*=1$, if $a_j=-1$ and $x_j=\frac{1}{2}$, and $a_j^*=a_j$ otherwise. We also define $\delta_j=1$, if $a_j=-1$ and $x_j=\frac{1}{2}$, and $\delta_j=0$ otherwise. Then $\sum_{j=1}^n a_jX_j$ has the same distribution as $\sum_{j=1}^n(a_j^*X_j-\delta_j)$. In consequence, $\sum_{j=1}^n a_jX_j$ has the same distribution of its deviation from the mean as $\sum_{j=1}^n a_j^*X_j=\sum_{j=1}^n|a_j|X_j$. For the latter, however, we may simply employ Raghavan's estimators.

In Subsection 4.6 we shall see that using the new version of the derandomization in the bit-wise derandomization reduced the rounding errors observed in experiments by roughly a third.

On the theoretical side, using this derandomization, for any $A\in\{0,1\}^{m\times n}$ and $x\in\{0,\frac{1}{2}\}^n$ we can compute in time $O(mn)$ a rounding y of x respecting the cardinality constraint such that $|(Ay)_i-(Ax)_i|\le(e-1)\sqrt{\max\{(Ax)_i,\ln(2m)\}\ln(2m)}$ for all $i\in[m]$ (where the constant $e-1$ is an improvement by a factor of two over the previous method of combining Lemma 4 in [Doe06] with Raghavan's derandomization). Since the constant of 90 in Theorem 2.2 depends superlinearly on the one of the $\{0,\frac{1}{2}\}$ case, this reduces the constant in Theorem 2.2 to 18 (this is $f(e-1)$ in [Doe06]).

THEOREM 3.3. *Let $A\in[0,1]^{m\times n}$. Let $x\in[0,1]^n$ such that $\sum_{j=1}^n x_j\in\mathbb{Z}$. Then for all $\ell\in\mathbb{N}$, in time $O(mn\ell)$ a rounding $y\in\{0,1\}^n$ of x can be computed such that $\sum_{j=1}^n y_j=\sum_{j=1}^n x_j$ and*

$$|(Ax)_i-(Ay)_i|\le 18\sqrt{\max\{(Ax)_i,\ln(4m)\}\ln(4m)}+n2^{-\ell}$$

for all $i\in[m]$.

From now on, when talking about the derandomized version of the bitwise approach in [Doe06], we shall always mean the improved variant just discussed.

4 Experiments

We will now present our experimental results. As mentioned in the introduction, we test randomized and derandomized versions of Raghavan's classical roundings [Rag88], Srinivasan's tree-based roundings [Sri01], and the first author's bitwise roundings [Doe06], making six methods in total. The tests are divided into two groups: random instances, with various parameters, and structured instances stemming from an approach to generate low-discrepancy point sets via randomized rounding respecting a cardinality constraint due to the first author and Gnewuch [DG06]. We will first give a short discussion of implementation details.

Throughout this section, we use shorthand names for the methods in graphs and tables: *ragh-rand* and *ragh-derand* for random respectively derandomized classical rounding, *sri-rand* and *sri-derand* for Srinivasan's method, and *doerr-rand* and *doerr-derand* for bitwise rounding.

4.1 Implementation. All methods were implemented by the authors in standard C code, and compiled by the GNU C compiler (gcc) version 4.1.1 under Linux. In general, double-precision floating point numbers were used for all real-valued numbers; no significant imprecision was detected (more on this below). For the randomizations, and other randomness called for, the GNU C library pseudo-random function was used, seeded from the Linux entropy pool.

The text in this section assumes familiarity with the methods. For descriptions, see Section 2 and the original papers.

4.1.1 Classical Roundings. The classical derandomized rounding was implemented following Raghavan's paper [Rag88], using floating-point numbers to emulate reals, but restricted to 0/1 entries in the matrix (the extended derandomization described in Section 3 that handles matrices with -1 entries was implemented for a special case only, for use in the bitwise derandomization). To get an $O(mn)$ running time, as described in the paper, the pessimistic estimators were calculated fully only initially (this calculation taking $O(mn)$ time), and subsequently modified for every assignment made. Though this was a potential source of accumulated imprecision, comparing the values of the modified estimators to the values of estimators recalculated from scratch revealed no significant errors (usually the error had roughly the same order of magnitude as the floating point epsilon used, 10^{-16}).

