The combinatorics of hidden diversity

Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The combinatorics of hidden diversity ☆

Juan Garay [a,*,1], David Johnson [2], Aggelos Kiayias [b,c,3], Moti Yung [d,e]

[a] Texas A&M University, United States of America
[b] University of Edinburgh, United Kingdom of Great Britain and Northern Ireland
[c] IOHK, Hong Kong
[d] Google Inc, United States of America
[e] Columbia University, United States of America

ARTICLE INFO

Article history:
Received 31 January 2019
Accepted 15 July 2019
Available online xxxx

In Memoriam. This paper is dedicated to Danny Breslauer and to co-author David S. Johnson

Keywords:
Balls and buckets
Combinatorial analysis
Secure multi-party computation
Hidden diversity

ABSTRACT

In this paper, we study the following "balls in buckets" problem. Suppose there is a sequence $B_1, B_2, \ldots, B_n$ of buckets having integer sizes $s_1, s_2, \ldots, s_n$, respectively. For a given *target fraction* α, $0 < \alpha < 1$, our goal is to sequentially place balls in buckets until at least $\lceil \alpha n \rceil$ buckets are full, so as to minimize the number of balls used, which we shall denote by $OPT_\alpha(I)$ for a given instance I.

If we knew the size of each bucket, we could obtain an optimal assignment, simply by filling the buckets in order of increasing size until the desired number had been filled. Here we consider the case where, although we know n and α, we do not know the specific bucket sizes s_i, and when we place a ball in bucket B_j, we only learn whether or not the bucket B_j is now full.

We study what can be done under four variants of incomplete information:

1. We know nothing at all about the bucket sizes;
2. we know the *maximum* bucket size;
3. we know the sizes $s_1 \leq s_2 \leq \cdots \leq s_m$ that occur in the instance; and
4. we know the *profile* of the sizes: the size list as above, and, for each size, s_i, the number k_i of buckets that have that size,

providing both algorithmic performance guarantees and lower bounds on the best that any algorithm can achieve.

The game above showcases the rich variety of interesting combinatorial and algorithmic questions that this setup gives rise to, and in addition has applications in an area of cryptography known as *secure multi-party computation*, where taking over ("corrupting") a party by an adversary has a cost, and where a *hidden diversity*—corresponding to lack of information on the amount of computational resources the adversary should invest to corrupt a participant—translates into robustness and efficiency benefits.

☆ This manuscript presents the expanded treatment of the combinatorial aspects of [4].

* Corresponding author.
E-mail addresses: garay@tamu.edu (J. Garay), akiayias@inf.ed.ac.uk (A. Kiayias), moti@cs.columbia.edu (M. Yung).

[1] Work partly done while the author was at AT&T Labs – Research.

[2] Work done at AT&T Labs – Research.

[3] Research supported by ERC project CODAMODA no. 259152.

https://doi.org/10.1016/j.tcs.2019.07.016
0304-3975/

1. Introduction

We start by describing the motivating application.

1.1. Resource-based corruptions in cryptography

The notion of *computing in the presence of an adversary* which controls or gets access to parts of the system is at the heart of modern cryptography. This paradigm has given rise to cryptosystems and protocols which achieve general tasks in the presence of powerful adversaries. A prime example of the paradigm are "completeness theorems" which show that a distributed cryptographic protocol exists where an adversary controlling any minority of the parties, cannot prevent the secure computation of any efficient functionality defined over their inputs [5]. Similar secure multi-party computation (MPC) results hold over secure channels (and no additional cryptography) with an adversary controlling less than a third of the parties [1,2]. Furthermore, these results are tight in the sense that when the adversary controls more parties there are functionalities that cannot be securely implemented.

In the works thus far however, the corruption of a party has been viewed as a simple, uniform and atomic operation where the adversary decides to corrupt a party and this party gets corrupted. The only limit the adversary has is on the number of individual party's corruptions and restrictions on when and how it can corrupt.

In this paper we are motivated by the fact that different players may require different resources to get corrupted. Indeed, the price paid for penetrating one organization, measured for example by the amount of money that a corrupt employee might have demanded, may differ from the cost of getting into another organization. Another example of a more cryptographic nature is when two organizations employ different password policies, or utilize VPN's relying on cryptosystems of different strengths. Taking such real-world considerations into account gives rise to the notion of *resource-based corruption*, where the adversary must invest certain resources in order to corrupt a party.

In a setting where the adversary has full information about the system, resource-based corruptions provide no fundamental divergence from the logic of the standard corruption model. The adversary, given an initial corruption budget, will target the largest subset of the system that it can afford and if such subset if large enough, the disruption of various security properties will ensue (as it follows from, e.g., [3]). The interesting case thus arises when the parties act on the system "anonymously" from the point of view of the adversary, in the sense that the association of parties' names and the individual corruption resources they require remains hidden. In this way, it seems plausible that a portion of the corruption budget will have to be wasted to learn the system configuration. The motivation of this paper is thus to investigate the quantitative effect of this type of *hidden diversity* in the context of resource-based corruptions.

1.2. A balls-and-buckets problem with incomplete information

The problem the adversary faces can be abstracted as the following combinatorial game. The adversary is given a number of balls ("corruption tokens") and is faced with a sequence of buckets with the objective to fill a certain fraction of them. While it knows all the bucket sizes, it does not know their correspondence to the individual buckets in the sequence. If OPT is the minimum number of balls required to fill a given percentage of the buckets had the adversary been privy to the correspondence between the buckets and their sizes, how many balls as a function of OPT would be required if such hidden diversity is provided? We analyze this setting as "the combinatorics of hidden diversity" and show a number of results concerning the initial partial knowledge given to the adversary and the strategies it can apply. The analysis leads to algorithms (corruption strategies) and lower bounds (corruption impossibilities) under various cases of partial knowledge and size parameters.

Importantly, we prove in particular that the "value of hidden diversity" can be *unbounded!* Specifically, for any B there are ways to choose buckets' sizes so that the required resources needed by the adversary to reach its target would be $B \cdot OPT$, i.e., an arbitrarily high inflation of the required adversarial budget.

In more detail, in this paper we study the following "balls in buckets" problem. Suppose there is a sequence $B_1, B_2, \ldots, B_n$ of buckets having integer sizes $s_1, s_2, \ldots, s_n$, respectively. For a given *target fraction* α, $0 < \alpha < 1$, our goal is to sequentially place balls in buckets until at least $\lceil \alpha n \rceil$ buckets are full, so as to minimize the number of balls used, which we shall denote by $OPT_\alpha(I)$ for a given instance I. In the language of our motivating application, the combinatorial game captures an adversary trying to corrupt an $\lceil \alpha \cdot n \rceil$ number of players. For convenience, sometimes in our exposition below we write from the point of view of the adversary.

If we knew the size of each bucket, we could obtain an optimal assignment, simply by filling the buckets in order of increasing size until the desired number had been filled. Here we consider the case where, although we know n and α, we do not know the specific bucket sizes s_i, and when we place a ball in bucket B_j, we only learn whether or not the bucket B_j is now full. (If a bucket has size 0, then it is full to begin with, and we know this before we start placing balls.)

We study what can be done under four variants of incomplete information ("adversarial ignorance"). In order of increasing knowledge, these are:

1. We know nothing at all about the bucket sizes [*No-Information*].
2. We know the maximum bucket size [*Max-Only*].

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

3. We know the sizes $s_1 \leq s_2 \leq \cdots \leq s_m$ that occur in the instance [*Sizes-Only*].
4. We know the *profile* of the sizes: the size list as above, and, for each size, s_i, the number k_i of buckets that have that size [*Known-Profile*].

We provide both algorithmic performance guarantees and lower bounds on the best that any algorithm can achieve. The algorithmic results, although positive in the ball-packing context, are actually negative when viewed in the context of our cryptographic application, as they show the power of a would-be corrupter. Similarly, our lower bounds can be viewed as positive results for our cryptographic application, corresponding to proofs of security.

We also note that all the algorithms we study apply in the No-Information, Max-Only, or Sizes-Only situations, but match, to varying extents, our lower bounds, which all hold even in the Known-Profile case. This suggests that the general algorithmic advantage to knowing the full profile may not be substantial.

1.3. Applications

Going back to our motivating application of secure multi-party computation (MPC), our results imply that hidden diversity in the context of resource-based corruptions enables us to get around impossibility results. In more detail, the impossibility result of MPC states that when the adversary has corruption resources that exceed those needed to control a *minority* of players, many functionalities have to be given up. The implications of our results (refer to [4] for more detailed exposition and a formal proof) are as follows. Let OPT be the optimal corruption budget for which the completeness of MPC is violated. Assuming hidden diversity, there exist configurations such that:

- For any B, the completeness of MPC holds against any adversary with less than $B \cdot OPT$ corruption budget assuming a sufficient number n of players (where $n = \Omega(\log(\frac{1}{\epsilon}) \cdot B)$, and ϵ is the probability of error).
- The above assumes the hardness of individual corruptions is not bounded. If, on the other hand, a bound M is imposed, the completeness of MPC holds against any adversary with less than $\sim (\frac{\sqrt{M}}{\log(\frac{1}{\epsilon})}) \cdot OPT$ corruption budget assuming $n \geq \sqrt{M}$.

In addition, we provide evidence that the above results are essentially tight. For example, for the second formulation above, when the adversary's budget reaches an amount $\sim \sqrt{M} \cdot OPT$, there is a strategy that always corrupts half the players.

