Randomness vs. Time in Anonymous Networks

Jochen Seidel Jara Uitto Roger Wattenhofer

ETH Zurich, Switzerland
{seidelj, juitto, wattenhofer}@ethz.ch

Abstract

In an anonymous network, symmetry breaking tasks can only be solved if randomization is available. But how many random bits are required to solve any such task? As it turns out, the answer to this question depends on the desired runtime of the algorithm.

Since any randomized anonymous network algorithm can be decomposed into a randomized 2-hop coloring stage and a deterministic stage, we tackle the question by focusing on the randomized stage. We establish that for any reasonable target function f, there is a randomized 2-hop coloring scheme running in $\mathcal{O}(f(n))$ time. Our coloring scheme allows to trade an increase in runtime by a factor of d for a decrease by the d^{th} root in the random bit complexity.

To show that the achieved trade-off is asymptotically optimal for any choice of f, we establish a trade-off lower bound. Our bounds yield that it is sufficient to consider the cases when f is between $\Omega(\log^* n)$ and $\mathcal{O}(\log \log n)$. We obtain that for the two extreme cases, i.e., where $f \in \Theta(\log^* n)$ and $f \in \Theta(\log \log n)$, the random bit complexity is $\Theta(\sqrt[d]{n})$ and $\Theta(\log n)$, respectively, for any constant d. The trade-off achieved by our scheme is asymptotically optimal for any f, i.e., reducing the runtime must lead to an increase in the random bit complexity.

1 Introduction

We consider randomized algorithms running in a network of n communicating nodes. The network is *anonymous*, as opposed to identified networks in which nodes can be distinguished by their unique identifiers (IDs). The computational power of deterministic anonymous network algorithms has been found to be rather limited [31]. When nodes have access to random bits however, many interesting tasks become solvable. But what is the amount of random bits, i.e., the *random bit complexity*, required to solve any such task?

Consider, for example, the fundamental symmetry breaking problem of graph coloring, where the goal is to assign colors to nodes so that every two neighbors get a different color. In a complete network, i.e., when every node is connected to all other nodes, a unique color must be used for every node. Therefore, for complete networks the answer is at least $\log n$ random bits. One result of our work is that in expectation $\Omega(\log n)$ random bits are required even if every node in the network has at most 3 neighbors. Moreover, we establish that $\mathcal{O}(\log n)$ random bits in expectation are also sufficient to solve all tasks in any network.

Alongside the random bit complexity, as a second efficiency measure, we consider the *runtime* required to solve such tasks. Increasing the runtime allows one to draw the random bits more carefully, thus reducing the number of unnecessarily drawn random bits. Conversely, it is true that drawing random bits more generously enables faster runtime. We study how exactly the random bit complexity relates to the runtime.

More precisely, we show that there is an *efficiency trade-off* between the runtime and the random bit complexity required to solve any task. Our contribution is to establish asymptotically tight lower and upper bounds on the achievable trade-off. Those bounds imply that using more than $\mathcal{O}(\log\log n)$ rounds to solve a task does not result in a better random bit complexity. Linial's local symmetry breaking lower bound, showing that one requires roughly $\log^* n$ rounds [27] to 3-color a ring, already hints that the interesting cases occur when the asymptotic runtime is between $\log^* n$ and $\log\log n$. In the respective extreme cases, i.e., when the runtime is $\log^* n$ or $\log\log n$, our lower bound states that the random bit complexity is $\Omega(\sqrt[d]{n})$ and $\Omega(\log n)$, correspondingly, where d is a constant that depends on the runtime.

For the upper bound we devise a randomized scheme that produces sufficiently many random bits for any anonymous network algorithm. To this end we introduce the notion of a *target function* f which specifies the desired runtime of our scheme, and consider the cases where $f(n)$ is asymptotically between $\log^* n$ and $\log\log n$. The trade-off achieved by our scheme asymptotically matches the lower bound with high probability[1] and in expectation, also for all runtimes f that lie between the two extremes.

Our scheme is *uniform*: The algorithm does not require any knowledge about the network topology, such as its size or diameter. Note that this rules out the trivial approach of drawing a unique identifier with $O(\log n)$ bits, which would succeed with high probability. Being uniform, our scheme can be used to devise new uniform algorithms for classic symmetry breaking problems by utilizing existing deterministic algorithms. This is due to the fact that those algorithms often assume IDs, but function correctly even if those IDs are only *locally* unique. As one example, consider the deterministic coloring algorithm from [35] which runs in $\mathcal{O}(\log^* n)$ time on graphs with bounded growth. By applying our scheme, we obtain a uniform coloring algorithm for anonymous networks with the same runtime. Our lower bounds imply that a $\mathcal{O}(\log^* n)$ runtime is the best possible. This speed comes at the cost of a relatively high random bit complexity, which is $\Theta(\sqrt[d]{n})$. Note, however, that d is a freely selectable parameter of our scheme (a constant) that is hidden in

[1] We say an event occurs *with high probability* (w.h.p.) if it occurs with probability $1 - n^{-c}$ for any constant c.

the big-$\mathcal{O}$ notation. If one is willing to sacrifice the asymptotic runtime, on the other end of the spectrum, our approach allows to solve the same task in $\mathcal{O}(\log\log n)$ time using as little as $\mathcal{O}(\log n)$ random bits. By tuning the f parameter, any trade-off between the two extremes can be achieved.

So how can we possibly bound the random bit complexity for any computable task? The answer to this *complexity* question can be based on a recent *computability* result by Emek et al. [14], where they showed that a 2-hop coloring[2] is necessary and sufficient to replace access to random bits in any anonymous network algorithm. We therefore establish our upper bound by devising a 2-hop coloring algorithm whose runtime and random bit complexity are tuneable by a target function f and a constant d.

1.1 Related Work

The theory of distributed computability began with Angluin's insight that leader election is impossible in anonymous rings [3]. A similar impossibility argument can be made for deterministic algorithms that solve local symmetry breaking tasks, e.g., coloring or MIS, and literally hundreds of more impossibilities are known [6]. In short, the computational power of deterministic anonymous network algorithms is limited [31].

Under the assumption of uniform algorithms, the leader election impossibility result from [3] extends to the case where randomization is available. In contrast to that, when randomization is available, there are well known algorithms that solve the local symmetry breaking problems coloring [27] and MIS [1, 28] also in anonymous networks. It is interesting to note that both randomized MIS algorithms are used to construct completely derandomized (deterministic) variants under the assumption that unique identifiers are available. How much randomization an anonymous network will ever need from a *computability* perspective can be characterized in terms of a 2-hop coloring [15]. In this paper, based on that observation, we tackle the *complexity* question, i.e., the random bits and runtime necessary to obtain a 2-hop coloring. When unique IDs are available, runtime and messages (size and quantity) can be traded, e.g., in MIS and coloring algorithms [22]. Focusing on anonymous algorithms, we study a trading runtime with a fourth complexity measure, namely the random bit complexity. Also outside of anonymous algorithms, randomization has many applications in distributed computing (cf. [7]), e.g., in agreement [4, 5], self stabilization [14], and non-uniform leader election [1].