The method is greedy in that when selecting between two assignments, it picks the assignment which minimizes the sum of the pessimistic estimators (representing the risk of breaking a bound). Parameters were chosen so that the initial sum of the estimators was 0.99, and this value subsequently decreases steadily during the run of the algorithm (literally not a single occasion was found where both assignments considered would lead to an increased estimated risk, not even due to floating point imprecision).

Classical randomized rounding provided no implementation difficulties beyond the issue of randomness touched on above.

4.1.2 Srinivasan's Tree-based Method. We have implemented Srinivasan's method for generating randomized roundings respecting disjoint cardinality constraints [Sri01], here referred to as the tree-based method since the repeated applications of the `simplify` procedure forms a tree of variable comparisons, as well as our new derandomization of this procedure given in Section 3.

The order of comparisons for both the randomized and derandomized versions is that of a balanced tree, with the variables taken in the order they are given (implemented in-place through a loop with increasing step lengths). Though the adjustment of probabilities is highly sequential, the precision in the calculations was again found to cause no problems.

The derandomization uses classical derandomized rounding, modified to consider two variables at a time. The specifics of this procedure are as given above.

4.1.3 Bitwise Rounding. Finally, we implemented the bitwise roundings of the first author [Doe06], using the extension described in Section 3. We chose 64-bit long integers as precision for the fix-precision numbers, to match the 52-bit mantissa of the doubles we used elsewhere.

For the pairing of variables that is performed when rounding each bit, we just pair the variables up in the order they are given (i.e. for each bit, the first two variables that are non-zero in this bit are paired, and the next two, and so on).

The derandomization uses the modification of Raghavan's derandomization given in Theorem 3.2 in a special-case modification that uses the fact that all variables to be rounded are 0 or $\frac{1}{2}$. We did not try to further optimize Raghavan's derandomization for this case, though we would expect some additional advantages. In particular, we did not use the opportunity to in this case exactly compute the conditional probabilities via binomial coefficients.

4.2 Running Time. Here, we show the running times for the methods. The times were measured on a CPU server with Intel Xeon CPUs at 3 GHz with 512 KB of cache, and 3 GB of memory.

Table 1 gives the running times for a rounding call for an instance with 10,000 variables and 10,000 rows of the matrix, for the variables having long and short bitstrings (i.e. random values resp. all $x_j = \frac{1}{2}$). The calculation of the rounding error for this size of instances takes approximately one second. Because the random methods round 10,000 variables so quickly that the measured times threaten to be mostly noise, these times are given for rounding a million variables as well.

Though the difference in time for the $O(n)$ random methods and the $O(mn)$ derandomized methods is quite big, the time required for the derandomizations is still manageable for these sizes, and as we shall see in the following sections, the derandomizations frequently have a big advantage in terms of solution quality.

4.3 Random Instances. Our first set of tests are square random matrices with 0/1 entries. The matrices are either dense, with a $\frac{1}{2}$ independent chance of any given entry being a 1, or sparse, with a $\frac{1}{20}$ independent chance, and the variables are either random, i.e., uniformly random in $(0,1)$, or all equal to $\frac{1}{2}$. For the methods that support hard constraints, one global constraint preserving the sum of all variables was used; the sum of the variables is an integer in each case. The source of randomness was again the GNU C library pseudo-random routine, seeded from the Linux entropy pool.

Figure 1 shows the average rounding errors (i.e., the average values of the (maximum) rounding errors) for the six methods, for each of these four types of random experiment; the average is taken over ten instances for each point. The largest difference is that between the random and the derandomized methods, which is presumably due to the greedy nature of the derandomizations, but also note how the behaviour of the classical roundings depends on the denseness of the matrix, and how that of the bitwise roundings varies with the bit-length of the probabilities. To show the results more precisely, Table 2 gives the average rounding errors adjusted by the outcome of classical randomized rounding. As can be seen, the effect of derandomization in these experiments is a gain of a factor of approximately two (with the exception of bitwise rounding for long bitstrings, as discussed elsewhere.) The line "theory bound" shows the value of relative size of the bound given in Theorem 2.1.

More discussion on these outcomes follows in Section 4.7.