Another way to exploit hidden diversity is to improve the efficiency of the implementation of MPC (as opposed to increasing the corruption budget the protocol can tolerate). Our results also imply that, assuming the same OPT corruption budget, hidden diversity forces the corruption threshold to drop from 1/2 to 1/3, in turn allowing the use of much more efficient (information-theoretic) MPC protocols. Again, refer to [4] for additional details.

To summarize, the above implications demonstrate that there exist settings of player diversity that can be quite beneficial from a security standpoint if the diversity is appropriately hidden.

1.4. Organization of the paper

We begin in Section 2 with the case where there are no restrictions on the input (and little in the way of good algorithmic performance), and then consider in Section 3 the effects of placing reasonable restrictions on the number of different item sizes, the number of buckets, and the maximum bucket size. We then consider in Section 4 alternative objective functions, such as filling as many buckets as possible for a given number of balls (in the presence of hidden diversity). We conclude in Section 5 with a summary of our results, on-going work, and open questions.

2. Unrestricted instances

Our first observation is that, even when we know the full profile, no fixed multiple of the optimal number of balls suffices to guarantee even a small probability of success if there are no restrictions on the input.

Theorem 2.1. *For any α, $0 < \alpha < 1$, and constants $B > 1$ and $\epsilon > 0$, there exists an instance I such that, assuming we first use a random permutation to relabel the buckets, any algorithm that knows nothing more than the profile of the instance, and has fewer than $B \cdot OPT_\alpha(I)$ balls, has probability less than ϵ of filling $\lceil \alpha n \rceil$ buckets.*

Proof. Our instance will have $n = \left\lceil ((2B/\epsilon) + 1)/(1 - \alpha) \right\rceil$ buckets. Of these buckets, the $\lceil \alpha n \rceil$ with smallest capacity will consist of $\lceil \alpha n \rceil - 1$ buckets of size 1 and one bucket of size $X = \lceil \alpha n \rceil + 1$. An optimal solution will thus be simply to fill these $\lceil \alpha n \rceil$ buckets, requiring $OPT_\alpha(I) = 2\lceil \alpha n \rceil$ balls. The remaining $n - \lceil \alpha n \rceil$ buckets will all have capacity $M = 2B\lceil \alpha n \rceil$.

Now let A be any algorithm that has access to fewer than $B \cdot OPT_\alpha(I) = 2B\lceil \alpha n \rceil$ balls. Clearly the algorithm cannot fill any of our size-M buckets, so it will have to fill all the $\lceil \alpha n \rceil$ buckets of the optimal solution, and, in particular, it must fill the bucket of size X. Consider the buckets of size X or greater, and let m be the number of such buckets. Given

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

the hidden depth of the buckets, the only way the algorithm can tell whether one of these buckets is the one that has size X is to place X balls in it. So it can test at most $2B-1$ buckets. Given that the identities of the buckets have been randomly permuted, the probability that the algorithm succeeds in finding the bucket of size X, and hence has any chance of filling $\lceil \alpha n \rceil$ buckets, is no more than $(2B-1)/m$, which will be less than ϵ so long as $m > (2B-1)/\epsilon$. But now note that

$$m \geq n - \lceil \alpha n \rceil \geq n(1-\alpha) - 1 = \left\lceil \frac{\frac{2B}{\epsilon}+1}{1-\alpha} \right\rceil (1-\alpha) - 1 \geq \frac{2B}{\epsilon},$$

as desired. □

Using essentially the same examples, parameterized by n instead of ϵ, we get the following alternative statement of the theorem.

Corollary 1. *For any α, $0 < \alpha < 1$, and constant $B > 1$, there exists a sequence of instances I_n such that, assuming we first use a random permutation to relabel the buckets, any algorithm that knows nothing more than the profile of the instance, and has fewer than $B \cdot OPT_\alpha(I_n)$ balls, has $O(1/n)$ probability of filling $\lceil \alpha n \rceil$ buckets.*

By setting $\epsilon = 1/2$ and replacing "B" by "$2B$" in the proof of Theorem 2.1, we obtain the following.

Corollary 2. *For any α, $0 < \alpha < 1$, and constant $B > 0$, there exists an instance I of our balls-in-buckets problem such that, assuming we first use a random permutation to relabel the buckets, the expected number of balls used by any algorithm that knows nothing more than the profile of the instance is at least $B \cdot OPT(I)$ balls.*

The proof of Theorem 2.1 also implies that, if we are willing to settle for small penalties and non-minuscule ϵ, neither n nor the maximum bucketsize need be impractically large. For instance, suppose $\alpha = 1/2$ and we only want hidden diversity to cause us a factor-of-two ball penalty with probability 1/3. Then the relevant instance is $I_{2,1/3}$, for which $n = \lceil (2 \cdot 2 \cdot 3 + 1)/(1/2) \rceil = 26$ and $M = \lceil n/2 \rceil + 1 = 14$.

In the proof of Theorem 2.1, our constructions only let us claim that, for fixed B and α, the error probability is $O(1/n)$. Ideally, we would like error probabilities that decline exponentially in n. A modification of our constructions allows us to do this.

Theorem 2.2. *For any α, $0 < \alpha < 1$, and constant $B > 1$, there exists a constant $a < 1$ and instances I_n, $n > 8B/(1-\alpha)$, such that, assuming we first use a random permutation to relabel the buckets, any algorithm that knows nothing more than the profile of the instance, and has fewer than $B \cdot OPT_\alpha(I)$ balls, has probability less than a^n of filling $\lceil \alpha n \rceil$ buckets.*

Proof. For a c, $0 < c \leq \alpha$, to be specified later, instance I_n will have $\lceil cn \rceil$ buckets of size $\lceil \alpha n \rceil + 1$, $\lceil \alpha n \rceil - \lceil cn \rceil$ buckets of size 1, and $n - \lceil \alpha n \rceil$ buckets of size $(\lceil cn \rceil + 2)B\lceil \alpha n \rceil$. An optimal solution will consist of the $\lceil \alpha n \rceil$ smallest buckets, and will have total size $(\lceil cn \rceil + 1)\lceil \alpha n \rceil$. This implies that we cannot afford to fill any of the largest buckets if we are to use fewer than $B \cdot OPT(I_n) = B(\lceil cn \rceil + 1)\lceil \alpha n \rceil$ balls, and any algorithm that fills $\lceil \alpha n \rceil$ buckets must find and fill all the buckets of size $\lceil \alpha n \rceil + 1$.

But now note that the only way the algorithm can tell whether a bucket has this size or is one of the larger ones is to place $\lceil \alpha n \rceil + 1$ balls in the bucket. Thus, if our algorithm is to use no more than $B(\lceil cn \rceil + 1)\lceil \alpha n \rceil$ balls, it can test no more than $B(\lceil cn \rceil + 1)$ buckets with size exceeding 1, of which there are $n - \lceil \alpha n \rceil + \lceil cn \rceil$. The probability of finding all $\lceil cn \rceil$ of the mid-size bucket is then at most

$$\binom{n - \lceil \alpha n \rceil}{B(\lceil cn \rceil + 1) - \lceil cn \rceil} \Big/ \binom{n - \lceil \alpha n \rceil + \lceil cn \rceil}{B(\lceil cn \rceil + 1)}$$

Setting $X = n - \lceil \alpha n \rceil$ and $Y = B(\lceil cn \rceil + 1) - \lceil cn \rceil$, this is equal to

$$\binom{X}{Y} \Big/ \binom{X + \lceil cn \rceil}{Y + \lceil cn \rceil} = \frac{(Y+1)(Y+2)\cdots(Y+\lceil cn \rceil)}{(X+1)(X+2)\cdots(X+\lceil cn \rceil)},$$

which we will be able to argue is sufficiently small if we can chose $c > 0$ such that, say, $Y + \lceil cn \rceil \leq (X + \lceil cn \rceil)/2$. This requires that $B\lceil cn \rceil + B < (n - \lceil \alpha n \rceil + \lceil cn \rceil)/2$, or $(2B-1)\lceil cn \rceil \leq n - \lceil \alpha n \rceil - 2B$, which will be true so long as $(2B-1)(cn+1) \leq n(1-\alpha) - 1 - 2B$, or $(2B-1)(cn) \leq n(1-\alpha) - 4B$, or $c \leq (1-\alpha)/(2B-1) - 4B/(n(2B-1))$. By our assumption that $n > 8B/(1-\alpha)$, this means that all we need for $Y + \lceil cn \rceil < (X + \lceil cn \rceil)/2$ is that $c \leq (1-\alpha)/(4B-2)$. By our construction, we also need $c \leq \alpha$, so it suffices to set $c = \min\{\alpha, (1-\alpha)/(4B-2)\}$.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

Given that $Y + \lceil cn \rceil < (X + \lceil cn \rceil)/2$, we then have that $(Y + i)/(X + i) \leq 1/2$ for $1 \leq i \lceil cn \rceil$, and hence

$$\frac{(Y+1)(Y+2)\cdots(Y+\lceil cn \rceil)}{(X+1)(X+2)\cdots(X+\lceil cn \rceil)} < \left(\frac{1}{2}\right)^{\lceil cn \rceil} \leq \left(\frac{1}{2}\right)^{cn}$$

which is less than a^n for $a = (1/2)^c < 1$, as desired. □

From the above results, we know that, if we place no restrictions on our instances, then hidden diversity imposes a penalty factor that is unbounded. Thus, the only situations in which the performance penalty for hidden diversity can be bounded will be ones in which we restrict the instances, for example by bounding n or the maximum bucket size, or by limiting the number of distinct bucket sizes. Note, however, that the proofs of Theorems 2.1 and 2.2 imply that we cannot avoid unbounded performance penalty as soon as there are three or more distinct sizes.