Still, one of the most basic tasks to solve in a distributed setting remains coloring, and often coloring and MIS algorithms go hand in hand. As such, they were studied thoroughly (please refer to [10] for an extensive overview), usually aiming to use at most $\Delta + 1$ (or at least some small function of Δ) many colors. Perhaps surprisingly, when identifiers are available, deterministic coloring algorithms are among the fastest. A recent series of results by Barenboim, Elkin, and Kuhn [8, 25, 11] yields a $\Delta+1$ coloring in $\mathcal{O}(\Delta+\log^* n)$ runtime by utilizing a new defective coloring technique. The picture is completed by the observation that colors can be traded for runtime [9], i.e., one can get $\mathcal{O}(\Delta^\varepsilon + \log^* n)$ for $\mathcal{O}(\Delta)$ colors or $\mathcal{O}(\log\Delta \cdot \log^* n)$ for $\mathcal{O}(\Delta^{1+\varepsilon})$ colors. These deterministic coloring algorithms have in common that they need to assume IDs. Also randomized algorithms (e.g. [35, 34]) often assume IDs and are not uniform, i.e., they assume knowledge about n or some other global network parameter. Relieving the algorithm from that knowledge, we focus on achieving a good random bit complexity instead of a low number of used colors, and refer to standard methods (e.g., the deterministic approach in [19]) to reduce this number. On the other hand, the $\mathcal{O}(\log n)$ algorithms for MIS [1, 28] and coloring [27] are uniform, and can be formulated

[2] A 2-hop coloring is a coloring of the network in which every node's color is different from the colors used by any other node within distance 2 (see Section 2).

even in very restricted models [36]. We improve on the runtime at the lowest possible price one needs to pay for that in terms of random bit complexity.

It is worth mentioning that in the context of self-stabilization [13], uniform MIS and (2-hop) coloring protocols were studied also for anonymous networks. For instance, [37] considers deterministic and randomized protocols that color paths and rings, and later [21] obtain randomized protocols for MIS and coloring in arbitrary networks. The recent work [12] presents a 2-hop coloring protocol for graphs of bounded degree. In the self-stabilization context, the difficulty lies in dealing with faults. The random bit complexity is of no concern in the protocols mentioned above, and the runtime of [12] is necessarily much higher than in our non-faulty environment.

Sequential probabilistic computability was pioneered by Gill [18], showing that, e.g., ZPP = RP $\cap$ co-RP, and Rabin [33], who reduced certain probabilistic automata to deterministic ones. Reducing the error probability using few additional random bits was studied, e.g., for the classes RP ([23], cf. [38]) and BPP (e.g., [2]), and [26] relates BPP to the polynomial hierarchy. Derandomization [29] is closely related to extracting randomness from low entropy sources [32, 38]. The field of randomized computability and complexity is covered in great detail in [30]. A distributed version of BPP, so called (p, q)-deciders, and derandomization in this setting were studied in [16]. We characterize how many random bits are necessary to solve any anonymous network task with probability 1 depending on the desired runtime.

A concept related to that of randomization is non-determinism. The distributed notion of this concept, where often IDs are assumed, was initiated by Naor and Stockmeyer [31], who studied what could be *checked* by deterministic constant-time algorithms if some labeling (non-determinism) is known in advance. Subsequently, the number of non-deterministic choices required to solve decision problems in this distributed manner was investigated [24]. A hierarchy of decidable problems depending on the necessary amount of non-determinism arises [20], also when the network is anonymous. Recently, it was found that in fact the *combination* of non-determinism with randomization allows distributed algorithms to decide any language in constant time [17].

2 Preliminaries

We model the network as a simple, undirected graph $G = (V, E)$, where V and E denote the set of nodes and edges, respectively. The *network size*, i.e., the cardinality of V, is denoted by n. Furthermore, the *exclusive neighborhood* of a node $u \in V$ in G is the set $\Gamma(u) = \{v : (u, v) \in E\}$. Similarly, we denote by $\Gamma^2(u) = \Gamma(u) \cup_{v \in \Gamma(u)} \{w : w \neq u, (v, w) \in E\}$ the *exclusive* 2*-hop neighborhood* of u. Note that throughout this paper, we assume that all logarithms are taken to base 2.

Uniform Randomized Algorithms. We consider randomized algorithms that always return a correct output and have finite expected runtime (Las Vegas algorithms). Our algorithms run under the synchronous broadcast model, i.e., the execution of an algorithm can be divided into discrete *rounds* starting from round 1. Furthermore, the execution of any round $r + 1$ for any node u begins only when every other node has finished executing round r. Round r executed by a node u is divided into 4 parts in the following manner.

1. **Receive.** Node u receives the messages sent by nodes in $\Gamma(u)$ in round $r - 1$.
2. **Randomized Computation.** Node u can perform arbitrary computations. During the computation u can draw a finite amount of random bits. The source of random bits for node u is independent from the source of random bits for any other node $v \in V$, and for the sake of simplicity we assume that each source is uniformly distributed.

3. **Output.** Node u can decide on an output value. An output is irrevocable, i.e., once u has decided on an output value, it cannot be changed.
4. **Send.** Node u sends a finite length broadcast message to all nodes in $\Gamma(u)$.

An algorithm $\mathcal{A}$ is called *deterministic* if $\mathcal{A}$ does not draw any random bits. When all nodes in the network have decided on an output value we say that $\mathcal{A}$ *has terminated.* We restrict ourselves to *uniform* algorithms, i.e., the nodes are unaware of any network parameter, e.g., the network size n, nor do they have unique identifiers (the network is *anonymous*).

We consider two complexity measures of an algorithm $\mathcal{A}$. (1) The *runtime* of $\mathcal{A}$ in some graph G is the number of rounds that are executed until all nodes terminate, and (2) the *random bit complexity* of $\mathcal{A}$ is the maximum number of random bits drawn by any node during the execution of $\mathcal{A}$.

2-Hop Colorings. Throughout the paper, we study algorithms that aim to color the input graph. For a graph G, a *k-coloring* is a function $\gamma : V \to \{1, \dots k\}$ such that $\gamma(v) \neq \gamma(u)$ for any $(u, v) \in E$, where k is the number of colors. When the number of colors is not of concern, γ is called simply a coloring. In other words, the color of u is different from the color of all $v \in \Gamma(u)$. This definition naturally extends to multiple hops and in this paper, we are especially interested in the 2-hop version of coloring, where $\gamma(u) \neq \gamma(w)$ for any $u, v, w \in V$ such that $w \neq u$, $(u, v) \in E$ and $(v, w) \in E$, i.e., the color of u is different from the color of any node $w \in \Gamma^2(u)$.

The Target Function $f(n)$. A function f is called a *target function* if f is positive, strictly increasing, and continuous. Note that the properties of a target function f ensure that the *inverse target function* $f^{-1}(n)$ of $f(n)$ is well-defined. For easier readability, we denote the inverse function by $g_f(n) = f^{-1}(n)$, or $g(n)$ if f is clear from the context.