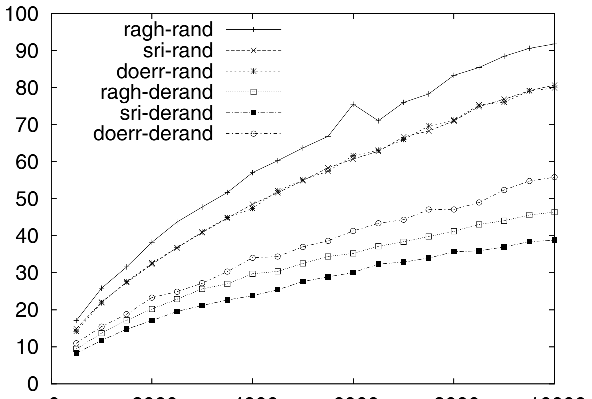
4.4 Low-discrepancy Pointset Instances. Our second set of tests are instances of a more structured type. The source of the instances is the problem of generating low-discrepancy pointsets, i.e. placing n points in a d-dimensional hypercube $[0,1]^d$ so that, roughly speaking, every subcube contains approximately the expected number of points. Doerr and Gnewuch [DG06] have proposed a method for creating such pointsets, where a critical part of the process involves solving a rounding problem.

The rounding instances of this section are discretizations of the problem: The axes are subdivided into k intervals, forming a non-regular grid of k^d small boxes, and the corresponding rounding instance is the problem of deciding the number of points to be placed in each such box. Thus, instances in this section are created deterministically from three parameters: the number of points n, the number of subdivisions k, and the number of dimensions d. The result is a rounding instance on k^d variables, with a $k^d \times k^d$ matrix of soft constraints describing the discrepancy of subcubes formed from the grid[1] and one global hard cardinality constraint stating that the number of points in total is to be preserved. If you are familiar with the topic, you should note that the rounding errors for these instances are not the final discrepancies, which will be higher, as the exact placements of the points inside their boxes have not yet been fixed.

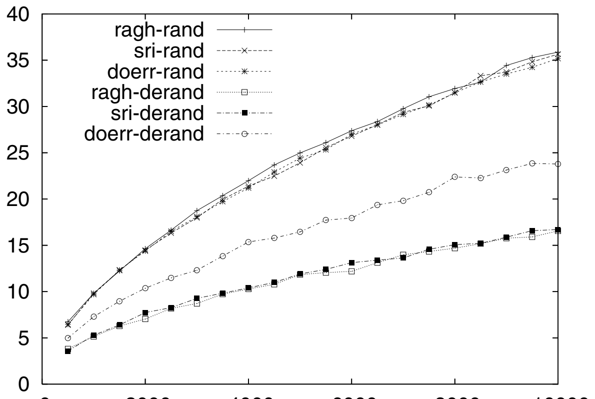
Our first test uses two dimensions only, with the number of subdivisions varying from 2 to 100 (representing instances with 4 to $10,000$ variables). Since the exact value of n turns out to have significant and irregular effects on the rounding errors, we did runs with n varied from 90 to 110 to introduce some variation. The results are displayed in Figure 2; the first three graphs display the range of variation for the random and derandomized versions of the three basic methods, and the fourth compares the average outcomes for the derandomized versions. For these experiments, there is no clear difference between Srinivasan's method and bitwise rounding, but it seems that ignoring the cardinality constraint not only leads to the wrong number of points, but to lower-quality solutions as well.

Our second test for these instances varies the number of dimensions from 2 to 14, while keeping the number of subdivisions constant at 2 (thus creating instances with $2^2 = 4$ to $2^{14} = 16384$ variables). The number of points was again varied from 90 to 110. The

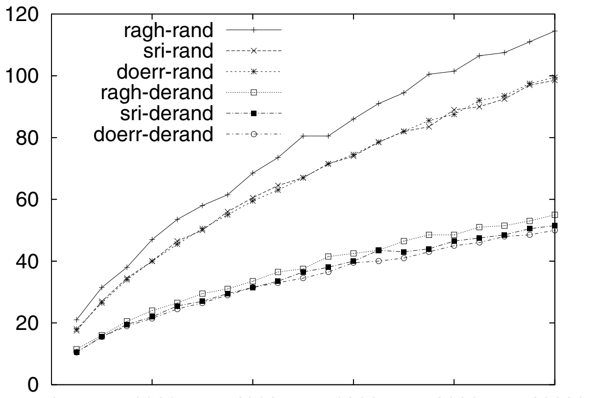
[1] For reasons we will not go into here, it suffices to consider subcubes with one corner in the origin.