3. Restricted instances and no-information algorithms

In what follows, we shall assume that the greatest common divisor Δ of the bucket sizes in our instances is 1 (as for instance will happen when there are buckets of size 1 themselves). It is easy to see that if $\Delta > 1$, then any algorithm can be modified to place Δ balls at once and will have the same relative performance in comparison to an optimal algorithm (and that there is no advantage to doing anything differently). We shall denote the maximum bucket size under this assumption as M.

In what follows, we shall consider not only lower bounds on the performance penalties for *all* algorithms, but also how well specific natural algorithms perform. When talking about specific algorithms, we will be interested in two worst-case measurements. For a given α, an algorithm A, and an n-bucket instance I, let $A_\alpha(I)$ be the number of balls used by algorithm A to fill $\lceil \alpha n \rceil$ buckets in instance I. Suppose $0 < \alpha < 1$, $M \geq 1$, and $n > 0$. If dA is a deterministic algorithm, define

$$R^{dA}_{\alpha,M,n} = \max\left\{ \frac{A_\alpha(I)}{OPT_\alpha(I)} : I \text{ is an instance with } n \text{ buckets, target fraction } \alpha, \text{ and } s_m \leq M \right\}$$

If rA is a randomized algorithm, define

$$R^{rA}_{\alpha,M,n} = \max\left\{ \frac{E[A_\alpha(I)]}{OPT_\alpha(I)} : I \text{ is an instance with } n \text{ buckets, target fraction } \alpha, \text{ and } s_m \leq M \right\}$$

Then define

$$R^{A}_{\alpha,\infty,n} = \limsup_{M\to\infty} R^{A}_{\alpha,M,n}$$
$$R^{A}_{\alpha,M,\infty} = \limsup_{n\to\infty} R^{A}_{\alpha,M,n}$$

In this section, we first discuss a sequence of algorithms and how they perform when either n or M is bounded, including a hybrid of the two tailored to the case when neither is bounded, but we are guaranteed that there are at most two distinct sizes. We begin with algorithms that operate in the No-Information domain and, are building blocks for later, more effective algorithms that follow and exploit the additional information available in our other cases. We present our lower bound proofs immediately following the algorithmic results they match.

3.1. *The* Fill *algorithm*

Algorithm "Fill" works by repeatedly choosing an empty bucket and adding balls to that bucket until it is full. Let $dFill$ represent any version of the algorithm that chooses the next bucket according to some deterministic policy, and $rFill$ be the version that chooses the next bucket randomly from the set of currently empty buckets, with all choices equally likely. It is easy to see that the following holds.

Theorem 3.1.

1. *For all* $M > 0$,

$$R^{dFill}_{\alpha,M,\infty} = \begin{cases} M, & 0 < \alpha \leq \frac{1}{2} \\ \frac{1-\alpha}{\alpha}M + 2 - \frac{1}{\alpha}, & \frac{1}{2} < \alpha < 1 \end{cases}$$
$$R^{rFill}_{\alpha,M,\infty} = \alpha + (1-\alpha)M, \quad 0 < \alpha < 1$$

2. *For all* α, $0 < \alpha < 1$, *and all* $n > 1/(1-\alpha)$,

$$R^{dFill}_{\alpha,\infty,n} = R^{rFill}_{\alpha,\infty,n} = \infty.$$

Note that for $\alpha = 1/2$, claim (1) reduces to $R^{dFill}_{\alpha,M,\infty} = M$ and $R^{rFill}_{\alpha,M,\infty} = (M+1)/2$.

Proof. For the cases where M is bounded, it is easy to see that a worst-case example I_n for n buckets consists of $\lceil \alpha n \rceil$ buckets of size 1 and the rest of the buckets having capacity M. The bounded-M result for $\alpha \leq 1/2$ is trivial. For $\alpha > 1/2$, $OPT_\alpha(I_n) = \lceil \alpha n \rceil$ and

$$dFill_\alpha(I_n) = (n - \lceil \alpha n \rceil)M + \lceil \alpha n \rceil - (n - \lceil \alpha n \rceil) = (n - \lceil \alpha n \rceil)M + 2\lceil \alpha n \rceil - n.$$

Dividing by $OPT_\alpha(I_n)$ and taking the limit as $n \to \infty$ gives the claimed bound. The result for $rFill$ follows from the fact that the expected size of a randomly chosen bucket is $\lceil \alpha n \rceil / n + (n - \lceil \alpha n \rceil)M/n$.

For the cases where n is bounded, we can use the same examples. The assumption that $n > 1/(1-\alpha)$ implies that $n > \lceil \alpha n \rceil$, so there will be at least one bucket with size M. Thus we have that in the worst-case, $dFill(I) \geq M$ for these instances, and the expected value of the solution generated by $rFill$ is at least M/n. Since $OPT_\alpha(I) = \lceil \alpha n \rceil < n$ and n is fixed, the claim follows. □

Thus both $dFill$ and $rFill$ can be unboundedly bad if M can be arbitrarily large, even if n is bounded, but have bounded worst-case behavior if M is bounded, even if n is unbounded.

3.2. *The* Lowest Level *algorithm*

A complementary algorithm is "Lowest Level" (LL), which has bounded worst-case behavior if *either* M or n is bounded. In this algorithm, as long as we haven't yet filled our quota of buckets, we place the next ball in an unfilled bucket that currently contains the fewest balls. Here let dLL denote any version of the algorithm where ties are broken in some deterministic fashion, and rLL denote the version where ties are broken randomly, with all choices equally likely. We then have

Theorem 3.2. *For all* α, $0 < \alpha < 1$,

1. *For all* $M \geq 2$, $R^{dLL}_{\alpha,M,\infty} = R^{rLL}_{\alpha,M,\infty} = \dfrac{1-\alpha}{\alpha}(M-1) + 1$.
2. *For all* $n > 1/(1-\alpha)$, $R^{dLL}_{\alpha,\infty,n} = R^{rLL}_{\alpha,\infty,n} = n - \lceil \alpha n \rceil + 1$.

(Note that for $\alpha = 1/2$, these claims reduce to $R^{dLL}_{\alpha,M,\infty} = R^{dLL}_{\alpha,M,\infty} = M$, the same as for $rFill$ and $dFill$, and $R^{dLL}_{\alpha,\infty,n} = R^{rLL}_{\alpha,\infty,n} = n/2 + 1$ (for even n) and $n/2 + 1/2$ (for odd n).

Proof. We first show the upper bounds for (1) and (2). Let I be an instance with $n > 1/(1-\alpha)$ buckets, and let s_k be the size of the largest bucket filled in an optimal solution. Note that $s_k/s_1 \leq M$. We will have

$$OPT_\alpha(I) \geq (\lceil \alpha n \rceil)s_1 + s_k - s_1 \geq \max(\alpha n, s_k).$$

As to the performance of our algorithms, and suppose there are y buckets of size s_k in an optimal placement. In this case, the algorithms will be finished as soon as they put the s_k'th ball into the y'th size-s_k bucket, leaving $n - \lceil \alpha n \rceil$ buckets with just $M-1$ balls. Thus

$$dLL_\alpha(I), rLL_\alpha(I) \leq OPT_\alpha(I) + (n - \lceil \alpha n \rceil)(s_k - 1)$$

and, hence, for $LL \in \{dLL, rLL\}$, we have

$$\begin{aligned} \frac{LL_\alpha(I)}{OPT_\alpha(I)} &\leq 1 + \min\left(\frac{1-\alpha}{\alpha}(s_k - 1), \left(n - \lceil \alpha n \rceil\right)\frac{s_k - 1}{s_k}\right) \\ &\leq 1 + \min\left(\frac{1-\alpha}{\alpha}(M-1), \left(n - \lceil \alpha n \rceil\right)\frac{M-1}{M}\right). \end{aligned}$$

The overall limiting upper bounds follow.

For the lower bounds, consider instances $I_{M,n}$ in which there are $\lceil \alpha n \rceil - 1$ buckets of size 1 and all the remaining buckets have size M. Here

$$OPT_\alpha(I_{M,n}) = \lceil \alpha n \rceil + M - 1 \le \alpha n + M,$$
$$dLL_\alpha(I) = rLL_\alpha(I) = OPT_\alpha(I_{M,n}) + (n - \lceil \alpha n \rceil)(M-1).$$

Fixing M and letting $n \to \infty$ yields the matching lower bound for (1), while fixing n and letting $M \to \infty$ yields the matching lower bound for (2). □

Note that, although LL dominates $Fill$ in terms of worst-case behavior when M is unbounded, the situation is different when M is fixed. Now $Fill$ actually can have better worst-case behavior in many situations. So long as $M > 1$ and $\alpha \le 1/2$, we have

$$R^{dFill}_{\alpha,M,\infty} = M \le ((1-\alpha)/\alpha)(M-1) + 1 = R^{dLL}_{\alpha,M,\infty}$$

with a strict inequality if $\alpha < 1/2$. The advantage of $R^{rFill}_{\alpha,M,\infty}$ over $R^{rLL}_{\alpha,M,\infty}$ in this situation is even more substantial. Both these algorithms, however, are beaten for fixed M by a Size-Only algorithm we shall be describing shortly.

In the meantime, we observe that the worst-case examples that we provided for both $Fill$ and LL only had two distinct bucket sizes. Interestingly, there is a No-Information algorithm that does much better than either on instances of this type.