The purpose of a target function is to capture the runtime of some deterministic algorithm $\mathcal{A}$. The runtime $f_*(n)$ of $\mathcal{A}$ is positive, but not necessarily strictly increasing in the input size n, nor continuous. However, for any $\varepsilon > 0$, there is a target function f such that $f_*(n) \leq f(n) \leq f_*(n) + \varepsilon$, i.e., f "captures" f_* at all integer values $n \geq 1$.

3 Tailor-Made 2-Hop Coloring

Our technical contribution starts by presenting a 2-hop coloring algorithm, called TAILOR-2-HOP-COLORING, with a customizable runtime. Specifically, our algorithm is parametrized by a target function f and two integers $a > 2, d \geq 2$. As discussed before, we assume that $f(n)$ is between $\log^* n$ [27] and $\log \log n$ (see Section 4). Then, the algorithm finds a 2-hop coloring in $3d \cdot f(n)$ rounds in expectation and with probability $1 - n^{2-a}$.

The main difficulty is to choose how quickly random bits should be drawn, without knowledge of n. From the discussion above we know that in some round $3d \cdot f(n)$, we should have drawn at least $\Omega(\log n)$ bits. If we draw the bits too quickly, however, we might draw too many bits in the last round before the algorithm finishes. To deal with that, we design our *bit drawing function* $b(i)$ for the target function f and the integer parameters a and d as follows. Let i be some positive integer, and write $i = dp + s$ with $0 \leq s \leq d - 1$, i.e., $p = \lfloor i/d \rfloor$ and $s = i \pmod d$. The bit drawing function for i is defined as

$$b(i) = b(dp + s) = a \cdot \lceil \log g(p) \rceil^{(d-s)/d} \cdot \lceil \log g(p+1) \rceil^{s/d}.$$

We describe TAILOR-2-HOP-COLORING from the perspective of node $u \in V$ (please refer to Algorithm 1 for a pseudo-code description). The algorithm progresses in phases p, starting from

Algorithm 1: Tailor-2-Hop-Coloring(f, a, d) as executed by node u.

```
Initialization:
    $g(n) \leftarrow f^{-1}(n)$
    $x \leftarrow \varepsilon$                          ▷ the empty bit string
Phase $p = 1, 2, \dots$:
    For sub-phase $s = 0, 1, 2, \dots, d-1$:
        ▷ Round 1 of sub-phase $s$:
        Append random bits to $x$ until $|x| = b(pd + s)$
        Send $x$ to all neighbors
        ▷ Round 2 of sub-phase $s$:
        Receive $x_1, \dots, x_\delta$ from each non-terminated neighbor $v_1, \dots, v_\delta \in \Gamma(u)$
        Send list $\langle x, x_1, \dots, x_\delta \rangle$ to all neighbors
        ▷ Round 3 of sub-phase $s$:
        Receive lists $L_1, \dots, L_\delta$ from each neighbor
        if $x$ appears exactly once in every list then
            Choose color $x$ and terminate
```

phase 1, and every phase consists of d sub-phases, which in turn consist of 3 rounds each.

Node u maintains a variable x storing all random bits drawn in the course of the execution. In the first sub-phase of each phase, u appends bits to x until the length of x is $b(dp)$. In the remaining $d-1$ sub-phases $s = 1, \dots, d-1$ of phase p, by appending bits to x, the number of used random bits is increased to $b(dp + s)$. This process takes place in the first round of each sub-phase. After drawing bits in round 1 of sub-phase i, u sends its (preliminary) color x to all nodes $v \in \Gamma(u)$.

In the beginning of the second round of sub-phase i, node u receives the colors chosen by all nodes in $\Gamma(u)$. The list consisting of u's own color x and all the received colors is then sent to all neighbors of u. In the beginning of the third round of sub-phase i node u receives such a list from each neighbor. If x occurs only once in each list, then u selects color x and terminates. Otherwise, if x was used by multiple nodes, the process continues.

The idea behind Tailor-2-Hop-Coloring is as follows. In the first sub-phase of each phase, every node u draws a random color x from the set $\{1, \dots, g(p)^a\}$. Our choice of b ensures that the remaining sub-phases of phase p are used to interpolate between $g(p)^a$ and $g(p+1)^a$ if the chosen colors are not a valid 2-hop coloring. The interpolation is performed so that within each phase p, the multiplicative increase in the number of random bits used in each sub-phase is fixed. If, for instance, Tailor-2-Hop-Coloring is in the first sub-phase of some phase $p = \lceil f(n) \rceil$, then the number of bits used by u is at least $a \log n$.

Please note that in round 3 of each sub-phase, a node chooses a color only if it does not violate the 2-hop coloring constraint. Thus, the output of Tailor-2-Hop-Coloring is always a valid 2-hop coloring. The remainder of this section is dedicated to establishing the following theorem.

Theorem 1. *The runtime of* Tailor-2-Hop-Coloring *with high probability and in expectation is $\mathcal{O}(f(n))$ rounds. The random bit complexity of* Tailor-2-Hop-Coloring *with high probability and in expectation is $\mathcal{O}(h(f(n)) \cdot \log n)$ bits, where*

$$h(i) = \sqrt[d]{\frac{\lceil \log g(i+1) \rceil}{\lceil \log g(i) \rceil}} .$$

It will sometimes be convenient to express the bit drawing function in terms of h:

$$b(pd+s) = b(dp) \cdot h(p)^s\,, \qquad \text{for } 0 \le s \le d, \text{ and} \tag{1}$$
$$b(pd+s+1) = b(dp+s) \cdot h(p)\,, \qquad \text{for } 0 \le s \le d\,. \tag{2}$$

Consider the last phase p and sub-phase s for which $b(pd+s) < a \log n$. In that case, $b(pd+s+1) \geq a \log n$ bits are drawn in the next step. Thus, due to the second expression, the essence of Theorem 1 is that Tailor-2-Hop-Coloring "overshoots" the necessary $a \log n$ bits by at most a factor of $h(p)$.

Recall that the target function f can be thought of as the runtime function of any deterministic algorithm that relies on a 2-hop coloring. Before getting into the details of the analysis, let us briefly put Theorem 1 into perspective by considering the corner cases where $f \in \Theta(\log\log n)$ or $f \in \Theta(\log^* n)$. In the former case $h(f(n))$ is in $\mathcal{O}(1)$, whereas in the latter case $h(f(n))$ is in $\mathcal{O}(\sqrt[d]{n})$. Thus, we obtain the following corollary from Theorem 1.

Corollary 2. *Consider a target function f, and let R denote the random bit complexity of* Tailor-2-Hop-Coloring.

1. *If $f(n) \in \Theta(\log^* n)$, then R is $\mathcal{O}(\sqrt[d]{n} \cdot \log n) \subseteq \mathcal{O}(\sqrt[d-1]{n})$ w.h.p. and in expectation.*
2. *If $f(n) \in \Theta(\log\log n)$, then R is $\mathcal{O}(\log n)$ w.h.p. and in expectation.*

The analysis of Tailor-2-Hop-Coloring's runtime and random bit complexity are done separately. We first establish the high-probability results, beginning with the runtime.