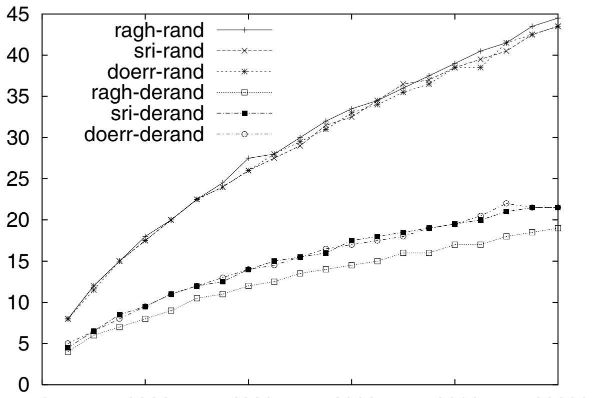
a) Dense matrix, random vector



b) Sparse matrix, random vector



c) Dense matrix, 1/2 vector



d) Sparse matrix, 1/2 vector

Figure 1: Rounding errors inflicted by the six rounding algorithms on four different types of random input instances (described in Section 4.3).

Instance	ragh-rand	sri-rand	doerr-rand	ragh-derand	sri-derand	doerr-derand
10,000 numbers, short	< 0.01s	< 0.01s	< 0.01s	49s	40s	24s
10,000 numbers, long	< 0.01s	< 0.01s	0.02s	52s	75s	666s
1 million numbers, short	0.05s	0.09s	0.38s	-	-	-
1 million numbers, long	0.05s	0.14s	1.20s	-	-	-

Table 1: Times needed to round instances with 10,000 and 1 million numbers. Short means that the numbers to be rounded have a 1-bit expansion, that is, are all $\frac{1}{2}$. Long numbers are 32-bit random numbers in $\{0, 2^{-32}, \ldots, 1 - 2^{-32}\}$. For the derandomized versions, a random $\{0, 1\}$ matrix of 10,000 rows was used to define the linear constraints.

results are shown in Figure 3. Again, the classical roundings are of lower quality, but here we also see a noticeable advantage of the derandomization of Srinivasan's method over derandomized bitwise rounding.

4.5 Effect of Tree Shape in Srinivasan's Method. In this section, we discuss an odd effect that occurred in testing the derandomization of Srinivasan's method.

Specifically, we found that for certain structured instances, the shape of the tree of `simplify` comparisons has a large effect on the rounding error in the derandomization (though none was seen for the random version of the method, or for random instances). The "tree" in our first implementation, as the issue was not considered, was actually a stack—simply a list, where the variables were assigned one by one, which viewed as a tree would have depth n for n variables. When tested on the discrepancy instances of the previous section, this less balanced, stack-like derandomization turned out to produce increasingly bad solutions with an increasing number of subdivisions. In addition, this effect seems to be a property of the tree shape itself, rather than of choosing a particularly bad variable ordering, as random permutations of the order of the variables ("shuffling") before the call was not found to compensate.

Figure 4 illustrates the effect, repeating the same $d = 2$, $2 \leq k \leq 100$ experiment as in Figure 2. On the left is the average outcomes of the variants of Srinivasan's method—the standard random procedure, the stack-like derandomization with and without shuffling the variables, and the derandomization using a balanced tree, the latter being the one used in the rest of this paper. On the right is a graph illustrating the ranges (tenth to ninetieth percentile of the tests, together with the median). Note that even when shuffling the variables it is rare to get an ordering which produces even as good results as the average outcome of the random version, not to mention the far superiour balanced derandomization.

We can offer no good explanation for such a specific occurrence. In the absence of better interpretation, all we can say is to recommend to avoid unbalanced trees in derandomizations of Srinivasan's method.

4.6 Effect of Using the Extended Raghavan Derandomization. In Section 3, we extended Raghavan's derandomization to include matrices having -1 entries. In Fig. 5 we depict the rounding errors produced by the bitwise derandomization using both the (old) decomposition trick and the new version of Raghavan's derandomization. We applied both algorithms on random instances as described in more detail in Section 4.3. We see that, roughly speaking, we pay for using the decomposition trick with a 50% increase in the rounding errors.

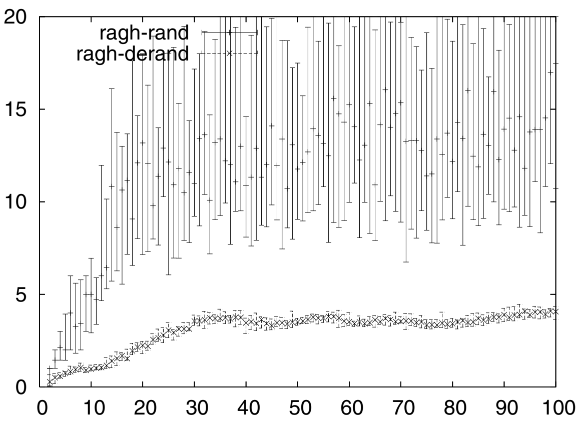
4.7 Discussion. We now summarize and interpret the results of the experiments we described in the previous subsections.