3.3. A hybrid no-information algorithm for the 2-size case and matching lower bound

The algorithm, which we shall call "Hybrid1" ($H1$), works on the *assumption* that there are just two sizes, and so can be about as bad as $Fill$ when there are three sizes, in particular, having $R^{dH1}_{\alpha,\infty,n} = R^{rH1}_{\alpha,\infty,n} = \infty$ for any deterministic and randomized implementations $dH1$ and $rH1$ of $H1$. However such implementations do surprisingly well when there are just two sizes. The algorithm proceeds as in LL until we fill a bucket. Let d be the number of balls in this bucket. We add balls to all the remaining buckets to bring them up to level d. Then, while we have not yet filled more than αn buckets, repeatedly pick a bucket that is not full bucket and add balls to it until it is full. Note that if there are only two item sizes, all implementations of $H1$, randomized or deterministic, will require the same number of balls.

Theorem 3.3. *Suppose $0 < \alpha < 1$. Let H represent any deterministic or randomized implementation of $H1$. Then for any instance I with at most two distinct bucket capacities, we are guaranteed to have*

$$H_\alpha(I) \le (1/\alpha) OPT_\alpha(I).$$

Proof. First note that if $k = 1$, then the claim holds trivially, since $H_\alpha(I) = OPT_\alpha(I)$. Thus we may assume that $k = 2$. Suppose the two sizes are $s_1 < s_2$, and there are k_1 buckets of size s_1 and $k_2 = n - k_1$ buckets of size s_2. There are two cases.

(a) If $k_1 > \alpha n$, then $OPT_\alpha(I) = \lceil \alpha n \rceil s_1 > \alpha n s_1$ and $H_\alpha(I) \le n s_1 < (1/\alpha) OPT_\alpha(I)$, as desired.
(b) If $k_1 \le \alpha n$, then $OPT_\alpha(I) = k_1 s_1 + (\lceil \alpha n \rceil - k_1) s_2 = \lceil \alpha n \rceil s_1 + (\lceil \alpha n \rceil - k_1)(s_2 - s_1)$ and $H_\alpha(I) = n s_1 + (\lceil \alpha n \rceil - k_1)(s_2 - s_1) < (1/\alpha) OPT_\alpha(I)$, again as desired. □

Matching this guarantee, we have the following lower bounds.

Theorem 3.4. *Suppose $0 < \alpha < 1$.*

(A) For any deterministic algorithm dA, even one that knows the profile, there exist instances I_n for each $n > 1/(1-\alpha)$, with maximum size bounded independently of n, such that

$$\lim_{n\to\infty} \frac{dA_\alpha(I_n)}{OPT_\alpha(I_n)} = \frac{1}{\alpha}.$$

(B) For any α, $0 < \alpha < 1$ and any δ, $\alpha < \delta < 1$, there is a constant $a < 1$ and a sequence of instances I_n, such that the following holds for all sufficiently large n: Assume we first use a random permutation to relabel the buckets. Then any algorithm A that knows nothing more than the profile of the instance, and has no more than $\delta(1/\alpha) OPT_\alpha(I_n)$ balls, has probability less than a^n of filling $\lceil \alpha n \rceil$ buckets.

Proof. For (A), let I_n be an instance with n buckets, $\lceil \alpha n \rceil$ of them with size 1, the remainder with size $\lceil 1/\alpha \rceil$. Note that this implies that $OPT_\alpha(I_n) = \lceil \alpha n \rceil$. Our adversary arranges things so that the first $n - \lceil \alpha n \rceil$ buckets into which the algorithm places balls all have size $\lceil 1/\alpha \rceil$. If dA completes the filling $\lceil \alpha n \rceil$ buckets before it has placed a ball in more than $n - \lceil \alpha n \rceil$ buckets, then we will have $dA_\alpha(I_n) \ge \lceil \alpha n \rceil \cdot \lceil 1/\alpha \rceil$ and hence $dA_\alpha(I_n)/OPT_\alpha(I_n) \ge 1/\alpha$, as desired. If, on the other hand, dA places balls in $n - \lceil \alpha n \rceil$ or more buckets, then every one of the required $\lceil \alpha n \rceil$ filled buckets will entail the spending at least one ball in excess of the first balls into the $n - \lceil \alpha n \rceil$ size-$\lceil 1/\alpha \rceil$ buckets, for a total of at least n balls. This implies $dA_\alpha(I_n)/OPT_\alpha(I_n) \ge n/\lceil \alpha n \rceil$, with the desired limiting ratio.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

For (B), our desired instances I_n are constructed by increasing the size of the size-$\lceil 1/\alpha \rceil$ buckets in the examples used for (B) to n. Note that the number of balls we are given is at most $p = (\delta/\alpha)\lceil \alpha n \rceil$. Although this is greater than $\lceil \alpha n \rceil$, since by assumption $\delta > \alpha$, it is less than γn for $\gamma = (\delta + 1)/2 < 1$, so long as $n > \frac{2\delta}{(1-\delta)\alpha}$. In what follows, we assume that n is at least this large. Thus, our algorithm cannot afford to fill any of the size-n buckets and so must identify all $\lceil \alpha n \rceil$ size-1 buckets, which are in random locations.

First, suppose it makes its choices non-adaptively. In other words, it picks p buckets ahead of time, and places one ball in each, succeeding if it manages to have chosen all $\lceil \alpha n \rceil$ size-1 buckets. The probability of success will be

$$\frac{\binom{p}{\lceil \alpha n \rceil}}{\binom{n}{\lceil \alpha n \rceil}} = \frac{p(p-1)\cdots(p-\lceil \alpha n \rceil + 1)}{n(n-1)\cdots(n-\lceil \alpha n \rceil + 1)}.$$

But now, since $(p-i)/(n-i) \le p/n \le \gamma$ for $1 \le i \le p$, and $\lceil \alpha n \rceil < p$, this is at most $\gamma^{\lceil \alpha n \rceil} \le \gamma^{\alpha n}$. Thus the probability of success declines as a^n for $a = \gamma^\alpha < 1$, as desired.

So (B) holds if our algorithm is non-adaptive. But note that adaptivity yields no advantage for this particular task. At each step, all the as-yet-untouched buckets all have the same probability of being size-1, given our initial random permutation of the buckets. □

3.4. A hybrid Max-Only algorithm for the bounded-M case and matching lower bounds

We can do better than either Fill or SLL in the case when M is bounded. This involves a second hybrid algorithm, which we shall call Hybrid2 (H2). The algorithm works in the Max-Only situation and proceeds as follows. Suppose we are given an upper bound M on the maximum bucket size.

1. Let $M' = \lfloor \sqrt{M} \rfloor$.
2. Apply *LL* until either $\lceil \alpha n \rceil$ buckets are full or all non-full buckets contain M' balls.
3. While less than $\lceil \alpha n \rceil$ buckets are full, pick an bucket that is not full and add balls until it is filled to capacity.

Recall that in the Max-Only situation, we may assume that M *is* the maximum bucket size; the following result holds even if it is only an upper bound.

Theorem 3.5. *For $0 < \alpha < 1$, any deterministic implementation dH2 of H2, and any instance I with maximum bucket size no greater than M, we have*

$$\frac{dH2_\alpha(I)}{OPT_\alpha(I)} \le 1 + \frac{\sqrt{M}}{\alpha},$$

and consequently $R^{dH2}_{\alpha,M,\infty}$ obeys the same bound.

Proof. Let I be any instance, and suppose I contains n buckets. Let M'' be the largest capacity of a bucket filled in an optimal solution, and let x be the number of buckets of this size in the optimal solution. Then we have

$$OPT_\alpha(I) \ge \lceil \alpha n \rceil + x(M'' - 1) \ge \alpha n + x(M'' - 1)$$

There are two cases to consider. First, suppose that $M'' \le M'$. Then dH2 simply constructs an LL packing and we have

$$dH2_\alpha(I) \le OPT_\alpha(I) + (n - \lceil \alpha n \rceil)M'' \le OPT_\alpha(I) + n(1-\alpha)M''.$$

Consequently, since by assumption $M'' \le M' \le \sqrt{M}$, we have

$$\frac{dH2_\alpha(I)}{OPT_\alpha(I)} \le 1 + \frac{1-\alpha}{\alpha}M'' < 1 + \frac{\sqrt{M}}{\alpha},$$

as required.

Suppose, on the other hand, that $M'' > M'$. Then, by the definition of M', we must have $M'' > \sqrt{M}$. In this case, we will have

$$dH2_\alpha(I) \le OPT_\alpha(I) + x(M - M'') + n(1-\alpha)M'.$$

Consequently, given that $M'' > \sqrt{M} \ge M'$, we have

$$\frac{dH2_\alpha(I)}{OPT_\alpha(I)} < 1 + \frac{M - M''}{M'' - 1} + \frac{1-\alpha}{\alpha}M' < 1 + \frac{M}{M''} + \frac{1-\alpha}{\alpha}M' < 1 + \frac{\sqrt{M}}{\alpha},$$

again as required. □

We do not have precisely matching lower bound examples for Theorem 3.5. However, the next two theorems, to be proved below, show that no algorithm, deterministic or randomized, can do qualitatively better, even if it knows the full profile.

Theorem 3.6. *Suppose $0 < \alpha \le 1$. If dA is any deterministic algorithm that knows nothing more than the profile of the instance, then for all $M \ge \max\{8, 1/\alpha^2\}$,*

$$R^{dA}_{\alpha,M,\infty} \ge \frac{\min(.79, 1-\alpha)}{1+\alpha}\sqrt{M}$$

and

$$\lim_{M\to\infty} \frac{R^{dA}_{\alpha,M,\infty}}{\sqrt{M}} \ge \frac{1-\alpha}{1+\alpha}.$$

Proof. The proof is based on examples I_n with n buckets for n an integral multiple of $K = \lceil\sqrt{M}\rceil$. This will suffice since $R^{dA}_{\alpha,M,\infty}$ is defined as a lim sup. Let us first consider the bound that is claimed for $M \ge 8$, in which case we have $\sqrt{M}/\lceil\sqrt{M}\rceil > 0.79$.