Lemma 3. Tailor-2-Hop-Coloring *terminates after at most $\mathcal{O}(f(n))$ rounds w.h.p.*

We validate the claim by showing that all nodes terminate in phase $f(n)$ with probability $1 - n^{2-a}$. This is sufficient, since each phase consists of exactly $3d$ rounds.

Proof. To establish Lemma 3 we will show that all nodes terminate in phase $p = \lceil f(n) \rceil$ with high probability. This is sufficient to establish the claim, since every phase consists of $3d$ rounds. In our proof, for any node z, we denote by x_z the random bit string stored in z's variable x after the first sub-phase of phase p.

First, consider some node $u \in V$ that did not terminate before phase p. By the definition of the bit drawing function b, it holds that $|x_u| \geq a \log g(p) \geq a \log n$. Node u terminates in round 3 of sub-phase 1 only if $x_u \neq x_v$ for all $v \in \Gamma^2(u)$. If v terminated in some phase $r < p$, then $|x_v| < |x_u|$ since v stopped drawing random bits in phase r, and in particular the probability that $x_u = x_v$ is 0. If on the other hand v still participates in phase p, then the probability that $x_u = x_v$ is at most $1/2^{a \log n} = 1/n^a$, since $|x_u| = |x_v| \geq a \log n$ in phase p.

Now consider the event T that all nodes terminate in phase p. The opposite event $\neg T$, i.e., the event that at least one node does not terminate in phase p, occurs if there are nodes $u, v \in V$ such that $x_u = x_v$ and $u \in \Gamma^2(v)$. By applying the union bound, we get

$$\Pr[\neg T] \leq \sum_{u,v \in V} \Pr[x_u = x_v] \leq \frac{1}{n^{a-2}}\,,$$

so that T occurs with probability at least $(1 - 1/n^{a-2})$. For any constant c, we obtain that T occurs with probability $1 - n^{-c}$ by setting $a = c + 2$ in Tailor-2-Hop-Coloring. Recalling that each phase consists of $3d$ rounds, the claim follows. □

The next lemma ensures the desired high probability result for the random bit complexity, and can be shown in a similar manner. However, this time our analysis takes the exact sub-phase in which Tailor-2-Hop-Coloring terminates (w.h.p.) into account.

Lemma 4. *The random bit complexity of* Tailor-2-Hop-Coloring *is at most $h(f(n)) \cdot a \log n$ with high probability.*

Proof. We prove the statement by taking the exact sub-phase in which Tailor-2-Hop-Coloring terminates (w.h.p.) into account for our analysis. To that end, let p and s be a phase and a sub-phase, correspondingly, such that

$$(h(p))^s \cdot \log g(p) \leq \log n \leq (h(p))^{s+1} \cdot \log g(p) \,. \tag{3}$$

In sub-phase $s+1$, each non-terminated node draws at least $a(h(p))^{s+1} \cdot \log g(p) \geq a \log n$ bits and therefore, the probability that any two nodes draw the same bit string is bounded from above by $1/2^{a \log n} = 1/n^a$. Let $\neg T$ denote the event that at least one node u does not terminate in sub-phase $s+1$ of phase p. Event $\neg T$ occurs if there is at least one node in $v \in \Gamma^2(u)$ that has drawn the same bit sequence as u, and applying the union bound yields that

$$\Pr[\neg T] \leq \sum_{u,v \in V} \frac{1}{n^a} \leq \frac{1}{n^{a-2}} \ .$$

As the last step of the proof, we bound the number of bits $a \cdot \log g(p) \cdot (h(p))^{s+1}$ used in sub-phase $s+1$ of phase p. Since Equation (3) implies that $p \leq f(n)$, we obtain that with high probability the bit complexity is

$$a(h(p))^{s+1} \cdot \log g(p) \leq ah(p) \cdot \log n \leq ah(f(n)) \cdot \log n \,,$$

as desired. □

Next, we establish the results for the expected values.

Lemma 5. *The runtime of* Tailor-2-Hop-Coloring *is at most $\mathcal{O}(f(n))$ in expectation.*

Our proof of the above lemma again considers the phase in which Tailor-2-Hop-Coloring terminates. The idea is to split the summation of the expected value into two parts, namely before and including phase $f(n)$, and after phase $f(n)$. Both terms can then be bounded individually.

Proof. Let P be the random variable denoting the phase in which Tailor-2-Hop-Coloring terminates. From the definition of the expected value, for the expected runtime $\mathbf{E}[P]$ we obtain

$$\mathbf{E}[P] = \sum_{i=1}^{\infty} i \cdot \Pr[P = i] = \sum_{i=1}^{\infty} \Pr[P \geq i] \,,$$

where $\Pr[P \geq i]$ corresponds to the probability of proceeding to phase i. Furthermore, the algorithm only proceeds to phase $i+1$ in the case that there is a pair of nodes u and v within 2-hops distance that get assigned the same color in phase i. The number of bits used in sub-phase 0 of any phase p is $a\lceil \log g(p) \rceil$. Therefore, we can apply the union bound and get that

$$\Pr[P \geq i+1] \leq \sum_{u,v \in V} \frac{1}{2^{a\lceil \log g(i) \rceil}} \leq \sum_{u,v \in V} \frac{1}{2^{a \log g(i)}} \leq \frac{n^2}{(g(i))^a} \,.$$

Recalling that $g(f(n)) = n$ we get that

$$\Pr[P \geq f(n) + i] \leq \frac{n^2}{(g(f(n)+i))^c} \leq \frac{n^2}{n^a \cdot 2^i} \leq \frac{1}{2^i} \ ,$$

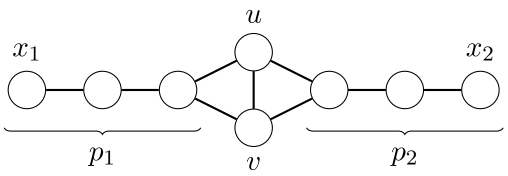

Figure 1: A (u, v)-gadget of length $i = 4$, consisting of $2i$ nodes: The two special nodes u and v, and the two paths p_1 and p_2 of length $i - 1$ with endpoints x_1 and x_2, respectively. Since the gadget is symmetric, symmetry between u and v can only be broken by their individual random coin tosses.

given that $a > 2$. We set $\varphi = f(n)$ and divide the summation of the expected runtime into the part before and after φ. It follows that

$$\begin{aligned}\mathbf{E}[P] = \sum_{i=1}^{\infty} \Pr[P \geq i] &= \sum_{i=1}^{\varphi} \Pr[P \geq i] + \sum_{i=1}^{\infty} \Pr[P \geq \varphi + i] \\ &\leq \sum_{i=1}^{\varphi} 1 + \sum_{i=1}^{\infty} \frac{1}{2^i} \leq \varphi + 1\end{aligned}$$

The claim follows since each phase consists of $3d$ rounds. □

Lemma 6. *If $f(n)$ is at least $\log^* n$, then the random bit complexity of* Tailor-2-Hop-Coloring *is $\mathcal{O}(h(f(n)) \cdot \log n)$ in expectation.*

The proof of Lemma 6, similar to that of Lemma 5, relies on carefully inspecting the round in which Tailor-2-Hop-Coloring terminates. However, due to the possibly large growth of g (which directly affects the growth of the bit drawing function), the analysis requires more attention. Instead of considering only the phase in which Tailor-2-Hop-Coloring terminates, we take the exact step in that phase into account. This yields a division of the expected value into 5 (instead of the previous 2) terms. Bounding each term individually leads to a rather lengthy proof, which is therefore deferred to the appendix. Theorem 1 is now established by combining Lemmas 3 to 6.