Price of Constraints. Contrary to what one might expect, the results show that adding a single hard constraint does not make rounding significantly harder, neither in terms of running times nor rounding errors.

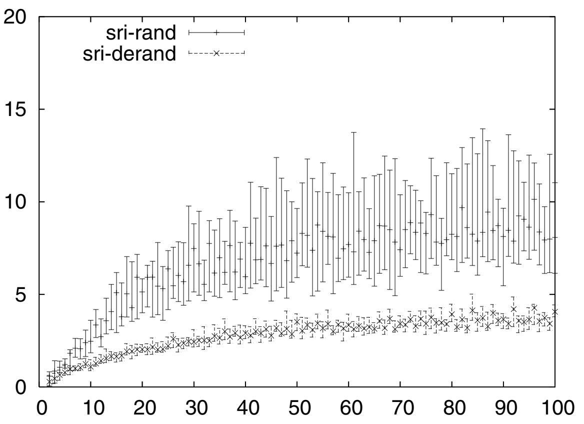
For rounding errors, often the opposite is true. As Table 2 demonstrates, in particular in the presence of constraints that involve many variables, an additional cardinality constraint involving all variables leads to rounding errors reduced by up to 38%. Note that a few such large constraints suffice to show this effect, as demonstrated by the rounding problem stemming from the higher-dimensional discrepancy problem.

Concerning running times, for full bit-length numbers the two bitwise approaches lose about a factor of 15 over Raghavan's approaches (without hard constraint), whereas Srinivasan's randomized approach and our derandomization thereof are about 50% slower. For all randomized versions, this is not too important since all algorithms round a million numbers in less than two seconds.

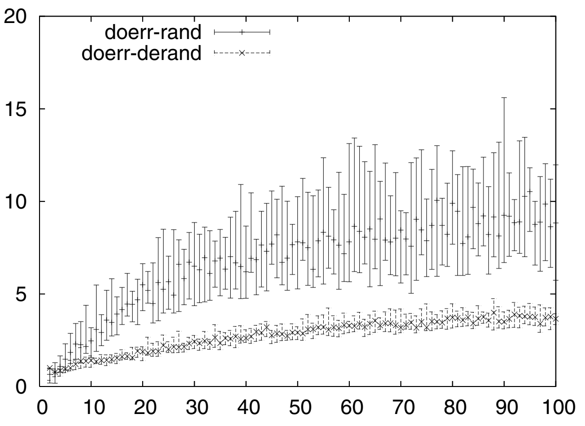
Tree-based or Bitwise? With two substantially different methods to generate randomized roundings



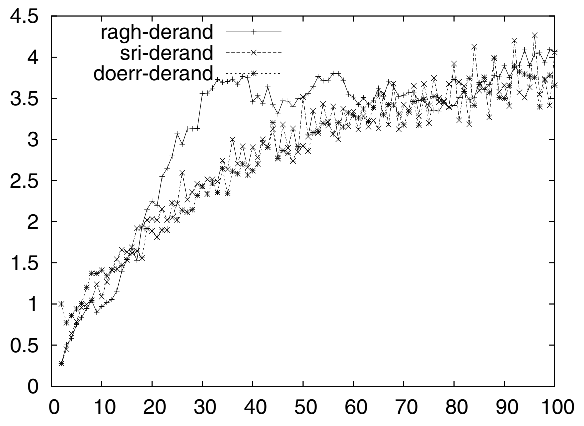
a) Classical rounding



b) Srinivasan's method



c) Bitwise rounding



d) Derandomizations only

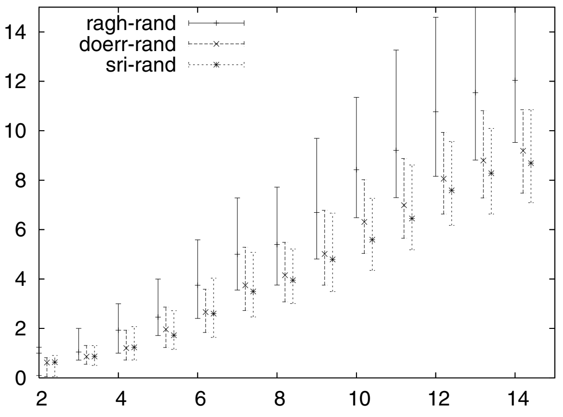
Figure 2: Rounding errors in low-discrepancy point set instances of dimension $d = 2$. The number k of subdivisions ranges from 2 to 100. In a) to c), each of the three algorithms in its randomized and derandomized version is applied. The number of points in the discrepancy problem ranges from 90 to 110. Depicted are the ranges of the (maximum) rounding errors for each of these 21 instances together with the average value. In d), the averages are compared for the three derandomizations.