Our instances have three sizes: 1, $K+1$ and M. Noting that, since $M \ge 1/\alpha^2$, we have $n/K \le \lceil\alpha n\rceil$, we have $\lceil\alpha n\rceil - n/K \ge 0$ buckets with size 1, n/K buckets with size $K+1$, and $n - \lceil\alpha n\rceil$ buckets with size M. An optimal placement fills all the buckets of the first two types, yielding

$$OPT_\alpha(I_n) = \lceil\alpha n\rceil + \frac{n}{K}(K) \le n(\alpha+1) + 1.$$

Our deterministic algorithm dA will confront an adversary that arranges things so that no bucket of size $K+1$ receives its second ball until all buckets of size M have received $K+1$ balls – recall that we may assume that if our algorithm puts a second ball in a bucket, it will keep placing balls there until it has reached $K+1$, the next valid size. There are two cases to consider.

(a) Suppose our algorithm ends up filling at least one bucket of size $K+1$. Then, by our assumption about the adversary, we must place at least $K+1$ balls in every bucket of size M. Thus, since $M - (\lceil\sqrt{M}\rceil + 1) \ge (\lceil\sqrt{M}\rceil + 1)$ whenever $M \ge 8$, we must in addition have enough balls to fill up all the buckets with size 1 or $K+1$, for a total of at least

$$\begin{aligned}\lceil\alpha n\rceil + n + (n - \lceil\alpha n\rceil)(K+1) &= 2n + (n - \lceil\alpha n\rceil)\lceil\sqrt{M}\rceil \\ &\ge 2n + (n(1-\alpha) - 1)\sqrt{M} \\ &\ge n(1-\alpha)\sqrt{M},\end{aligned}$$

the last inequality holding as soon as $n \ge \lceil\sqrt{M}\rceil/2$.

(b) Suppose our algorithm fills *none* of the size-$K+1$ buckets. Then it must fill at least n/K buckets of capacity M entirely, for a total number of balls at least

$$\frac{n}{\lceil\sqrt{M}\rceil}M \ge (.79)n\sqrt{M}.$$

Combining the conclusions of cases (a) and (b), we must use $\min(.79, (1-\alpha))n\sqrt{M}$ balls, and hence

$$\limsup_{n\to\infty} \frac{dA_\alpha(I_n)}{OPT_\alpha(I_n)} \ge \frac{\min(.79, (1-\alpha))n\sqrt{M}}{n(\alpha+1)+1} = \frac{\min(.79, (1-\alpha))\sqrt{M}}{\alpha+1},$$

as claimed for the case of $M \ge 8$.

For the second claimed bound, we note that it is already satisfied in case (a), while in case (b), it follows since $\lim_{M\to\infty}(\sqrt{M}/\lceil\sqrt{M}\rceil) = \sqrt{M}$. □

Theorem 3.7. *Suppose we are given α, $0 < \alpha < 1$, and $\epsilon > 0$, and let $\delta = (1-\alpha)/\big(1.25(1+\alpha)\max(6, \log(1/\epsilon))\big)$. For any $M \ge 8$, and $n > \max(2/(1-\alpha), 1/\alpha^2)$ that is divisible by $\lceil\sqrt{M}\rceil$, there is an instance I_n with n buckets and maximum bucket size M such that, assuming we first use a random permutation to relabel the buckets, any algorithm that knows nothing more than the profile of the instance, and has fewer than $(\delta\sqrt{M})OPT_\alpha(I_n)$ balls, has probability less than ϵ of filling $\lceil\alpha n\rceil$ buckets.*

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

Proof. We will restrict attention to the instances I_n introduced in the previous proof. Recall that, for these instances, we have $OPT_\alpha(I_n) \le n(\alpha+1)+1$. Thus, by hypothesis, our algorithm will be allowed

$$\left(\delta\sqrt{M}\right) OPT_\alpha(I_n) \le \delta(n(\alpha+1)+1)\sqrt{M}$$

balls for instance I_n. One implication of this is that our algorithm cannot fill more than $\delta(n(\alpha+1)+1)/\sqrt{M}$ size-M buckets in such an instance. To understand the significance of this number, recall that our hypothesis that $M \ge 8$ implies $\sqrt{M}/\lceil\sqrt{M}\rceil > 0.79$, and that we are also assuming that $n > 1/\alpha^2$. Thus, the maximum number of size-M buckets we can fill is

$$\begin{aligned}
\delta\frac{(\alpha+1)n+1}{\sqrt{M}} &= \left(\frac{1-\alpha}{1.25(1+\alpha)\max(6,\log(1/\epsilon)}\right)\left(\frac{(\alpha+1)n+1}{\sqrt{M}}\right)\\
&\le \left(\frac{(1-\alpha)}{7.5(1+\alpha)}\right)\left(\frac{(\alpha+1)n+1}{\sqrt{M}}\right)\\
&\le \frac{(\alpha+1)n+1-\alpha-\alpha(\alpha+1)/\alpha^2}{7.2(1+\alpha)\sqrt{M}} = \frac{(\alpha+1)n-\alpha-1/\alpha^2}{7.5(1+\alpha)\sqrt{M}}\\
&\le \frac{(\alpha+1)n}{7.5(1+\alpha)\sqrt{M}} = \frac{n}{7.2\sqrt{M}} < \frac{n}{7.5(.79)\lceil\sqrt{M}\rceil} < \frac{n}{5\lceil\sqrt{M}\rceil}.
\end{aligned}$$

Thus, in order to fill our desired $\lceil\alpha n\rceil$ buckets, we will have to fill more than 4/5 of the $n/\lceil\sqrt{M}\rceil$ buckets of size $\lceil\sqrt{M}\rceil+1$. In order to tell whether a bucket with size exceeding 1 has this size or size M, we must place $\lceil\sqrt{M}\rceil+1$ balls in the bucket. Therefore, the number of such buckets we can test is at most

$$\frac{\delta((\alpha+1)n+1)\sqrt{M}}{\lceil\sqrt{M}\rceil+1} < \delta((\alpha+1)n+1).$$

Given our initial random permutation of the buckets, every bucket tested must be chosen randomly from the as yet untested buckets with size exceeding 1. Thus, the probability of us finding at least 13/14 of the size-$(\lceil\sqrt{M}\rceil+1)$ buckets, i.e., $\left\lceil 13n/14\lceil\sqrt{M}\rceil\right\rceil$ such buckets, is no more than that of finding at least that many black balls when picking $\lfloor\delta((\alpha+1)n+1)\rfloor$ balls, without replacement, from an urn containing $n/\lceil\sqrt{M}\rceil+n-\lceil\alpha n\rceil$ balls, only $n/\lceil\sqrt{M}\rceil$ of which are black. Hoeffding [6] has shown that this probability is no more than that implied by Chernoff bounds for the sum of $\lfloor\delta((\alpha+1)n+1)\rfloor$ independent random 0-1 variables, which are 1 with probability $(n/\lceil\sqrt{M}\rceil)/(n/\lceil\sqrt{M}\rceil+n-\lceil\alpha n\rceil)$.

The particular form of the Chernoff bound we use is from [7, p. 64, Theorem 4.4(3)]:

Chernoff Bound Lemma. *Suppose $(X_1, X_2, \ldots, X_q)$ is a sequence of independent Poisson trials with probability of success p. Let $X = \sum_{i=1}^{q} X_i$ and $\mu = E[X]$. Then, for all $R \ge 6\mu$,*

$$Pr(X \ge R) \le 2^{-R}.$$

Now, in our case, we have

$$\begin{aligned}
\mu = qp* &= \lfloor\delta((\alpha+1)n+1)\rfloor\left(\frac{\frac{n}{\lceil\sqrt{M}\rceil}}{\frac{n}{\lceil\sqrt{M}\rceil}+n-\lceil\alpha n\rceil}\right) < \left(\delta((\alpha+1)n\right)\left(\frac{\frac{n}{\lceil\sqrt{M}\rceil}}{n(1-\alpha)}\right)\\
&= \frac{1-\alpha}{1.25(1+\alpha)\max(6,\log(1/\epsilon)}\left((\alpha+1)n\right)\left(\frac{\frac{n}{\lceil\sqrt{M}\rceil}}{n(1-\alpha)}\right)\\
&\le \frac{n}{1.25\max(6,\log(1/\epsilon)\lceil\sqrt{M}\rceil}.
\end{aligned}$$

Thus, the ratio of the number of size-$(\lceil\sqrt{M}\rceil+1)$ buckets needed to the expected number found is at least

$$R = \left(\frac{4n}{5\lceil\sqrt{M}\rceil}\right)\Big/\left(\frac{n}{1.25\max(6,\log(1/\epsilon)\lceil\sqrt{M}\rceil}\right) = \max(6,\log(1/\epsilon).$$

Thus, the lemma applies for this R, and we can conclude that we fill the required number of buckets with probability less than $\min(1/64,\epsilon) \le \epsilon$, as desired. □

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

(Note that there is some numeric slack in the above proof, introduced to simplify the arguments. We did not precisely optimize the value of δ, nor did we take into account the fact that if some size-M buckets were filled, we would have fewer balls left for filling size-$(\lceil\sqrt{M}\rceil+1)$ buckets. Consequently, the bounds of Theorem 3.7 and its corollary could be improved slightly at the cost of a more complicated proof.)