4 Trade-off Lower Bound

Our goal in this section is to show that the trade-off achieved by Tailor-2-Hop-Coloring's bit drawing function is asymptotically optimal. For this effort, it is sufficient to study lower bounds for the 1-hop variant of the coloring problem, since every 2-hop coloring is also a 1-hop coloring. More precisely, we are going to establish the following:

Theorem 7. *Let $\mathcal{A}$ be any randomized uniform anonymous coloring algorithm. If the expected runtime of $\mathcal{A}$ is asymptotically smaller than that of* Tailor-2-Hop-Coloring*, then $\mathcal{A}$'s expected random bit complexity is asymptotically larger than that of* Tailor-2-Hop-Coloring.

The rough idea is that in order to break symmetry, the nodes have to draw random bits according to some (possibly randomized) scheme. We distinguish two cases: In the first case, $\mathcal{A}$ may try to break symmetry quickly by using many random bits. We show that then, the expected random bit complexity of $\mathcal{A}$ needs to be large. For the second case, where $\mathcal{A}$ prevents this behavior, we show that the expected runtime of $\mathcal{A}$ is asymptotically as large as that of Tailor-2-Hop-Coloring.

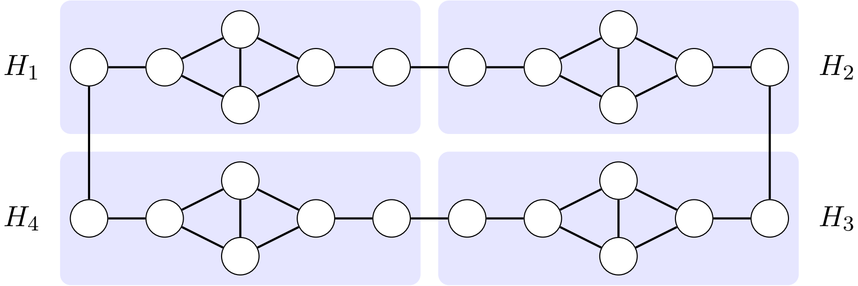

Figure 2: The graph $G(4,3)$, consisting of 4 (u,v)-gadgets H_1, H_2, H_3, and H_4, each of length 3.

Our proof relies on a graph construction consisting of several so-called (u,v)*-gadgets*. A (u,v)-gadget of length i (depicted in Figure 1) consists of $2i$ nodes, namely two paths p_1, p_2 of length $i-1$ and two special nodes u and v, connected by an edge. Furthermore, nodes u and v are connected to one endpoint of both p_1 and p_2. The other endpoints of p_1 and p_2 are referred to as x_1 and x_2, respectively. We obtain the graph $G(m,i)$ utilized in our lower bound proofs by connecting m (u,v)-gadgets of length i in a ring-like topology. This is done by simply chaining the m gadgets together by their endpoint nodes x_1 and x_2—please refer to Figure 2 for an illustration. We note that $G(m,i)$ consists of $2im$ nodes.

Consider, for example, the graph $G = G(2^k, 3)$ for some arbitrarily large k. Since the graph G is symmetric from the perspective of each (u,v)-pair in any of the gadgets, every such pair can break symmetry only by their individual random coin tosses. Assume now for the sake of contradiction, that there is a coloring algorithm $\mathcal{A}$ with an expected bit complexity $\beta \in o(\log n)$. In that case, with arbitrarily large probability, at least one of the (u,v)-pairs tosses exactly the same sequence of random bits. This contradicts the claim that $\beta \in o(\log n)$, and thus we obtain the following result from our graph construction.

Corollary 8. *Any coloring algorithm must have an expected random bit complexity in* $\Omega(\log n)$.

In our effort to prove the trade-off lower bound we would like to have a better grip than that on the random coin tosses made by the nodes. Specifically, for any algorithm $\mathcal{A}$ and (u,v)-gadget H, we denote by $B_{\mathcal{A}}(i,H)$ the random variable taking on the maximum number of random bits drawn by nodes u and v in H until and including round i. Whenever $\mathcal{A}$ is clear from the context, we omit it in the notation and write $B(i,H)$ instead. The following insight about those random variables in the graph $G(m,i)$ will be helpful in our proof of Theorem 7.

Lemma 9. *Consider any algorithm $\mathcal{A}$, and let H be a single (u,v)-gadget of length i. Let $m \geq 2$ be an integer, and denote by $H_1, \ldots, H_m$ the m (u,v)-gadgets in the graph $G(m,i)$. For any $j \leq i$, all the random variables $B(j,H_k)$, obtained from an execution of $\mathcal{A}$ in $G(m,i)$, are independent and distributed like $B(j,H)$.*

Proof. Observe that in every H_k, the nodes u and v are i hops away from each endpoint, and that $j \leq i$. Consider any (u,v)-gadget H_k in $G(m,i)$. Since $j \leq i$, the execution of $\mathcal{A}$ until round j for nodes u and v cannot depend on any node w from a different gadget $H_l \neq H_k$. It thus holds that all $B(j,H_k)$ are independent and distributed like $B(j,H)$. □

As noted before, the proof for Theorem 7 is divided into two parts, depending on how $\mathcal{A}$ chooses to draw random bits (in expectation). For that, based on the bit drawing function b used by Tailor-2-Hop-Coloring (for fixed parameters f, a, and d), we introduce a threshold for the number of random bits drawn by some algorithm as follows.

Definition (Drawing few/a lot of random bits)**.** Fix a bit drawing function b parametrized by a target function f, and two constants $a > 2$ and $d \geq 2$. Let H be a (u, v)-gadget of length i, and let $\mathcal{A}$ be a randomized algorithm. We say that $\mathcal{A}$ draws *a lot of random bits* if

$$\exists i_0 \forall i \geq i_0 \quad \mathbf{E}[B(i, H)] \geq b(3i)/4\,.$$

If $\mathcal{A}$ does not draw a lot of random bits, then we say that $\mathcal{A}$ *draws few random bits.*

Due to Lemma 9, properties of single (u, v)-gadgets can be lifted to instances of $G(m, i)$. One such property we will use is encapsulated in the following technical lemma:

Lemma 10. *Let $\mathcal{A}$ be any coloring algorithm. If $\mathcal{A}$ draws a lot of random bits, then*

$$\exists i_0 \forall i \geq i_0 \exists j \leq i \quad \mathbf{E}[B(j, H)] \leq b(i)/4, \textit{ and } \mathbf{E}[B(j+1, H)] \geq b(i+2)/4\,,$$

where H is a (u, v)-gadget of length i.