respecting hard constraints, the important question naturally is which one is the better. For the randomized versions, surprisingly the results lead to almost no distinction. At most Fig. 3 shows a few-percent lead of the tree-based approach in average, with the invervals in which the errors lie mostly overlapping.

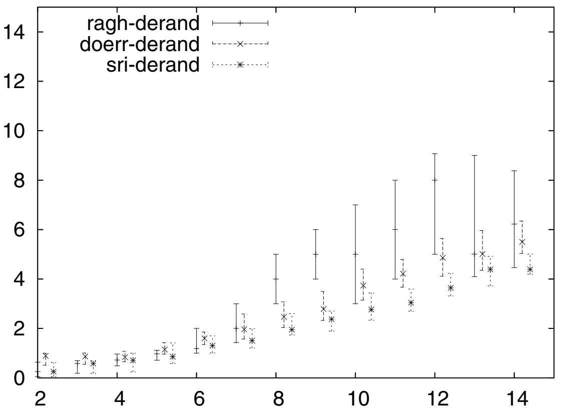
For the derandomized versions, our derandomization of Srinivasan's tree-based approach typically is clearly superior to the bitwise one. As a comparison of Fig. 1 a) and b) versus c) and d) shows, this stems from rounding each bit-level separately without taking care for previously accrued errors. The lead of the tree-based approach was strongest for dense random instances (41.4% vs. 57.6% of the error caused by independent randomized rounding), but less significant for the discrepancy instances. For the 2-dimensional case, both derandomizations seem to be equally good in average, with the tree-based method showing greater variation in the quality from instance to instance (cf. Fig. 2 d)).

It should be noted that the derandomization of Srinivasan's approach can produce much worse results depending on the tree that is used. The theoretical bounds proven in [GKPS06] hold for any tree, so no preference is expressed there. However, the results in Section 4.5 show that using the natural stack-type tree (pairing the variables one after the other) can produce five times as large rounding errors.

Randomized or Derandomized? All experiments show that the derandomized algorithms achieve smaller rounding errors. For the 2-dimensional matrix rounding



a) Random methods



b) Derandomized methods

Figure 3: Rounding errors in low-discrepancy instances of dimension 2 to 14 with $k = 2$ subdivisions. The number of points varies from 90 to 110. Depicted are the ranges of the discrepancies observed together with the average values. For the three randomized algorithms in a), all numbers given are the averages over 10 runs.

Method	$[0,1]$ vector, dense	$[0,1]$ vector, sparse	constant 1/2, dense	constant 1/2, sparse	$d = 2$, $90 \le k \le 100$	$12 \le d \le 14$ $k = 2$
ragh-rand	100%	100%	100%	100%	100%	100%
sri-rand	86.1%	98.8%	86.2%	97.7%	61.9%	70.9%
doerr-rand	85.8%	98.1%	86.2%	97.9%	62.5%	74.7%
ragh-derand	49.2%	45.2%	47.5%	42.5%	26.7%	56.3%
sri-derand	41.4%	45.9%	44.8%	49.5%	25.2%	35.9%
doerr-derand	57.6%	66.3%	43.0%	49.6%	25.5%	44.7%
Theory bound	284%	234%	232%	192%	259%	324%

Table 2: Adjusted rounding errors for the different instances, averaged over different problem sizes. For the random instances, sizes vary from 8,000 to 10,000. For each instance size, the observed discrepancies are rescaled to give 100% for ragh-rand.

problem and Raghavan's method, we observe nearly a factor four advantage of the derandomization. The superiority of the derandomizations naturally stems from the greediness of these approaches. Nevertheless, we did not expect to gain that much. Depending on the application, this might be worth the extra computation effort.

This finding seems to be mostly independent from having hard constraints. To the best of our knowledge, however, it has not been published yet even in the classical setting without constraints. Note that repeatedly generating a randomized solution until a good one is found will give inferiour results. The problem is that computing the rounding error is quite costly (about 1s for the 10k×10k instances in Table 4.2). Hence doing so between 50 and 100 times is as costly as running a derandomized algorithm, but is unlikely to find an equally good solution.