Note that Theorem 3.7 does not provide as strong a lower bound as does Theorem 3.4(C), since δ is not independent of ϵ, and increasing n does not decrease the success probability. We do not currently know whether there exist examples where this does happen – both n and M cancel out in the analysis of our current proof. The theorem is, however, strong enough to imply this easy corollary obtained by setting $\epsilon = 1/64$.

Corollary 3. *Suppose we are given α, $0 < \alpha < 1$. Then, for any randomized algorithm rA that knows nothing more than the profile of the instance and any $M \geq 8$,*

$$R^{rA}_{\alpha,M,\infty} \geq \frac{1-\alpha}{7.5(1+\alpha)}\sqrt{M}.$$

3.5. Algorithms exploiting the sizes-only case

Knowing the possible bucket sizes $s_1, s_2, \ldots, s_m$ occurring in our instance would seem to offer significant advantages. First, note that we may now assume that our algorithm operates in stages, as follows. Let $s_0 = 0$. At the beginning of each stage, every bucket contains s_i balls for some i, $0 \leq i \leq m$. This trivially holds at the beginning of the first stage, when all buckets contain $s_0 = 0$ balls. In each stage, we pick a bucket that is not full and add $s_{i+1} - s_i$ balls to it, where s_i is the number of balls the bucket contained at the beginning of the stage.

It is easy to see that any algorithm that does not operate in stages can be converted to one that does and never uses more balls. Suppose not. Then the algorithm must contain ball placements that have no possibility of providing us with new information. For example, suppose that buckets are of size 2 and 5. We learn no new information by placing a single ball in an empty bucket, or fewer than three additional balls in a bucket that contains 2 balls and is not full. Thus, given an algorithm that makes such "informationless" placements, we can obtain a staged algorithm that does no worse by omitting such placements if the algorithm never places enough balls into the given bucket to reach the next allowed capacity, and otherwise postponing the placements until just before the placement of the ball that filled the bucket to that next capacity.

Let "Staged Lowest Level" (SLL) be the algorithm that, when we have not yet filled our quota of buckets, picks an unfilled bucket containing the fewest balls, and adds to it enough balls to fill it up to the next possible bucket size. By the argument of the previous paragraph, SLL will perform no worse than LL, and it can clearly outperform LL in many situations. For example, in contrast to the results for LL, SLL can match the behavior of H1 on instances with just two capacities.

Theorem 3.8. *Suppose $0 < \alpha < 1$. For any variant dSLL of SLL based on a deterministic tie-breaking rule, and for any instance I with at most two distinct bucket capacities, we are guaranteed to have*

$$dSLL_\alpha(I) \leq (1/\alpha)OPT_\alpha(I).$$

(This result is proved in exactly the same way as Theorem 3.3.)

Unfortunately, as soon as there are three or more sizes, knowing them no longer buys us much under SLL, as we can essentially recreate the same lower bounds as for LL in Theorem 3.2 by using sizes 1, $M-1$, and M in the lower bound examples, as the reader can easily verify.

SLL can outperform LL with respect to a different metric, however – the asymptotic ratio $R^{A}_{\alpha,M,\infty}$ holding when the number n of buckets is fixed but the maximum buckets size M is not. For this metric, the best result we have seen so far is that of Theorem 3.2 for the Lowest Level algorithm, which says that $R^{LL}_{\alpha,\infty,n} = n - \lceil\alpha n\rceil + 1$.

Consider the randomized version rSLL of SLL that at each step chooses a bucket randomly (and uniformly) from among the least filled buckets, and adds balls to it to bring it up to the next possible size.

Theorem 3.9. *For $0 < \alpha < 1$ and $n > 1/(1-\alpha)$, $R^{rSLL}_{\alpha,\infty,n} \leq (n - \lceil\alpha n\rceil)/2 + 1$.*

Moreover, this bound is tight and no algorithm can do better, even if it knows the full profile.

Proof. Suppose we are given an instance I with $n > 1/(1-\alpha)$ and hence $n - \lceil\alpha n\rceil > 0$. The optimal solution for I consists of the $\lceil\alpha n\rceil$ smallest buckets, i.e., those with sizes $s_1 \leq s_2 \leq \cdots \leq s_{\lceil\alpha n\rceil}$. Thus we know that $OPT_\alpha(I) \geq s_{\lceil\alpha n\rceil}$. Let $\delta = OPT_\alpha(I)/\lceil\alpha n\rceil - 1$, so that $OPT_\alpha(I) = (1+\delta)s_{\lceil\alpha n\rceil}$.

If $\delta \geq 1$ we are done. For note that $rSLL_\alpha(I) \leq OPT_\alpha(I) + (n - \lceil\alpha n\rceil)s_{\lceil\alpha n\rceil}$, and so

$$\frac{rSLL_\alpha(I)}{OPT_\alpha(I)} \leq 1 + \frac{(n-\lceil\alpha n\rceil)s_{\lceil\alpha n\rceil}}{2s_{\lceil\alpha n\rceil}} = \frac{n-\lceil\alpha n\rceil}{2} + 1,$$

as claimed.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

So assume $\delta < 1$. Then we must have $s_{\lceil \alpha n \rceil - 1} \leq \delta s_{\lceil \alpha n \rceil} < s_{\lceil \alpha n \rceil}$, and so at some point the algorithm will have reached a state in which each bucket contains a number of balls equal to the minimum of its size and $s_{\lceil \alpha n \rceil - 1}$, for a total of $\lceil \alpha n \rceil - 1$ full buckets and $OPT_\alpha(I) - s_{\lceil \alpha n \rceil} + (n - \lceil \alpha n \rceil + 1)s_{\lceil \alpha n \rceil - 1}$ balls. Algorithm rSLL will then randomly choose buckets from among the $n - \lceil \alpha n \rceil + 1$ currently not full buckets containing $s_{\lceil \alpha n \rceil - 1}$ balls, adding $s_{\lceil \alpha n \rceil} - s_{\lceil \alpha n \rceil - 1}$ balls to each, until it encounters one for which the result is a full bucket. At this point it will have filled $\lceil \alpha n \rceil$ buckets and will halt. Elementary calculations show that the expected number of buckets that will have received these additional balls when that full bucket is obtained is $(n - \lceil \alpha n \rceil + 2)/2$. Thus the expected total number of balls used by rSLL is

$$\begin{aligned} &OPT_\alpha(I) - s_{\lceil \alpha n \rceil} + (n - \lceil \alpha n \rceil + 1)s_{\lceil \alpha n \rceil - 1} + \frac{(n - \lceil \alpha n \rceil + 2)(s_{\lceil \alpha n \rceil} - s_{\lceil \alpha n \rceil - 1})}{2} \\ &\quad = OPT_\alpha(I) + \frac{(n - \lceil \alpha n \rceil)s_{\lceil \alpha n \rceil - 1}}{2} + \frac{(n - \lceil \alpha n \rceil)(s_{\lceil \alpha n \rceil})}{2} \\ &\quad \leq OPT_\alpha(I) + \frac{(n - \lceil \alpha n \rceil)(1 + \delta)s_{\lceil \alpha n \rceil})}{2}. \end{aligned}$$

Then, since $OPT_\alpha(I) = (1 + \delta)s_{\lceil \alpha n \rceil}$, we have

$$\frac{rSLL_\alpha(I)}{OPT_\alpha(I)} \leq 1 + \frac{(n - \lceil \alpha n \rceil)(1 + \delta)s_{\lceil \alpha n \rceil}}{2(1 + \delta)s_{\lceil \alpha n \rceil}} = \frac{n - \lceil \alpha n \rceil}{2} + 1,$$

as claimed. □

Theorem 3.10. *Let A be any algorithm that knows the full profile, but only begins packing after the buckets have been randomly permuted. Then, for $0 < \alpha < 1$ and $n > 1/(1 - \alpha)$, $R^A_{\alpha,\infty,n} \geq (n - \lceil \alpha n \rceil)/2 + 1$.*

Proof. Our lower bound is based on n-bucket instances with $OPT_\alpha(I_n) = n - \lceil \alpha n \rceil + X$ for a constant X to be determined later. The instance consists of $\lceil \alpha n \rceil - 1$ buckets of size 1, one bucket of size $X + 1$, and $n - \lceil \alpha n \rceil$, much larger, buckets having size $OPT_\alpha(I_n)\big((n - \lceil \alpha n \rceil)/2 + 1\big)$. Filling any one of these latter buckets would use the number of balls claimed by our proposed lower bound, so we can assume that the algorithm A does not fill any of them. Therefore, in order to fill $\lceil \alpha n \rceil$ buckets, it must fill the bucket of size $X + 1$. The algorithm cannot, however, distinguish the size-$(X + 1)$ bucket from a bucket with the larger size without putting $X + 1$ balls in the bucket in question. Given that the buckets were initially randomly permuted, the expected number of buckets that must be so tested before the bucket of size $X + 1$ is found is again $(n - \lceil \alpha n \rceil + 2)/2$, of which one is filled and the remaining $(n - \lceil \alpha n \rceil)/2$ are not. Thus the expected number of balls used is at least $OPT_\alpha(I_n) + (n - \lceil \alpha n \rceil)(X + 1)/2$. We thus have that

$$\frac{rSLL_\alpha(I)}{OPT_\alpha(I)} \geq 1 + \frac{(n - \lceil \alpha n \rceil)(X + 1)}{2(X + \lceil \alpha n \rceil)}$$

Taking the limit as $X \to \infty$ gives the desired result. □

Note that this lower bound result, although tight, is weaker than our earlier ones, since it only refers to expected values. The best probabilistic bound we can currently prove derives from Theorem 2.2, by fixing n in that result rather than B. We then obtain (roughly, and for large n), that there is a constant $a < 1$ and instances I'_n such that, assuming we randomly permute the buckets at the beginning, any algorithm that only knows the profile and uses fewer than $OPT_\alpha(I'_n)(n - \lceil \alpha n \rceil)/8$ balls has probably less than α^n of success. This can probably be improved a bit, but it is not clear that there is a similar result when the denominator 8 is replaced by something arbitrarily close to the 2 of our lower bound on expected behavior.