Proof. Assume the statement is false. By induction,

$$\begin{aligned}
\mathbf{E}[B(0, H)] &= 0 = b(0) < b(i)/4 \\
\mathbf{E}[B(1, H)] &< b(i+2)/4 \\
\mathbf{E}[B(2, H)] &< b(i+4)/4 \\
&\vdots \\
\mathbf{E}[B(i, H)] &< b(i+2i)/4
\end{aligned}$$

This contradicts the premise that $\mathcal{A}$ draws a lot of random bits, i.e., that $\mathbf{E}[B(i, H)] \geq b(3i)/4$ for the (u, v)-gadget H of length i. □

We now have the essential tools to prove Theorem 7, and first consider the case where $\mathcal{A}$ draws a lot of random bits. In that case, for sure, the runtime of $\mathcal{A}$ can be better than that of Tailor-2-Hop-Coloring. Imagine for example a process that draws infinitely many random bits in the first round—one would immediately obtain a 2-hop coloring within a single round with probability 1, albeit at the cost of an infinite random bit complexity. The essential insight of the following Lemma 11 is that no matter how "smartly" one tries to draw a lot of random bits in hopes to get a better runtime, the expected bit complexity will be asymptotically worse than that of Tailor-2-Hop-Coloring.

Lemma 11. *Let $\mathcal{A}$ be any coloring algorithm. If $\mathcal{A}$ draws a lot of random bits, then $\mathcal{A}$'s expected random bit complexity is $\Omega(h(f(n))^2 \cdot \log n)$.*

In our proof, we carefully choose a gadget graph of a certain size. We then utilize Lemma 9 to "copy" the property obtained from Lemma 10 for a single (u, v)-gadget to all gadgets in the graph. Applying Markov's inequality twice, the choice of the gadget graph then allows us to derive the desired lower bound.

Proof. Consider the i_0 promised by the fact that $\mathcal{A}$ draws a lot of random bits. Let k be such that $i = dk > i_0$, and let j be the integer obtained from Lemma 10 for this i. By Markov's inequality it holds that $\Pr[B(j, H) \leq b(i)/8] \leq 1/2$, where H denotes the (u, v)-gadget of length i. Now consider the graph $G = G(m, i)$, where $m = 2^{b(i)}/(2i)$, and denote by $H_1, \ldots, H_m$ the m copies of H in G. Then, G consists of $n = 2^{b(i)}$ many nodes, i.e., $\log n = b(i)$.

Let N denote the random variable taking on the number of gadgets H_k for which $B(j, H_k) \leq \log(n)/8$. Due to Lemma 9, we obtain that all $B(j, H_k)$ are independent and distributed like $B(j, H)$. Therefore, $\mathbf{E}[N]$ is at least $m/2$, and once again applying Markov's inequality, we get $\Pr[N \geq m/4] \leq 1/2$. The probability p to terminate in round j can now be bounded as

$$p \leq \frac{1}{2} + \frac{1}{2}\left(1 - \frac{1}{2^{\log(n)/8}}\right)^{m/4} \leq \frac{1}{2} + \frac{1}{2}\left(1 - \frac{1}{\sqrt[8]{n}}\right)^{n/(8i)} .$$

Note that since $b(di) = \log g(i)$ and $b(d(i+1)) = \log g(i+1)$, and $g(n) \geq 2^n$, it must hold that $b(d(i+1)) \geq b(di) + 1$. Therefore,

$$b(i) \geq \sum_{\alpha=1}^{\lfloor i/d \rfloor} \underbrace{b(\alpha d) - b((\alpha-1)d)}_{\geq 1} \geq \left\lfloor \frac{i}{d} \right\rfloor \geq \frac{i}{2d} .$$

We obtain that $i \leq 2db(i) \leq 2d \log n$. This means for p that

$$p \leq \frac{1}{2} + \frac{1}{2}\left(1 - \frac{1}{\sqrt[8]{n}}\right)^{n/(16d\log n)} \leq 2/3 ,$$

for large i. The probability of entering round $j+1$ is thus at least $1/3$. Recalling from Lemma 10 that $\mathbf{E}[B(j+1, H)] \geq b(i+2)/4$, we obtain that the expected bit complexity is at least $\frac{1}{3} \cdot \frac{1}{4} b(i+2)$. Now, since $f(n) = f(2^{b(i)}) = f(2^{b(dk)}) = f(2^{a \log g(k)}) \geq k$, we obtain that $df(n) \geq i$. We can therefore bound the expected bit complexity from below by

$$\frac{1}{12} b(i+2) \geq \frac{1}{12} b(df(n)+2) = \frac{1}{12} h(f(n))^2 \cdot \log g(f(n)) \in \Omega(h(f(n))^2 \cdot \log n) ,$$

as desired. □

With the next lemma we consider the opposite case where $\mathcal{A}$ draws only few random bits.

Lemma 12. *Let $\mathcal{A}$ be any coloring algorithm. If $\mathcal{A}$ draws few random bits, then the expected runtime of $\mathcal{A}$ is $\Omega(df(n))$.*

Our proof follows similar lines as that for Lemma 11. The key difference is how the size of the gadget graph is chosen.

Proof. Fix some i_0, and let $i \geq i_0$ be the constant guaranteed by the fact that $\mathcal{A}$ draws few random bits, i.e., $\mathbf{E}[B(i, H)] < b(3i)/4$. Therefore, due to the Markov inequality, we get that $\Pr[B(i, H) < b(3i)/2] > 1/2$. Now consider the graph G consisting of $m = 2^{b(3i)}/(2i)$ many copies $H_1, \ldots, H_m$ of H, connected in a ring topology by their endpoints. Then, G consists of $n = 2^{b(3i)}$ many nodes, i.e., $\log n = b(3i)$.

Again, due to Lemma 9, all $B(j, H_k)$ are independent and distributed like $B(j, H)$. Let N denote the random variable taking on the number of gadgets H_k in G for which $B(j, H_k) < \log(n)/2$. Then it holds that $\mathbf{E}[N] > m/2$, and with Markov's inequality we get that $\Pr[N > m/4] > 1/2$. Let p be the probability that some gadget H_k does not terminate in round i. We get that

$$p \geq \frac{1}{2}\left(1 - \left(1/\sqrt{n}\right)^{m/4}\right) \geq \frac{1}{3} ,$$

for sufficiently large i_0. Now, observe that $f(n) = f(2^{b(3i)}) \leq f(2^{\log g(\lceil 3i/d \rceil)}) = f(g(\lceil 3i/d \rceil)) \leq 3i/d + 1$. We can thus bound the expected runtime of $\mathcal{A}$ from below by $(i+1)/3 \geq (d-1)f(n)/9$, which is in $\Omega(df(n))$ as desired. □

We obtain the desired optimality of Tailor-2-Hop-Coloring from Lemma 11 only if $h(f(n))^2 \in \omega(h(f(n)))$. In the case where $f \in \mathcal{O}(\log\log n)$, however, $h(f(n))$ is bounded from above by a constant. It may thus appear that such an f is not covered by our lemmas.