5 Conclusions

In this work, we regarded the problem of generating randomized roundings with a single cardinality constraint and the corresponding derandomizations from the algorithm engineering perspective. This is motivated by recent results of Srinivasan and the first author. Guided and motivated by our experimental findings, we improved the bitwise derandomization given in [Doe06] and developed the first derandomization of Srinivasan's tree-based randomized approach [Sri01]. We then experimentally compared the resulting (and previous) algorithms. To the best of our knowledge, this is the first experimental investigation of randomized rounding even in the case of no hard constraints.

Both for random instances and more structures ones

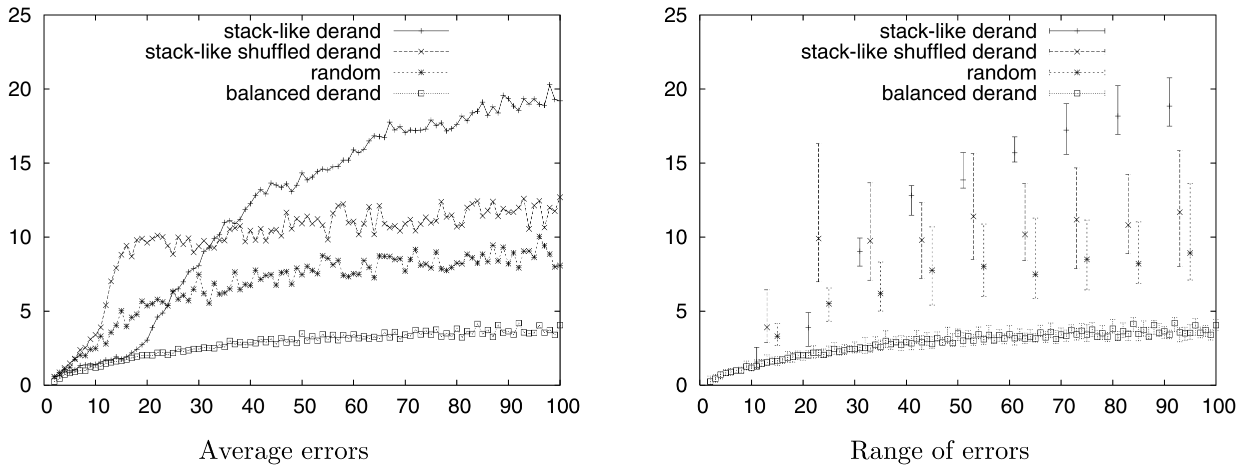


Figure 4: Influence of the shape of the tree used in Srinivasan's derandomization. Shown are discrepancies occurring for low-discrepancy point set instances in dimension $d = 2$ when using different tree shapes in the derandomization.