As a final remark about sizes-only algorithm, we note that, analogously to our modifying LL to the staged version SLL, we can also adapt the H2 algorithm of the previous section to a staged version SH2 that might perform better in practice. Here, instead of setting $M' = \lfloor \sqrt{M} \rfloor$, we can let it be the maximum bucket size s_i that is less than $\sqrt{M}$. The proof of Theorem 3.5 will still apply.

3.6. Algorithms that exploit the known-profile case

In light of the lower bound Theorems 2.1, 3.4, 3.7, and 3.10, there would seem to be little opportunity to exploit the extra information available in a known profile to obtain a general qualitative improvement. We ourselves have not yet found one, and leave this as an open problem for the reader.

4. A bucket-based metric

In this section we consider a metric based on bucket counts rather than ball counts, and in particular how much hidden diversity reduces the number of buckets we can fill, given that we have enough balls to fill our target number of buckets.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

As before, let us fix an α, $0 < \alpha < 1$. For an algorithm A, an instance I with n buckets, and a number b of balls, let $A(I, b)$ be the number of buckets filled when A has placed b balls – we assume that the algorithm assumes it has an unbounded supply of balls and does not know b in advance. Let $OPT_\alpha(I)$ be the number of balls in an optimal placement for filling $\lceil \alpha n \rceil$ buckets. For a deterministic algorithm dA, define

$$X^{dA}_{\alpha,n} = \min\left\{ \frac{A(I, OPT_\alpha(I))}{\lceil \alpha n \rceil} : \ I \text{ is an instance with } n \text{ buckets} \right\}.$$

For a randomized algorithm rA, define

$$X^{rA}_{\alpha,n} = \min\left\{ \frac{E[A(I, OPT_\alpha(I))]}{\lceil \alpha n \rceil} : \ I \text{ is an instance with } n \text{ buckets} \right\}.$$

In both cases, define

$$X^{A}_{\alpha,\infty} = \liminf_{n\to\infty} X^{A}_{\alpha,n}.$$

Note that we are not parameterizing by the maximum bucket capacity M here. A maximum bucket size of M automatically guarantees that $X^{dFill}_{\alpha,\infty} = 1/M$, but getting exact results for sophisticated algorithm when M is bounded is likely to be complicated. Consequently, we shall leave such questions for later. The results for unbounded M are interesting in their own right.

With respect to this metric, best algorithm we currently know is a randomized one which we shall called "d-Sampled Lowest Level" algorithm (d-*Sample*), where $0 < d < 1$).

1. Randomly pick $\lfloor d\lceil \alpha n \rceil \rfloor$ buckets.
2. Perform LL on just these buckets until there are no more balls left.

Theorem 4.1. *For all α, $0 < \alpha < 1$, and d, $0 < d \le (1 - \sqrt{1-\alpha})/\alpha$,*

$$d\left(\frac{1-d}{1-d\alpha}\right)\alpha \le X^{d-Sample}_{\alpha,\infty} \le d\alpha,$$

where the lower bound is always at least $d^2\alpha$ and is maximized when d equals its upper bound, which itself exceeds $1/2$.

(Note that, as $\alpha \to 1$, the ratio between the upper bound and the best lower bound goes to 1, but, as $\alpha \to 0$, the ratio goes to $1 - d$. Thus there is clearly still work to do in analyzing the algorithm. For $\alpha = 1/2$, the upper bound on d is approximately 0.58579 and, with d set to this value, we have $0.17157 < X^{d-Sample}_{\alpha,\infty} < 0.2929$.)

Proof. First, let us deal with the numerical claims. The upper bound will be greater than $1/2$ so long as $\sqrt{1-\alpha} < 1 - \alpha/2$, which is equivalent to $1 - \alpha < 1 - \alpha + \alpha^2/4$, which is true since $\alpha > 0$. For future reference, also note that our upper bound $(1 - \sqrt{1-\alpha})/\alpha$ is always less than 1, since $\alpha < 1$, which implies $1 - \alpha < \sqrt{1-\alpha}$.

To see that $(1-d)/(1-d\alpha) \ge d$, we must show that $1 - d \ge d(1 - d\alpha)$ or $d^2\alpha - 2d + 1 \ge 0$. For the latter, we have equality when $d = \left(1 \pm \sqrt{1-\alpha}\right)/\alpha$, with the negative option the only one that yields a d in the allowed range. Note that because both d and α are less than 1, the function $d^2\alpha - 2d + 1$ has a negative derivative and so is a decreasing function of d. Thus the claimed bound $(1-d)/(1-d\alpha) \ge d$ holds for all allowed values of d.

To see that the bound $d\,((1-d)/(1-d\alpha))\,\alpha$ is maximized when d attains its maximum allowed value of $(1 - \sqrt{1-\alpha})/\alpha$, we take a derivative with respect to d, obtaining

$$\frac{(1-2d)(1-d\alpha) - (d - d^2)(-\alpha)}{(1-d\alpha)^2} = \frac{d^2\alpha - 2d + 1}{(1-d\alpha)^2}.$$

Our original function will be either minimized or maximized when the numerator equals 0, which we already know happens when $d = 1 - \sqrt{1-\alpha})/\alpha$. Since the derivative is positive for $d = 1/2$ and the smallest zero for the derivative is the one given above, this means that the function is maximized at this value.

Now let us turn to the asymptotic bounds for the algorithm, starting with the upper bound, which is straightforward. It is based on the following instances I_n, $n > 1/(d\alpha)$. In I_n, we have $\lceil \alpha n \rceil$ buckets of size 1, and the remaining buckets all have size $K = \lceil \alpha n \rceil + 1$. Thus we have $OPT_\alpha(I_n) = \lceil \alpha n \rceil$ and our algorithm has this many balls to play with. The expected number of size-1 buckets in our sample is $\lfloor d\lceil \alpha n \rceil \rfloor (\lceil \alpha n \rceil / n)$ which approaches the desired upper bound $(d\alpha)\,OPT_\alpha(I_n)$ on filled buckets as $n \to \infty$. And note that we do not have enough balls to fill any other buckets.

For the lower bound, let us warm up by fixing $d = 1/2$ and proving a weaker but easier bound $d\alpha/2$. Let $B_1, B_2, \ldots, B_n$ be an indexing of the buckets by increasing size, ties broken arbitrarily. Recall that this means that $OPT_\alpha(I) = \sum_{i=1}^{\lceil \alpha n \rceil} s_i$. Let $t = \lceil \lceil \alpha n \rceil / 2 \rceil$, and call all the buckets with indices $i \le t$ "good". Note that we will have filled all good buckets by the time

LL has brought every bucket in the sample up to the minimum of its size and s_t, which will require at most $s_t \lfloor d\lceil \alpha n \rceil \rfloor$ balls. But now note that

$$OPT_\alpha(I) \geq \sum_{i=1}^{t} s_i + s_t(\lceil \alpha n \rceil - t) \geq \sum_{i=1}^{t} s_i + s_t \lceil \alpha n \rceil / 2 \geq s_t \lfloor d\lceil \alpha n \rceil \rfloor .$$

Consequently we have enough balls to fill all the good buckets in our sample, of which the expected number is $d\lceil \alpha n \rceil (t/n) = (d\lceil \alpha n \rceil \lfloor \lceil \alpha n \rceil / 2 \rfloor)/n$, which approaches $(d\alpha/2)\lceil \alpha n \rceil$ as $n \to \infty$, proving our lower bound claim.

There is considerable slack in the above argument, even when $d = 1/2$. This is because we can use the balls corresponding to the unused $\sum_{i=1}^{t} s_i$ component of the lower bound on OPT to fill the good buckets, so that the balls corresponding to the $s_t(\lceil \alpha n \rceil - t)$ component only need to be used to fill non-good buckets. This is what we exploit in the general lower bound of the Theorem, which we now prove.

Let $d^* = (1-d)/(1-d\alpha)$. For all $\epsilon > 0$, we will show that $X_{\alpha,\infty}^{d-Sample} \geq (1-\epsilon)dd^*\alpha$, which will imply the lower bound. We may assume without loss of generality that $\epsilon < 1$. Suppose I is an arbitrary instance with n buckets. Let t now equal $\lfloor (1-\epsilon)d^* \lceil \alpha n \rceil \rfloor$, and call a bucket B_i "good" if $i \leq t$, and "bad" otherwise. Let $r = \lfloor d\lceil \alpha n \rceil \rfloor$, the number of buckets we sample. Then the expected number g of good buckets in our sample of r buckets is

$$E[g] = r\left(\frac{t}{n}\right) \sim d\alpha t \sim (1-\epsilon)dd^*\alpha \lceil \alpha n \rceil,$$

where we use "$\sim$" to mean essentially equal as $n \to \infty$. Note that this limiting value is just the number of buckets we need to fill for the claimed lower bound on $X_{\alpha,\infty}^{d-Sample}$ to hold.