To see that this is not an issue, observe that the constant 3 in the definition of drawing a lot of random bits was chosen arbitrarily. In other words, when $h(f(n))$ is bounded by some constant ρ, one may replace 3 in the above definition with $\rho{+}3$. This way, we obtain that the coloring algorithm $\mathcal{A}$ draws "ρ-few" random bits. We can now apply the same reasoning as in the proof of Lemma 12 to obtain that the runtime of $\mathcal{A}$ is in the same order as that of Tailor-2-Hop-Coloring. This concludes our effort to establish Theorem 7.

References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567 – 583, 1986.

[2] Alexander E. Andreev, Andrea E. F. Clementi, José D. P. Rolim, and Luca Trevisan. Weak random sources, hitting sets, and BPP simulations. *SIAM J. Comput.*, 28:2103–2116, 1999.

[3] Dana Angluin. Local and global properties in networks of processors (extended abstract). In *STOC*, 1980.

[4] James Aspnes and Orli Waarts. Randomized consensus in expected o(n $\log^2$ n) operations per processor. *SIAM J. Comput.*, 25:1024–1044, 1996.

[5] Hagit Attiya and Keren Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55, 2008.

[6] Hagit Attiya and Faith Ellen. *Impossibility Results for Distributed Computing.* Morgan & Claypool Publishers, 2014.

[7] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics.* John Wiley & Sons, 2004.

[8] Leonid Barenboim and Michael Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In *STOC*, 2009.

[9] Leonid Barenboim and Michael Elkin. Deterministic distributed vertex coloring in polylogarithmic time. *J. ACM*, 58:23, 2011.

[10] Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments.* Morgan & Claypool Publishers, 2013.

[11] Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (delta+1)-coloring in linear (in delta) time. *SIAM J. Comput.*, 43:72–95, 2014.

[12] Jean R. S. Blair and Fredrik Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444:28–39, 2012.

[13] Shlomi Dolev. *Self-Stabilization.* Mit Press, 2000.

[14] Shlomi Dolev and Nir Tzachar. Randomization adaptive self-stabilization. *Acta Inf.*, 47:313–323, 2010.

[15] Yuval Emek, Christoph Pfister, Jochen Seidel, and Roger Wattenhofer. Anonymous networks: randomization = 2-hop coloring. In *PODC*, 2014.

[16] Pierre Fraigniaud, Mika Göös, Amos Korman, Merav Parter, and David Peleg. Randomized distributed decision. *Distributed Computing*, 27(6):419–434, 2014.

[17] Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60:35, 2013.

[18] John Gill. Computational complexity of probabilistic turing machines. *SIAM J. Comput.*, 6:675–695, 1977.

[19] Andrew V. Goldberg, Serge A. Plotkin, and Gregory E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM J. Discrete Math.*, 1:434–446, 1988.

[20] Mika Göös and Jukka Suomela. Locally checkable proofs. In *PODC*, 2011.

[21] Maria Gradinariu and Sébastien Tixeuil. Self-stabilizing vertex coloration and arbitrary graphs. In *OPODIS*, 2000.

[22] Roger Wattenhofer Johannes Schneider. Trading bit, message, and time complexity of distributed algorithms. In *DISC*, 2011.

[23] Richard Karp, Nicholas Pippenger, and Michael Sipser. A time-randomness tradeoff. In *AMS Conference on Probabilistic Computational Complexity*, 1985.

[24] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. In *PODC*, 2005.

[25] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *SPAA*, 2009.

[26] Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17:215–217, 1983.

[27] Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 1992.

[28] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *STOC*, 1985.

[29] Michael Luby and Avi Wigderson. Pairwise independence and derandomization. *Foundations and Trends in Theoretical Computer Science*, 1, 2005.

[30] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[31] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24:1259–1277, 1995.

[32] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.*, 58:148–173, 1999.

[33] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[34] Johannes Schneider, Michael Elkin, and Roger Wattenhofer. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theor. Comput. Sci.*, 509:40–50, 2013.

[35] Johannes Schneider and Roger Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *PODC*, 2008.

[36] Alex Scott, Peter Jeavons, and Lei Xu. Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In *PODC*, 2013.

[37] Sandeep Kumar Shukla, Daniel J. Rosenkrantz, and S. S. Ravi. Developing self-stabilizing coloring algorithms via systematic randomization. In *Proceedings of the International Workshop on Parallel Processing*, 1994.

[38] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7:1–336, 2012.

Appendix

A Omitted Proof of Lemma 6

We now present the omitted proof for Lemma 6. The proof begins by splitting the expected value into 5 terms. In the subsequent lengthy conclusion, each term is bounded individually. The brave reader has been warned.

Proof (of Lemma 6)**.** Let P again be the random variable that denotes the phase in which Tailor-2-Hop-Coloring terminates. In addition, let S be the random variable that denotes the sub-phase of phase P in which Tailor-2-Hop-Coloring terminates, given that Tailor-2-Hop-Coloring terminates in phase P. Let $B = b(dP + S)$ be the random variable denoting the number of random bits used at any node by Tailor-2-Hop-Coloring.

Let φ be the phase for which $g(\varphi) \leq n < g(\varphi + 1)$. The expected value of B can be expressed as

$$\begin{aligned}
\mathbf{E}[B] = \;& \mathbf{E}[b(dP+S)] \\
= \;& \underbrace{\mathbf{E}[b(dP+S) \mid P < \varphi]\Pr[P < \varphi]}_{X} && (4) \\
&+ \underbrace{\mathbf{E}[b(dP+S) \mid P = \varphi]\Pr[P = \varphi]}_{Y_{1,1}} && (5) \\
&+ \underbrace{\mathbf{E}[b(dP+S) \mid P = \varphi+1 \wedge S = 0]\Pr[P = \varphi+1 \wedge S = 0]}_{Y_{1,2}} && (6) \\
&+ \underbrace{\mathbf{E}[b(dP+S) \mid P = \varphi+1 \wedge S \geq 1]\Pr[P = \varphi+1 \wedge S \geq 1]}_{Y_{2}} && (7) \\
&+ \underbrace{\mathbf{E}[b(dP+S) \mid P > \varphi+1]\Pr[P > \varphi+1]}_{Z} && (8)
\end{aligned}$$

The goal is now to bound each term of the sum individually.

$$\begin{aligned}
X &= \sum_{p=0}^{\varphi-1} \Pr[P = p] \sum_{s=0}^{d-1} \Pr[S = s \mid P = p] b(dp + s) \\
&\leq b(d\varphi) \cdot \sum_{p=0}^{\varphi-1} \Pr[P = p] \sum_{s=0}^{d-1} \Pr[S = s \mid P = p] \\
&= b(d\varphi) \cdot \sum_{p=0}^{\varphi-1} \Pr[P = p] \\
&\leq b(d\varphi)\,,
\end{aligned}$$

since the events in each above sum are disjoint.