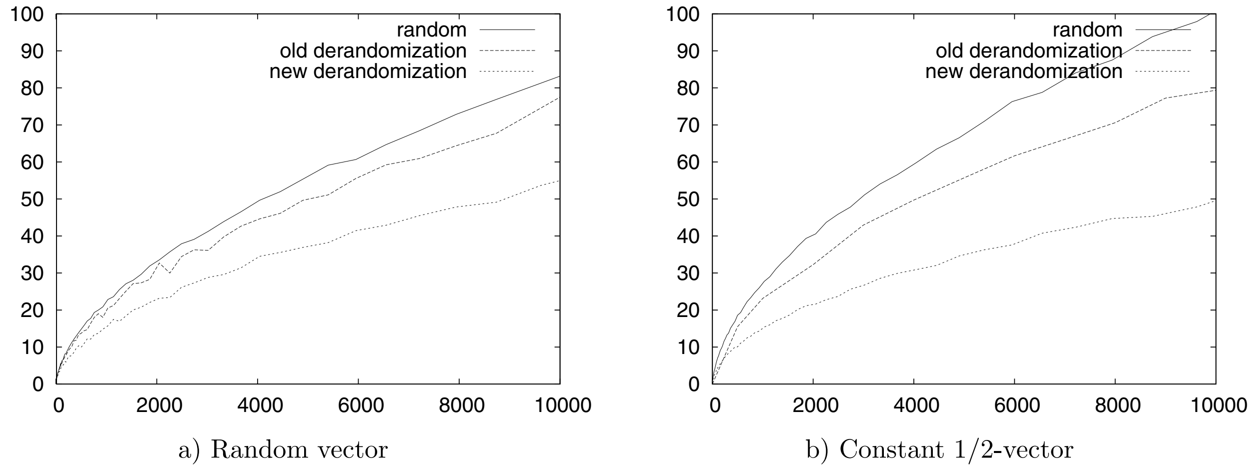


Figure 5: Rounding errors produced the bitwise rounding approach in its randomized version, with the old derandomization and the new derandomization of Section 3.2. The rounding problems used for this comparison are dense random matrices as described in Section 4.3 of dimension up to 10,000 together with vectors a) taking random entries in $[0, 1]$ and b) having all entries equal $\frac{1}{2}$.

stemming from modelling a geometric discrepancy problem, our experiments show that adding a single cardinality constraint reduces the rounding errors (by up about 40%) without seriously increasing the running times. In general, the derandomized tree-based approach is superior to the derandomized bitwise one, while the randomized versions produce very similar rounding errors. When implementing the derandomized tree-based approach, however, the choice of the tree used can make a huge difference (a factor of up to five in the rounding errors).

References

[AS99] 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)*, volume 1610 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 1999.

[AS00] 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*, volume 1879 of *Lecture Notes in Computer Science*, pages 32–41. Springer, 2000.

[AS04] 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.

[AFK02] 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.

[AKOT03] T. Asano, N. Katoh, K. Obokata, and T. Tokuyama. Matrix rounding under the L_p-discrepancy measure and its application to digital halftoning. *SIAM J. Comput.*, 32:1423–1435, 2003.

[CCE85] B. D. Causey, L. H. Cox, and L. R. Ernst. Applications of transportation theory to statistical problems. *Journal of the American Statistical Association*, 80:903–909, 1985.

[CN06] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36:498–515, 2006.

[DR89] G.S. Ditlow and P. Raghavan. Timing-driven partitioning of PLAs. Technical Report 31, IBM Technical Disclosure Bulletin, 1989.

[Doe04] B. Doerr. Nonindependent randomized rounding and an application to digital halftoning. *SIAM Journal on Computing*, 34:299–317, 2004.

[Doe05] B. Doerr. Roundings respecting hard constraints. In *Proceedings of STACS'05*, volume 3404 of *Lecture Notes in Computer Science*, pages 617–628, 2005.

[Doe06] B. Doerr. Generating randomized roundings with cardinality constraints and derandomizations. In *Proceedings of STACS'06*, volume 3884 of *Lecture Notes in Computer Science*, pages 571–583, 2006.

[Doe07] B. Doerr. Randomly rounding rationals with cardinality constraints and derandomizations. In *Proceedings of STACS'07*, volume 4393 of *Lecture Notes in Computer Science*, pages 441–452, 2007.

[DG06] B. Doerr and M. Gnewuch. Construction of low-discrepancy point sets of small size by bracketing covers and dependent randomized rounding. In A. Keller, S. Heinrich and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 299–312, 2008. Springer-Verlag.

[GHK$^+$03] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. In *Proceedings of ICALP'03*, volume 2719 of *Lecture Notes in Computer Science*, pages 164–175, 2003.

[GKPS02] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *Proceedings of FOCS'02*, pages 323–332, 2002.

[GKPS06] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53:324–360, 2006.

[GKR$^+$03] V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J. Comput. Syst. Sci.*, 67:473–496, 2003.

[Hro05] J. Hromkovič. *Design and analysis of randomized algorithms. Introduction to design paradigms.* Springer-Verlag, Berlin, 2005.

[LA08] D. L. Lau and G. R. Arce. *Modern Digital Halftoning.* CRC Press. 2008.

[MNR97] R. Motwani, J. Naor, and P. Raghavan. Randomized approximation algorithms in combinatorial optimization. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 447–481. PWS Publishing Co., Boston, MA, U.S.A, 1997.

[NRT87] A.P-C. Ng, P. Raghavan, and C.D. Thompson. Experimental results for a linear program global router. *Computers and Artificial Intelligence*, 6:229–242, 1987.

[Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.*, 37:130–143, 1988.

[RT87] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[RT91] P. Raghavan and C. D. Thompson. Multiterminal global routing: a deterministic approximation scheme. *Algorithmica*, 6:73–82, 1991.

[Sri01] A. Srinivasan. Distributions on level-sets with applications to approximations algorithms. In *Proceedings of FOCS'01*, pages 588–597, 2001.