To complete the proof, we need to show that, in asymptotic expectation, our algorithm will be guaranteed to fill all the good buckets in our sample as soon as it has placed a number of balls that is no more than $OPT_\alpha(I)$. Let b denote the number of bad buckets in our sample. All of the good buckets in our sample are guaranteed to be filled as soon as LL has placed enough balls in the buckets of the sample for each to contain the minimum of its size and s_t. This number is clearly no more than $\sum_{i=1}^{t} s_i + bs_t$. On the other hand, the number of balls we will have available is at least $OPT_\alpha(I) \geq \sum_{i=1}^{t} s_i + s_t(\lceil \alpha n \rceil - t)$. Thus we will have enough balls so long as $b \leq \lceil \alpha n \rceil - t$.

Now note that

$$E[b] = d\lceil \alpha n \rceil - E[g] \sim d\lceil \alpha n \rceil - d\alpha t \sim d\alpha n\big(1 - (1-\epsilon)d^*\alpha\big)$$

and thus

$$\begin{aligned}(\lceil \alpha n \rceil - t) - E[b] &= \lceil \alpha n \rceil - \left\lfloor (1-\epsilon)d^* \lceil \alpha n \rceil \right\rfloor - E[b] \\ &\sim \alpha n\big(1 - d - (1-\epsilon)d^*(1-d\alpha)\big) \\ &= \alpha n\big(1 - d - (1-\epsilon)(1-d)\big) = (1-d)\alpha n\epsilon.\end{aligned}$$

Thus, we will fill all the good buckets in our sample unless we have

$$b - E[b] > (1-d)\alpha n\epsilon \sim \frac{(1-d)\alpha n\epsilon E[b]}{d\alpha n\big(1-(1-\epsilon)d^*\alpha\big)} > \left(\frac{\epsilon(1-d)}{d(1-(1-\epsilon)d^*\alpha)}\right) E[b].$$

This is a constant fraction β of $E[b]$, and so another Chernoff bound theorem ([7, p. 64, Theorem 4.4(2)], applicable again via the arguments of [6]) implies that the probability that this holds is no more than $e^{-(\beta^2/3)E[b]} = a^n$ for some constant $a < 1$ since $E[b]$ is itself linear in n.

Let f be the number of good buckets actually filled after we have placed $OPT_\alpha(I)$ balls (a lower bound on the total number of buckets filled). Then we have

$$E[f] \geq \sum_{i=1}^{d\lceil \alpha n \rceil} (i \cdot Pr[g = i] \cdot Pr[\text{all good buckets filled}])$$

Now, as observed above, all good buckets will be filled unless $b > E[b] + (1-d)\alpha n\epsilon$ or, equivalently, $g < E[g] - (1-d)\alpha n\epsilon$, and this happens with probability less than a^n. Thus, in the above expression for $E[f]$, the only summands which can take on values less than the full count i are ones whose cumulative probability is less than a^n, which drops exponentially in n, whereas the maximum value for i is no more than $d\lceil \alpha n \rceil$, which grows only linearly. Thus, as $n \to \infty$, the contribution of these terms to $E[f]$ goes to 0, and, in the limit, the expected number of buckets we fill is at least the expected number of good buckets in our sample, which we have already seen is precisely what is needed to prove our lower bound on $X_{\alpha,\infty}^{d-Sample}$.

(The above argument, rendered informal by the use of "$\sim$", can be made rigorous by interpreting $A \sim B$ to mean $B \in [(1-\delta)A, (1+\delta)A]$ where $\delta > 0$ is an arbitrarily small constant, and the inclusion holds for all $n > n_\delta$, where n_δ is a constant depending only on δ. All the statements involving "$\sim$" can then be restated in terms of such δ, with our claims holding by taking yet another limit, this one for $\delta \to 0$.) □

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

In contrast, we have the following general upper bound theorems.

Theorem 4.2. *For all α, $0 < \alpha < 1$, if dA is any deterministic algorithm, even one that knows the profile, we have*

$$X^{dA}_{\alpha,\infty} = 0.$$

Proof. Assume n is sufficiently large that $\Delta = n - \lceil \alpha n \rceil > 0$. Given dA, we will show how to construct an instance I_n with n buckets for which $dA(I_n, OPT_\alpha(I_n)) = 0$. The theorem will follow. We start with n buckets of undetermined size, and apply dA for $n\lceil \lceil \alpha n \rceil / \Delta \rceil$ ball placements, each time informing the algorithm that the placement did not fill the chosen bucket. Now declare each bucket to have size one more than the number of balls it currently contains. Divide the buckets into two sets: C, the $\lceil \alpha n \rceil$ buckets containing the fewest balls (ties broken arbitrarily), and D, the remaining Δ buckets. Let T denote the number of balls in the buckets of set C. Then $OPT_\alpha(I_n) = T + \lceil \alpha n \rceil$. Now note that the Δ buckets in D contain at least $\Delta\lceil \lceil \alpha n \rceil / \Delta \rceil \geq \lceil \alpha n \rceil$ balls. Thus, when we stopped dA, it had already placed T balls in the buckets of C and at least $\lceil \alpha n \rceil$ balls in the buckets of D, for a total of at least $OPT_\alpha(I_n)$ balls, without filling any bucket. So $dA(I_n, OPT_\alpha(I_n)) = 0$, as required. □

Theorem 4.3. *For any randomized algorithm rA,*

$$X^{rA}_{\alpha,\infty} \leq \alpha.$$

Proof. We exhibit a sequence of instances I_n that imply the upper bound on $X^{rA}_{\alpha,\infty} \leq \alpha$ for any randomized algorithm rA. These instances are quite simple: I_n contains $\lceil \alpha n \rceil$ buckets of size 1, and the remaining buckets all have capacity $n + 1$. This means that $OPT_\alpha(I_n) = \lceil \alpha n \rceil$, and so, when restricted to $OPT_\alpha(I_n)$ balls, algorithm rA can only place balls in this many buckets. It can only *fill* a bucket when it chooses one with size 1, since all the others have size exceeding $OPT_\alpha(I_n)$. However, since it has no information about the identity of the buckets, the expected number of buckets of size 1 that it will choose is only $\big(\lceil \alpha n \rceil\big)\big(\lceil \alpha n \rceil / n\big)$. This means that the ratio of this number to the target number is $\lceil \alpha n \rceil / n$, yielding the desired limit as $n \to \infty$. □

Theorem 4.4. *Suppose we are given α, $0 < \alpha < 1$, and $\delta > 0$. Then there is a constant $a < 1$ and a sequence of instances I_n such that the following holds for sufficiently large n: Assuming we first use a random permutation to relabel the buckets, any algorithm that knows nothing more than the profile of the instance, and has no more than $OPT_\alpha(I)$ balls, has probability less than a^n of filling $(1 + \delta)\alpha\lceil \alpha n \rceil$ buckets.*

Proof. We use the same examples as in the proof of Theorem 4.3. The algorithm can only afford to fill buckets of size 1, and has $\lceil \alpha n \rceil$ balls with which to do so. Given the random relabeling of the buckets, its success is thus simply the probability that, in making $\lceil \alpha n \rceil$ random bucket choices, it will find at least $(1 + \delta)\alpha\lceil \alpha n \rceil$ size-1 buckets, where the expected number of size-1 buckets it chooses goes to $\alpha\lceil \alpha n \rceil$ as $n \to \infty$. The claim follows by an application of the Chernoff bound used in the proof of Theorem 4.1. □

5. Summary and open questions

In this paper we studied a peculiar "balls in buckets" problem under incomplete information, which we call *hidden diversity*, and which is motivated by the possibility in cryptographic multi-party protocols of "masking" the effort an adversary would have to invest in order to take over some of the participants. We formulated several variants of such masking in the balls-and-buckets setting and presented algorithms and lower bounds on the best that any algorithm can achieve. An interesting variant of the problem, which is the subject of on-going work, is to consider the effect of having to "pay for security"—i.e., assigning a cost, which could be decreasing or increasing, to the number and/or depth of the buckets, and answer natural questions such as for a given budget B, what is the best system in the full-knowledge regime, or how many extra balls does the adversary need for such systems under hidden diversity. Other open questions include to tighten the gaps between upper and lower bounds where they exist, and understand the real costs of building such systems and of attackers, so as to develop more accurate models.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016

References

[1] Michael Ben-Or, Shafi Goldwasser, Avi Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract), in: Janos Simon (Ed.), STOC, ACM, 1988, pp. 1–10.
[2] David Chaum, Claude Crépeau, Ivan Damgard, Multiparty unconditionally secure protocols, in: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC '88, ACM, New York, NY, USA, 1988, pp. 11–19.
[3] Richard Cleve, Limits on the security of coin flips when half the processors are faulty (extended abstract), in: Juris Hartmanis (Ed.), STOC, ACM, 1986, pp. 364–369.
[4] Juan A. Garay, David S. Johnson, Aggelos Kiayias, Moti Yung, Resource-based corruptions and the combinatorics of hidden diversity, in: Robert D. Kleinberg (Ed.), Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013, ACM, 2013, pp. 415–428.
[5] Oded Goldreich, Silvio Micali, Avi Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, in: Alfred V. Aho (Ed.), STOC, ACM, 1987, pp. 218–229.
[6] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[7] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, Cambridge, UK, 2005.

Please cite this article in press as: J. Garay et al., The combinatorics of hidden diversity, Theoret. Comput. Sci. (2019), https://doi.org/10.1016/j.tcs.2019.07.016