$$
\begin{aligned}
Z &= \sum_{p=\varphi+2}^{\infty} \Pr[P=p]\left(\Pr[S=0 \mid P=p]b(dp) + \sum_{s=1}^{d-1}\Pr[S=s \mid P=p]b(dp+s)\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\Pr[P=p]b(dp) + \sum_{s=1}^{d-1}\Pr[S=s \mid P=p]\Pr[P=p]b(dp+s)\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\Pr[P\ge p]b(dp) + \sum_{s=1}^{d-1}\Pr[S\ge s \mid P=p]\Pr[P\ge p]b(dp+s)\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{n^2\cdot b(dp)}{2^{b(d(p-1))}} + \sum_{s=1}^{d-1}\frac{n^2\cdot b(dp+s)}{2^{b(dp+s-1)}}\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{n^2\cdot b(dp)}{2^{b(d(p-1))}} + \sum_{s=1}^{d-1}\frac{n^2\cdot b(d(p+1))}{2^{b(dp)}}\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{n^2\cdot b(dp)}{2^{b(d(p-1))}} + \frac{dn^2\cdot b(d(p+1))}{2^{b(dp)}}\right)\\
&= \sum_{p=\varphi+2}^{\infty}\left(\frac{n^2\cdot a\log g(p)}{2^{a\log g(p-1)}} + \frac{dn^2\cdot a\log g(p+1)}{2^{a\log g(p)}}\right)\\
&= \sum_{p=\varphi+2}^{\infty}\left(\frac{n^2\cdot a\log g(p)}{g(p-1)^a} + \frac{dn^2\cdot a\log g(p+1)}{g(p)^a}\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{an^2}{g(p-1)^{a-1}} + \frac{adn^2}{g(p)^{a-1}}\right), \text{ since } 2^{g(p)} \ge g(p+1), \text{ i.e., } g \text{ is at most the power-tower}\\
&= \sum_{p=\varphi+2}^{\infty}\left(\frac{an^2}{g(p-1)\cdot g(p-1)^{a-2}} + \frac{adn^2}{g(p)\cdot g(p)^{a-2}}\right)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{an^2}{g(p-1)n^{a-2}} + \frac{adn^2}{g(p)n^{a-2}}\right), \text{ since } p-1\ge\varphi+1\ge f(n)\\
&\le \sum_{p=\varphi+2}^{\infty}\left(\frac{1}{g(p-1)} + \frac{1}{g(p)}\right) = \sum_{p=0}^{\infty}\left(\frac{1}{g(p+\varphi+1)} + \frac{1}{g(p+\varphi+2)}\right)\\
&\le 2\sum_{p=0}^{\infty}\frac{1}{2^p} \le 4\ .
\end{aligned}
$$

Finally, let s^* be an integer such that $b(d\varphi+s^*)\le a\log n < b(d\varphi+s^*+1)$. Then, we get that

$$
\begin{aligned}
Y_{1,1} &= \Pr[P=\varphi]\sum_{s=0}^{d-1}\Pr[S=s \mid P=\varphi]b(d\varphi+s)\\
&= \Pr[P=\varphi]\sum_{s=0}^{s^*+1}\Pr[S=s \mid P=\varphi]b(d\varphi+s) + \sum_{s=s^*+2}^{d-1}\Pr[S=s \mid P=\varphi]b(d\varphi+s)\\
&\le db(d\varphi+s^*+1) + \sum_{s=s^*+2}^{d}\Pr[S\ge s \mid P=\varphi]b(d\varphi+s)
\end{aligned}
$$

Applying the union bound yields

$$\begin{aligned}
Y_{1,1} + Y_{1,2} &\leq db(d\varphi + s^* + 1) + \sum_{s=s^*+2}^{d-1} \frac{n^2 \cdot b(d\varphi + s)}{2^{b(d\varphi+s-1)}} + \frac{n^2 \cdot b(d(\varphi+1))}{2^{b(d\varphi+d-1)}} \\
&= db(d\varphi + s^* + 1) + \sum_{s=s^*+2}^{d} \frac{n^2 \cdot b(d\varphi + s)}{2^{b(d\varphi+s-1)}}, \text{ since } b(d\varphi + d) = b(d(\varphi+1)) \\
&= db(d\varphi + s^* + 1) + a \sum_{s=s^*+2}^{d} \frac{n^2}{2^{\frac{a-1}{a} b(d\varphi+s-1)}} \cdot \frac{\log g(\varphi)h(\varphi)^s}{2^{\log g(\varphi)h(\varphi)^{s-1}}} \\
&\leq db(d\varphi + s^* + 1) + a \sum_{s=s^*+2}^{d} \frac{n^2}{2^{(a-1)\log n}} \cdot \frac{\log g(\varphi)h(\varphi)^s}{2^{\log g(\varphi)h(\varphi)^{s-1}}} \\
&= db(d\varphi + s^* + 1) + a \sum_{s=s^*+2}^{d} \underbrace{\frac{n^2}{n^{(a-1)}}}_{\leq 1 \text{ if a} > 2} \cdot \frac{\log g(\varphi)h(\varphi)^s}{2^{\log g(\varphi)h(\varphi)^{s-1}}} \\
&\leq db(d\varphi + s^* + 1) + a \sum_{s=s^*+2}^{d} \frac{\log g(\varphi+1)}{2^{\log g(\varphi)}} \\
&\leq db(d\varphi + s^* + 1) + a \sum_{s=s^*+2}^{d} 1 \text{ , again, since } 2^{g(p)} \geq g(p+1) \\
&\leq db(d\varphi + s^* + 1) + ad \in \mathcal{O}(b(d\varphi + s^* + 1)) \subseteq \mathcal{O}(b(d\varphi + s^*)h(\varphi)) \subseteq \mathcal{O}(\log n \cdot h(f(n))) \ ,
\end{aligned}$$

$$\begin{aligned}
Y_2 &= \Pr[P = \varphi + 1] \sum_{s=1}^{d-1} \Pr[S = s \mid P = \varphi + 1] b(\varphi + 1, s) \\
&\leq \Pr[P \geq \varphi + 1] \sum_{s=1}^{d-1} \Pr[S \geq s \mid P = \varphi + 1] b(\varphi + 1, s) \\
&\leq \Pr[P \geq \varphi + 1] \sum_{s=1}^{d-1} \Pr[S \geq s \mid P = \varphi + 1] b(\varphi + 2, 0) \\
&\leq d \frac{n^2 \cdot b(\varphi + 2, 0)}{2^{b(\varphi+1,s-1)}} \leq d \frac{n^2 \cdot b(\varphi + 2,)}{2^{b(\varphi+1,0)}} \\
&\leq \frac{n^2}{2^{(a-1)\log g(\varphi+1)}} \cdot \frac{\log g(\varphi+2)}{2^{\log g(\varphi+1)}} \\
&\leq \frac{n^2}{g(\varphi+1)^{a-1}} \leq \frac{1}{n^{a-3}} \leq 1,
\end{aligned}$$

for $a \geq 3$. Putting things back together, we get that

$$\begin{aligned}
\mathbf{E}[B] &= X + Y_{1,1} + Y_{1,2} + Y_2 + Z \\
&\in b(\varphi, 0) + \mathcal{O}(h(f(n)) \cdot \log n) + 1 + 4 \\
&\subseteq a \log n + \mathcal{O}(h(f(n)) \cdot \log n) \\
&\subseteq \mathcal{O}(h(f(n)) \cdot \log n) \, .
\end{aligned}$$

□